Review of Deep Transfer Learning Models for Image Classification

https://doi.org/10.3991/ijes.v10i01.29783

Neelesh Karthikeyan(✉)
College of Engineering Guindy, Anna University, Chennai, India
neelesh.ceg@gmail.com

Abstract—With the rapid rise in urbanization, solid waste generation has increased exceedingly. This study aims to develop a Convolutional Neural Network (CNN) to classify biodegradable and non-biodegradable wastes. The TrashNet dataset was used in this study, and image augmentation was employed to make the models robust against translation invariance. Transfer Learning methods based on CNN have shown promising outcomes on diverse image classification problems. This paper reviews the deep learning models available with pre-trained weights in the Keras library. The performance metrics of the models were compared, and the model based on NASNetMobile had the highest accuracy of 97%. The hyper-parameters were tuned, and the significance of each hyper-parameter on a model's accuracy was studied.

Keywords—deep learning, transfer learning, waste classification, CNN, hyper-parameter, TrashNet

1 Introduction

Waste segregation has become an increasingly widespread problem in the recent years. According to a study published by the World Bank in 2018 [1], 242 million tonnes of plastic waste were produced globally in 2016, accounting for 12% of total solid waste. Non-biodegradable wastes persist in the environment for a long time. They do not naturally degrade, causing havoc on the environment. Biodegradable garbage has no negative environmental impact. They are degraded by natural forces such as fire, water, air, microorganisms, and soil. Convolutional Neural Network (CNN) models are used in deep learning to train and test large datasets of images [2]. Every input image will be processed by a set of convolutional layers that include filters, pooling layers, and fully connected layers [3]. Transfer learning is a powerful deep learning technique that can aid in solving the problem of insufficient training data. Because of its wide range of applications, transfer learning has become a promising area in machine learning [4]. This paper aims to review the Transfer learning models in the Keras library and train CNNs to sort waste into biodegradable and non-biodegradable categories. In addition, the impact of hyper-parameters like activation function, batch size, learning rate, and optimizers on waste image classification has been investigated. The best

hyper-parameters are selected based on the CNN model's train, test, and validation accuracy. The following is a break-down of the paper's structure. The second section discusses a review on the related works. In the third section, the dataset and waste classification methodologies are demonstrated. The concluding section summarizes the study's findings.

2 Related works

TrashNet is a dataset created by Yang and Thung [5] containing 2527 images divided into six categories: glass, paper, cardboard, plastic, metal, and trash. The TrashNet dataset has been explained in detail in the upcoming sections. Bircanoglu et al. [6] developed RecycleNet, a lightweight CNN model for trash classification. Even though RecycleNet only achieved around eighty percent accuracy on the TrashNet, it reduced time complexity by reducing the number of parameters from seven to seven to three million. AlexNet was proposed in 2012 by Alex Krizhevsky et al., and it performed well in the image categorization challenge [7]. Kennedy et al. used the Visual Geometry Group 19 (VGG-19) [8] to classify trash and attained an accuracy of 88 percent, proving the utility of VGG-19's ability to extract features [9]. Radiuk [10] looked at the effect of batch size on CNN image classification performance in 2017. The larger the batch size, the higher the network accuracy, according to the author's findings, implying that batch size has a substantial impact on CNN performance. A batch size of thirty-two, according to Bengio [11], is a good starting point, and the number can be tuned further. The LeNet-5 CNN was proposed by Yann LeCun, Leon Bottou, Yosuha Bengio, and Patrick Haffner in 1998 [12]. It was one of the earliest CNNs containing 5880 connections and twelve trainable parameters. The seven layers include three convolutional layers, two pooling layers, and one fully connected layer. There are eighty-four units in the fully connected layer [13]. The AlexNet was founded in 2012 and had over sixty-two million parameters. This network achieved a top-5 error of 15.3% in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [14]. The ILSVRC was used to develop this model, which used a part of the ImageNet database. The data holds 1.4 million images over one thousand object classes. The ReLU (Rectified Linear Unit) function [14] introduces nonlinearity in the network. VGG16 and VGG19 networks were proposed by K. Simonyan et al. and A. Zisserman et al. [15]. The weight layers in VGG16 are thirteen convolutional layers, three fully connected layers, and a softmax layer. VGG19, on the other hand, has nineteen layers: sixteen convolutional layers, three fully connected layers, and a softmax layer. There are over 138 million parameters in VGG16 and 143 million in VGG19. There are sixty-four filters of size 3×3 in the first two layers. The input image dimension is changed to $224 \times 224 \times 64$ with a max-pooling layer of stride two. Residual Neural Network (ResNet) network introduces the term 'residual learning' [16]. The fifty-layer ResNet won the ILSVRC in 2015. There are four stages to the ResNet50 model, including a Convolution and Identity block. Each convolution block and each identity block have three convolution layers. In the ResNet architecture, the first convolution is done with 7×7 size filters. At the same time, a 3×3 size filter is used to perform max pooling. To improve the network's performance, the Deep Residual Network uses bottleneck residual block architecture. InceptionV3,

