IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 23 March 2025, accepted 7 April 2025, date of publication 10 April 2025, date of current version 18 April 2025.

*Digital Object Identifier 10.1109/ACCESS.2025.3559794*

RESEARCH ARTICLE

# Vision Transformers Versus Convolutional Neural Networks: Comparing Robustness by Exploiting Varying Local Features

**MUHAMMED CIHAD ARSLANOGLU**[1], **ABDULKADIR ALBAYRAK**[2,3], **AND HUSEYIN ACAR**[1]
[1]Department of Electrical and Electronics Engineering, Dicle University, 21280 Diyarbakır, Türkiye
[2]Department of Computer Engineering, Dicle University, 21280 Diyarbakır, Türkiye
[3]Department of Generative AI Program Operations, Hospital Operations, Mayo Clinic, Rochester, MN 55905, USA

Corresponding author: Muhammed Cihad Arslanoglu (cihad.arslanoglu@dicle.edu.tr)

**ABSTRACT** Robustness of deep learning models is a crucial point that should be evaluated in computer vision tasks. Deep learning models must be able to produce the same results under different conditions such as brightness, image rotation, etc. Deep learning models must have high level robustness to be reliable to address real world problems. Unreliable systems may be harmful for critical systems such as security, healthcare, and military. In this study, it is aimed to evaluate image rotation robustness of Convolutional Neural Networks (CNNs) and Transformer-based deep learning models: MobileNetV2, Residual Network 18 (ResNet18), Visual Geometry Group 16 (VGG16), EfficientNetB1, Vision Transformer (ViT), Data-Efficient Image Transformers (DeiT), and Pooling-based Vision Transformer (PiT). Canadian Institute for Advanced Research 100 classes (CIFAR100), Canadian Institute for Advanced Research 10 classes (CIFAR10) and California Institute of Technology 256 classes (Caltech256) datasets were used to evaluate the performance of the models. All deep learning models were trained with original train datasets. Image embeddings of train, test datasets, and their rotated forms were extracted. A Support Vector Classifier (SVC) was trained with embeddings of original dataset and concatenation of original and rotated form of train dataset. Performances on original test dataset and their rotated forms were evaluated. 10, 50, and 90 degrees rotated form of the datasets were used for rotated test dataset. F1-score metrics were used compare model performances. In conclusion, Transformer-based classification algorithms are more robust to image rotation than the CNN-based models.

**INDEX TERMS** Convolutional neural networks, vision transformers, classification, robustness, feature extraction.

## I. INTRODUCTION

Computer Vision is widely employed to address numerous real-world problems, including face recognition, car plate recognition, and image reconstruction [1], [2], [3]. Computers are preferred for solving these problems due to their speed, accuracy, and cost-effectiveness compared to humans. These capabilities make computer vision highly desirable in the industry and pave the way for academic studies. While tasks like bill reading and data entry are tedious and time-consuming for humans, computers and computer vision algorithms can handle them in a matter of minutes with less effort. Computer vision algorithms are also used in critical systems such as healthcare [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Szidonia Lefkovits.

In the early stages of computer vision, hand-crafted features were employed to address such problems [5]. Examples of hand-crafted feature extraction algorithms include Histogram of Gradient (HoG), Local Binary Pattern (LBP), and Sobel edge detection algorithm. However, these algorithms are sensitive to environmental conditions. Factors like brightness level, device orientation, and image quality may decrease the accuracy of hand-crafted feature extraction algorithms.

The development of machine learning algorithms and advancements in hardware capacities have streamlined these processes and increased their robustness to environmental factors. Numerous techniques have been applied to deep learning algorithms, especially Convolutional Neural Networks (CNNs), to enhance their accuracy [6], [7]. However, it is crucial to note that the robustness of deep learning models is also a significant consideration that should be evaluated. Deep learning models must be able to maintain their performance in different conditions to be considered reliable. Particularly, low-level reliable deep learning models may lead to harmful results in fields such as healthcare and security.

