Big Data Coursework

ALEX WOODROOF | UP2118496

Contents

Masked Face Recognition

ALEX WOODROOF | UP2118496

Introduction

Face coverings became a common occurrence during the COVID-19 pandemic, underscoring the critical need for advanced pattern recognition systems capable of identifying individuals with obscured faces [1]. This task is inherently challenging as it involves detecting objects of special significance under circumstances where zero or more of these objects may appear within a single frame.

The primary objective is to immerse in the intricate domain of detecting faces with diminished visibility, specifically discerning whether individuals are wearing or not wearing face masks. This analysis draws insights from datasets such as WIDER FACE and MASKED FACE (MAFA) [2], providing a comprehensive exploration of facial recognition challenges in diverse scenarios. [4]

Understanding the problem…

Facial recognition is already a challenging task when all facial features are readily available. This challenge becomes exponentially more intricate when attempting to identify someone with only select features visible. Covering prominent facial features makes it nearly impossible to discern an individual's identity. Fortunately, the objective is not to identify the individual but rather to determine a simpler fact - whether they are or are not wearing a mask. The dataset provides this information in the form of occluder_type [2], giving us a distinctive new facial feature to work with: the presence or absence of a mask.

Abstraction into subtasks…

The task is to generate several models to detect masked individuals, highlight their location within an image, and compare efficiency. Crucial steps include:

- Data Preprocessing: Clean and standardize the MAFA datasets to ensure quality and consistency.
- Feature Extraction and Augmentation: Implement both handcrafted and non-handcrafted feature extraction methods such as k-NN. Creating image variations for each including rotating and blurring to boost image recognition of worse quality images.
- Train and Test: Train carefully selected models on the training dataset then test using images provided in the test set.
- Evaluation: Evaluate the effectiveness of each model, considering computational power, time, etc. Conduct a comparative analysis between deep learning, classic machine learning, and a combination of the two.

Methodology

Data Collection/Exploration

Any task in Big Data requires one very important element. Data. The MAFA dataset [2] contains carefully curated training images with labels that we will be using to train our models…Then we'll use the testing images to validate our data. In order to explore and properly understand the data, we need to extract the labels given to each image…This was done using the scipy.io

package to extract the data from the MATLAB file into a csv, structuring it into useable data. To ensure that we were indeed able to access the data I went ahead and visualized it, including its associated labels.

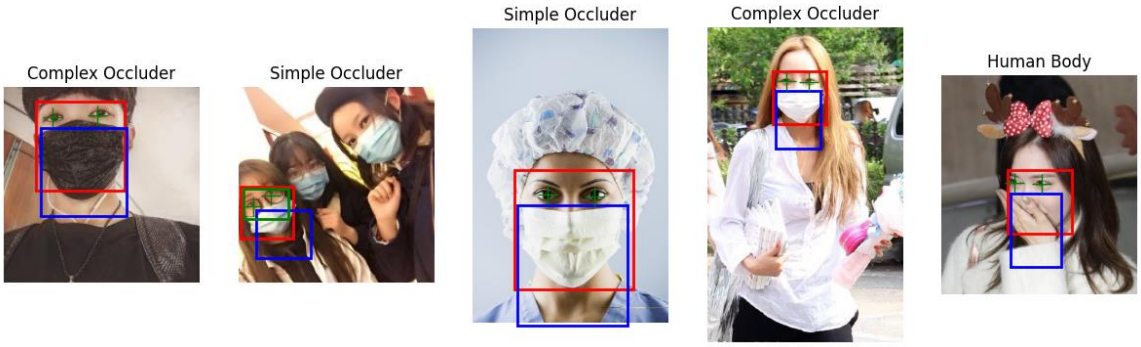

Figure 1 - Basic Label Visualization outlining each images labels

This was where I ran into a slight problem that I didn't discover until later down the line…I hadn't realized that multiple faces were included in the training set because I had ditched the MATLAB file and opted for the csv, I was missing the labels for the faces where they are multiple people. Here is the corrected result:

