International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V72I8P107

Volume 72 Issue 8, 42-52, August 2024

Original Article

E-Stylist: A Machine Learning Aided Fashion Stylist

Priyank Singh[1], Nishtha Ahuja[2]

[1,2]*Rochester Institute of Technology, USA.*

[1]*Corresponding Author : erpriyanksingh@gmail.com*

Received: 10 June 2024 Revised: 15 July 2024 Accepted: 05 August 2024 Published: 29 August 2024

Abstract *- The common key to success in all sectors is perfect attire. The first thing a person showcases is his/her personality, a significant portion of which is taken by the attire. The art of dressing up in the right manner is not known to all. Not everyone is skilled to be a fashion stylist. But it is important that one has his/her style right to get recognized. This project uses machine learning to create an application that will act as a fashion stylist for the end user. The application takes as input the event the user wants to dress for and an image of the user. The image can be captured in real-time, or a pre-existing image can be fed to the application. The application performs feature extraction on the image and displays certain features as a result. These include gender, hair color, hair length, height, body shape, skin tone and the event for which the attire seems best suited. The system then recommends an image that suggests an attire the user can consider wearing for the kind of event he/she mentioned according to his/her body features. This recommendation is based on the features extracted from the user's image and the previous learnings of the model. The technologies used for the project are Python and TensorFlow.*

Keywords *- Machine Learning, E-Stylist, Fashion, Neural network, CNN.*

1. Introduction

Rachel Zoe says," Style is a way to say who you are without having to speak" [1]. A person's style says hundreds of words about her without actually saying anything. Having said that, not everyone knows the art of styling themselves, and not everyone can afford a personal stylist. The project's aim is to create a machine learning-based electronic personal stylist for daily use with just a few clicks. We present an application that is artificially engineered to suggest attire for a person based on factors such as the individual's body shape, hair length, the kind of event he/she wants to dress up for, etcetera. The application uses state-of-the-art machine learning techniques, including convolution neural networks, to reach its goal.

The approach used in the project to create all eight classification models is called Data driven approach in that the model is fed with training data that consists of pre-labeled images. The models look at these examples and learn the type and other details about the class. The hyperparameters of the models were tuned by dividing the training dataset further to obtain a set of validation data from it. The adjustment was made to ensure that the model was behaving well. Hyperparameters are the high-level concepts defined before the training process starts, such as setting up the learning rate or the number of layers. The subsequent chapters will discuss the adjustments made in the hyperparameters. The following sections will talk about a few concepts used in the creation of the application in detail.

1.1. Neural Network

A neural network is a biologically inspired system that tries to act like a human brain to perform activities like classifying an object [2]. It is a network or a graph of many neurons that are connected to each other in an acyclic fashion to avoid any infinite loop. The network is arranged in the form of various layers consisting of neurons. The neurons of one layer are connected to the ones of the next layer. However, the neurons in the same layer are not connected to each other. A layer in which all the neurons are connected to all the other neurons of the adjacent layers is called a fully connected layer. The architecture of a two-layered neural network is shown below.

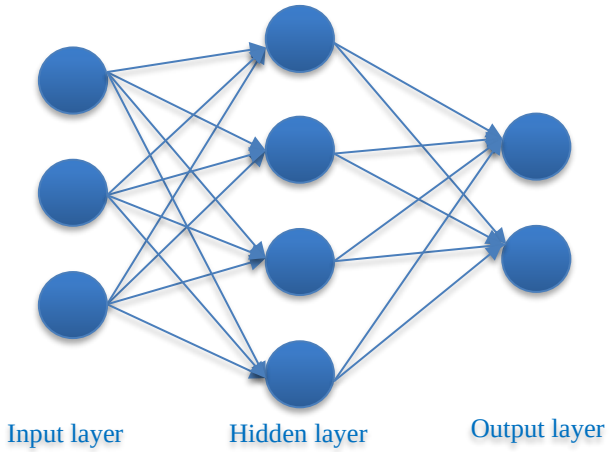

Fig. 1 Neural network architecture

A neural network is made up of an input layer, an output layer, and some hidden layers [3]. In Figure 1, the example has one hidden layer and (4+2) = 6 neurons. The input is not considered as a layer, and neither are the neurons it has, in the total count for neurons. The number of hidden layers in a neural network is variable. In Figure 1, there is only one hidden layer. There is a transfer of knowledge between the neurons of one layer to the other, just like the neurons in a brain. The number of layers varies from model to model. The more the number, the better the handling of complex data. But, a lot of layers can sometimes lead to overfitting of the model. Overfitting is a phenomenon in which a model is too fit to classify a particular type of data, thereby resulting in a biased prediction. Therefore, for a simpler dataset, fewer layers are preferred.

1.2. Convolutional Neural Network (CNN)

A CNN is a type of neural network with only some differences. The most important difference is that the input to a CNN is a dataset composed of images. The network assumes that the dataset will consist of images and is, therefore, prepared to handle three dimensions: width, height, and depth. The CNN architecture is shown in the Figure 2.

1.3. Inception Model

1.3.1. TensorFlow

TensorFlow is a platform to execute machine learning algorithms and visualize the results [14]. This project uses TensorFlow for computer vision and information retrieval. It provides a lot of APIs enabling the implementation of deep neural networks. The advantage of using TensorFlow is that the result is device-independent, making it flexible [14].

1.3.2. Inception in TensorFlow

This project uses Inception-v3 [17] as the base to build the application. Inception is a software based on convolutional neural networks and uses TensorFlow. This model has been trained on the ImageNet [15] dataset and is available for research and development.

