World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(Research Article)

Survey of image classification models for transfer learning

Livinus Ifunanya Umeaduma *

Department of Computer Science, Western Illinois Unversity, Macomb IL, USA.

World Journal of Advanced Research and Reviews, 2024, 21(01), 373–383

Publication history: Received on 22 November 2023; revised on 01 January 2024; accepted on 03 January 2024

Article DOI: https://doi.org/10.30574/wjarr.2024.21.1.0006

Abstract

Training models for image classification is a very time-consuming task. It has always been a challenge for researchers and practitioners to train a model partly because of the large dataset required, which is complex and sometimes almost impossible to source. This has recently led to the use of pre-trained models for image classifications. Pre-trained models have gained popularity because they initialize the model with appropriate weight and significantly reduce the training time and dataset required. There are many image classification pre-trained models in use today, and this paper will focus on investigating the performance of the ten top models (ConvNext, DenseNet, EfficientNet, InceptionResNet, Inception, mobileNet, ResNet, VGG, Xception, NasNet) using caltech101 dataset containing 101 object classes and caltech256 dataset containing 256 object classes. The models are all trained on the ImageNet-1k dataset. Tensorflow and Keras were used as the frameworks for developing the experiments. The accuracy, precision, recall, and f1-score were used as metrics for model performance evaluation. The findings and analysis underscore the significance of training time, number of epochs, and choice of model in image classification.

Keywords: Pre-trained models; Caltech101; Caltech256; Convolutional Neural Networks; Keras; Transfer Learning

1. Introduction

The significant challenges facing artificial intelligence and deep learning are data collection, quality, and data size. Artificial intelligence has recently seen advances in algorithms and methods, which have done significantly well with good results. This is partly because of advances in computational techniques where GPUs are used for heavy data crunching and big data made available by the internet boom [27]. Nonetheless, the application of artificial intelligence in real life is faced with the challenges of the availability of training data in some domains. Sometimes, we have a small amount of data or almost nonexistent data.

The nonavailability of data in some domains makes applying artificial intelligence difficult or sometimes impossible in such domains. There can be a class imbalance problem where the training data is dominated by data belonging to one class. Most learning systems assume that the training data is balanced, and using imbalanced data in training a model can lead to bias in favor of the majority class from which the model was trained [11]. Two different approaches are used to address the problem of imbalanced data, which are data-level and algorithm-level methods [21]. Different techniques are employed at the data level to solve the imbalanced data problem. Some of the techniques are undersampling the majority, over-sampling the minority, and their combinations to balance the class distribution [11].

The lack of data in some domains has led to the adopting of transfer learning in many image classification applications. Transfer learning is a situation where a model that is trained for one task is re-used for another related task [17]. The model reuse results in less data required for training and significantly reduces the computation required to train the adopted model for the new task. In transfer learning, the source and target domains have to be related for transfer learning to achieve a good result [22]. Deep learning has done tremendously well in the fields of image recognition and

* Corresponding author: Livinus Umeaduma

speech recognition, natural language understanding, and protein-encoding [29]. Notably, deep learning using a convolutional neural network (CNN) has been shown to excel in the image classification [27]. Top CNN models using transfer learning have achieved significant results and, in some domains, exhibit intelligence that surpasses that of humans in image classification.

This study used ten popular pre-trained models for image classification based on a convolutional neural network (CNN). These models were first trained on ImageNet and used a resolution of 224 x 224. The ten pre-trained models investigated in this study are ConvNext, DenseNet [8], EfficientNet [18], InceptionResNet [6], Inception [6], mobileNet [3], ResNet50 [12], VGG [13]. These models were initially trained on ImageNet [10], an extensive database of over one million images with 1000 different classes.

- The goal of this paper is to identify
- The best-performing model for the classification.
- The easiest model to train is one where ease is measured by time and computation resources needed to traina model.
- The best model for inference

2. Related Works

2.1. Image Classification

Image classification categorizes images into predefined classes [27]. It is a critical problem in computer vision, mainly used as a benchmark to measure advancement in the computer vision [24]. Though image classification is a primary task for humans, it is challenging for a computer system. It forms the basis for tasks in computer vision like localization, detection, and segmentation. Before the advent of deep learning, the image classification problem was solved using two approaches. At first, handcrafted features are extracted from the image using feature descriptors, which serve as a classifier's input. The accuracy of the approach is highly dependent on the design of the feature extraction stage which is a problematic task [28].

Recently, deep learning models have proven to be very good in feature extraction and transformation. Most significantly, convolutional neural networks (CNN) have been shown to excel in image classification, recognition, and detection tasks [29]. High-performing pre-trained models for image classification are primarily trained on the ImageNet 1000-class dataset, mainly used for the yearly ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [20]. The ImageNet database contains images organized according to the WordNet [1] hierarchy in which hundreds and thousands of images depict each node of the hierarchy. There are 1000 object classes containing 1,281,167 training images and 50,000 and 100,000 validation and testing images, respectively.

