

Illinois Institute of Technology
CS584: Machine Learning Final Project



Final PROJECT

Deep Neural Network with Residual Connections for Image Classification

Group 14

Group Members Names	CWID
Dhanvanth Voona	A20543395
Neeraj Vardhan Buneeti	A20545853
Sai Charan Gangili	A20543155

Table of Contents

S.No	Topic	Contribution
1	Background	Dhanvanth, Neeraj, Sai Charan
2	Problem Definition	Sai Charan
2.1	Vanishing Gradient	Dhanvanth
2.2	Long Computational Time	Neeraj
3	Proposed Algorithm	Dhanvanth
3.1 a)	Facial Expression Recognition Dataset	Neeraj
3.2 b)	Butterfly Dataset	Sai Charan
3.2	Custom Model	Dhanvanth
3.3	VGGNET	Sai Charan
3.4	RESNET	Neeraj
3.5	Pertained CNN Models and Transfer Learning	Sai Charan
3.6	Fine Tuning	Neeraj
4	Model Training and Experimentation	Dhanvanth
5	Results and Observation	Dhanvanth
5 a)	Confusion Matrix	Sai Charan
5 b)	Accuracy	Sai Charan
5 c)	Precision	Neeraj
5 d)	Recall	Neeraj
6	Conclusion and Future Enhancement	Dhanvanth

Model Training Contribution

<u>Dhanvanth</u>	<u>Neeraj</u>	<u>Sai Charan</u>
Custom Model [FER]	Custom Model [Butterfly]	ResNet-50 [Butterfly]
ResNet-50, ResNet-50 (Fine – tuned) [FER]	VGG19 [FER]	ResNet-50(Fine-tuned) [Butterfly]
VGG19 (Fine-tuned) [FER]	VGG19 (Fine-tuned) [Butterfly]	VGG19 [Butterfly]

Deep Neural network with residual connections for image classification

Abstract

In the realm of computer vision, the deployment of very deep neural networks, such as VGG19 and ResNet50, has significantly enhanced the capability to extract intricate features from images. However, these architectures are plagued by challenges such as gradient vanishing and protracted training times. To address these issues, this project introduces a custom deep neural network incorporating principles of depth scaling and skip connections, amalgamating the strengths of VGGNet [1] and ResNet [2]. Two distinct datasets are employed for evaluation purposes: butterfly types and facial emotion recognition (FER). The custom model, along with VGG19 and ResNet50, is trained and tested on both datasets to compare their performance. Remarkably, the custom model exhibits superior accuracy, achieving 88.82% on the butterfly and moth types of dataset [17]. VGG19 yields a comparatively higher accuracy of 67.34% while custom model provides accuracy of 61.24% on the FER dataset [16], emphasizing the custom model's versatility in handling diverse datasets.

Keywords: Computer vision, Deep Neural Networks, Vanishing gradient, skip connections, depth scaling, Performance.

1. Background (Literature survey)

Very deep convolution neural networks (CNNs) [3, 4] have significantly advanced image classification, yielding a sequence of notable achievements [4, 5, 6]. These deep networks seamlessly incorporate low, mid, and high-level features [5], along with classifiers, within an end-to-end multilayer structure. The augmentation of feature "levels" is achieved by the addition of stacked layers, commonly referred to as depth. Recent empirical findings [1, 7] underscore the critical role of network depth, with top-performing models [1, 7] on the challenging ImageNet dataset [8] consistently utilizing "very deep" architectures [1] ranging from sixteen [1] to thirty layers [9].

Motivated by the profound impact of depth in neural networks, a fundamental query emerges: Is the enhancement of learning in neural networks as straightforward as augmenting the number of layers? Answering this question historically encountered a formidable challenge known as the vanishing/exploding gradients problem [10, 11, 12]. This issue obstructs convergence right from the outset. However, substantial strides have been made in mitigating this problem, primarily through the adoption of normalized initialization techniques [13, 12, 14, 15] and the incorporation of intermediate normalization layers [9]. Still, in the case of very deep networks, the gradient vanishing has still obstructed the convergence leading to longer computational time for training.

2. Problem Definition

The realm of very deep neural networks is marked by its immense potential for intricate tasks in computer vision. However, two significant challenges cast shadows on their widespread adoption.

2.1. Vanishing Gradient:

As the depth of a neural network increases, the back propagation of errors during training encounters a critical hurdle known as the vanishing gradient problem [10, 11, 12]. This challenge arises due to the

diminishing magnitude of gradients as they traverse through numerous layers during back propagation. In essence, the gradients dwindle to near-zero values, rendering them ineffective in updating the weights of early layers. Consequently, these layers fail to learn meaningful representations, impeding the overall convergence of the network. The vanishing gradient problem becomes particularly pronounced in very deep architectures, where the prolonged path that gradients traverse exacerbates the attenuation effect.

2.2. Long Computational Time:

Simultaneously, the computational demands associated with training very deep neural networks escalate dramatically. The intricate interplay of an increasing number of layers and parameters necessitates extensive computational resources and time. The sheer complexity of the network structure contributes to prolonged training times, making it a formidable challenge to efficiently harness the potential of these deep architectures within reasonable timeframes. The extended training periods not only strain computational resources but also hinder the swift development and deployment of models for practical applications.

To address these challenges, we have introduced a custom model with fewer parameters and enhanced performance. The details of the proposed algorithm are discussed in Section 3. Subsequently, Model training and experimentation, Results and discussions are presented in Sections 4 and 5, respectively.

3. Proposed algorithm

The proposed methodology begins with selection of two distinct datasets, followed by data preprocessing and data splitting into train, test, and validation sets. In the subsequent stage the custom model and selected state-of-the-art (SOTA) models are trained and tested on the datasets. Furthermore, fine-tuning was carried out by unfreezing the significant portion of the pre-trained networks and the performance is compared based on the selected performance metrics. Figure 1. Provides a pictorial overview of the proposed system.

