Butterfly and Moth Classification

Xinyi Li
Faculty of Science
Western University
London, Canada
xli3322@uwo.ca

Yue Wan
Faculty of Science
Western University
London, Canada
ywan5974@uwo.ca

Miles Xi
Faculty of Science
Western University
London, Canada
zxi27@uwo.ca

***Abstract*—Butterflies are important indicators of environmental health, with species identification offering valuable insights for biodiversity studies. However, manual butterfly classification is time-consuming and requires expert knowledge. Recent advancements in image classification provide an opportunity to automate this process, which could significantly enhance ecological research and conservation efforts. In this project, we aim to leverage convolutional neural network (CNN) models to classify butterfly species based on a labeled image dataset. We experimented with a simple CNN model and several pre-trained models. We assessed performance using accuracy and confusion matrix metrics, refining it through hyperparameter tuning. This approach aims to identify the best model for distinguishing visually similar butterfly species. The final results indicate over 95% test accuracy, suggesting that deep learning can support fine-grained species classification and large-scale ecological monitoring.**

***Index Terms*—Deep Learning, Convolutional Neural Network, ResNet, EfficientNet, Image Classification**

I. Introduction

Butterflies and moths are not only beautiful insects but also essential to the health of ecosystems. As important pollinators, they contribute to the reproduction of many plant species, helping maintain biodiversity. Their presence or absence can indicate the overall health of an ecosystem, making accurate species identification crucial for scientists and conservationists.

Automating butterfly species recognition could significantly improve the efficiency of ecosystem assessments, allowing for faster, more extensive monitoring of biodiversity. This could be particularly valuable in tracking environmental changes such as climate change and habitat loss. By improving identification methods, we could enhance both ecological research and conservation strategies, enabling quicker responses to shifts in species populations and more effective protection of endangered species. Ultimately, these advances would contribute to the preservation of biodiversity and the long-term stability of ecosystems.

A. Importance of image classification

Image classification plays a critical role in identifying butterfly species, especially when faced with challenges such as intricate patterns and varied backgrounds. Recent studies, such as [1], highlight the effectiveness of using pre-trained convolutional neural networks (CNNs), like EfficientNet, for butterfly classification, achieving high accuracy through optimizations in depth, width, and resolution. Additionally, models like ResNet and DenseNet have shown significant success in handling complex datasets [2]. EfficientNetB3, combined with transfer learning, has also proven effective for insect species classification, further improving accuracy and taxonomy [3]. These advances are essential for speeding up species identification, making it more efficient and accessible for large-scale ecological monitoring.

Traditional methods of species identification rely on expert knowledge and are time-consuming and labor-intensive. In contrast, deep learning models can automate this process, enhancing efficiency and accessibility for non-experts while accelerating biodiversity research and conservation efforts. However, different CNN models exhibit varying preferences for datasets, and deeper, more complex models are not always the best choice. In this project, we compare several fine-tuned CNN models on a specific dataset, considering both classification accuracy and time efficiency to identify the most effective model for practical use.

B. Importance of comparing CNN Models

Comparing CNN models is crucial for identifying the best architecture for butterfly species classification. Research has shown the success of various models, such as EfficientNet [1], ResNet, and DenseNet, in accurately handling diverse datasets [2]. These models have demonstrated significant potential for improving classification performance, with EfficientNetB3 being particularly effective when combined with transfer learning [3]. The ongoing comparison of different architectures enables us to optimize the classification process, ultimately supporting more accurate and efficient species identification, which is essential for biodiversity conservation.

The remainder of the paper is organized as follows: Section II provides the background and related work in detail, offering an overview of the research context and previous studies that informed this project. Section III

describes the applied methodology, starting with the research objectives and then delving into the research methodology, outlining the core technical concepts and strategies used to address the research problem, explaining the selection of specific techniques or methods employed and the rationale behind these choices. In Section IV, we presented the empirical estimations, provided a detailed account of the key findings, supported by visual aids such as charts and tables, and analyzed the implications of the findings. Finally, Section V draws conclusions in relation to the research objectives and discusses potential directions for future work.

