IdentiFusion: A Multimodal Approach to Facial Attribute Recognition using FaceBERT and LightenedMOON

Efe Tarhan *EEE B.Sc, Bilkent University,* 22002840
Ege Yüceel *EEE B.Sc, Bilkent University,* 22003324
M.Berk Alemdar *EEE B.Sc, Bilkent University,* 22003066

***Abstract*—We present a novel multimodal system that synergistically combines Natural Language Processing (NLP) and Computer Vision to match textual face descriptions with corresponding facial images. Central to our approach is FaceBERT, an NLP model we proposed and fine-tuned specifically for extracting facial features from the text. FaceBERT is trained on SynthFace-3K, a synthetic dataset we developed to map input sequences to 40 human facial attributes. For Computer Vision, we utilize the LightenedMOON model, trained on the CelebA dataset, for facial attribute recognition from images. Our system integrates a comparator to predict the face that aligns with the textual face description, effectively merging the outputs of the NLP and Computer Vision models. Our unique contribution lies in the proposal of FaceBERT and the development of the SynthFace-3K dataset, enabling the effective extraction of facial features from text descriptions. The development of a comparator for textual and visual facial descriptions also presents advancement in multimodal learning. The report provides a comprehensive overview of the design of the multimodal system, its performance, the development and characteristics of the SynthFace-3k dataset. This work is exemplary in demonstrating the power of integrating NLP and Computer Vision techniques and has potential application in areas like image retrieval, matching, and recommendation systems to aid in law enforcement and forensic investigations.**

***Index Terms*—Statistical Natural Language Processing, Computer Vision, Facial Attribute Recognition, Feature Extraction, LightenedMOON, BERT, SynthFace-3k,**

I. Introduction

THE intersection of Natural Language Processing (NLP) and Computer Vision represents a burgeoning field of study in artificial intelligence, with the potential to create systems that can understand and interpret both textual and visual data, including image captioning [1], visual question answering [2], text-to-image synthesis [3], content-based image retrieval [4], and automatic surveillance [5]. This project is situated within this exciting context, aiming to bridge the gap between these two domains by developing a system capable of comparing textual descriptions and visual representations of human faces.

The task of understanding and generating descriptions of human faces has been a topic of interest in both NLP and Computer Vision. In NLP, recent advancements in transformer-based models, such as BERT [6], XLNet [7], and ELECTRA [8] have shown remarkable capabilities in understanding the semantics and context of language. In Computer Vision, deep learning models like MOON [9] have demonstrated impressive performance in facial attribute recognition. However, the integration of these two fields to create a cohesive system that can understand and compare facial descriptions in both text and images remains a challenging task. The purpose of this work is to explore this integration, leveraging the strengths of both NLP and Computer Vision to create a system capable of predicting the face that matches a given textual face description. This is an important question to tackle as it has numerous practical, real-life applications, ranging from image retrieval and recommendation systems to aiding in law enforcement and forensic investigations.

Our methodology involves fine-tuning a pre-trained BERT model on a synthetic dataset that we have created named SynthFace-3K, which maps input sequences to 40 human facial attributes. The output of the model is a 40-dimensional vector, each dimension corresponding to one facial attribute. On the visual side, we employ the Mixed Objective Optimization (LightenedMOON) model trained on the Celeb-A dataset to extract facial attributes from images. A comparator is then used to compare the outputs of the NLP and Computer Vision models and make predictions. The main contributions of this paper can be summarized as, (1) integration of NLP and Computer Vision with multimodal learning for facial attribute recognition by effectively integrating BERT and LightenedMOON models, (2) development of a comparator that could potentially be used in other tasks involving comparing textual and visual data, (3) development of a synthetically created dataset, which could potentially mitigate some of the biases present in real-world data.

We expect that our approach will demonstrate the feasibility and effectiveness of integrating NLP and Computer Vision for the task of comparing textual and visual facial descriptions. Moreover, we anticipate that our work will contribute to a broader understanding of how these two domains can be effectively combined.

The remainder of this report is organized as follows: Section 2 provides a detailed elaboration of the related work with respect to facial attribute classification, multi-label learning, and feature extraction. Section 3 provides a detailed description of our methodology, including the specifications of the datasets Celeb-A, and our synthetically created dataset, SynthFace-3K, fine-tuning of the FaceBERT model, the training of the LightenedMOON model, and the design of the comparator. Section 4 presents the results of our experiments. Finally,

Section 5 discusses the implications of our findings, and potential applications of our system.

II. Related Work

The intersection of Natural Language Processing (NLP) and Computer Vision has been a topic of interest in recent years, with several works exploring different aspects of this intersection. Our project, which involves mapping textual descriptions to facial attributes and comparing them with visual representations, builds upon several lines of research in these fields. Taking these into account, this section will outline the state-of-the art approaches for feature extraction from text input sequence, and facial attribute classification, and comparing them to the models that were employed in this paper.

A. Facial Attribute Classification

Facial Attribute Classification (FAC) has gained significant interest in computer vision, due to applications in image retrieval [10], face recognition [11], [12], face verification [13], person re-identification [14], and recommendation systems [15]. The grounds of facial attribute classification were first laid out by Kumar et. al. [11], where the classifiers depended heavily on face alignment with respect to a frontal template, where each attribute used AdaBoost-learnt combinations from manually picked regions of the face. Hence, different features were learnt by the model for each specific attribute, and a single RBF-SVM per attribute was trained independently. However, this system was inefficient in feature extraction and classification, taking into account the high dimensional varying length features. Considering the superior performance of Convolutional Neural Networks (CNNs), the current state-of-the-art methods employ CNNs categorically in two ways: (1) single-learning based [16], [17], [18], and (2) multi-label learning based methods [19], [20], [21].

Facial attribute classification (FAC) methods based on single-label learning typically employ CNNs to extract features from facial images, which are then classified as facial attributes with the use of a Support Vector Machine (SVM) classifier. However, these methods have a limitation in that they predict each attribute independently, thereby overlooking the potential correlations between different attributes.

On the other hand, FAC methods based on multi-label learning, which are capable of predicting multiple attributes at once, address this limitation. They can extract common features from the lower layers of the CNN, and then employ attribute-specific classifiers on the CNN's upper layers. This approach allows for the simultaneous prediction of multiple attributes, taking into account the relationship between attributes. In that regard, the model used in this paper, namely LightenedMOON, is a multi-task, multi-label learning model that has a deep convolutional neural network (DCNN) architecture; where a loss function that mixes multiple task objectives (mixed objective optimization) with domain readaptive re-weighting propogation is utilized. This allows for the model to optimize for all tasks jointly, which addresses the multi-label imbalance problem, i.e. appropriate weight assignment for each different attribute based on the performance of the model on that attribute.

As of the writing of this report, the most accurate state-of-the-art model is Debiasing Alternate Networks (DebiAN) [22], which consists of two networks: a Discoverer and a Classifier. This model carries out a process of two steps, bias discovery and bias mitigation. Unlike some previous methods, DebiAN does not treat bias identification and bias mitigation as two isolated steps, but rather carries them out simultaneously through an alternate training scheme, where the Discoverer and Classifier are updated in an interleaving fashion. DebiAN has shown to surpass any other model in achieving gender bias mitigation results on CelebA and bFFHQ datasets. Although the model achieves better accuracy than the preferred LightenedMOON model, the DebiAN system is not specific to this task and is not as equipped to handle imbalanced training data. Additionally, the efficient and less computationally intensive nature of the MOON model makes it a more viable model for practical use. Furthermore, since the synthetic dataset that we have created is balanced and free of biases, bias mitigation scheme of DebiAN model becomes less salient.

B. Feature Extraction

Feature extraction in Natural Language Processing (NLP) is a crucial step that involves transforming raw text data into a set of features that the machine learning algorithm can understand and process. Many aspects of the text can be extracted from these features, such as the presence of specific words, the frequency of certain phrases, syntactic patterns, semantic roles. The goal of this task is to capture as much relevant information from the text as possible, while reducing the complexity and dimensionality of the data.

In recent years, the state-of-the-art in feature extraction for NLP has been revolutionized by transformer-based models, such as BERT [6], GPT-3 [23], RoBERTa [24], XLNet [7], ELECTRA [8]. These models are pre-trained on a large corpora of text and learn to generate contextualized word embeddings, trying to understand the meaning of a word in the context of its respective sentence. This represents a significant advance over traditional methods of feature extraction, which often rely on static word embeddings that do not capture context. In the context of feature extraction, GPT-3 model uses a left-to-right (unidirectional) training approach, i.e. it predicts the next word in a sentence given the previous words. For the same task, XLNet, proposes a generalized autoregressive model that captures the bidirectional context by maximizing the expected likelihood over all permutations of the input sequence, while ELECTRA, unlike BERT, is trained to distinguish between "real" and "fake" tokens in a given sentence, instead of predicting what the masked tokens are.

