Image Colorization using Convolutional Neural Network

Mamata Poudel[1], Rajesh Nepal[2], Sagun Acharya[3], Krishma Manandhar[4], Bharat Bhatta[5]

Sagarmatha Engineering College, Tribhuvan University, Sanepa, Lalitpur

Abstract
With its advent and extension in many areas, technology has become crucially integrated with human lifestyle and activities. Various branches of technology have found ways to progress and they serve imperatively in various fields of development, entertainment and utility. Artificial Intelligence and domains within it encompass a huge portion of technological applications, and deep learning being a subset of AI can be used for many such implementations. Image colourization is one such domain which can utilize the deep learning capabilities of a machine to effectively produce results. Colorization of gray-scale images is generally carried out using photo editing software and is a tedious and expensive job.With the collaboration of different pre-trained models in a baseline CNN model,, we analyzed the one with the highest accuracy and the least loss,comparing it to the others.Based on evaluation metrics such as root mean square error, loss, and accuracy, we determined that the best model is the one incorporating EFFicientB0. Hence, our project makes this process automated and accurate utilizing the convolutional neural architecture combined with EfficientNetB0. In this project, we have realized and implemented several compositions of the baseline CNN model with pre-trained models to deduce the best version.

Keywords: Deep Learning, Image colorization, EfficientNetB0, CNN, Convolution neural architecture

1. Introduction

Photographs provide valuable information about history, emotions, and cultural heritage. Color photography emerged in the mid-19th century, but colorizing grayscale images remains a challenging and time-consuming process. Our project aims to develop an efficient application that automatically colorizes historical grayscale images using convolutional neural networks (CNNs). Colorizing grayscale images accurately and conveniently is currently a costly and labor-intensive procedure. In Nepal, the rich history captured in black and white photographs remains unexplored in terms of color. Additionally, there is a lack of easily accessible and affordable image colorization options in the Nepalese digital market. Our project addresses these challenges by developing an application that provides accurate and accessible colorization of historical photographs. The project's objectives are to compare and evaluate the performance of different pre-trained models combined with a baseline CNN model for image colorization. Additionally, the project aims to transform grayscale historical photographs of Nepal into three-channel-colored images.
Our project's main feature is the ability to colorize grayscale images of historical importance, providing a visual transformation that enhances their significance.

The technical feasibility of our project relies on utilizing Convolutional Neural Networks (CNNs) trained with authentic datasets. This allows for efficient and accurate colorization of grayscale images. The project's operational feasibility lies in its ability to contribute to the preservation and celebration of history by providing an automated and cost-effective solution for colorizing historical photographs. By addressing the lack of accessible and affordable image colorization options in the Nepalese digital market, our project fills a significant gap and offers a valuable solution within the ML domain.

2. Literature Review

Colorization of gray-scale images using CNN has been extensively studied, with different approaches proposed [2]. Scribble-based and example-based methods have limitations in terms of color blending and dependence on human skills [2]. Learning-based approaches, particularly CNN models, have shown effectiveness in image colorization[2]. Some networks focus on producing plausible colorizations rather than true ground truth colors [3]. Our project is inspired by these models and aims to enhance accuracy using an EfficientNetB0 model [3]. Various research articles have proposed colorization approaches using different datasets, including heritage images [2]. We acquired our dataset from various websites on the internet using web scraping. Our project will hence implement the understandings gained from these information and experiences to aim towards higher accuracy of output, targeting historical images from Nepal.

3. Methodology

3.1.1. Preparation of Dataset

The data required to train the model was collected from a variety of online resources using web scraping i.e Beautiful Soup library. No. of images we trained is 1448.

3.1.2. Preprocessing of Dataset

The collected images were first resized to the dimension of 256*256 to be fed into the model. Moreover, the colored RGB images were converted to a lab model by converting it to CIELab color space. A CIELab color image also has three channels where L stands for luminance or lightness, a* and b* for the four unique colors of human vision. We opted for CIELab since the L channel can be set to gray-scale for all images and we only have to predict the a and b channels. Next, the image was branched into two images where one is gray-scale (L-channel) and the other having a and b components. During the conversion to lab model, we can consider that the L component is the input and the a and b components are the required output.

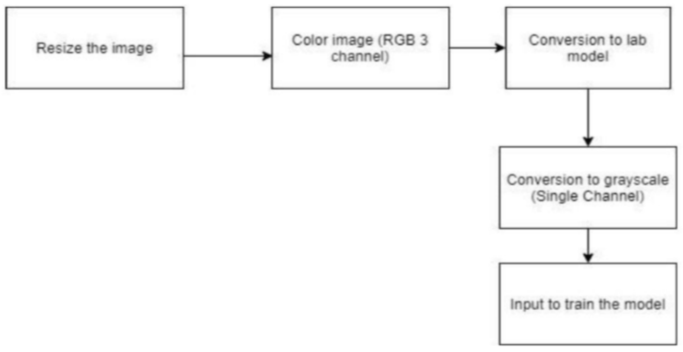
Resize the image
Color image (RGB 3 channel)
Conversion to lab model
Conversion to grayscale (Single Channel)
Input to train the model

Fig 3.1: Preprocessing of dataset

3.1.2. Splitting the Dataset for Model Training

The collected images were first resized to the dimension of 256*256 to be fed into the model. Moreover, the colored RGB images were converted to a lab model by converting it to CIELab color space. A CIELab color image also has three channels where L stands for luminance or lightness, a* and b* for the four unique colors of human vision. We opted for CIELab since the L channel can be set to gray-scale for all images and we only have to predict the a and b channels. Next, the image was branched into two images where one is gray-scale (L-channel) and the other having a and b components. During the conversion to lab model, we can consider that the L component is the input and the a and b components are the required output.

The dataset was then splitted into three different categories to train our model.
The three categories are:

- Training data
- Validation data
- Test data

3.2. Model Building (Training)

To increase the efficiency and accuracy of the baseline working CNN model, we had to incorporate a pre-trained model. The need for the addition of a pre-trained model to our architecture is further defined by the following reasons:

To solve any complex problem, we needed to add more layers in the Deep Neural Network which would help to enhance the accuracy and performance. These layers help the model learn more complex features. For example, in the case of recognizing images, the first layer may learn to detect edges, the second layer may learn to identify textures and similarly the third layer may learn to detect objects and so on.

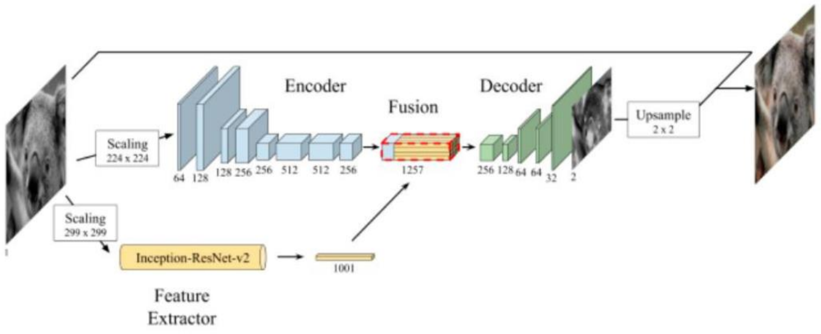

Fig 3.2: Baseline System Architecture (source: https://lcsrg.me/deep-koalarization/)

But, the addition of layers would also result in a greater percentage of error. Most pre-trained models solve this problem as they are made up of Residual Blocks. Hence, we incorporated 6 different pre-trained models to our baseline CNN model in order to determine the one with the highest accuracy and performance. The 6 different pre-trained models that we used to produce a comparative analysis are: EfficientNetB0, Inception-Resnet-V2, Inceptionv3, VGG16, VGG19 and Resnet50.