also known as GoogleNet, is an upgraded version of InceptionV1 [17]. The InceptionV3 model consists of a basic convolutional block, an improved Inception module, and a classifier. There are forty-eight layers and over twenty-four million parameters in it. Inception is divided into three modules: Inception-A, Inception-B, and Inception-C. The eleven convolutional kernels are commonly used to reduce the number of feature channels and speed up training [18]. At the end of the Inception modules, three fully connected layers allows to use the pre-trained model and fine-tune the parameters [19]. InceptionResNetV2, also known as the hybrid inception model, is designed on the Inception architecture while additionally including residual connections. Each branch has a concatenated set of filters with sizes ranging from 1 × 1 to 3 × 3, 5 × 5. This model includes Inception ResNet-A, Inception ResNet-B, and Inception ResNet-C [19]. Each Inception block is linked to a 1 × 1 filter expansion layer, which is used to scale up the dimensions of the filter bank before adding it. Xception includes a redesigned inception block that replaces several spatial dimensions such as 1 × 1, 5 × 5, and 3 × 3 with a single dimension followed by a 1x1 convolution to reduce computational complexity. Entry Flow, Middle Flow, and Exit Flow are the three major blocks. CNNs are used in thirty-six layers. Except for the first and last modules, these layers are divided into 14 modules surrounded by linear residual connections. The data first passes through the entering flow, then through the middle flow eight times, and finally through the exit flow. All Convolution and depth wise separable Convolution layers are batch normalized [20]. With 132 layers and over eight million parameters, DenseNet121 is a densely connected convolutional network [21]. In the DenseNet design, the first is the convolution block, which is the same as the identity block in ResNet. The dense block, in which the convolution blocks are concatenated and densely coupled, is the second part. The Dense block [21] is the principal component of the DenseNet architecture. The feature maps in the dense block are all the same size. The final layer is the transition layer, connecting two continuous dense blocks. The use of a transition block reduces the dimension of feature maps. With over three million parameters [23], MobileNetV2 is based on a linear bottleneck layer and an inverted residual structure [22]. There are two types of blocks in MobileNetV2. The first, with a stride of one, is the residual block. A residual block with a stride of two for shrinking and three layers in each block is another choice. A ReLU 1x1 convolutional layer [22] is the first layer. The depth wise convolution layer employs 3 × 3 depth-wise separable convolution, and the third layer is a linear 1 × 1 convolutional layer. NasNetMobile is a scalable CNN made up of twelve blocks and over five million parameters. Depending on the network's capacity requirements, each block consists of a few basic processes repeated numerous times. The block is the smallest unit in NASNet [23][24]. NasNetLarge is a CNN with over eighty-eight million parameters trained on over a million pictures from the ImageNet database. Normal cell and reduction cell are two of the themes [25]. Normal cells return a feature map in the same dimension as convolutional cells. In reduction cells, convolutional cells return a feature map with dimensions lowered by a factor of two [26]. The NAS-Neural Network Search algorithm [27] identifies the best neural network architecture. The RNN controller of the NAS algorithm samples the blocks and assembles them to construct an end-to-end architecture. The list of pre-trained CNN architectures available in the Keras library, with the year of publication, and number of parameters are summarized in Table 1.

