Turkish Journal of Science & Technology
18(2), 461-475, 2023

Research Paper
https://doi.org/10.55525/tjst.1317713

A Hybrid Classification Approach for Fasteners Based on Transfer Learning with Fine-Tuning and Deep Features

Canan TASTIMUR[1*], Erhan AKIN[2]
[1] Department of Computer Engineering, Faculty of Engineering-Architecture, Erzincan Binali Yildirim University, Erzincan, Turkey
[2] Department of Computer Engineering, Faculty of Engineering, Firat University, Elazig, Turkey
[*1] ctastimur@erzincan.edu.tr, [2] eakin@firat.edu.tr

(Geliş/Received: 20/06/2023; Kabul/Accepted: 28/08/2023)

Abstract: Deep learning, which has seen frequent use in recent studies, has helped solve the problem of classifying objects of many different types and properties. Most studies both create and train a convolutional neural network (CNN) from scratch. The time spent training the network is thus wasted. Transfer learning (TL) is used both to prevent the loss of time due to training the dataset and to more effectively classify small datasets. This study performs classification using a dataset containing eighteen types of fastener. Our study contains three different TL scenarios. Two of them use TL with fine-tuning (FT), while the third does so with feature extraction (FE). The study compares the classification performance of eighteen different pre-trained network models (i.e., one or more versions of EfficientNet, DenseNet, InceptionResNetV2, InceptionV3, MobileNet, ResNet50, Xception, and VGGNet) in detail. When compared to other research in the literature, our first and second scenarios provide excellent implementations of TL-FT, while our third scenario, TL-FE, is hybrid and produces better results than the other two. Furthermore, our findings are superior to those of most previous studies. The models with the best results are DenseNet169 with an accuracy of 0.97 in the TL-FT1 scenario, EfficientNetB0 with 0.96 in TL-FT2, and DenseNet169 with 0.995 in TL-FE.

Key words: Classification, fastener, feature extraction, fine tuning, transfer learning.

Derin Öznitelik ve İnce-Ayar ile Aktarım Öğrenme Tabanlı Bağlantı Elemanlarının Hibrit Sınıflandırma Yaklaşımı

Öz: Son yıllarda yapılan çalışmalarda sıkça kullanılmaya başlanan derin öğrenme, birçok farklı tür ve özellikteki nesnelerin sınıflandırılması sorununun çözülmesine yardımcı olmuştur. Çoğu çalışma, sıfırdan bir evrişimsel sinir ağı (CNN) oluşturur ve eğitir. Ağı eğitmek için harcanan zaman böylece boşa harcanır. Transfer öğrenme (TL) hem veri setinin eğitilmesinden kaynaklanan zaman kaybını önlemek hem de küçük veri setlerini daha etkin bir şekilde sınıflandırmak için kullanılmaktadır. Bu çalışma, on sekiz tip bağlantı elemanı içeren bir veri seti kullanarak sınıflandırma yapmaktadır. Çalışmamız üç farklı TL senaryosu içermektedir. Bunlardan ikisi ince ayar (FT) ile TL kullanırken, üçüncüsü özellik çıkarma (FE) ile yapmaktadır. Çalışma, on sekiz farklı önceden eğitilmiş ağ modelinin (yani EfficientNet, DenseNet, InceptionResNetV2, InceptionV3, MobileNet, ResNet50, Xception ve VGGNet) sınıflandırma performansını ayrıntılı olarak karşılaştırmaktadır. Literatürdeki diğer araştırmalarla karşılaştırıldığında, birinci ve ikinci senaryolarımız TL-FT' nin iyi sonuçlarla uygulamalarını sağlarken, üçüncü senaryomuz TL-FE hibrit bir yöntem olup diğer iki senaryodan daha iyi sonuçlar üretmiştir. Ayrıca, bulgularımız literatürdeki çalışmaların çoğundan daha üstün olduğu fark edilmiştir. En iyi sonuçlara sahip modeller TL-FT1 senaryosunda 0,97 doğrulukla DenseNet169, TL-FT2'de 0,96 ile EfficientNetB0 ve TL-FE'de 0,995 ile DenseNet169'dur.

Anahtar kelimeler: Sınıflandırma, bağlantı elemanı, özellik çıkarma, ince ayar, transfer öğrenme.

1. Introduction

Fasteners are critical to the proper operation of industrial machinery. Quality control and fault diagnosis applications must ensure that fasteners on machines are complete and flawless. The detection of problems with fasteners can benefit from the use of computer vision technologies. Potential major problems in industrial machines—fastener failure, breakage, wear and tear, and so on—can be avoided thanks to advanced fault diagnosis using computer vision technologies. Using computer vision techniques to determine the condition of industrial machinery will help ensure the safety of industrial machines by avoiding the unnecessary use of human resources and costly expenses due to failures.

Recently, the deep learning (DL) approach has commonly been used in fault diagnosis, detection, and classification. Fasteners are highly similar to each other, which complicates the identification, detection, and classification of the carrying element. Many studies have used DL for object detection and classification, and the transfer learning (TL) method can help provide more effective results by improving learning capacity. TL is a

[*] Corresponding author: ctastimur@erzincan.edu.tr. ORCID Number of authors: [1] 0000-0002-3714-6826, [2] O 0000-0001-6476-9255

technique in which models are trained on large datasets, allowing the trained network to be used ready-made from other datasets. In classification studies, a pre-trained network saves time and provides high performance in classifying small datasets. Thanks to the information obtained from a model previously trained on a large-scale TL dataset, classification uses a new, different dataset with the same or a different model. Thus, the information in the pre-trained model is transferred and the performance of the network improves. Since our dataset is small, TL provided better performance than in many other studies in the literature.

Using a fine-tuning-based and feature-inference-based transfer learning method, it is intended to identify data sets with high similarity. This study classified 18 fastener types using TL-based feature extraction and fine-tuning techniques. The proposed method includes three different classification scenarios. The first scenario involves fine-tuning by updating only the number of classes in the classifier layer of the pre-trained model. In the second, after adding two dense layers to the end of the pre-trained model, fine-tuning has been applied with the classifier layer updated. In the last scenario, features were extracted, with the pre-trained network using the feature extractor capability of the pre-trained model. So, the new CNN model takes inputs as local features, leading to more accurate results.

The main contributions of this study are listed below:

- It proposes three different methods to classify TL-based fasteners. Two of them use the fine-tuning (FT) approach, while the third adopts the feature extraction (FE) approach. While studies in the literature classify fasteners using only one of the FT methods, our study compares three different scenarios in detail. From this point of view, it is a very comprehensive and useful TL resource, as compared to the studies in the literature.
- Classification using FE, the third of the proposed classification scenarios, is a hybrid method. Examining the studies in the literature shows this to be an important study. In the third scenario, which combines CNN architecture and a pre-trained TL network, the features extracted by the TL network, rather the raw image, are given as input to the CNN network, thus improving the success of the network.
- Examining the studies in the literature that use FT shows that FT can be performed in two ways. Our study compares performance outputs by applying both FT approaches separately. This led to more successful results than in most studies in the literature.
- Existing studies involving FT use a maximum of 6 pre-trained TL models, whereas our study aims to select the model that gives the best results by applying almost all (specifically, 18) of the TL models used for classification in the literature. This study is thus the most comprehensive TL study in the literature.
- In addition, because the deep CNN network is trained by using TL-FT and TL-FE on a dataset with a low number of images, classification results have very high performance.