As of the writing of this report, the most accurate state-of-the-art model is XY-LENT [25], proposed by Microsoft, which extends the transformer architecture with a novel cross-attention mechanism and a length-adaptive relative position encoding. The cross-attention mechanism allows the model to attend to different parts of the input sequence, which can be particularly useful for tasks that require understanding

the relationship between different elements in the input. The length-adaptive relative position encoding enables the model to handle longer sequences more effectively.

The model used in this paper for the feature extraction process, BERT, has achieved state-of-the-art results on various NLP benchmarks, and differentiates from other state-of-the-art models by its bidirectional training mechanism. While XY-LENT'S cross-attention mechanism and length-adaptive relative position encoding can potentially improve performance on certain tasks, they also make the model more complex and potentially more computationally expensive. Taking this into consideration, and also considering extensively proven performance of BERT for this task, BERT appears to be the more viable option in the context of this report.

III. Methods

This section delves into the intricate details of the datasets utilized in our study, namely the CelebA dataset and our synthetically generated dataset, SynthFace-3K, elucidating their unique features and specifications. It provides a comprehensive examination of the CelebA dataset's inherent bias, and sheds light on the construction and characteristics of our novel SynthFace-3K dataset, designed specifically to map input sequences to 40 human facial attributes. This section further outlines the methodology employed in training a robust facial attribute classification model, underscoring the nuances of model architecture, and training parameters. It delves into the process of fine-tuning the pre-trained BERT model, which we have aptly named FaceBERT, elucidating the adjustments made to fit the model to our specific task. Moreover, it provides an exploration of the design and functioning of the comparator, a key component of our system that integrates the outputs of the NLP and Computer Vision models to predict the face that aligns with the textual face description.

A. CelebA Dataset

The CelebA dataset [26] is a large-scale face attributes dataset that has been widely used in the machine learning community for various facial recognition tasks. It contains 202599 face images with 10177 identities, each with 40 attribute annotations derived from 5 landmark locations. The images in this dataset cover large pose variations and background clutter, making it a challenging and comprehensive dataset for this task, namely facial attribute recognition.

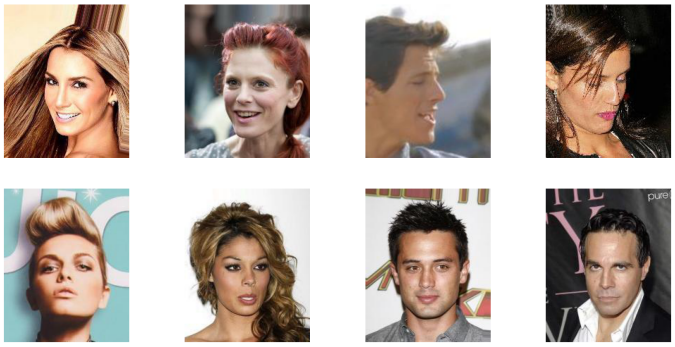

Fig. 1. First 6 samples of the face images of celebrities present in the CelebA dataset. As can be seen, the face images have varying identities, pose angles, and backgrounds, which render this a challenging dataset.

Each image in the CelebA dataset is annotated with 40 binary attributes, such as "Wearing Lipstick", "Big Nose", "No Beard", and "Narrow Eyes", among others. These attributes cover a wide range of facial characteristics and therefore can be used for multi-label classification. The dataset also provides bounding box annotations for face detection and landmark annotations for facial part localization. The large number of images and the diversity of attributes make CelebA a valuable benchmark for facial analysis tasks.

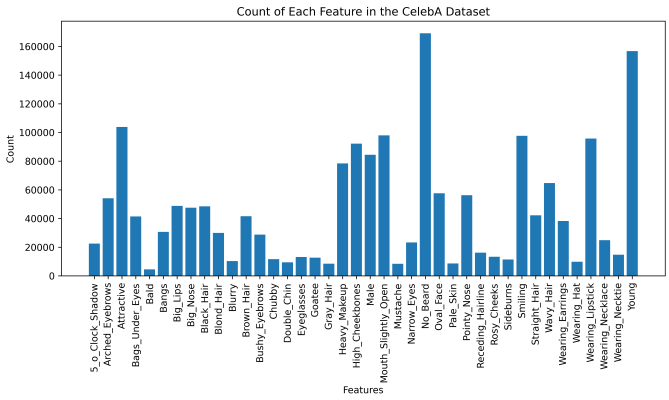

Fig. 2. This figure presents the frequency distribution of the 40 facial attributes in the CelebA dataset. Each bar represents one of the attributes, with the height of the bar indicating the frequency of that attribute in the dataset. The figure clearly illustrates the inherent bias in the dataset, with certain attributes appearing much more frequently than others.

As can be seen in Figure 2, in the context of this project, which involves utilizing the CelebA dataset for facial attribute recognition from images and comparing these with attributes derived from textual descriptions, it's important to note the inherent bias within the CelebA dataset. The dataset exhibits an imbalance, with certain attributes being represented more frequently than others. This uneven distribution could potentially impact the performance of the MOON model trained on this dataset, as it may develop a bias towards predicting attributes that occur more frequently. Consequently, when the output of the MOON model is compared with the facial attributes extracted from text using the fine-tuned BERT model, there might be inconsistencies, particularly for attributes that are less represented in the CelebA dataset. Therefore, it's essential to acknowledge this bias when interpreting the results

of the system, and consider implementing strategies such as oversampling of underrepresented attributes or applying class weighting during the training phase of the MOON model.

B. SynthFace-3K Dataset

This dataset has been generated for fine-tuning the BERT model to extract features from sentences describing human faces. Due to the unavailability of labeled datasets online, the fine-tuning process necessitated the creation of a synthetic dataset using the GPT API. The subsequent section will outline the dataset design procedure and its specifications.

In the initial step, a set of 40 labels obtained from the celebA dataset underwent tokenization and lemmatization processes. This preprocessing technique aimed to expand the model's ability to comprehend a wider range of descriptive sentences. By breaking down the labels into tokens, the model gains a more granular understanding of their underlying meaning. Additionally, lemmatization aids in normalizing the words, reducing inflectional variations and increasing linguistic coherence.

After tokenization, an NLTK library synonym generator function was applied to identify synonyms for each tokenized label. This step aimed to enrich the dataset with synonymous terms, further broadening the model's semantic understanding. By incorporating synonyms, the model can capture a diverse array of expressions and variations related to the facial features described in the dataset. The resulting tokenized features, along with their corresponding synonyms, were seamlessly integrated into the preprocessed dataset which is ready for the generation step.

The selection of the OpenAI 'davinci-3' engine was based on its well-established capabilities within the GPT-based architecture, making it an ideal choice for optimizing the sentence generation process.

To enhance the dataset design, careful consideration was given to various parameters. For concise and focused input, a maximum token size of 50 was set for descriptive sentences. Moreover, the dataset was structured to include 3 to 5 features per sentence, ensuring exemplary illustrations of sentence structures.

To strike a balance between coherence and creativity, the generator temperature was set to 0.7. This choice introduced a controlled level of randomness, allowing the model to generate original and diverse sentences while maintaining overall consistency. Consistency across the dataset was ensured by utilizing a text prompt. This prompt served as a guiding framework, establishing a consistent descriptive outline for all sentences.

As a result of these design parameters, the statistics of the dataset can be examined from Fig.3 where the number of features with label '1' are given.

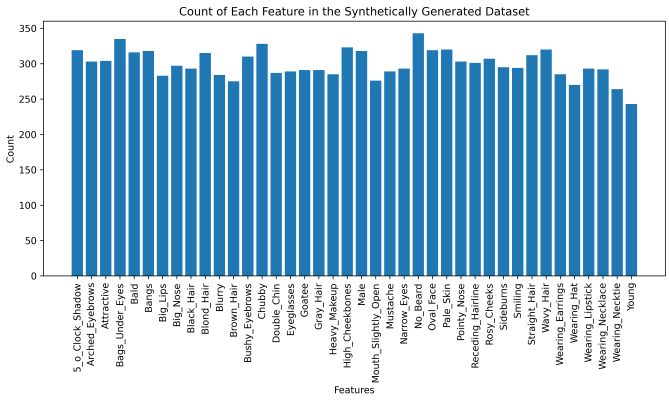

Fig. 3. This figure presents the frequency distribution of the 40 facial attributes in the SynthFace-3K dataset. Each bar represents one of the attributes, with the height of the bar indicating the frequency of that attribute in the dataset with label '1'.

