Introduction to Machine Learning

Saima Sharleen Islam
University of Trento
Trento, Italy
saimasharleen.islam@studenti.unitn.it

ABSTRACT

This study investigates image classification and retrieval using various deep learning models, including a CNN built from inception and transfer learning-based models such as ResNet and VGG. Limitations and opportunities for future advancement in this discipline are identified by the research. The size of the dataset is a limitation, indicating that a larger dataset is required to enhance model generalization. Analyzing the evaluation metrics highlights the importance of integrating additional metrics for a comprehensive evaluation. Model selection and architecture justification, as well as the investigation of model interpretability and explainability, are identified as areas for enhancement. The study recommends potential improvements to fine-tuning strategies, data augmentation techniques, and the investigation of alternative architectures. In conclusion, the abstract highlights the importance of resolving these limitations and conducting additional research to enhance the performance and comprehension of image classification and retrieval systems.

KEYWORDS

Machine Learning, Human Facial Recognition, CNN, Transfer Learning, Keras Pretrained models

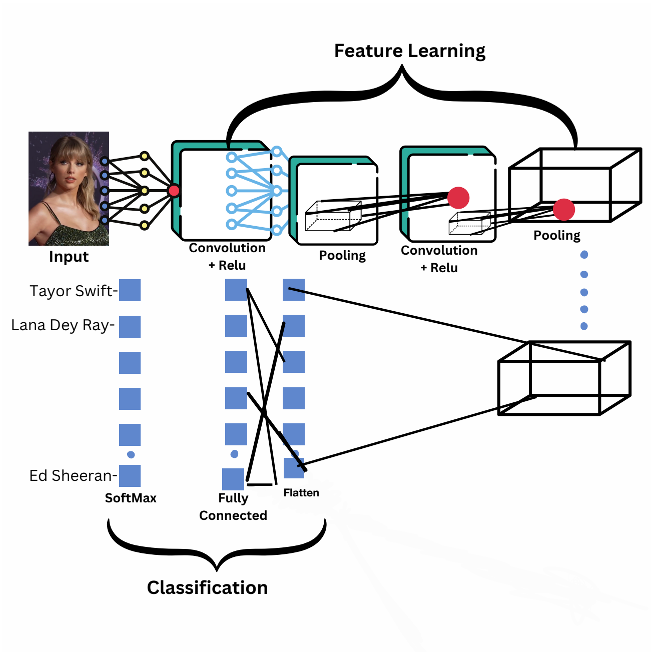


Figure 1: Deep Learning

1 INTRODUCTION

Image retrieval is a critical task in various domains, including criminology, medicine, and web image search [3]. This study aims to present the development of an image retrieval system designed to retrieve images that closely resemble a given query image set. The system comprises two primary components: feature extraction and similarity computation. Unlike other tasks that may depend on labeled data, our image retrieval problem necessitates the selection of a machine-learning model capable of extracting the most relevant features[7]. The similarity between the query image and the images in the gallery is computed based on the distance or dissimilarity between their respective features in the feature space. The choice of distance metric is crucial, as it directly influences the accuracy and precision of the retrieval process[5].

In modern day image search engines, the search-by-example functionality has become an essential feature. This feature's effectiveness is largely attributable to the implementation of a robust image similarity metric.To classify and label images, image classification models are used to construct image search engines. In addition, they employ image similarity metrics to retrieve and rank images according to their visual similarity to a given query image. In the context of image classification, photographs of various public figures such as "Taylor Swift," "Ed Sheeran," and "Lana Del Rey" would all be categorized as "celebrities."

In contrast, the objective of similar image ranking, given a query image portraying "Taylor Swift," is typically to rank "Lana Del Rey" images higher than those of other celebrities such as "Ed Sheeran." This illustrates the distinction between image classification, which focuses on categorization, and similar image ranking, which prioritizes fine-grained image similarity. The primary objective of this project is to develop an image search engine capable of accepting a query image as input and returning the N most similar images from a gallery. To exemplify the problem, we consider the scenario of retrieving images that depict the same location using a visual search algorithm. Specifically, given an input query image, the algorithm should be able to match it with another gallery image that portrays the same location. Matching is typically accomplished by defining a similarity or distance metric based on the extracted image features. Once the features of the query image have been extracted, the similarity or distance measure can be computed between the query features and each gallery image. The matches can then be sorted based on their feature distance, with the top-k matches representing the gallery images with the lowest feature distance from the query image.