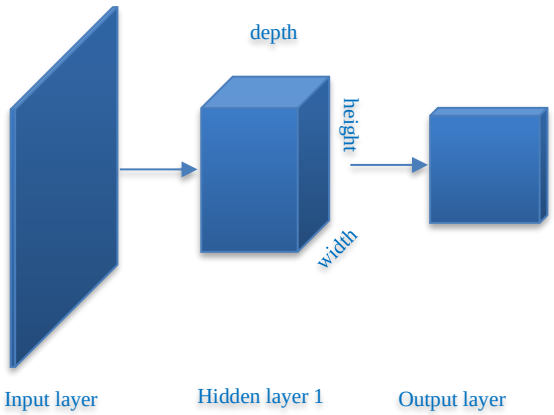

Fig. 2 CNN architecture

The project uses the Inception model for development because of its low computation cost and better performance [17], making it the best fit. The details about the changes made to the inception model for this project are discussed in Chapter 2.

1.4. Kivy

The UI of the project has been designed using Kivy [13] which is an open source software. Kivy allows the same application to be deployed on three platforms, viz. Desktop, Android, and IOS. This project uses the services for the Desktop version of the software. Additionally, Kivy provides a broad range of tools to access the hardware of the device and various support libraries to make the user interface appear fancy.

1.5. Dataset

The image dataset used in the project development has been collected from Liu et al. [16]. The authors provide a rich dataset comprising 800,000 images which are highly annotated and are taken against a variety of backgrounds. The project uses some of these images for feature detection of clothes worn by humans against a plain and bright background. Approximately 1000 images were used from this dataset to extract the necessary information. The techniques used to obtain this information are described above.

1.6. Related Work

Machine learning has impacted fashion in many ways. Companies like Amazon are using recommendation systems to suggest outfits to users based on their search history. They train their models according to the user requirements. But the field still has a way to go. The following are some of the inspirations for our work.

1.6.1. Style-Me

Style-Me [18] is an application that uses AI techniques to create a fashion stylist. The authors use a score-based method to rank nearly 500 looks. The dataset created for this application consists of" *32 dresses and 20 shoes for 4 different events"* [18]. The application is based on the user's perspective, and the scores are adjusted accordingly. The authors have trained the models using Artificial Neural Network [18]. The system architecture of Style-Me consists of 5 elements, viz." *an Initialization program, Database, Style Engine, Learning Components and User Interface"* [18]. The application starts with the initialization program, which takes input from the user in the form of a quiz. The quiz consists of 8 questions, the result of which is a style out of the six predefined styles in the application, namely, Classic, Dramatic, Gamin, Ingenue, Natural, and Romantic. Every style has its database which is accessed once the style is obtained from the user's quiz results. The database consists of various tables, some of which include information related to the clothing, and one table consists of a view. The view stores

information obtained by querying multiple tables to keep them in one place. The output of the quiz is also fed to the Style Engine, which pairs different clothing attributes, such as dresses and shoes and stores them in a new table in the database. Learning components comprise an artificial neural network that is trained on the dataset obtained from the database portion of the architecture. The output of the learning components is a score indicating how good the newly designed look is. The authors performed five different experiments to adjust the hyperparameters such that the correlation coefficient with the item's attribute is the maximum. The final choice included using a Multilayer Perceptron classifier with one hidden layer consisting of 20 hidden units at a learning rate of 0.3 and a momentum term of 0.1. Momentum is used to speed up the process and improve the performance.

1.6.2. Google Muze

Google, in collaboration with Zalando and StinkDigital, proposed a design engine that is capable of creating an outfit for a particular person [10]. The person acts as a muse for the neural network, which has been trained already. The system asks the user some questions about their preferences and recommends fashion looks tailor-made for them. The dataset used to train the model was created by 600 fashion stylists [10]. The design is built on TensorFlow. However, the results of the project are not compelling and need a lot of improvements before the application can be used in real-time as a fashion stylist [6].

2. Design

The project's aim is to create a computer vision-aided application that acts as a personal stylist for a human being. The project has two parts: extract and display certain features of a person based on their image taken in real-time or fed directly to the application, and recommend an outfit to the user as per their features and the kind of event that they selected. The application takes a picture of the user and asks him/her about the kind of event for which he/she would like to dress up. After analyzing the image and the event, the application recommends an outfit to the user. The following subsections explain the architecture of the application and the working of all the components.

2.1. Architecture

The system design of the project is shown in Figure 3. It consists of three components that are explained in the following sections.

2.1.1. Input

The first screen of the application accesses the camera of the device and takes a picture of the user. The image is obtained by taking a screenshot of the application screen. The image thus obtained is cropped to extract the central part to help increase the performance of the models. For example, a test image of a user capturing the application is shown in Figure 4. The image cropped from the original image and fed to the models is shown in Figure 5.