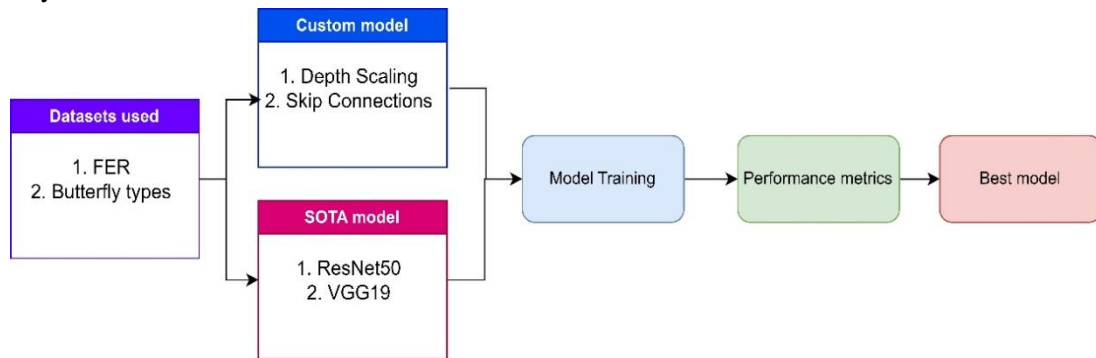


Figure 1. Proposed architecture.

3.1. Datasets Used








a. FER (Facial Expression Recognition)

The Facial Expression Recognition dataset comprises grayscale images of faces standardized to 48x48 pixel dimensions. Each image encapsulates distinct facial expressions conveying various emotions, categorized into seven primary classes: Angry (0), Disgust (1), Fear (2), Happy (3), Sad (4), Surprise (5), and Neutral (6). This dataset is designed for the explicit task of accurately classifying facial expressions into these specific emotion categories.

Encompassing a total of 28,709 examples, this segment serves as the primary resource for training deep-learning models to classify facial expressions into their respective emotion categories accurately. Each image in this set is labeled with the appropriate emotion category. This dataset is used for training purposes. The test set, comprising of 3,589 examples, acts as an independent benchmark for evaluating the trained models' performance and generalization capabilities. Similar to the training set, each image in this test set is associated with the corresponding emotion category for evaluation purposes. From Table 1, we can observe the unbalanced characteristic of FER dataset and visual overview of 7 basic emotions.

Facial Expression Recognition (FER) finds applications in various external domains, including Lie Detection, where it analyzes expressions and subtle facial cues crucial for forensic investigations. Additionally, in Security and Surveillance, FER plays a pivotal role in detecting suspicious or abnormal behavior based on facial expressions and bolstering security measures in public spaces, airports, and other critical locations.

Table 1. Visual representation of FER dataset.

Facial Expression Recognition Classes						
Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
						
No of Samples of Facial Expression Classes For Test Dataset						
958	111	1024	1774	1247	831	1233
No of Samples of Facial Expression Classes For Train Dataset						
3995	436	4097	7215	4965	4830	3171

b. Butterfly types

This dataset comprises distinct species of butterflies collected through real-time web capture. Each image representing a species is of size 224x224x3 pixels and is in JPG format. The entire collection of images in the dataset has been divided into three distinct subsets, serving the purposes of model development and evaluation.

This dataset is used for training the model, containing 12,594 photos in total, carefully organized into 100 subdirectories to provide an equitable representation of every species of butterfly. This section is the main source for deep neural network model training, with each species containing roughly 100-200 images for strong learning. The test and validation set consists of 500 images each distributed across 100 subdirectories, with each species represented by 5 images. The model's hyper parameters, such as learning rate and regularization, can be fine-tuned based on the validation set's feedback, enhancing the model's ability to generalize. The test set acts as an independent benchmark to thoroughly evaluate the model's performance on unseen data, thereby determining its degree of accuracy and generalizability.

The classification of distinct butterfly species holds paramount importance, notably in education and tourism. This classification not only contributes to ecological studies and scientific research, serving as indicators for environmental changes and climate impacts but also aids tourists in species identification, enriching nature-based tourism activities, and fostering awareness about biodiversity and conservation efforts.

The deliberate choice of these two distinct datasets is motivated by the aim to comprehensively assess the custom model's performance across a spectrum of real-life applications. By confronting the model with diverse challenges presented by human facial expressions and intricate butterfly and moth categorizations, we aim to validate its efficacy and generalizability, ensuring its robust performance across varied and complex tasks.

3.2. Custom model:

The custom CNN model is designed with a strategic combination of skip connections, depth scaling, and carefully selected convolutional layers to optimize both parameter efficiency and performance in computer vision tasks. We have tried different depths by extending the filterers from 64 to 512 and observed the best performance at 512 filters. We understood by going further extend could lead to increase in parameters, leading to increase in computation time. Hence we have proceeded with 512 filters at the final stage.

a. Skip connections

Skip connections are integrated strategically to facilitate the flow of gradients through the network, mitigating the vanishing gradient problem commonly encountered in very deep neural networks. Specifically, skip connections are established in each stage, allowing the network to learn both local and global features effectively. The Add () layer is used to employ the skip connections, which combines the output of a convolutional block with the output of a previous layer.

b. Depth Scaling

The model exhibits depth scaling principles by gradually increasing the number of filters in convolutional layers, providing a balance between model expressiveness and computational efficiency. The convolutional layers start with 32 filters and progressively scale up to 512, allowing the model to capture hierarchical features in a computationally manageable manner.

c. Layer Architecture

The model architecture comprises multiple stages, each containing a series of convolutional layers, batch normalization, and activation functions (ReLU). Short paths, implemented as additional convolutional layers, are incorporated to preserve and enhance feature information.

The inclusion of an Average Pooling layer further reduces spatial dimensions, aiding in abstraction.

d. Output Layer

The model culminates in a fully connected layer, followed by a SoftMax activation, tailored for multi-class classification tasks. The number of neurons in the output layer corresponds to the number of classes in the target dataset. The visual representation of the custom model can be observed in Figure 2.

