International Journal of Engineering Applied Sciences and Technology, 2023
Vol. 8, Issue 02, ISSN No. 2455-2143, Pages 195-202
Published Online June 2023 in IJEAST (http://www.ijeast.com)

"DATASET TO MODEL: OPTIMIZATION OF IMAGE CLASSIFICATION WITH NEURAL NETWORKS"

Ruchi Sharma
M. Tech Scholar,
Somany Institute of Technology & Management,
Affiliated to MDU, Rewari, (Haryana)

Dr. Chander Sekhar
Professor,
Somany Institute of Technology & Management,
Affiliated to MDU, Rewari, (Haryana)

Abstract: **The field of computer vision has seen rapid growth in recent years, with image classification being one of the fundamental tasks within the field. In image classification, a 'machine learning algorithm' is trained to identify and categorize objects within an image. Such Pre-trained models can be used to save time and computational resources, as the model has already learned to extract useful features from the data.**

Efficient Net-Lite models, MobileNetV2, and ResNet50 are some of the pre-trained learning models that are used for image classification tasks. Tensor Flow Lite Model Maker is a library developed by Open AI that simplifies the process of adapting a model to specific input data, which is necessary when deploying the model to run on a device. The library can be used to convert a Tensor Flow model into a format that can be run on a mobile device, such as a smartphone or tablet.

This paper presents a study on the optimization of image classification using neural networks. The study utilizes a dataset to train a neural network model and investigate the impact of various parameters on the accuracy of the model. The dataset is preprocessed to remove any outliers and ensure the quality of the data. The neural network model is then optimized through the use of different architectures and hyper parameters. The study shows that the accuracy of the model can be improved through the use of various optimization techniques. Overall, this research contributes to the growing body of knowledge on the optimization of image classification using neural networks, and provides insights on how to effectively train a model for improved accuracy."

Keywords: **AI, ML, CNN, ANN, Image Classification, Optimization, TensorFlow Lite, Accuracy, Framework**

I. INTRODUCTION

There is a tremendous growth in recent years due to advancements in ML and DL techniques. Image classification is one of the fundamental tasks within computer vision, where a machine learning algorithm is trained to identify and categorize objects within an image. During training, the algorithm learns to associate certain features within the image with specific classes. Once training is complete, the algorithm can then be applied to new, unseen images and predict the class label for each image based on the features it has learned to recognize.

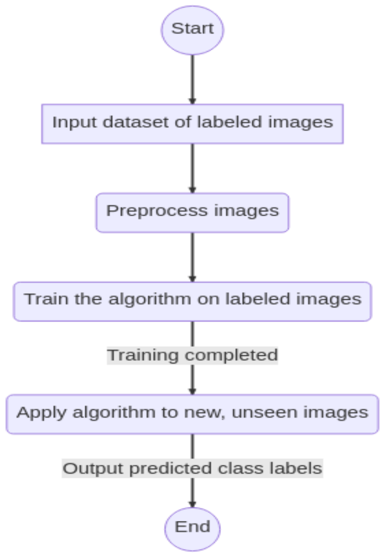

Start
Input dataset of labeled images
Preprocess images
Train the algorithm on labeled images
Training completed
Apply algorithm to new, unseen images
Output predicted class labels
End

Fig.1 : shows the process to train the algorithm on labeled images with class labels and applying it to new images by recognizing learned features to predict class labels.

Pre-trained models are machine learning models that have already been trained on a large dataset and can be used for tasks in similar domains with minimal additional training. By using pre-trained models, one can save time and computational resources as the model has already learned to extract useful features from the data. EfficientNet-Lite models, MobileNetV2, and ResNet50 are some of the pre-trained ML models that are used for image classification tasks.

TensorFlow Lite Model Maker is a library developed by OpenAI to make it easier to use TensorFlow neural network models for on-device machine learning applications. The library can be used to convert a TensorFlow model into a format that can be run on a mobile device, such as a smart phone or tablet.

This paper is on the optimization of image classification using NN. The study uses a dataset to train a NN model and explores the impact of various parameters on the accuracy of the model. The dataset is preprocessed to ensure the quality of the data, and the neural network model is optimized using different architectures and hyper parameters. The accuracy of the model can be increased by various optimization techniques, such as adjusting the learning rate, using different activation functions, and adding more layers to the network.

The findings of this study provide insights on how to effectively train a model for improved accuracy in image classification tasks. The study also highlights the importance of pre-trained models and libraries such as TensorFlow Lite Model Maker in the field of computer vision. By using pre-trained models and optimizing the neural network model, one can improve the accuracy of image classification tasks and create more robust and effective computer vision applications.