II. Background & Related Work

Classifying butterfly species using machine learning and deep learning has been a popular research topic, as it deals with challenges such as different backgrounds, complex wing patterns, and similarities between species. Different types of convolutional neural networks (CNNs) have been tested to handle these difficulties more effectively.

A study [1] used EfficientNet and pre-trained deep convolutional neural networks to classify butterfly species. They applied the YOLOv3 model for object detection on the Beautiful Butterflies dataset, achieving high accuracy thanks to EfficientNet's scalability and efficiency. Another study used ResNet50, emphasizing the importance of residual connections in overcoming vanishing gradient problems. This allowed for deeper feature extraction, making it especially useful for classifying butterflies and moths [2].

EfficientNet has been shown to outperform traditional CNN architectures by effectively balancing depth, width, and resolution. This compound scaling method improves resource usage while maintaining high classification performance [3]. Comparative studies have assessed DenseNet and ResNet architectures for butterfly classification. DenseNet's dense connections help with efficient gradient flow and better use of parameters, while ResNet is known for its robustness and ease of training, making both architectures competitive options [4].

Recent evaluations of various architectures, such as ResNet18, ResNet50, and EfficientNet, have shown the strengths and weaknesses of each model in butterfly classification tasks. ResNet50 and EfficientNet generally outperformed simpler models due to their advanced designs [5]. The reviewed studies emphasized the progress in using deep learning models for butterfly classification. EfficientNet, with its compound scaling, often delivers superior accuracy and efficiency. ResNet50 remains a strong choice, especially for datasets with high diversity or complexity, while DenseNet offers excellent parameter efficiency, making it ideal for resource-limited environments.

III. Methods

A. Research Objectives

The first goal of this project is to compare the performance of different image classification models that have different settings/hyperparameters during training and on the validation set. Given these different models, we would like to select the best one for our task: butterfly and moth image classification, and evaluate the final model on a test set.

B. Research Methodology

1) An overview of different model architectures

Simple CNN. The architecture of a convolutional neural network (CNN) consists of two specialized types of layers: convolutional layers and pooling layers [6]. Convolutional layers have filters which, using learned filter weights, perform convolution operations on pixel values of each channel of an image represented by a feature map, extracting local features. Pooling layers reduce the dimensions of a feature map by, for example, selecting a maximum value from each of the small regions in that feature map and thus downsampling it. The process of convolution and pooling is often repeated multiple times, and then the feature maps are flattened and passed through a fully connected layer, which produces the output. In our project, we implemented a simple CNN with three convolution and pooling steps followed by two fully connected layers. The number of channels and the size of the feature maps after each operation are shown in Figure 1 below. ReLu activation functions and max pooling were used. This model served as a baseline and could be compared with the pretrained models in the following text.

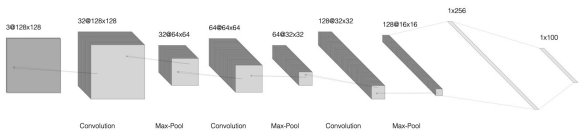

Fig. 1: Simple CNN Architecture

ResNet18. Our second model was ResNet18, which consists of 17 convolutional layers and a fully connected layer. Residual Networks allow the input to be directly added to the output of a later layer and to skip some convolutional layers in between, forming a residual block [7]. With the residual blocks, Residual Networks can train deeper models without suffering from the problem of vanishing/exploding gradients [8]. ResNet18 was used in our project for its balance between model performance and runtime. We froze all the layers except the fully connected one for training.

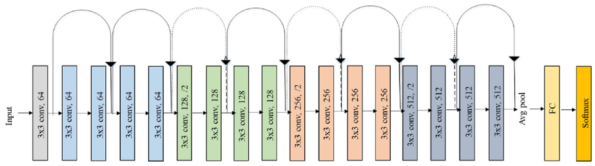

Fig. 2: Original ResNet18 Architecture [9]

EfficientNet-B0. The last model we used was EfficientNet-B0, the base model of the Efficient Nets family. Efficient nets use the compound scaling method to scale up the depth, width, and input image resolution of the network [10]. This highly effective model scaling method allows Efficient Nets to have fewer parameters, reduce computational cost, and improve efficiency while maintaining strong performance for image classification. In addition, the model incorporates inverted residual blocks and Squeeze-and-Excitation optimization into its architecture [11]. Similarly, only the parameters of the fully connected layer were trained in the pre-trained EfficientNet-B0 model.

