



Multilabel Classification on Celebrity Face attributes

Using Deep Learning Techniques

Group 19: Bharath Velamala, Pranshu Singh Rawat

Dataset Details



The dataset which we have chosen is from Kaggle which is a [Celeb Face Attributes dataset](#), the dataset is annotated with around 40 attributes. Original data and banner image source came from a [research paper](#).

- 202,599 number of face images of various celebrities
- 40 binary attribute annotations per image such as Male, Attractiveness, Young etc.,

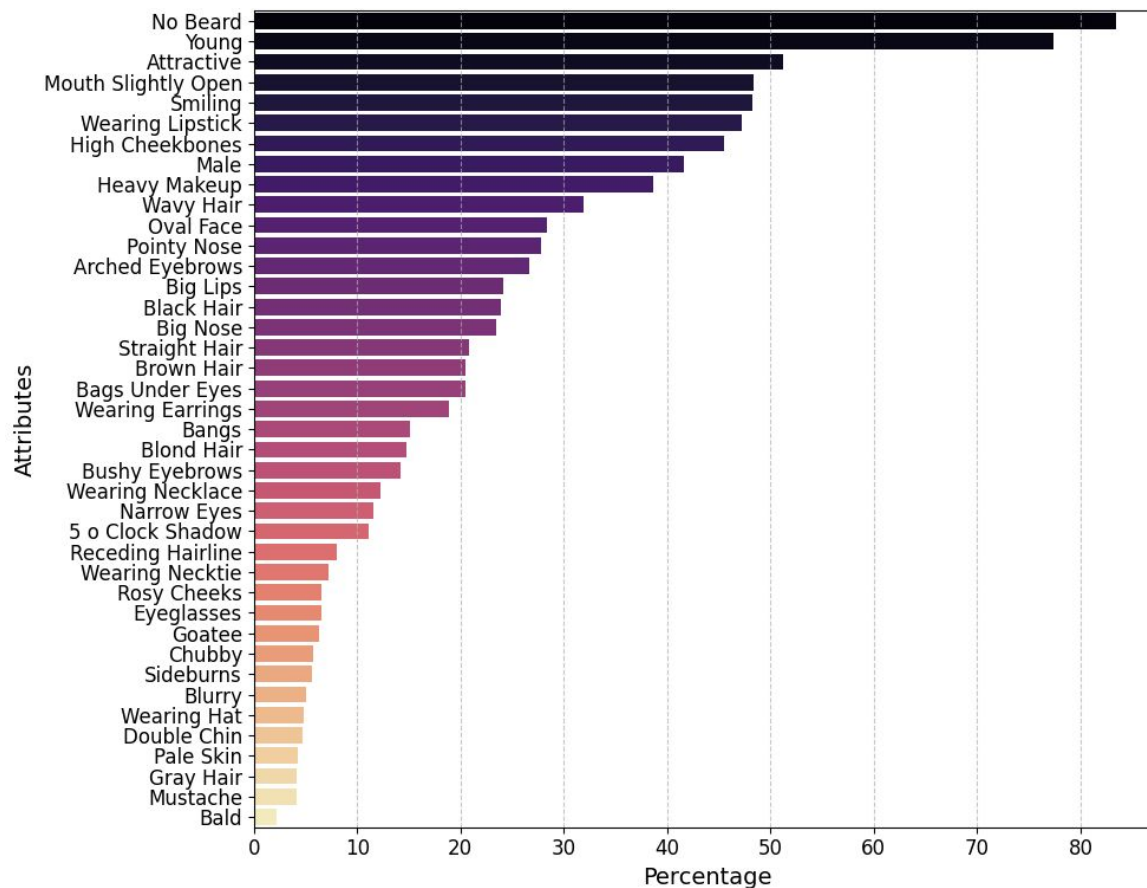
Problem Statement



The objective of this project is to create a multi-label classification model utilizing the dataset, which comprises over 200,000 celebrity images.

- The aim is to develop a Deep Learning model capable of accurately predicting multiple attributes associated with each image.
- **Content-Based retrieval** : Images can be retrieved on the basis of certain attributes (Can be used to identify attributes like big eyes, big nose to filter out images from a group of images)
- **Entertainment Industry** : Casting decisions can be made (Can be used for deciding if a face is attractive or not)

- The dataset is a class imbalanced dataset for which we balanced the dataset using class weights while training the model.



