Face Recognition using Self-Supervised Learning (SimCLR) and Transfer Learning

Abstract:

This project focuses on developing a Face Recognition model using the SimCLR (Contrastive Learning) approach and Transfer Learning. It employs Convolutional Neural Networks (CNNs), specifically the ResNet18 architecture, to learn complex facial features for robust representation. Utilizing the Kaggle Faces Dataset, the Sim-CLR technique is applied for unsupervised pre-training, enhancing the model's ability to discern subtle facial differences. Data preprocessing and augmentation techniques diversify the dataset, improving generalization. The pre-trained encoder undergoes fine-tuning with the Sim-CLR contrastive loss function. A downstream ResNet18 CNN is integrated for face recognition classification, utilizing Sim-CLR encoder representations as input. Rigorous testing evaluates model performance, emphasizing ethical considerations and regulatory adherence in the face recognition technology landscape. This project advances face recognition by combining SimCLR's unsupervised learning with CNNs for accurate and efficient recognition across diverse conditions, contributing to real-world applications such as security surveillance and social media platforms.

Key Words: Self-Supervised Learning**,** Contrastive Learning, SimCLR, Transfer learning, ResNet18

Introduction:

Problem Definition:

This project addresses the challenge of Face Recognition through Self-Supervised Learning employing SimCLR. The problem involves training a model, specifically a ResNet18 architecture, to autonomously learn facial features without labeled data. Leveraging the SimCLR technique, the model undergoes unsupervised pre-training on a diverse dataset like Kaggle Faces. The objective is to enhance the model's ability to discern subtle facial differences for robust representation. The self-supervised approach enables nuanced feature extraction, contributing to accurate and efficient face recognition across diverse poses, expressions, and lighting conditions. The project aims to advance facial recognition capabilities without relying on traditional supervised methods, demonstrating the potential of self-supervised learning with SimCLR.

Background Study:

Transfer learning involves leveraging knowledge gained from one task to improve performance on a different but related task. SimCLR (Simultaneous Contrastive Learning) is a self-supervised learning technique designed for effective transfer learning in computer vision. SimCLR facilitates feature learning by maximizing the similarity between augmented views of the same image while minimizing similarity across different images. Through this unsupervised pre-training, SimCLR generates rich representations suitable for downstream tasks. In essence, SimCLR is a powerful tool within the domain of transfer learning, demonstrating its effectiveness in autonomously learning useful features from un-labelled data, subsequently enhancing the performance of various vision-related tasks.

Objective:

Develop a processed Dataset by extracting the face from the images by scaling the original images.

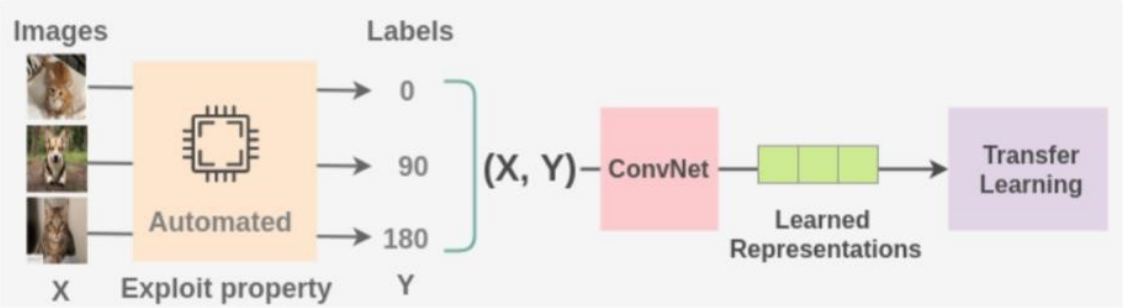
Develop a self-supervised model for face classification using SimCLR.