This dataset will serve as the foundation for fine-tuning the BERT-based transformer model, which will be further discussed in subsequent sections of this report. By fine-tuning the BERT-based transformer model using this dataset, we aim to optimize its performance and tailor it specifically for the task at hand. The details of this fine-tuning process and its impact on the model's capabilities will be elaborated upon in the following sections.

C. LightenedMOON Architecture

The image feature extractor component of our project leverages the LightenedMOON model, a deep convolutional neural network (DCNN) specifically designed for facial attribute recognition. The MOON model is known for its ability to handle imbalanced training data and improve performance across multiple tasks. In this specific task, we aspire to maximize the accuracy of our prediction over all 40 attributes simultaneously.

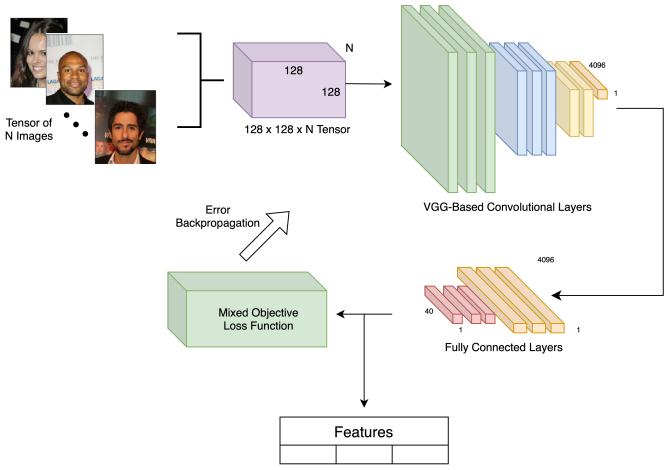

Fig. 4. Architecture schematic of the used model that takes in a tensor of N images, passes it through a CNN model with a lightened VGG backbone and fully connected layers with a domain-adapted multitask loss function.

As can be seen in Figure 4, the architecture of the MOON model is composed of several layers, including convolutional

layers inherited from VGG-net, pooling layers, and fully connected layers. The pre-trained convolutional layers that are adaptive to capture image features trained on the 'imagenet' dataset are responsible for extracting features from the input images. These layers apply a series of convolutional operations to the images with its pre-trained kernels, creating a set of feature maps that capture different aspects of the image, such as edges, shapes and textures and their combinations like eyes, face outline, hair, beard, etc. The pooling layers then reduce the dimensionality of these feature maps, making the model more computationally efficient and less prone to overfitting.

The fully connected layers at the end of the network, map the extracted features to the final output. In our case, the output is a 40-dimensional vector, where each dimension corresponds to one of the 40 facial attributes. One of the key features of the MOON model is its loss function, which combines multiple task objectives with domain adaptive re-weighting of propagated loss. This allows the model to handle imbalanced training data, where some facial attributes may be under-represented in the dataset. By adaptively re-weighting the loss, the model can learn to recognize these under-represented attributes more effectively. Mathematically, let M be the total number of attributes, $\mathbf{X}$ be a data tensor containing N input images, and $\mathbf{Y}$ be a corresponding N *x* M matrix of labels. Then, the aforementioned loss function can be described as [9]:

$$L(\mathbf{X}, \mathbf{Y}) = \sum_{j=1}^{N} \sum_{i=1}^{M} p\left(i \mid Y_{ji}\right) \left\| f_i\left(X_j\right) - Y_{ji} \right\|^2 \qquad (1)$$

By substituting the conventional loss layer of a Deep Convolutional Neural Network (DCNN) with a layer that implements Equation (1), we obtain the Mixed Objective Optimization Network (MOON) architecture. This architecture takes into account the correlations between attributes and has the capacity to adjust the bias inherent in the training dataset towards a desired distribution.

This model architecture and its pre-trained parameters have been used for this project to increase the maximum prediction performance.

D. FaceBERT Architecture

In this subsection, we will discuss the implementation of a fine-tuned language model for a specific application that involves mapping texts containing facial information to 40 predefined binary features.

The language model deployed for this task has been pre-trained and fine-tuned to cater specifically to this application. Its design focuses on the effective processing and analysis of text pertinent to facial attributes and detection. The architectural configuration of this language model can be viewed in Figure 5.

The CelebA dataset, which we use, features 40 distinct labels describing various human facial characteristics. Each image within the dataset has a binary label associated with each feature. As such, effective face detection hinges on the precise mapping of descriptive text to this 40-dimensional feature vector, which will subsequently be utilized in a comparator mechanism discussed in the following section.

Regrettably, rule-based methods fall short when it comes to comprehending descriptions of facial features within text, primarily due to their inability to discern dispersed relationships within sentences. As such, there's a clear need for a language model capable of accurately understanding facial feature descriptors like 'blonde hair', regardless of sentence structure.

- "The girl with blond hair"
- "She has fairy bright hair"
- "He has marvelous golden hair"

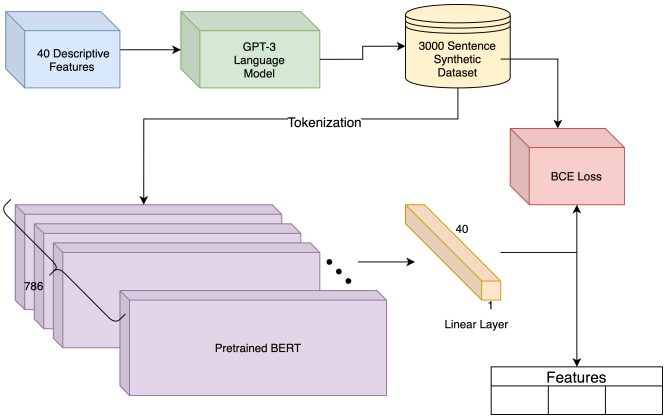

Fig. 5. Schematic of the NLP model architecture used for this assignment. The synthetic data has been generated by GPT language model and BERT model has been fine tuned by training with this data.

The proposed architecture, which facilitates mapping these sentences to the corresponding feature, can be reviewed in Figure 5. This was achieved using a pretrained language model based on BERT, augmented with an additional fully connected layer on top.

The main challenge in adopting a fine-tuning approach is the requirement of a labeled dataset, given its reliance on supervised learning methods. To address this, we created a synthetic dataset using a GPT model, which generated 3,000 distinct sentences. Each sentence contains between three and five phrases describing facial features. We initially tokenized the names of 40 features, corresponding to image categories. The tokens and their lemmas were used to create synonyms. These were then fed into a GPT sentence generation loop. The loop iteratively constructs a sentence up to a predetermined maximum length and saves the generated sentences along with their corresponding labels into a file with a CSV extension. The characteristics of this synthetically generated dataset are explained in detail in preceding sections.

We have fine-tuned BERT using this synthetically created dataset. The 768-dimensional CLS token, positioned at the end of the sentence, is funneled through a fully connected linear layer that reduces its dimensions to 40. Errors are then back-propagated throughout the model using Binary Cross-Entropy (BCE) loss, a widely preferred method for binary labeled classification problems. This process has enabled the model to successfully identify facial description phrases, leading to high accuracy.

The BERT transformer employed in this project was initialized with the parameters of the 'bert-based-uncased' pretrained model. We did not freeze the model parameters given the similarity between the original pretraining data and the fine-tuning data. Moreover, our experiments revealed that when BERT's parameters are frozen, the model converges to a specific loss and ceases training. However, by allowing the model to remain fully trainable, we observed it reaches minimum loss for both the training and validation sets.

E. IdentiFusion: FaceBERT and LightenedMOON Integration

After completing the training of the language model and the visual classifier, the subsequent step involves merging these two models to create the final architecture. The language model was trained using synthetic data and produces a (1x40) vector as output when given a descriptive sentence as input. Similarly, the visual classifier generates another (1x40) vector output based on the identified features of a given face.

Face identification and extraction from group images utilize Google's Facenet, [27] a deep learning architecture for face recognition tasks. This facilitates efficient face detection and cropping within images, enhancing the accuracy and efficiency of facial analysis by focusing on isolated faces.

Our main objective is to select the vector representation of a face from a set of faces that exhibits the highest similarity with the vector obtained from the language model. To achieve this, we utilize a method that assigns weights based on the bias of the model depending on the frequency statistics of the model's prediction outcomes. This method has been named as dynamic weight adjustment which updates the weights of the comparator function depending on the statistics of the prediction outcomes.