Vision Transformer (ViT) algorithm represents a groundbreaking approach in the field of computer vision, revolutionizing how machines perceive and understand visual information [8]. Traditional CNNs have long dominated image recognition tasks, but ViT introduces a novel concept by leveraging the power of transformers, originally developed for natural language processing (NLP) [9]. By incorporating self-attention mechanisms, ViT enables machines to capture global dependencies and relationships within images, fundamentally reshaping the way visual data is processed. ViT breaks down an input image into patches and treats them as sequences of tokens, similar to how words are treated in language tasks. These patches are then processed through multiple transformer layers to learn representations that capture both local and global information. This paradigm shift opens up exciting possibilities for advancements in image recognition, object detection, and scene understanding areas.

CNNs, which have been the dominant approach in image classification for many years, are characterized by their hierarchical structure and local receptive fields. They consist of convolutional layers that apply filters across the input image, followed by pooling layers that reduce spatial dimensions, and fully connected layers for classification. CNNs leverage the convolution operation to capture local patterns and spatial hierarchies, enabling them to extract meaningful features from images. In terms of image classification performance, CNNs have a proven track record of achieving high accuracy and have been extensively optimized for this task.

ViT, on the other hand, has shown remarkable performance as well, often achieving competitive results with CNNs. However, ViT may require larger amounts of training data and computational resources compared to CNNs due to the self-attention mechanism and the reliance on patch-based processing. The key difference, therefore, lies in how the information is processed and represented. CNNs excel at capturing local features and spatial hierarchies, making them effective for tasks where such properties are critical, such as edge detection and object localization. ViT, on the other hand, leverages the power of self-attention to capture long-range dependencies and contextual information, allowing it to capture global patterns and relationships within the image. The way in which CNNs and ViT methods learn images obtained from different angles needs to be analyzed. This is crucial to determine how the specified models can be trained and tested taking into account the hardware resources.

Obtaining feature vectors from intermediate layers and using them for classification is an approach to improve classification performance [10], [11]. Especially after the widespread use of deep learning methods, it has become one of the frequently used approaches. Getting feature vectors from the intermediate layers of deep learning models has proven to be highly effective in various applications. These feature vectors, also known as embeddings, capture important and meaningful representations of the input data that can be used for tasks like similarity analysis, clustering, and transfer learning. The performance of obtaining feature vectors from intermediate layers depends on several factors. Firstly, deep learning models are typically designed to learn hierarchical representations, with earlier layers capturing low-level features (e.g., edges, textures) and next layers capturing high-level features (e.g., object shapes, semantic concepts) [12]. Feature vectors obtained from intermediate layers tend to strike a balance between these levels of abstraction, making them useful for a wide range of tasks.

Secondly, the performance can be influenced by the architecture and depth of the deep learning model. Deeper models often have more expressive power but may also be prone to overfitting, especially when dealing with limited training data. It's important to consider the trade-off between model complexity and generalization performance. Additionally, the task-specific nature of feature vectors should be taken into account for further analysis in machine learning. For certain tasks, such as object recognition, feature vectors from intermediate layers of pre-trained models (e.g., ImageNet) have been found to be transferable and effective. However, for domain-specific or highly specialized tasks, fine-tuning or training models from scratch may yield better results.

In this study, we evaluated f1-score performance of CNN and Transformer-based deep learning models under various image rotation conditions. We selected the well known CNN-based models that are VGG16, ResNet18, MobileNetv2, and EfficientNetB1. For Transformer-based architecture, we used vanilla ViT, PiT, and DeiT models. We used pretrained version of all these models since it is proved that transfer learning boost performance of models [13]. We used the SVC because it has been shown in the literature to give better results than the linear layer [14]. We split trainig stage into two part: deep learning model training and classifier training. In deep learning model training stage, we trained both CNN and Transformer-based models with CIFAR100, CIFAR10, and Caltech256 train datasets. Then we extracted feature vectors of used train datasets and its 10, 50 and 90 degrees rotated forms.