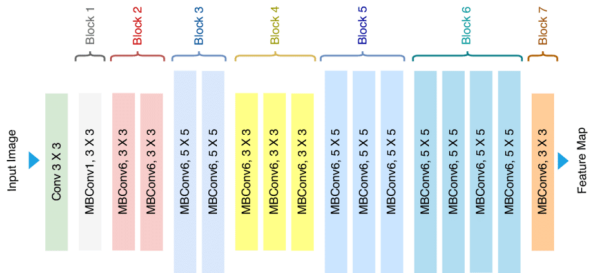

Fig. 3: EfficientNet-B0 Architecture [12]

2) Data and data preprocessing

The dataset, sourced from Kaggle, in total contains 13,594 images of butterflies and moths labeled by their species. It includes 12,594 images in the training set, 500 validation images, and 500 test images. These images are evenly distributed across 100 classes, making the class highly balanced. All images are colored and therefore have three channels, with a size of 224×224 pixels. Figure 4 shows a sample of images from the training set of Adonis blue species.

Fig. 4: Sample Butterfly Images

Before fitting the image classification models to our training data, images were preprocessed to meet the requirements of PyTorch pretrained models for training. Typical steps include: 1) resize the image such that the shortest side of it was 256 pixels, 2) crop a centered square region of the image, with dimensions 224×224 pixels, 3) convert images to tensor objects, and 4) normalize the pixels values in respective channels to a mean of [0.485, 0.456, 0.406] and standard deviation of [0.229, 0.224, 0.225]. This project followed this procedure except for the second step: we did not use the aforementioned input size for the ease of training. Additionally, to alleviate the problem of a relatively small training set, we augmented the training data through random horitonal flip and rotation, creating a transformed version of training data.

IV. Results

A. Training and validation performance

1) Hyperparameter Settings

The performance of each model is influenced by a variety of factors, including the training dataset quality, and hyperparameters such as learning rate, batch size and input size. In the first step, we conducted controlled experiments within each model on these hyperparameters to find the best fine-tuned model. We also recorded the execution time as a measure of model efficiency. We set up groups of parameters of input_size, batch_size, learning_rate and training set. The compared training set is a combination of untransformed training data and the transformed training data, which was double sized. The baseline group is input_size = (64×64), batch_size = 64, learning_rate = 0.001, original training set. The number of epochs in this step is 10.

2) CNN

TABLE I: Simple CNN Parameters Evaluation

Hyperparameters	Validation Accuracy	Execution Time
input_size = (64x64), batch_size = 64, learning_rate = 0.001, original training set	0.7220	3m 10.1s
input_size = (64x64), batch_size = 64, learning_rate = 0.0001, original training set	0.5260	3m 10.6s
input_size = (64x64), batch_size = 128, learning_rate = 0.001, original training set	0.6920	2m 52.2s
input_size = (64x64), batch_size = 64, learning_rate = 0.001, expanded training set	0.7860	5m 40.0s
input_size = (128x128), batch_size = 64, learning_rate = 0.001, original training set	0.7260	7m 11.5s

For the simple CNN Model, compared to the baseline parameters set, using a larger batch size didn't make a big difference. Using a larger input size also showed similar performance but was much slower. A smaller learning rate was not considered since the validation results were much worse and we assumed

it might need many more epochs to achieve a similar performance. Using an expanded training set has the best result. However, considering some other models did not have the best performance on an expanded training set, we decided not to choose it as well in simple CNN. Therefore, we stayed on the baseline group of parameters.