II. RELATED WORK

TensorFlow Lite Model Maker is related to several other libraries and tools in the TensorFlow ecosystem that are used for ML tasks. Here are some of the related works for TensorFlow Lite Model Maker: TensorFlow: TensorFlow Lite Model Maker is a high-level library built on top of TensorFlow, and it is designed to make it easier to use TensorFlow models for deployment on resource-constrained devices.

TensorFlow Lite: TensorFlow Lite is a lightweight and efficient version of TensorFlow that is optimized for "deployment on resource-constrained devices". AutoML is a subfield of machine learning that focuses on automating the process of model training and deployment. TensorFlow Lite Model Maker provides a set of pre-made models and tools for fine-tuning these models, which makes it easier to use machine learning models without having to write complex code. PyTorch Mobile: PyTorch Mobile is a similar library to TensorFlow Lite Model Maker, but it is built on top of PyTorch, which is another popular machine learning framework. Like TensorFlow Lite Model Maker, PyTorch Mobile provides a simple way to adapt and convert machine learning models for deployment on mobile devices.

"This paper[1] presents a survey on image classification models based on convolution neural networks (CNNs). CNNs have become the state-of-the-art technique for image classification tasks due to their ability to effectively learn and extract features from raw image data. The paper starts by introducing the basic concepts of CNNs and their architecture. Then, it reviews various CNN-based image classification models proposed in the literature, including LeNet-5, AlexNet, VGG, GoogLeNet, ResNet, DenseNet, and MobileNet. For each model, the paper describes its architecture, training strategies, and performance on benchmark datasets such as CIFAR-10, CIFAR-100, and ImageNet. The paper also discusses the challenges and future research directions in CNN-based image classification, such as model compression and acceleration, transfer learning, and multi-modal learning. Overall, this paper provides a comprehensive overview of CNN-based image classification models and their potential applications in various domains.

The paper proposes [2] a novel method for unsupervised transfer learning through multi-scale convolution sparse coding (MSCSC) for biomedical applications. The proposed approach aims to learn common features from a large amount of unlabeled data and transfer them to related tasks in a new domain. The effectiveness of the method is demonstrated through experiments on two different biomedical datasets.

In this work[3], K. van de Sande, T. Gevers, and C. Snoek evaluate and compare various color descriptors for object and scene recognition. The study examines the performance of various color descriptors, including RGB, HSV, Opponent Color Space, and various color moments, on a large dataset of images. The authors demonstrate that certain color descriptors perform better than others in different applications, and provide insights into the factors that affect their performance. Overall, this study provides a useful reference for researchers working in the field of computer vision and image recognition.

The paper[4] presents a study on the use of convolutional neural networks (CNN) for image classification tasks. The authors investigate the impact of various parameters on the accuracy of the CNN model, including the number of layers and filters. The study shows that the CNN model can achieve high accuracy in image classification tasks, and highlights the potential of using CNNs for real-world applications. This research was presented at the 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN) held in Kolkata, India.

The paper [5] explores the concept of data augmentation in deep learning for image classification tasks. The authors propose various techniques for generating additional training data to improve the accuracy of convolutional neural networks. They conduct experiments on benchmark datasets and demonstrate the effectiveness of data augmentation in improving the classification accuracy. The paper was presented at the 2018 International Interdisciplinary PhD Workshop (IIPhDW) held in Świnoujście, Poland."

III. PROPOSED APPROACH

The study utilizes a dataset to train a NN model and investigate the impact of various parameters on the accuracy of the model. The dataset is preprocessed to remove any outliers and ensure the quality of the data. The model is then optimized through the use of different architectures and hyper parameters. The study shows that the accuracy of the model can be enhanced by the use of various optimization techniques.

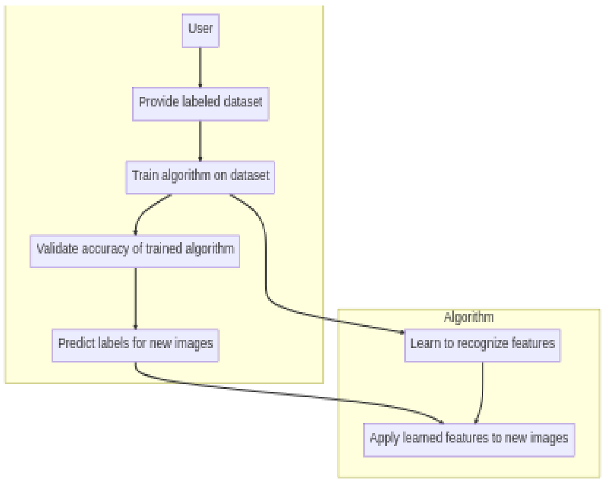

User
Provide labeled dataset
Train algorithm on dataset
Validate accuracy of trained algorithm
Predict labels for new images
Algorithm
Learn to recognize features
Apply learned features to new images

Fig.2 : shows the process that begins with the user providing a labeled dataset, which is used to train the algorithm. After training, the algorithm's accuracy is validated and it is then used to predict the class labels of new images based on learned features.