We trained a SVC algorithm with original train datasets and its rotated forms separately. As a result, we obtained 7 trained SVC model that is trained with embeddings of original train dataset, original dataset embeddings and its 10 degree rotated form, original dataset embeddings and its 50 degree rotated form, original dataset embeddings and its 90 degree rotated form, embeddings of 10 and 50 degrees rotated form of original train dataset, embeddings of 10 and 90 degrees rotated form of original train dataset, embeddings of 50 and 90 degrees rotated form of original train dataset. In the test stage, we extracted feature vectors of test datasets and their 10, 50 and 90 degrees rotated forms. Consequently, we evaluated f1-score performance of 7 different SVC algorithm with test dataset and its rotated forms. We used only 10, 50, and 90 degrees since after 90 degrees is like mirroring the image. We performed rotation operation around center of the image and to only left direction to eliminate randomness. We used f1-score evaluation metrics to compare models performances since f1-score is harmonic average of recall and precision which means it is more reliable to unbalanced datasets.

The contributions of this study to the literature can be stated as follows:

- Robustness comparison of CNN-based and Transformer-based algorithms on images taken from different angles and different datasets.
- Examining the classification performance of the embeddings obtained from the subsequent layers of the standard ViT algorithm.
- Performance evaluation of local and global feature extraction algorithms
- Obtaining competitive f1-score while reducing time that is spent in training stage.
- Found optimum rotation degrees to improve model's robustness for rotation operations.

We organized the rest of this paper as follows: In the second section, the relevant studies with our paper are examined. We explained used technique with details in section III. Then obtained results are shared, analyzed, and environment setup settings are mentioned. We discussed the final results in section V. The conclusion and future works were stated at the last section.

## II. RELATED WORKS

We categorized related works into deep learning models, hand-crafted feature extraction, and combination of both algorithms. These algorithms exhibit advantages and disadvantages compared to each other. Hand-crafted feature extraction methods are easier to implement and require less computation power since they focus on low-level features such as edges, colors, or contours.

On the other hand, deep learning based models focus on high-level and low-level features of the image at the same time. This capability makes deep learning models more robust and enhance their adaptability to various conditions. Deep learning-based techniques need more resources than standard techniques since they include a greater number of parameters and calculations [15], [16]. CNN is one of the most used deep learning model in the literature for many fields such as object detection, classification problems, or segmentation tasks. ResNet [17] and EfficientNet [18] models are competitive examples of CNN-based algorithms. It is also possible to combine both hand-crafted features and deep learning models to boost performance [19].

While accuracy score of an approach has a significant role in real life problems or academic studies, their robustness should be considered too. Weak robustness may become an irreversible danger for critical systems such as healthcare or security departments. A proposed model should consistently produce the same results under varying conditions, remaining unaffected by factors such as weather, sunlight, or slight differences. The model also should keep this performance for intentional attacks such as inserting malicious pixels into the images, perturbations and so on. There are many factors that impact robustness of a deep learning model such as inductive bias, feature extraction way or parameter numbers of the model.

Many techniques were applied to increase the robustness of deep learning models [20], [21]. Having a clear image may not be possible for every time. Different conditions can be simulated using data augmentation techniques since collecting noised, blurred or rotated images need extra effort. Yedla et al. states that CNN has weak robustness performance when the images have low and high-frequency information. They propose a stochastic filtering data augmentation technique to make CNN more robust [22]. In [23], a data augmentation technique is proposed. A combination of two different images are given to the model and the loss also is calculated by using label combination. Reference [24] compare performance of AlexNet on the original facial expression recognition dataset and noised one. Obtained results show that training AlexNet with noised dataset makes the model more robust compared to clean dataset. Kanjar and Marius create color distortion ImageNet dataset to study effect of colors on CNN-based state-of-the-art models. They state that colors and color spaces has a significant role on performance of most used CNN models [25]. In [26], it is shown that increasing shape bias and reducing texture bias has positive impact on robustness of the model. They applied style transfer to original dataset due to increase shape bias of the model. Zhang et al propose a technique that combines two different images before given the model. The target label is also combination of each image [23]. Madry et al. evaluate robustness as an optimization problem. They find a point that maximum loss between original and attacked images. Meanwhile, classification loss is minimized [27]. It is also possible to teach the model attacked images due to increase its robustness. In [28], attacked images are added into original dataset to make deep learning model more robust. Although the more parameters increase data learning capability of the model, it also causes to overfitting especially if there is no sufficient data compared to model complexity. Dropout layer prevent the model for overfitting and helps to learn most general features of the dataset [29]. Increasing robustness may decrease accuracy of the model. However, Yang et al. reveal that there is no need to trade off between robustness and accuracy. They also evaluate robustness performance of two models using dropout and some training strategy [30]. In [31], robustness of ResNet and ViT are compared on many aspects such as texture bias, model size, layer collapse or perturbations. They show that, ViT is more robust than ResNet when it is trained with sufficient data. Moreover, texture bias of ViT has reverse relation with patch size.

