ISSN No:-2456-2165

Creation of an Android Application and the Use of Transfer Learning to Recognize Insect Species

Prabhat Kumar Singh*
Electronics and Communication Engineering
Maharaja Surajmal Institute of Technology (GGSIPU)
Delhi, India

Pawan Kumar
Electronics and Communication Engineering
Maharaja Agrasen Institute of Technology (GGSIPU)
Delhi, India

Abshar Imam
Electrical and Electronics Engineering
Maharaja Agrasen Institute of Technology (GGSIPU)
Delhi, India

Abstract:- Insect numbers are dwindling over the world, and some species have gone extinct in the past. Exploration of seldom seen Insect species has therefore become a difficult endeavor for Entomologists and Insect Watchers. We created an Android application based on deep learning to assist users in recognizing 280 different insect species, making insect categorization much more user-friendly. We employ Convolutional Neural Networks (CNN) pre-trained on ImageNet Dataset as freeze layers of the network in this article, then train the final output layer, which has 280 separate classes. The accuracy of CNN models such as EfficientNet-lite0, InceptionV3, Xception, ResNet-50, MobilenetV2, and InceptionResNetV2 has been evaluated, and the mobile app's operation has been discussed. Maximum train data accuracy of 99.81 percent and test data accuracy of 98.62 percent is accomplished.

***Keywords*:-** *Transfer Learning, Classification of Bird Species, Deep Learning, Recognition of Image, CNN, Android Application.*

I. INTRODUCTION

Following favorable results in psychoanalysis, it has been proven that viewing insects and hearing to their melodies can assist in psychotherapy. Learning how the number and diversity of essential well-being benefit providers, such as insects, contribute to nature's total advantages could be crucial for determining how best to use them.

Entomologists study insect ecology under extreme situations, conduct research, and develop strategies to increase species' survival because insects are so important to humans' existence.

Insect species categorization has become a serious challenge after taking everything into account. A user-friendly mobile application is simple to carry about and may provide Entomologists with several benefits. This application may be more useful for an Insect watcher who sees an unnoticed Insect, takes a picture of it, and then identifies the species.

There are a variety of strategies that can help in picture categorization. Deep convolutional networks may be used to extract features from pictures in a computer vision approach called image classification. The Keras core library includes some of the best-performing Convolutional Neural Networks (CNNs), which can categorize 1000 different object categories and are largely trained on the ImageNet dataset.

These pre-trained Neural networks aren't only limited to ImageNet data; they can also generalize over a wide range of categories thanks to transfer learning, which includes feature extraction and fine-tuning.

Convolutions are preferred over completely linked layers because they allow for parameter sharing and connection sparsity. The key difference with CNN is that by conducting the Convolution process, the initial image matrix's dimensions are reduced to a smaller dimension in the first layer itself. Despite this, the network's final layer is a fully linked layer resulting from convolution and label classification of the picture matrix.

We're employing TensorFlow Hub pre-trained models as the foundation for our CNN network, and we're evaluating model accuracy using two optimizers: Adam and RMSprop. As a result, the focus of our efforts is on developing a flexible platform for facilitating access to such resources. The description of CNN, Transfer Learning, software design, and how the application works are all included in the following parts and subsections. There has been no study on huge species categorization up to 280 classes, which is why we have created a useful software with improved accuracy. In addition to the test dataset, we test our software on a variety of other images. We have used drawing and cartoon images to test the app's functionality, and it does a fantastic job on them.

II. APPROACH AND TECHNIQUES USED

A. Convolutional Neural Network

The Convolutional Neural Network (CNN) is a kind of neural network that uses convolutional Neural Network (CNN) , also known as ConvNet and is combined with the feed- forward architecture. When compared to other networks with completely linked layers, ConvNets have an incredible

capacity to generalize. The principle of weight sharing is critical to the success of CNN. As a result, the number of training parameters is drastically decreased, resulting in effective generalization [1]. As a result, parameters are lowered, the CNN training process is resolved, and the odds of overfitting are minimized [2]. CNNs are now widely employed in a variety of fields, including image classification, object identification, face detection, voice recognition, vehicle recognition, facial expression recognition, and many more, thanks to their superior performance. The following components can be used to characterize CNN:

- An input layer is a matrix of input raw pictures with the dimensions HxWxC, where H is the image's height, W is its width, and C is the number of channels employed. RGB pictures, for example, have three colours, hence the number of channels is three.
- The Convolution Layer is a smaller dimensional matrix than the input layer. It employs a filter matrix and conducts convolution across the whole size of the input matrix while maintaining a constant stride. The output layer's dimension same amount of parameters. Xception has a linear of depthwise separable convolution layers preceded residual connections as its architecture. In this section, a theory is advanced that is more prominent than the hypothesis of the may be evaluated by formula

$$\left(\frac{n+2p+s}{s}\right) + 1$$

Inception design, and it explains how cross-channel interactions and spatial interactions in the feature maps of convolutional neural networks may be completely

In the following equation, n denotes the layer size of the input, p represents padding, f specifies the number of channels used, and s is used for stride.

- Pool layers, also known as down sampling, decrease the size of the layer to speed up calculation. It employs two methods: average pooling and maximum pooling. The most common pooling method is max-pooling. When a filter is applied to different sections of the matrix, the max-pooling approach assigns the maximum value to the output layer for that specific region.
- Fully Connected Layers are the network's last layer, corresponding to output classes.

➢ *Transfer Learning*

We deployed six CNN models trained on huge picture datasets such as Imagenet in this study. The transfer learning approach makes pre-trained networks relevant for analogous new situations. Transfer learning is often applicable when the current dataset to train is less than the initial dataset used to train the pre-trained model [3]. In this study, we train a dataset of insect species using prominent models as EfficientNet-Lite0, ResNet50, Xception, InceptionV3, InceptionResnetV2, and MobilenetV2. Furthermore, we employ these models as the neural network's freeze layer to extract image features and substitute the network's final softmax layer to generate 280 output classes, evaluating the performance of the stated pre-trained neural networks based on their accuracy.

EfficientNets [4] are a series of models that offer substantially higher accuracy than earlier Convolutional Neural Networks. On the ImageNet dataset, EfficientNet-B7 achieves state-of-the-art top-1 accuracy of 84.3 percent. EfficientNet-B7 is also 8.4 times smaller and 6.1 times quicker than the best-known Convolutional Network. EfficientNet models have the benefit of being easily scaled up, and by employing fewer parameters, models can outperform state-of-the-art accuracy.

The Lite version of EfficientNet runs on TensorFlow Lite and is designed to run on mobile CPUs, GPUs, and EdgeTPUs. EfficientNet-Lite comes in five ranges, ranging from Lite0 to Lite4, with Lite0 being the smallest and Lite4 being the largest.

ResNet50 [5] is built using residual networks created by connecting blocks of 34 parameters in the ImageNet design with skip connections. To develop it a 50-layer ResNet, these skip connections were made using 2-layer blocks in a 34-layer design, and each 2-layer block was switched with a 3-layer bottleneck block.

InceptionV3 [6] is the runner-up for image classification at the 2015 ILSVRC. It is an optimised version of GoogleNet (also known as InceptionV1 and InceptionV2). Inception Networks are useful for large volumes of data since they have a lower processing cost than VGGNet. It has fewer parameters and can be accessed if memory is constrained. The **Xception** [7] and Inception V3 architectures have the decoupled.

InceptionResNetV2 [8] is a leftover form of inception networks which employs 1 1 convolution blocks without activation and utilizes cheaper Inception blocks. Filter-expansion layers are another name for these block confluences. This layer is being used to scale up filter banks and then match the input depth by conducting addition to compensate for the Inception block's dimensionality reduction. InceptionRes-NetV2 and the newly introduced Inception-v4 networks have the same raw costs. Batch-normalization was only implemented on top of the conventional layers in this network, unlike non-residual Inception networks.

MobileNetV2 [9] the first fully convolutional layer with 32 filters is accompanied by 19 residual bottleneck layers. The term bottleneck comes out from neck of a bottle, and it refers to the layer that has less dimensions than the node before it. The framework of MobileNetV2 is constructed on depth-separable linear bottleneck convolution with inverted residuals. It just reduces computation costs by a tiny decline in accuracy when compared to standard convolutions. This is ideally suited for smart phone computer vision models because of its lightweight feature.

III. WORK PROPOSED

A. Dataset

We're using a Kaggle dataset including 280 insect species. The model may overfit when a network is well adapted to a small number of data points. It means the model perceives noise and random abnormalities in training data as a concept, that will have a negative impact on new data. As a result, the term "data augmentation" is used to represent the process of arbitrarily removing 224 224 patches from the original photographs, blurring, colour contrast, and zooming in and out to increase the dataset magnitude. The ImageDataGenerator class has been used to resize pixel values from 0-224 to 0-1 for neural network models before training.