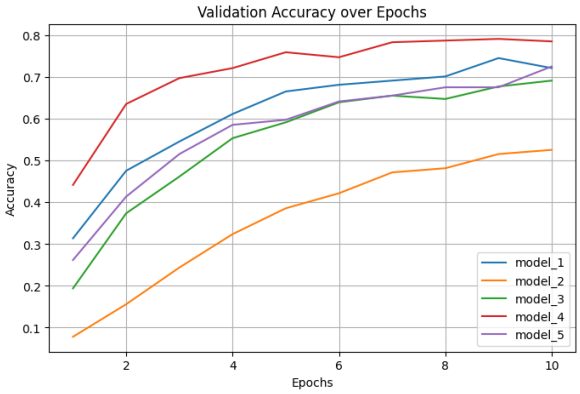

Fig. 5: Validation Accuracy of CNN Models

3) ResNet18

TABLE II: ResNet18 Parameters Evaluation

Hyperparameters	Validation Accuracy	Execution Time
input_size = (64x64), batch_size = 64, learning_rate = 0.001, original training set	0.8580	5m 12.9s
input_size = (64x64), batch_size = 64, learning_rate = 0.0001, original training set	0.9180	5m 6.5s
input_size = (64x64), batch_size = 128, learning_rate = 0.001, original training set	0.8980	4m 47.3s
input_size = (64x64), batch_size = 64, learning_rate = 0.001, expanded training set	0.9040	10m 6.6s
input_size = (128x128), batch_size = 64, learning_rate = 0.001, original training set	0.9180	13m 5.1s

For the ResNet18 Model, compared to the baseline parameters set, all other sets showed similar performances. Remarkably, the performance of using a smaller learning rate achieved the best at a small running time, despite using the same number of epochs. We believe this set of parameters can give the best performance for ResNet18 on more epochs efficiently.

4) EfficientNet

For the EfficientNet-B0 Model, compared to the baseline parameters set, all the other groups gave similar and good performance. With a smaller learning rate, the accuracy was slightly lower, but remained above 0.90. Similar to ResNet18, we believe this group with a smaller learning rate has the potential to achieve better results efficiently.

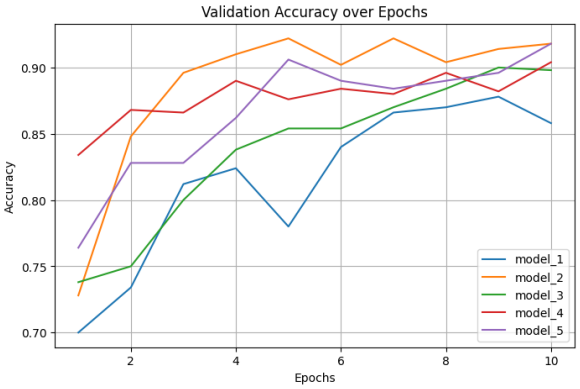

Fig. 6: Validation Accuracy of ResNet18 Models

TABLE III: EfficientNet Parameters Evaluation

Hyperparameters	Validation Accuracy	Execution Time
input_size = (64x64), batch_size = 64, learning_rate = 0.001, original training set	0.9080	7m 47.9s
input_size = (64x64), batch_size = 64, learning_rate = 0.0001, original training set	0.9060	7m 30.2s
input_size = (64x64), batch_size = 128, learning_rate = 0.001, original training set	0.9260	4m 7m 5.8s
input_size = (64x64), batch_size = 64, learning_rate = 0.001, expanded training set	0.9280	14m 45.9s
input_size = (128x128), batch_size = 64, learning_rate = 0.001, original training set	0.9340	21m 17.4s