2. Related Works

Several studies have used TL to classify objects. In one study on how to detect COVID, a hospital took 4986 chest CT images, and used DenseNet121, DenseNet201, VGG16, VGG19, InceptionResnetV2, and Xception models to test TL [1]. Fine-tuning was accomplished by freezing feature extractor layers on DenseNet201, which produced the best COVID classification results based on these chest images. Fine-tuning hyper-parameters were adjusted with Dropout 0.2, 0.3, and 0.3 after Conv2D (3,3x3), Global Average Pooling, and three fully connected layers with 256,128,64 neurons. After comparing the performance of these TL models, DenseNet201 achieved an accuracy rate of 0.9818. Meanwhile, another COVID detection study [2] used TL with VGG19, ResNet50V2, DenseNet121, and MobileNet. Fine-tuning involved adding Conv2D, Average Pooling, Flatten, and Sigmoid layers to the last layers of the models for a 460-image dataset. The best result was achieved with MobileNet. [3] used the ResNet50 model with fine-tuning for disease classification on 20,639 crop images. This model modified the values of the batch size, epoch, and learning rate hyper-parameters. In addition, the study removed the model's final three layers and added Conv2D, Average Pooling, Fully Connected, and Softmax layers, in that order, to the model. Classifying 15 different disease states with 300 images from each class yielded an accuracy of 99.26.

Kudva et al. applied hybrid TL using VGG16 and AlexNet networks for cervical cancer detection based on 2198 positive and negative images of the cervix [4]. The hybrid TL method built a CNN from scratch using the initial weights of AlexNet and VGG16. Successful results were obtained, with an accuracy rate of 0.9146. Another study is based on the fact that farmers have a hard time identifying insect pests since they seem identical during the growing stages of the crop [5]. This problem was solved with a deep CNN, with three datasets used to classify the insects. There were 40 classes in the first dataset, 24 in the second, and 40 in the third. The study compared the AlexNet, VGGNet, GoogLeNet, and Resnet TL approaches for insect categorization. Hyper-parameter adjustment for pre-trained networks improved classification performance: the accuracy rate was 0.9675 in the first dataset, 0.9747 in the second, and 0.9547 in the third. All pre-trained models were fine-tuned by adding Average Pooling, Fully Connected, and Softmax layers [5]. Talo et al. used the CNN-based ResNet-34 TL model to analyze MRI images in order to detect abnormalities in the brain [6]. They upgraded the model's last layers, the Dense and

Softmax layers, and achieved 100% accuracy with a 5-fold classification on 613 MR images. Another study, to classify brain tumors, only updated the Softmax part of the pre-trained model [7]. The networks used in the study were AlexNet, GoogLeNet, SqueezeNet, ResNet50, and ResNet101. After hyper-parameter adjustment, AlexNet achieved the highest accuracy rate, 0.9904.

Yang et al. classified spare parts into three categories using TL [8]. They used the VGG19, Alexnet, and ResNet50 models, with the VGG19 model yielding the highest success rate at 0.963. Another study classified benign and malignant skin lesions using a deep CNN [9]. This method was compared to AlexNet, VGG16, DenseNet, MobileNet, and ResNet and found to give the best results, with an accuracy of 0.9143. A skin lesion classification study [10] used ResNet152V2, DenseNet201, and Xception, with accuracy rates of 0.874, 0.874, and 0.891. This study added Global Average Pooling, Dense, Dropout, and Softmax layers while fine-tuning the model, and used two different datasets for malware classification. Another study used VGG16, VGG19, ResNet50, and InceptionV3 as both feature extractors and classifiers to classify malware images [11]. It added the last two layers of VGG16 to the new Dense layers, and only updated the Softmax layers of the other pretreated networks. [12] used fine-tuning in the VGG16 model to classify flowers into five categories by updating only the number of classes in the Softmax layer.

A study on the classification of 21 different cat and dog breeds performed fine-tuning with the VGG16 and VGG19 models [13], with two 4096 sized fully connected layers added to the end of its network. VGG16 and VGG19 achieved accuracy levels of 0.9847 and 0.9859, respectively. For plastic waste classification with FT, [14] utilized InceptionResNetV2, VGG19, VGG16, InceptionV3, and MobileNet. This study added Dropout, Flatten, Dense, Dropout, and Softmax layers to the end of the pre-trained model, and ran experiments on the model's hyper-parameters. A model to classify fresh tea leaves achieved a success rate of 0.98 by applying the InceptionV3 model [15]. In another study, chicks, hatching eggs, and unhatched eggs were classified using VGG16 and VGG19 with Flatten and Softmax layers, achieving success rates of 0.90 and 0.92, respectively [16]. Indian food was classified with InceptionV3, VGG16, VGG19, and ResNet [17]. Global Average Pooling and Softmax layers were applied to the pre-trained network, with InceptionV3 attaining the highest accuracy rate. [18] classified nutrients using EfficientNetB0, EfficientNetB4, Dense201, and MobileNetV3. The last of these had the highest accuracy rate, with Global Average Pooling, Dropout, and Softmax layers added to the end of the model.

3. Materials and Method

TL is a technique for storing information obtained while solving one problem and then implementing it to solve another, related problem. That is, it is a machine learning method in which a model trained for one task can be reused for another [19]. Using this method, the training of the network will take less time and computational power. Furthermore, TL can be used when the number of images in a dataset is insufficient for classification. As a result, classifying small datasets with TL can help achieve high performance. The FT technique, which is used in association with the TL approach, is devoted to making layer or parameter changes to a TL model. There are two broad approaches to TL that use FT in this study. The first of the FT methods duplicates the layer weight values of a trained CNN for a model designed for a new dataset [20]. The training is done only at the classification layer level in the first option, whereas in the second option, a specific part is transferred to the new model rather than copying the weights of all the trained model layers.

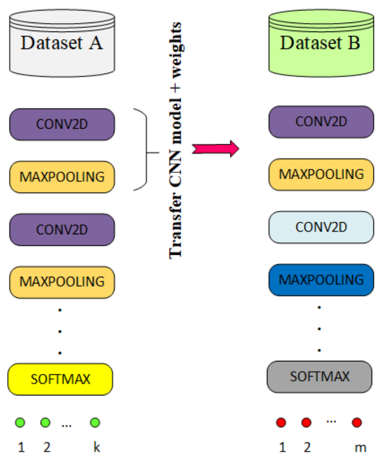


Figure 1. A visual representation of the transfer of some layers and weights from the model trained on Dataset A to the model trained on Dataset B.

Several methods use pre-trained models in applications. The TL technique has improved; as pre-trained networks can now classify new images other than the ImageNet dataset. Performing FT on a pre-trained model changes the hyper-parameters and/or network structure of the model. FT is carried out without changing the weight values of the pre-trained network. There are 3 ways to apply FT to a model.

1. *Extraction of features*: This is the use of the pre-trained network as a feature extractor. After the output layer is deleted from the network, the entire network is used as the new dataset to be classified. Our study used this option and named it the TL-FE scenario.
2. *Use the pre-trained model's architecture*: By assigning the weights of the pre-trained model randomly and retraining the entire network, only the architecture of the network is used.
3. *Fine Tuning*: Another way to use a pre-trained network is to only partially train it and keep initial weight values constant. There are four ways to perform fine-tuning:

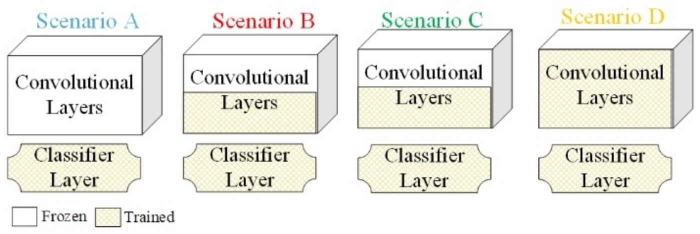


Figure 2. FT framework scenarios from TL.

Scenario A is used when the new dataset is small and very similar to that used by the pre-trained network. In this case, we do not need to train the network from scratch; we simply update the hyper-parameter value in the output layer. The TL-FT1 scenario in our study works according to this logic.

Scenario B applies when the new dataset is small and has little similarity to the ImageNet dataset. The first layers of the pre-trained model are frozen, and the remaining layers are trained. This scenario involves such updates as adding new layers to the network, removing existing layers, and changing the hyper-parameter values of the network. This is how the TL-FT2 scenario in our study works.

Scenario C is utilized when the new dataset is large and highly similar to the ImageNet dataset. In this case, the architecture and initial weights of the pre-trained model remain constant and the model is retrained.