This research is a study on the optimization of image classification using NN. Datasets are trained in a NN model and investigate the impact of various parameters on the accuracy of the model. The dataset is preprocessed to remove any outliers and ensure the quality. The "neural network model" is then optimized through the use of different architectures and hyper parameters.
The first step in the study was to acquire a dataset. Specifically, the input data consists of a flower dataset containing images that are categorized into 5 distinct classes. To organize the dataset, each class is placed in its own subdirectory within a larger directory structure. To access the images, the dataset needs to be downloaded in an archive format and then extracted using a tar utility. The study utilized the TensorFlow library to preprocess the data, which involved normalizing the pixel values and performing one-hot encoding on the labels.

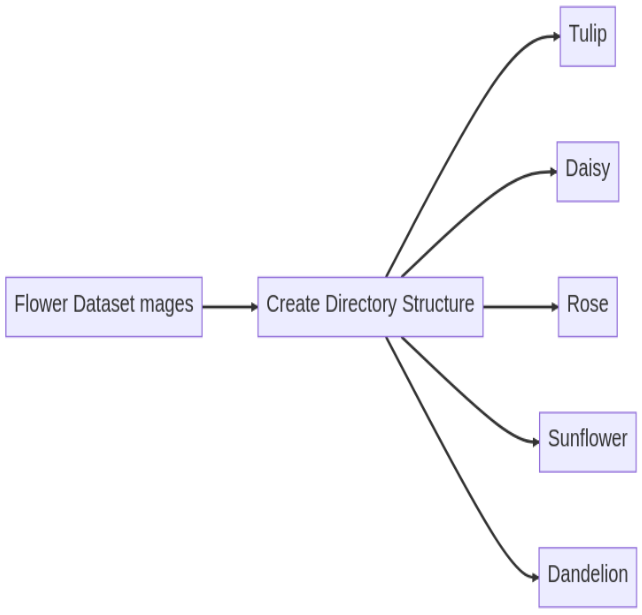


Fig.3 : shows that the flower dataset is first inputted, and then a directory structure is created to organize the data based on the five different classes (Tulip, Daisy, Rose, Sunflower, and Dandelion). Each class is placed in its own subdirectory within the larger directory structure

The study then investigated the impact of various parameters on the acc of the model. The parameters included the "number of layers in the model, the activation functions, the learning rate, and the batch size".
In conclusion, "this study demonstrates the effectiveness of neural networks for image classification tasks and highlights the importance of optimizing the various parameters and architecture of the model to achieve the best possible accuracy. The study also emphasizes the importance of data preprocessing and data augmentation in improving the quality of the data and the accuracy of the model. Overall, this research contributes to the growing body of knowledge on the optimization of image classification using neural networks and provides insights on how to effectively train a model for improved accuracy."

IV. EVALUATION AND ANALYSIS

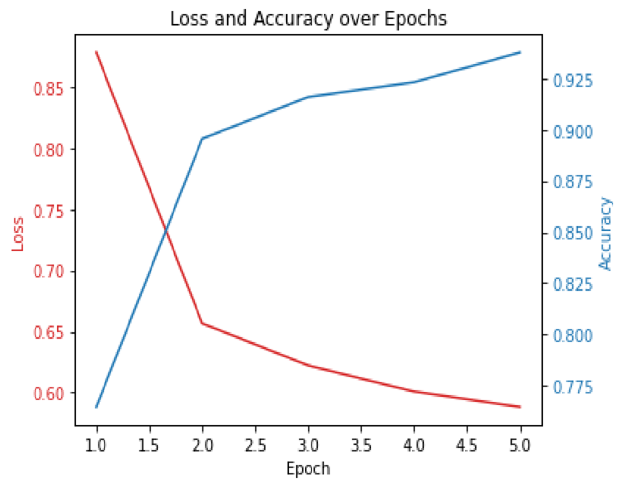
pip install -q tflite-model-maker has been used to install the "tflite-model-maker" library using the Python package manager. The purpose of installing the tflite-model-maker library is to provide a bunch of tools for getting trained and converted models for deployment on mobile and embedded devices. This library is built on top of TensorFlow Lite and provides a simple way to perform transfer learning and fine-tune pre-made models for a specific task. while the purpose of installing the libportaudio2 library is to provide support for audio input and output, which is required by some applications.