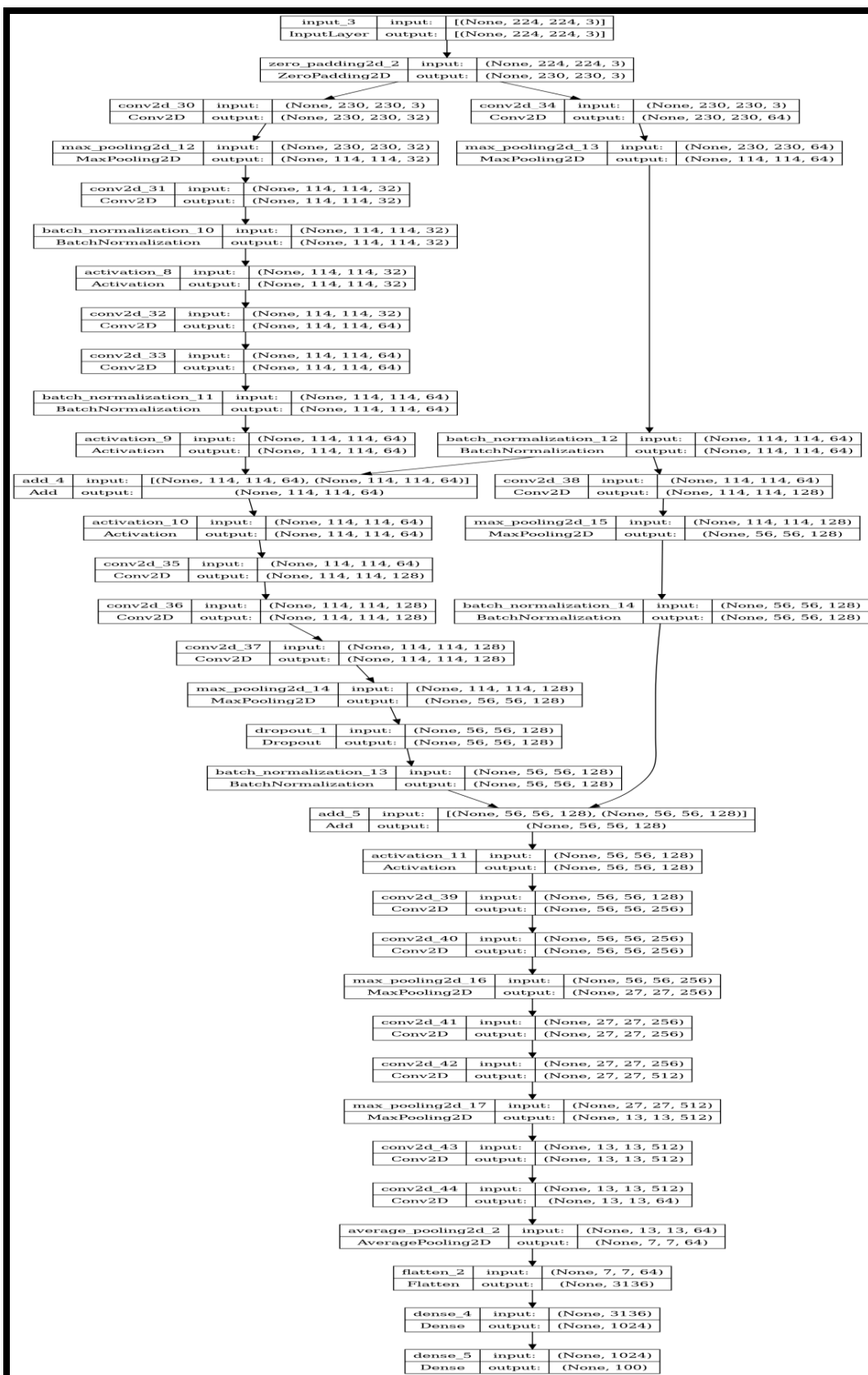


Figure 2: Visual representation of the custom model.

3.3. VGGNET

In our proposed methodology, we incorporate the VGGNet architecture, a widely recognized deep neural network for image classification. During training, input images with dimensions of 224×224 pixels are employed to train the VGG architecture, with the subsequently generated weights utilized for testing. As depicted in Figure 3, the architecture follows a pattern where the number of kernels increases with each deeper stage in the network [1]. Rectified Linear Units (ReLU) serve as the activation function in both convolutional and fully connected layers, excluding the final dense layer. The latter utilizes Softmax activation for multi-class classification. In the back propagation phase, Adam is employed as the optimization algorithm, while categorical cross-entropy serves as the loss function.

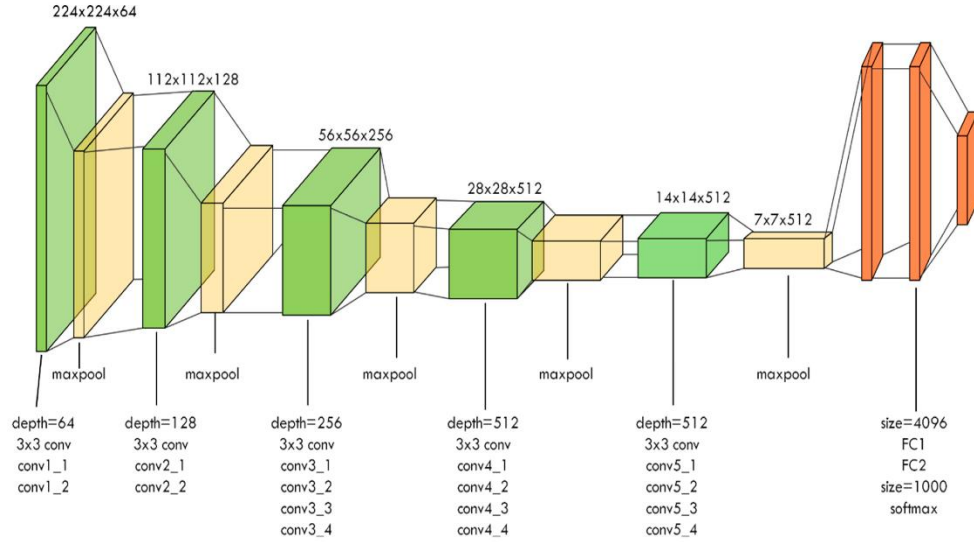


Figure 3: VGG-19 network architecture: conv stands for convolution, FC stands for fully connected [28].