Scenario D is for when the new dataset is large but not very similar to the ImageNet dataset. In this case, the neural network is trained from scratch.

3.1. Pre-Trained Models

Of the pre-trained models used with TL, those examined in this study, briefly explained below, are as follows: VGG16, VGG19, InceptionV3, InceptionResNetV2, Xception, ResNet50, MobileNet, MobileNetV2, DenseNet169, DenseNet201, EfficientNetB0, EfficientNetB1, EfficientNetB2, EfficientNetB3, EfficientNetB4, EfficientNetB5, EfficientNetB6, and EfficientNetB7.

A. The VGGNet Model

The VGG16 model is a CNN-based architecture developed by Simonyan and Zisserman [21]. The VGG16 model consists of 16 layers, of which 13 are convolutions and 3 are fully connected. The filter size in the convolution layers is 3x3 pixels.

The VGG19 model, meanwhile, has 16 convolutions, 5 pooling, and 3 fully connected layers. Since VGG19 has a deep network, the filters used in the convolution layer help reduce the number of parameters. The size of the filter used in the architecture is 3x3 pixels.

B. The Inception Net Model

This is the third version of the DL convolution architecture series developed by Google [22]. It is one of the most advanced architectures used in the field of image classification, and involves a model that combines multiple, differently sized convolutional filters into a new filter, thus reducing both the number of parameters to be trained and the computational complexity [23].

InceptionV3 has a depth of 22 sets of layers and contains 144 layers [23]. The Inception module uses a variety of filters to reduce size. The filter elements in the Inception module are of size 1x1, 3x3, and 5x5. Unlike other DL architectures, this model creates a deep structure rather than a layered one.

InceptionResnNetV2 is a variation of InceptionV3, but with a significantly deeper structure. It is a combination of the Inception structure and a residual connection. Multidimensional convolution filters are combined with residual connections [24], which reduces the training time of the network.

C. The MobileNet Model

This is a deep neural network proposed by Google in 2017. It is smaller and faster than other models. Using depthwise separable convolutions, it applies a single filter in each input channel and a 1x1 filter is used in some convolutions [25]. The separable layers are similar to the convolutional layers in terms of depth, but they differ because they carry out the filtering and merging operations by separating them into two layers. There is a total of 28 layers, excluding point convolutions. Except for the fully connected layer that feeds the softmax layer, each layer is followed by batch normalization and relu activation layers [26].

The MobileNetV2 architecture is a development of the MobileNet model, but faster and more efficient. The size of the feature maps is reduced through a 1x1 convolution. In addition, thanks to the skip connection technique used in the ResNet models, the calculation process is faster [27].

D. The DenseNet Model

When training neural networks, feature maps are reduced due to convolution and subsampling operations. There is also a loss of image quality when transitioning between layers. DenseNet has been developed to use image feature information more effectively [28]. In this model, each layer is fed forward to the other layers, so any layer can access feature information from all previous layers [29]. Thus, the propagation rate of the features to the network increases and the number of parameters decreases.

Figure 3 discusses the common and different parameter numbers and values of the layers in the DenseNet169 and DenseNet201 architectures.

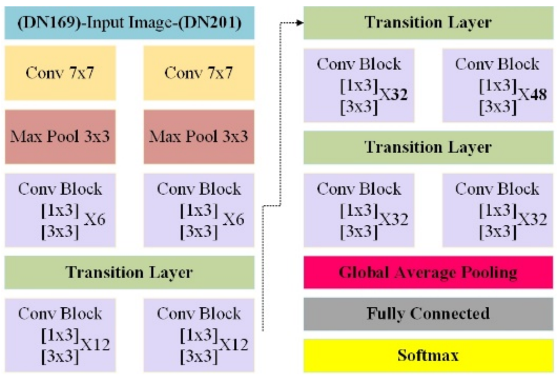

(DN169)-Input Image-(DN201)
Conv 7x7
Conv 7x7
Max Pool 3x3
Max Pool 3x3
Conv Block [1x3] [3x3] X6
Conv Block [1x3] [3x3] X6
Transition Layer
Conv Block [1x3] [3x3] X12
Conv Block [1x3] [3x3] X12
Transition Layer
Conv Block [1x3] [3x3] X32
Conv Block [1x3] [3x3] X48
Transition Layer
Conv Block [1x3] [3x3] X32
Conv Block [1x3] [3x3] X32
Global Average Pooling
Fully Connected
Softmax

Figure 3. DenseNet169 and DenseNet201 model architectures.

E. The EfficientNet Model

This is a network of eight models. As the number of models increases, the number of parameters calculated remains relatively stable [31]. Unlike other advanced models, EfficientNet produces more efficient results by scaling in terms of depth, width, and resolution while trying to reduce the model. EfficientNet requires the channel size to be a multiple of 8. The network width, depth, and resolution are scaled evenly by a set of fixed scaling factors.

F. The ResNet Model

This is the architecture developed by the Microsoft research team to reduce the difficulty of training neural networks at great depth. Unlike standard CNN architecture, this architecture uses short-path connections [32]. Shortcut links have no extra parameters and do not increase computational complexity [33]. They allow the transfer of important information from the previous layer to the next. This architecture includes global average pooling and fully connected layers at the end of the network. The ResNet50 has 50 weight layers.

G. The Xception Net Model

This is basically an evolving network added atop the InceptionV3 network. It is its convolutional layer that makes it different from other networks. A normal network convolution section creates operations by moving a filter over multidimensional matrices, such as width, height, and depth. Xception, on the other hand, provides two distinct approaches to convolution, namely depth wise convolution and pointwise convolution [34]. Depth wise convolution reaches the result by processing only one channel. Since this will cause a loss of features, pointwise convolution is then applied to the image obtained by processing over a channel; the result obtained is a classical convolution 1x1xchannel number [35].

3.2. The Proposed Method

This study proposes three TL-based approaches for the classification of screw, bolt, and nut fasteners. Two of the proposed approaches use FT, and the other uses FE. In scenario 1, only the classifier layer in the pre-trained network model is updated, using TL and FT. Scenario 1 is designated as TL-FT1. Scenario 2 adds TL and FT in the last layers of the model, i.e., it adds new layers to the pre-trained model to obtain a new one. Scenario 2 is designated as TL-FT2. Scenario 3 proposes a new approach: a hybrid structure is created by making use of TL's feature extraction capability and using FE and CNN together. Scenario 3 is designated as TL-FE. Fig. 4 provides an overview of this study.

Given that the datasets to be classified are small, this study uses pre-trained networks because of their high performance and low computation time. Before the implementation of each scenario, the dataset needs to be prepared. In TL-FT1, the classifier layer of the pre-trained network is updated to 18, then the Flatten layer is added to the end of the network. The weight and layer values of the pre-trained network are completely frozen. In TL-FT2, after all the layers and weights of the pre-trained network are frozen, global average pooling, 256 dense, activation relu, 256 dense, and activation relu layers are added to the end of the network, and the number of softmax classifier classes is updated to 18. Network training is completed by training the newly added parts of the network and freezing the previous parts.

In TL-FE, after freezing all the layers and weights of the pre-trained network, the features obtained by training the first network serve as input into the CNN network built from scratch. Unlike other classification studies in the literature, which use images as input to the CNN network, this study uses features as input. Thus, the classification performance of the network significantly improves.