2.2. CNN Architecture

A Convolutional Neural Network (also CNN or ConvNet) is a deep learning approach that takes an image, assigns weights to certain aspects of the objects in the image, and can differentiate them from the others [4]. A CNN is like the traditional artificial neural network (ANN) since it comprises neurons optimized through learning. They are feedforward networks; hence, information flows in one direction. A CNN consists of several building blocks, like an input layer, an output layer, and hidden layers. The hidden layers consist of convolutional, pooling, and fully connected layers. The convolution and pooling layers do the feature extraction, while the fully connected layer maps the output layer into the final out (classification). The convolution layer plays a vital role in CNN and mainly comprises mathematical operations like convolution. Forward propagation is the steps that transform input through these layers to output. The pixel values in digital images are stored in a two-dimensional grid as an array of numbers, and a small grid of parameters called kernel serves as a feature extractor when applied to the image position. This makes CNN very good for image processing because a feature may occur anywhere in the image. Extracted features may progressively become more complex as one layer feeds its output onto the next layer in CNN. The system undergoes learning (training) by repeatedly optimizing parameters through an optimization algorithm like backpropagation and gradient descent.

2.2.1. Convolutional Layer

A convolution layer is a significant component of the CNN architecture that performs feature extraction. This consists of linear convolution operation and non-linear activation function.

2.2.2. Convolution

Convolution is a mathematical operation used in feature extraction, which provides a way of multiplying two arrays of numbers (input tensor and kernel) with different sizes and the same dimensionality, producing a third array of numbers with the same dimensionality. The operation is performed by sliding the kernel over the image from the top left corner, moving the kernel through all the positions where the kernel fits the boundaries of the image. Multiple kernels serve as feature extractors. The input tensor is usually padded with zeros on each side of the input tensor to retain the height and width of the output feature map. The distance between two kernels is a stride, and one is usually shown, but a large number can be used for the down-sampling of the feature maps.

The main feature of convolution operation is weight sharing. The kernels are shared across the image positions through the convolution operation. The training process in the convolution layer is to identify the kernel that works best for a given task based on the training dataset. The number of kernels, padding, stride, and size of the kernels are the hyperparameters that need to be set before the training.

2.2.3. Activation function

The output of the convolution operation is passed through the activation function. There are several options for the activation function to use such as hyperbolic tangent function, sigmoid activation function, and ReLU. The ReLU is a non-linear activation function. The ReLU (rectified linear unit) is more efficient than other functions because all the neurons are not activated simultaneously [26].

2.2.4. Pooling Layer

Pooling in image processing is like reducing resolution. Pooling is a down-sampling operation to reduce the complexity of other following layers. There is no learnable parameter in the pooling layers, and it does not reduce the number of filters. The primary pooling method is max-pooling which involves partitioning the image into sub-region rectangles and returning the maximum value in that sub-region. The typical max-pooling size in use is 2x2 with a stride of 2. Stride 1 can be used in max-pooling to avoid the down-sampling [2] [23].

2.2.5. Fully Connected Layer

This layer is similar to the one in the traditional artificial neural network and produces a class score from the activation to be used in the classification [19]. Every input to this layer is connected to the output by a learnable weight mapped by a subset of fully connected layers to the final output. This output is usually probabilities for each class in the classification problem. The number of output nodes of the fully connected layers is usually the same as the number of classes.

2.3. Transfer Learning

Transfer learning has been researched since 1995 under different names like knowledge transfer, learning to learn, lifelong learning, inductive transfer, multitask learning, knowledge consolidation, context-sensitive learning, knowledge-based inductive bias, meta-learning, and incremental learning. Transfer learning extracts knowledge from one or more source tasks and applies it to a target task [22]. Transfer learning can be categorized into three types: inductive transfer learning, transductive transfer learning, and unsupervised transfer learning. In inductive transfer learning, the source task differs from the target task, and the similarity of the source and target domains does not matter. The source and target tasks are the same for transductive transfer learning, but the source and target domains differ. In unsupervised transfer learning, the target task is different but related to the source task [25] [22]. Transfer learning is usually used when the training data available is far less than the dataset used in training the model [7].

2.4. Xception

Xception is an improvement to the Inception architecture, which stands for Extreme Inception. It has 36 convolutional layers serving as the feature extraction base of the network. The image resolution is 299 x 299 with about 30 million parameters and a depth of 81 layers. The logistic regression layer follows the convolutional base, and a fully connected layer can be inserted before the logistic regression layer. The 36 convolutional layers are structured into 14 modules with linear residual connections. The novelty of the Xception approach is the introduction of depthwise separable convolution layers to the inception model [5].