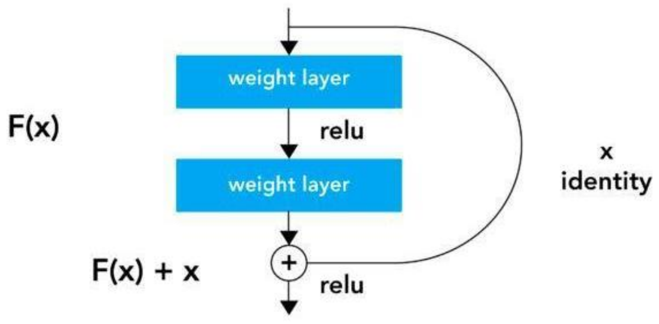

Fig 3.3: Residual Learning
(source:https://www.researchgate.net/figure/Residual-Learning-Building-Block-26_fig2_350957703)

The very first thing we noticed to be different was that there was a direct connection which skips some layers (may vary in different models) in between. This connection is called 'skip connection' and is the core of residual blocks. Due to this skip connection, the output of the layer is not the same now. Without using this skip connection, the input 'x' gets multiplied by the weights of the layer followed by adding a bias term. Next, this term goes through the activation function, f(x) and we get our output as H(x).

$$H(x) = f\,(wx + b) \qquad \text{(Equation 1)}$$

or,

$$H(x) = f(x) \qquad \text{(Equation 2)}$$

Now with the introduction of skip connection, the output is changed to

$$H(x) = f(x) + x \qquad \text{(Equation 3)}$$

We have implemented all 6 models and obtained conclusions about their performance (depicted by the graphs in Chapter 5: Result and Analysis). Out of all 6 models, the highest accuracy and performance was produced by EfficientNetB0.

3.2.1. System Architecture

The units of our system and their functionalities are described as follows:

- **Preprocessing**

The gray-scale image is resized or scaled according to the model.

- **Encoder**

The encoder extracts the low-level features and middle-level features from the image that is taken as input after preprocessing.

- **Decoder**

This is the exact opposite orientation of the encoder which works to get the second final output of the image. After Up-sampling (the resulting output will be increased by twice), we get the final colorized image which concatenates the final chrominance map to the luminance component of the input image.

- **Feature Extractor**

It extracts the high -level features from the input (scaled) image.

- **Fusion layer**

It merges the low-level, middle-level and high-level features to get the con-catenated features of the image.

The 6 different pre-trained layers that we used took different dimensions of input im-age; the input dimension for VGG19, VGG16 and Resnet50 network was 224x224x3 whereas that for EfficientNetB0, ResnetV2 and Inception V3 was 299x299x3.

3.2.2. Role of the Pre-trained Layer: EfficientNetB0

The skip connections in EfficientNetB0 solved the problem of vanishing gradient in deep neural networks by allowing this alternate shortcut path for the gradient to flow through. The other way that these connections helped is by allowing the model to learn the identity functions which ensured that the higher layer will perform at least as good as the lower layer, and not worse. Explaining this further, say, we have a shallow network and a deep network that maps an input ‘x’ to output ’y’ by using the function H(x). We want the deep network to perform at least as good as the shallow network and not degrade the performance as in case of plain neural networks (without residual blocks). One way of achieving so is if the additional layers in a deep network learn the identity function and thus their output equals inputs which do not allow them to degrade the performance even with extra layers. Residual blocks made it exceptionally easy for layers to learn identity functions as seen by the formulas. In plain networks the output would be:

$$H(x) = f(x) \qquad \text{(Equation 4)}$$

$$f(x) = 0 \qquad \text{(Equation 5)}$$

$$H(x) = x \qquad \text{(Equation 6)}$$

All we needed is to make f(x)=0 which is easier and we got x as output which was also our input. In the best-case scenario, additional layers of the deep neural network can better approximate the mapping of ‘x’ to output ‘y’ than it’s the shallower counterpart and reduce the error by a significant margin.

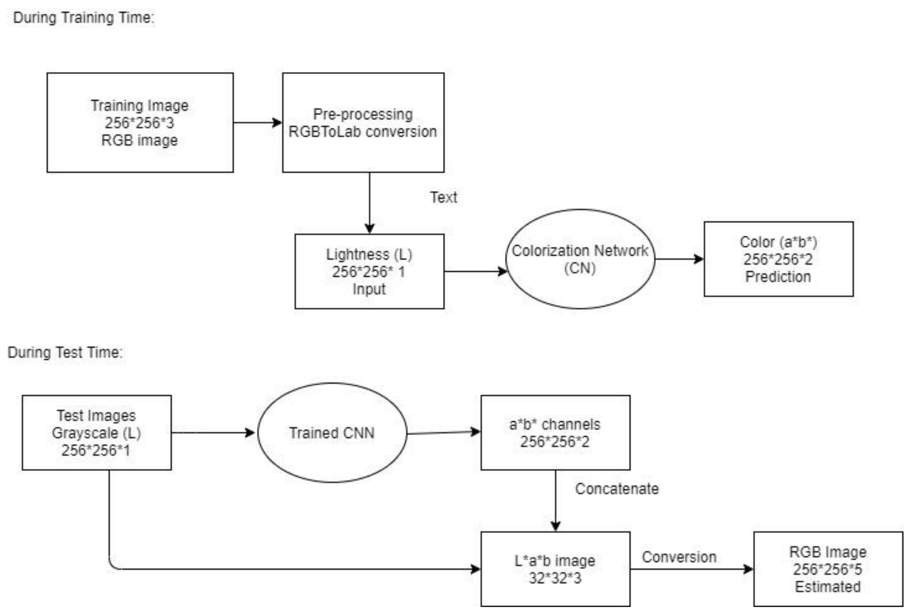
During Training Time:
Training Image 256*256*3 RGB image
Pre-processing RGBToLab conversion
Text
Lightness (L) 256*256* 1 Input
Colorization Network (CN)
Color (a*b*) 256*256*2 Prediction
During Test Time:
Test Images Grayscale (L) 256*256*1
Trained CNN
a*b* channels 256*256*2
Concatenate
L*a*b image 32*32*3
Conversion
RGB Image 256*256*5 Estimated

Fig 3.4: Training and Testing of model

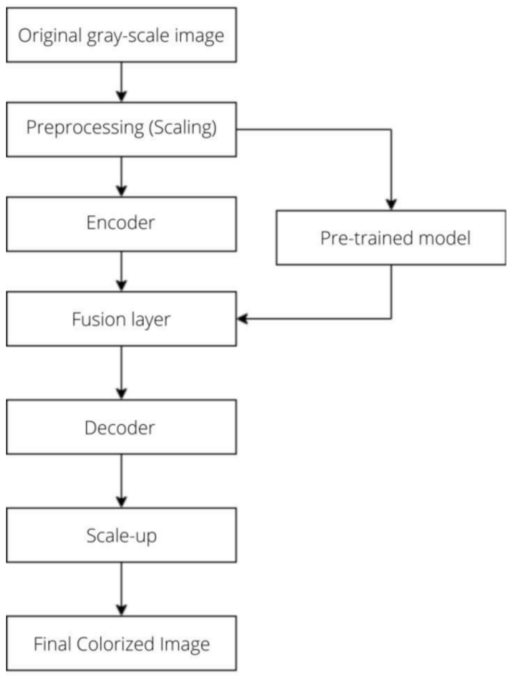

Fig 3.5: System Architecture

EfficientNetB0 baseline model takes in an input image with a 224x224x3 dimension. The model then extracts features throughout the layers by using multiple convolutional layers. Compared to other DCNNs, EfficientNetB0 scales each dimension using a fixed set of scaling coefficients. This approach surpasses other state-of-the-art models that are trained on the ImageNet dataset . We run our final model with the incorporated EfficientNetB0 for 120 epochs with a learning rate of 0.0001.