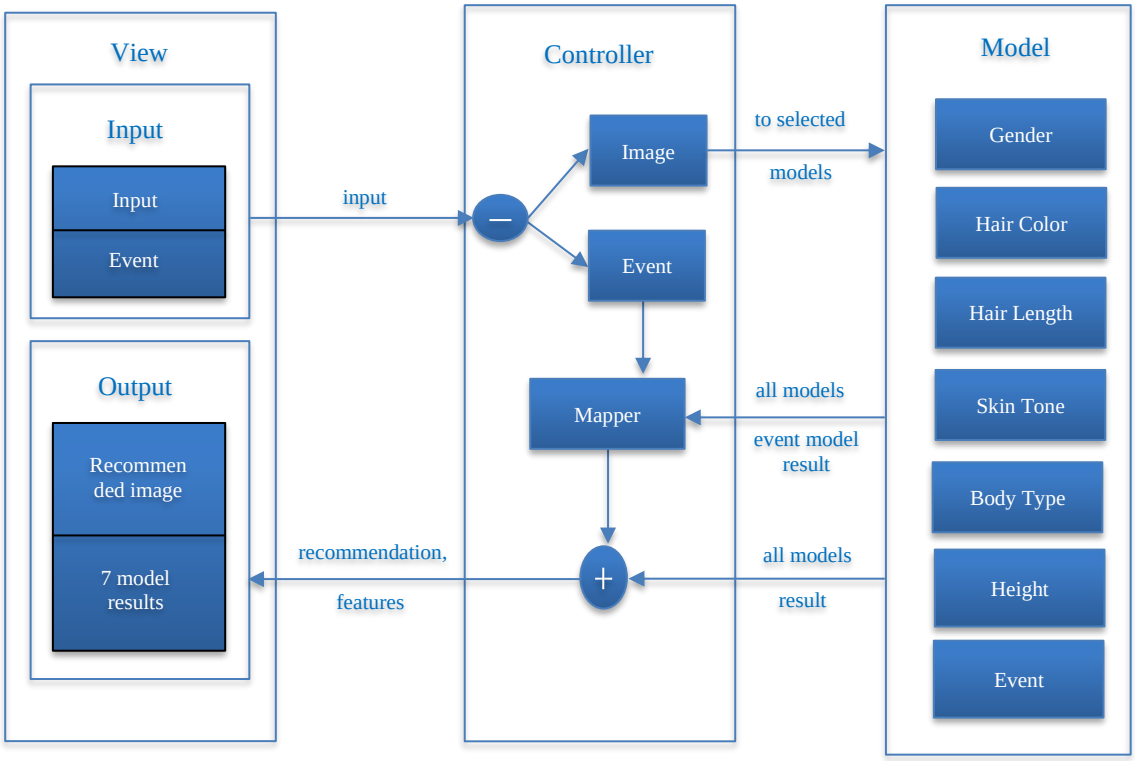

Fig. 3 System design

The processed images are fed to all the models, which have been pre-trained, to obtain the features of a person, including the' event' for which the person seems to be dressed up. A second input taken from the user is the actual occasion for which he/she wants to dress up. This value is used to recommend an outfit to the user that he/she might consider wearing.

2.1.2. Models

There are eight models in the project, one for gender, hair length, hair color, skin tone, men's body type, women's body type and event each. The models are created using the inception model, as mentioned in subsection 1.3.2. The architecture of the inception model is given in the figure 6. The inception model is trained on the ImageNet dataset [15]. This model could work well for our project only if it was trained on our dataset so that it could learn various features and aspects of the new images. The dataset used for the same has been mentioned in detail in section 2.2. The deep network of the inception model consists of 22 layers. The last layer of this model was retrained using the dataset for each of the new models. The model training will be described in Chapter 3. Once all eight models were re-trained, the testing was performed on the testing dataset to obtain the accuracy of the model. The final models were then used in the application to operate in real time. The input image from the user was fed to the' gender' model first to obtain the gender of the person. The value of the gender helped in determining the body type model to be used, i.e., men's body type for a male and women's body type for a female. The image was then fed to all seven models, and the results of the same were displayed on the output screen for the user's information.

2.1.3. Mapping

The output from the seven models was fed to the mapper to find the data that resembles the features of the user the most. This process was done in the following seven steps.

The data was filtered according to the *gender* of the user. This filtration reduced the mapping data. The next filter applied was that of the *event*. The user gave as input an image and an event. It is important that the outfit suggested to the user be well suited for the kind of event the user chooses. Therefore, the event filter had to be the second one. The remaining data was filtered according to the user's *body type*. The body type was given preference because the body type of a person is crucial in determining the kind of clothes they should be wearing. In the case of a rollback from any further steps, it would be safe to determine attire based on the body type of a person. [7] says that a woman with a rectangular shape should try to wear a long jacket as it makes her look lean. It is well proven that the outfit worn according to the body type is the one that fits the best. The body shape filter was followed by the *height* filter. The height of a person plays a vital role in the outfit determination. For example, as [12] says, shorter men should go for vertical stripes in their outfits. It does not just look appealing but also makes one seem taller. After filtering the data with the height, the *hair color* filter was applied. The reason behind choosing this feature was the vast number of categories in it, precisely 24. The number of images distributed in each category was very less. Therefore, the dataset was reduced massively. The next filter was chosen to be the skin *tone* of the person. Skin tone is not a predominant factor, but it can sometimes determine the type of colors one should prefer to magnify or suppress a bold look. The last filter applied was that of *hair length* because hair length is not an important factor in determining the outfit. At any step, if the number of results was empty, the mapper was rolled back to the previous step. The rollback was done assuming that the order chosen for the filters was the best fit for every case.

2.1.4. Output

The output is shown to the user in two forms: their body features and an outfit recommended as per their event request and their body features. Figure 7 shows a sample output screen. It consists of the cropped input image that was fed to all the models, the various features displayed in the form of a list and an image of the recommended outfit per the event the user selected.

2.2. Data Collection and Preprocessing

The image dataset used in the project has been obtained from Liu et al. [16]. This dataset consists of 800,000 images of models posing in different attires. There are 50 categories of labels for all the images and nearly 1000 attributes which leaves no room for redundancy and helps better the learning of the model. The images are annotated to give a diverse range of information about the outfit. The authors have created four benchmarks called" Attribute Prediction, Consumer-to-shop Clothes Retrieval, In-shop Clothes Retrieval, and Landmark Detection" [16]. The work is available to everyone for further enhancement. The seven models created for the project required training. The information needed to train the models was the image and the ground truth value of the feature the model was representing for that image. The dataset obtained from Liu et al. consisted of the images, but the ground truth value was not available which was the limitation in the dataset. There were two options for obtaining the ground truth values: asking a human to determine it or asking a machine to identify it. There are certain built-in libraries in computer vision to recognize values of features like the eye color of the person in a picture. However, these libraries are not available for all the features that were required for the project.