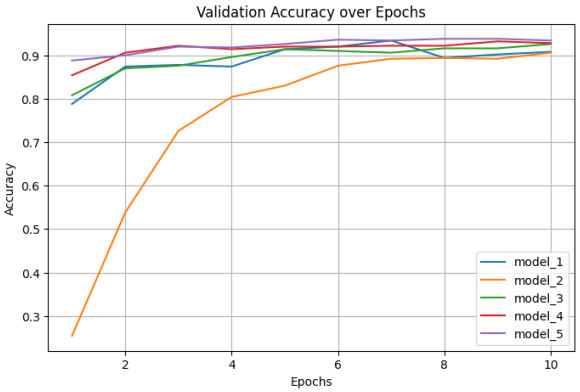

Fig. 7: Validation Accuracy of EfficientNet Models

5) Model Selection

For the final model selection, we evaluated three candidate models on the training and validation sets. Considering that the accuracy of a classification tool is still the most important factor in practical usage, we chose the final model based on validation accuracy. In this step, we trained all models on 20 epochs. To get a fair comparison and visualization over all 20 epochs, we didn't put any early stopping triggers. Fortunately, the plots didn't show a lot of fluctuation near the

end of each execution, thus we assumed there was no influence of not having the early stopping.

TABLE IV: Model Selection 1

Model	Validation Accuracy	Execution Time
CNN	0.7920	6m 21.1s
ResNet18	0.9220	10m 37.2s
EfficientNet	0.9320	15m 39.2s

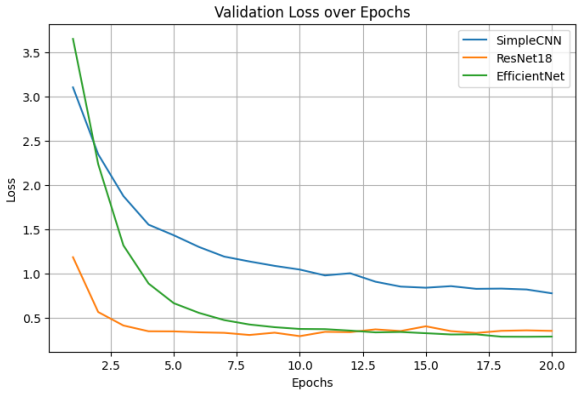

Fig. 8: Validation Loss of All Models

Since our dataset is not that large, it was unnecessary to run a large number of epochs. Another piece of supporting evidence was that the validation loss curves (Fig.8) appeared to stabilize and level off near the end of all 20 epochs, indicating that there was no need to continue training. See other related plots in Appendix Fig.10, Fig.11.

From the table (Table V), EfficientNet shows the highest values in the evaluation metrics on the validation set. TPrecision and Recall values were calculated by averaging across all 100 classes. The EfficientNet model has the highest precision (0.9410), recall (0.9320), and F1 score (0.9307). It also achieved the highest accuracy at 0.9320. We concluded that EfficientNet was the best model.

TABLE V: Model Selection 2

Model	Precision	Recall	F1_Score	Accuracy
CNN	0.8350	0.7920	0.7785	0.7920
ResNet18	0.9340	0.9220	0.9190	0.9220
EfficientNet	0.9410	0.9320	0.9307	0.9320

B. Model evaluation on the test set

The final model performance on the test set shows an accuracy at 0.9520. The model also has an average AUC score at 0.9983 (Fig.12).

TABLE VI: Final Model Performance

Model	Precision	Recall	F1_Score	Accuracy
EfficientNet	0.9587	0.9520	0.9508	0.9520

At the scope of each class, we conducted a confusion matrix of the test result. Taking the first class as an example, we can see all 5 test images were correctly predicted (Fig.9).

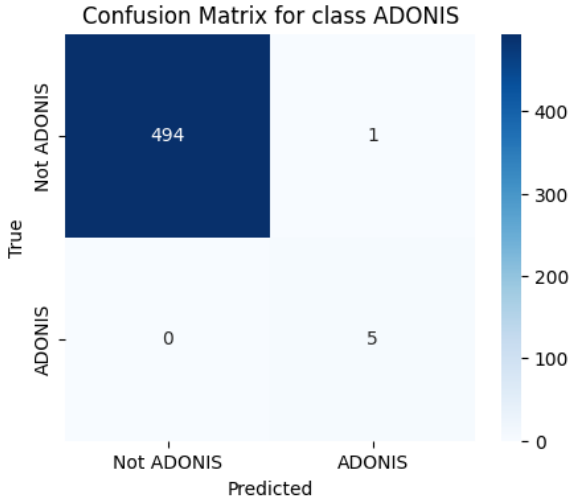

Fig. 9: Sample Confusion Matrix

V. Conclusion & Future Work

A. Conclusion

In this project, we successfully implemented and evaluated various models, ultimately identifying an optimal model to address the problem of butterfly and moth classification.