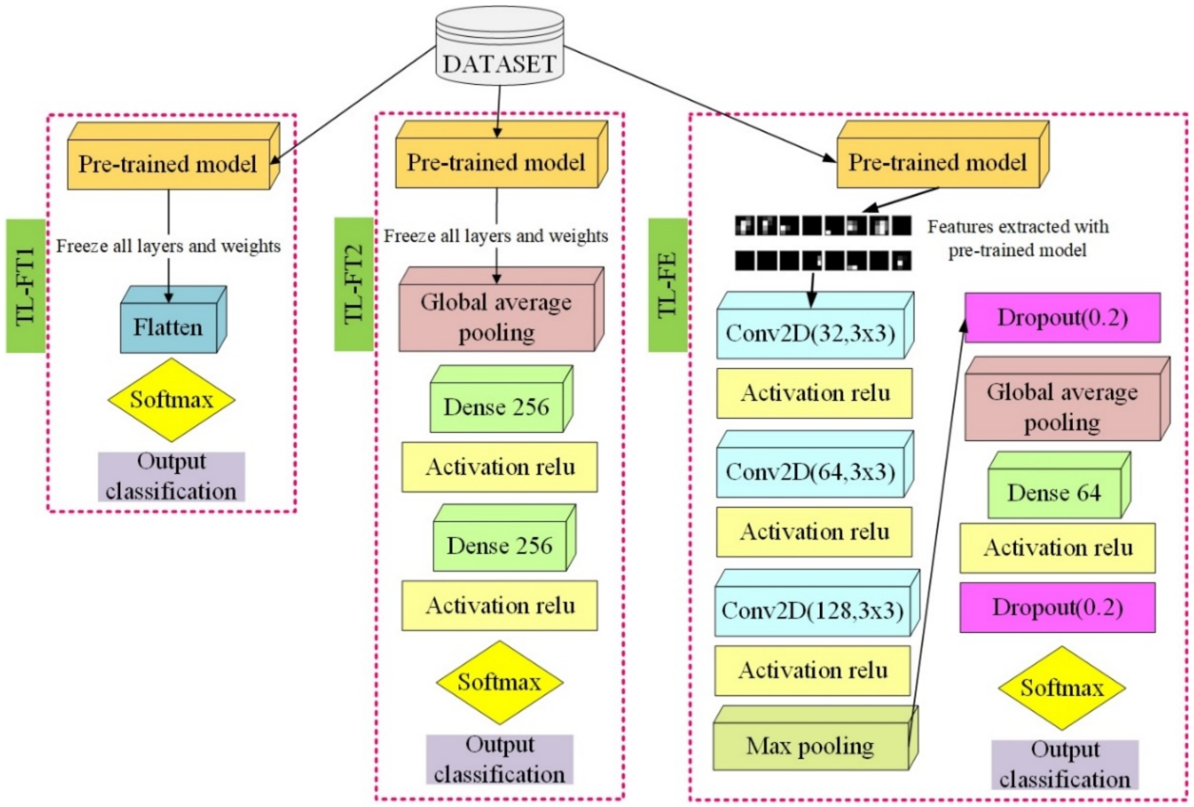


Figure 4. The framework of the proposed approach.

4. Experimental Results

Different scenarios using pre-trained networks have been proposed to classify fasteners. This study examined the performance outputs of each scenario in detail by using 18 different pre-trained networks based on VGGNet, DenseNet, EfficientNet, MobileNet, InceptionNet, ResNet, and Xception. The dataset used for classification includes bolts, screws, and nuts. The performance of each approach proposed in our study was examined in detail, and our study was compared with other studies in the literature. The epoch value in our study is 100, the optimizer algorithm is Adam, the batch size is 32, the learning rate is 0.01, and the image size is 150x150. The hardware features of the computer used in this study are an Intel i7 processor 1.8 GHz CPU, 8GB RAM, and NVIDIA GeForce MX150 GPU.

4.1. Creating Dataset

The dataset contains 150x150 pixel fastener images in RGB format. The fastener images, which contain 18 different classes, include six screw-type, five nut-type, and seven bolt-type classes. The images in the dataset were augmented with data augmentation techniques. For this purpose, each image was shifted by 0.2 on the x axis and by 0.2 on the y axis, as well as being rotated by a 30-degree angle, tilted by 0.2, and magnified by 0.2 percent. In addition, the images were rotated on the horizontal axis. Each class includes approximately 100 images, with 1760 images for the training and 176 images for the testing.

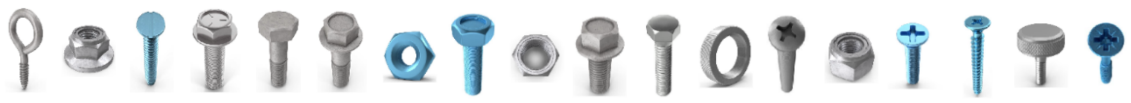
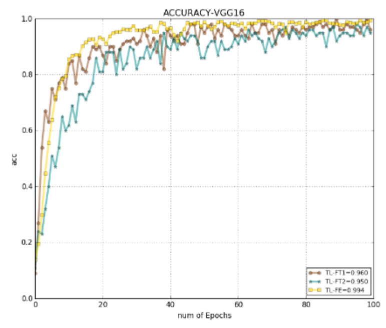


Figure 5. Sample images for the classes in the dataset used in this study.

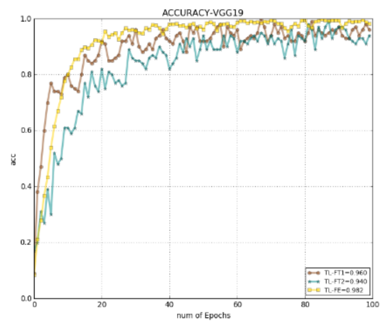
In Fig. 6, there are some results related to testing different transfer learning models for three different scenarios proposed. When the results of the VGG16 and VGG19 models from the VGGNet architecture were examined, it was observed that the VGG16 achieved higher performance and the best result was 99.43% with the TL-FE scenario. When the results of InceptionV3 and InceptionResNetV2 models from the InceptionNet architecture are examined, it is observed that InceptionResNetV2 achieves higher performance and the best result is 99.2% with the TL-FE scenario. Similarly, when DenseNet and MobileNet architectures are examined, the best performance is in DenseNet169(99.5%) and MobileNet(97.5%) models, respectively. When the XceptionNet and ResNet architectures are examined, the best performance was achieved with TL-FE scenario, Xception(99.1%) and ResNet50(97.2%), respectively.

The Fig.7 and Fig.8 depict the accuracy rates of each of the EfficientNet models over the three scenarios. The highest performance was achieved with the TL-FE scenario on all EfficientNet models. When the success rates of each scenario are examined, EfficientNetB0 93% in TL-FT1, EfficienNetB0 96% in TL-FT2 and EfficientNetB6 99.4% in TL-FE.

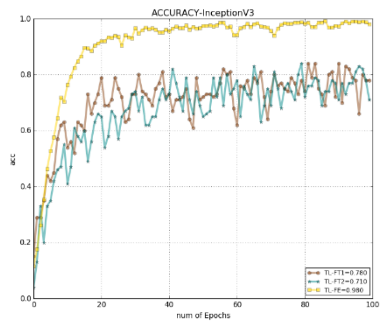
The Fig. 9, Fig.10 and Fig.11 shows the success rates of all TL models for TL-FT1, TL-FT2 and TL-FE scenarios, respectively. When the figures are examined, it can be noticed that the TL-FE scenario has a higher performance than the other two scenarios.