3.4. RESNET

Our methodology also integrates a Residual Network (ResNet), the winner of the 2015 ImageNet challenge. Its remarkable performance is attributed to the introduction of 'skip connections' within its architecture [2]. During back propagation, conventional deep neural networks encounter the challenge of vanishing gradients, particularly in depth-scaled architectures, hindering model convergence. To counter this issue, ResNet introduces convolutional and identity blocks with inherent skip connections. These connections enable the network to bypass certain layers, mitigating the vanishing gradient problem. The final dense layer of the ResNet architecture is adapted to accommodate different classes, employing the Softmax activation function. It is noteworthy that the model is trained from scratch using our dataset.

3.5. Pre-Trained CNN Models and Transfer Learning

Deep CNN architectures, originally trained on extensive datasets with annotated images, yield valuable feature extractors, commonly referred to as model weights. These weights prove beneficial when training on smaller datasets, mitigating the risk of over fitting and reducing training time. Previous studies [18, 19] have emphasized the efficacy of generic features derived from pre-trained CNN models for tasks such as

image classification, object detection, and localization in natural images. Transfer learning, the process of leveraging information from a model trained on a similar domain, significantly enhances the performance of models trained on specific domains of interest. Notably, utilizing pre-trained models, especially those trained on ImageNet [18, 19], has demonstrated improved results with small datasets. Additional experiments [20, 21] highlight the superior classification outcomes achieved by deep CNN-based transfer learning when compared to knowledge derived from related tasks with extensive datasets. In a specific instance [22], authors replaced fully connected layers with logistic layers in a pre-trained classifier CNN model, freezing all layers during training except for the logistic regression layers. In our project, we have used fully connected layers as classifiers and trained on the above-mentioned datasets.

3.6. Fine-Tuning

To further enhance the effectiveness of the methodology and address its challenges, an advanced fine-tuning technique is employed. This process occurs subsequent to transfer learning, where a subset of CNN layers is unfrozen, allowing the weights to be trainable alongside features from the fully connected layers. This strategic fine-tuning step is crucial for adapting the pre-trained model to the specifics of the real-time dataset at hand. Previous work [23] exemplifies this by unfreezing all necessary CNN layers and training each weight using a dataset specific to distinguishing interstitial lung diseases. Results from experiments [24, 25] emphasize the consistent improvement in performance achieved through fine-tuning pre-trained CNN models, showcasing superior outcomes with the same datasets. Successful instances of fine-tuning are well-documented [26, 27]. Notably, we strategically choose to unfreeze a limited number of layers to prevent over fitting, recognizing the depth and complexity of state-of-the-art models. This selective approach ensures effective training on the collected dataset, while frozen layers retain weights derived from the ImageNet dataset, contributing to model generalization. Details regarding the number of unfrozen and trained layers are available in Table 2.

Table 2. Fine-tuning of deep CNN models.

S. No.	Model	Tot. No. of Layers	Unfrozen layers
1	ResNet50	179	49
2	VGG19	26	12

4. Model Training & Experimentation:

To ensure uniformity and facilitate effective model training, an input size of 224 by 224 pixels has been chosen for all three models: the custom model, VGGNet, and ResNet. The ImageDataGenerator has been employed to seamlessly load and preprocess the dataset. The deep learning models were constructed using the Keras library, with TensorFlow employed as the backend. The experiments make use of pre-trained neural networks from tensorflow.keras.applications. To enhance the computational efficiency and accelerate training processes, a Keras notebook equipped with GPU support is employed.

Given the scale of the datasets used in these experiments, the computational power is bolstered by utilizing the GPU T4 x 2 accelerator. The pre-trained weights are weights of the deep neural networks trained on ImageNet dataset.

The choice of batch size plays a crucial role in neural network training, as it dictates the number of samples processed in each iteration before the model updates its weights. Smaller batch sizes make the model more sensitive to individual data points, increasing the risk of over fitting. On the other hand, a high batch size demands robust GPU support. To strike a balance, we opted for a moderate batch size of 32, aligning with the training requirements of the discussed models. Considering the large size of the datasets, we have trained the model for 20 epochs and observed different training times for each model. Table 3 provides the training time for each model on Butterfly dataset.

Table 3. Average training time for each model on Butterfly dataset for one epoch.

S. No.	Model	Training time (in seconds)
1	Custom model	195
2	ResNet50	175
3	ResNet50(Fine-tuned)	179
4	VGG19	187
5	VGG19(Fine-tuned)	190

The learning rate is another crucial hyperparameter in machine learning training. When the learning rate is set to a high value, it may lead to erratic fluctuations in the weights of the loss function after each iteration. On the other hand, a learning rate that is excessively low can impede the convergence rate, resulting in a prolonged and sluggish learning process. Hence, we have employed dynamic learning rate with initial learning rate of 0.0001. The validation accuracy is monitored, and if there is no consecutive improvement for 2 epochs, the learning rate is reduced by a factor of 0.2. Additionally, we set a minimum learning rate of 0.00001 to prevent the learning rate from decreasing excessively. This dynamic learning rate approach aims to optimize the convergence speed while avoiding instability in the training process. We have implemented a dropout rate of 30% which involves randomly "dropping out" (i.e., setting to zero) a fraction of the neurons in the network during each iteration of training. This prevents any single neuron from becoming overly reliant on specific features, promoting the development of a more robust and generalized model. Table 3 gives a detail overview of the hyper-parameters used while training the neural network.