In this paper, we rank similar images using various deep learning models[2], including we built-up CNN from scratch, use pretrained model on ResNet28, and transfer models on VGG16, VGG19,

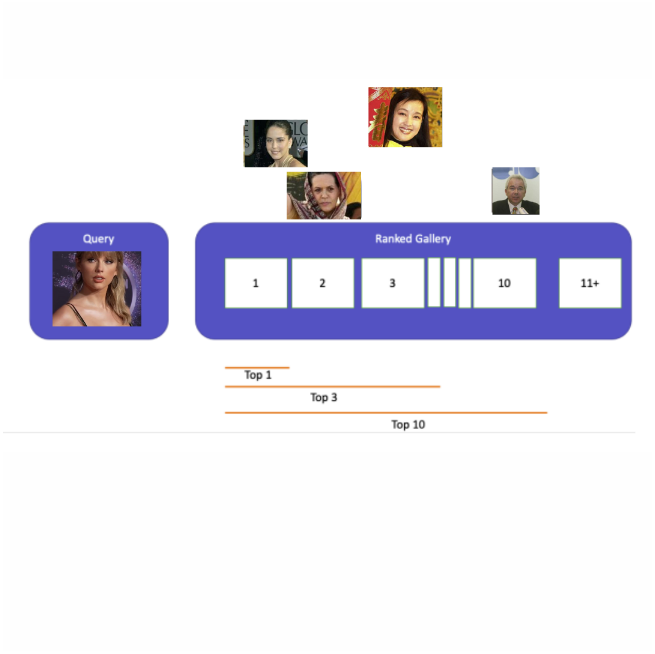


Figure 2: Query from Gallery

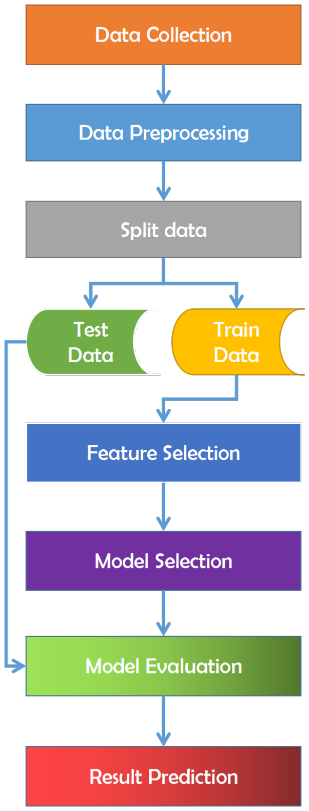


Figure 3: Overview of the employed methodology for this research.

ResNet50, ResNet50V2, and others available in Keras Applications [1]. In the following sections, we will examine the image retrieval process and data acquisition in greater depth, emphasizing the significance of these aspects in the development of an effective and efficient retrieval system.

2 METHODOLOGY

To begin, data collection is the first stage in this research. The obtained data is then analyzed, and a feature selection process is carried out. Figure 4 refers to the methodology that has been applied for this work.

Algorithm 1 Pseudocode for Image Similarity

```
1. Load images from query and gallery sets using two separate dataloaders with batch size = 1
2. Extract features from data using a pretrained Residual Neural Network (ResNet) model
3. Store features in two dictionaries: one for query images and one for gallery images
4. For each query image, compute similarity and store it
5. Use a top-k function to return a final dictionary mapping each query image to the gallery images that are most similar to it
```

2.1 Data Collection

For the purposes of our study, we employed two data collection techniques: web scraping and the use of an image download tool. We successfully assembled a large training dataset. However, due to computational limitations, we were only able to utilize a subset of this extensive dataset in our research containing 148 classes and 3,578 training images. Additionally, we compiled 718 images for validation. Finally, the processed data is used to implement all of the deep learning algorithms.