2.5. NasNet

NasNet is a convolutional neural network architecture trained on ImageNet-1k at 331 x 331 image resolution by Google researchers. It has about 88.8 million parameters with layers ranging from 389 to 533. It uses a reinforcement approach by searching for the best combination of parameters of the given search space of several filter sizes, layers, and strides. NasNet used a new regularization technique, ScheduledDropPath, which improves the generalization of the NasNet model [30].

2.6. ConvNext

ConvNext is a convolutional neural network architecture inspired by the design of Vision Transformers. The architecture is a modernization of ResNet with inspiration from Swin Transformer. ConvNext is trained on ImageNet-1k dataset at 224 x 224 image resolution [16].

2.7. DenseNet

DenseNet is a convolutional neural network architecture developed by researchers at Facebook AI Research and trained on the ImageNet-1k dataset at 224 x 224 image resolution. It uses a dense connection, which connects each layer to every other layer in a feedforward form. It contains many dense blocks, which contain many convolutional layers and a transition layer that reduces the spatial dimension of the output. There are different versions, and the parameters vary between 8 and 20 million, with layers ranging from 240 to 400 [8].

2.8. EfficientNet

EfficientNet is a convolutional neural network architecture trained on ImageNet-1k dataset at 224 x 224 image resolution. Using a compound coefficient, it uniformly scales all depth, width, and resolution dimensions. The compounding scaling method comes from the intuition that if the input image is bigger, the network needs more layers to increase the receptive field and more channels to capture more fine-grained patterns on the bigger image. The base EfficientNet-b0 network is based on inverted bottleneck residual blocks of MobileNetv2 and squeeze and excitation blocks [18].

2.9. InceptionResNet

Inception-ResNet-v2 is a convolutional neural network architecture trained on the ImageNet-1k dataset at 299 x 299 image resolution. This architecture developed by researchers at Google has about 55.8 million parameters with 189 layers. It builds on the Inception family architecture but also uses residual connections, which replace the filter concatenation stage of the Inception architecture [6].

2.10. Inception

InceptionV4 is a convolutional neural network architecture trained on the ImageNet-1k dataset at 299 x 299 image resolution. It builds on the previous iteration of the inception family, simplifying the architecture and using more inception modules than InceptionV3. The main goal of the network is to increase its size to improve performance, but this risks overfitting, which was its major drawback and led to an increase in computational resources. Parallel layers or multiple filters of different sizes are used on the same level to avoid overfitting to make the architecture broader rather than deeper. It has about 42 million parameters [6].

2.11. MobileNet

MobileNetV2 is a convolutional neural network architecture trained on the ImageNet-1k dataset at 224 x 224 image resolution. MobileNet is based on an inverted residual structure where the residual structures are between the bottleneck layers. It uses depthwise separable convolutions to construct deep convolutional neural networks providing efficient mobile application models. It has about 3.5 million parameters and between 55 and 105 layers [3].

2.12. ResNet50

ResNet is a convolutional neural network architecture trained on ImageNet-1k at 224 x 224 image resolution. ResNet democratized residual learning and skip connections concept to solve the vanishing gradient problem. It uses a 34layer network architecture inspired by the VGG-19 model to which the shortcut connections are added. These shortcut connections convert the architecture into the residual network. Depending on the version, this architecture has between 25 and 65 million parameters, with 107 to 307 layers deep [5].

2.13. VGG

VGG-16 and VGG-19 are two similar network architectures by K. Simonyan and A. Zisserman, respectively, from the Visual Geometry Group, which took the first and second position in the image localization and classification tracks at ILSVRC. The input is a 224 × 224 RGB image, and the only preprocessing involves subtracting the mean RGB value computed on the training set from each pixel. The network consists of convolutional layers with small 3 × 3 filters, and in some configurations, 1 × 1 convolution filters are used. The convolution stride is fixed at 1 pixel, and spatial padding is applied to preserve spatial resolution. Max-pooling layers follow some convolutional layers, using a 2 × 2 pixel window with a stride of 2. The convolutional layers are followed by three fully connected (FC) layers, with the final layer performing 1000-way ILSVRC classification. All hidden layers use rectification (ReLU) nonlinearity. Notably, Local Response Normalisation (LRN) is generally not used, as it does not improve performance on the ILSVRC dataset but increases memory consumption and computation time.

3. Methods

3.1. Experimental Environment

The experiment source code was written in Python, employing the Jupyter Notebook hosted on Google Colab. The computational environment was configured with robust hardware resources, featuring an NVIDIA Tesla V100 GPU with 16GB of dedicated memory and a substantial 52GB of RAM. The development of the experiment was facilitated by integrating the Tensorflow and Keras libraries, which provided essential tools and frameworks for machine learning tasks. This well-orchestrated combination of programming language, computational infrastructure, and powerful libraries ensured the seamless execution and efficient implementation of the experimental design.