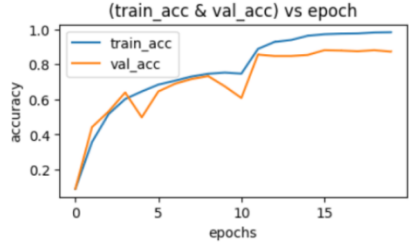
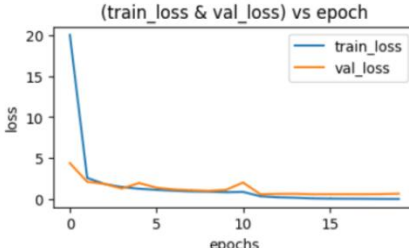
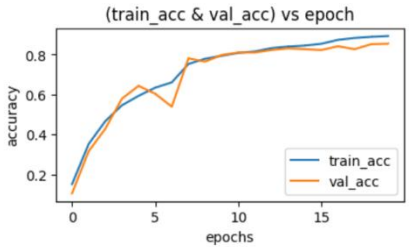
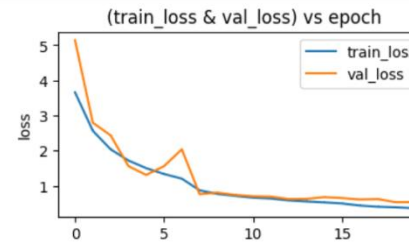
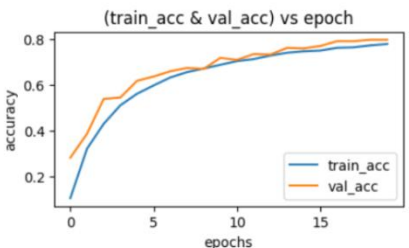
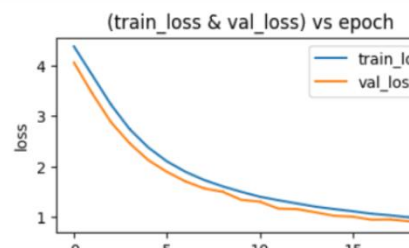
Table 3. Hyper-parameters used in the training phase.

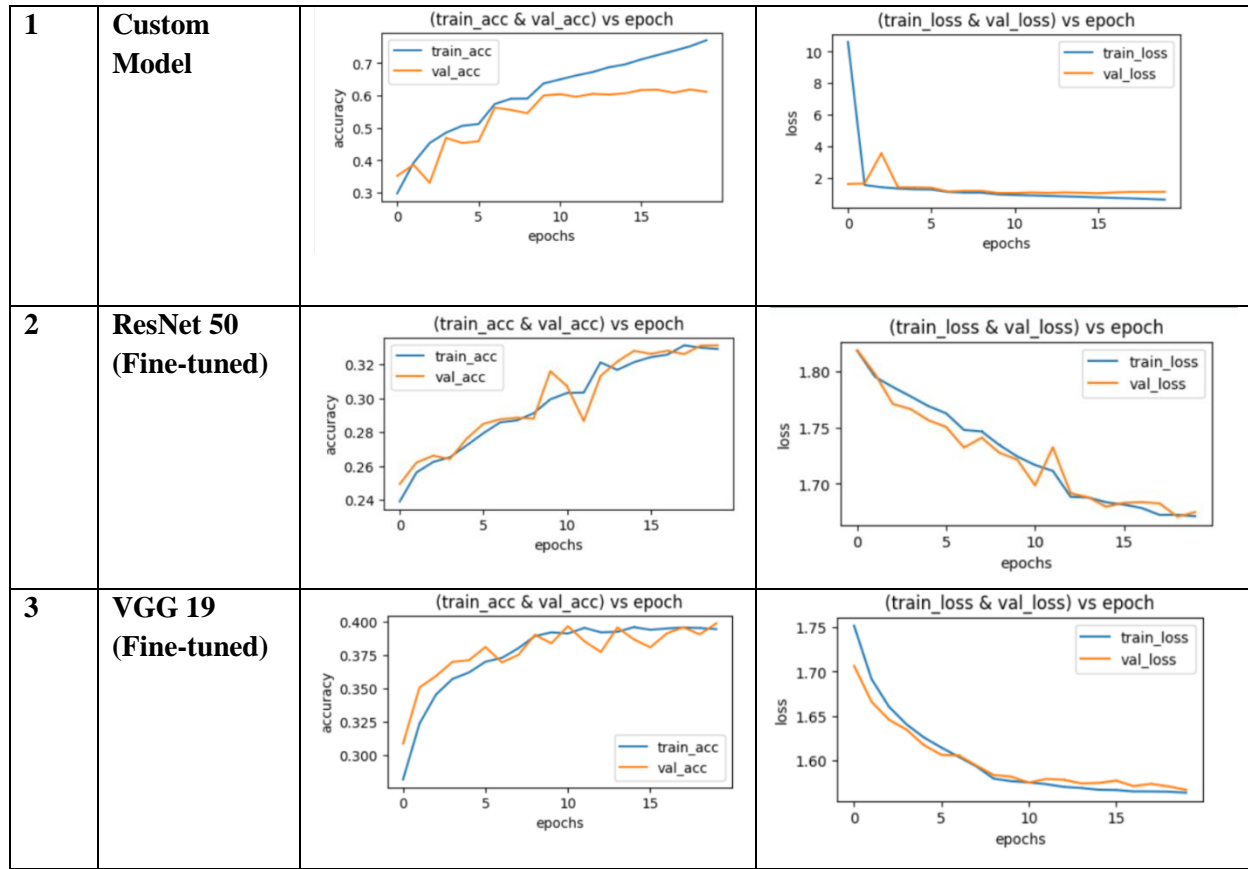
Hyper-parameter	Parameter used
Learning rate (LR)	Initial LR: 0.0001
Optimizer	Adam
Activation function	Relu, Final layer (Softmax)
Kernel Initializer	Glorot uniform
Batch size	32
Input size	224, 224
Dropout	0.30

5. Results and observations

Table 5 provides an overview of the accuracy and loss progression for both the training and validation sets. In the butterfly and moth species dataset, the training and validation accuracies aligned closely for all three models, indicating the absence of over fitting during the training phase. However, in the Facial Expression Recognition dataset, the custom model demonstrated an over fitting tendency as the gap between the training and validation accuracies continued to widen from the 10th epoch. This indicates that the model performed well in classifying known data (training set) with higher accuracy but struggled to generalize its performance to unknown data (validation set) at the same scale.