Moreover, the accuracies of these libraries are not 100% implying that the ground truth values will not be 100% accurate. That said, if a human determines the value, such as the hair color of a person in a picture, he/she may not be 100% correct either because every human perceives things differently than any other human. Therefore, the accuracy of

the model trained with a dataset for which the ground truth has been determined by a human may not be 100%, but it is still more reliable than a machine, given the fact that machines have yet to match the accuracy of a human brain by many folds. The ground truth values were obtained from humans in two different forms. One set of data was obtained in the form of a survey, and another set was created by us. The survey was created using the software Qualtrics courtesy of the Rochester Institute of Technology [11]. There were two different surveys, one for men to take [4] and another for women to take [9]. The surveys had five images each and six questions about the images, such as the body type and height of the person in the image. Adding questions to the Qualtrics software was an uncomplicated task, but adding a new image in each section required tweaking in the Javascript. These images were generated and added randomly to the page so that the same image was not analyzed again and again. At the end of the survey, a few questions about the survey taker were asked to gather information about their features. This information was taken with the initial idea of testing the model using these.

However, the idea was later dropped, and the information was not used anywhere in the project. A total of 195 people took the survey, out of which 121 were females and 74 were males, leading to a total of 975 data records. The second set of the dataset was created by us manually by filling out the ground truth values of the features. It comprised a total of 122 records. The dataset obtained by both the methods described above was not enough to train the models. Each model had a set of categories into which the dataset was divided, such as six categories for skin tones. The number of images left in each category was very few at the end of the distribution process and did not meet the minimum requirement of the Inception model. It required that each folder had at least 20 images in it to train the model. Therefore, the dataset needed augmentation. The number of records was increased 11 times by augmenting data in 11 different ways. These included flipping horizontally, rotating, blurring, zooming, adding random noise, adding salt and pepper noise, swirling, affining, contrasting, increasing intensity, decreasing intensity and adding histogram equalization to the images. The new dataset comprised 10,725 images for the survey dataset and 1,342 images for the dataset created by us. However, the ground truth values of the augmented images were the same as the original images.

The final dataset was split into three categories: Training, Validation, and Testing in the ratio 80:10:10. This split was done by the inception model. The training dataset is the subset that is used to train the models. The models learn trends and features from the training dataset so that they can classify new images later. The validation dataset is used to test the model during the training period to make sure that the model is doing good. If the accuracy during validation is substandard, the hyperparameters are adjusted to increase it. We faced a situation like this during the training process and had to make the necessary adjustments, which will be discussed in Chapter 3. If the validation accuracy is exceptionally high, it implies that the model is overfitting and the dataset needs to be changed accordingly. We did not face this issue during model training. Once the models were trained and validated, they were tested on the test dataset. The testing dataset has to be completely new to the models to avoid bias and obtain accurate results. The accuracy of both the models will be discussed in Chapter 4.

2.3. System Requirements

2.3.1. Hardware

Camera Requirements

The desktop application requires a camera in the device to capture the image of the user in real time. The camera used during the project experimentation was the inbuilt 720p camera of the MAC laptop.

Device Specifications

The device used for the project was a MAC laptop with MacOS Sierra, a 2.7 GHz Intel Core i5 processor consisting of 8 GB memory and 128 GB storage.

2.3.2. Software

Language

The programming for the application was done in Python version 2.7.11.

UI Framework

The Python framework used for the creation of UI is Kivy [13]. The system requires Kivy to run the application.

Model Framework

The Python framework used for the creation of neural network models for feature extraction is InceptionModel [8].

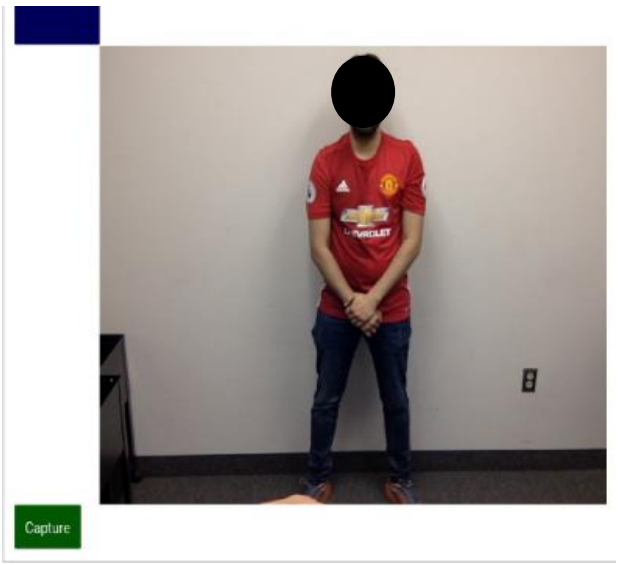

Fig. 4 Originally captured image

Fig. 5 Cropped image

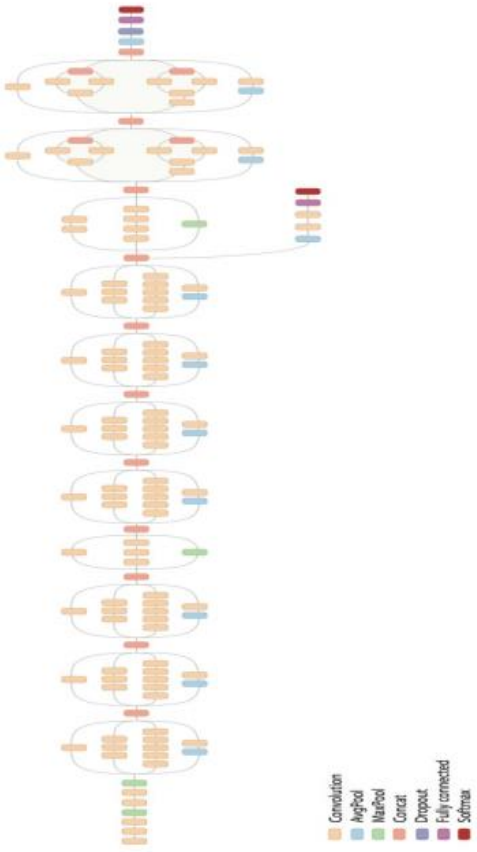

Fig. 6 Inception Model Architecture, Image Courtesy: [8]