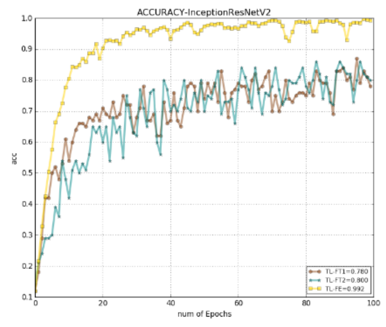
(a) VGG16



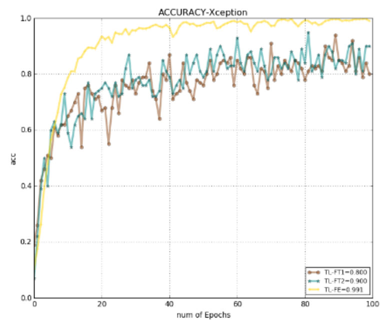
(b) VGG19



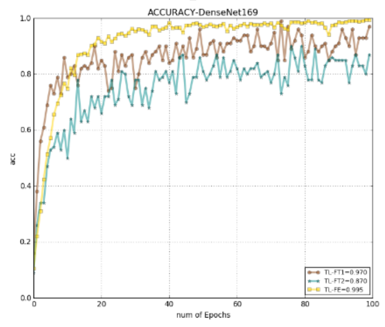
(c) InceptionV3



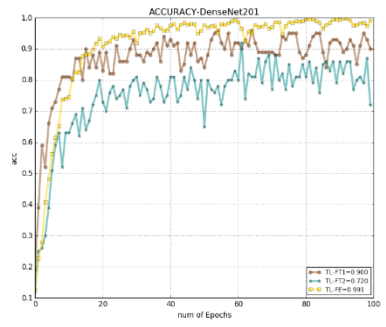
(d) InceptionResNetV2



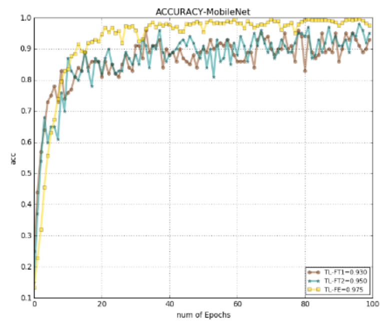
(e) Xception



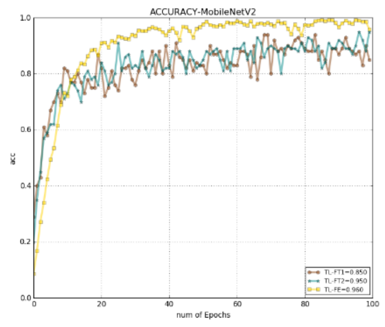
(f) DenseNet-169



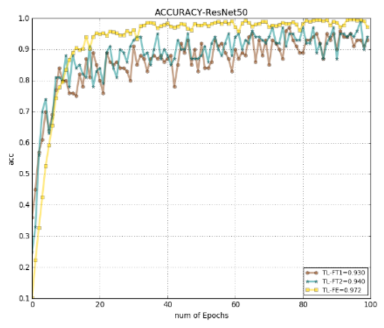
(g) DenseNet-201



(h) MobileNet

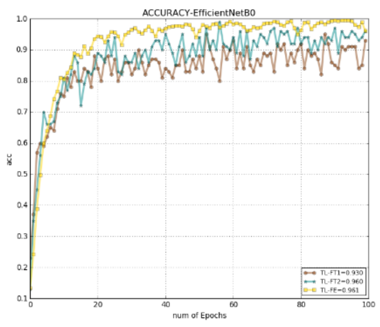


(i) MobileNetV2

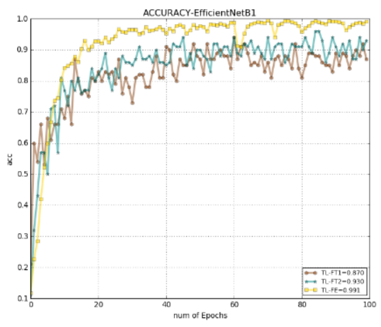


(j) ResNet50

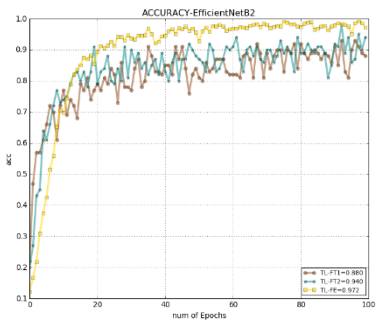
Figure 6. Accuracy distribution of the VGGNets, InceptionNets, XceptionNet, DenseNets, MobileNets and ResNet models according to TL-FT1, TL-FT2, and TL-FE.



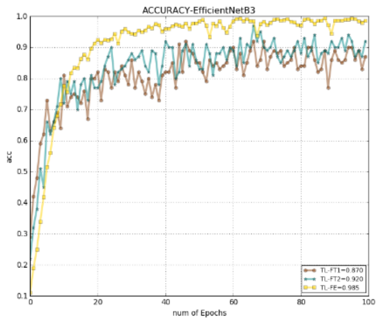
(a) EfficientNetB0



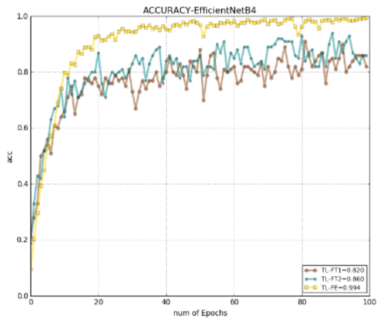
(b) EfficientNetB1



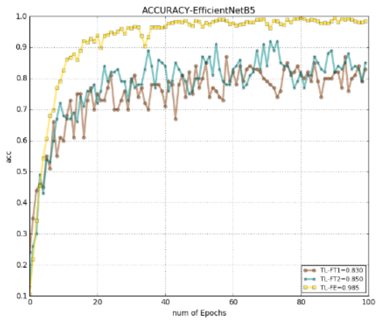
(c) EfficientNetB2



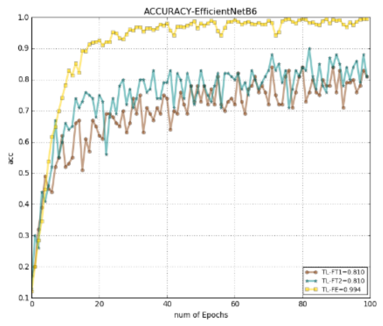
(d) EfficientNetB3



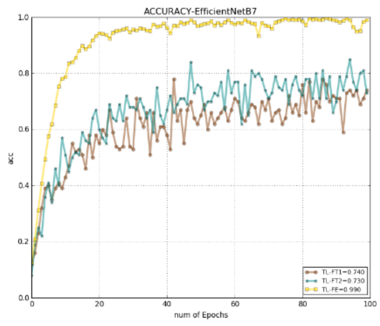
(e) EfficientNetB4



(f) EfficientNetB5

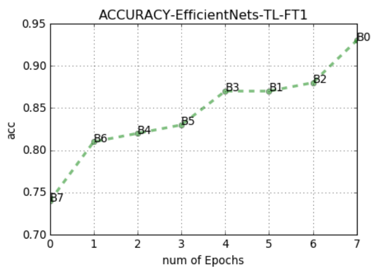


(g) EfficientNetB6

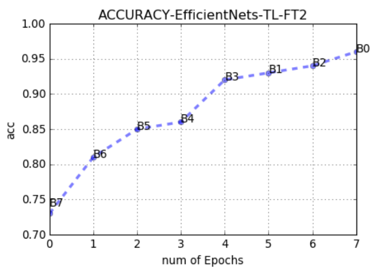


(h) EfficientNetB7

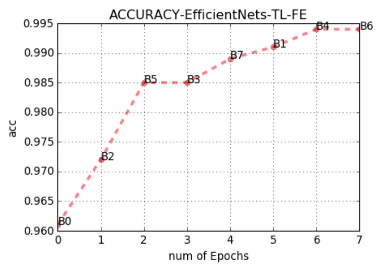
Figure 7. Accuracy distribution of the EfficientNet models according to TL-FT1, TL-FT2, and TL-FE.



(a) TL-FT1



(b) TL-FT2



(c) TL-FE

Figure 8. Accuracy rates of EfficientNets according to (a), (b), and (c) scenarios.

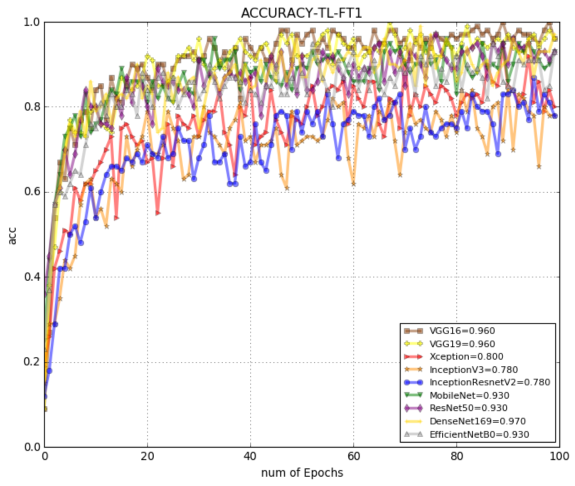


Figure 9. Accuracy rates of all models according to TL-FT1 scenario.