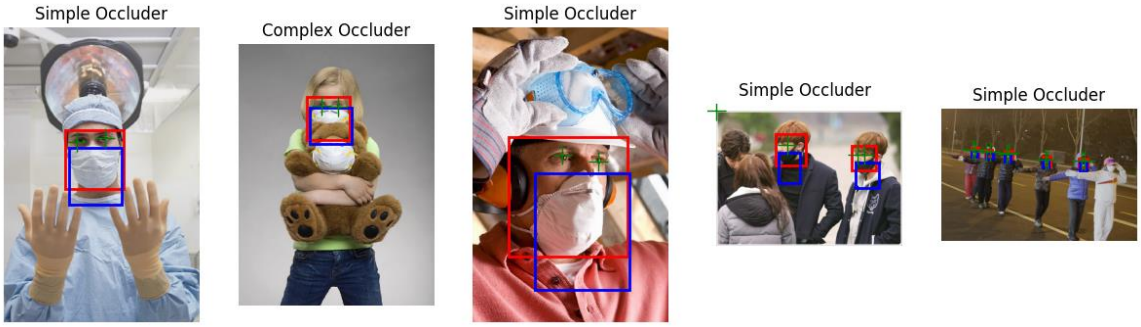

Figure 2 - Basic Label Visualization, accounting for above error

Data Preprocessing

Image Resizing

Image resizing plays a crucial role in maintaining data consistency by ensuring uniformity across all data processing tasks as well as enhancing performance for operations such as feature extraction.

Figure 3 – Image Resizing

Image Augmentation

Image augmentation involves altering images to create variations, enhancing dataset diversity for robust model training. It's crucial for improving model generalization, mitigating overfitting, and ensuring effective data utilization in machine learning tasks.

Figure 4 - Image Augmentation for more accurate training

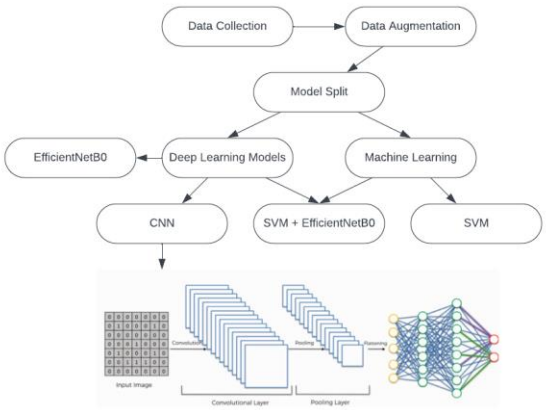

Figure 5 - Flow chart of approach

This details the methodology to create the model. Most of the steps can be consolidated for all modes i.e. Collection, dimensionality reduction, augmentation, and structuring. Splitting of the training data can also be done before training of the actual model. Whilst it may seem unnecessary, all of these step's lead to a more efficient experience when building your models (lower training and testing time.)

Model Selection

Deciding on the best model is no easy feat. In this case, a few deep learning techniques and one classic machine learning technique were used. A simple CNN was the first choice due to its ability to effectively capture spatial patterns within images (facial regions or Occluder regions).

Next, pretrained CNNs were used, leveraging transfer learning. Initially, EfficientNetB0 was chosen for its computational efficiency compared to models like ResNet or VGG. However, after encountering overfitting issues, the popular VGG16 and VGG19 models were explored, yielding noticeably more promising results. The VGG models' deep architectures with stacked convolutional layers enabled the extraction of increasingly complex and abstract representations, facilitating better discrimination between masked and unmasked faces.

An SVM (Support Vector Machine) with handcrafted feature extraction was also employed, known for its interpretability for datasets with many features. And finally, a hybrid approach combining EfficientNetB0 and SVM was explored. The handcrafted features extracted by the

SVM provide interpretable image characteristics, while the pre-trained CNNs provide more complex and abstract features from ImageNet, potentially improving classification performance.