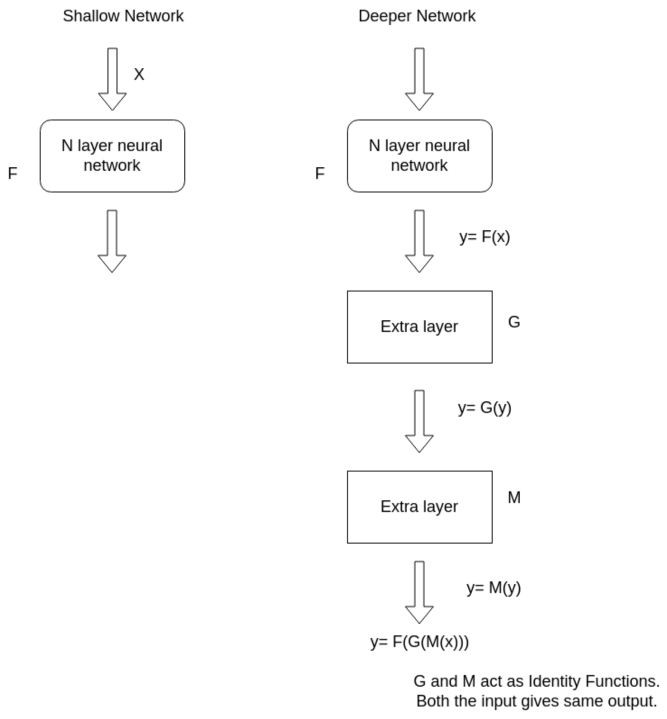

Fig 3.6: How the Additional Layer (EfficientNetB0) Helps

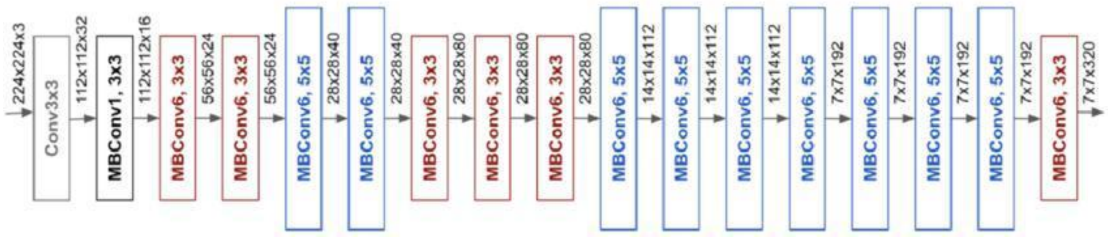

Fig 3.7: EfficientNetB0 Baseline Model Architecture
(source:researchgate.net/figure/EfficientNetB0-baseline-model-architecture-33_fig2_348915715)

The block diagram that follows depicts the baseline CNN architecture incorporated with EfficientNetB0. It shows the different convolution layers along with the basic functions that are used to process the dataset.

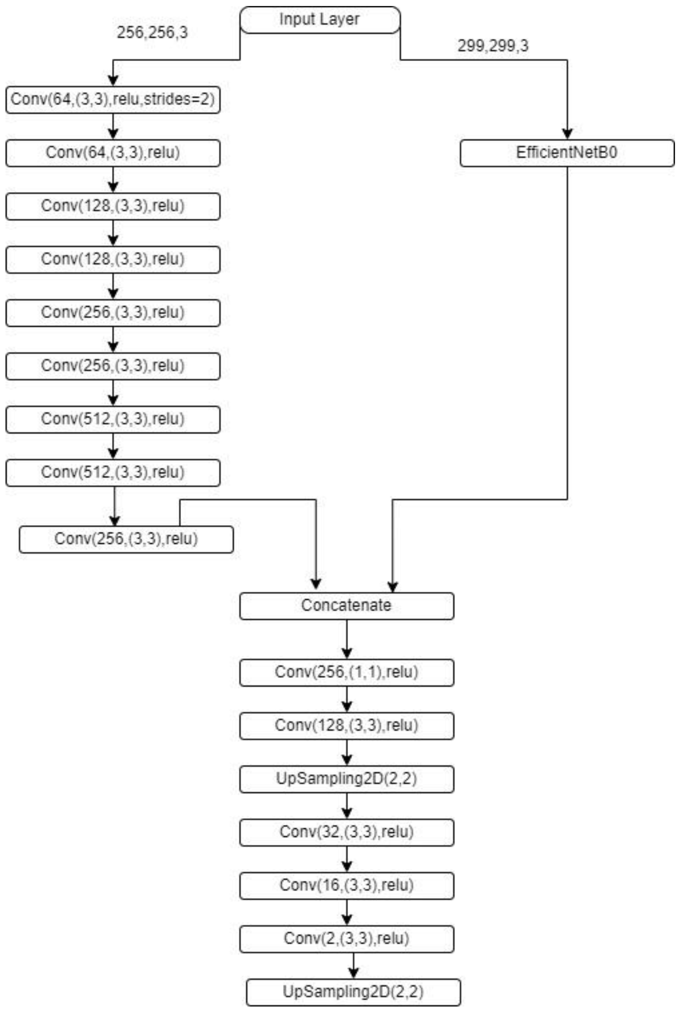

Fig 3.8: Baseline CNN with EfficientNetB0

4. Result and Analysis

4.1. Experiment Setup

We created a baseline CNN model to be incorporated with other pre-trained models in order to enhance the accuracy of the result. The six pre-trained models that we implemented are: EfficientNetB0, Inception Resnetv2, Inceptionv3, VGG16, VGG19 and Resnet50.

4.2. Evaluation Metrics

The evaluation metrics that we observed and based the evaluation of our results on are training and validation loss, training and validation accuracy, training and validation root mean squared error, training loss, Validation loss, training accuracy, Validation accuracy, Training root mean squared error, validation root mean squared error.

The number of epochs we ran to train, validate and test these models was 20.

4.3. Experiments/Results

The evaluation matrices were plotted against the number of epochs to determine the result. The graphs that depict the plots for all the 6 pre-trained models are illustrated individually as follows:

4.3.1. Training and Validation Accuracy vs Epoch

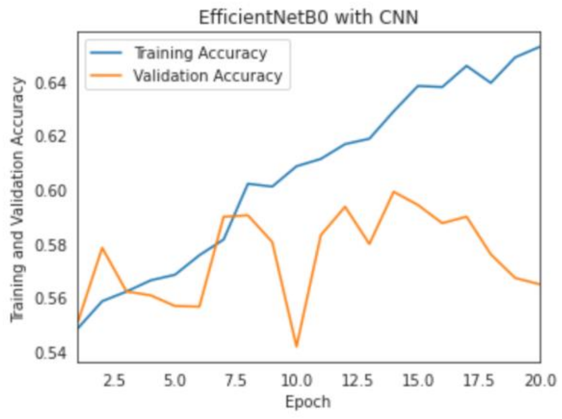

Fig 4.1: Training & validation accuracy vs Epoch- EfficientNetB0

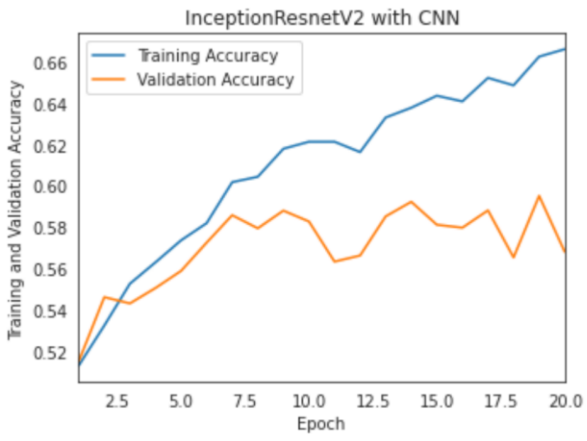

Fig 4.2: Training & validation accuracy vs Epoch-InceptionResnetV2

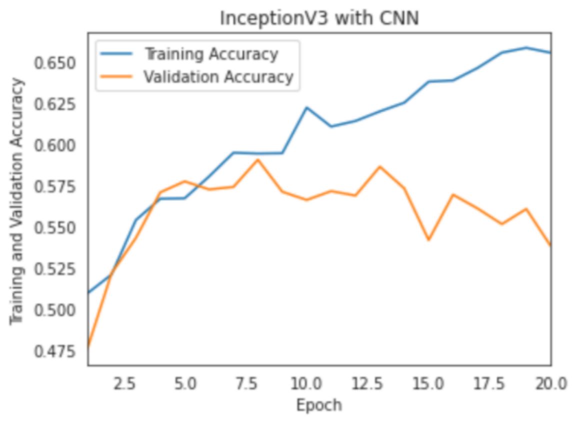

Fig 4.3: Training & validation accuracy vs Epoch- InceptionV3

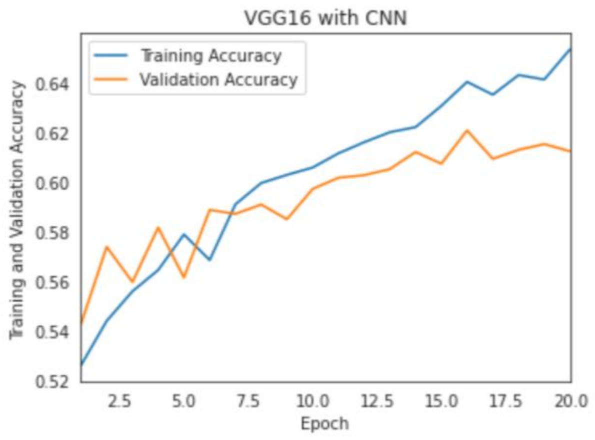

Fig 4.4: Training & validation accuracy vs Epoch-VGG16

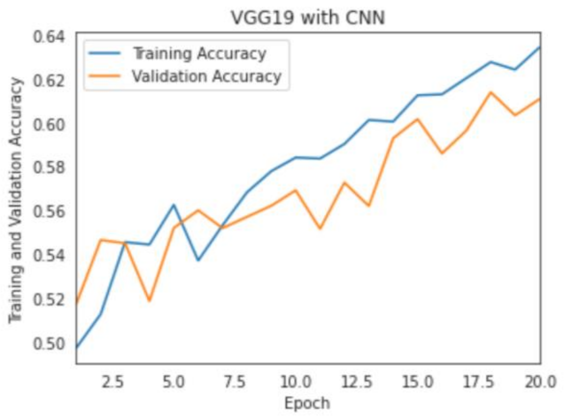

Fig 4.5: Training & validation accuracy vs Epoch- VGG19

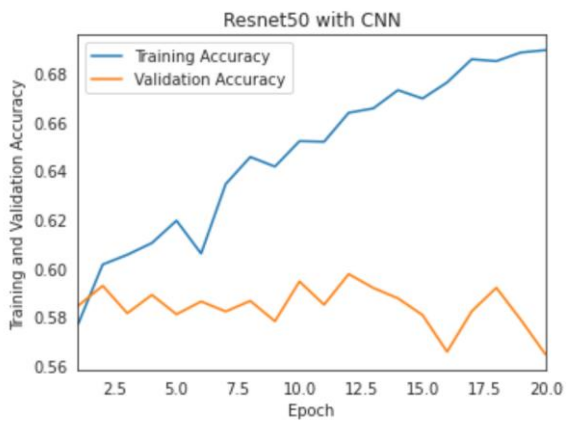

Fig 4.6: Training & validation accuracy vs Epoch- Resnet50

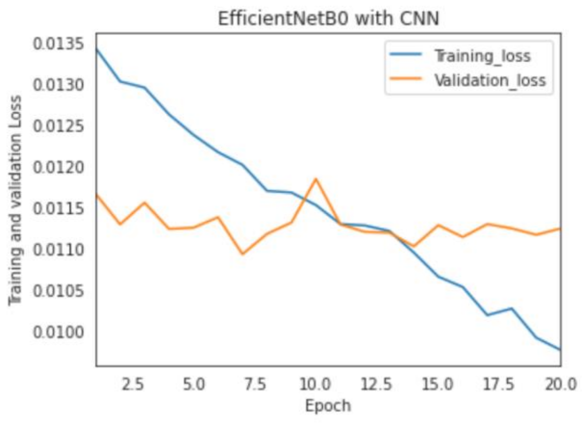

Fig 4.7: Training & validation loss vs Epoch- EfficientNetB0

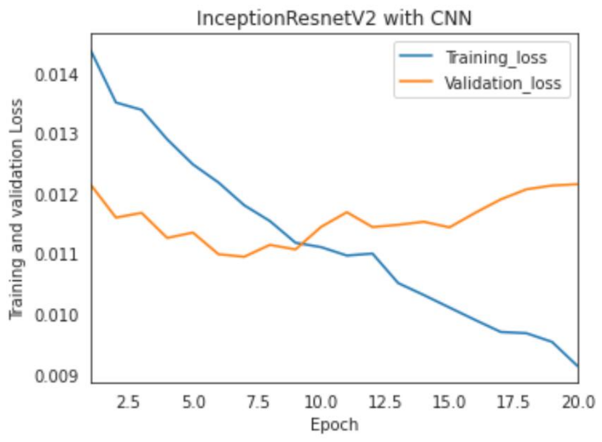

Fig 4.8: Training & validation loss vs Epoch- InceptionResnetV2

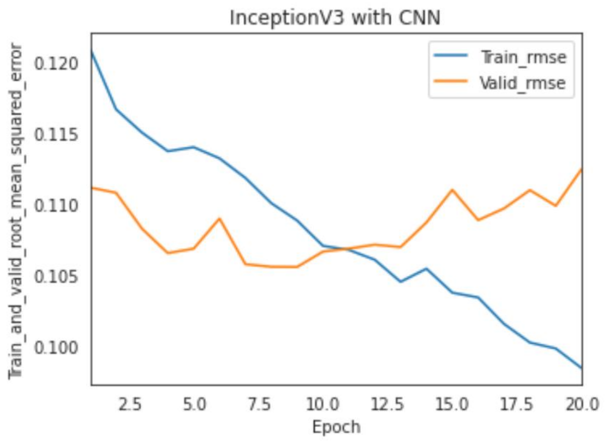

Fig 4.9: Training & validation loss vs Epoch- InceptionV3

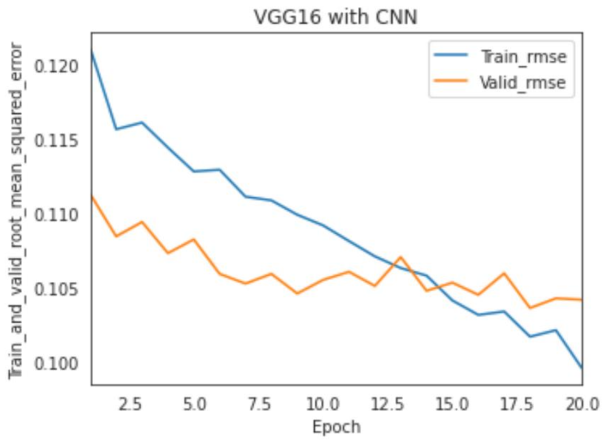

Fig 4.10: Training & validation loss vs Epoch- InceptionResnetV2

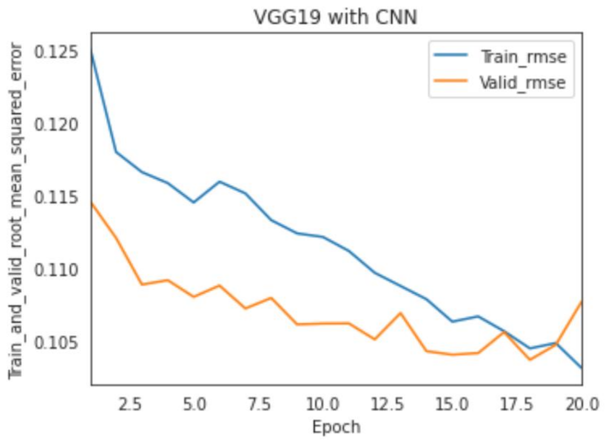

Fig 4.11: Training & validation accuracy vs Epoch- VGG19

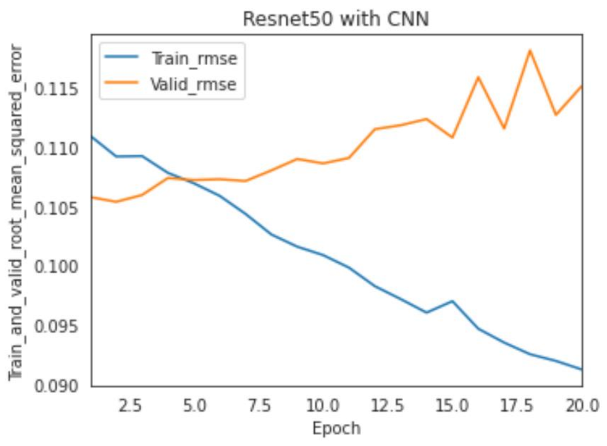

Fig 4.12: Training & validation loss vs Epoch- Resnet50

4.3.2 Training Accuracy and Validation Accuracy vs Epoch

The training accuracy and Validation accuracy are individually plotted for all 6 different models as follows:

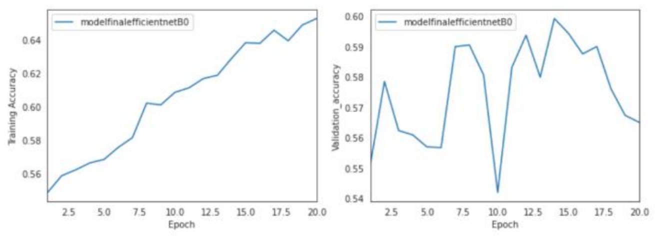

Fig 4.13: Training accuracy vs Epoch - EfficientNetB0 Validation accuracy vs Epoch - EfficientNetB0