3.2. Experimental data

The study was carried out using two different datasets from Caltech. These datasets are the calTech101 [15] dataset and the calTech256 [9] dataset. The calTech101 dataset contains pictures of objects belonging to 101 categories. Each image in caltech101 is approximately 300 x 200 pixels, and each category has 40 to 800 images. The calTech256 dataset contains a set of 256 object categories containing 30,607 images, with the minimum number in any category being 80. The images in the datasets are partitioned into training, validation, and test datasets.

3.3. Experimental models

The models were trained initially on the ImageNet-1k [20] dataset. The specific models used in the experiment are ConvNeXtSmall, DenseNet121, EfficientNetV2B0, InceptionResNetV2, InceptionV3, MobileNetV2, Resnet50, Xception, NASNetMobile, and VGG19. All the pre-trained models expect input images normalized in the same way, i.e., mini-batches of RGB images with shape 3 × H × W, where H and W are expected to be: 299 pixels for InceptionResNetV2, InceptionV3 and 224 pixels for all the other models considered. All the architecture under consideration was trained for ten epochs.

3.4. Experimental Procedure

The Caltech101 and Caltech256 datasets have been partitioned into training, validation, and testing subsets. Given the diverse resolutions of images within the dataset, a uniform size is enforced through resizing. Subsequently, image augmentation is performed, and the dataset is cached in memory to enhance computational efficiency. A standardized batch size of 32 is maintained throughout the training process. The data pipeline performance is achieved by incorporating prefetching and auto-tuning mechanisms.

Image preprocessing is applied to the dataset utilizing preprocessing layers. These layers employ random rotation (with a factor of 0.15) and random translation (with a factor of 0.1) along both height and width directions. Additionally, random flip and random contrast adjustments (with a factor of 0.1) are implemented. Notably, the dataset labels undergo one-hot encoding.

The transfer learning paradigm is instituted by initializing the models with pre-trained ImageNet weights and fine-tuning using our specific dataset. Initially, all layers in the models, excluding the top layer (fully connected layer or classifier), are frozen. The classifier is then replaced to align with the number of classes in our dataset. During this phase, a relatively higher learning rate (1e-4) is selected, and training is executed for 15 epochs utilizing the Adam [14] optimizer—a stochastic gradient descent method incorporating adaptive estimation of first-order and second-order moments.

The subsequent step involves un-freezing the top 20 layers of the model while maintaining the batch normalization layer in a frozen state. Fine-tuning is conducted with a reduced learning rate (1e-5) over 15 epochs, employing Adam's optimizer. Evaluation metrics such as accuracy, precision, recall, and f1-score are incorporated. Ultimately, the trained model undergoes testing using the designated testing dataset.

4. Results

The evaluation of the performance of the models was done using metrics like accuracy, precision, recall, and F1-score. The validation accuracy, training accuracy, and their changes in the different stages of training are highlighted in the table and charts.

4.1. Model performance on Caltech101 Datasets

Table 1 and Figure 1 display validation metrics for different models on the Caltech101 dataset. Notably, Resnet and InceptionResnet achieve outstanding performance, surpassing 99% accuracy and excelling in precision, recall, and F1Score. VGG also demonstrates strong results, with accuracy close to 99% and balanced precision, recall, and F1Score values. EfficientNet exhibits robust performance, particularly in precision (96.89%), while NasNet and MobileNet achieve high accuracy with a relatively balanced trade-off between precision and recall. Densenet and ConvneXt deliver competitive results across all metrics, showcasing their effectiveness on the Caltech101 dataset.

Table 2 summarizes performance metrics for various models on the Caltech101 dataset. EfficientNet achieves the highest accuracy at 96.89%, while Inception and ResNet excel in precision, recall, and F1Score. Inception achieves near-perfect precision, recall, and F1Score. VGG exhibits balanced performance across all metrics, and ConvneXt follows closely behind with solid accuracy and precision.

The chart in Figure 2 and Figure 3 show the model training accuracy and validation accuracy, respectively. It can be observed that the accuracy increased sharply during the feature extraction stage in the first ten epochs and slowed down afterward.

Table 1 Caltech101 Model Validation Metrics

Model	Accuracy	Precision	Recall	F1Score
VGG	0.9896	0.9895	0.9863	0.9872
EfficientNet	0.9372	0.9689	0.9020	0.9368
Inception	0.9938	0.9941	0.9932	0.9935
Resnet	0.9948	0.9945	0.9941	0.9944
Xception	0.9792	0.9853	0.9824	0.9840
NasNet	0.9641	0.9553	0.9193	0.9367
Densenet	0.9733	0.9823	0.9769	0.9795
MobileNet	0.9629	0.9806	0.9557	0.9665
ConvneXt	0.9782	0.9884	0.9697	0.9820
InceptionResnet	0.9935	0.9948	0.9941	0.9942

Table 2 Caltech101 Model Evaluation Metrics

Model	Accuracy	Precision	Recall	F1Score
VGG	0.8959	0.9092	0.8924	0.8810
EfficientNet	0.9689	0.9689	0.9020	0.9368
Inception	0.9305	0.9941	0.9932	0.9935