Model Rundown

There are a few key notes before I detail each individual model…Each model was trained with image size of 100 by 100 to keep the amount of memory used within a reasonable range, in hindsight, maybe decreasing the number of images may have served more effectively. The loss function was kept constant throughout (binary_crossentropy).

I. CNN

A convolutional neural network was independently trained using labels to outline the face and Occluder regions. The model underwent a 10-epoch training cycle with 3 convolutional layers using ReLU activation, succeeded by max-pooling layers. After flattening the data, it was passed through two dense layers with a dropout layer (rate 0.5) to mitigate overfitting. The output layer consisted of two nodes with SoftMax activation for binary classification. The Adam optimizer and binary cross-entropy loss function were used.

II. Pre-Trained CNN, EfficientNetB0 [5]

The training spanned 10 epochs, leveraging the pre-trained EfficientNetB0 model with transfer learning. Custom layers were added for binary classification, using the Adam optimizer and binary cross-entropy loss function.

Convergence in accuracy was observed in both deep learning models, suggesting overfitting and the need for alternative approaches like data rework or fine-tuning. To combat this, I decided to look at some different pretrained models further from efficientNetB0

III. Pre-Trained CNN, VGG16 & VGG19 [6]

Altering the pre-trained model aimed to address the overfitting issue. The VGG models proved successful, with VGG19 achieving comparable performance to VGG16 in fewer training cycles (20 vs. 50 respectively), indicating its efficiency in learning relevant features for mask detection.

The VGG models' success can be attributed to their ability to leverage transfer learning from pre-trained weights on large datasets (ImageNet). By freezing the base layers and fine-tuning the top layers, the models effectively captured and adapted the learned features to the mask detection domain. The addition of custom layers, including dense layers and dropout, allowed for effective feature combination and regularization, leading to improved generalization and less to overfitting.

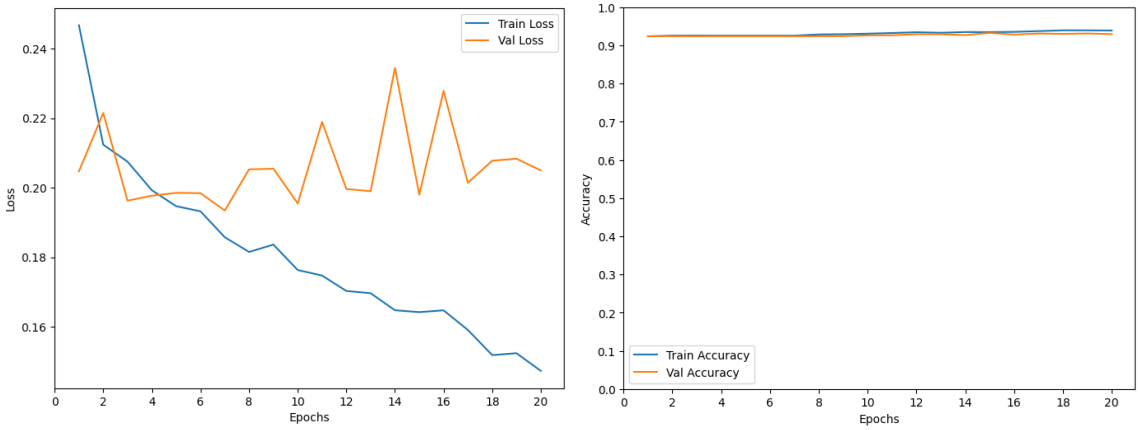

Figure 6 - Training Visualization

IV. SVM with handcrafted features

A Support Vector Machine (SVM) model with handcrafted feature extraction (color histograms and texture features) was trained to classify images. The SVM used a linear kernel and learned to distinguish between masked and unmasked faces based on the extracted features. Training iterations were capped at 1000 due to slow training speed. Uses imagenet as a weight.

V. Hybrid Approach – SVM + One of the pretrained CNN (EfficientNetB0) [7]

The pre-trained CNN served as a feature extractor, capturing representations of image features. These features were then fed into the SVM classifier, which learned to distinguish between masked and unmasked faces. This hybrid approach leveraged the power of deep learning through transfer learning, potentially capturing more nuanced features relevant to mask detection.