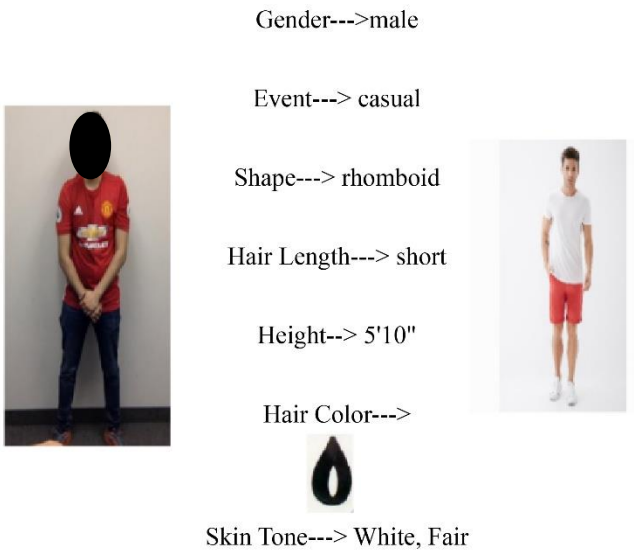

Fig. 7 Output screen sample

3. Implementation

The project has an MVC (Model-View-Controller) design. The user interface of the application consists of a screen which asks for text input from the user and an image of the user, as described in Section 2. The data is collected by the View part of the application and is given to the controller. The controller checks the image for the gender of the person and passes it to the appropriate models. For example, if the gender is male, the image will be passed to the men-body-type model and not the women-body-type model. The models process the results and give them to the controller. The controller finds the most appropriate result and sends it to the view to display to the user.

3.1. View

The UI of the application is created using a library of python called Kivy [13]. Kivy offers hardware support over three platforms: desktop computer, Android, and IOS. It has many widgets for the construction of an application, such as an input box, dropdowns, images, buttons, etcetera. Three of the many widgets were used in the application UI, viz. camera, button, and dropbox. The dropbox widget was used to offer a list of options for the events from which the user could choose one. The camera widget was used to access the camera of the device on which the application was running. Once the camera was active, a screenshot of the screen was taken using the button widget and the screenshot feature of the window provided by Kivy. The event and the image were passed to the controller for further processing.

3.2. Controller

The controller took the image and passed it to the appropriate models as mentioned before. The output of all the models was displayed to the user. Based on the output of the Hair Color model, an image of the hair color was passed to the view to display to the user. The image of the hair color

was displayed to help the user understand the color, which becomes very difficult with just the name of the color, given the vast variety of hair colors in the classification. The controller also acted as the mapper for the second part of the application. The second part of the application was a recommendation system where an image was presented to the user as a recommendation of what he/she could wear based on the event he/she provided. The controller chose an image from the database that resembled the features of the user the most and passed it to the view. The matching was done by the process described in the subsection 2.1.3.

3.3. Models

The Inception model has been used as the base model to create eight different models to extract various features of a person in an image. The last layer of the inception model was retrained on each dataset for each model. The training process was iterative, as the hyperparameters needed adjustments because of low accuracies. In the first training cycle, the learning rate was set to 0.01, and the number of iterations was set to 4000. These hyperparameters gave poor accuracies with the lowest accuracy being 34.3% for the height model. The learning rate was then reduced to 0.001, but the results became worse than for the previous learning rate. Therefore, the learning rate was kept stagnant at 0.01 for the rest of the experimentation. The number of iterations was then increased to 70000, 90000, 110000 and 120000. The best results were obtained from the model with a learning rate of 0.01 and a number of iterations of 120,000. Figures 8, 9 and 10 show the decrease in entropy with the increase in the number of iterations. Entropy, in physics, refers to randomness. But in terms of big data, entropy refers to unpredictability [5]. It is a measure of determining the amount of useful information in the data. The higher the entropy, the more unpredictable the data is. Therefore, it can be inferred that the value of a hyperparameter that leads to lower entropy must be preferred. It can be seen that the entropy for the 120,000 iterations is almost 0. On the contrary, the entropy for 70,000 iterations is almost 0.5, and that for 90,000 iterations is almost 0.25. Therefore, it can be concluded that 1,20,000 iterations were the best choice for the project.