Develop a Model that uses pretrained ResNet18 and Train the model for classification

Methodology:

Self-Supervised Learning:

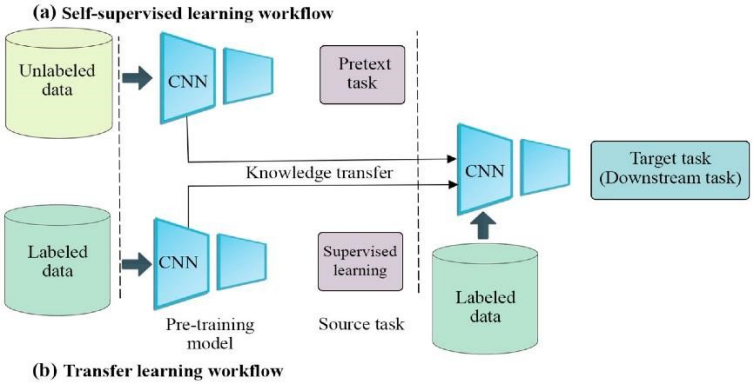
Self-supervised learning is a machine learning paradigm where a model learns to make predictions about its input data without relying on external labels or annotations. In traditional supervised learning, models are trained on labelled datasets where each input is associated with a corresponding output label. However, in self-supervised learning, the model itself generates its training labels from the input data. The key idea in self-supervised learning is to design pretext tasks, which are auxiliary tasks that can be created from the input data itself. The model is then trained to solve these pretext tasks, and the knowledge gained during this process is transferred to the primary task of interest. Once the model is trained on these pretext tasks, it can be fine-tuned on a smaller labelled dataset for the target task.



Common pretext tasks in self-supervised learning include:

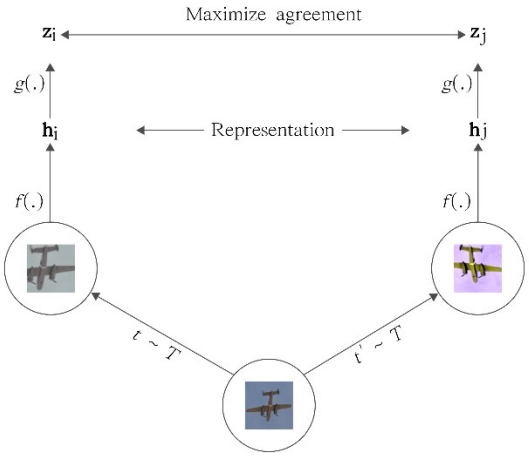
- Contrastive Learning
- Rotation Prediction
- Context Prediction

Self-supervised learning has shown promising results in various domains, including computer vision and natural language processing, and it is particularly useful when labelled data is scarce or expensive to obtain. The learned representations from self-supervised learning can often be transferred to downstream tasks, leading to improved performance.



Contrastive learning is a self-supervised learning paradigm that aims to teach a model to differentiate between similar and dissimilar pairs of data. The core idea is to maximize the similarity between positive pairs (instances that share certain characteristics) while minimizing the similarity between negative pairs (instances that differ). By doing so, the model learns to encode useful features that capture the underlying structure of the data. SimCLR (Simultaneous Contrastive Learning) is a specific approach within contrastive learning that has gained significant attention, particularly in computer vision tasks. Introduced by Chen et al. in 2020, SimCLR focuses on maximizing agreement between augmented views of the same image while minimizing agreement between views of different images.

In SimCLR, the training process involves creating positive pairs through data augmentation. Each instance is augmented twice, resulting in two augmented views of the same original image. The model then projects these augmented views into a shared embedding space, where the contrastive loss is computed. The contrastive loss encourages the model to bring representations of positive pairs close together in the embedding space and push representations of negative pairs apart.



SimCLR employs a specific normalization technique called "normalized temperature-scaled cross-entropy loss" to scale the similarity scores between representations. This temperaturescaled contrastive loss helps in generating robust and generalized representations.

One distinctive feature of SimCLR is the use of a large batch size during training, which enhances the model's ability to capture diverse features. Additionally, SimCLR employs a non-linear projection head to map the representations into a more discriminative space.