Table 5. Training accuracy and loss of Deep CNN models over 20 epochs.

S.No	Model	Loss	Accuracy
1. Butterfly and moth species			
1	Custom Model		
2	ResNet 50 (Fine-tuned)		
3	VGG 19 (Fine-tuned)		
2. Facial Expression Recognition			



In evaluating the performance of our Deep Convolutional Neural Network (DCNN) models, various performance metrics were employed and are thoroughly examined in this section.

a. Confusion matrix

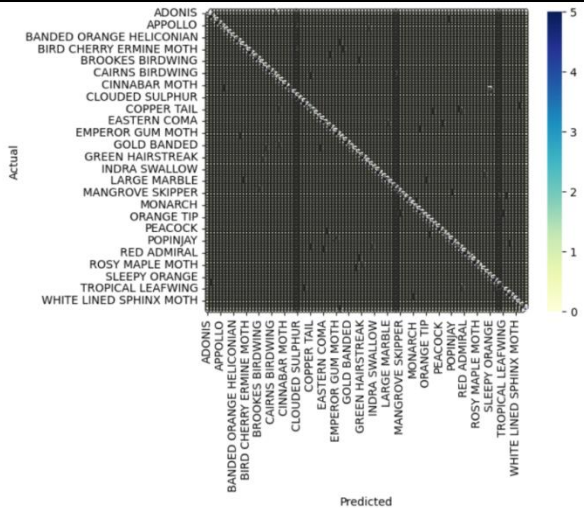
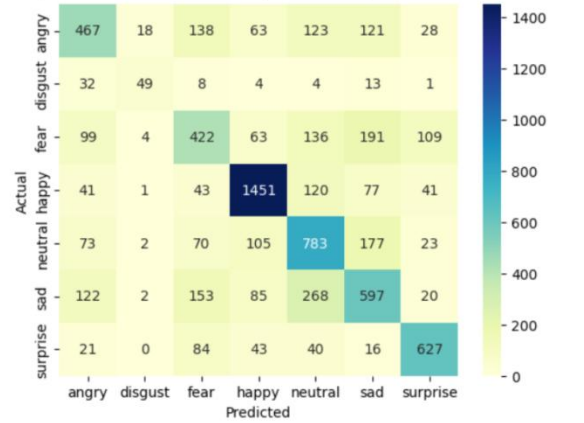
The confusion matrix visually illustrates the accuracy of the model's predictions, relying on elements like true positive (TP), true negative (TN), false positive (FP), and false negative (FN). TP signifies instances where both actual and predicted values are true. For instance, in facial emotion recognition, TP denotes accurately identifying angry faces. TN represents cases where both actual and predicted negative conditions are true, such as correctly not recognizing angry faces as other emotions like disgust or fear. FP accounts for predictions that were true compared to the actual false values. For instance, when the model identifies faces as surprise when they're actually angry. FN reflects instances where predicted values are false despite being true. For instance, when the model detects faces as angry when they're actually sad. The matrix offers insights into how accurately the model predicts emotions in facial recognition tasks. Here we are considering the matrix taking Actual on the row vector and Prediction on the column vector.

Confusion Matrix	Actual 0	Actual 1
Predicted 0	A	B
Predicted 1	C	D

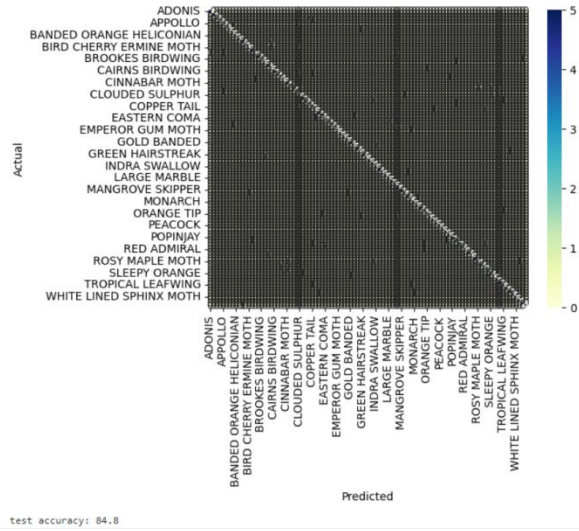
A = True Negatives, B = False Negatives, C = False Positives, D = True Positives

From Table 6, we can examine the confusion matrix of DCNNs for both datasets. In the Butterfly dataset, the custom model stands out in performance, as evidenced by the confusion matrix where the diagonal elements are white, indicating their higher values. Given the dataset's 100 classes, it might be challenging to notice significant observations. In the FER dataset, VGG19 stands out with the highest test accuracy of 67.84%, while ResNet50 underperforms with a test accuracy of 47.19%. Due to the very limited size of the disgust emotion class in both train and test samples, ResNet50 failed to predict the disgust emotion class completely. This is one of the disadvantages of having an unbalanced dataset.

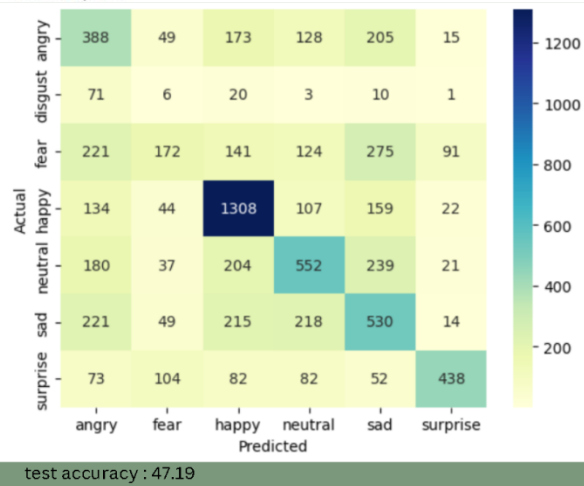
Table 6. Confusion matrix of the Deep CNN models for the test sets.