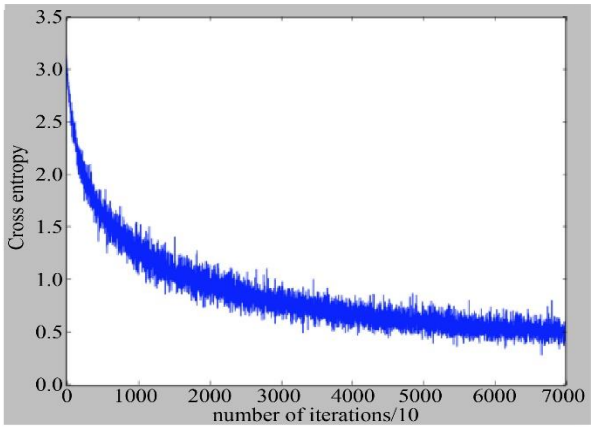

Fig. 8 Cross entropy for 70,000 iterations

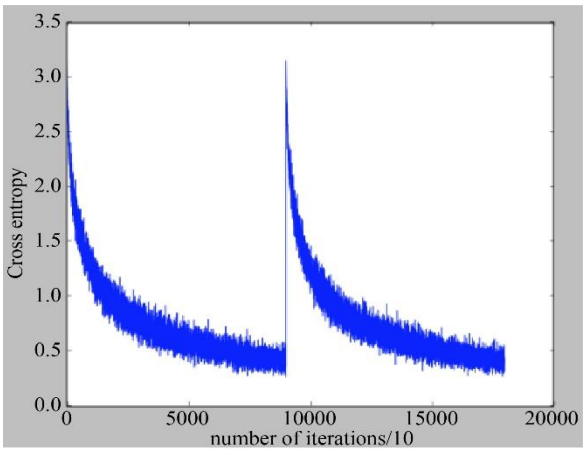

Fig. 9 Cross entropy for 90,000 iterations

4. Analysis

4.1. Experiments

The application was tested in real-time on a few people with different features in front of the same background and the same lighting. Four of the subject results are discussed below. The ground truth values for the measurement of accuracy were obtained by asking for the true feature values of the test subjects. The results of the questionnaire were compared to those obtained from our application. The accuracies are listed in Table 1.

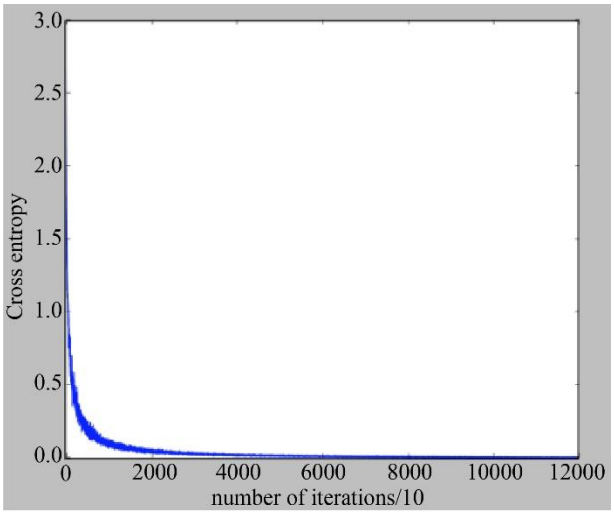

Fig. 10 Cross entropy for 1,20,000 iterations

4.1.1. Subject 1

Figure 11 shows the results for subject 1. It can be seen from Table 1 that the combined accuracy of the models is 42.86%. The correctly identified features were Gender, Hair length and skin tone. The result of the Body Type feature was' rhomboid' which is very close to the truth value' inverted triangle'. In the image fed to the model, as shown in the left image of Figure 11, the shoulders seem a little broad because of the pose. Therefore, the model confused it with the rhomboid shape.

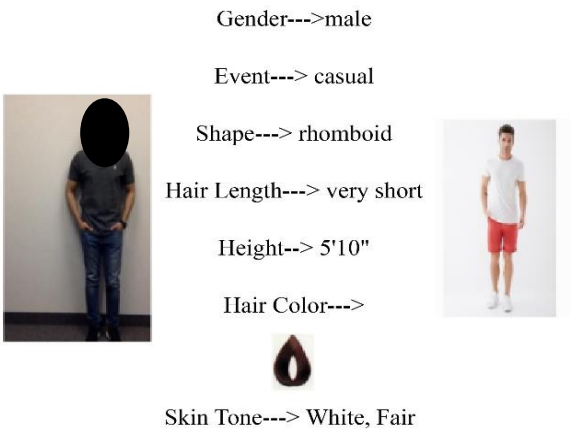

Fig. 11 Application results on subject 1

4.1.2. Subject 2

Figure 12 shows the results for subject 2. It can be seen from Table 1 that the combined accuracy of the models was 28.57%. The correctly identified features were gender and skin tone. The subject's body shape is oval, but in the image, it looks like a triangle, confusing the model. The hair length of the user is long.

In the image, the hair is visible as growing upwards. But for the model, long hair length implies that the hair is growing downwards beyond the head.

Fig. 12 Application results on subject 2

4.1.3. Subject 3

Figure 13 shows the results for subject 3. It can be seen from Table 1 that the combined accuracy of the models is 57.14%. The correctly identified features are Gender, Body Shape, Hair length, and Event.

The skin tone of the subject is very light. But in the image, it seems a little darker because of the poor lighting. Hence, the incorrect skin tone by the feature. Although the value is given by the model was white and fair, which is very close to the actual value.

Fig. 13 Application results on subject 3

4.1.4. Subject 4

Figure 14 shows the results for subject 4. It can be seen from Table 1 that the combined accuracy of the models is 28.57%. The correctly identified features are Gender, Body Shape, and Hair Color. The hair length of the subject is very long. However the hair is not visible in the image because of occlusion by the black clothes. It can be seen from the left image in Figure 14 that the hair is barely visible in the dark clothes. Therefore, the model could not recognize it and produced poor results Lighting is also the reason behind the incorrect skin tone. The model classified the subject as very dark because of the dark surroundings. In fact, the image, on the whole, is very dark because of the dark colored clothes. The height of all the subjects was incorrectly classified by the application. The most important reason behind the poor classification is the image aspect ratio, i.e. the ratio of the width and height and the closeness of the person to the camera. The closer the person, the bigger he/she seems. Another possible reason is described in the subsection 4.