Let $\mathbf{V}(\mathbf{i})$ and $\mathbf{N}(\mathbf{t})$ represent the outputs of the computer vision and natural language processing models $\mathbf{V}$ and $\mathbf{N}$, respectively, given the inputs i and t. The outputs are then used to form vectors $\widetilde{\mathbf{v}} \in \mathbb{R}^{40}$ and $\widetilde{\mathbf{n}} \in \mathbb{R}^{40}$ as follows:

$$\widetilde{\mathbf{v}} = \mathbf{V}(i) = (\widetilde{v}_1, \widetilde{v}_2, \ldots, \widetilde{v}_{40})^\top \tag{2}$$

$$\widetilde{\mathbf{n}} = \mathbf{N}(t) = (\widetilde{n}_1, \widetilde{n}_2, \ldots, \widetilde{n}_{40})^\top \tag{3}$$

If there are m faces in an image , the algorithm compares the values of weighted dot products to find the best correlation among the face image feature vectors $\mathbf{i_1}, \mathbf{i_2}, \ldots, \mathbf{i_m}$. The largest correlation can be found by the following function:

$$\widetilde{\mathbf{v}}_\mathbf{k} = \mathbf{V}(\mathbf{i}_k) \tag{4}$$

Also, let $w \in \mathbb{R}^{40}$ be the weight vector, where each element w_i corresponds to the prediction frequency of the i^{th} class.

The comparison function, $C(n, v, w)$, takes the dot product of the NLP and CV outputs, and then weights them according to the prediction frequency of each class. The function is defined as follows:

$$C(\widetilde{\mathbf{v}}, \widetilde{\mathbf{n}}, \mathbf{w}) = \mathbf{w} \cdot (\widetilde{\mathbf{v}} \cdot \widetilde{\mathbf{n}}) \tag{5}$$

where $\{\cdot\}$ denotes the dot product. The output of this function gives a measure of similarity between the text description and the image, taking into account the model's prediction frequencies for each class.

$$\mathbf{i}_{detected} = \underset{k}{\operatorname{argmax}}(C(\widetilde{\mathbf{v}}, \widetilde{\mathbf{n}}, \mathbf{w}))$$

The given diagram on Fig.5 illustrates this algorithm.

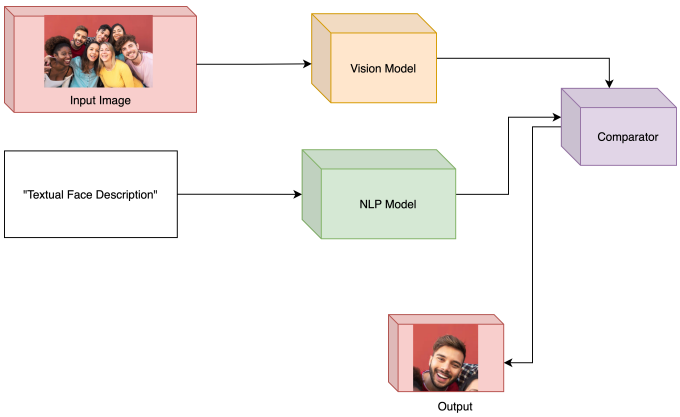

Fig. 6. This schematic illustrates the integration of our system's Natural Language Processing (NLP) and Computer Vision models. The NLP model, FaceBERT, extracts facial features from text, while the LightenedMOON model recognizes facial attributes from images. Their outputs are compared to predict the face matching the textual description, demonstrating the synergy of NLP and Computer Vision in our multimodal system.

This approach's prime advantage is its aptitude to tackle the complications presented by rare or outlier predictions. It reduces the risk of overfitting and undue emphasis on outlier data points by assigning lower weights to infrequent predictions, thus bolstering the model's generalization capabilities and ensuring robust, accurate predictions across diverse scenarios. Moreover, the weighted dot product technique empowers the model to utilize the collective wisdom encapsulated in the dataset. By considering prediction frequencies, it draws upon the consensus among data points, essentially applying a collective knowledge-based weighting mechanism. This strategy enables the model to make more informed decisions, significantly enhancing prediction accuracy and reliability.

Despite its advantages, this technique may have limitations due to reliance solely on prediction frequencies, potentially ignoring valuable but infrequent predictions. Such predictions can carry crucial information for specific tasks or contexts. Hence, while prediction frequencies effectively prioritize data, incorporating factors like domain expertise and contextual relevance ensures a comprehensive, accurate model.

IV. Results

In this section, we critically evaluate the performance metrics of our multimodal system. We start with the FaceBERT model, assessing its learning efficiency through training and validation loss plots. Next, we analyze LightenedMOON's attribute recognition accuracy against sample size, highlighting its handling of bias in the CelebA dataset. Finally, we gauge the success of our multimodal system as a whole.

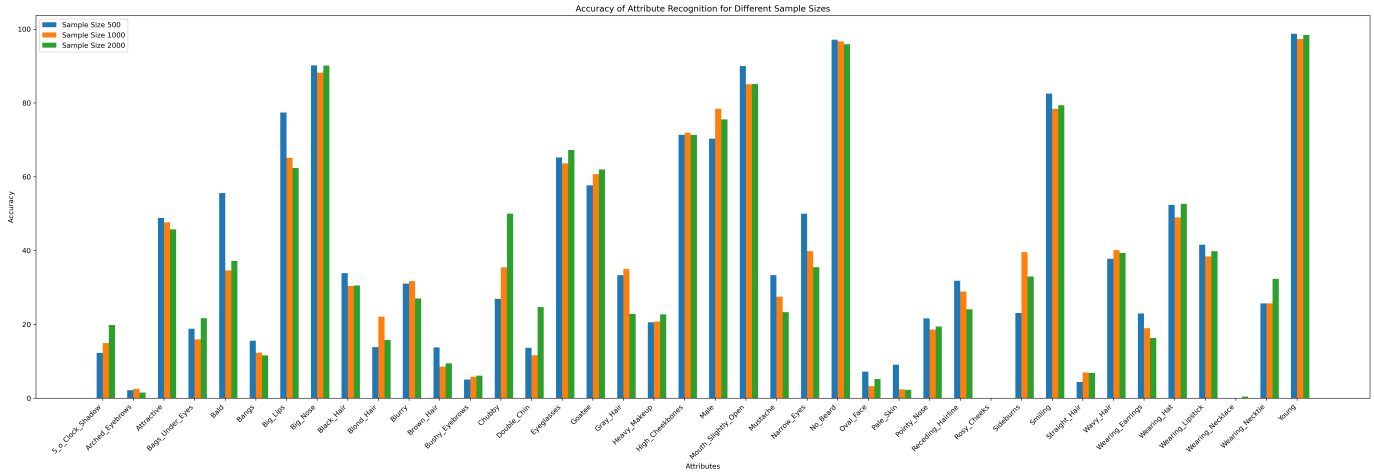

Fig. 7. Bar plot showcasing attribute accuracy across varied sample sizes in our multimodal system. Each attribute is represented by three bars, denoting sample sizes of 500, 1000, and 2000. The plot demonstrates the incremental improvement in attribute recognition with larger sample sizes, affirming the efficacy of our approach.

The training process for the FaceBERT model was carried out over 50 epochs, utilizing a learning rate of 2e-5. The training and validation loss plots, Figures 8 and 9 respectively, for the FaceBERT component of the IdentiFusion project shows positive results. At the start of training, the loss was high at 0.40 and 0.30 respectively, indicating a large gap between the model's predictions and actual values. However, both losses decreased exponentially during training, reaching a significantly lower level of 0.02 by epoch 40. This suggests that FaceBERT is effectively learning to predict 40 human facial attributes from text descriptions in the synthetic dataset, FaceSynth-3k, and successfully fine-tuning its internal parameters without overfitting. The low training and validation losses indicate high performance of the FaceBERT model, which is a crucial part of the multimodal system.

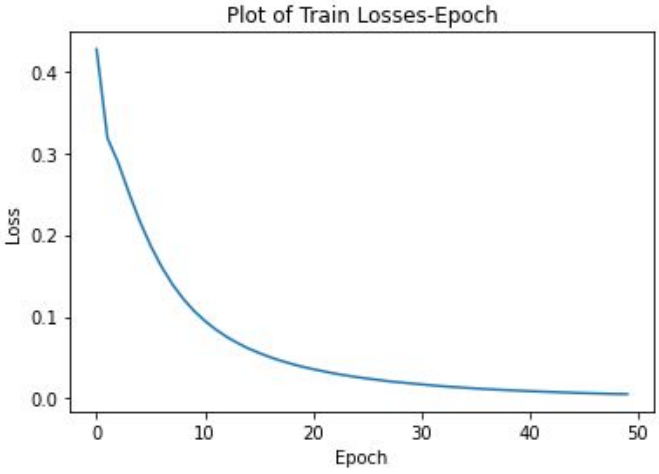

Fig. 8. This plot displays the reduction of training loss throughout the training of our NLP component. The loss diminishes exponentially. The decreasing trend indicates the successful learning and parameter adjustments of our NLP model, FaceBERT.