Our dataset is stored as.jpg, .jpeg, .png files in various folders, with each folder bearing the name of the image class represented by the images contained within. We import the images using the ImageDataBunch.from_folder() function and label them based on the name of the folder they were read from.

Figure 4: Data Explorations

2.2 Data Preprocessing

Several transformations and segmentation techniques were applied to the dataset during the data preparation phase. These included rescaling, shearing, zooming, rotation, horizontal flipping, and an 80/20 division for validation and training.

The images were rescaled to ensure that their sizes and proportions were uniform. This step serves to normalize the images and prepare them for subsequent processing. By employing a variety of shear transformations, the dataset is augmented with variations in the images' perspectives, thereby enhancing the model's ability to account for a variety of viewing angles. By employing a zoom range, the dataset is augmented with images of varying levels of magnification, enabling the model to accommodate varying levels of detail. Introducing variations in the orientation of the objects into the dataset by rotating the images within a specified range. This enhancement makes the model more resistant to rotations in realistic scenarios. To horizontally flip an image, its vertical axis must be mirrored. This enhancement enlarges the dataset by flipping the original images, providing the model with more examples to learn from and enhancing its ability to recognize objects from varying perspectives.

The dataset was split into two subsets, with 70% of the images designated for training and 30% for validation. This partition ensures that the model is trained on a substantial portion of the data while retaining an independent set for testing and evaluation. These data preparation techniques, which include rescaling, shearing, zooming, rotation, horizontal flipping, and the 80/20 divide, contribute to a more diverse and representative dataset, allowing the model to learn effectively and generalize well to new, unseen images.

2.3 Feature Selection

2.3.1 CNN(From Scratch): CNN is an abbreviation for Convolutional Neural Network, which is a specialized neural network for processing data with a 2D matrix input structure, such as images.

2.3.2 Keras Pre-trained Models: This method applies the knowledge gained from training on a large dataset to a new task with a smaller dataset to enhance performance. Initial features are derived from the pre-trained features, and the final few layers are fine-tuned on the specific dataset to modify the model to the new task.

2.3.3 Transfer Learning Models: In the context of deep learning and transfer learning, the ResNet50 model contains a feature extraction component. Through its convolutional layers, this pre-trained model has learned to extract high-level features from images. Utilizing the pre-trained ResNet50 model, which has already learned a set of representative features from a large dataset (ImageNet), implicitly performs feature selection.

By adding a Flatten layer after the output of the pre-trained model and connecting it to a Dense layer, the code creates a new classification model that learns to classify images based on the extracted features. Instead of explicitly selecting features from the input data, the feature selection is conducted by leveraging the pre-trained ResNet50 model's feature extraction capabilities.

2.4 Models and Implementations

2.4.1 CNN(From Scratch): Using TensorFlow and Keras, a Convolutional Neural Network (CNN) model was developed from inception. The dataset and labels are converted into NumPy arrays, with the normalized dataset values and binarized labels[6]. For evaluation purposes, the dataset is divided into training and testing sets.

Sequential API from Keras is used to define the model architecture, which includes convolutional layers, batch normalization, max pooling, and dropout. The add_conv_block function facilitates convolutional block addition. The final layer uses the softmax activation function to classify multiple classes[12].

With the categorical cross-entropy loss function, the Adam optimizer, and the accuracy metric, the model is compiled. The summary method provides an architectural overview (Table:1) of the model.

Layer (type)	**Output Shape**	**Param #**
conv2d	(None, 128, 128, 32)	896
batch_normalization	(None, 128, 128, 32)	128
conv2d_1	(None, 126, 126, 32)	9248
max_pooling2d	(None, 63, 63, 32)	0
conv2d_2	(None, 63, 63, 64)	18496
batch_normalization_1	(None, 63, 63, 64)	256
conv2d_3	(None, 61, 61, 64)	36928
max_pooling2d_1	(None, 30, 30, 64)	0
conv2d_4	(None, 30, 30, 128)	73856
batch_normalization_2	(None, 30, 30, 128)	512
conv2d_5	(None,28 ,28 ,128)	147584
max_pooling2d_2	(None ,14 ,14 ,128)	0
flatten(Flatten)	(None ,25088)	0
dense(Dense)	(None ,98)	2458722
Total params: 2746626		
Trainable params: 2746178		
Non-trainable params: 448		