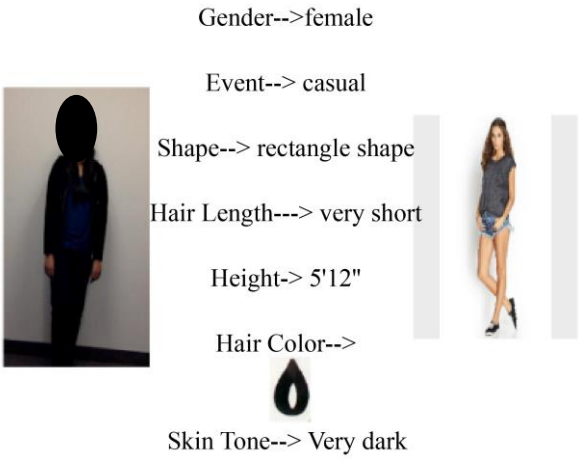

Fig. 14 Application results on subject 4

4.2. Accuracy Analysis

The test accuracies of the models on the two categories of datasets chosen to train and test the model are listed in Table 2. It can be seen from Table 2 that the accuracies of the models trained using the dataset we created are way better than those of the ones trained with the dataset obtained from the survey. The most probable reason behind this is the diversity in the nature of the survey takers. Every person perceives the contents of an image differently. Therefore, the dataset varies so much. A similar body structure might be classified into two different.

Table 1. Real-time test results

Subject	Correctly identified features (out of 7)	Accuracy
Subject 1	3	42.86%
Subject 2	2	28.57%
Subject 3	4	57.14%
Subject 4	3	28.57%

Table 2. Model accuraciesbody types by two survey takers

Model	Survey data	Self-created data
Gender	100%	100%
Hair Color	53.7%	78.7%
Hair Length	65.1%	83.1%
Height	50.8%	77.8%
Men Shape	53.7%	78.3%
Women Shape	64.1%	84%
Skin Tone	60.4%	84.2%
Event	54.8%	87.1%

These perceptions leave the model confused. The model learns trends from the dataset. It considers the data value as the ground truth for the image. In a situation where there is a conflict in the decision of two records, the model learns all the decisions and is rendered confused in the end, thereby leading to false results. On the contrary, the dataset composed of a single person has one point of view.

The model learns that point of view and produces results accordingly. The proof of this statement is the better accuracy in the case of the self-created dataset. The last layer of the inception model was re-trained on the dataset for each model. The models had particular concerns which were addressed separately for all. Some of those are listed below.

The height model is a scaled variant. The output of the model is dependent on how much frame space the person takes and not on his/her real height. The same person, when moved closer to the camera, is shown taller. Tests were conducted on different subjects using the same location and the same distance between the person and the camera to avoid any discrepancy caused by the scale variance, and the results were recorded. The hair color model gives poor results because of two reasons.

The first reason is the vast number of classes. The model classifies the image in one of the 24 color classes. Therefore, the training data is very less for each class, which is the reason behind the poor training of the model. The second reason is occlusion. The part of the body containing hair is much less as compared to the rest of the body, especially for' very short' and' short' hair. Therefore, the object of interest, in this case, the hair color, get occluded, thereby leading to poor image classification.

The skin tone model results are highly dependent on illumination conditions. The model gives different results for the same person tested under different light conditions. It is important that the user wears clothes with colors which contrast with the background, as this helps in avoiding any background clutter.

5. Conclusion

5.1. Limitations

The project has certain limitations, which will be dealt with in the later versions of the project. The restrictions are listed below:

5.1.1. Dataset Quantity Restrictions

The dataset used for the project is not sufficient to produce excellent results. The models used to obtain the features of a person in an image require a lot of images to learn enough to be able to classify a new image. But, the ground truth values of these datasets are not available and are required to be produced manually thereby restricting the quantity of data used for the project.

5.1.2. Dataset Collection Restrictions

The ground truth values of the features obtained from the survey have a sense of heterogeneity. The reason behind this is the difference in the view of every person filling the survey. Calibration can be done to resolve this issue, by giving an example of how to fill the survey to the user so that they know what is expected of them.

5.1.3. Input Image Restrictions

The UI of the application takes as input an image of the user. The results produced by the application are best if the background of the image is clear, light and contrasting with the person in the image. Moreover, the person is required to stand in the center of the frame while clicking a picture because the image fed to the models is a cropped image that extracts the center part of the captured image.

5.1.4. Camera Restrictions

The camera used to take pictures for the experimentation during the project does not take pictures of good quality. The specifications have been discussed in section 2.3. The results of the models were affected by the poor quality of the images.

5.1.5. Application Platform Restrictions

Currently, the project runs as a desktop application. Therefore, the user needs to run the main Python file manually to run the application.

5.2. Current Status

The current project has two parts: identifying features of a person in an image and recommending an outfit to them based on their input for the event for which they would like to dress up. The project uses an image dataset to train a model for feature extraction. Eight models have been trained to identify the value of each feature. The results of these models are displayed to the user and then used as input for the second part of the application. The second part of the application maps the features of the person with the dataset and displays the image that matches the most and is suited for the kind of event the user chose initially. The application is running in real-time on a desktop computer. The accuracy of all the models is not very good, especially for the ones with a lot of classes, such as hair color. But some of the models give good results such as gender and body type.

5.3. Future Work

Future work includes creating Android and IOS applications for use on devices other than desktop computers. The applications can be created using the Kivy library used to create the user interface. The dataset used to train the models is not sufficient for good results. In the future, more datasets will be collected to get better accuracy. The project can be expanded by creating models which will classify the image on their own. Every event type has a certain set of rules. For example, for business casual attire, a woman could wear a short-sleeved top, a skirt and a pair of open-toe shoes. The dataset can be created using these rules. This process will again require human resources to classify the images according to the rules. But the accuracy will be better.

5.4. Lessons Learned

One of the valuable lessons learned during this project is that obtaining the appropriate dataset is a tough task. Cleaning and preparing the dataset is another arduous task. The more the dataset, the better the model learns and the better the results are. However, handling such a massive amount of data is not an easy task. It requires multi-core processors or more in-memory to be able to process the large data. During the experimentation phase of the project, the training of each model took an average of 5-6 hours. The hours increased as the size of the dataset increased. These hours multiplied by eight, and the number of times the training was done after adjusting the hyperparameters almost led to weeks of the training process. The time could have been reduced if the processing was fast. The lesson learned from this is that big data management requires better tools that can handle it well. Another important lesson is that machine learning is very powerful, but there is so much still left unexplored in this field.