Resnet	0.9171	0.9945	0.9941	0.9944
Xception	0.9192	0.9853	0.9824	0.9840
NasNet	0.8910	0.9553	0.9193	0.9367
Densenet	0.8989	0.9823	0.9769	0.9795
MobileNet	0.9142	0.9451	0.8936	0.9046
ConvneXt	0.9373	0.9648	0.9130	0.9354
InceptionResnet	0.9303	0.9393	0.9260	0.9299

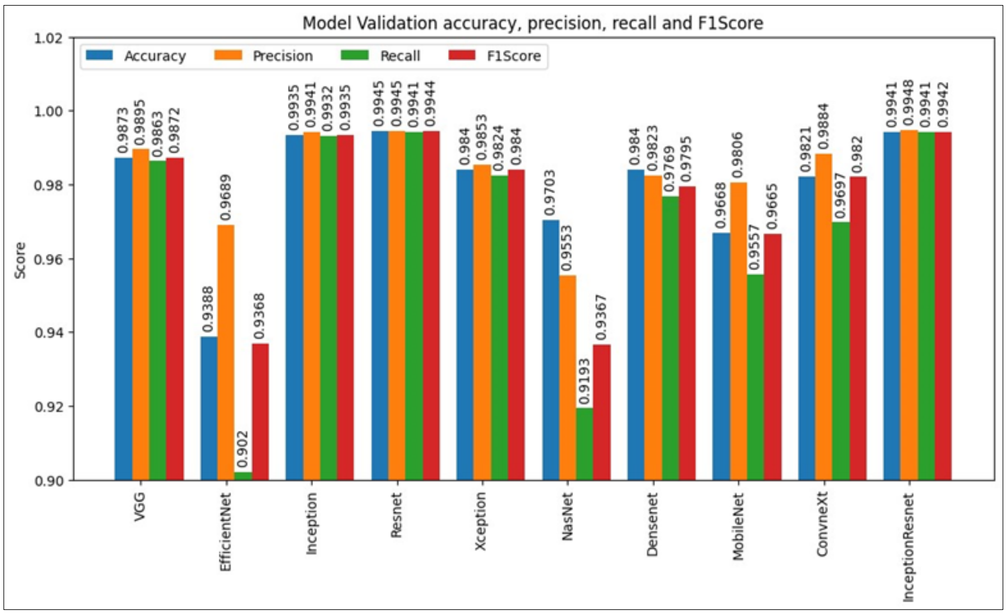

Figure 1 Model performance metrics showing the validation accuracy, validation precision, validation recall, and validation F1-score