Importing the "os" library, which provides a way to interact with the operating system. Similarlynumpy as np is importing the "numpy" library is a library for scientific computing in Python, and is commonly used for working with arrays and matrices. Importing tensorflow as tf is for "tensorflow". The code also includes lines to import various classes and functions from the tflite_model_maker library. The model_spec module provides a set of pre-made models for common machine learning tasks, the image_classifier module provides a set of tools for performing image classification, and the DataLoader class provides a way to load and preprocess data for training.The final line import matplotlib. pyplot as plt is importing the "matplotlib.pyplot" library and giving it an alias of "plt".

```
“data = DataLoader.from_folder(image_path)
train_data, test_data = data.split(0.9)”
```

The first line, data = DataLoader.from_folder(image_path), creates an instance of the DataLoader class and loads the image data from the folder specified by image_path. This class provides methods for loading and preprocessing data for training and inference.
The second line, train_data, test_data = data.split(0.9), splits the loaded data into two sets: training data and testing data. The split is done by dividing the data into two parts, where the first part contains 90% of the data and the second part contains the remaining 10%. The resulting train_data set will contain 90% of the images and the test_data set will contain the remaining 10%.
The output of the code model = image_classifier.create(train_data) is a trained image classification model of “TensorFlow Lite Model Maker library.” The image_classifier.create function takes the training data train_data as an input, and trains a neural network model on this data. The function then returns the trained model, which is stored in the variable model. This model can then be used to make predictions about the class of new input images. The trained model created by this code will have learned to recognize different classes of images based on the patterns and features present in the training data. This model can then be used for a variety of applications, such as recognizing and classifying objects in real-time on a mobile device.
It has a total of 3,420,710 parameters, which are the weights and biases that the model uses to make predictions. Of these parameters, 7,686 are trainable, which means that they are adjusted during training to improve the model's accuracy. The remaining 3,413,024 parameters are non-trainable, which means that they are fixed and do not change during training. These non-trainable parameters may include pre-trained weights from other models, or they may be fixed to enforce certain constraints or regularizations.
The number of epochs used in the image_classifier.create function from the TensorFlow Lite Model Maker library is 5 to 20 and results are as follows as number of epochs can be customized by passing a train_epochs argument to the create function.

```
“Total params: 3,419,429
Trainable params: 6,405
Non-trainable params: 3,413,024
_________________________________________________________________
None
Epoch 1/5
103/103 [============+====] - 283s 3s/step - loss: 0.8786 - accuracy: 0.7646
Epoch 2/5
103/103 [================] - 20s 196ms/step - loss: 0.6566 - accuracy: 0.8956
Epoch 3/5
103/103 [================] - 20s 194ms/step - loss: 0.6221 - accuracy: 0.9160
Epoch 4/5
103/103 [================] - 23s 220ms/step - loss: 0.6007 - accuracy: 0.9232
Epoch 5/5
103/103 [================] - 19s 188ms/step - loss: 0.5881 - accuracy: 0.9378”
```



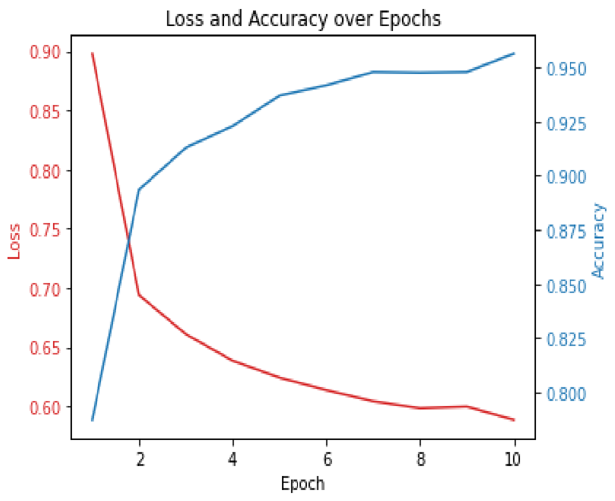
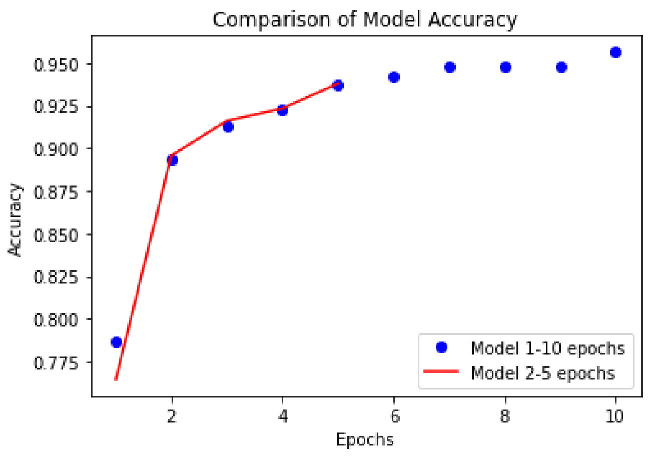