B. Comparison of algorithms (1)Architecture of the model

EfficientNet-Lite0 has been the model that we eventually deployed on Android, and its architecture consists of an input layer, one Entirely-connected layer, and the end softmax layer in outcome. As a result, each layer consists convolution. Pre-trained models have been used for feature extraction. E very convolution layer in the EfficientNet-lite0 model uses ReLU 6 activation for making the list of these characteristics. The MBConvterm is recognized as an Inverted Residual Block in architecture. In Figure 1, the terms layers, channels, resolution, and operator refers to the size of both the filter and operator, the dimension of the input matrix, the number of channels, and the number of each time block is repeated accordingly, Skip connections are utilized internally in convolutional design. It is, as the name implies, a technique for bypassing some layers and feeding the outcome of one layer towards the next non-skipped layer. The bypass connection is a different route that allows the gradient to be more converged. In Fig. 2, we selected a picture and used it as an input, displaying the entire feature extraction and classification method.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224 × 224	32	1
2	MBConv1, k3x3	112 × 112	16	1
3	MBConv6, k3x3	112 × 112	24	2
4	MBConv6, k5x5	56 × 56	40	2
5	MBConv6, k3x3	28 × 28	80	3
6	MBConv6, k5x5	14 × 14	112	3
7	MBConv6, k5x5	14 × 14	192	4
8	MBConv6, k3x3	7 × 7	320	1
9	Conv1x1 & Pooling & FC	7 × 7	1280	1

Fig. 1. EfficientNet's Architectural style

- *Experimental Work*

For improving the CNN's performance we used transfer learning. We froze the pre-trained model and only changed the Neural Network's final softmax layer. EfficientNet-Lite, Xception, and MobileNetV2 are all associated together.

By training the InceptionV3, and InceptionResNetV2 models on the 280 insect species dataset, we were able to determine their correctness. To train the model, we used a free online cloud-based Collaboratory with GPU. The figure of parameters that each model is made up of is shown in Table I. In general, the larger the model and the more parameters it contains, the better the result. However, this raises prediction delay, as well as using extra energy and running slowly on a big scale, gradually heating up the device. A huge model also increases the size of the software. A smaller model will operate quicker and lower power consumption than a larger one, but it will give less accuracy. To close this gap, we used the Efficient-Lite0 model, that is smaller than Table I but has greater accuracy. Using the fine-tuning strategy, we enhanced the accuracy of the Efficient-Lite0 model by eliminating some layers, adding certain dense layers with activation functions, and adding some normalizing layers.

Optimizers are that kind of algorithms which alter weights and learning rates in order to decrease losses and improve efficiency. This model is being trained by using Adam and RMSprop optimizers.

Rmsprop optimizer is a gradient descent acceleration technique. When we finally implement gradient descent, there will be oscillations in both the vertical and horizontal directions. To slow down the vertical learning rate, we utilise the RMSprop optimizer, which attempts to increase the learning rate in the horizontal direction by taking larger steps.