Model Descriptions:

Model 1:

In this project we implemented a SimCLR (Contrastive Learning) model for self-supervised learning on face data. Here's an explanation of the key components:

1. **Encoder Class:**
   The `Encoder` class defines the architecture for the SimCLR encoder. It takes a base encoder (in this case, ResNet18) and adds a projection head for dimensionality reduction. The encoder consists of a base encoder (all layers except the last one) and a projection head that reduces the dimensionality to the specified **out_dim**. The **forward** method takes an input tensor **x** and passes it through the base encoder and the projection head, producing the final encoded representation.
2. **Contrastive Loss (InfoNCE):**
   ContrastiveLoss class implements the InfoNCE (Noise Contrastive Estimation) loss used in SimCLR. It computes the loss based on the cosine similarity between pairs of encoded representations. The loss is calculated using a cross-entropy loss with a temperature parameter to scale the similarity values.

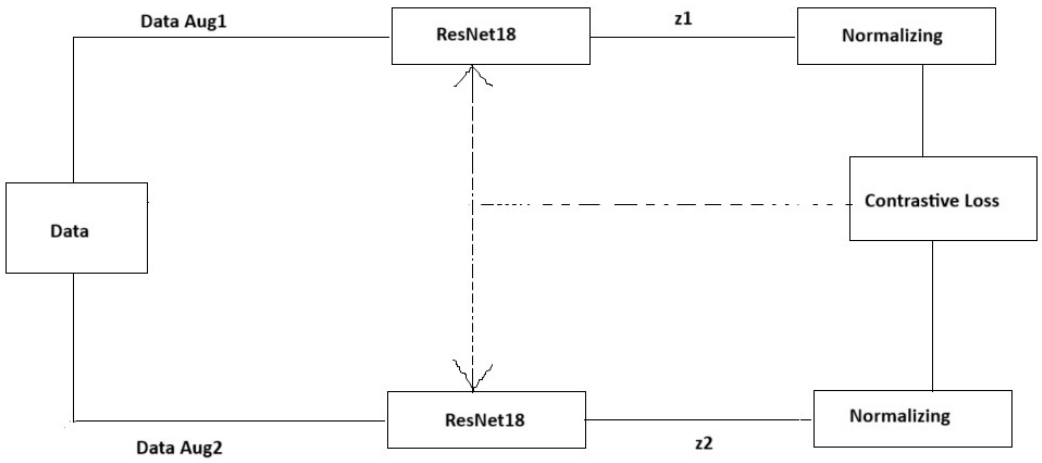
3. **Data Loading and Augmentation:**
   We prepared the datasets (train_dataset and val_dataset) and corresponding data loaders using PyTorch's Dataset and DataLoader classes. Data augmentation is performed on the training set by creating positive pairs through data cloning.

4. **Accuracy Calculation:**
The accuracy function computes the top-1 and top-3 accuracies, comparing the model's predictions with the ground truth labels.

5. **SimCLR Encoder Initialization:**
The code initializes the SimCLR encoder using a pre-trained ResNet18 model. The encoder is moved to the specified device (GPU if available).

6. **Loss and Optimizer:**
The contrastive loss is defined using the `ContrastiveLoss` class, and the Adam optimizer is used for training the SimCLR encoder.

7. **Training Loop:**
The model is trained through a loop over multiple epochs. During each epoch, positive pairs are created using data augmentation. The SimCLR encoder processes these pairs, and the contrastive loss is calculated for unsupervised learning. The model is updated using backpropagation and the Adam optimizer.



Model 2:
We implemented a classification model using a ResNet18 architecture for a face recognition task. Here's an explanation of the key components:

1. **ResNet18 Model Initialization:**
The code initializes a ResNet18 model using `torchvision.models.resnet18`. The model is configured for a classification task with 6 output classes (num_classes=num_labels). The model is moved to the specified device (device), which is either CPU.

2. **Optimizer and Loss Function:**

The Adam optimizer is employed with a learning rate of 0.0003 and weight decay of 0.0008. Categorical Cross Entropy Loss is chosen as the loss function, suitable for multiclass classification tasks.

3. **Training Loop:**

The model is trained through a loop over 5 epochs (`epochs`). For each epoch, the training dataset (train_loader) is iterated. The model processes batches of input data (x_batch) and corresponding labels (y_batch). Logits are computed using the model, and the Cross Entropy Loss is calculated between the logits and the ground truth labels.