Deep Learning Methods



We are using two CNN architectures to perform the multi-label classification.

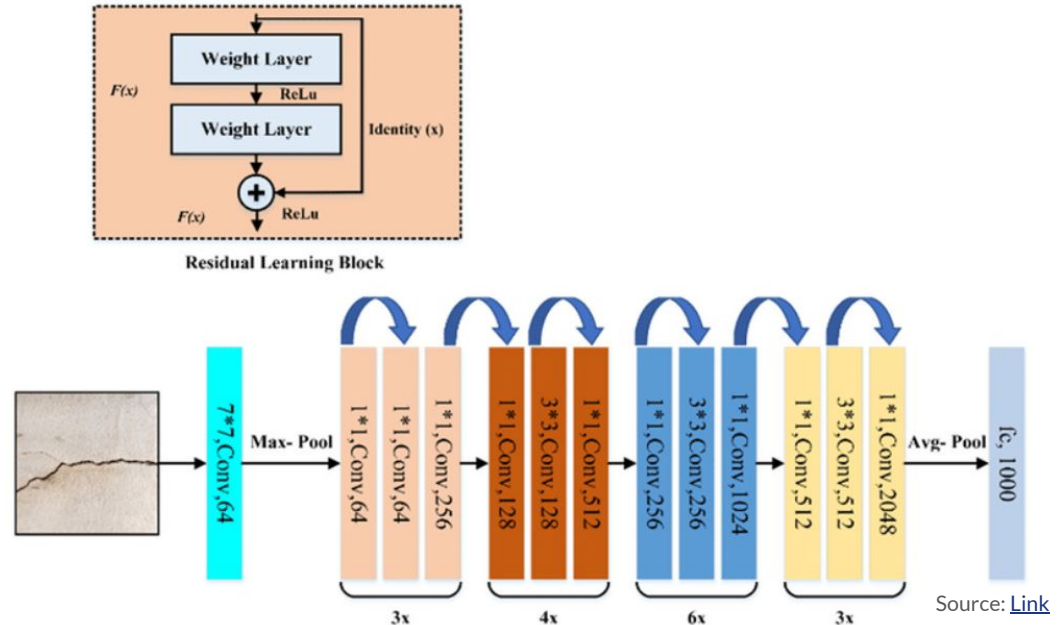
- **ResNet50:** ResNet-50 is a convolutional neural network architecture which consists of 50 layers with a unique residual connection design, facilitating the training of deeper networks.
- **DenseNet121:** DenseNet is a convolutional neural network architecture renowned for its densely connected layers, fostering feature reuse and enhancing gradient flow.

ResNet50

Residual connections in the architecture allow for the preservation of information across layers, mitigating the vanishing gradient problem.

Design Rules:

- Number of filters in each layer
- Feature map size and number of filters
- 7×7 kernel convolution layer
- 9 layers for stage 2
- 12 layers for stage 3
- 18 layers for stage 4
- 9 layers for stage 5