Analysis

The models achieved the following performance:

Model	Accuracy	Precision	Recall	F1-Score
CNN	0.924	0.924	1.0	0.96
SVM	0.867	0.924	0.932	0.928
EfficientNetB0	0.924	0.924	1.0	0.96
VGG16	0.936	0.946	0.988	0.966
VGG19	0.934	0.94	0.99	0.965
SVM + EfficientNetB0	0.921	0.924	0.996	0.959

While the masked face detection task proved challenging, the models explored in this work demonstrated promising results. However, there were certain limitations and areas for improvement. One major issue encountered was overfitting, as evidenced by the convergence in accuracy observed in the initial CNN and EfficientNetB0 models. This suggests that further training may not lead to significant performance gains, and alternative approaches such as data rework or fine-tuning are necessary. The transition to deeper architectures like VGG16 and VGG19 helped mitigate this issue, likely due to their ability to capture more complex and abstract representations from the input images.

It's worth noting that the VGG19 model, despite being trained for only 20 epochs, achieved comparable performance to VGG16 trained for 50 epochs. This indicates VGG19's efficiency in learning relevant features for mask detection, potentially due to its deeper architecture and increased capacity to extract intricate patterns. Both had a default learning rate of 0.001. There were time constraints with training but If I were to reattempt, I would run them for the same number of cycles to get a representative result.

The SVM with handcrafted features performed reasonably well, with an accuracy of 0.867. However, lagged behind the deep learning models, which could be attributed to the limitations of handcrafted features in capturing complex representations compared to the learned features from CNNs. The hybrid approach combining EfficientNetB0 & SVM showed promising results, with an accuracy of 0.921 and F1-score of 0.959. This approach aimed to leverage the strengths of deep learning & classic machine learning techniques, with the handcrafted features providing interpretable image characteristics and the pre-trained CNNs contributing more complex and abstract features. The results seemed to even out between the two models as expected.

Regarding computational complexity, it's reasonable to assume that the deeper VGG models and hybrid approaches would be more computationally intensive compared to the simpler CNN and SVM models. This trade-off between model complexity and performance is a common consideration in real-world applications thus the caps on things such as training times and cycles. Potential area for improvement could be exploring more advanced data augmentation techniques beyond simple rotations and blurring or increasing training time/resources.

Overall, the analysis suggests that while deep learning models like VGG19 showed promising performance. Its deeper architecture and ability to capture intricate features through transfer learning from ImageNet proved advantageous for this problem. It's worth noting that the VGG19 model was only trained for 20 epochs, indicating its efficiency in learning relevant features compared to VGG16, which required 50 epochs to achieve slightly higher performance.

If I were to attempt this task again, I might investigate transfer learning a little more and possibly try to utilize the WIDER Face dataset to create a more in-depth version of the human face before identifying whether they are wearing a mask or not. I would consider fine-tuning the pre-trained models or exploring other methods that combine multiple models' predictions. Mixed model techniques could help mitigate individual model biases and potentially yield better performances.

Results

The following images represent the test outputs generated by the trained CNN model. It is evident that the model's accuracy is variable, with some predictions displaying low confidence or complete inaccuracy. These discrepancies may be attributed to factors such as the quality of the images or the extent to which the faces are obscured within the images...