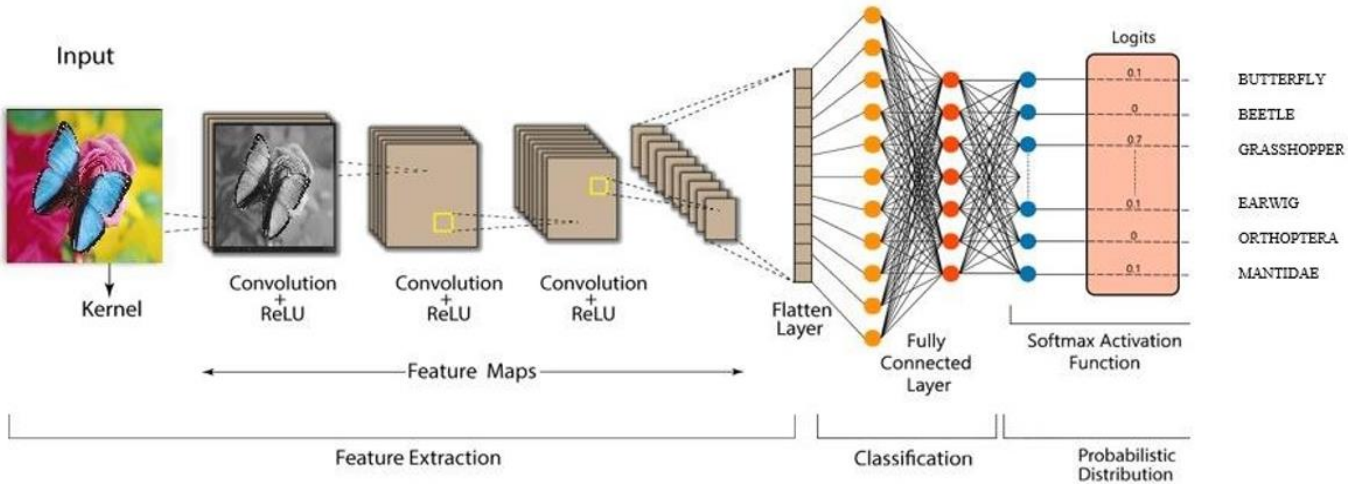


Fig. 2. Architecture of EfficientNet. TABLE I.SIZE COMPARISON OF MODELS

Model	Parameters (Millions)
InceptionResnetV2	55.97
InceptionV3	23.83
ResNet-50	23
Xception	22.8
EfficientNet-lite0	4.7
MobileNetV2	**3.4**

Table 1:- Size Comparison Of Models

Adam refers for Adaptive Momentum estimation; it combines momentum and RMSprop with a sparse gradients handling technique.

$$mt = \beta 1 mt{-}1 + (1 - \beta 1)gt \quad (1)$$

Model	Train (%)	Test (%)	Test Loss	Run Time (s)
InceptionV3	94.69	95	0.210	113s
InceptionResnetV2	90.62	92.77	0.272	130s
Xception	99.30	95.31	2.711	116s
MobileNetV2	**99.71**	96.93	0.092	**94s**
ResNet50	97.36	97.26	0.092	105s
EfficientNet-Lite0	97.46	**98.76**	**0.047**	106s

Table 2:- Model Comparison With Adam Optimizer

Model	Train (%)	Test (%)	Test Loss	Run Time (s)
InceptionV3	97.12	92.45	0.23	118s
InceptionResnetV2	94.26	92.62	0.26	130s
Xception	98.67	94.67	12.33	108s
MobileNetV2	99.73	96.47	0.12	95s
ResNet50	99.53	96.76	0.1	99s
EfficientNet-Lite0	**99.81**	**98.62**	**0.059**	**90s**

Table 3:- Model Comparison with Rmsprop Optimizer

$$vt = \beta 2 vt{-}1 + (1 - \beta 2)gt^2 \quad (2)$$

m & v represents moving averages, and g is the gradient on the current mini batch in the equations above. These may be used to calculate the mean (first instant) and variance (second moment) (second moment).

For the moment, $\beta 1$ and $\beta 2$ are exponentially decaying percentages or we can say hyper-parameters for evaluating exponentially weighted mean.

$$\hat{m}t = mt/(1 - \beta^1 t) \quad (3)$$

$$\hat{v}t = vt/(1 - \beta^2 t) \quad (4)$$

Equations (3) and (4) are bias-corrected estimators for the first and second moments, respectively. Tables II and III show the performance of several models with the Adam and RMSprop optimizers, respectively. Run Time refers to the amount of time it takes for a model to be trained per epoch. Tables II and III show that Adam optimizer is superior than RMSprop (105s) for training the model in less time and with higher accuracy (90s). Table II shows that EfficientNetLite0 with Adam optimizer performs the best on train data, with an accuracy of 99.81 percent. MobileNetV2 performs remarkably well with the RMSprop optimizer shown in Table III, obtaining an accuracy of 99.71 percent, which is greater than EfficientNet-Lite0 (97.47 percent).

We are loading the trained model weights in (.h5) extension at the time of testing. After additional analysis, we can find that EfficientNet-Lite0 has the best test accuracy with both the optimizers, including Adam (accuracy 98.62 percent) and RMSprop (accuracy 98.76 percent), and the least loss on test data of all. We created complete two-dimensional graphs in Fig. 3 and 4 to better visualize the comparison of all model's accuracy using a popular Py-thon tool called Matplotlib.

Lastly, we analyzed EfficientNet-Lite0 using Adam optimizer since it performs better over both test & train data and computational cost than EfficientNet--Lite0 with RMSprop. The finished model's Confusion Matrix is represented in Figure 5.