The model parameters are updated using backpropagation and the Adam optimizer. Training accuracy (top-1) is computed and accumulated for later reporting.

4. **Validation Loop:**

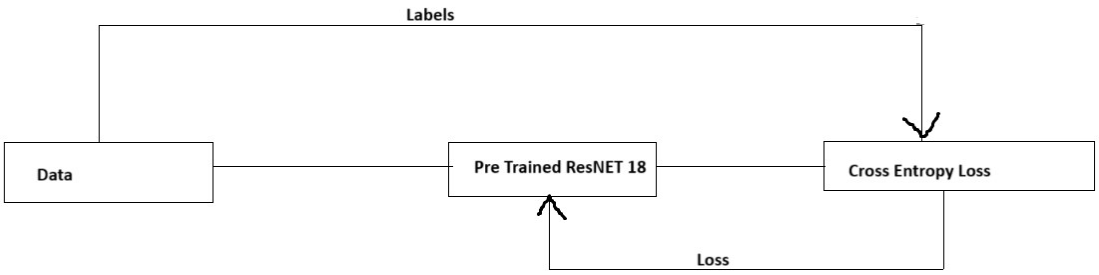
After each epoch, the model is evaluated on the validation dataset (val_loader) to assess its performance on unseen data. Validation accuracy (both top-1 and top-3) is computed and reported.

5. **Accuracy Calculation:**

The `accuracy` function computes the top-1 accuracy during both training and validation. Optionally, top-3 accuracy is calculated during validation.

6. **Performance Reporting:**

At the end of each epoch, the code prints out the training accuracy, top-1 test accuracy, and top-3 test accuracy.



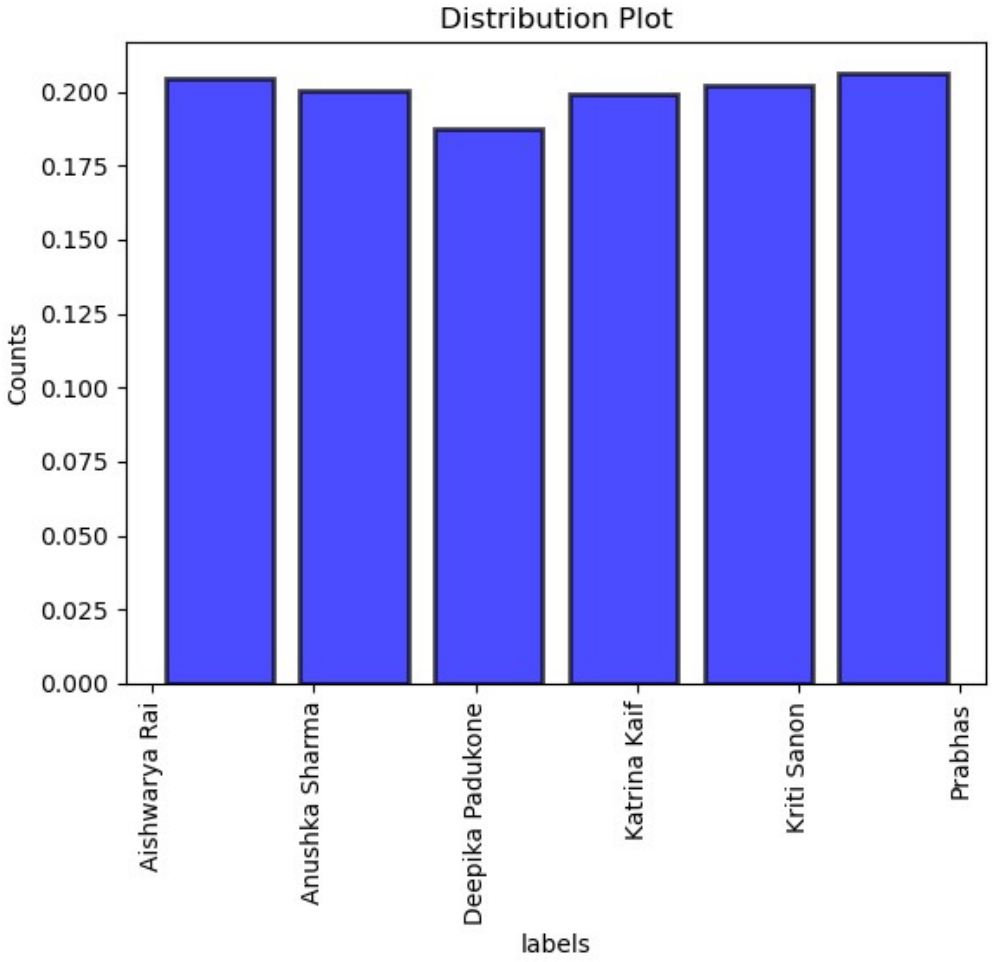
Experiment And Results:

Database:

Link: https://www.kaggle.com/datasets/hemantsoni042/celebrity-images-for-facerecognition/data

The Celebrity dataset Collected from Kaggle has 98 Celebrity Folders each folder contains multiple images of the celebrity in different poses, expressions, and lighting conditions.

Distribution of images in each folder (counts in Thousands)



a. Data Pre-processing

Reducing the image Count: From the dataset we are considering the folder with image count greater then 200 and coping those folders into the Preprocessed dataset folder. By doing this we got 6 Folder with more than 205 images in each folder

```
Copying subfolder: Aishwarya Rai, Number of images: 218
Copying subfolder: Anushka Sharma, Number of images: 209
Copying subfolder: Deepika Padukone, Number of images: 227
Copying subfolder: Katrina Kaif, Number of images: 206
Copying subfolder: Kriti Sanon, Number of images: 212
Copying subfolder: Prabhas, Number of images: 245
```

Dataset Splitting:

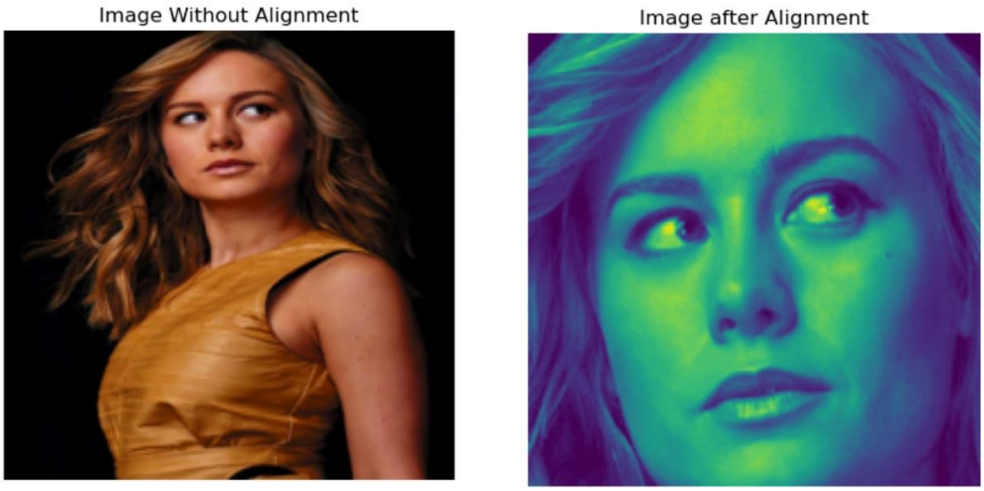
The pre-processed images are then split into training (80%), validation (10%), and testing (10%) sets using the split folders library. This step ensures that the model is trained on a diverse dataset and evaluated on distinct samples to assess its performance.

Detecting the Faces:

Prior to scaling, the faces in the images are detected using a pre-trained Haar Cascade face detector. The algorithm identifies faces within the images, and the coordinates of the bounding boxes around these faces are obtained.

Image Alignment:

The faces are then aligned to ensure consistency in size and orientation. The region of interest (ROI) corresponding to each detected face is cropped and resized to a standard dimension of 224x224 pixels. This standardization is crucial for creating consistent inputs for the subsequent phases of the model development. Then we are renaming the images with the Celebrity name and storing them in train, test and validation sets present in Scaled dataset.