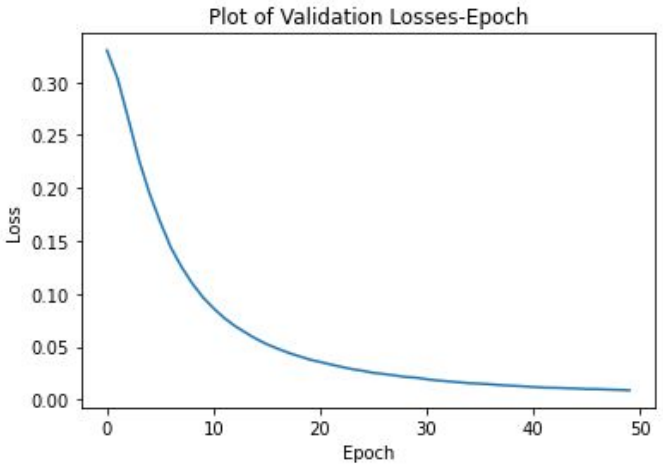

Fig. 9. This plot displays the reduction of validation loss throughout the training of our NLP component. The loss diminishes exponentially. The plot highlights the effective learning and generalization of our NLP model, FaceBERT.

In examining the frequency statistics of the accuracy for the vision model, as depicted in Figure 7, we have updated the weights of the comparator function. This was achieved by utilizing weights that normalize the accuracies. Table I showcases the experimentally derived accuracies using the vision model. The plot notably illustrates that features predicted less frequently, such as "Rosy_Cheeks" and "Pale_Skin", are assigned larger weights compared to those that are more commonly predicted. This nuanced weighting mechanism helps improve the performance of the model, particularly in the context of less frequent facial features. Following this, the individual models were integrated to formulate the IdentiFusion ensemble. We tested the predictions and performance of this model using an array of group photos as well as individual face photos from our surrounding environment to assess its accuracy.

After combining the models a series of tests on the IdentiFusion has been performed to assess the overall performance of the project. For this purpose, the Figure 10 is used as the

TABLE I
FEATURES AND WEIGHTS

Feature	Weight	Feature	Weight
5_o_Clock_Shadow	0.0629	Arched_Eyebrows	1.3514
Attractive	0.0221	Bags_Under_Eyes	0.0344
Bald	0.0233	Bangs	0.2024
Big_Lips	0.0165	Big_Nose	0.0112
Black_Hair	0.0283	Blond_Hair	0.0546
Blurry	0.0310	Brown_Hair	0.0855
Bushy_Eyebrows	0.1675	Chubby	0.0262
Double_Chin	0.0633	Eyeglasses	0.0165
Goatee	0.0175	Gray_Hair	0.0300
Heavy_Makeup	0.0568	High_Cheekbones	0.0139
Male	0.0131	Mouth_Slightly_Open	0.0117
Mustache	0.0400	Narrow_Eyes	0.0347
No_Beard	0.0104	Oval_Face	0.1567
Pale_Skin	100.0	Pointy_Nose	0.0507
Receding_Hairline	0.0409	Rosy_Cheeks	100.0
Sideburns	0.0240	Smiling	0.0120
Straight_Hair	0.3165	Wavy_Hair	0.0233
Wearing_Earrings	0.0510	Wearing_Hat	0.0140

face set. Figure 10 contains 25 people with various ethnicity, gender, age and features. This diversity will enable to model to differentiate between different descriptions.

Fig. 10. Evaluation images for the multi-modal system. This set of 25 diverse face images from stock photos [28]was used to evaluate our multi-modal system's performance in matching textual face descriptions with visual representations.

Now, the input will consist of a descriptive prompt, and the text model will generate responses along with corresponding images and their visually detected features for multiple instances.

1) "She possessed a captivating appearance with her plump, attractive big lips, noticeable bags under her eyes, and a short, flowing blond hair."

Fig. 11. The face image in row 2, column 5 of the evaluation images set in Figure 10.

Detected Vision Features: Blond_Hair, Mouth_Slightly_Open, No_Beard, Wearing_Lipstick, Young
Detected Textual Features: Attractive, Bags_Under_Eyes, Big_Lips, Blond_Hair

2) With black hair and a big nose, man has a distinctive and striking appearance.

Fig. 12. The face image in row 1, column 2 of the evaluation images set in Figure 10.

Detected Vision Features: Attractive, Bags_Under_Eyes, Black_Hair, Bushy_Eyebrows, High_Cheekbones, Male, Narrow_Eyes, No_Beard, Smiling, Straight_Hair, Young
Detected Textual Features: : Big_Nose, Black_Hair, Male

3) The man had a five o'clock shadow, arched eyebrows, and a confident smile.

Fig. 13. The face image in row 2, column 2 of the evaluation images set in Figure 10.

Detected Vision Features: 5_o_Clock_Shadow, Black_Hair, High_Cheekbones, Male, Mouth_Slightly_Open, Narrow_Eyes, No_Beard, Smiling, Straight_Hair, Young
Detected Textual Features: : 5_o_Clock_Shadow, Arched_Eyebrows, Male, Smiling

4) A brown haired attractive woman has a beautiful smile with her big lips and her heavy makeup

Fig. 14. The face image in row 3, column 2 of the evaluation images set in Figure 10.

Detected Vision Features: Arched_Eyebrows, Attractive, Big_Lips, Blurry, Heavy_Makeup, High_Cheekbones, Mouth_Slightly_Open, No_Beard, Smiling, Wearing_Lipstick, Young
Detected Textual Features: : Attractive, Big_Lips, Brown_Hair, Heavy_Makeup, Smiling

Overall the detected features are matching with the given feature except with slight mismatches in the given description. For example, the first detected woman with blond hair does not have bags under her eyes, yet she is still chosen. This is probably due to the limited number of people present in the image. Similarly, the male in the third example does not have arched eyebrows, but the overall description almost matches with our expectations. This concludes the explanation of the results of the model that has been proposed for detecting faces from group images according to facial descriptions.

V. Discussion & Conclusion

Our study has demonstrated that the overall efficacy of the analysis process heavily relies on the synergistic operation of the visual AI and Natural Language Processing (NLP) models. The NLP model's performance has been quite promising, as it consistently recognized features in a sentence, regardless of the complexity of the grammatical structure or the synonymous usage. The model proved its capability to navigate through diverse sentence forms and structures, suggesting a solid foundation for a comprehensive analysis.

However, the visual AI model's performance has manifested some potential areas for improvement. Although it was adept at handling certain tasks, its accuracy dwindled in a few areas. Notably, it grappled with recognizing features with comparatively small data sets. Furthermore, the image captioning module, despite being capable of generating informative captions, occasionally faltered to retain contextual relevancy. This inconsistency was more pronounced for features associated with insufficient training data, pointing towards a potential area of improvement.

To bridge these gaps and bolster the pre-trained vision model's performance, we devised a strategy employing a weighted cosine similarity approach. This method assigns more significant weight to unique features, considering them more critical in the process of individual identification. For instance, the attribute "blond hair" might prove more significant than the common parameter "young" due to its discriminative potential. This approach aims to fine-tune the model's focus toward features with greater individualization potential, thereby augmenting its identification accuracy.

Through this novel weighted scheme, our research aims to bolster the visual AI model's performance, specifically, its ability to prioritize and consider discriminative features. This would subsequently enhance its overall efficacy in individual identification tasks. Nonetheless, despite the strides made, further improvements and optimizations are necessary to harness the full potential of the AI models. Future work could focus on refining the model's accuracy for feature identification with scarce training data, as well as improving the contextual relevance in image captioning tasks. Additionally, augmenting the training dataset might prove beneficial, with a particular focus on increasing the diversity and volume of less common features. We believe these advancements would significantly contribute to improving the robustness of our AI model, thus providing a more holistic and accurate analysis tool for future applications.

References

[1] Vinyals, Oriol, et al. "Show and Tell: A Neural Image Caption Generator." ArXiv.org, 2014, arxiv.org/abs/1411.4555.
[2] Agrawal, Aishwarya, et al. "VQA: Visual Question Answering." ArXiv:1505.00468 [Cs], 26 Oct. 2016, arxiv.org/abs/1505.00468.
[3] Reed, Scott, et al. "Generative Adversarial Text to Image Synthesis." ArXiv.org, 2016, arxiv.org/abs/1605.05396.
[4] Lin, Kevin K, et al. "Deep Learning of Binary Hash Codes for Fast Image Retrieval." Computer Vision and Pattern Recognition, 7 June 2015, https://doi.org/10.1109/cvprw.2015.7301269. Accessed 30 Apr. 2023.