Table 1. CNN architectures

S. No	Model	Year Published	Parameters
1	LeNet-5	1998	60,850
2	AlexNet	2012	62,378,344
3	VGG16	2014	138,357,544
4	VGG19	2014	143,667,240
5	ResNet50	2015	25,636,712
6	ResNet101	2015	44,707,176
7	ResNet152	2015	60,419,944
8	Xception	2016	22,910,480
9	ResNet50V2	2016	25,613,800
10	ResNet101V2	2016	44,675,560
11	ResNet152V2	2016	60,380,648
12	InceptionV3	2016	23,851,784
13	DenseNet121	2017	8,062,504
14	DenseNet169	2017	14,307,880
15	DenseNet201	2017	20,242,984
16	InceptionResNetV2	2017	55,873,736
17	MobileNet	2018	4,253,864
18	MobileNetV2	2018	3,538,984
19	NASNetMobile	2018	5,326,716
20	NASNetLarge	2018	88,949,818

3 Methodology

The images in this dataset were taken with a variety of mobile devices. The photos were taken in front of a white background with natural or artificial lighting. The original dataset is over 3.5 GB in size, and each image has been reduced to 512 × 384 pixels in size. The data distribution is shown in Table 2, while Figure 1 shows examples from each class in this dataset. Data Augmentation artificially increases the size of the training dataset, allowing the model to learn and generalize better on future unseen data [28]. To eliminate any bias in the model, Upsampling was used. It makes sure that both classes had the same number of images before training.

Table 2. Distribution of data in TrashNet

S. No	Class Name	Number of Images
1	Cardboard	403
2	Paper	594
3	Glass	501
4	Plastic	482
5	Metal	410
6	Trash	137

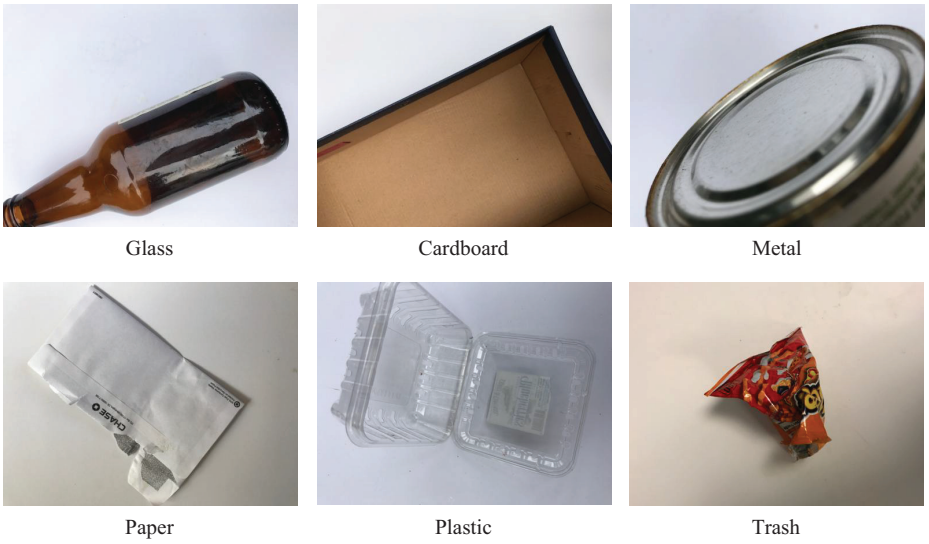

Fig. 1. Sample images from the TrashNet dataset

Random rotation, brightness adjustment, horizontal & vertical flips, channel shift, horizontal & vertical shifts, and channel shift were among the image manipulation techniques used. While increasing the dataset size, these picture changes were used to accommodate the various orientations of material. Biodegradable and non-biodegradable photos were further divided into two categories. Metal, glass, and plastic are non-biodegradable by nature, while paper and cardboard are. The final dataset contains 19500 photos, with 9750 belonging to class-one (biodegradable wastes) and the remaining 9750 to class-two (non-biodegradable wastes). The dataset was split into three parts: 80% for training, 10% for validation, and 10% for testing. Figure 2 shows the augmented images of one sample from the glass class.