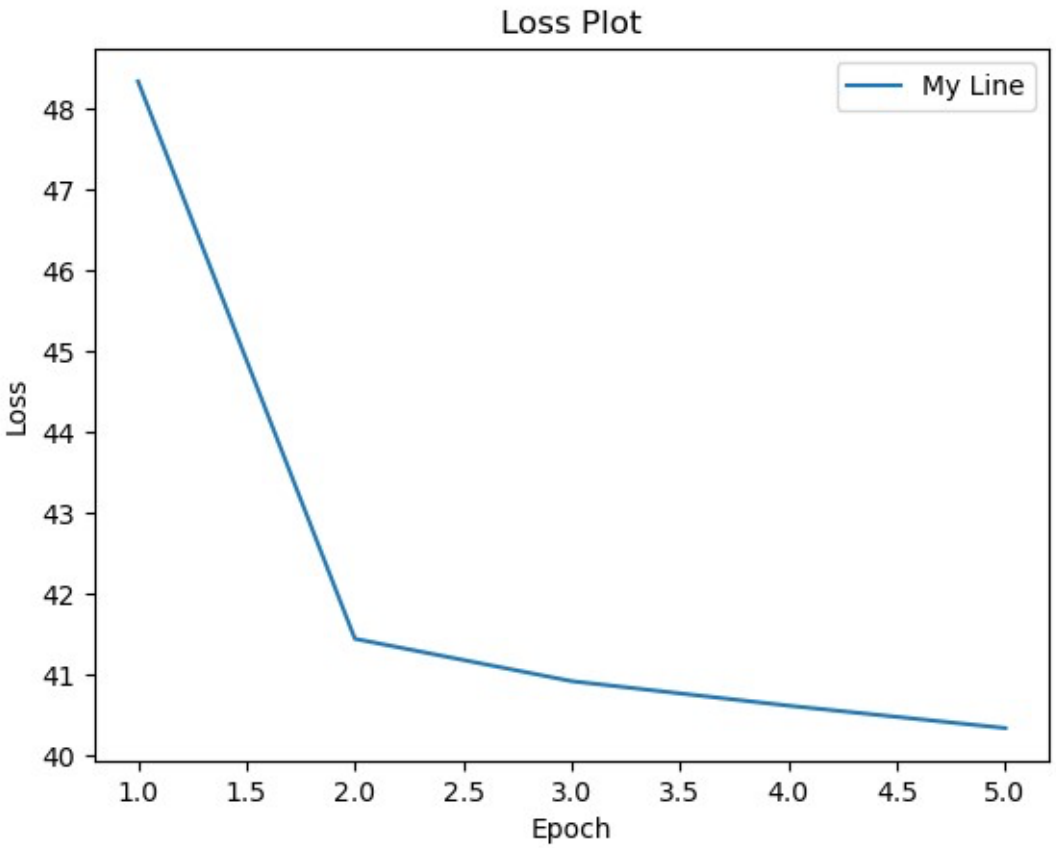


Training and Testing Logs:

The first model is a SIM-CLR model for semi supervised learning we trained the model as mentioned in methodology the following are the results that generated from this model.