[5] Denman, Simon, et al. “Automatic Surveillance in Transportation Hubs: No Longer Just about Catching the Bad Guy.” Expert Systems with Applications, vol. 42, no. 24, Dec. 2015, pp. 9449–9467, https://doi.org/10.1016/j.eswa.2015.08.001. Accessed 29 Jan. 2023.
[6] Devlin, Jacob, et al. “BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding.” ArXiv.org, 11 Oct. 2018, arxiv.org/abs/1810.04805.
[7] Yang, Zhilin, et al. “XLNet: Generalized Autoregressive Pretraining for Language Understanding.” ArXiv.org, 2019, arxiv.org/abs/1906.08237.
[8] Clark, Kevin, et al. “ELECTRA: Pre-Training Text Encoders as Discriminators rather than Generators.” ArXiv:2003.10555 [Cs], 23 Mar. 2020, arxiv.org/abs/2003.10555.
[9] Rudd, Ethan, et al. “MOON: A Mixed Objective Optimization Network for the Recognition of Facial Attributes.” ArXiv.org, 2016, arxiv.org/abs/1603.07027.
[10] B. Siddiquie, R.S. Feris, and L.S. Davis, “Image ranking and retrieval based on multi-attribute queries,” in Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2011, pp. 801-808.
[11] N. Kumar, A.C. Berg, P.N. Belhumeur, and S.K. Nayar, “Attribute and simile classifiers for face verification,” in Proc. IEEE Int. Conf. Comput. Vis., 2009, pp. 365-372.
[12] N. Kumar, A.C. Berg, P.N. Belhumeur, and S.K. Nayar, “Describable visual attributes for face verification and image search,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 10, pp. 1962- 1977, 2011.
[13] A. Bergamo, L. Bazzani, D. Anguelov, and L. Torresani, ”Self-taught object localization with deep networks,” arXiv preprint arXiv:1409.3964, 2014.
[14] S. Khamis, C.H. Kuo, V.K. Singh, V.D. Shet, and L.S. Davis, “Joint learning for attribute-consistent person re-identification,” in Proc. Eur. Conf. Comput. Vis., 2014, pp. 134-146.
[15] G. Qi, X. Hua, and H. Zhang, “Learning semantic distance from community-tagged media collection,” in Proc. 17th ACM Int. Conf. Multimedia, 2009, pp. 243-252.
[16] S. Kang, D. Lee, and C.D. Yoo, “Face attribute classification using attribute aware correlation map and gated convolutional neural networks,” in Proc. IEEE Int. Conf. Image Process., 2015, pp. 4922- 4926.
[17] Y. Zhong, J. Sullivan, and H. Li, “Leveraging mid-level deep representations for predicting face attributes in the wild,” in Proc. IEEE Int. Conf. Image Process., 2016, pp. 3239-3243.
[18] N. Zhang, M. Paluri, M. Ranzato, T. Darrell, and L. Bourdev, “Panda: Pose aligned networks for deep attribute modeling,” in Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2014, pp. 1637-1644.
[19] F. Wang, H. Han, T. Almaev, and S. Shan, “Deep multi-task learning for joint prediction of heterogeneous face attributes,” in Proc. IEEE conf. Autom. Face Gesture Recog., 2017, pp. 173-179.
[20] M. Xu, F. Chen, L. Li, C. Shen, P. Lv, B. Zhou, and R. Ji, “Bio-Inspired deep attribute learning towards facial aesthetic prediction,” in IEEE Trans. Affective Comput., doi: 10.1109/TAFFC.2018.2868651, 2018.
[21] N. Zhuang, Y. Yan, S. Chen and H. Wang, “Multi-task learning of cascaded CNN for facial attribute classification,” in Proc. Int. Conf. Pattern Recog., 2018, pp. 2069-2074.
[22] Li, Zhiheng, et al. “Discover and Mitigate Unknown Biases with Debiasing Alternate Networks.” ArXiv.org, 7 Sept. 2022, arxiv.org/abs/2207.10077.
[23] Brown, Tom B., et al. “Language Models Are Few-Shot Learners.” Arxiv.org, 28 May 2020, arxiv.org/abs/2005.14165.
[24] Liu, Yinhan, et al. “RoBERTa: A Robustly Optimized BERT Pretraining Approach.” ArXiv.org, 2019, arxiv.org/abs/1907.11692.
[25] Patra, Barun, et al. Beyond English-Centric Bitexts for Better Multilingual Language Representation Learning.
[26] Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: International Conference on Computer Vision, IEEE (2015) 3730–3738
[27] P. F. T. Madio, ”A FaceNet-Style Approach to Facial Recognition,” Towards Data Science, 2019. [Online]. Available: https://towardsdatascience.com/a-facenet-style-approach-to-facial-recognition-dc0944efe8d1
[28] O. Alija, ”Outlay of 25 Multiracial Faces,” iStock, Apr. 8, 2015. [Online]. Available: https://www.istockphoto.com/tr/foto

I. Appendix

A. Group Members' Contributions

1) Efe Tarhan's Contribution: He attempted to train several computer vision models after finding the celebA dataset which is proper for the project, including some designed with a transfer learning approach. However, since the accuracy of these models fell short of the expected level, he delegated this task to Ege Yüceel and shifted his focus to the training of the BERT model. He fine-tuned this model using the synthetically generated SynthFace-3K dataset. His work included plotting statistical data and loss functions, designing the figures of the architectures of he models and he made significant contributions to the methodology and results sections of the report. Importantly, throughout the project, he worked collaboratively and maintained close contact with his teammates to ensure a coordinated effort.

2) Ege Yüceel's Contribution: He delved into the foundational concepts required for the computer vision aspect of the project and leaded the training of the CelebA dataset. Recognizing the potential of the LightenedMOON model, he integrated it into the project to work in coherence with the NLP model. He was also responsible for generating the synthetic dataset and performed several modifications on both vision and NLP models, incorporating feedback from his teammates. Furthermore, he facilitated the alignment of images with detected features from the description sentences, presenting the correlated data in the results section of the report. He made substantial contributions to the results, discussion, and conclusion sections of the report. Importantly, throughout the project, he worked collaboratively and maintained close contact with his teammates to ensure a coordinated effort.

3) Berk Alemdar's Contribution: Berk Alemdar carried out essential research pertinent to the project for finding approaches on both NLP and vision identifying valuable resources in the literature. He designed a dynamic weight adjustment mechanism to determine and compare vector similarities, thereby developing the final ensemble of the models.His contributions to the written project include the introduction, related work, and methodology sections. Notably, his commitment to teamwork ensured a collaborative atmosphere and consistent communication with his peers, crucial for a harmonized project effort.

B. SynthFace-3K Dataset Generation Code

```
from nltk.corpus import wordnet
from random import choice, randint
import random
import openai
import csv
openai.api_key = "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"

features = ["5_o_Clock_Shadow","Arched_Eyebrows","Attractive","Bags_Under_Eyes","Bald
    ", "Bangs","Big_Lips","Big_Nose",
            "Black_Hair","Blond_Hair","Blurry","Brown_Hair","Bushy_Eyebrows","Chubby
                ","Double_Chin","Eyeglasses","Goatee",
            "Gray_Hair", "Heavy_Makeup","High_Cheekbones","Male","Mouth_Slightly_Open
                ","Mustache","Narrow_Eyes","No_Beard",
            "Oval_Face","Pale_Skin","Pointy_Nose","Receding_Hairline","Rosy_Cheeks","
                Sideburns","Smiling","Straight_Hair",
            "Wavy_Hair","Wearing_Earrings","Wearing_Hat","Wearing_Lipstick","
                Wearing_Necklace","Wearing_Necktie","Young"]

def get_synonyms(word):
    synonyms = set()
    for syn in wordnet.synsets(word):
        for lemma in syn.lemmas():
            synonyms.add(lemma.name())
    return list(synonyms)

def generate_sentence_template():
    num_features = random.randint(3, 5)  # choose a random number of features
    chosen_features = random.sample(features, num_features)  # choose random features

    prompt = "Generate a single sentence template describing a person with unique
        sentences with the following features: "
    prompt += ", ".join(chosen_features) + "."
```

```
    prompt += "Dont just write features next to each other as it is. Use a unique
       sentence structure and with different types of expressions. You may sometimes
       use synonyms."

    response = openai.Completion.create(
        engine="text-davinci-003",
        prompt=prompt,
        temperature=0.7,
        max_tokens=50,
        top_p=0.9,
        n=1
    )

    template = response.choices[0].text.strip()
    return template, chosen_features

output_file = "sentences.csv"

with open(output_file, mode="w", newline="") as file:
    writer = csv.writer(file)
    writer.writerow(["sentence"] + features)
    count = 0
    for _ in range(3000):
        count +=1
        template, chosen_features = generate_sentence_template()
        feature_synonyms = {}
        for feature in chosen_features:
            synonyms = get_synonyms(feature)
            if synonyms:
                feature_synonyms[feature] = synonyms
        sentence = template
        feature_values = []
        for feature in features:
            if feature in chosen_features:
                value = 1
            else:
                value = 0
            feature_values.append(value)

        writer.writerow([sentence] + feature_values)
        print("Sentence {} completed".format(count))
```