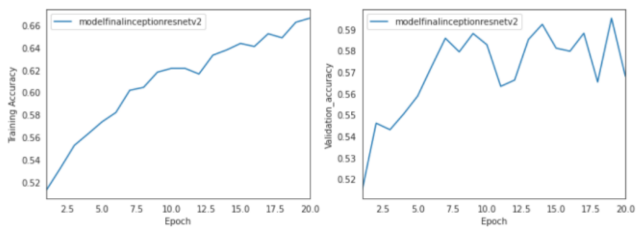

Fig 4.14: Training accuracy vs Epoch - ResNetV2 Validation accuracy vs Epoch - ResNetV2

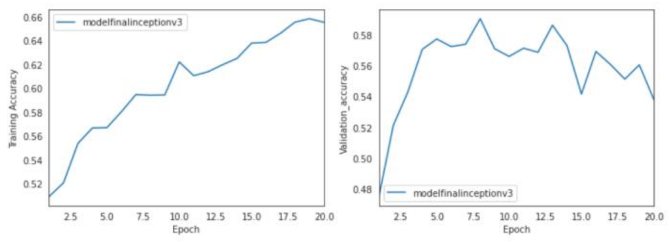

Fig 4.15: Training accuracy vs Epoch - InceptionV3 Validation accuracy vs Epoch - InceptionV3

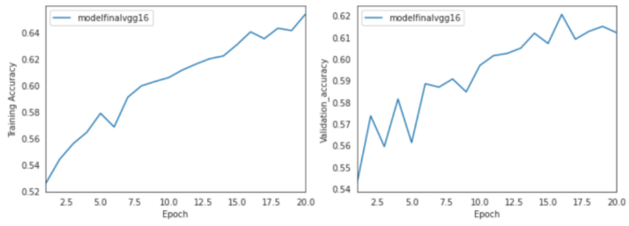

Fig 4.16: Training accuracy vs Epoch - VGG16 Validation accuracy vs Epoch - VGG16

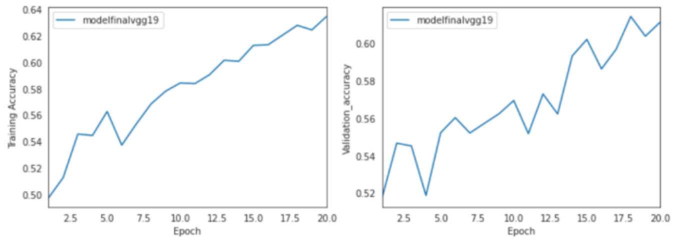

Fig 4.17: Training accuracy vs Epoch - VGG19 Validation accuracy vs Epoch - VGG19

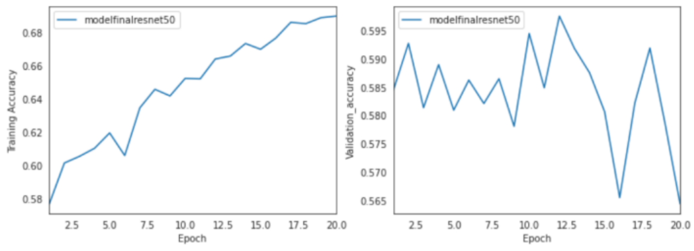

Fig 4.18: Training accuracy vs Epoch - Resnet50 Validation accuracy vs Epoch - Resnet50

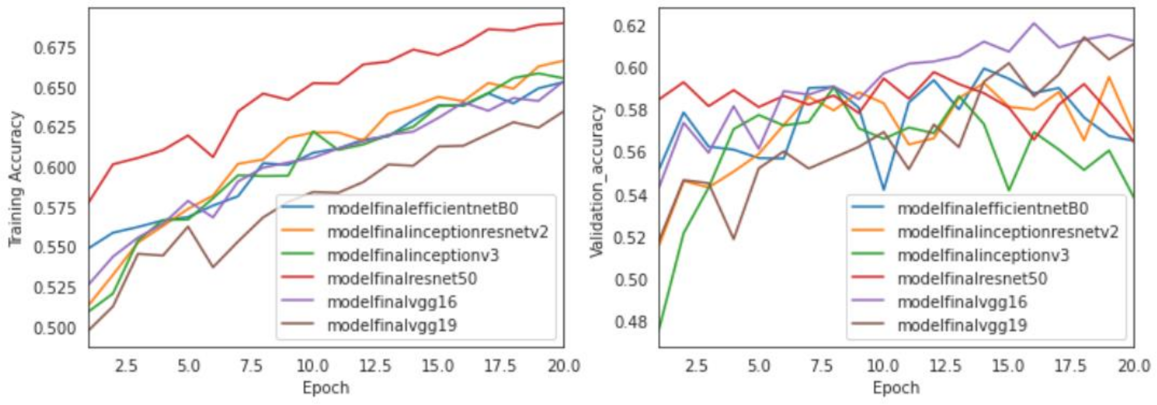

Fig 4.19: Comparative training accuracy and validation accuracy of all models

Based on the provided graph, it is evident that the graph displays the individual training and validation accuracy of each model.
However, the last two graphs specifically focus on only one model, the EfficientNetB0 with CNN. These two graphs highlight
that the EfficientNetB0 with CNN model exhibits less fluctuation in comparison to the other models depicted in the individual

graphs.

Throughout the graph, various models are evaluated based on their training and validation accuracy as the number of epochs progresses. Among all the models presented, the EfficientNetB0 with CNN model stands out in terms of its stability and consistency. The last two graphs, dedicated solely to this model, provide a clear visual representation of its performance.
In these last two graphs, the training and validation accuracy of the EfficientNetB0 with CNN model remain relatively steady as the epochs increase. This consistency suggests that the model is less susceptible to overfitting and maintains its ability to generalize well to unseen data.

In contrast, the other models depicted in the individual graphs exhibit more fluctuation in their training and validation accuracy curves. This fluctuation implies potential issues with overfitting or difficulties in generalizing to new, unseen data.

In summary, the graph effectively illustrates the performance of different models concerning their training and validation accuracy. The specific attention given to the EfficientNetB0 with CNN model in the last two graphs underscores its superior performance in terms of stability and lower fluctuation, making it a promising candidate for practical applications compared to
the other models shown.

4.3.3 Training Loss and Validation Loss vs Epoch

The training loss and Validation loss are individually plotted for all 6 different models as follows:

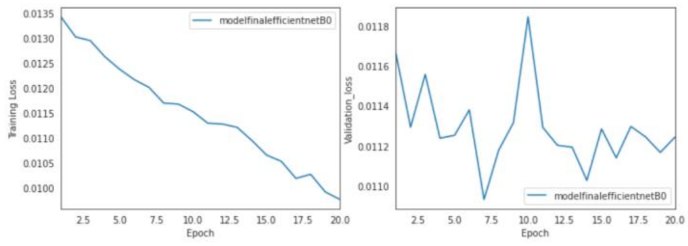

Fig 4.20: Training loss vs Epoch - VGG19 Validation loss vs Epoch - VGG19

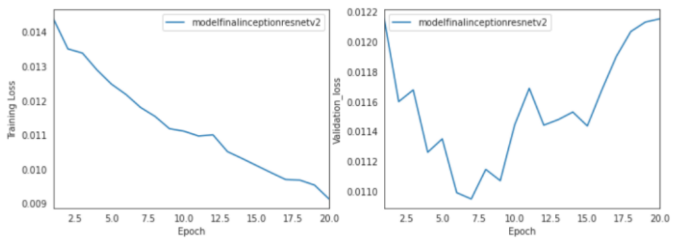

Fig 4.21: Training loss vs Epoch - InceptionResnetV2 Validation loss vs Epoch - InceptionResnetV2

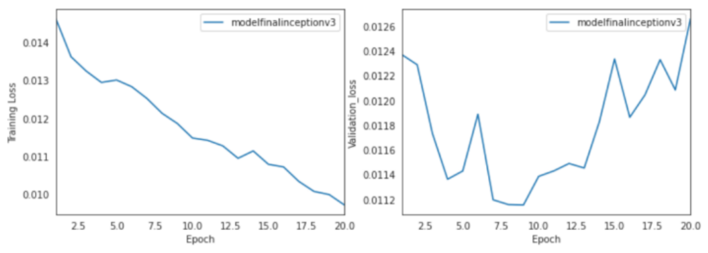

Fig 4.22: Training loss vs Epoch - InceptionV3 Validation loss vs Epoch - InceptionV3