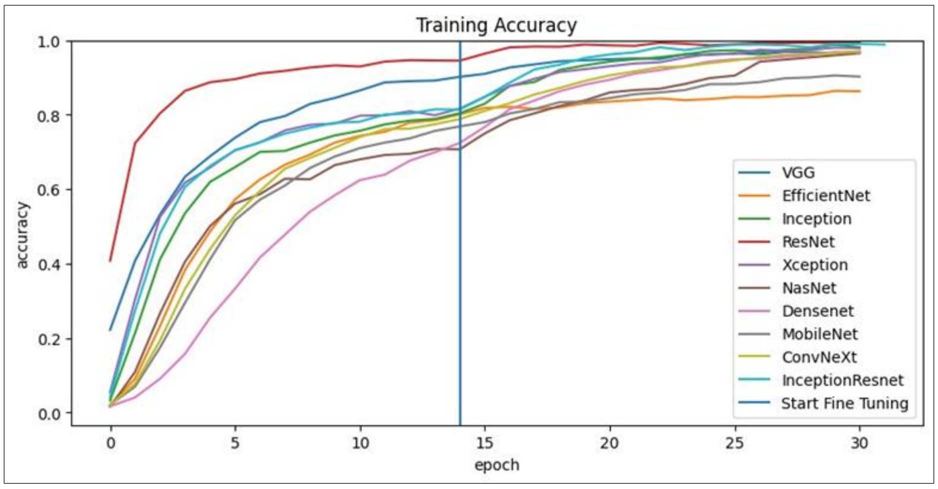

Figure 2 Caltech101 dataset model training accuracy

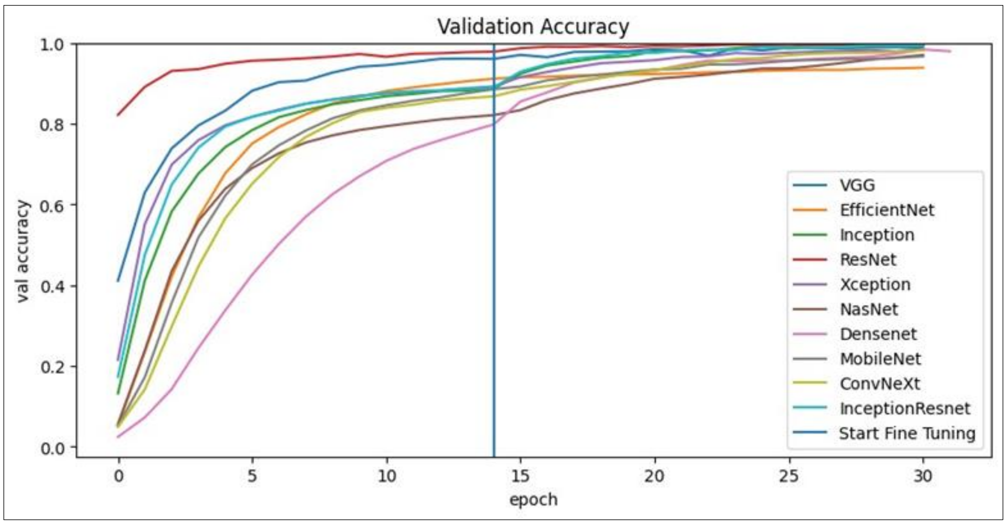

Figure 3 Caltech101 dataset model validation Image

4.2. Model Performance on Caltech256 Datasets

The model validation metrics in Table 3 shows that ConvneXt is the top-performing model, showcasing the highest accuracy at 89.02% and precision at 92.97%. EfficientNet excels in recall, achieving the highest value at 80.53%. The F1 score, balancing precision and recall, is maximized by ConvneXt at 87.63%. ConvneXt is the best model, offering a comprehensive performance across accuracy, precision, recall, and F1 score metrics. However, the choice of the "best" model should align with specific priorities and goals within the application context.

Table 4 and figure 4 present evaluation or testing metrics for various models on the Caltech256 dataset. ConvneXt is the top performer, with the highest accuracy at 89.52%, precision at 92.83%, recall at 87.03%, and F1 score at 88.64%. EfficientNet also demonstrates strong performance, particularly in precision (91.47%), and achieves an overall accuracy of 85.30%. InceptionResnet and Xception showcase competitive results across multiple metrics, with accuracy scores of 83.75% and 83.16%, respectively. The metrics, including precision, recall, and F1 score, comprehensively assess each model's performance on image classification tasks.

The chart in Figure 5 and Figure 6 show the model training accuracy and validation accuracy, respectively. It can be observed that the accuracy increased sharply during the feature extraction stage in the first half of the epochs. The accuracy after the 10th epoch slowly increases after unfreezing some top layers.

Table 3 Caltech256 Model Validation Metrics

Model	Accuracy	Precision	Recall	F1Score
VGG	0.7104	0.7522	0.6987	0.6846
EfficientNet	0.8425	0.9062	0.8053	0.8244
Inception	0.8008	0.8282	0.7880	0.7768
Resnet	0.7545	0.7816	0.7429	0.7232
Xception	0.8297	0.8559	0.8193	0.8109
NasNet	0.8057	0.8620	0.7818	0.7814
Densenet	0.8103	0.8488	0.7967	0.7928
MobileNet	0.7961	0.8678	0.7630	0.7756
ConvneXt	0.8902	0.9297	0.8637	0.8763
InceptionResnet	0.8412	0.8646	0.8335	0.8244

Table 4 Caltech256 Model Evaluation Metrics

Model	**Accuracy**	**Precision**	**Recall**	**F1Score**
VGG	0.7079	0.7459	0.6945	0.6877
EfficientNet	0.8530	0.9147	0.8111	0.8413
Inception	0.8025	0.8321	0.7923	0.7873
Resnet	0.7531	0.7846	0.7446	0.7309
Xception	0.8316	0.8555	0.8203	0.8176
NasNet	0.8061	0.8596	0.7796	0.7909
Densenet	0.8127	0.8513	0.7976	0.7966
MobileNet	0.7949	0.8691	0.7609	0.7799
ConvneXt	0.8952	0.9283	0.8703	0.8864
InceptionResnet	0.8375	0.8572	0.8299	0.8230