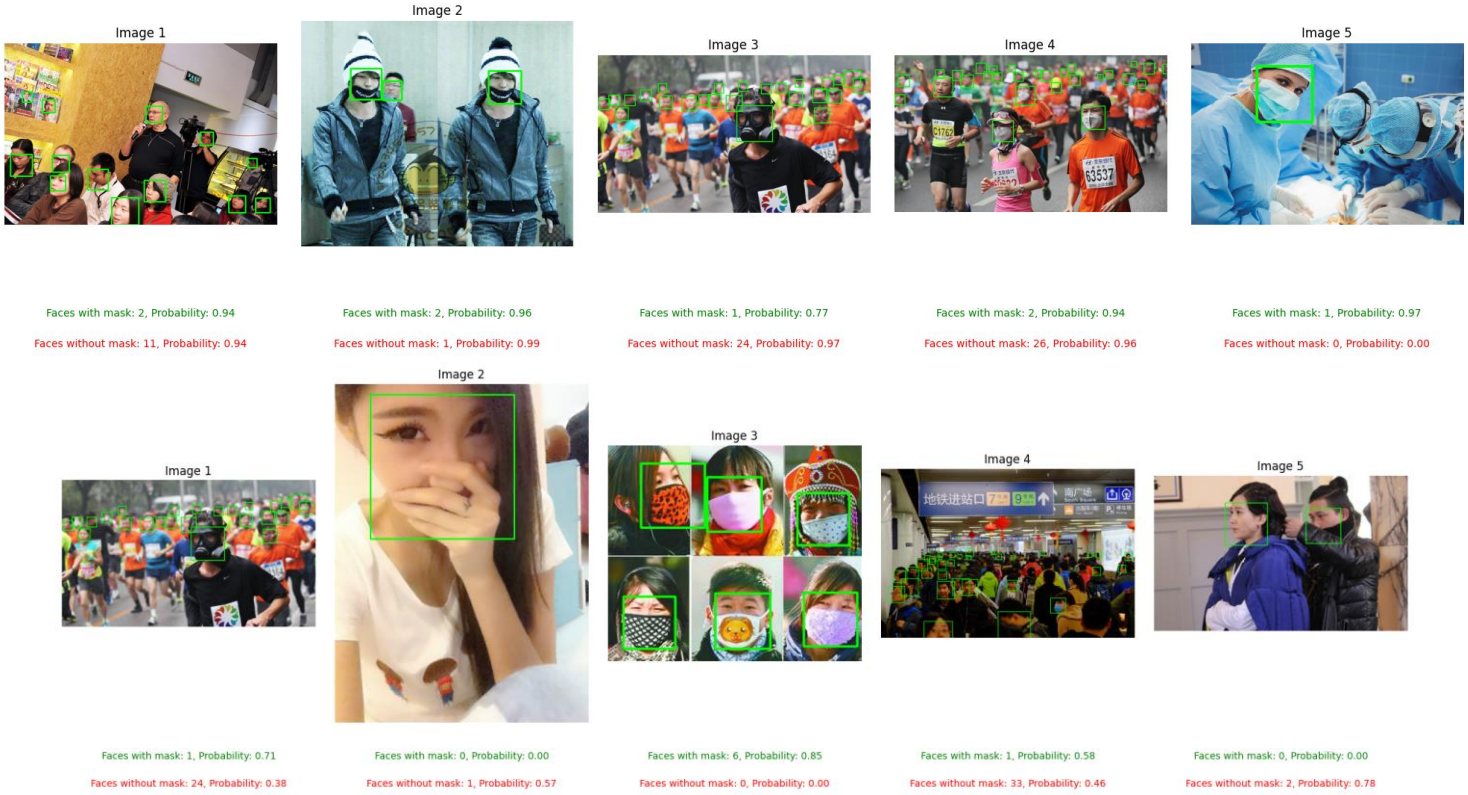

Figure 7 - Example Face Mask Detection

Movie Recommender

ALEX WOODROOF | UP2118496

Introduction

In 2023, it is estimated that over 200 billion hours of content were consumed, which is equivalent to 23,700,000 years of viewing time in a single year. This staggering consumption is partly due to the ability of industry giants such as Netflix [1] and YouTube to suggest content that appeals to specific viewers through automated recommender systems. The more interaction a user has, the more tailored the recommendation algorithm becomes.

The primary objective of this report is to delve into recommender systems and how content-based filtering works, along with an attached implementation. This analysis draws insights from the TMDB 5000 dataset [3], containing 5000 movies from The Movie Database (TMDB), which holds detailed information about movies such as crew, cast, budget, and more.