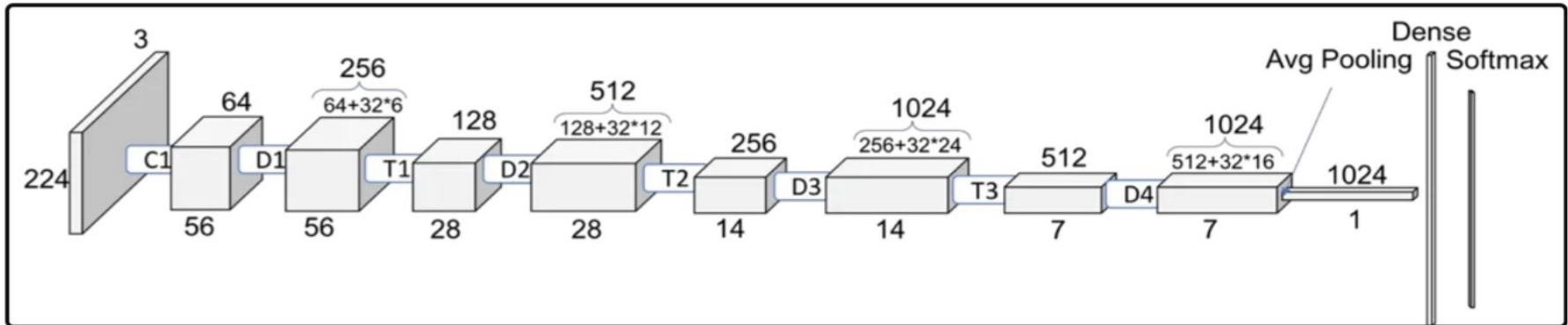


DenseNet121

Unlike traditional architectures, each layer receives input from all preceding layers, promoting feature propagation throughout the network.

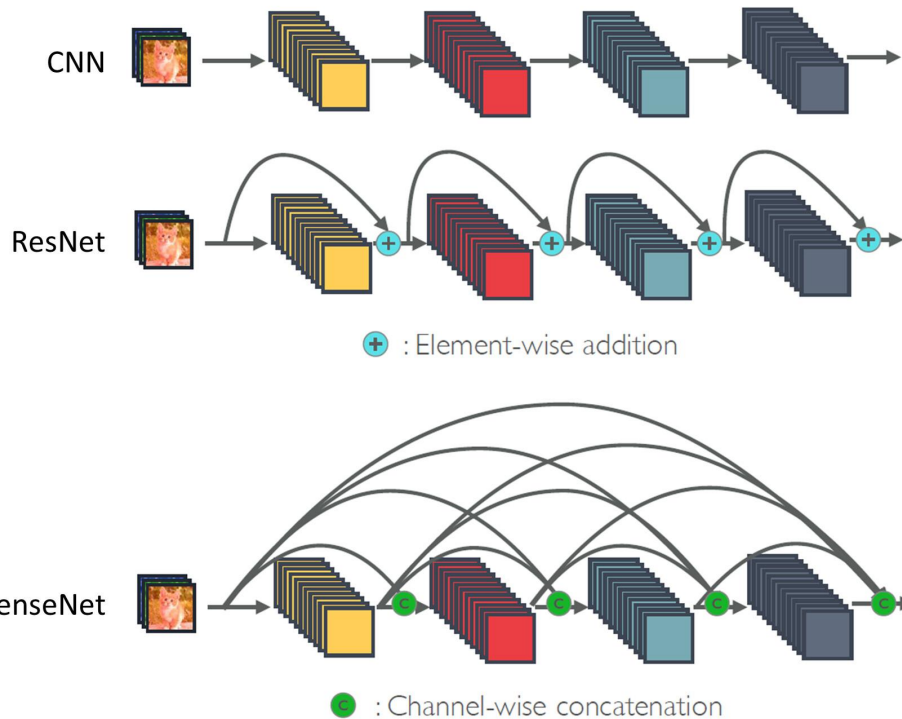
- 7×7 kernel convolution layer
- 2×6 Layers Dense Block with Transition Layer
- 2×12 Layers Dense Block with Transition Layer
- 2×24 Layers Dense Block with Transition Layer
- 2×16 Layers Dense Block

Source: [Link](#)



Resnet50 vs DenseNet121

By concatenating feature maps instead of adding them, DenseNet preserves fine-grained information and facilitates feature reuse.



Training and Validation



On our training data we only used a sample size of 29500 images using the ResNet and the Densenet architectures to compare the performance.