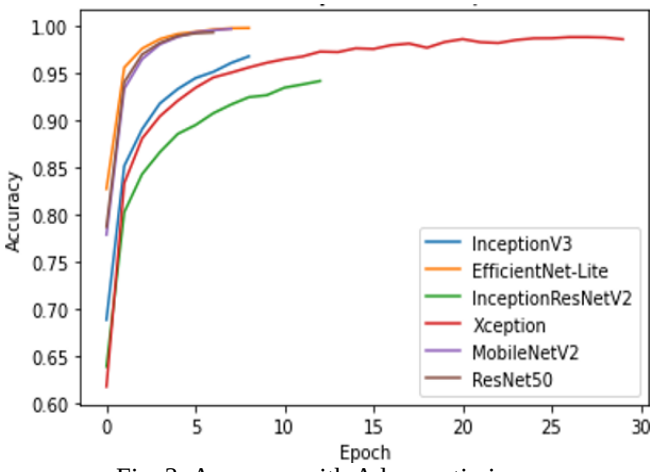


Fig. 3. Accuracy with Adam optimizer.

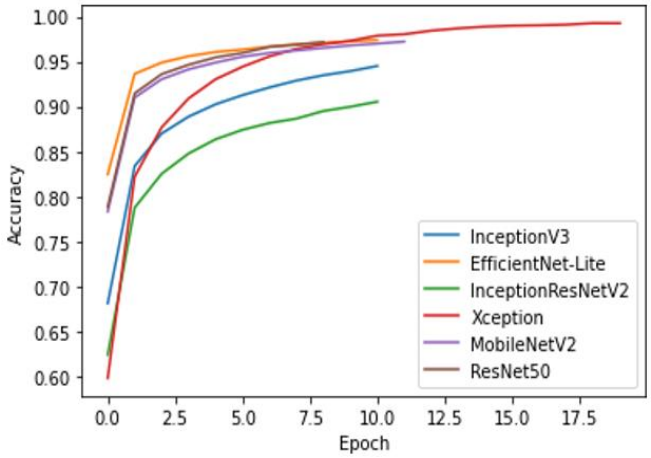


Fig. 4. Accuracy with RMSprop optimizer.

A model's weight is another important attribute to have before it can be deployed on the Android platform. We can see from Table I that MobileNetV2 has strong training accuracy and minimum weight, but EfficientNet-Lite0 has superior test accuracy and test loss, resulting in the second lighter model after MobileNetV2, therefore we install the TFLite version of EfficientNet-Lite0 on the android application.

C. Implementation and working of model on android app

The app's functionality is seen in Figure 6. As demonstrated, clicking on the Load Picture button will take you to the gallery; once the image has been added, clicking on Identify will display information about the image on the screen.

The values of the weights, biases, gradients, and other variables were recorded in the checkpoint file as an extension (.ckpt) once the model was trained. We changed the stored model to (.tflite) format in order to further deploy this trained model on Android. The TFLite file aids in the reduction of model file size and introduces non-accurate optimizations.