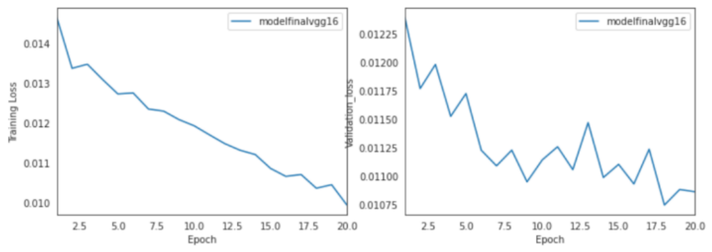

Fig 4.23: Training loss vs Epoch - VGG16 Validation loss vs Epoch - VGG16

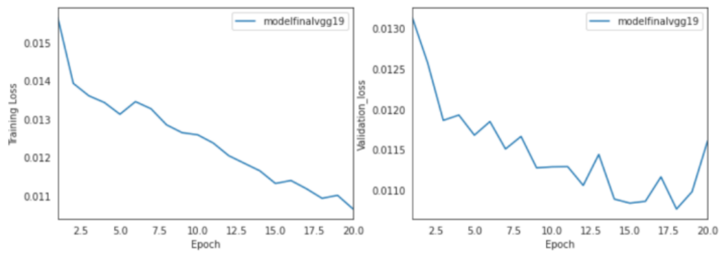

Fig 4.24: Training loss vs Epoch - VGG19 Validation loss vs Epoch - VGG19

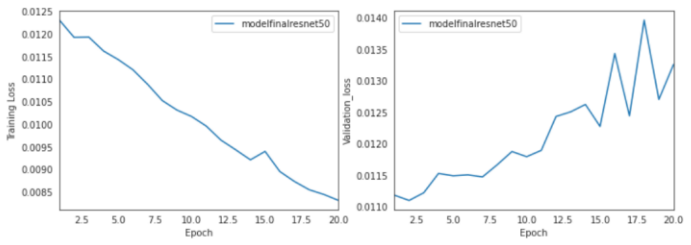

Fig 4.25: Training loss vs Epoch - Resnet50 Validation loss vs Epoch - Resnet50

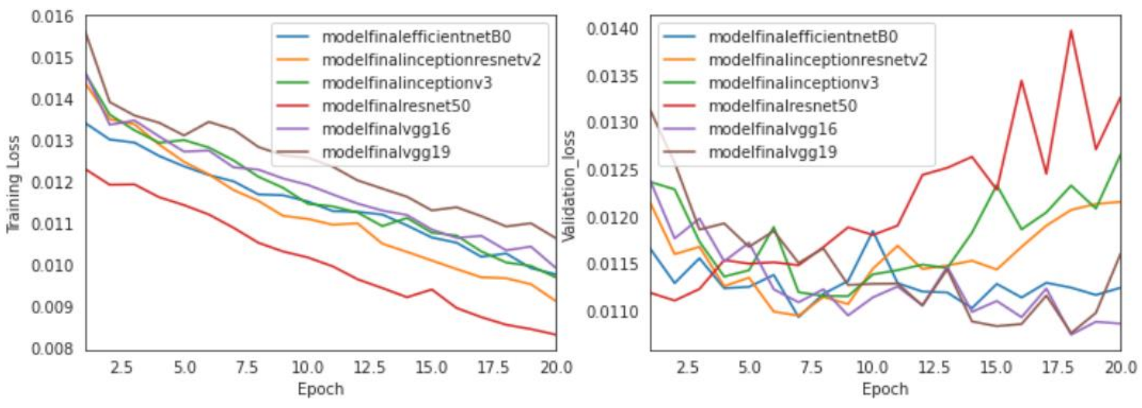

Fig 4.26: Comparative training loss and validation loss of all models

The provided graph clearly presents the individual training and validation loss for each model. However, the focu of the last two graphs is solely on the EfficientNetB0 with CNN model. These two specific graphs reveal a significan advantage of the EfficientNetB0 with CNN model over the other models depicted in the individual graphs specifically in terms of its lower loss and better performance regarding training and validation loss as the epoch progress.

As we analyze the graph data, it becomes evident that various models have been evaluated based on their trainin and validation loss throughout whole epochs. Among all the models displayed, the EfficientNetB0 with CNN mode emerges as a standout performer due to its consistent and remarkable reduction in loss as the epochs increase. Thi model demonstrates excellent convergence and efficiency in minimizing the training loss while maintaining lov validation loss, indicating its ability to generalize effectively to new data.

On the contrary, the other models showcased in the individual graphs exhibit more erratic behavior in their trainin and validation loss curves. The fluctuations in their loss values imply potential challenges with convergence o overfitting, and they may struggle to maintain low validation loss as the training progresses.

In conclusion, the graph effectively visualizes the performance of various models concerning their training an validation loss. The last two graphs focusing on the EfficientNetB0 with CNN model highlight its exceptiona capability to achieve lower loss and superior performance compared to the other models presented. Thi characteristic makes the EfficientNetB0 with CNN model a compelling and reliable choice for tasks wher minimizing loss and ensuring robust generalization are crucial factors.

4.3.4 Training RMSE and Validation RMSE vs Epoch

The training RMSE and Validation RMSE are individually plotted for all 6 different models as follows:

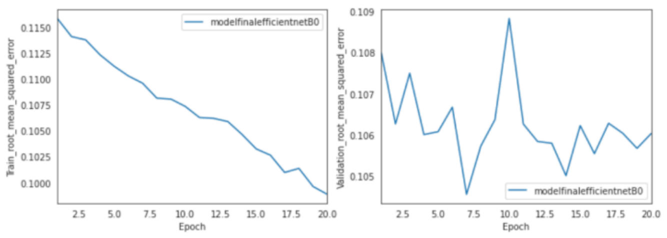

Fig 4.27: Training RMSE vs Epoch - VGG19 Validation RMSE vs Epoch - VGG19

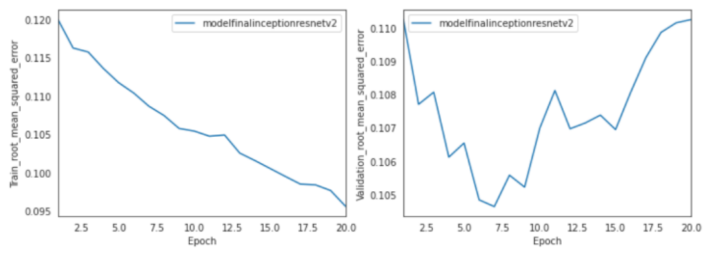

Fig 4.28: Training RMSE vs Epoch - Resnet50 Validation RMSE vs Epoch - Resnet50

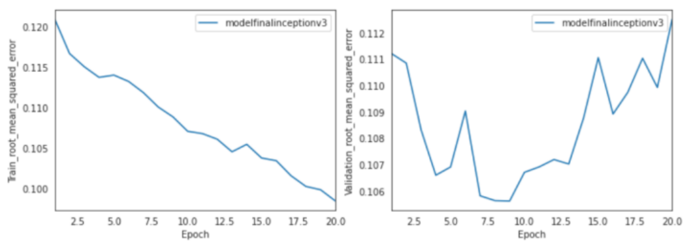

Fig 4.29: Training RMSE vs Epoch - InceptionV3 Validation RMSE vs Epoch - InceptionV3