During the first step of our experiment, we observed that the small size of the training data could be a limitation, especially on our simple CNN models. While simple models struggled to capture complex patterns in small datasets, large models, particularly pre-trained ones, demonstrated improved performance by leveraging their ability to extract more detailed features and generalize better, even with limited data. A smaller learning rate benefited larger models more than simple ones which allows finer adjustment on complex parameters. Modification in this parameter effectively improved the performance without sacrificing efficiency. Theoretically, passing larger input images sizes is essential, since they would provide more features and details. However, for our small and simple dataset, this parameter did not make a significant difference. As a result, we selected three candidate models.

To have a better evaluation of these models, we performed training on a larger number of epochs in

this step. Considering that our data set is balanced, we simply use accuracy as our evaluation criterion. The performance of simple CNN was improved though more training cycles, while the pre-trained ones did not show notable improvement. From the results, ResNet18 and EfficientNet-B0 had similar validation performance, while EfficientNet outperformed at an accuracy of over 0.93. This model then showed even better test performance at an accuracy of over 0.95. This result indicated that the final model successfully dealt with new images of butterflies and moths.

B. Future Work

Since a larger input image size provides more features and details for the model to learn from, we experimented with an additional EfficientNet model using a 128×128 input size. This allowed the model to capture more intricate patterns and finer details in the images, potentially improving its performance. After training this modified model, its accuracy on the test set reached 0.9680, showing a noticeable improvement compared to the previous models. However, this model showed some signs of overfitting, as indicated by a slightly increasing validation loss. Future work may involve adjustments such as regularization techniques or further hyperparameter tuning to improve its robustness and prevent overfitting. We had also tried the ResNet50 model, thinking that its deeper architecture could have better performance than ResNet18, which it did. The tradeoff, however, was that it was very difficult to train. The moral of this was that in practice, time and resource constraints should be one of the top considerations in training deep networks.

Butterflies and moths outside these 100 classes are not expected to be classified correctly, as our model was trained exclusively on these classes. In future classification tasks, we recognize the importance of expanding the dataset by collecting more diverse and well-categorized source images.

Furthermore, this classification tool could be applied in practical settings, such as a mobile application for butterfly and moth identification. By integrating the model into an app, users would be able to capture images of butterflies and moths and receive immediate classification results. Additionally, expanding the dataset to include more species and fine-tuning the model for edge devices would further improve the tool's performance and usability in practical applications.

REFERENCES

[1] R. Faerie Mattins, M. Vergin, Azrina Abd Aziz, and S. Srivarshan, "Object detection and classification of butterflies using efficient CNN and pre-trained deep convolutional neural networks," Multimedia Tools and Applications, Nov. 2023, doi: https://doi.org/10.1007/s11042-023-17563-4.

[2] Scitepress.org, 2018. https://www.scitepress.org/publishedPapers/2023/127985/pdf/ index.html (accessed Dec. 20, 2024).

[3] R. Pillai, N. Sharma, D. Upadhyay, Sarishma Dangi, and G. Kaur, "From Pixels to Taxonomy: Advanced Insect Species Classification with EfficientNetB3 Transfer Learning," vol. 3, pp. 1623–1628, Jul. 2024, doi: https://doi.org/10.1109/icscss60660.2024.10625223.

[4] Y. Yang et al., "A comparative analysis of eleven neural networks architectures for small datasets of lung images of COVID-19 patients toward improved clinical decisions," Computers in Biology and Medicine, vol. 139, p. 104887, Dec. 2021, doi: https://doi.org/10.1016/j.compbiomed.2021.104887.

[5] J. Sharma, "EfficientNetB3 for High-Performance Insect Identification," pp. 955–959, Aug. 2024, doi: https://doi.org/10.1109/icoici62503.2024.10696281.