Understanding the Problem

A recommender system is a type of information filtering system that predicts the preferences or ratings a user would give to a particular item, such as a movie, song, or product. There are various content recommendation strategies, including collaborative filtering, content-based, context-based, knowledge-based, and hybrid recommender systems [2]. The challenge lies in determining which strategy will most likely provide the user with content they would enjoy.

Methodology

Data Collection / Exploration

For this task, I'll be using the TMDB dataset [3] which is a reliable and diverse dataset, consisting on movies and numerous features relating to it such as cost, cast, popularity.

Figure 8 - Budget vs Popularity vs Revenue

These scatter plots examine the relationships between a movie's budget, revenue, and popularity score. All plots show a positive correlation: higher budgets and revenues often lead to higher

popularity. Outliers suggest exceptional cases, and various factors such as genre and marketing may also influence results.

The image below represents the frequency occurrence of certain movies throughout the entire dataset…

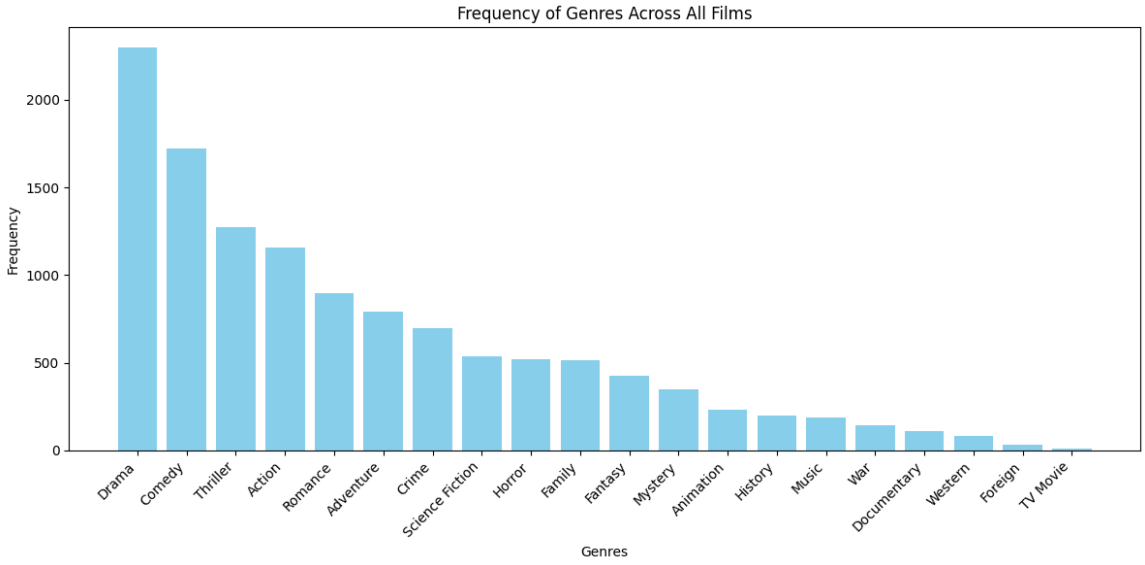

Figure 9 - Frequency of genres

Model Selection

Collaborative filtering is one of the most widely used strategies…this is a method in which content is recommended based on similar users' profiles. However, it is not without its flaws…it encounters a significant limitation when there is a lack of data for a user. Thus, companies such as Netflix will employ a hybrid model that make the most of both world's combining both content-based and collaborative filtering to create a weighted, switching, or mixed hybrid. These tend to be extremely complex to implement so our next ideal would be a collaborative filtering model but given limitations on the data we have available. We lack sufficient knowledge to begin recommending movies based on similar users, so we'll go with a content-based filtering method.

Content-based filtering recommends items by matching items the user has watched with similar items. There are both pros and cons to this strategy…on one hand, it is completely personal. It is purely based on the user's preference and only the users preferences, it cannot be tainted by the preferences of other users. It mitigates what is known as the cold start problem…where there is limited or no user data available for new users. But it is not without its drawbacks… it has limited diversity. Since it suggests items that are similar to previously watched items, the user is most likely being suggested films of a particular type, Hindering discovery of new content. [4]

Recommender Algorithm

With an abundance of movies available, determining the most suitable content to recommend to users poses a significant challenge. One approach to address this challenge is through the utilization of similarity equations, such as cosine similarity, to quantify the likeness between

movies based on their attributes. By leveraging such techniques, I can effectively recommend movies to users that align with their preferences and interests.