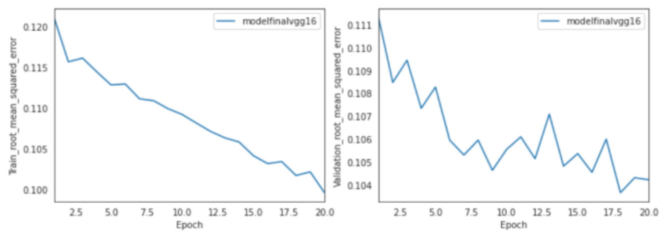

Fig 4.30: Training RMSE vs Epoch - VGG16 Validation RMSE vs Epoch - VGG16

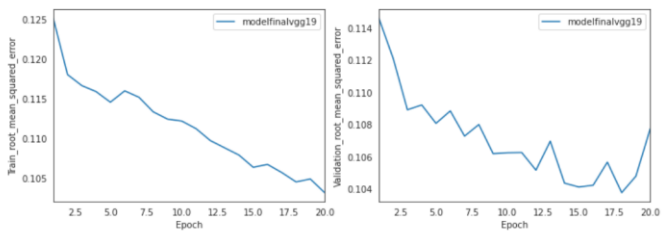

Fig 4.31: Training RMSE vs Epoch - VGG19 Validation RMSE vs Epoch - VGG19

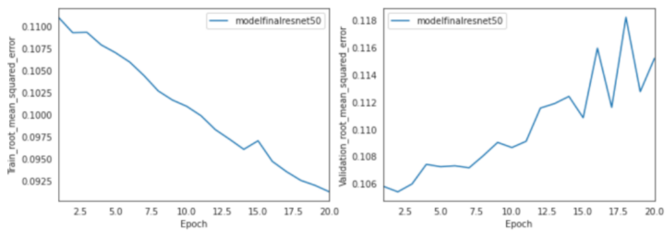

Fig 4.32: Training RMSE vs Epoch - Resnet50 Validation RMSE vs Epoch - Resnet50

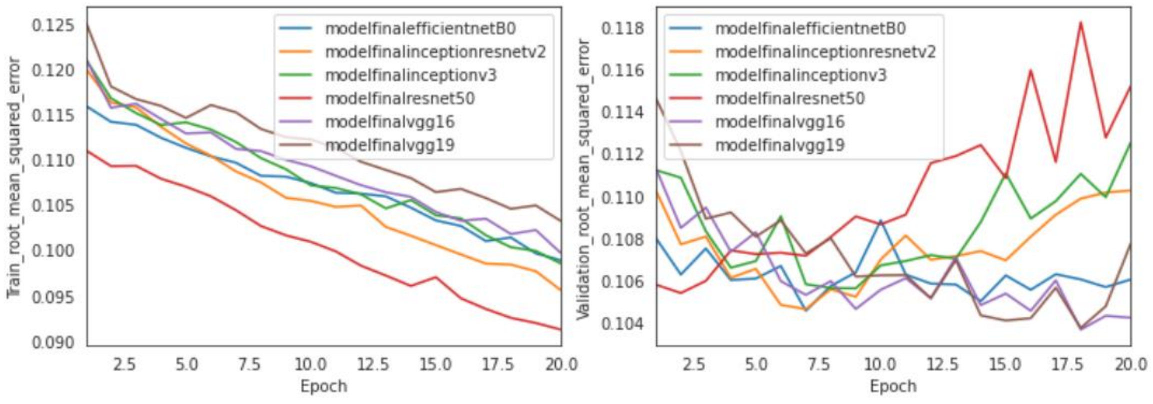

Fig 4.33: Comparative training RMSE and validation RMSE of all models

Based on the given graph, it provides a clear representation of the individual training and validation root mean square error (RMSE) for each model. However, the focus of the last two graphs is exclusively on the EfficientNetB0 with CNN model. These two specific graphs highlight a prominent advantage of the EfficientNetB0 with CNN model over the other models shown in the individual graphs, specifically in terms of its lower RMSE and superior performance concerning training and validation RMSE as the epochs progress.

Upon analyzing the graph data, it becomes apparent that various models have been evaluated based on their training and validation RMSE values throughout the training process. Among all the models depicted, the EfficientNetB0 with CNN model stands out as a top performer due to its consistent and remarkable reduction in RMSE as the epochs advance. This model demonstrates exceptional convergence and efficiency in minimizing the training RMSE while maintaining a low validation RMSE, which signifies its capacity to generalize effectively to new and unseen data.

Conversely, the other models displayed in the individual graphs exhibit more erratic behavior in their training and validation RMSE curves. The fluctuations in their RMSE values suggest potential challenges with convergence or overfitting, and they might struggle to achieve and sustain low validation RMSE as the training proceeds.

In conclusion, the graph effectively illustrates the performance of various models concerning their training and validation RMSE. The last two graphs, which focus solely on the EfficientNetB0 with CNN model, emphasize its exceptional ability to attain lower RMSE and outperform the other models presented. This characteristic positions the EfficientNetB0 with CNN model as a compelling and reliable choice for tasks where minimizing RMSE and ensuring robust generalization are pivotal factors.

4.3.5 Bar Graph for Comparative Analysis

Training Accuracy: The ResNet50 has the highest training accuracy whereas VGG19 has the least training accuracy.
Training RMSE: The VGG19 has the highest training RMSE whereas ResNet50 has the least training RMSE.

Validation Accuracy: The VGG16 has the highest validation accuracy whereas InceptionV3 has the least validation accuracy.
Validation RMSE: The ResNet50 has the highest validation RMSE whereas VGG16 has the least validation RMSE.
Test Accuracy: The EfficientNetB0 has the highest test accuracy whereas InceptionResNetV2 has the least test accuracy.
Test Loss: The EfficientNetB0 has the least test loss whereas InceptionRes-NetV2 has the highest test loss.
Test RMSE: The EfficientNetB0 has the least test RMSE whereas Inception-ResNetV2 has the highest test RMSE.