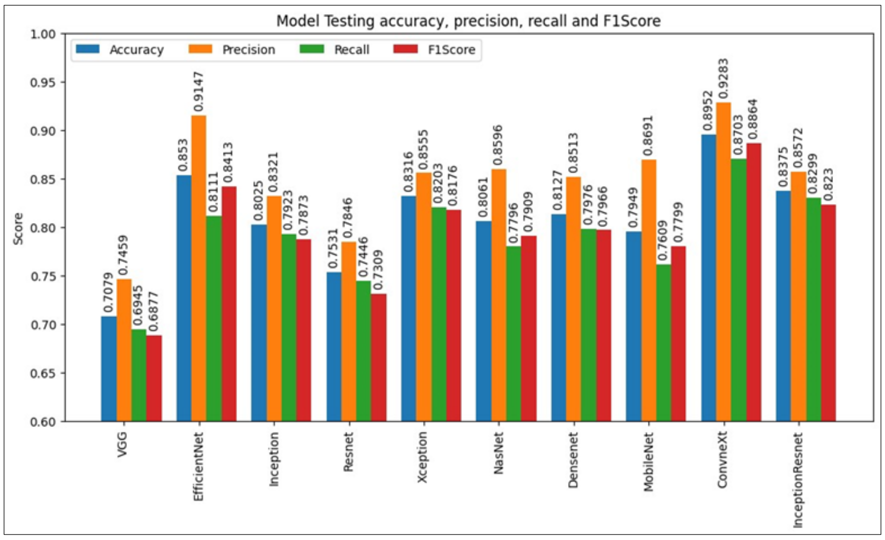

Figure 4 Model performance metrics showing the validation accuracy, validation precision, validation recall, and validation F1-score

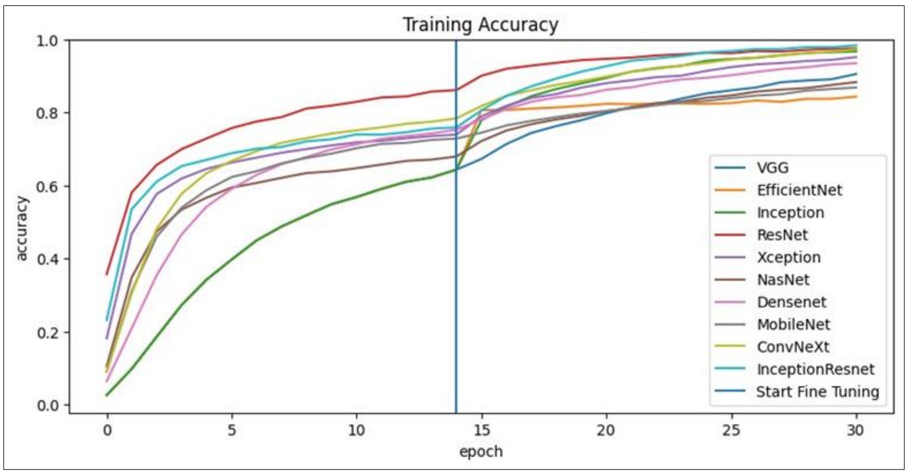

Figure 5 Caltech256 dataset model training accuracy

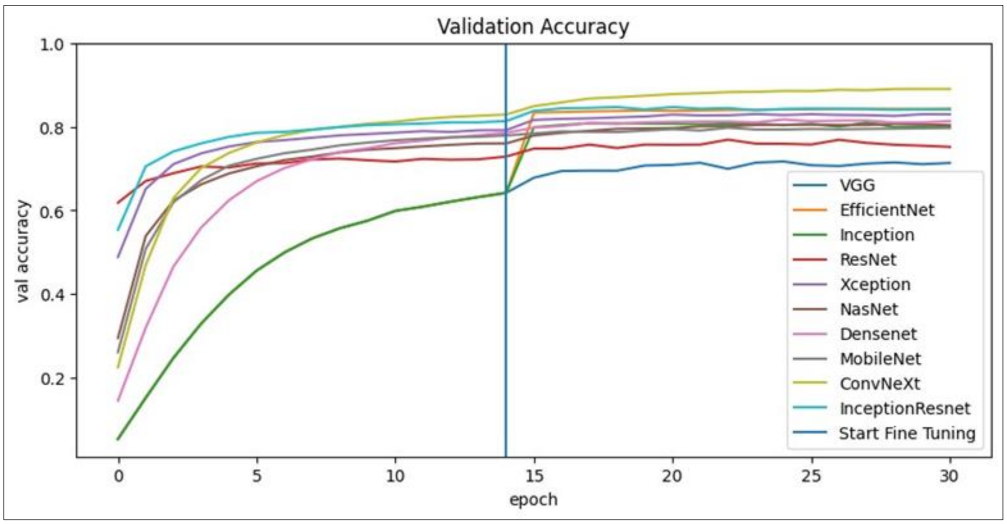

Figure 6 Caltech256 dataset model validation accuracy

5. Conclusion

Exploring this study's pre-trained image classification models has underscored the opportunities and challenges of training such models. The experiment results show the pragmatic advantage of using pre-trained models, especially their ability to reduce training time and alleviate the complexities associated with obtaining large datasets. The findings also highlight the nuanced impact of training time, the number of epochs, and the choice of model on the overall performance of image classification tasks. Researchers and practitioners can draw from these insights to make informed decisions when selecting pre-trained models based on their specific requirements and constraints.

Compliance with ethical standards

Acknowledgments

I want to thank all the contributors who have participated in this research to make it a success. I acknowledged the contributions made by the reviewers before the manuscript was published.

Disclosure of conflict of interest

All authors of this manuscript agreed and contributed significantly to the success of this research without a conflict of interest.