Cosine similarity [6] is favored over other methods due to its computational efficiency, scale invariance, and angle independence, making it ideal for comparing high-dimensional vectors like those found in documents or feature sets. Alternative methods include Euclidean distance, Jaccard similarity, or Pearson correlation coefficient, may be more suitable in certain contexts based on factors like data distribution, feature representation, or domain-specific considerations

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$

Figure 10 - Cosine Similarity Equation

Analysis

The code uses content-based filtering with TF-IDF (Term Frequency-Inverse Document Frequency) vectorization to convert movie overviews into numerical feature vectors for computing similarity. Scikit-learn's TfidfVectorizer creates a TF-IDF matrix, capturing word importance within overviews and rarity across the corpus. Cosine similarity measures the similarity between movies' TF-IDF vectors, and the function returns the top 10 most similar movies based on similarity scores.

This approach provides personalized, interpretable recommendations but may face limitations due to overspecialization and scalability issues. Combining content-based filtering with collaborative filtering and incorporating additional features like genres and cast may improve recommendation quality and diversity. We had to drop in applicable films such as pre-release or in production…

The images below represent two separate attempts at find similar movies…It appears that they offer good recommendation choices as you have a range of movies placed in similar categories.

```
get_cosine_recommendations('Toy Story')
```

	title	overview
42	Toy Story 3	Woody, Buzz, and the rest of Andy's toys haven...
343	Toy Story 2	Andy heads off to Cowboy Camp, leaving his toy...
1779	The 40 Year Old Virgin	Andy Stitzer has a pleasant life with a nice a...
2869	For Your Consideration	Three actors learn that their respective perfo...
891	Man on the Moon	A film about the life and career of the eccent...
3873	Class of 1984	Andy is a new teacher at a inner city high sch...
3379	Factory Girl	In the mid-1960s, wealthy debutant Edie Sedgwi...
3065	Heartbeeps	Heartbeeps stars Andy Kaufman and Bernadette P...
3383	Losin' It	Set in 1965, four Los Angeles school friends -...
2569	Match Point	Match Point is Woody Allen's satire of the Bri...

```
get_cosine_recommendations('Avatar')
```

	title	overview
3604	Apollo 18	Officially, Apollo 17 was the last manned miss...
2130	The American	Dispatched to a small Italian town to await fu...
634	The Matrix	Set in the 22nd century, The Matrix tells the ...
1341	The Inhabited Island	On the threshold of 22nd century, furrowing th...
529	Tears of the Sun	Navy SEAL Lieutenant A.K. Waters and his elite...
1610	Hanna	A 16-year-old girl raised by her father to be ...
311	The Adventures of Pluto Nash	The year is 2087, the setting is the moon. Plu...
847	Semi-Pro	Jackie Moon is the owner, promoter, coach, and...
775	Supernova	Set in the 22nd century, when a battered salva...
2628	Blood and Chocolate	A young teenage werewolf is torn between honor...

Figure 11 - Example Movie Recommendation

References

Face Mask Recognition detection:

[1] (Sakthimohan et al., 2022)

[2] (Mingxin Jin et al., 2023)

[3] (Ge et al., 2017)

[4] (Muhtahir O. Oloyede et al., 2020)

[5] (Dewan et al., 2023)

[6] (Pednekar et al., 2023)

[7] (Datt & Kukreja, 2022)

Movie Recommender System:

[1] (Netflix, January - June 2023), Source

[2] (Institute of Electrical and Electronics Engineers, n.d.)

[3] (Google, n.d.)

[4] (Thannimalai & Zhang, 2021)

[5] (F'unakoshi & Ohguro, n.d.)

[6] (Schwarz et al., 2017)