C. FaceBERT Training Code

```
import torch
import torch.nn as nn
import pandas as pd
from torch.utils.data import DataLoader, Dataset
from sklearn.model_selection import train_test_split
from transformers import BertTokenizer, BertModel, AdamW

# Define the custom dataset class
class FacialFeatureDataset(Dataset):
    def _init_(self, sentences, features, tokenizer):
        self.sentences = sentences
        self.features = features
        self.tokenizer = tokenizer
```

```
    def _len_(self):
        return len(self.sentences)

    def _getitem_(self, idx):
        sentence = self.sentences[idx]
        feature = self.features[idx]
        encoded_inputs = self.tokenizer.encode_plus(
            sentence,
            add_special_tokens=True,
            padding="max_length",
            truncation=True,
            max_length=128,
            return_tensors="pt",
        )
        return {
            "input_ids": encoded_inputs["input_ids"].squeeze(),
            "attention_mask": encoded_inputs["attention_mask"].squeeze(),
            "features": torch.tensor(feature, dtype=torch.float),
        }

# Define the transformer model with additional layers for classification
class FacialFeatureModel(nn.Module):
    def _init_(self, pretrained_model_name, num_features):
        super(FacialFeatureModel, self)._init_()
        self.bert = BertModel.from_pretrained(pretrained_model_name)
        self.classifier = nn.Linear(self.bert.config.hidden_size, num_features)

    def forward(self, input_ids, attention_mask):
        outputs = self.bert(input_ids=input_ids, attention_mask=attention_mask)
        pooled_output = outputs.pooler_output
        logits = self.classifier(pooled_output)
        return logits

# Set random seed for reproducibility
torch.manual_seed(42)

# Load the dataset from the CSV file
df = pd.read_csv('your_dataset.csv')
sentences = df['sentence'].tolist()
features = df.iloc[:, 1:].values

# Split the dataset into train, validation, and test sets
train_sentences, test_sentences, train_features, test_features = train_test_split(
    sentences, features, test_size=0.2, random_state=42
)
train_sentences, val_sentences, train_features, val_features = train_test_split(
    train_sentences, train_features, test_size=0.1, random_state=42
)

# Initialize the tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Create dataset objects
train_dataset = FacialFeatureDataset(train_sentences, train_features, tokenizer)
val_dataset = FacialFeatureDataset(val_sentences, val_features, tokenizer)
test_dataset = FacialFeatureDataset(test_sentences, test_features, tokenizer)
```

```
# Define the batch size and number of training epochs
batch_size = 16
num_epochs = 10

# Create data loaders
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=batch_size)
test_loader = DataLoader(test_dataset, batch_size=batch_size)

# Initialize the model and move it to the appropriate device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = FacialFeatureModel(pretrained_model_name='bert-base-uncased', num_features
   =40).to(device)

# Freeze the parameters of the BERT base model
for param in model.bert.parameters():
    param.requires_grad = False

# Define the optimizer and loss function
optimizer = AdamW(model.classifier.parameters(), lr=2e-5)
criterion = nn.BCEWithLogitsLoss()

# Training loop
for epoch in range(num_epochs):
    model.train()
    train_loss = 0.0

    for batch in train_loader:
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        features = batch['features'].to(device)

        optimizer.zero_grad()

        outputs = model(input_ids=input_ids, attention_mask=attention_mask)
        loss = criterion(outputs, features)
        loss.backward()
        optimizer.step()

        train_loss += loss.item()

    # Validation loop
    model.eval()
    val_loss = 0.0

    with torch.no_grad():
        for batch in val_loader:
            input_ids = batch['input_ids'].to(device)
            attention_mask = batch['attention_mask'].to(device)
            features = batch['features'].to(device)

            outputs = model(input_ids=input_ids, attention_mask=attention_mask)
            loss = criterion(outputs, features)

            val_loss += loss.item()

    train_loss /= len(train_loader)
    val_loss /= len(val_loader)
```

```
    print(f"Epoch {epoch+1}: Train Loss: {train_loss:.4f}, Val Loss: {val_loss:.4f}")

# Evaluation on test set
model.eval()
test_loss = 0.0

with torch.no_grad():
    for batch in test_loader:
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        features = batch['features'].to(device)

        outputs = model(input_ids=input_ids, attention_mask=attention_mask)
        loss = criterion(outputs, features)

        test_loss += loss.item()

test_loss /= len(test_loader)
print(f"Test Loss: {test_loss:.4f}")
```