References

[1] George A and Miller. "WordNet: a lexical database for English". In: Commun. ACM 38 (Nov. 1995), pp. 39–41.DOI: 10.1145/219717.219748.

[2] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network". In: 2017 International Conference on Engineering and Technology (ICET). 2017, pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[3] Howard Andrew et al. In: Searching for MobileNetV3 (2019). DOI: 10.48550/arXiv.1905.02244.

[4] Singh Ankita and Singh Pawan. "Image Classification: A Survey". In: Journal of Informatics Electrical and Electronics Engineering 1 (2 Nov. 2020), pp. 1–9. DOI: 10.54060/JIEEE/001.02.002.

[5] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. 2017. arXiv: 1610.02357 [cs.CV].

[6] Szegedy Christian et al. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: (2016). DOI: 10.48550/arXiv.1602.07261.

[7] Larsen-Freeman Diane. “Transfer of Learning Transformed”. In: Language Learning 63 (2013), pp. 107–129.DOI: 10.1111/j.1467-9922.2012.00740.x.

[8] Huang Gao et al. “Densely Connected Convolutional Networks”. In: (2018). DOI: 10.48550/arXiv.1608. 06993.

[9] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech 256. Apr. 2022. DOI: 10.22002/D1.20087.

[10] Deng Jia et al. “ImageNet: A large-scale hierarchical image database”. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition (June 2009), pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[11] Gu Jie, Zhou Yuanbing, and Zuo Xianqiang. “Making Class Bias Useful: A Strategy of Learning from Imbalanced Data”. In: Intelligent Data Engineering and Automated Learning -IDEAL 2007 (2007), pp. 287–295. DOI: 10.1007/978-3-540-77226-2_30.

[12] He Kaiming et al. “Deep Residual Learning for Image Recognition”. In: (2015). DOI: 10.48550/arXiv. 1512.03385.

[13] Simonyan Karen and Zisserman Andrew. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: (Nov. 2015).

[14] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. 2017. arXiv: 1412.6980 [cs.LG].

[15] Fei-Fei Li et al. Caltech 101. Apr. 2022. DOI: 10.22002/D1.20086.

[16] Zhuang Liu et al. A ConvNet for the 2020s. 2022. arXiv: 2201.03545[cs.CV].

[17] Shaha Manali and Pawar Meenakshi. “Transfer Learning for Image Classification”. In: 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA) (Mar. 2018), pp. 656– 660.

[18] Tan Mingxing, Quoc V, and Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: (2020). DOI: 10.48550/arXiv.1905.11946.

[19] Keiron O’, Shea Ryan, and Nash. “An Introduction to Convolutional Neural Networks”. In: (Nov. 2015).

[20] Russakovsky Olga et al. “ImageNet Large Scale Visual Recognition Challenge”. In: Int J Comput Vis 115 (Dec. 2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[21] Kumar Pradeep et al. “Classification of Imbalanced Data:Review of Methods and Applications”. In: IOP Conf. Ser.: Mater. Sci. Eng 1099 (1 Mar. 2021), p. 12077. DOI: 10.1088/1757-899X/1099/1/012077.

[22] Sinno Jialin Pan Qiang and Yang. “A Survey on Transfer Learning”. In: IEEE Transactions on Knowledge and Data Engineering 22 (Oct. 2010), pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.

[23] Yamashita Rikiya et al. In: Convolutional neural networks: an overview and application in radiology. Insights Imaging 9 (Aug. 2018), pp. 611–629. DOI: 10.1007/s13244-018-0639-9.

[24] Wightman Ross, Touvron Hugo, and Jégou Hervé. “ResNet strikes back: An improved training procedure in timm”. In: (Nov. 2021).

[25] Niu Shuteng et al. “A Decade Survey of Transfer Learning”. In: IEEE Transactions on Artificial Intelligence 1 (2 2010), pp. 151–166. DOI: 10.1109/TAI.2021.3054609.

[26] Sharma Siddharth, Sharma Simone, and Athaiya Anidhya. “ACTIVATION FUNCTIONS IN NEURAL NETWORKS”. In: IJEAST 04 (May 2020), pp. 310–316. DOI: 10.33564/IJEAST.2020.v04i12.054.

[27] Rawat Waseem and Wang Zenghui. “Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review”. In: Neural Computation 29 (June 2017), pp. 1–98. DOI: 10.1162/NECO_a_00990.

[28] Lecun Y et al. “Gradient-based learning applied to document recognition”. In: Proceedings of the IEEE 86 (11 Nov. 1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[29] Lecun Yann, Bengio Yoshua, and Hinton Geoffrey. “Deep learning”. In: Nature 521 (May 2015), pp. 436–444.DOI: 10.1038/nature14539.

[30] Barret Zoph et al. Learning Transferable Architectures for Scalable Image Recognition. 2018. arXiv: 1707. 07012[cs.CV].