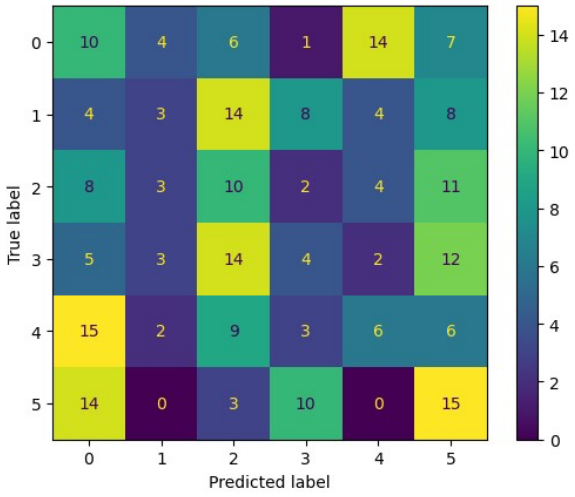
```
for batch 4     Top1 Test accuracy: 21.21394157409668   Top5 test acc: 55.949520111083984
Epoch [1/5], Contrastive Loss: 1.1601
for batch 4     Top1 Test accuracy: 20.703125   Top5 test acc: 56.12980651855469
Epoch [2/5], Contrastive Loss: 1.0740
for batch 4     Top1 Test accuracy: 21.09375    Top5 test acc: 55.64904022216797
Epoch [3/5], Contrastive Loss: 1.0486
for batch 4     Top1 Test accuracy: 20.3125     Top5 test acc: 53.87620162963867
Epoch [4/5], Contrastive Loss: 1.0613
for batch 4     Top1 Test accuracy: 20.522836685180664  Top5 test acc: 48.347354888916016
Epoch [5/5], Contrastive Loss: 1.1023
```

We have considered different methods to compute loss in the SimCLR model. As the loss must be calculated based on the prediction of output of two augmented images of the same image. The comparison is done based on the predicted values without comparing it with the true labels. This created many problems. Finally these are the values we ha obtained.



	precision	recall	f1-score	support
0	0.18	0.24	0.20	42
1	0.20	0.07	0.11	41
2	0.18	0.26	0.21	38
3	0.14	0.10	0.12	40
4	0.20	0.15	0.17	41
5	0.25	0.36	0.30	42
accuracy			0.20	244
macro avg	0.19	0.20	0.18	244
weighted avg	0.19	0.20	0.19	244

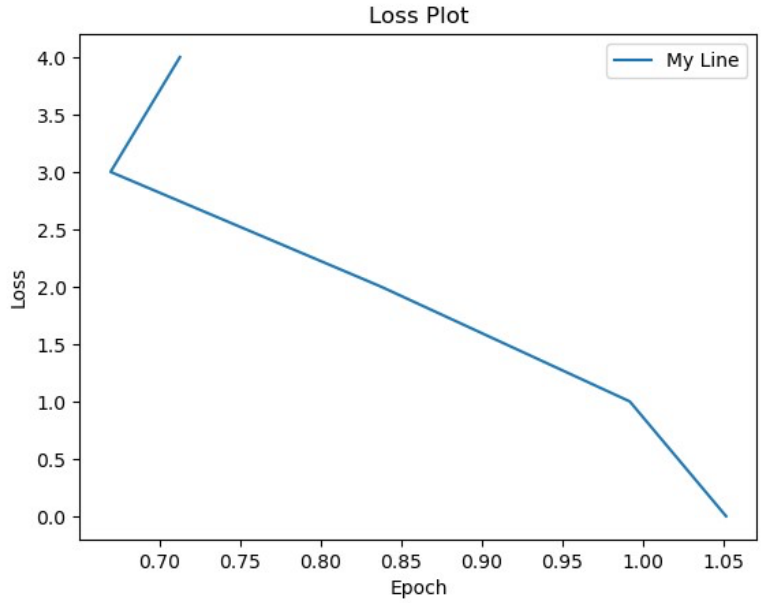
```
[[10  4  6  1 14  7]
 [ 4  3 14  8  4  8]
 [ 8  3 10  2  4 11]
 [ 5  3 14  4  2 12]
 [15  2  9  3  6  6]
 [14  0  3 10  0 15]]
```