Fig. 2. Augmented images of a sample from glass class

Table 3. Model's accuracy on train and validation set

Model	Validation Loss	Training Loss	Validation Accuracy	Training Accuracy
NASNetMobile	0.2023	0.0780	0.9236	0.9709
NASNetLarge	0.6230	0.1746	0.9128	0.9628
DenseNet201	0.2116	0.1351	0.9108	0.9467
DenseNet121	0.2594	0.1535	0.8964	0.9375
DenseNet169	0.2555	0.1714	0.8933	0.9287
ResNet50V2	0.3048	0.1349	0.8856	0.9475
ResNet101V2	0.2788	0.1424	0.8851	0.9437
MobileNet	0.2829	0.1744	0.8836	0.9308
ResNet152V2	0.3115	0.1526	0.8826	0.9382
Xception	0.2833	0.1310	0.8821	0.9497
MobileNetV2	0.3243	0.1756	0.8682	0.9285
InceptionResNetV2	0.3478	0.2311	0.8554	0.8998
InceptionV3	0.3454	0.2723	0.8549	0.8815
LeNet-5	0.4338	0.2961	0.8174	0.8712
VGG19	0.4282	0.2983	0.8144	0.8675
VGG16	0.4088	0.3055	0.8108	0.8628
AlexNet	0.5730	0.4129	0.7354	0.8061
ResNet50	0.6165	0.5931	0.6621	0.6905
ResNet101	0.6607	0.6139	0.5923	0.6616
ResNet152	0.6744	0.6195	0.5903	0.6613

Figure 3 show the variation of accuracy and loss in the training and validation data for over twenty epochs of the TrashNet. At the end of twenty epochs, the model had a training accuracy of 0.985 and a validation accuracy of 0.94. The loss on the training data is 0.03, while the loss on the validation data is 0.22. This shows overfitting, which can be solved by adding more images to the data set or adding more features that retain the image's original dimensions.

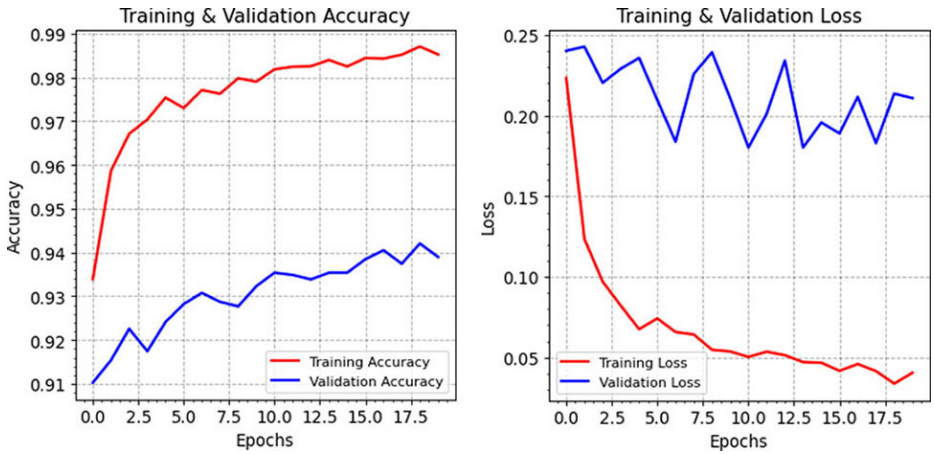

Fig. 3. Accuracy and loss of NASNetMobile model over twenty epochs

The twenty models were trained for up to five epochs, and the accuracies and losses are tabulated in Table 3. NASNetMobile was the best performing model with the highest validation accuracy and least validation loss. ResNet152 had the least accuracy among the twenty models.