Type of dataset and model	Confusion matrix
Custom model (Butterfly dataset)	 <p>test accuracy: 88.8</p>
Custom model (Facial Expression Recognition)	 <p>test accuracy: 61.24268598495403</p>

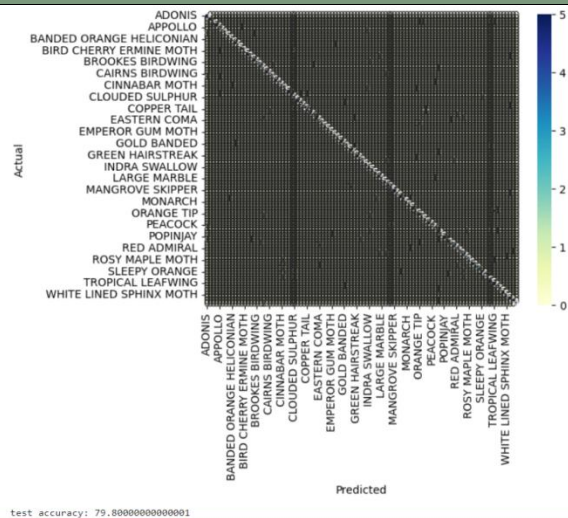
ResNet –50 [Fine-tuned] (Butterfly dataset)

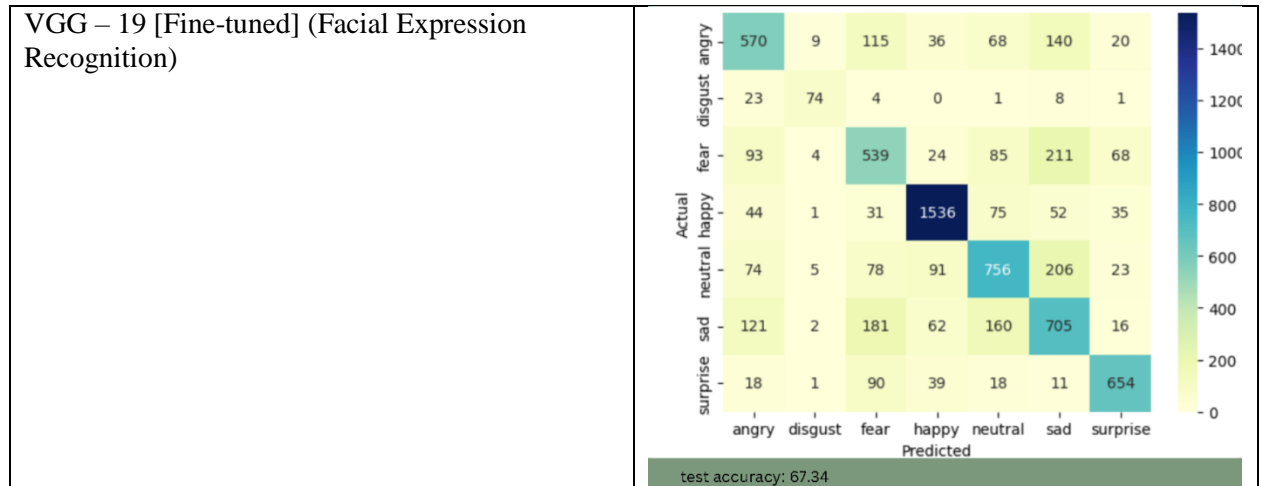


ResNet – 50 [Fine-tuned] (Facial Expression Recognition)



VGG – 19 [Fine-tuned] (Butterfly Dataset)





b. Accuracy

Accuracy in machine learning measures how well a model performs by gauging its correct predictions against the total predictions available in the dataset. For instance, in facial emotion detection, it quantifies the accurate identification of emotions such as anger, happiness, and others. Let's consider a custom model's predictions: 1451 for happy faces, 467 for angry faces, 49 for disgust, 422 for fear, 783 for neutral expressions, 597 for sadness, and 627 for surprise. Adding these correct identifications amounts to 4396. Meanwhile, the total faces in the dataset stand at 7178.

The accuracy computation involves tallying the total correctly identified emotions against the entire dataset: Correctly identified emotions / Total images in the dataset. For this scenario, it yields an accuracy of 61.2426% for the custom model's facial emotion recognition.

c. Precision

Precision measures how accurately a model predicts true values. It's calculated by dividing the number of true positives by the sum of true positives and false positives. Precision values range from 0 to 1, with higher values indicating better precision. In our facial emotion recognition models, the custom model achieves a precision of 0.6030, while the custom model's butterfly dataset reaches 0.9015. These precision scores contribute to an accuracy of 88.82%.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

d. Recall

Recall gauges the actual true positives among all the true samples in the dataset. Precision, which varies between 0 and 1, indicates better precision with higher values. In our facial emotion recognition model, the custom model achieves a recall of 0.5357, whereas the custom model's butterfly dataset attains 0.8880.

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

Table 7. Performance analysis of DCNNs on both the datasets.