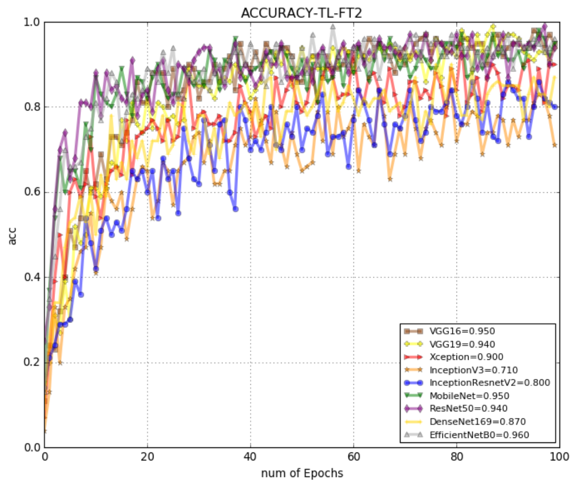


Figure 10. Accuracy rates of all models according to TL-FT2 scenario.

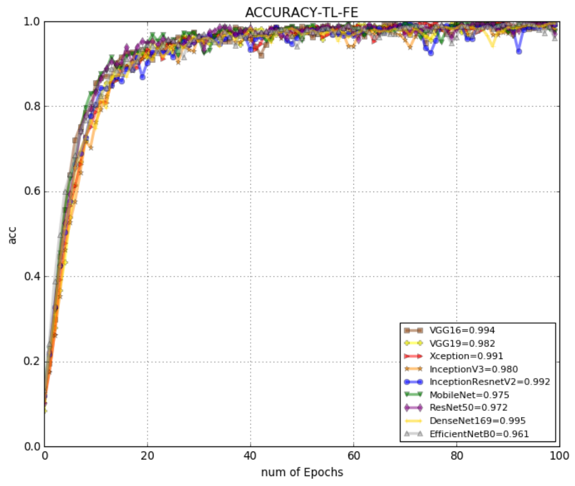


Figure 11. Accuracy rates of all models according to TL-FE scenario.

Table 1. Hyper-parameter values of all scenarios.

Hyper-parameters	**Value**
Optimizer algorithm	Adam
Iteration	100
Batch size	32
Learning rate	0.001
Input image size	150x150

Table 2. Comparison of the accuracy rates of all scenarios.

Model	**TL-FT1**	**TL-FT2**	**TL-FE**
VGG16	0.960	0.950	0.9943
VGG19	0.960	0.940	0.982
InceptionV3	0.780	0.710	0.980
InceptionResNetV2	0.780	0.800	0.992
Xception	0.800	0.900	0.991
MobileNet	0.930	0.950	0.975
MobileNetV2	0.850	0.950	0.960
ResNet50	0.930	0.940	0.972
DenseNet169	**0.970**	0.870	**0.995**
DenseNet201	0.900	0.710	0.991
EfficientNetB0	0.930	**0.960**	0.961
EfficientNetB1	0.870	0.930	0.991
EfficientNetB2	0.880	0.940	0.972
EfficientNetB3	0.870	0.920	0.985
EfficientNetB4	0.820	0.860	0.994
EfficientNetB5	0.830	0.850	0.985
EfficientNetB6	0.810	0.810	0.994
EfficientNetB7	0.740	0.730	0.989

Table 1 contains the hyper-parameter values of all scenarios. Table 2 shows that the TL-FE scenario gives the best results among all models. This means that the proposed new model is successful. As shown in Table 2, the models with the best results were DenseNet169 with an accuracy of 0.97 in the TL-FT1 scenario, EfficientNetB0 with 0.96 in TL-FT2, and DenseNet169 with 0.995 in TL-FE.

Table 3. Performance evaluation of all models according to TL-FT1.

Model	**Accuracy**	**Validation accuracy**	**AUC**	**Validation AUC**	**Precision**	**Validation precision**	**Recall**	**Validation Recall**
VGG16	0.960	0.860	0.964	0.964	0.899	0.899	0.820	0.821
VGG19	0.960	0.880	0.964	0.964	0.897	0.898	0.814	0.814
InceptionV3	0.780	0.510	0.819	0.819	0.563	0.564	0.537	0.537
InceptionResNetV2	0.870	0.270	0.781	0.781	0.494	0.494	0.465	0.465
Xception	0.800	0.480	0.826	0.825	0.605	0.605	0.597	0.597
MobileNet	0.930	0.700	0.896	0.896	0.747	0.747	0.742	0.743
MobileNetV2	0.850	0.510	0.848	0.848	0.659	0.659	0.655	0.655
ResNet50	0.930	0.030	0.741	0.741	0.498	0.498	0.456	0.456
DenseNet169	**0.970**	0.460	**0.995**	0.753	**0.970**	0.460	**0.970**	0.460
DenseNet201	0.900	0.430	0.979	0.721	0.900	0.430	0.900	0.430
EfficientNetB0	0.930	0.500	0.984	0.792	0.930	0.505	0.930	0.500
EfficientNetB1	0.870	0.420	0.962	0.731	0.870	0.420	0.870	0.420
EfficientNetB2	0.880	0.310	0.962	0.676	0.880	0.310	0.880	0.310
EfficientNetB3	0.870	0.320	0.957	0.671	0.869	0.327	0.860	0.320
EfficientNetB4	0.820	0.370	0.926	0.706	0.820	0.374	0.820	0.320
EfficientNetB5	0.830	0.420	0.951	0.723	0.838	0.420	0.830	0.420
EfficientNetB6	0.810	0.210	0.925	0.595	0.816	0.210	0.800	0.210
EfficientNetB7	0.740	0.240	0.892	0.598	0.740	0.232	0.740	0.230

Table 4. Performance evaluation of all models according to TL-FT2.

Model	**Accuracy**	**Validation accuracy**	**AUC**	**Validation AUC**	**Precision**	**Validation precision**	**Recall**	**Validation Recall**
VGG16	0.950	0.860	0.960	0.960	0.890	0.890	0.737	0.738
VGG19	0.940	0.840	0.959	0.959	0.890	0.890	0.713	0.714
InceptionV3	0.710	0.510	0.870	0.870	0.606	0.606	0.487	0.488
InceptionResNetV2	0.800	0.320	0.803	0.803	0.498	0.498	0.400	0.400
Xception	0.900	0.570	0.886	0.886	0.659	0.660	0.563	0.564
MobileNet	0.950	0.660	0.941	0.941	0.783	0.783	0.716	0.716
MobileNetV2	0.950	0.480	0.906	0.906	0.702	0.702	0.626	0.627
ResNet50	0.940	0.070	0.858	0.858	0.897	0.897	0.426	0.426
DenseNet169	0.870	0.420	0.995	0.812	0.884	0.603	0.840	0.350
DenseNet201	0.720	0.410	0.971	0.830	0.758	0.479	0.690	0.230
EfficientNetB0	**0.960**	0.540	**1.000**	0.885	**0.970**	0.587	**0.960**	0.540
EfficientNetB1	0.930	0.430	0.999	0.851	0.948	0.442	0.920	0.420
EfficientNetB2	0.940	0.460	**1.000**	0.795	0.959	0.469	0.940	0.380
EfficientNetB3	0.920	0.550	0.993	0.875	0.935	0.538	0.870	0.420
EfficientNetB4	0.860	0.410	0.990	0.814	0.903	0.463	0.840	0.380
EfficientNetB5	0.850	0.300	0.993	0.720	0.878	0.304	0.790	0.280
EfficientNetB6	0.810	0.350	0.991	0.823	0.839	0.491	0.780	0.280
EfficientNetB7	0.730	0.330	0.966	0.840	0.788	0.340	0.670	0.320

Table 5. Performance evaluation of all models according to TL-FE.