Table 1: Model Summary

2.4.2 Keras Pre-trained Models: Pretrained models refer to models that have been trained on large datasets, such as ImageNet, to learn general features and patterns. These models are typically trained on a specific task, such as image classification, and their learned weights capture valuable information about the data. These pretrained models can be used as a starting point for other tasks, where the pre-trained weights are leveraged to extract meaningful features from new data. By using pretrained models, researchers and developers can benefit from the knowledge and representations learned from vast amounts of data.

In this study, a pre-trained ResNet18 Convolutional Neural Network (CNN) model is used through transfer learning[8]. This approach allows for the knowledge gained from a pre-existing neural network, initially trained for image recognition tasks, to be adapted to a specific use case through fine-tuning. The motivation for using transfer learning is the challenge of obtaining enough labeled data to effectively train a neural network from scratch.

Typically, a large number of image samples, often 300,000 or more, are necessary to train a neural network model that delivers satisfactory performance. However, in this scenario, the available training set consists of only about 3,578 images. Training a neural network from scratch with such limited data would likely yield suboptimal results. Transfer learning offers a practical solution by leveraging the knowledge gained from a pre-trained model trained on a large-scale dataset, such as the ImageNet database.

2.4.3 Transfer Learning Models: Transfer learning, on the other hand, is a broader concept that encompasses the use of pretrained models. Transfer learning involves taking a model that has been pretrained [11]on one task or dataset and applying it to a different but related task or dataset. The idea is to transfer the knowledge and representations learned from the source task to the target task, which typically has a smaller dataset. By utilizing the pre-existing knowledge, the model can generalize better and achieve improved performance with less training data.

Importing the essential libraries and instantiating the ResNet50 model, which has been pre-trained on the ImageNet dataset, is the first step of this part. The ResNet50 model's layers are then made non-trainable to preserve the pre-trained weights. To adapt the ResNet50 model to the specific classification assignment, a new layer is added. This layer is comprised of a Flatten layer and a Dense layer with softmax activation to generate the final classification predictions. Using TensorFlow's Model[13] class, the input and output layers of the resultant model are specified.

3 RESULTS

3.0.1 CNN from Scratch: During the training and evaluation of a CNN model implemented with Keras. Initially, a learning rate scheduler function is defined, which modifies the training learning rate based on the epoch number. At specified intervals, in this case every epoch, the learning rate decays by a specified rate.

The model is then trained by passing the training dataset (x_train and y_train) and the validation dataset (x_test and y_test) to the fit method. The training is conducted for 10 epochs with 128-person batches. During training, two callbacks are utilized: EarlyStopping and modelCheckpoint. EarlyStopping monitors validation accuracy and terminates training if it does not improve after three consecutive epochs, thereby preventing overfitting. ModelCheckpoint stores the optimal model based on validation precision.

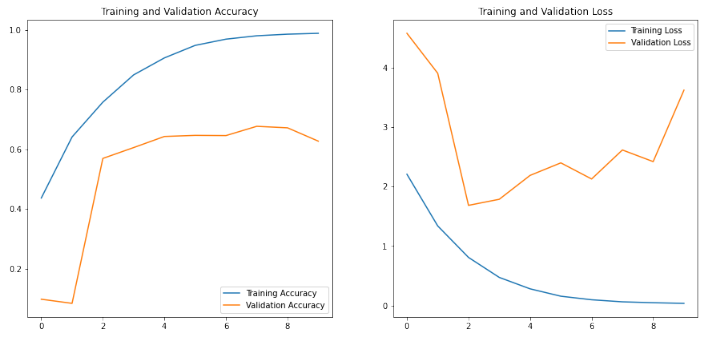


Figure 5: Training and Validation Loss of CNN

3.0.2 Keras Pre-trained Models: The pre-trained ResNet18 model[4] is incorporated using the "create_cnn()" function. This function allows for the loading of the pre-trained network and enables fine-tuning and adaptation to the specific image classification task[10]. By using transfer learning and starting with the pre-trained ResNet18 model, the extensive pre-existing knowledge can be harnessed to optimize the model's performance on the relatively limited training data.