D. IdentiFusion Code that Includes the NLP and Vision Model

```
#unified

import torch
import torch.nn as nn
import pandas as pd
from torch.utils.data import DataLoader, Dataset
from sklearn.model_selection import train_test_split
from transformers import BertTokenizer, BertModel, AdamW
import argparse
import cv2
import glob
import mxnet as mx
import numpy as np
import os
import pandas as pd
from mxnet_moon.lightened_moon import lightened_moon_feature
from pathlib import Path
import sys


face_proto_file = "C:/Users/yuceelege/Desktop/vision/model/facenet/
    opencv_face_detector.pbtxt"
face_model_file = "C:/Users/yuceelege/Desktop/vision/model/facenet/
    opencv_face_detector_uint8.pb"
age_proto_file = "C:/Users/yuceelege/Desktop/vision/model/age/age_deploy.prototxt"
age_model_file = "C:/Users/yuceelege/Desktop/vision/model/age/age_net.caffemodel"
gender_proto_file = "C:/Users/yuceelege/Desktop/vision/model/gender/gender_deploy.
    prototxt"
gender_model_file = "C:/Users/yuceelege/Desktop/vision/model/gender/gender_net.
    caffemodel"
image_dir = 'C:/Users/yuceelege/Desktop/vision/Dataset/'
app_root_directory = 'C:/Users/yuceelege/Desktop/vision/'

face_model = cv2.dnn.readNet(face_model_file, face_proto_file)
```

```
age_model = cv2.dnn.readNet(age_model_file, age_proto_file)
gender_model = cv2.dnn.readNet(gender_model_file, gender_proto_file)


def detectFaces(model, input_image, confidence_threshold=0.7):
    copy_image = input_image.copy()
    img_height = copy_image.shape[0]
    img_width = copy_image.shape[1]
    blob_from_image = cv2.dnn.blobFromImage(copy_image, 1.0, (300, 300), [104, 117,
       123], True, False)
    model.setInput(blob_from_image)
    model_detections = model.forward()
    detected_faces = []
    for i in range(model_detections.shape[2]):
        confidence_score = model_detections[0, 0, i, 2]
        if confidence_score > confidence_threshold:
            start_x = int(int(model_detections[0, 0, i, 3] * img_width)*0.96)
            start_y = int(int(model_detections[0, 0, i, 4] * img_height)*0.96)
            end_x = int(int(model_detections[0, 0, i, 5] * img_width)*1.04)
            end_y = int(int(model_detections[0, 0, i, 6] * img_height)*1.04)

            detected_faces.append([start_x, start_y, end_x, end_y])
            cv2.rectangle(copy_image, (start_x, start_y), (end_x, end_y), (0, 255, 0)
               , int(round(img_height/150)), 8)

    return copy_image, detected_faces

def predictGender(input_image):
    MEAN_VALUES_MODEL=(78.4263377603, 87.7689143744, 114.895847746)
    genderTypes=['Male','Female']
    extra_padding=20
    cropped_face=input_image
    blob_from_image=cv2.dnn.blobFromImage(cropped_face, 1.0, (227,227),
       MEAN_VALUES_MODEL, swapRB=False)

    gender_model.setInput(blob_from_image)
    predictions_gender=gender_model.forward()
    predicted_gender=genderTypes[predictions_gender[0].argmax()]

    return predicted_gender
def crop_and_save_image(image_path, coordinates, output_path):
    image = cv2.imread(image_path)
    start_x, start_y, end_x, end_y = coordinates
    cropped_image = image[int(0.98*start_y):int(end_y*1.02), int(start_x*0.98):int(
       end_x*1.02)]

    _, extension = os.path.splitext(output_path)

    supported_extensions = [".jpg", ".jpeg", ".png"]
    if extension.lower() not in supported_extensions:
        raise ValueError("Unsupported file extension. Please use .jpg, .jpeg, or .png
           .")
    cv2.imwrite(output_path, cropped_image)
from sklearn.metrics.pairwise import cosine_similarity
def Comparator(people,vector):
    max_sim_val = 0
    max_sim = 0
```

```
    for i in people:
        weights =
            [0.06285355122564425,1.3513513513513513,0.022055580061755623,0.034376074252320386,0

        0.20242914979757085,0.016485328058028357,0.011228385358185492,0.02826455624646693,0.05

        0.08547008547008547,0.16750418760469013,0.026246719160104987,0.06333122229259025,0.016

        0.030003000300030006,0.056753688989784334,0.013869625520110958,0.013144058885383806,0.

        0.010393929944912172,0.15673981191222572,100.0,0.05070993914807303,0.0409165302782324,

        0.3164556962025316,0.02328288707799767,0.05099439061703213,0.013999720005599887,0.0262

        pred_vec = np.array(i[1])
        pred_vec = [a * b for a, b in zip(pred_vec, weights)]
        if np.dot(pred_vec,vector)> max_sim_val:
            max_sim_val = np.dot(pred_vec,vector)
            max_sim = i
    return max_sim


torch.manual_seed(42)
#%%
Sentence =  "An attractive woman with a straight hair and nice lips is looking at you
    "


#%%
features = ["5_o_Clock_Shadow","Arched_Eyebrows","Attractive","Bags_Under_Eyes","Bald
    ", "Bangs","Big_Lips","Big_Nose",
            "Black_Hair","Blond_Hair","Blurry","Brown_Hair","Bushy_Eyebrows","Chubby
                ","Double_Chin","Eyeglasses","Goatee",
            "Gray_Hair", "Heavy_Makeup","High_Cheekbones","Male","Mouth_Slightly_Open
                ","Mustache","Narrow_Eyes","No_Beard",
            "Oval_Face","Pale_Skin","Pointy_Nose","Receding_Hairline","Rosy_Cheeks","
                Sideburns","Smiling","Straight_Hair",
            "Wavy_Hair","Wearing_Earrings","Wearing_Hat","Wearing_Lipstick","
                Wearing_Necklace","Wearing_Necktie","Young"]
#%%
device = torch.device("cpu")

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')


tokenized_inputs = tokenizer(Sentence, padding=True, truncation=True, return_tensors
    ='pt')


input_ids = tokenized_inputs['input_ids'].to(device)
attention_mask = tokenized_inputs['attention_mask'].to(device)


class FacialFeatureModel(nn.Module):
    def __init__(self, pretrained_model_name, num_features):
        super(FacialFeatureModel, self).__init__()
        self.bert = BertModel.from_pretrained(pretrained_model_name)
        self.classifier = nn.Linear(self.bert.config.hidden_size, num_features)
```

```
    def forward(self, input_ids, attention_mask):
        outputs = self.bert(input_ids=input_ids, attention_mask=attention_mask)
        pooled_output = outputs.pooler_output
        logits = self.classifier(pooled_output)
        return logits
model = FacialFeatureModel(pretrained_model_name='bert-base-uncased', num_features
   =40).to(device)
state_dict = torch.load('NLP_TXT.pt',map_location=torch.device('cpu'))
model.load_state_dict(state_dict)
model.eval()



with torch.no_grad():
    logits = model(input_ids, attention_mask)

sigmoid = nn.Sigmoid()

text_vector = sigmoid(logits)
if text_vector[0][20] < 0.2:
    text_vector[0][20] = -10
else:
    text_vector[0][20] = 10

people = []
gpu_ids = None
image_size = 128
padding = None
model_prefix = 'C:/Users/yuceelege/Desktop/vision/model/lightened_moon/
   lightened_moon_fuse'
feature_extractor = lightened_moon_feature(num_classes=40, use_fuse=True)
devices = mx.cpu() if gpu_ids is None else [mx.gpu(int(i)) for i in gpu_ids.split
   (',')]
_, parameters, auxiliary_parameters = mx.model.load_checkpoint(model_prefix, 82)
image_directory = os.path.join(image_dir)

if os.path.exists(image_directory):
    file_names = os.listdir(image_dir)
    image_path = image_dir+file_names[0]
    input_image = cv2.imread(image_path)
    gray_img = cv2.imread(image_path, -1)
    result_image, face_boxes = detectFaces(face_model, input_image)
    if not face_boxes:
        print("There is no face indetifed.")
    count = 0
    output_path = "C:/Users/yuceelege/Desktop/vision/images/"
    for box in face_boxes:
        index = "Number_{}.jpeg".format(count)
        crop_and_save_image(image_path,box,output_path+index)
        count += 1



people = []
file_names = os.listdir("C:/Users/yuceelege/Desktop/vision/images/")
for i in file_names:
    path = "C:/Users/yuceelege/Desktop/vision/images/"+i
```

```
input_image = cv2.imread(path)
gray_img = cv2.imread(path, -1)
detected_gender = predictGender(input_image)
start_x = box[0]
box_width = box[2] - box[0]
start_y = box[1]
box_height = box[3] - box[1]
end_x = box[2]
end_y = box[3]
grayscale_image = cv2.cvtColor(gray_img, cv2.COLOR_BGR2GRAY)
grayscale_image = cv2.resize(grayscale_image, (image_size, image_size))/255.0
final_img = np.expand_dims(np.expand_dims(grayscale_image, axis=0), axis=0)
parameters['data'] = mx.nd.array(final_img, devices)
executor = feature_extractor.bind(devices, parameters, args_grad=None, grad_req="
    null", aux_states=auxiliary_parameters)
executor.forward(is_train=False)
executor.outputs[0].wait_to_read()
output_data = executor.outputs[0].asnumpy()
attribute_names = ["5_o_Clock_Shadow","Arched_Eyebrows","Attractive","
    Bags_Under_Eyes","Bald", "Bangs","Big_Lips","Big_Nose",
                    "Black_Hair","Blond_Hair","Blurry","Brown_Hair","
                        Bushy_Eyebrows","Chubby","Double_Chin","Eyeglasses","Goatee
                        ",
                    "Gray_Hair", "Heavy_Makeup","High_Cheekbones","Male","
                        Mouth_Slightly_Open","Mustache","Narrow_Eyes","No_Beard",
                    "Oval_Face","Pale_Skin","Pointy_Nose","Receding_Hairline","
                        Rosy_Cheeks","Sideburns","Smiling","Straight_Hair",
                    "Wavy_Hair","Wearing_Earrings","Wearing_Hat","Wearing_Lipstick
                        ","Wearing_Necklace","Wearing_Necktie","Young"]
predictions = np.ones(40)
attribute_dict = {}
detected_attributes = []
for i in range(40):
    attr_name = attribute_names[i].rjust(20)
    if output_data[0][i] < 0:
        attribute_dict[attr_name] = 'No'
    else:
        attribute_dict[attr_name] = 'Yes'
        detected_attributes.append(attribute_names[i])
attributes = ["5_o_Clock_Shadow","Arched_Eyebrows","Attractive","Bags_Under_Eyes
    ","Bald", "Bangs","Big_Lips","Big_Nose",
            "Black_Hair","Blond_Hair","Blurry","Brown_Hair","Bushy_Eyebrows","
                Chubby","Double_Chin","Eyeglasses","Goatee",
            "Gray_Hair", "Heavy_Makeup","High_Cheekbones","Male","
                Mouth_Slightly_Open","Mustache","Narrow_Eyes","No_Beard",
            "Oval_Face","Pale_Skin","Pointy_Nose","Receding_Hairline","Rosy_Cheeks
                ","Sideburns","Smiling","Straight_Hair",
            "Wavy_Hair","Wearing_Earrings","Wearing_Hat","Wearing_Lipstick","
                Wearing_Necklace","Wearing_Necktie","Young"]
attribute_indicator = [0]*40
for i, attr in enumerate(attributes):
    if attr in detected_attributes:
        attribute_indicator[i] = 1
if detected_gender == 'Male':
    attribute_indicator[20] = 10
else:
    attribute_indicator[20] = -10
detection = (input_image, attribute_indicator)
```

```
    people.append(detection)

import matplotlib.pyplot as plt
a = Comparator(people,text_vector[0].reshape(-1,1))
plt.imshow(a[0])
```