## III. MATERIAL AND METHODS

We used 7 models which are CNN-based MobileNetV2, ResNet18, VGG16, EfficientNetB1 and Transformer-based Vanilla ViT, DeiT, and PiT models. We initialized model weights using pretrained weights that are trained on mainstream which is ImageNet1k. We made transfer learning on CIFAR10 downstream dataset. Transfer learning reduce training time and helps model to learn more representative features more easily. We explained how ViT classification algorithm works with its details in section III-A.

### A. VISION TRANSFORMER (ViT)

ViT is a Transformer-based classification algorithm. ViT is a special form of attention mechanism and transformer encoder that used in NLP at the first time by Vaswani et al. [9]. Architecture of ViT is so similar to its original form in NLP. There are three main components in ViT: patch embedding, transformer encoder and classification stage, respectively. In patch embedding stage, an image is split into patches and linear layer is applied to these patches to reduce size of patches and extract significant features from these patches. Obtained feature vectors are also called as visual tokens. A similarity matrix is calculated using the extracted visual tokens in the transformer encoder. Transformer encoder has a crucial role for ViT algorithm. Each row and column in the similarity matrix represent similarity between the patches. At the last layer, a classifier, mostly linear layer, is used to get final result and calculate loss between actual and predicted output. ViT focuses on global dependencies using patch embedding strategy and transformer encoder block unlike CNN-based algorithms which extract local features. Since global dependencies are not so sensitive to little changes like noise or real-life scenarios such as image rotation, environment brightness and blurring, ViT is more robust compared to CNN-based algorithms. Figure 1 displays high level architecture of ViT algorithm.

#### 1) PATCH EMBEDDING

An image with $H \times W \times C$ size is split into patches that each patch has $L \times (P \times P) \times C$ shape. Defined parameters H, W, C, L and P indicates height, width, channel count, patch count and patch size, respectively. These patches are also named as visual words since they are similar to words in NLP. A linear layer is used to reduce dimension of these visual words and extract most valuable features as in Eq. 1 where n, w and d represents each pixel of image, weights and projection dimension. These obtained feature vectors also are called as visual tokens $v \in R^{L\times d}$ where L and d means visual token count and projection dimension respectively. Learnable position embeddings are added into the visual tokens like in Eq. 2 due to specify position of each visual tokens. A special token which is known as classification token is concatenated with the visual tokens like in Eq. 3. Finally, these tokens are fed into transformer encoder which is one of the most important part of ViT algorithm.

$$v = \begin{bmatrix} \sum_{i\in B_1}^{P\cdot P\cdot C} n_i\cdot w_1 i,\ \sum_{i\in B_1}^{P\cdot P\cdot C} n_i\cdot w_2 i,\ \ldots,\ \sum_{i\in B_1}^{P\cdot P\cdot C} n_i\cdot w_d i \\ \sum_{i\in B_2}^{P\cdot P\cdot C} n_i\cdot w_1 i,\ \sum_{i\in B_2}^{P\cdot P\cdot C} n_i\cdot w_2 i,\ \ldots,\ \sum_{i\in B_2}^{P\cdot P\cdot C} n_i\cdot w_d i \\ \vdots \\ \sum_{i\in B_L}^{P\cdot P\cdot C} n_i\cdot w_1 i,\ \sum_{i\in B_L}^{P\cdot P\cdot C} n_i\cdot w_2 i,\ \ldots,\ \sum_{i\in B_L}^{P\cdot P\cdot C} n_i\cdot w_d i \end{bmatrix}_{Lxd} \tag{1}$$

$$v' = v + p, \quad p, v \in R^{L\times d} \tag{2}$$

$$v' = concat(v, cl), \quad v' \in R^{(L+1)\times d}, \quad cl \in R^{1\times d} \tag{3}$$

#### 2) TRANSFORMER ENCODER

Attention mechanism is used to find relationship between visual tokens - image patches- in Transformer Encoder due to highlight value of strong relations and reduce the weak ones. Transformer Encoder block includes four component which are normalization [33], multi-head self-attention block, MLP layer and residual connections.