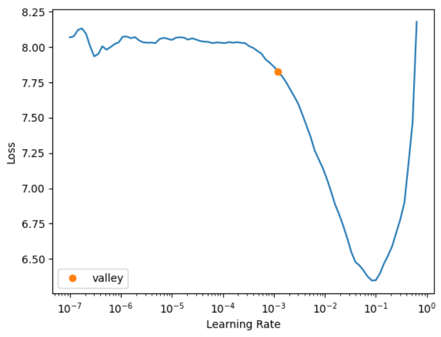


Figure 6: Learning Rate of pre-trained model

Epoch	Train Loss	Valid Loss	Accuracy	Time
0	0.405578	0.290146	0.901283	05:33
1	0.383602	0.299907	0.902708	04:29
2	0.336319	0.306456	0.911974	05:29
3	0.343718	0.326254	0.892374	05:30
4	0.364961	0.304092	0.906985	05:29
5	0.515605	0.546500	0.875624	05:30

Table 2: Training Results of pre-trained model

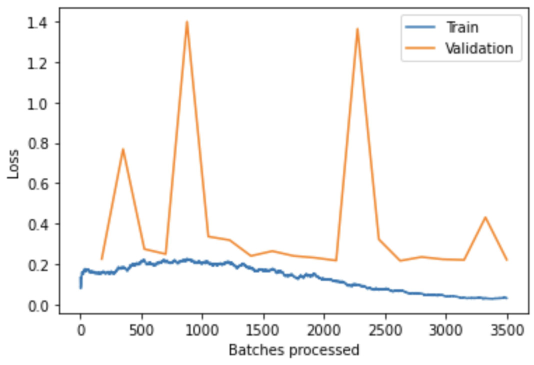


Figure 7: Training and validation Loss of pre-trained model

3.0.3 Transfer Learning Models: The final four model layers are then made trainable by assigning their trainable attribute to True. This allows for the fine-tuning of these layers while the pre-trained weights of the prior layers remain intact. The model is then compiled with the specified loss function (categorical cross-entropy), optimizer (Adam), and metric of evaluation (accuracy)[9]. To increase the diversity of training samples, the ImageDataGenerator applies data augmentation to the training data by performing transformations such as rescaling, shearing, zooming, and horizontal rotating. Using the flow_from_directory function, which loads the images from their respective directories and applies the specified preprocessing steps, the training and validation sets are generated. A learning rate scheduler is defined in order to modify the training learning rate based on the epoch number. The model is then trained by specifying the training and validation sets, number of epochs, steps per epoch, and validation steps within the fit function. Multiple callbacks, including early pausing, model checkpointing, and the learning rate scheduler, are utilized. The training history, which includes accuracy and loss values, is extracted from the training procedure and plotted with Matplotlib. The resulting diagrams illustrate the variations in accuracy and loss throughout the training epochs. The provided code builds a classification model using transfer learning and the ResNet50 model. Except for the last four layers, which are fine-tuned for the specific task, the pre-trained weights of the ResNet50 layers are frozen. The model is trained with data augmentation and monitored with multiple callbacks to guarantee optimal performance. The results 3 indicate that the

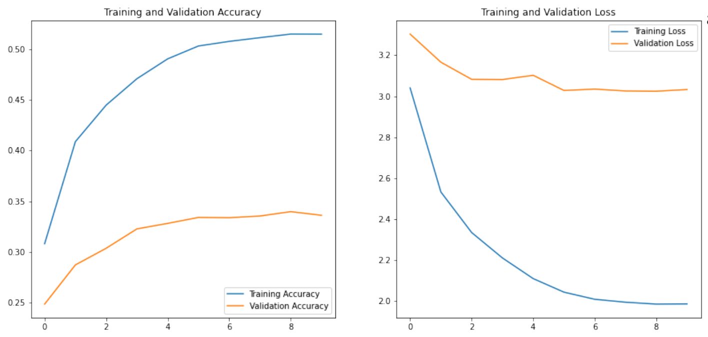


Figure 8: Training and validation Loss of transfer learning model

Models	Top-1	Top-5	Top-10
Transfer Learning on ResNet50	0.42	0.45	0.5
Transfer Learning on VGG16	0.3	0.29	0.397
Transfer Learning on ResNet50V2	0.49	0.5	0.615
CNN from Scratch	0.31	0.33	0.5
Pre-trained Model on ResNet18	0.47	0.56	0.59