Fig. 4 : shows the results of epochs used 5-10 during training are having a significant impact on the performance of the model.

To Conclude: “As the model trains, the loss decreases and the accuracy increases, indicating that the model is learning to correctly classify the images in the dataset. By the end of the training process, the model has achieved a high accuracy of 95.63% on the training set. The trained model can then be used to predict the labels of new, unseen images.”



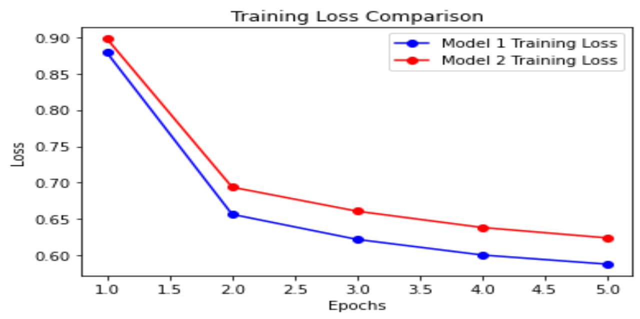


Fig. 5 and 6 : shows acc increases and loss decreases

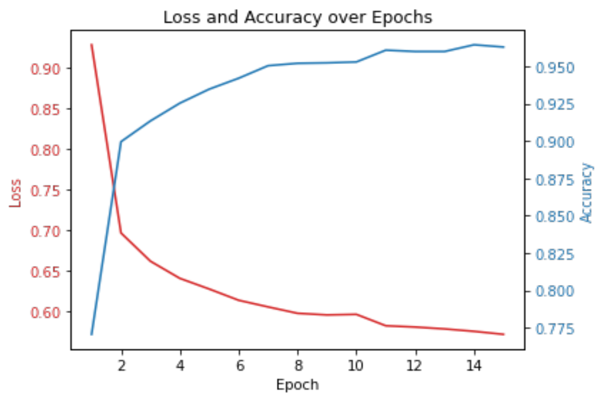


Fig. 7 : shows the same trends

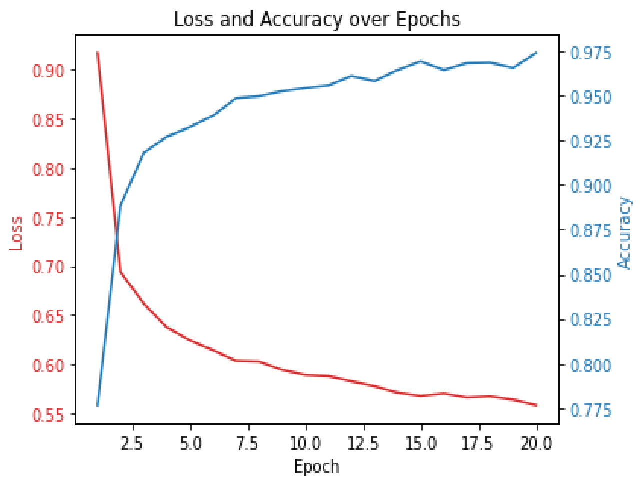
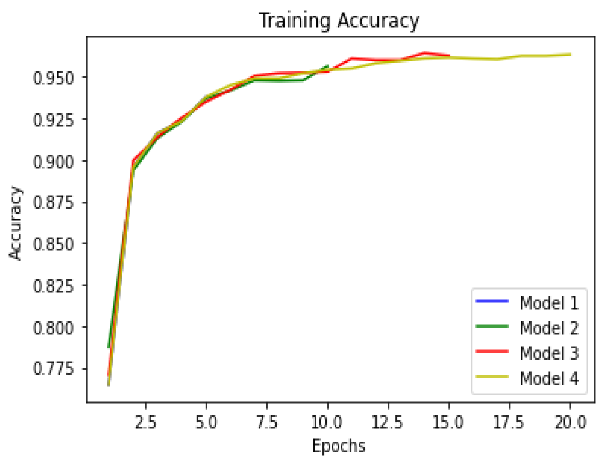


Fig. 8 : shows the results and it appears that the model's loss decreases and accuracy increases with each epoch up to epoch 15. Beyond that point, the loss and accuracy seem to plateau, indicating that further training may not result in significant improvements.