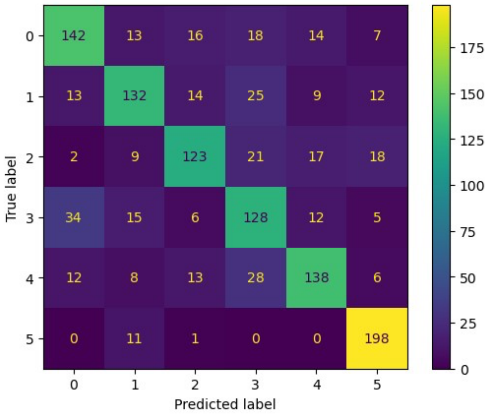
The Second Model is a pretrained ResNet18 architecture for a face recognition task as mentioned in methodology the following are the results of this model.

```
Epoch 0 Top1 Train accuracy 97.72135162353516   Top1 Test accuracy: 69.35095977783203   Top3 test acc: 88.88220977783203
Epoch 1 Top1 Train accuracy 98.046875   Top1 Test accuracy: 71.00360107421875   Top3 test acc: 89.96394348144531
Epoch 2 Top1 Train accuracy 98.60025787353516   Top1 Test accuracy: 69.74158477783203   Top3 test acc: 90.26441955566406
Epoch 3 Top1 Train accuracy 99.0234375  Top1 Test accuracy: 71.60456848144531   Top3 test acc: 93.47956848144531
Epoch 4 Top1 Train accuracy 98.79557037353516   Top1 Test accuracy: 71.39422607421875   Top3 test acc: 91.52644348144531
```

The loss in this model is calculated using categorical cross entropy bby comparing the true labels with predicted labels. So this model provided a better comparison and the resulting loss function updated the weights I the model better. As the model is a pretrained model, with just 5 epochs the model reached a 98% Top 3 accuracy. These are the results obtained from the model.



	precision	recall	f1-score	support
0	0.70	0.68	0.69	210
1	0.70	0.64	0.67	205
2	0.71	0.65	0.68	190
3	0.58	0.64	0.61	200
4	0.73	0.67	0.70	205
5	0.80	0.94	0.87	210
accuracy			0.71	1220
macro avg	0.70	0.70	0.70	1220
weighted avg	0.71	0.71	0.70	1220



Comparison and Discussion:

In comparing the two models, it is evident that the choice of training approach significantly influences performance. Model 1, employing SIM CLR with a pretrained ResNet18, faced challenges in achieving satisfactory accuracy and exhibited a slower rate of convergence during the training process. The utilization of a contrastive learning method like SIM CLR may have introduced complexities that hindered effective feature extraction from the pretrained ResNet18.

On the other hand, Model 2, leveraging transfer learning with ResNet18, showcased superior performance with a commendable 71% accuracy. The accelerated convergence and higher accuracy suggest that the pretrained ResNet18 model effectively transferred knowledge from its prior training, enhancing its ability to generalize to the specific classification task at hand. The classification report for Model 2 further emphasizes its balanced performance across various classes, indicating a robust and reliable model.

Conclusion

Looking at the results obtained we can come to a conclusion that training the SimCLR Model with a pretrained ResNet18, achieved a lower accuracy of 20% and exhibited slower training convergence over 5 epochs. The classification report indicates poor performance across multiple classes, with low precision and recall. In contrast, Model 2, employing transfer learning with ResNet18, demonstrated superior results with a 71% accuracy and faster training convergence. The classification report for Model 2 indicates well-balanced precision, recall, and F1-scores across different classes. The utilization of transfer learning with ResNet18 appears more effective, providing improved accuracy and efficiency compared to SIM CLR in this scenario.

References:

[1]. Ting Chen, Simon Kornblith, Mohammad Norouzi, Geoffrey Hinton: A Simple Framework for contrastive Learning of Visual Representations, 2020 **https://arxiv.org/pdf/2002.05709v3.pdf**

[2]. Siladittya Manna, SimCLR in PyTorch USING JUPYTER NOTEBOOK, Jun 30, 2021 **https://medium.com/the-owl/simclr-in-pytorch-5f290cb11dd7**

[3]. Gaurav Singhal, Transfer Learning with ResNet in PyTorch, May 5, 2020 **https://www.pluralsight.com/guides/introduction-to-resnet**

[4]. Nikolas Adaloglou, Self-supervised learning tutorial: Implementing SimCLR with pytorch lightning ,2022-03-31 **https://theaisummer.com/simclr/**

[5]. https://www.kaggle.com/datasets/hemantsoni042/celebrity-images-for-facerecognition/data

[6]. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

[7]. https://pytorch.org/vision/stable/index.html