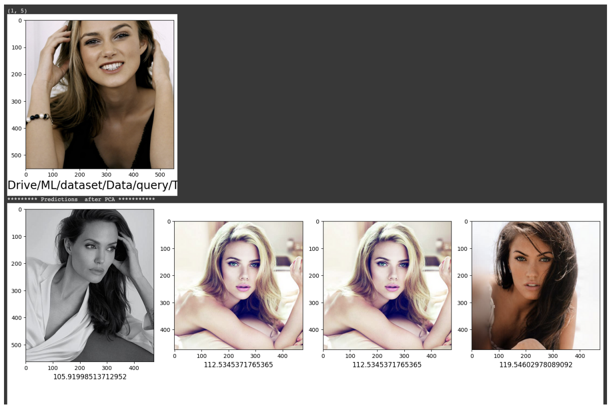
Table 3: Model Performance Comparison

transfer learning-based models, especially those utilizing ResNet50 and ResNet50V2, outperformed the CNN from Scratch model and the VGG16 model. Transfer learning enables the use of the knowledge and learned features of pre-trained models from large datasets, resulting in superior performance compared to training models from start. The unique architecture and pre-trained weights employed by each model contribute to performance differences. Top-1 accuracy was 0.31, Top-5 accuracy was 0.33, and Top-10 accuracy was 0.5 for the CNN from Scratch model. This indicates that the performance of this model was comparatively inferior to models based on transfer learning.

Transfer Learning on ResNet50 achieved a Top-1 accuracy of 0.42, which means that 42% of the predictions made by this model were correct for the top-ranked result. Similarly, the Top-5 and Top-10 accuracies for this model were 0.45 and 0.5, respectively. Transfer Learning on VGG16 had a lower performance with a Top-1 accuracy of 0.3, indicating that only 30% of the predictions for the top-ranked result were correct. The Top-5 and Top-10 accuracies for this model were 0.29 and 0.397, respectively. Transfer Learning on ResNet50V2 performed better than the previous models, achieving a Top-1 accuracy of 0.49, a Top-5 accuracy of 0.5, and a Top-10 accuracy of 0.615. This indicates a higher percentage of correct predictions for both the top-ranked result and the subsequent ranks.

The Top-1 accuracy of the Pre-trained Model on ResNet18 was 0.47, the Top-5 accuracy was 0.56, and the Top-10 accuracy was 0.59. This model performed better than CNN from Scratch, but marginally worse than ResNet50 and ResNet50V2 transfer learning models.

By sampling a portion of the results, we can also evaluate their quality. Since verifying for every result would be time-consuming, a randomly selected query is chosen, and its five most similar images are displayed side-by-side using VGG16 Transfer Learning.


```
(1, 5)
```

```
********* Predictions  after PCA ***********
```

Figure 9: Randomly Selected Query and top similar images

4 FUTURE WORK

The work presented on image classification using CNN from scratch and transfer learning-based models has yielded valuable insights and commendable outcomes. However, there are a number of limitations and opportunities for future development. A limitation is the relatively small size of the dataset, which could be increased to improve the generalization capabilities of the model. In addition, the evaluation metrics predominantly focused on top-1, top-5, and top-10 accuracy, whereas incorporating additional metrics such as precision, recall, or F1 score would have resulted in a more comprehensive evaluation. The selection of models and architectures lacked justification and comparison to alternate options, necessitating a more comprehensive model selection procedure. In addition, the interpretability and explainability of the models' decisions could be investigated for additional insight. Potential improvement areas include fine-tuning strategies, data augmentation techniques, and domain-specific optimizations. Additionally, examining additional architectures, such as DenseNet, Inception, and MobileNet, could yield useful comparisons. Addressing these limitations and pursuing future research in these areas would improve the performance and comprehension of models.