References

[1] Yanan Liu et al., "Energy Consumption and Emission Mitigation Prediction Based on Data Center Traffic and PUE for Global Data Centers," *Global Energy Interconnection*, vol. 3, no. 3, pp. 272-282, 2020. [Crossref] [Google Scholar] [Publisher Link]

[2] Zhen Xiao, Weijia Song, and Qi Chen, "Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107-1117, 2013. [Crossref] [Google Scholar] [Publisher Link]

[3] Einollah Jafarnejad Ghomi, Amir Masoud Rahmani, and Nooruldeen Nasih Qader, "Load-Balancing Algorithms in Cloud Computing: A Survey," *Journal of Network and Computer Applications*, vol. 88, pp. 50-71, 2017. [Crossref] [Google Scholar] [Publisher Link]

[4] Y. H. H, and L. X. Zhang, "Energy-Efficient Load Balancing in Cloud Data Centers Using Decision Tree Algorithms," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 5, no. 1, pp. 1-12, 2016.

[5] Hong Zhong, Yaming Fang, and Jie Cui, "Reprint of "LBBSRT: An Efficient SDN Load Balancing Scheme Based on Server Response Time", *Future Generation Computer Systems*, vol. 80, pp. 409-416, 2018. [Crossref] [Google Scholar] [Publisher Link]

[6] X. Y. Y. Z. Y, and L. L. Chen, "An Intelligent Load Balancing Scheme for Cloud Data Centers Using AI-Based Prediction," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 9, no. 1, pp. 1-16, 2020.

[7] L. X. J. Z. Y, and L. L. Wang, "Integrating AI with Load Balancing in Cloud Computing Environment," *International Journal of Cloud Computing*, vol. 7, no. 2, pp. 112-127, 2018.

[8] Jaimeel M Shah et al., "Load Balancing in Cloud Computing: Methodological Survey on Different Types of Algorithm," *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, Tirunelveli, India, pp. 100-107, 2017. [Crossref] [Google Scholar] [Publisher Link]

[9] Valeria Cardellini, Michele Colajanni, and Philip S. Yu, "Dynamic Load Balancing on Web-Server Systems," *IEEE Internet Computing*, vol. 3, no. 3, pp. 28-39, 1999. [Crossref] [Google Scholar] [Publisher Link]

[10] J. W. Y. W. H, and Z. W. Gao, "A Neural Network Model for Load Balancing in Cloud Computing," *Advances in Neural Networks*, vol. 10, no. 1, pp. 205-210, 2014.

[11] Akshat Verma, Puneet Ahuja, and Anindya Neogi, "pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems," *ACM/IFIP/USENIX 9th International Middleware Conference Leuven*, Belgium, pp. 243–264, 2008, vol 5346. [Crossref] [Google Scholar] [Publisher Link]

[12] Rajkumar Buyya et al., "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599-616, 2009. [Crossref] [Google Scholar] [Publisher Link]

[13] Anton Beloglazov et al., "A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems," *Advances in Computers*, vol. 82, pp. 47-111, 2011. [Crossref] [Google Scholar] [Publisher Link]

[14] Yuang Jiang et al., "Resource Allocation in Data Centers Using Fast Reinforcement Learning Algorithms," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4576-4588, 2021. [Crossref] [Google Scholar] [Publisher Link]

[15] S. WilsonPrakash, and P. Deepalakshmi, "Artificial Neural Network Based Load Balancing On Software Defined Networking," *2019 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS)*, Tamilnadu, India, pp. 1-4, 2019. [Crossref] [Google Scholar] [Publisher Link]

[16] N. G. V. R, and C. N. Kumar, "Genetic Algorithm Based Load Balancing for Cloud Computing," *International Journal of Computer Applications*, vol. 92, no. 10, pp. 1-5, 2018.

[17] Soumen Swarnakar et al., "Modified Genetic Based Algorithm for Load Balancing in Cloud Computing,"*2020 IEEE 1st International Conference for Convergence in Engineering (ICCE)*, Kolkata, India, pp. 255-259, 2020. [Crossref] [Google Scholar] [Publisher Link]

[18] Nawaf Alhebaishi, "An Artificial Intelligence (AI) Based Energy Efficient and Secured Virtual Machine Allocation Model in Cloud," *2022 3rd International Conference on Computing, Analytics and Networks (ICAN)*, Rajpura, Punjab, India, pp. 1-8, 2022. [Crossref] [Google Scholar] [Publisher Link]

[19] Jiayin Li et al., "Online Optimization for Scheduling Preemptable Tasks on IaaS Cloud Systems," *Journal of Parallel and Distributed Computing*, vol. 72, no. 5, pp. 666-677, 2012. [Crossref] [Google Scholar] [Publisher Link]

[20] H. G. H. W. Q, and D. G. Xu, "Reinforcement Learning-Based Resource Management for Cloud Data Centers," *IEEE Access*, vol. 5, pp. 13118-13128, 2017.

[21] Xin Sui et al., "Virtual Machine Scheduling Strategy Based on Machine Learning Algorithms for Load Balancing," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, pp. 1-16, 2019. [Crossref] [Google Scholar] [Publisher Link]

[22] Jim Gao Richard Evans, DeepMind AI Reduces Google Data Centre Cooling Bill by 40%, Google Deepmind, 2016. [Online]. Available: https://deepmind.google/discover/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-by-40/.

[23] Emmanuel Okyere, How DeepMind's AI Framework Made Google Energy Efficient, Nural Research, 2021. [Online]. Available: https://www.nural.cc/deepmind-ai-framework/.