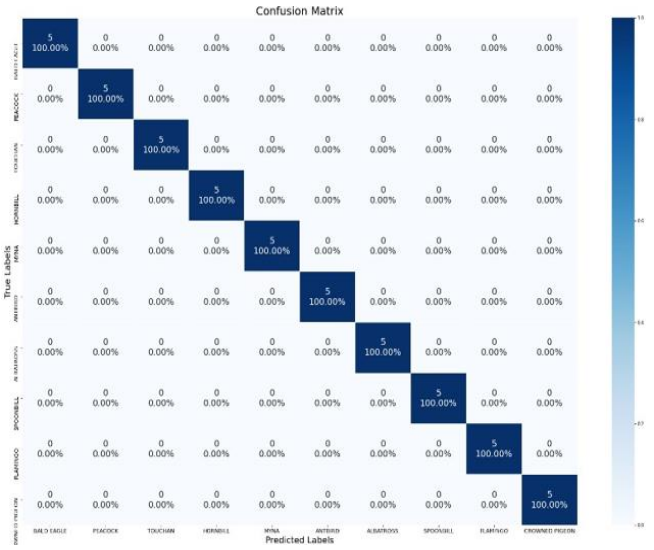
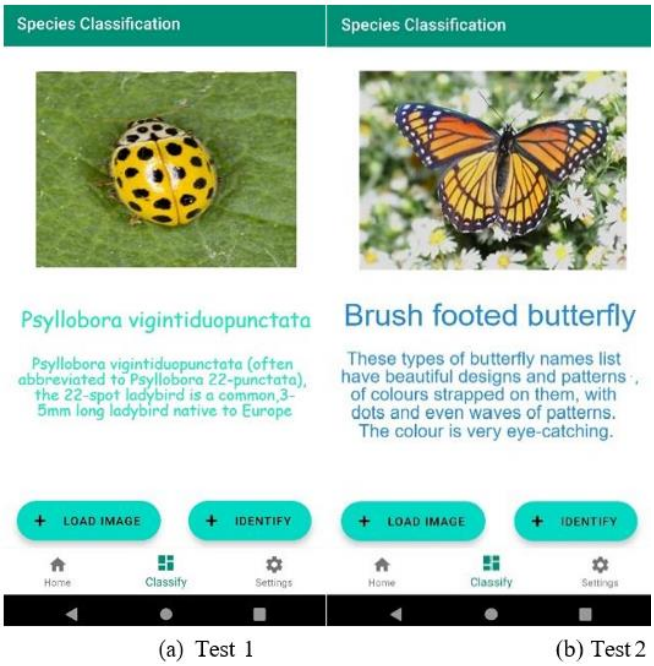


Fig. 5. Confusion Matrix.



(a) Test 1 (b) Test 2

Fig. 6. App interface for determining insect species.

The label.txt file, which contained all 280 insect classes, and the model.tflite files were uploaded to the IDE (integrated development environment), and the image was predicted using a computing job.

Home, Classify, and Settings are the three activities that make up this user-friendly software. As illustrated in Fig. 7(a), it represents a card view in the home, with numerous species' pictures and names. All identification procedures are carried out in the Classify section with the user's assistance, including the provision of name and information about specific species. The backdrop theme may also be changed from light to dark under Settings. We upload photos of drawings and cartoons to our program to further test it, and it accurately classifies them as shown in Figure 7(b).

(a) Test on Drawing of Insect

Fig. 7. App interface for determining insect species.

IV. CONCLUSION AND FUTURE SCOPE

Finally, we applied EfficientNet-Lite0 to determine insect species from submitted photos. The planned Android app includes a categorization system for 280 insect species. The app does not require an internet connection and displays the result in real time, saving you time. In comparison to previous studies, our app provides a greater number of insect species classifications, with a model accuracy of 98.62 percent on testing data. The derived F1 score is 0.9859, with Precision and Recall of 0.9901 and 0.9861, respectively. It outperforms the MobileNetV2 model, which was created specifically for its lightweight and high accuracy results on single picture categorization. When compared to its competitors, our model offers the lowest test loss.EfficientNet-lite0, to our knowledge, produces the greatest results, and our implementation in the Android app further confirms that EfficientNet-lite can eventually replace MobileNets.

Future development might involve testing the app in real- world circumstances and using the fresh data to further train the model for improved outcomes. Because insects are eaten as a meal they can be recognized by the consumer. Moreover, the app can be used by the farmers to detect a particular type of insect that is destroying their crop so that they can use specific pesticides. More innovative applications can be created with a larger dataset to help scientists. Furthermore, the software may be enhanced with sounds of various insects, allowing a normal user to identify an insect based just on the sound captured in their environment.

REFERENCES

[1]. S. Indolia, A. Goswami, S. Mishra, and P. Asopa, "Conceptual under- standing of convolutional neural network- a deep learning approach," *Procedia Computer Science*, vol. 132, pp. 679–688, Jan 2018.

[2]. Y. Wang, Z.-P. Bian, J. Hou, and L.-P. Chau, "Convolutional neural networks with dynamic regularization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 2299–2304, May 2021.

[3]. R. Ribani and M. Marengoni, "A survey of transfer learning for convolutional neural networks," in *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*, pp. 47–57, 2019.

[4]. M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for con- volutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114, PMLR, Jun 2019.

[5]. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[6]. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Confer- ence on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818– 2826, 2016.

[7]. F. Chollet, "Xception: Deep learning with depthwise separable convo- lutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, 2017.

[8]. C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelli- gence*, AAAI'17, p. 4278–4284, AAAI Press, 2017.

[9]. M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," pp. 4510– 4520, Jun 2018