Model	**Accuracy**	**Validation accuracy**	**AUC**	**Validation AUC**	**Precision**	**Validation precision**	**Recall**	**Validation Recall**
VGG16	0.994	0.972	**1.000**	0.994	**0.995**	0.972	0.994	0.972
VGG19	0.982	0.977	0.999	0.997	0.983	0.977	0.980	0.972
InceptionV3	0.980	0.977	0.999	0.994	0.981	0.983	0.978	0.977
InceptionResNetV2	0.992	0.972	**1.000**	0.994	0.993	0.972	0.992	0.972
Xception	0.991	0.943	0.999	0.988	0.991	0.943	0.991	0.938
MobileNet	0.975	0.943	0.999	0.996	0.977	0.949	0.974	0.943
MobileNetV2	0.960	0.938	0.997	0.990	0.961	0.938	0.959	0.938
ResNet50	0.972	0.909	0.997	0.981	0.973	0.909	0.972	0.909
DenseNet169	**0.995**	0.989	**1.000**	1.000	**0.995**	1.000	**0.995**	0.989
DenseNet201	0.991	0.989	**1.000**	0.997	0.992	0.989	0.989	0.989
EfficientNetB0	0.961	0.926	0.998	0.985	0.967	0.931	0.960	0.926
EfficientNetB1	0.991	0.972	**1.000**	0.994	0.991	0.972	0.989	0.972
EfficientNetB2	0.972	0.966	0.998	0.997	0.973	0.966	0.970	0.966
EfficientNetB3	0.985	0.989	**1.000**	1.000	0.987	0.989	0.984	0.989
EfficientNetB4	0.994	0.972	**1.000**	0.997	0.994	0.972	0.993	0.972
EfficientNetB5	0.985	0.926	**1.000**	0.973	0.987	0.926	0.985	0.926
EfficientNetB6	0.994	0.960	**1.000**	0.988	0.994	0.960	0.993	0.960
EfficientNetB7	0.990	0.938	**1.000**	0.985	0.991	0.938	0.990	0.938

Tables 3,4 and 5 show the performance evaluations of TL-FT1, TL-FT2 and TL-FE. The data examined in the table are accuracy, precision, recall and AUC values. When Table 3 is examined, the best results in accuracy, precision, recall and AUC values were 97%, 99.5%, 97% and 97% with DenseNet169, respectively. When Table 4 is examined, the best results in accuracy, precision, recall and AUC values were 96%, 100%, 97% and 96%, respectively, with EfficientNetB0. When Table 5 is examined, the best results in terms of accuracy, precision, recall and AUC values were 99.5% 100%, 99.5% and 99.5% with DenseNet169, respectively.

Table 6. Comparison of the proposed study with the methods in the literature.

Reference	**Number of classes**	**Dataset size**	**Model**	**Method**	**Accuracy ratio**	
[1]	2	4986	DenseNet201 DenseNet121 VGG19 VGG16 InceptionResNetV2 Xception	TF-FT2	0.9818 0.9577 0.9637 0.9437 0.9457 0.9195	
[2]	2	460	VGG19 DenseNet121 ResNet50 MobileNet	TF-FT2	0.8511 0.8533 0.8206 0.8642	
[3]	15	20.639	ResNet50	TF-FT2	0.9926	
[4]	2	2198	VGG16	TF-FT2	0.9146	
[5]	2	Dataset1 with 40 classes Dataset2 with 24 classes Dataset3 with 40 classes	AlexNet VGGNet GoogLeNet	TF-FT2	0.9675 0.9747 0.9597	
[6]	2	613	ResNet-34	TF-FT1	1.000	
[8]	4	236	AlexNet VGG19 ResNet50	TF-FT2	0.9153 0.9587 0.8983	
[10]	7	38.000	ResNet150V2 Xception DenseNet201	TF-FT2	0.874 0.874 0.891	
[11]	25	9339	VGG16 VGG19 InceptionV3 ResNet50	TF-FT1	0.9839 0.9871 0.9625 0.9871	
[12]	5	3520	VGG16	TF-FT1	0.9767	
[13]	21	20.574	VGG16 VGG19	TF-FT2	0.9847 0.9859	
[14]	3	1596	Xception VGG16 VGG19 InceptionV3 InceptionResNet2 MobileNet	TF-FT2	0.7830 0.7212 0.7006 0.7430 0.7840 0.7237	
[15]	3	11.862	InceptionV3	TF-FT1	0.9800	
[16]	3	3924	VGG16 VGG19	TF-FT1	0.9000 0.9200	
[17]	20	10.000	VGG16 VGG19 InceptionV3	TF-FT1	0.7890 0.7820 0. 8790	
[18]	7	5648	EfficientNetB4 DenseNet201 EfficientNetB0 MobileNetV3	TF-FT1	0.9552 0.9405 0.9298 0.9640	
[36]	4	1908	DenseNet201	TL-FT1	0.7333	
[37]	2	5000	MobileNetV2 VGG16 ResNet152V2 DenseNet201	TL-FT1	0.8880 0.9140 0.8958 0.9089	
Our study	18	1760	VGG16 VGG19 InceptionV3 InceptionResNetV2 Xception MobileNet ResNet50 Dense201	TF-FT1 TF-FT2	TL-FT1 0.960 0.960 0.780 0.780 0.800 0.930 0.930 0.900	TL-FT2 0.950 0.940 0.710 0.800 0.900 0.950 0.940 0.710

The accuracy rate of VGG16 is 0.90 and that of VGG19 is 0.92 in [16], applying the TL-FT1 scenario. That of VGG16 is 0.789 and that of VGG19 is 0.782 in [17], applying the TL-FT1 scenario. In our study, the accuracy of VGG16 is 0.960, and that of VGG19 is 0.960.

In [1], implementing the TL-FT2 scenario, the accuracy of VGG16 is 0.9437 and that of VGG19 is 0.9637. In [2], implementing the TL-FT2 scenario, the accuracy of ResNet is 0.8206, that of MobileNet is 0.8642, and that of VGG19 0.8511. In [4], implementing the TL-FT2 scenario, the accuracy of VGG16 is 0.9146. In [8], implementing the TL-FT2 scenario, the accuracy of ResNet50 is 0.8983. In [10], implementing the TL-FT2 scenario, the accuracy of Xception is 0.874. In [14], implementing the TL-FT2 scenario, the accuracy of VGG16 is 7212, that of VGG19 is 0.7006, that of Xception is 0.783, that of InceptionV3 is 0.743, that of InceptionResNetV2 is 0.784, and that of MobileNet is 0.7237.

In our study, the accuracy of VGG16 is 0.950, that of VGG19 is 0.940, that of MobileNet is 0.950, that of Xception is 0.90, that of InceptionV3 is 0.710, and that of InceptionResNetV2 is 0.80.

5. Conclusions

Deep Learning has recently been used to classify many varieties of objects. Since the classification performance of large datasets is higher than that of small datasets, transfer learning (TL) methods have been developed that can also be used on small datasets. Thanks to TL, high computational power is not required during the training of the network, and a network trained on large datasets can also be used to train small ones. Thus, the use of TL has become widespread. There are three different approaches to implementing TL: feature extractor, training from scratch, and fine tuning using the architecture of the pre-trained network. This study suggests the use of TL to classify fasteners. Three different scenarios have been proposed for TL implementation via fine-tuning and feature extraction approaches. The first is to update only the classifier layer of the network using fine-tuning. The second is to add new layers to the end of the mesh using fine-tuning. The last scenario uses the pre-trained network as a feature extractor. The first and second scenarios have been frequently used in the literature. However, the third scenario involves a new method, used here for the first time. In this scenario, the features obtained as a result of the pre-trained network are given as input to a CNN network built from scratch. Thus, the network is guaranteed highly accurate results. In addition, this study tested 18 different models on each scenario one by one, and compared the performance output in detail with results from similar studies in the literature. Those with the best results are DenseNet169 with 0.97 in TL-FT1, EfficientNetB0 with 0.96 in TL-FT2 and DenseNet169 with 0.995 in TL-FE. Of the three scenarios, TL-FE produced the best performance, with results between 0.960 and 0.995. In addition, this study was comprehensive, including more models than other studies in the literature. This study is also important due to including three different approaches and presenting an entirely new one, namely TL-FE. This study developed 1760 images for the classification of fastener datasets, and the approaches proposed in this study can easily be adapted to other datasets.