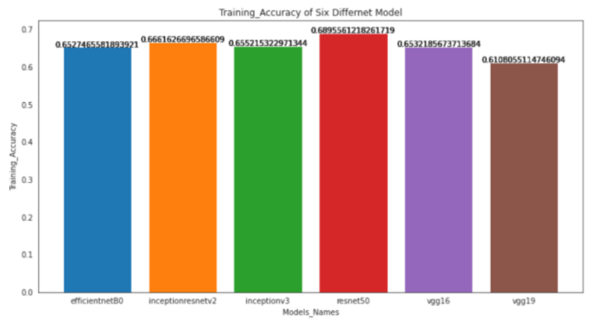

Fig 4.34: Bar graph comparing the training accuracy of all models

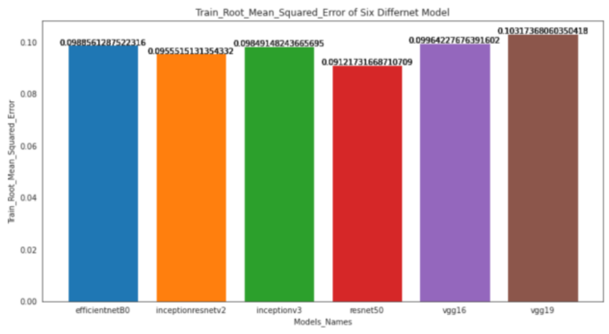

Fig 4.35: Bar graph comparing the training RMSE of all models

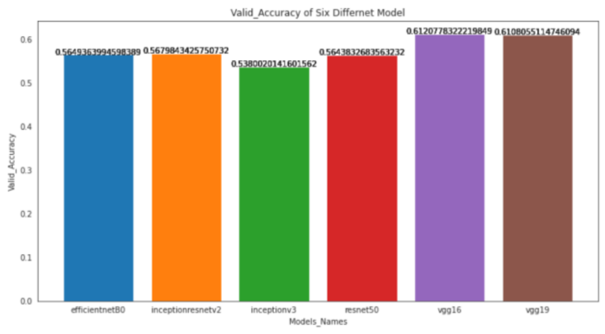

Fig 4.36: Bar graph comparing the validation accuracy of all models

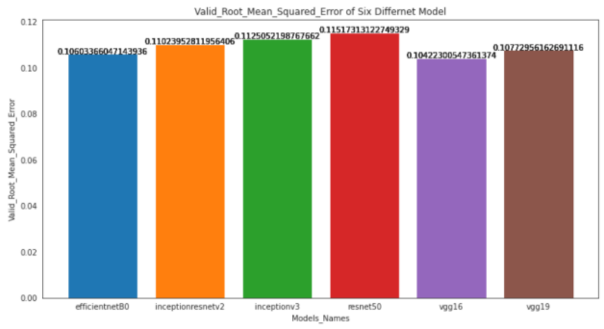

Fig 4.37: Bar graph comparing the validation RMSE of all models

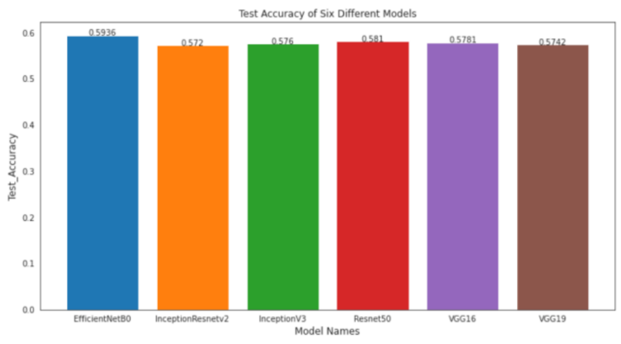

Fig 4.38: Bar graph comparing the test accuracy of all models

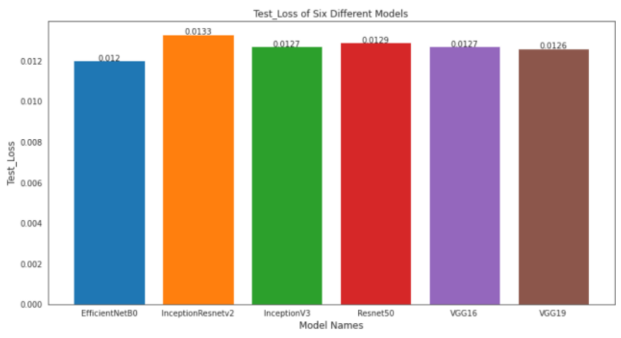

Fig 4.39: Bar graph comparing the test loss of all models

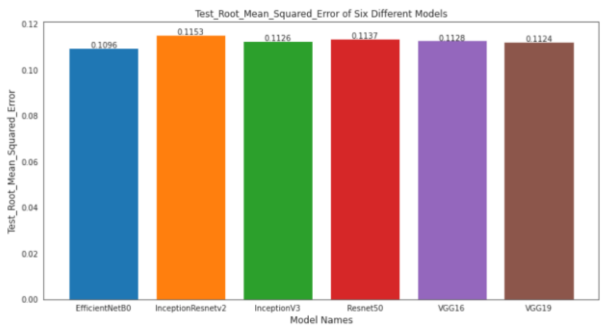

Fig 4.40: Bar graph comparing the test RMSE of all models

4.3.6 OUTPUT

Analysing accuracy and RMSE as the evaluation matrices of all the models, it could be concluded that the best model for our project was EfficicentNetB0. Hence, we have used EfficientNetB0 with our baseline CNN model as the final model to colorize the input gray-scale images.

The training and validation loss, accuracy and RMSE of the final model i.e. CNN with EfficientNetB0 is as follows:

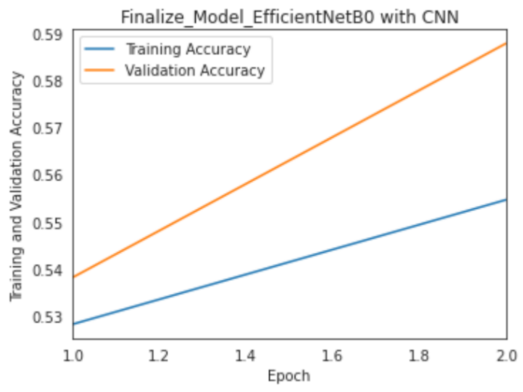

Fig 4.41: Training and Validation Accuracy of final model

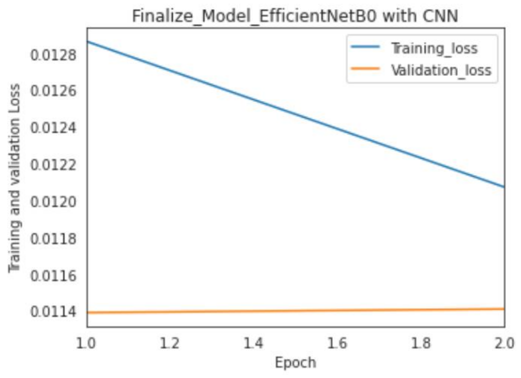

Fig 4.42: Training and Validation loss of final model

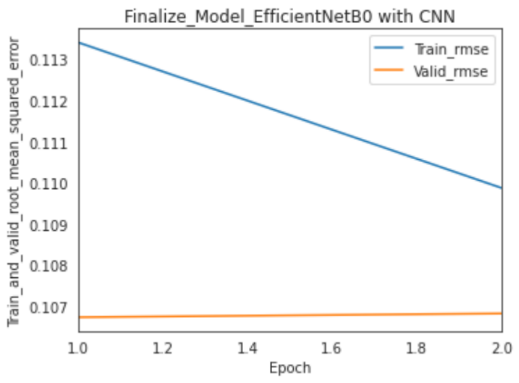

Fig 4.43: Training and Validation rmse of final model

4.3.7 Output from final model

The output obtained from our final model are as follows:

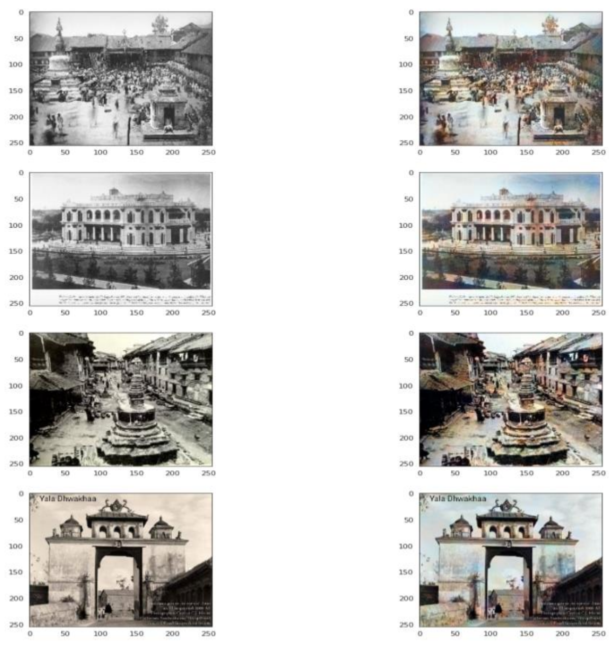

Fig 4.44: Output from final model

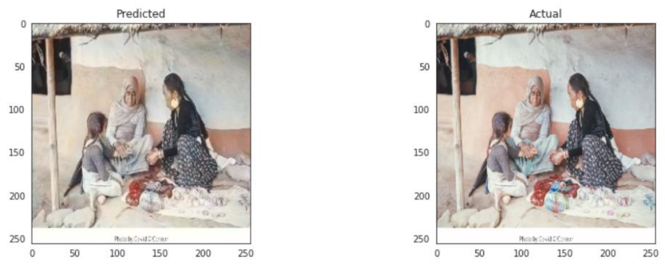

Fig 4.45: Actual vs predicted output

5. Conclusion

Hence, our model can be used to effectively colorize black and white images. The model is exclusively implemented using Nepalese dataset making the predictions authentic to Nepalese background. The incorporation of EfficientNetB0 as the pre-trained model has increased the accuracy and performance.

References

1. A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 2012.
2. M.R. Joshi, L. Nkenyereye, G.P. Joshi, S.M.R. Islam, M.A.A. Wadud, and S. Shrestha. Auto-colorization of historical images using deep convolutional neural networks, 2020.
3. R. Zhang, P. Isola, and A.A. Efros. Colorful image colorization, 2016.
4. Hiromu Yakura, Yoshihiro Oyama, and Jun Sakuma.Malware analysis of imaged binary samples by convolutional neural network with attention mechanism,2018.
5. 4 layers of convolutional neural network - data science novice. https://www.datasciencenovice.com/2020/09/4-layers-of-convoluitonal-neural-network.html.

6. François Petitjean, Geoffrey I Webb, and Charlotte Pelletier. Temporal convolutional neural network for the classification of satellite image time series,2019.
7. Lucas Rodés-Guirao. Deep koalarization: Image colorization using cnns and inception-resnet-v2, 2017.
8. What is resnet or residual network | How resnet helps? https://www.mygreatlearning.com/blog/resnet/.
9. Francis Jesmar Montalbo and Alvin Alon. Empirical analysis of a fine-tuned deep convolutional model in classifying and detecting malaria parasites from blood smears. KSII Transactions on Internet and Information Systems, 2021.