Fig. 4. Accuracy of different CNN architectures on train and validation data

Figure 4 shows the accuracy of various models on the training and validation data. Residual Networks had the least validation accuracy of around 0.6 while all other models achieved a validation accuracy of more than 0.7.

4 Results

The trained models were tested on the test data, and the predictions were compared with the true labels. The accuracy metrics for each model are tabulated in Table 4. The NASNetMobile was the best performing model. The confusion matrix, shown in Figure 5, is a summary of prediction results on a classification problem. The model was tested using different activation functions and the accuracy metrics are tabulated in Table 5. Among the four activation functions, the model using Exponential Linear Unit (ELU) activation function had the highest accuracy. Table 6 describes the accuracy metrics of the model when different optimizers were used. The Adam optimizer, outperformed RMSprop and SGD. Three different learning rates were used while training the model and the accuracy metrics are shown in Table 7. From Table 8, it can be found that the model had the highest accuracy when the batch size was thirty-two.

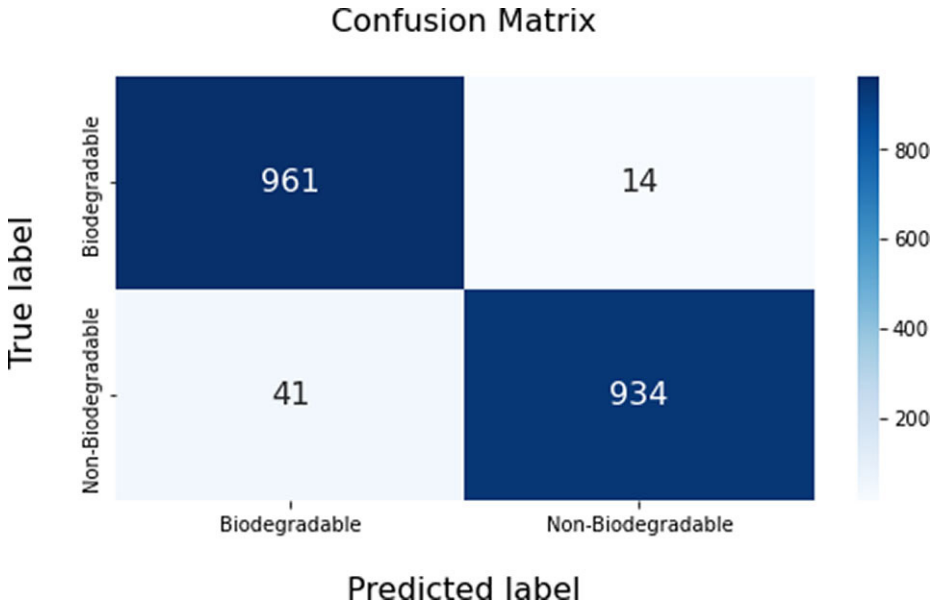

Fig. 5. Confusion matrix with true vs predicted label for NASNetMobile

- Test data accuracy = (961 + 934) / (961 + 14 + 41 + 934) = 0.97179
- Precision = (961) / (961 + 41) = 0.959082
- Recall = (961) / (961 + 14) = 0.985641
- F1-score = (2 * Precision * Recall) / (Precision + Recall) = 0.97218

Table 4. Model's accuracy on test set

Model	Test Accuracy	Precision	Recall	F1-score
NASNetMobile	0.97179	0.95908	0.98564	0.97218
NASNetLarge	0.96154	0.94379	0.98154	0.96229
DenseNet201	0.95333	0.94467	0.96308	0.95378
DenseNet169	0.95333	0.94378	0.96410	0.95383
Xception	0.93180	0.92440	0.94051	0.93238
DenseNet121	0.93128	0.91675	0.94872	0.93246
ResNet101V2	0.92923	0.90671	0.95692	0.93114
MobileNet	0.92718	0.91859	0.93744	0.92792
ResNet50V2	0.92462	0.88692	0.97333	0.92812
ResNet152V2	0.91333	0.88454	0.95077	0.91646
MobileNetV2	0.91282	0.89345	0.93744	0.91492
InceptionV3	0.89641	0.89723	0.89539	0.89630
InceptionResNetV2	0.89590	0.85874	0.94769	0.90102
VGG16	0.87949	0.89278	0.86256	0.87741
VGG19	0.87180	0.89189	0.84615	0.86842
LeNet-5	0.82923	0.79833	0.88103	0.8628
ResNet50	0.72256	0.72939	0.70769	0.8061
AlexNet	0.70821	0.64732	0.91487	0.6905
ResNet152	0.65231	0.65138	0.65538	0.6616
ResNet101	0.63692	0.58824	0.91282	0.6613