[6] G. James, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor, An Introduction to Statistical Learning. Springer Nature, 2023. Available: https://www.statlearning.com/.

[7] Wikipedia Contributors, "Residual neural network," Wikipedia, Aug. 14, 2019. https://en.wikipedia.org/wiki/Residual_neural_network.

[8] pawangfg, "Residual Networks (ResNet) - Deep Learning," GeeksforGeeks, Jun. 03, 2020. https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/.

[9] F. Ramzan et al., "A Deep Learning Approach for Automated Diagnosis and Multi-Class Classification of Alzheimer's Disease Stages Using Resting-State fMRI and Residual Neural Networks," Journal of Medical Systems, vol. 44, no. 2, Dec. 2019, doi: 10.1007/s10916-019-1475-2.

[10] M. Tan and Q. Le V., "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," arXiv.org, May 28, 2019. https://arxiv.org/abs/1905.11946

[11] N. Klingler, "EfficientNet: Pushing the Boundaries of Deep Learning Efficiency," viso.ai, Mar. 18, 2024. https://viso.ai/deep-learning/efficientnet/.

[12] T. Ahmed and N. H. N. Sabab, "Classification and Understanding of Cloud Structures via Satellite Images with EfficientUNet," SN Computer Science, vol. 3, no. 1, Dec. 2021, doi: 10.1007/s42979-021-00981-2.

APPENDIX

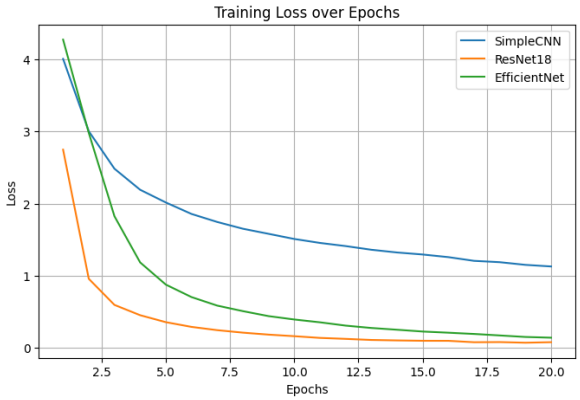

Fig. 10: Training Loss of All Models

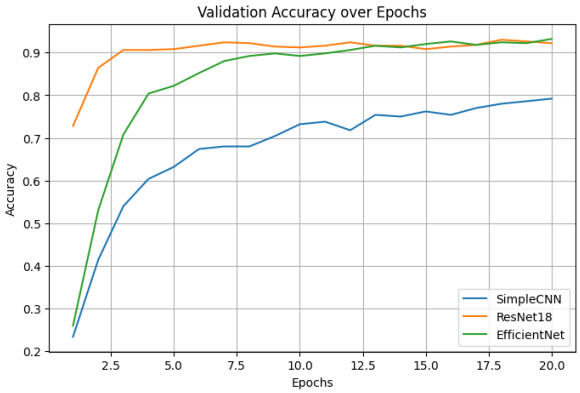

Fig. 11: Validation Accuracy of All Models

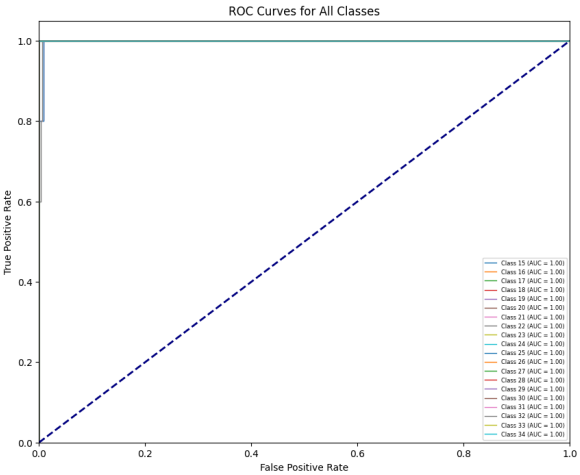

Fig. 12: AUC Plot on Selected Classes