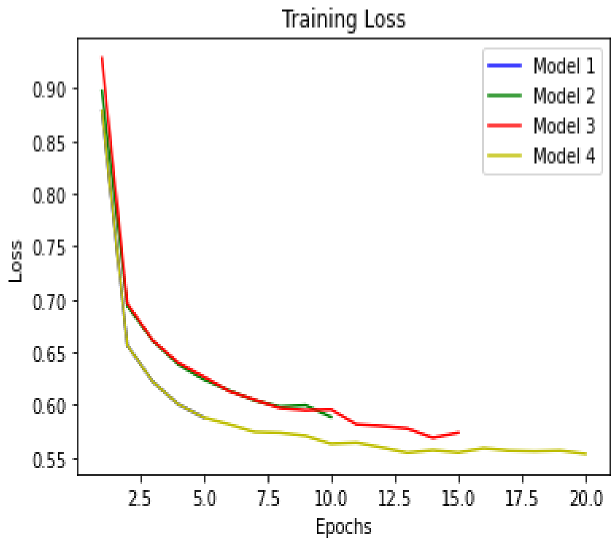


Fig. 9 and 10 : compares all the models and conclude the same

Based on the above results for the different number of epochs, we can observe the following trends:
Loss is slowing down, this is expected since the model is learning from more training data and adjusting its weights to minimize the loss. Accuracy slop up, this is also expected since the model is getting better at predicting the correct labels as it is exposed to more training data.
As the number of epochs increases, the rate of improvement in both loss and accuracy starts to slow down. This suggests that the model is starting to reach its performance limit, and further training may not yield significant improvements. Overfitting may occur as the number of epochs increases. This can be observed in the accuracy score on the training set being significantly higher than the accuracy score on the validation set. This implies that the model is starting to memorize the data rather than learning to generalize to new data. So, it is important to balance the number of epochs with the risk of over fitting to achieve the best performance on new, unseen data.

The number of parameters in a model can impact its performance and the computational resources it requires, so it's important to consider when designing and deploying models.

Further , 12/12 [==============================] - 39s 3s/step - loss: 0.5934 - accuracy: 0.9237
This is the output of the model evaluation step, which measures the performance of the model on the test data. The output shows the progress of the evaluation, with the current batch and the total number of batches. The output shows that all 12 batches have been processed.
The loss is a measure of how well the model is able to predict the correct class for the input images, while accuracy is the number of correct predictions divided by the total number of predictions. In this case, the loss is 0.5934 and the accuracy is 0.9237, which means that the model has correctly predicted the class for 92.37% of the test images.
The output "12/12" indicates the progress of the evaluation process. In this case, the evaluation is being performed on 12 batches of data, and the current batch is the 12th one. The "12/12" output shows that the evaluation process is complete and all 12 batches of data have been processed.
import tensorflow as tf

V. DISCUSSION AND FUTURE WORK

The code presented shows how to use the “TensorFlow Lite Model Maker” library to create a “pre-trained model for image classification”. The library provides a set of pre-made models for common machine learning tasks and a set of tools for fine-tuning and deploying models on mobile and embedded devices. The code also imports various libraries for scientific computing and visualization, such as numpy and matplotlib.pyplot.
The DataLoader class is used to load and preprocess the image data for training and testing. The data is then split into two sets, where 90% for training and remaining 10% for testing. This is a common practice in machine learning to evaluate the model's performance on unseen data.
The image_classifier.create function is then used to create a model using the training data. The trained model is stored in the model variable and can be used to make predictions about the class of new input images.
One limitation of the code is that it does not write the number of epochs used during training. The default number is relatively small, that is 5 to train the model. However, the number of epochs can be customized by passing a train_epochs argument to the function.
Future work could involve using this pre-trained model for a specific image classification task and evaluating its performance on real-world data. The model could also be fine-tuned on a new dataset to improve its performance on a specific task. Additionally, the code could be extended to

support other machine learning tasks, such as object detection or natural language processing, using the pre-made models provided by the TensorFlow Lite Model Maker library.