References

[1] Pathak Y, Shukla PK, Tiwari A, Stalin S, & Singh S. Deep transfer learning based classification model for COVID-19 disease. Pattern Recognit. Lett. 2020; 152: 122-128.

[2] Akgun D, Kabakuş AT, Senturk Z.K, Senturk A, & Kucukkulahli E. A transfer learning-based deep learning approach for automated COVID-19 diagnosis with audio data. Turk. J. Electr. Eng. Comput. Sci. 2021; 29(SI-1): 2807-2823.

[3] Sravan V, Swaraj K, Meenakshi K, & Kora P. A deep learning based crop disease classification using transfer learning. Mater. Today Proc. 2020.

[4] Kudva V, Prasad K, & Guruvare S. Hybrid transfer learning for classification of uterine cervix images for cervical cancer screening. J. Digital Imaging 2020; 33(3): 619-631.

[5] Thenmozhi K, & Reddy U.S. Crop pest classification based on deep convolutional neural network and transfer learning. Comput. Electron. Agric. 2019; 164.

[6] Talo M, Baloglu UB, Yıldırım O, & Acharya UR. Application of deep transfer learning for automated brain abnormality classification using MR images. Cogn. Syst. Res. 2019; 154: 176-188.

[7] Mehrotra R, Ansari M, Agrawal R, & Anand RS. A transfer learning approach for AI-based classification of brain tumors. Mach. Learn. Appl. 2020; 2.

[8] Yang K, Yang T, Yao Y, & Fan SD. A transfer learning-based convolutional neural network and its novel application in ship spare-parts classification. Ocean Coastal Manage. 2021; 215.

[9] Ali MS, Miah M.S, Haque J, Rahman MM, & Islam MK. An enhanced technique of skin cancer classification using deep convolutional neural network with transfer learning models. Mach. Learn. Appl. 2021; 5.

[10] Rahman Z, & Ami AM. A transfer learning based approach for skin lesion classification from imbalanced data. In: 2020 11th International Conference on Electrical and Computer Engineering (ICECE), 2020; Dhaka, Bangladesh. pp. 65-68.

[11] Kumar S, & Janet B. DTMIC: Deep transfer learning for malware image classification. J. Inf. Secur. Appl. 2022; 64.

[12] Giraddi S, Seeri S, Hiremath P.S, & Jayalaxmi GN. Flower Classification using Deep Learning models. In: 2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE), 2020; Karnataka, India. pp. 130-133.

[13] Wang I. H. Lee KC, & Chang SL. Images Classification of Dogs and Cats using Fine-Tuned VGG Models. In: 2020 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE), 2020; Yunlin, Taiwan: IEEE. pp. 230-233.

[14] Lee SW. Novel classification method of plastic wastes with optimal hyper-parameter tuning of Inception_ResnetV2. In: 2021 4th International Conference on Information and Communications Technology (ICOIACT): 2021; IEEE. pp. 274-279.
[15] Qian Y, Li G, Lin X, Zhang J, Yan J, Xie B, & Qin J. Fresh tea leaves classification using inception-V3. In: 2019 IEEE 2nd International Conference on Information Communication and Signal Processing (ICICSP), 2019; Weihai, China: IEEE. pp. 415-419.
[16] Junaidi A, Lasama J, Adhinata FD, & Iskandar AR. Image Classification for Egg Incubator using Transfer Learning of VGG16 and VGG19. In: 2021 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT), 2021; Malang: IEEE. pp. 324-328.
[17] Rajayogi JR, Manjunath G, & Shobha G. Indian food image classification with transfer learning. In: 2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS9), 2019; Miami, Fla: IEEE. pp. 1-4.
[18] Espejo-Garcia B, Malounas I, Mylonas N, Kasimati A, & Fountas S. Using EfficientNet and transfer learning for image-based diagnosis of nutrient deficiencies. Comput. Electron. Agric. 2022.
[19] Ribani R, & Marengoni M. A survey of transfer learning for convolutional neural networks. In: 2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T), 2019; Rio de Janeiro, Brazil: IEEE. pp. 47-57.
[20] Krishna ST, & Kalluri HK. Deep learning and transfer learning approaches for image classification. Int J Recent Technol Eng. 2019; 7(5S4): 427-432.
[21] Simonyan K, and Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv Prepr. 2014; arXiv1409.1556.
[22] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, and Wojna Z. Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016; Las Vegas, NV, USA: IEEE. pp. 2818–2826.
[23] Ucar M. Diagnosis of Glaucoma Disease using Convolutional Neural Network Architectures. Dokuz Eylul Uni. Fac. of Eng. J. of Sci. and Eng. 2021; 23(68): 521-529.
[24] Nguyen LD, Lin D, Lin Z, & Cao J. Deep CNNs for microscopic image classification by exploiting transfer learning and feature concatenation. In: 2018 IEEE International Symposium on Circuits and Systems (ISCAS), 2018; Floransa, İtaly: IEEE. pp. 1-5.
[25] Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, and Adam H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv Prepr. 2017; arXiv /1704.04861.
[26] Zeren MT. Comparison of ssd and faster r-cnn algorithms to detect the airports with data set which obtained from unmanned aerial vehicles and satellite images. MSc, Beykent University, Istanbul, Turkey, 2020.
[27] Baydilli YY. Polen Taşıyan Bal Arılarının MobileNetV2 Mimarisi ile Sınıflandırılması. Eur. J. Eng. Sci. Tech. 2021; 21: 527-533.
[28] Huang G, Liu Z, Maaten LVD, & Weinberger KO. Densely Connected Convolutional Networks. In; IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017; Hawaii, ABD: IEEE.
[29] Aktas A. Image processing applications with deep learning methods. MSc, Marmara University, Istanbul, Turkey, 2020.
[30] Bayram B, Kilic B, Özoğlu F, Erdem F, Bakirman T, Sivri S, & Delen A. A Deep learning integrated mobile application for historic landmark recognition: A case study of Istanbul. Mersin Photogramm. J. 2020; 2(2): 38-50.
[31] Tan M, and Le QV. EfficientNet: Rethinking model scaling for convolutional neural networks. In: 36th Int. Conf. Mach. Learn. ICML, 2019; pp. 10691–10700.
[32] Bayram B, Kılıc B, Ozoglu F, Erdem F, Sivri S, Delen A, Bayrak OC. A study on object recognition with deep learning. In: 10. Turkiye Ulusal Fotogrametri ve Uzaktan Algılama Birligi Teknik Sempozyumu (TUFUAB 2019) 2019; Aksaray, Turkey.
[33] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016; Las Vegas, NV, USA: IEEE. pp. 770-778.
[34] Dandil E, and Serin Z. Breast Cancer Detection on Histopathological Images Using Deep Neural Networks. Eur. J. Eng. Sci. Tech. 2020; Special Issue: 451-463.
[35] Chollet F. Xception: Deep learning with depthwise separable convolutions. In: IEEE conference on computer vision and pattern recognition 2017; Los Alamitos, California: IEEE.
[36] Sivari E. Güzel M. S. Bostanci E. & Mishra A. A Novel Hybrid Machine Learning Based System to Classify Shoulder Implant Manufacturers. Healthc. (Basel) 2022; 10(3), MDPI.
[37] Kalkan M. Bostancı GE. Güzel MS. Kalkan B. Özsarı Ş. Soysal Ö. & Köse G. Cloudy/clear weather classification using deep learning techniques with cloud images. Comput. Electr. Eng. 2022; 102, 108271.