5 CONCLUSIONS

This study compared CNN from scratch and transfer learning models for image classification and found that transfer learning models, particularly those based on ResNet50 and ResNet50V2, outperformed CNN from scratch and VGG16. The ResNet50-based transfer learning model obtained a Top-1 accuracy of 0.42, while the VGG16-based model performed poorly with a Top-1 accuracy of 0.30. ResNet50V2-based model performed even better, with Top-1 accuracy of 0.49. Future research in this area should address several limitations and investigate opportunities for improvement, such as increasing the dataset size, expanding evaluation metrics, justifying the selection of models and architectures, investigating interpretability and explainability, enhancing fine-tuning strategies and data augmentation techniques, and examining additional architectures.

REFERENCES

[1] [n. d.]. Keras Applications. https://keras.io/api/applications/.

[2] [n. d.]. Module: tf.keras.applications | TensorFlow v2.12.0. https://www.tensorflow.org/api_docs/python/tf/keras/applications.

[3] Ibtihaal M Hameed, Sadiq H Abdulhussain, and Basheera M Mahmmod. 2021. Content-based image retrieval: A review of recent trends. *Cogent Engineering* 8, 1 (2021), 1927469.

[4] Rabie Helaly, Seifeddine Messaoud, Soulef Bouaafia, Mohamed Ali Hajjaji, and Abdellatif Mtibaa. 2023. DTL-I-ResNet18: facial emotion recognition based on deep transfer learning and improved ResNet18. *Signal, Image and Video Processing* 17 (2023), 2731–2744. https://link.springer.com/article/10.1007/s11760-023-02490-6

[5] Afshan Latif, Aqsa Rasheed, Umer Sajid, Jameel Ahmed, Nouman Ali, Naeem Iqbal Ratyal, Bushra Zafar, Saadat Hanif Dar, Muhammad Sajid, Tehmina Khalil, et al. 2019. Content-based image retrieval and feature extraction: a comprehensive review. *Mathematical problems in engineering* 2019 (2019).

[6] LearnOpenCV. 2021. Implementing a CNN in TensorFlow Keras. https://learnopencv.com/implementing-cnn-tensorflow-keras/.

[7] Fuhui Long, Hongjiang Zhang, and David Dagan Feng. 2003. Fundamentals of content-based image retrieval. In *Multimedia Information Retrieval and Management: Technological Fundamentals and Applications*. Springer, 1–26.

[8] Microsoft. 2021. Build your own image classifier using Transfer Learning. https://learn.microsoft.com/en-us/cognitive-toolkit/build-your-own-image-classifier-using-transfer-learning.

[9] Neptune. 2021. Transfer Learning Guide: A Practical Tutorial With Examples for Images and Text in Keras. https://neptune.ai/blog/transfer-learning-guide-examples-for-images-and-text-in-keras.

[10] Pluralsight. 2021. Introduction to ResNet. https://www.pluralsight.com/guides/introduction-to-resnet.

[11] TensorFlow. 2021. Transfer learning and fine-tuning. https://www.tensorflow.org/tutorials/images/transfer_learning.

[12] Analytics Vidhya. 2021. Building a Convolutional Neural Network using TensorFlow Keras. https://www.analyticsvidhya.com/blog/2021/06/building-a-convolutional-neural-network-using-tensorflow-keras/.

[13] Shin'ya Yamaguchi, Sekitoshi Kanai, Atsutoshi Kumagai, Daiki Chijiwa, and Hisashi Kashima. 2022. Transfer Learning with Pre-trained Conditional Generative Models. *arXiv preprint arXiv:2204.12833* (2022). https://arxiv.org/abs/2204.12833

A APPENDIX

This appendix contains the following resources related to the project:

GitHub: The GitHub repository for the project provides version control and collaboration capabilities. You can find the source code, documentation, and project-related files in the repository. The GitHub link will allow you to browse, contribute, and review the project's codebase.

Github Link: Project Github link.