Table 5. Accuracy of different activation functions on validation, train, and test set

Activation Function	Validation Accuracy	Training Accuracy	Test Accuracy
ELU	0.9650	0.9607	0.9500
ReLU	0.9625	0.9564	0.9500
LeakyReLU	0.9625	0.9521	0.9475
SELU	0.9575	0.9435	0.9425

Table 6. Accuracy of different optimizers on validation, train, and test set

Optimizer	Validation Accuracy	Training Accuracy	Test Accuracy
Adam	0.9700	0.9585	0.9430
RMSprop	0.9650	0.9435	0.9389
SGD	0.9600	0.9557	0.9451

Table 7. Accuracy of different learning rates on validation, train, and test set

Learning Rate	Validation Accuracy	Training Accuracy	Test Accuracy
0.001	0.9625	0. 9578	0. 9475
0.0001	0.9450	0. 9650	0. 9375
0.01	0.9400	0. 9528	0. 9175

Table 8. Accuracy of different batch sizes on validation, train, and test set

Batch Size	Validation Accuracy	Training Accuracy	Test Accuracy
32	0.9725	0.9635	0.9375
64	0.9675	0.9635	0.9550
128	0.9625	0.9621	0.9550

5 Conclusion

Today's computer vision algorithms enable the classification of trash into several categories. A dataset of about 39000 images was developed as the first contribution of this study. NASNetMobile had the highest test accuracy of 97.18%, with the lowest validation loss of 0.20, among all the models when the various hyperparameters such as optimizer and loss function were kept constant. Other models, such as NASNetLarge, DenseNet201, and DenseNet169 had accuracy levels over 95%. In image classification, hyperparameters must be adjusted depending on the dataset used to train a model. For large learning rates, there is a strong link between learning rate and batch size. Based on the findings, it can be stated that the learning rate has a greater influence on accuracy than the batch size. Waste classification is a complicated task with a lot of elements to consider. Unlike other image classification problems using simplified images or quantitative data, waste classification needs to consider the object's surroundings. The model must also avoid the most common object recognition mistakes, such as misidentifying multiple independent objects as a single object or vice versa, finding the same thing in multiple classes, and not detecting hidden objects. However, the success rate in real systems can be lower because of the small amount of data and the white background of all the images.

6 References

[1] S. Kaza, L. C. Yao, P. Bhada-Tata, and F. V. Woerden, "What a waste 2.0," World Bank Publications, vol. 30317. Washington, DC, USA: The World Bank, 2018. [Online]. Available: https://ideas.repec.org/ b/wbk/wbpubs/30317.html

[2] T. Zhou, S. Ruan, and S. Canu, "A review: Deep learning for medical image segmentation using multimodality fusion," Array, vol. 3–4, no. August, p. 100004, 2019, https://doi.org/10.1016/j.array.2019.100004

[3] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," Artif. Intell. Rev., pp. 1–70, 2020, https://doi.org/10.1007/s10462-020-09825-6