- Used Binary Cross Entropy loss
- Used Adam optimizer
- Hyperparameters used:
 - ◆ Learning Rate - 0.01, 0.005, 0.001
 - ◆ Number of features - 512
 - ◆ Batch Size - 64
 - ◆ Drop Out - 0.3, 0.5
- Epochs trained on : 8
- Hyper parameter tuning

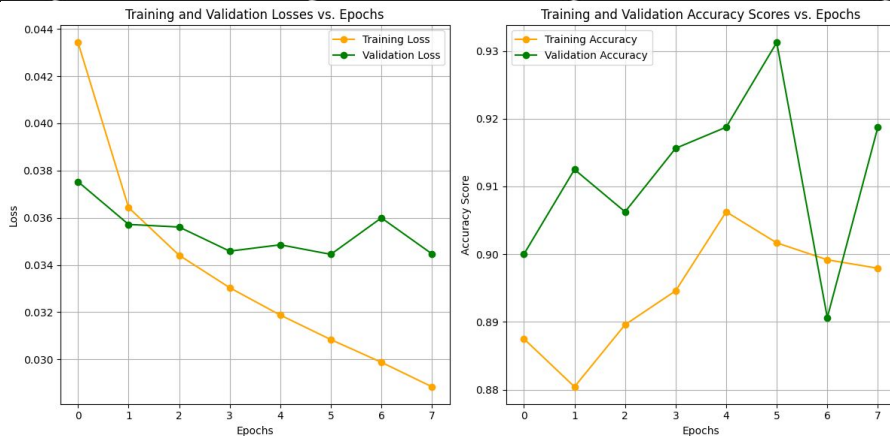
Model Evaluation - ResNet50

- After hyper parameter tuning we have taken the best performing model on the training and validation data for *Learning Rate - 0.001, Batch Size - 64, Drop Out - 0.3*.
- Training accuracy is increasing but Validation accuracy is not exactly increasing with each epoch.

Model: | batch_size = 64 learning_rate = 0.001, num_features = 512, drop_out = 0.3 |

Epoch 1/8: 100% | 625/625 [01:08<00:00, 9.17batch/s, accuracy_score=0.887, loss=0.0383]

Epoch [1/8], Train Loss: 0.0434, Train accuracy Score: 0.8875, Val Loss: 0.0375, Val accuracy Score: 0.9000



Model Evaluation - DenseNet121

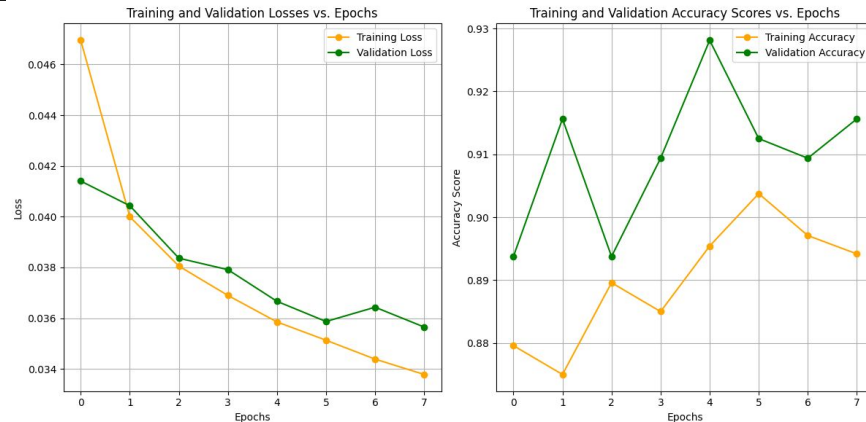
- After hyper parameter tuning we have taken the best performing model on the training and validation data for *Learning Rate - 0.001, Batch Size - 64, Drop Out - 0.3*.
- Training accuracy and Validation accuracy either decreasing or increasing with each epoch.

```
Model: | batch_size = 64 learning_rate = 0.001, num_features = 512, drop_out = 0.3 |
```

```
=====
```

```
Epoch 1/8: 100%|██████████| 625/625 [01:11<00:00, 8.68batch/s, accuracy_score=0.88, loss=0.0389]
```