Algorithm	Accuracy	Total Trainable parameters	Precision	Recall	F1- Score
Butterfly and Moth Dataset					

Custom Model	88.82	90,40,807	0.9015	0.8880	0.8846
ResNet50	8.24	22,00,676	0.1246	0.16	0.1216
ResNet50(Fine-Tuned)	84.8	1,82,97,444	0.8681	0.8480	0.8433
VGG19	11.21	6,27,812	0.1323	0.1594	0.1454
VGG19(Fine-Tuned)	79.80	1,83,26,628	0.8350	0.7980	0.7987
Facial Expression Recognition					
Custom Model	61.2426	90,40,807	0.6030	0.5753	0.5853
ResNet50	12.3014	44,59,527	0.2777	0.2493	0.2267
ResNet50(Fine-Tuned)	47.1997	1,82,02,119	0.4039	0.3872	0.3854
VGG19	11.9671	5,32,487	0.3164	0.3057	0.2936
VGG19(Fine-Tuned)	67.3446	1,46,91,335	0.6761	0.6599	0.6673

Table 7 presents the performance analysis of the DCNNs on both datasets. For the Butterfly and Moth Dataset, the custom model exhibited the best performance with the highest accuracy, precision, recall, and F1-score, followed by ResNet50 (Fine-tuned) and VGG19 (Fine-tuned). The transfer learning of ResNet50 and VGG19 experienced significant challenges in classifying the species, highlighting the substantial improvement achieved by unfreezing certain layers of the DCNN model. In the case of FER dataset, VGG19 (fine-tuned) has proven to be the best model with highest performance metrics, followed by Custom model and ResNet (fine-tuned). The total trainable parameters represent the number of weights and biases that the network can adjust during training to learn the data. Training with high number of parameters requires more computational resources such as memory and processing power. The custom model has the fewest trainable parameters, with 90,40,807, compared to ResNet50 (Fine-tuned) and VGG19 (Fine-tuned), which have 1,82,97,444 and 1,83,26,628 parameters, respectively. This observation indicates that our custom model has delivered better performance with the least consumption of computational resources, which was the main objective of our project.

The unbalanced datasets led to biased results for some of the DCNN models. Hence balancing the dataset by applying different data augmentation techniques could provide more robust models. As per our knowledge, this is the first work focused on classifying butterfly and moth species with 100 types. Extensive training on this dataset could lead to better performance model which can be later used in ecological studies. In the case of the FER dataset, the custom model can be equipped with additional over fitting prevention methods, such as regularization and early stopping, to achieve a more robust model.

References

- [1] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. 2014 Sep 4.
- [2] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
- [3] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural computation, 1989.

- [4] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.
- [5] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. In ECCV, 2014.
- [6] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In ICLR, 2014.
- [7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In CVPR, 2015.
- [8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. arXiv:1409.0575, 2014.
- [9] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015.
- [10] S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. Diploma thesis, TU Munich, 1991.
- [11] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2):157–166, 1994.
- [12] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In AISTATS, 2010.
- [13] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Muller. Efficient backprop. " In Neural Networks: Tricks of the Trade, pages 9–50. Springer, 1998.
- [14] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. arXiv:1312.6120, 2013.
- [15] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In ICCV, 2015.
- [16] Goodfellow, I.J., Erhan, D., and Carrier, P.L.: ‘Challenges in representation learning: a report on three machine learning contests’. Neural Information Processing Systems (NIPS), Lake Tahoe, NV, USA, December 2013, pp. 117–124
- [17] @article{butterflies, title={Butterfly and moth Image Classification 100 species}, author={Gerald Piosenka}, month={November}, year={2022}, url={<https://www.kaggle.com/datasets/gpiosenka/butterfly->

`images40-species},`
`publisher= {Kaggle} }`

- [18] Oquab, M.; Bottou, L.; Laptev, I.; Sivic, J. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, 23–28 June 2014; pp. 1717–1724.
- [19] Razavian, A.S.; Azizpour, H.; Sullivan, J.; Carlsson, S. Cnn features off-the-shelf: An astounding baseline for recognition. *arXiv* **2014**, arXiv:1403.6382.
- [20] Kieffer, B.; Babaie, M.; Kalra, S.; Tizhoosh, H.R. Convolutional neural networks for histopathology image classification: Training vs. Using pre-trained networks. In *Proceedings of the 2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, Montreal, QC, Canada, 28 November–1 December 2017; pp. 1–6.
- [21] Li, X.; Pang, T.; Xiong, B.; Liu, W.; Liang, P.; Wang, T. Convolutional neural networks based transfer learning for diabetic retinopathy fundus image classification. In *Proceedings of the 2017 10th International Congress on Image and Signal Processing, Biomedical Engineering and Informatics (CISP-BMEI)*, Shanghai, China, 14–16 October 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–11.
- [22] Carneiro, G.; Nascimento, J.; Bradley, A.P. Unregistered multiview mammogram analysis with pre-trained deep learning models. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*; Springer: Cham, Switzerland, 2015; pp. 652–660.
- [23] Gao, M.; Bagci, U.; Lu, L.; Wu, A.; Buty, M.; Shin, H.-C.; Roth, H.; Papadakis, G.Z.; Depeursinge, A.; Summers, R.M.; et al. Holistic classification of ct attenuation patterns for interstitial lung diseases via deep convolutional neural networks. *Comput. Methods Biomech. Biomed. Eng. Imaging Vis.* **2015**, *6*, 1–6.
- [24] Shin, H.-C.; Roth, H.; Gao, M.; Lu, L.; Xu, Z.; Nogues, I.; Yao, J.; Mollura, D.; Summers, R. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE Trans. Med. Imaging* **2016**, *35*, 1285–1298.
- [25] Tajbakhsh, N.; Shin, J.; Gurudu, S.; Hurst, T.; Kendall, C.; Gotway, M.; Liang, J. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE Trans. Med. Imaging* **2016**, *35*, 1299–1312.
- [26] Azizpour, H.; Razavian, A.S.; Sullivan, J.; Maki, A.; Carlsson, S. From generic to specific deep representations for visual recognition. In *Proceedings of the Computer Vision and Pattern Recognition Workshops*, Boston, MA, USA, 7–12 June 2015; pp. 36–45.
- [27] Penatti, O.A.B.; Nogueira, K.; Dos Santos, J.A. Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? In *Computer Vision and Pattern Recognition Workshops*; IEEE: Piscataway, NJ, USA, 2015; pp. 44–51.

[28] Reka SS, Murthy Voona VD, Sai Nithish PV, Paavan Kumar DS, Venugopal P, Ravi V. Performance Analysis of Deep Convolutional Network Architectures for Classification of Over-Volume Vehicles. *Applied Sciences*. 2023; 13(4):2549. <https://doi.org/10.3390/app13042549>

For details of the code, please refer to our Github repository:

https://github.com/dhanvanth342/Group14_CS584