[4] Y. Zhu et al., "Heterogeneous transfer learning for image classification," 2011.
[5] M. Yang and G. Thung, ''Classification of trash for recyclability status,'' Mach. Learn., Stanford, CA, USA, Project Rep. CS229, 2016.
[6] C. Bircanoglu, M. Atay, F. Beser, O. Genc, and M. A. Kizrak, ''RecycleNet: Intelligent waste sorting using deep neural networks,'' in Proc. INISTA, 2018, pp. 1–7, https://doi.org/10.1109/INISTA.2018.8466276
[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. 2012, 25, pp. 1097–1105.
[8] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv 2014, arXiv:1409.1556.
[9] T. Kennedy, OscarNet: Using transfer learning to classify disposable waste. In CS230 Report: Deep Learning; Stanford University: Stanford, CA, USA, 2018.
[10] P. Radiuk, Impact of training set batch size on the performance of convolutional neural networks for diverse datasets, Inf. Technol. Manag. Sci. 20 (2017), https://doi.org/10.1515/itms-2017-0003
[11]Y. Bengio, Practical recommendations for gradient-based training of deep architectures, 2012, Arxiv, https://doi.org/10.1007/978-3-642-35289-8_26
[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Ha, "LeNet," Proc. IEEE, no. November, pp. 1–46, 1998, https://doi.org/10.1109/5.726791
[13] A. El-Sawy, H. EL-Bakry, and M. Loey, "CNN for handwritten arabic digits recognition based on lenet-5BT - proceedings of the international conference on advanced intelligent systems and informatics 2016," 2017, pp. 566–575, https://doi.org/10.1007/978-3-319-48308-5_54
[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Handb. Approx. Algorithms Metaheuristics, pp. 1–1432, 2007, https://doi.org/10.1201/9781420010749
[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for largescale image recognition," 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc., pp. 1–14, 2015.
[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2016-Decem, pp. 770–778, 2016, https://doi.org/10.1109/CVPR.2016.90
[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2016-Decem, pp. 2818–2826, 2016, https://doi.org/10.1109/CVPR.2016.308
[18] X. Xia, C. Xu, and B. Nan, "Inception-v3 for flower classification," in 2017 2nd International Conference on Image, Vision, and Computing (ICIVC), 2017, pp. 783–787, https://doi.org/10.1109/ICIVC.2017.7984661
[19] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inceptionResNet and the impact of residual connections on learning," 31st AAAI Conf. Artif. Intell. AAAI 2017, pp. 4278–4284, 2017.
[20] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017, vol. 2017-Janua, pp. 1800–1807, 2017, https://doi.org/10.1109/CVPR.2017.195
[21] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017, vol. 2017-Janua, pp. 2261–2269, 2017, https://doi.org/10.1109/CVPR.2017.243
[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 4510–4520, 2018, https://doi.org/10.1109/CVPR.2018.00474

[23] F. Saxen, P. Werner, S. Handrich, E. Othman, L. Dinges, and A. Al- Hamadi, "Face attribute detection with mobilenetv2 and nasnet- mobile," Int. Symp. Image Signal Process. Anal. ISPA, vol. 2019-Septe, no. October, pp. 176– 180, 2019, https://doi.org/10.1109/ISPA.2019.8868585

[24] K. Radhika, K. Devika, T. Aswathi, P. Sreevidya, V. Sowmya, and K. P. Soman, Performance analysis of NASNet on unconstrained ear recognition, vol. SCI 871, no. January. Springer International Publishing, 2020, https://doi.org/10.1007/978-3-030-33820-6_3

[25] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 8697–8710, 2018, https://doi.org/10.1109/CVPR.2018.00907

[26] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Zoph learning transferable architectures CVPR 2018 paper," Proc. IEEE Conf. Comput. Vis. pattern Recognit., pp. 8697–8710, 2018.

[27] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc., pp. 1–16, 2019.

[28] M. A. Tanner and W. H. Wong, "The calculation of posterior distributions by data augmentation," J. Am. Stat. Assoc., vol. 82, no. 398, pp. 528–540, Jul. 1987, https://doi.org/10.2307/2289457

7 Author

Neelesh Karthikeyan is an Industrial Engineering undergraduate from the College of Engineering Guindy, Anna University, India (email: neelesh.ceg@gmail.com)

Article submitted 2022-01-26. Resubmitted 2022-02-27. Final acceptance 2022-02-27. Final version published as submitted by the authors.