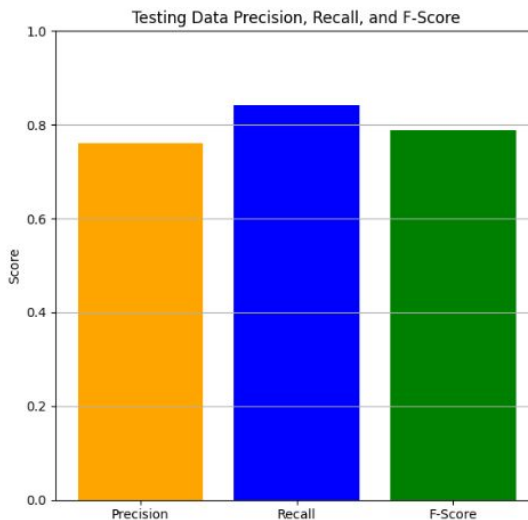
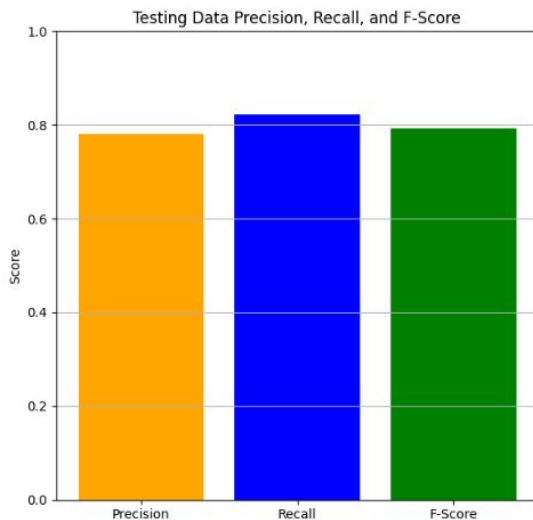
```
Epoch [1/8], Train Loss: 0.0470, Train accuracy Score: 0.8796, Val Loss: 0.0414, Val accuracy Score: 0.8938
```



Testing



ResNet50:
Accuracy - 0.89683
Precision - 0.7764
Recall - 0.8118
F-Score - 0.7863



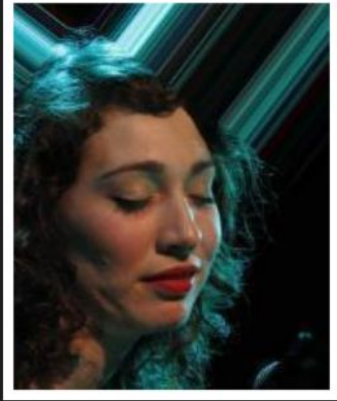



DenseNet121:
Accuracy - 0.892025
Precision - 0.7577
Recall - 0.82703
F-Score - 0.77685

Predictions

The predictions are being correctly generated for an image with some issues. We have used left image from test dataset and the right image which is an unseen data of our own.

- Misclassifying some attributes (like No Beard for ResNet50)
- Missing some attributes (like Eyeglasses for DenseNet121)

Model	Image	Code	Output
ResNet50		<code>predict(r"extracted_images/test_1/182639.jpg")</code>	<pre>[67] ... ---> No Beard <--- ---> Wearing Lipstick <--- ---> Young <---</pre>
		<code>predict(r"extracted_images/unseen_data/bharath.jpg")</code>	<pre>[39] ... ---> Big Nose <--- ---> Eyeglasses <--- ---> Male <--- ---> No Beard <---</pre>
DenseNet121		<code>predict(r"extracted_images/test_1/182639.jpg")</code>	<pre>[37] ... ---> No Beard <--- ---> Young <---</pre>
		<code>predict(r"extracted_images/unseen_data/bharath.jpg")</code>	<pre>[31] ... ---> Big Nose <--- ---> Black Hair <--- ---> Goatee <--- ---> Male <--- ---> Mustache <---</pre>

Future work and Conclusion

- We have currently applied ResNet 50 and DenseNet 121 architectures for our 2 models, we are also interested in the studying the results using the Inception V3 architecture before the final submission.
- We tried to apply the Inception V3 architecture but due to lack of computational resources we were only able to compute the former two architectures. **We tried implementing it but it was taking a lot of time to run so we were not able to run it.**
- As currently we have only used 29500 images out of the total dataset, we would try to retrain our model using a bigger dataset and getting some other major attributes. **We have trained with more images but not the entire dataset.**
- We would also like to change parameters and try to get a better understanding on the performance of the model. **We change some parameters such as epochs, batch size and tried but the model performance became worse it was unable to capture obvious attributes.**
- We tried to decrease the number of classes as well form 40 to 25 which didn't work in our favor so we retraced to our original implementation.