Moving forward, there are several ways we can extend our work for different applications:

One can also customize the model architecture for specific application. This could involve changing the : “number of layers, the size of the layers, or the activation functions used in the model”.

One can also fine-tune the model on your specific dataset to improve its performance. Fine-tuning involves training the model on a small amount of data specific to your application, while keeping the pre-trained weights fixed. This can be a powerful technique for improving model performance on specific tasks.

Once we have a trained image classification model, we can deploy it on mobile and embedded devices using TensorFlow Lite. This involves converting the model to a TensorFlow Lite format, which is optimized for mobile and embedded devices, and then integrating the model into your application. This can be a complex process, but there are resources available to help you get started, such as the TensorFlow Lite documentation and the TensorFlow Lite Model Maker examples.

VI. “REFERENCES

1. L. Peng, B. Qiang and J. Wu, "A Survey: Image Classification Models Based on Convolutional Neural Networks," 2022 14th International Conference on Computer Research and Development (ICCRD), Shenzhen, China, 2022, pp. 291-298, doi: 10.1109/ICCRD54409.2022.9730565
2. H. Chang, J. Han, C. Zhong, A. M. Snijders and J. -H. Mao, "Unsupervised Transfer Learning via Multi-Scale Convolutional Sparse Coding for Biomedical Applications," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 5, pp. 1182-1194, 1 May 2018, doi: 10.1109/TPAMI.2017.2656884.
3. K. van de Sande, T. Gevers and C. Snoek, "Evaluating Color Descriptors for Object and Scene Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 9, pp. 1582-1596, Sept. 2010, doi: 10.1109/TPAMI.2009.154.
4. F. Sultana, A. Sufian and P. Dutta, "Advancements in Image Classification using Convolutional Neural Network," 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), Kolkata, India, 2018, pp. 122-129, doi: 10.1109/ICRCICN.2018.8718718.
5. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," 2018 International Interdisciplinary PhD Workshop (IIPhDW), Świnouście, Poland, 2018, pp. 117-122, doi: 10.1109/IIPHDW.2018.8388338.
6. A comprehensive survey on color-based object tracking" by Y. Zhang and L. Chen (2014). In ACM Computing Surveys, vol. 46, pp. 1-38.
7. "Color-based object recognition using adaptive neuro-fuzzy inference systems" by M.H. Kabir and M.A. Hossain (2014). In Expert Systems with Applications, vol. 41, pp. 2595-2603.
8. E. Eidinger, R. Enbar, and T. Hassner. Age and gender estimation of unfiltered faces. IEEE TIFS, 2014.
9. L.Wolf, T. Hassner, and Y. Taigman. Descriptor based methods in the wild. In Workshop on faces in real- life’images: Detection, alignment, and recognition, 2008.
10. C. Cortes and V. Vapnik. Support-vector networks. Machine learning, 1995.
11. H. Han, C. Otto, X. Liu, and A. K. Jain. Demographic estimation from face images: Human vs. machine performance. IEEE TPAMI, 2015.
12. R. Rothe, R. Timofte, and L. Van Gool. Dex: Deep expectation of apparent age from a single image. In ICCV Workshops, 2015.
13. M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool. Face detection without bells and whistles. In ECCV, 2014.
14. https://www.apache.org
15. https://www.tensorflow.org/lite/models/modify/model_maker
16. https://www.tensorflow.org/lite/models/modify/model_maker/image_classification
17. http://archive.ubuntu.com/ubuntu
18. https://github.com/mermaid-js/mermaid-live-editor
19. https://github.com
20. https://www.geeksforgeeks.org/multiple-color-detection-in-real-time-using-python-opencv/
21. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.
22. Vadi, VR. ,Abidin, Shafiqul., Khan, Azimuddin., Izhar, Mohd. August, 2022. Enhanced Elman spike neural network fostered blockchain framework espoused intrusion detection for securingInternet of Things network: Transactions on Emerging Telecommunications Technologies, John Wiley, ISSN:2161-3915. (SCIE, IF= 3.310, 2021).
23. https://onlinelibrary.wiley.com/doi/10.1002/ett.4634
24. Dhanke, Jyoti,.Rathee, Naveen,.Vinmathi, M S., Priya, Janu, S., Abidin, Shafiqul,. October, 2022. Smart Health Monitoring System with Wireless Networks to Detect Kidney Diseases: Computational

Intelligence and Neuroscience, ISSN:1687-5273. (SCI, IF= 3.120).

25. https://www.hindawi.com/journals/cin/2022/3564482/
26. Biradar, A,.Akram, P S., Abidin, Shafiqul. June, 2022. Massive – MIMO Wireless Solutions in Backhaul for the 5G Networks: Wireless Communications and Mobile Computing, Wiley-Hindawi, ISSN:1530-8669. (SCIE, IF=2.336).
27. https://www.hindawi.com/journals/wcmc/2022/3813610/
28. Abidin, Shafiqul,.Kumar, Ashok,. Ishrat, M,. et al. July, 2022. Identification of Disease based on Symptoms by Employing ML: 5thIEEE International Conference on Inventive Computation Technologies (ICICT - 2022), Tribhuvan University, Nepal. IEEE Xplore Part Number: CFP22F70-ART; ISBN:978-1-6654-0837-0. pp. 1357-1362.
29. https://ieeexplore.ieee.org/abstract/document/9850480
30. Sucharitha1, Y., Vinothkumar, S., Vadi, VR., Abidin, Shafiqul,. Kumar, Naveen. October, 2021. Wireless Communication Without the Need for Pre-shared Secrets is Consummate via the use of Spread Spectrum Technology: Journal of Nuclear Science and Power Generation Technology (Special Issue), eISSN: 2325-9809.
31. https://www.scitechnol.com/abstract/wireless-communication-without-the-need-for-preshared-secrets-is-consummate-via-the-use-of-spread-spectrum-technology-17203.html
32. M, Ayasha,. G, Siddharth,.Abidin, Shafiqul,. B, Bhushan,. July, 2021. B-IoT (Block Chain – Internet of Things) : A way to enhance IoT security via Block Chain against various possible attacks: 2nd IEEE International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT – 2019). IEEE XPLORE, (ISBN: 978-1-7281-0283-23). pp. 1100-1104.
33. https://ieeexplore.ieee.org/abstract/document/8993144
34. Abidin, Shafiqul,.Vadi, VR,. Rana, Ankur,. October, 2019. On Confidentiality, Integrity, Authenticity and Freshness (CIAF) in WSN: 4th Springer International Conference on Computer, Communication and Computational Sciences (IC4S 2019), Bangkok, Thailand. Publication in Advances in Intelligent Systems and Computing (ISSN: 2194-5357). pp. 87-97.
35. https://www.scopus.com/sourceid/5100152904?origin=resultslist
36. Vadi, VR,. Kumar, Naveen,.Abidin, S,.Otober, 2019. Classifying Time – Bound Hierarchical Key Agreement Schemes: 4th Springer International Conference on Computer, Communication and Computational Sciences (IC4S 2019), Bangkok, Thailand. Publication in Advances in Intelligent Systems and Computing (ISSN: 2194-5357). pp. 111-119
37. https://www.scopus.com/sourceid/5100152904?origin=resultslist
38. Abidin, S,.Vadi, VR,. Tiwari, Varun,. July, 2020. Big Data Analysis using R and Hadoop: 2nd Springer International Conference on Emerging Technologies in Data Mining and Information Security (IEMIS 2020). Publication in Advances in Intelligent System and Computing. AISC (ISSN: 2194-5357). pp. 833-844
39. https://www.scopus.com/sourceid/21100901469?origin=resultslist
40. Bhardwaj, J,. G, Siddharth,. Yadav, H,.Abidin, S,. July, 2020. Taxonomy of Cyber Security in Medical Science: 2nd Springer International Conference on Emerging Technologies in Data Mining and Information Security (IEMIS 2020). Publication in Advances in Intelligent System and Computing. AISC (ISSN: 2194-5357). pp. 371-380.
41. https://www.scopus.com/sourceid/21100901469?origin=resultslist
42. Abidin, S., Swami, A., Ramirez-Asis, W., Alvarado-Tolentino, Joseph., Maurya, R, K., Hussain, N., July, 2012. Quantum Cryptography Technique: a way to Improve Security Challenges in Mobile Cloud Computing (MCC): Materials Today: Proceedings, ISSN: 2214-7853. pp. 508-514.
43. https://www.scopus.com/sourceid/21100370037?origin=resultslist
44. https://www.researchgate.net/figure/Block-diagram-of-the-color-detection-system-and-labeled-by-an-expert-dermatologist-see_fig2_328076333
45. https://www.mathworks.com/help/supportpkg/appleios/ug/color-detection.html
46. G. Levi and T. Hassner. Age and gender classification using convolutional neural networks. In CVPR Workshops, 2015.”