Visual tokens are normalized before processed by attention mechanism like Eq. 4. Normalized visual tokens are copied three times and named as Query(Q), Key(K) and Value(V) vectors. Self attention mechanism calculates an attention map between Q and K using cosine similarity. Softmax activation function is applied to the attention map due to squeeze values between 0 and 1 as in Eq. 5. These values represent how much a visual token has relation with others. The more attention map value the more relation between the patches. V feature matrix and attention map are multiplied due highlight most valuable features and reduce affect of undesired features like noise as in Eq. 6. An MLP block that includes linear layer and GeLU activation function [34] is applied to the weighted feature vector like in Eq. 7. This all processes are done in transformer encoder block.

$$v' = normalize(v') \tag{4}$$

$$QK = softmax\left(\frac{Q\times K^T}{\sqrt{d}}\right) \tag{5}$$

$$x = QK \times V \tag{6}$$

$$\begin{bmatrix} linear(x) \\ GeLU(x) \\ linear(x) \\ GeLU(x) \end{bmatrix} \tag{7}$$

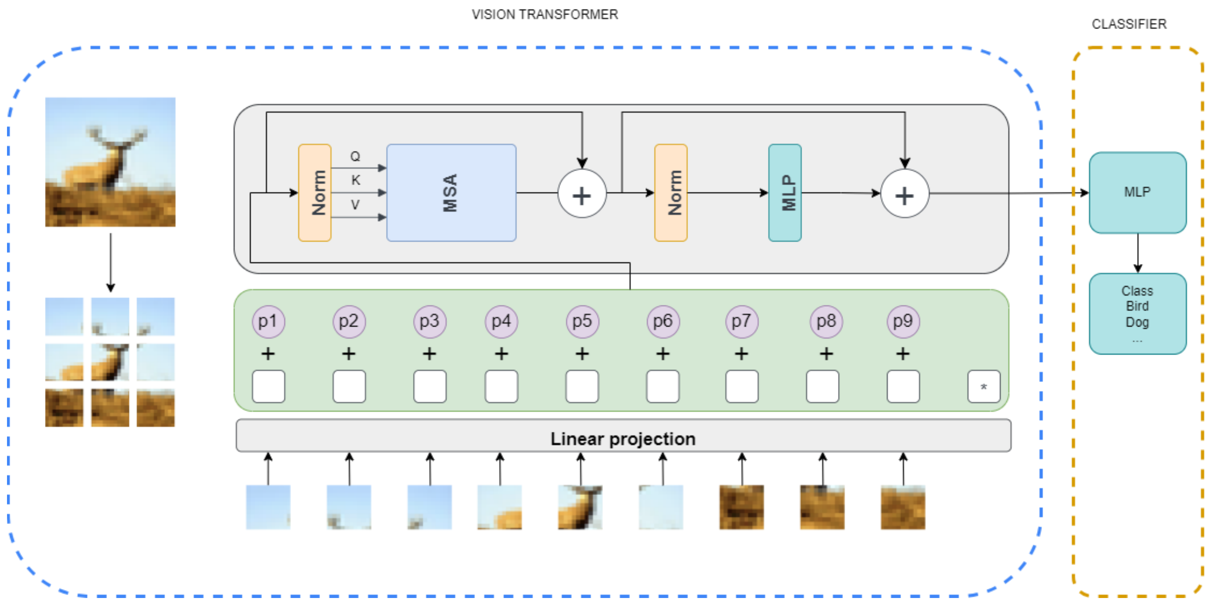

**FIGURE 1.** **High-level architecture of Vision Transformer (ViT). ViT processes images by splitting them into patches. Each patch is then projected using a linear layer to reduce its dimension and extract the most valuable features. Obtained embedding is also called as visual words. Positional information is added to these visual words to maintain spatial relationships. Next, a multi-head attention mechanism calculates the relationships between each embeddings. Finally, an MLP (Multi-Layer Perceptron) layer is trained using these embeddings to complete the process and classify the images.**

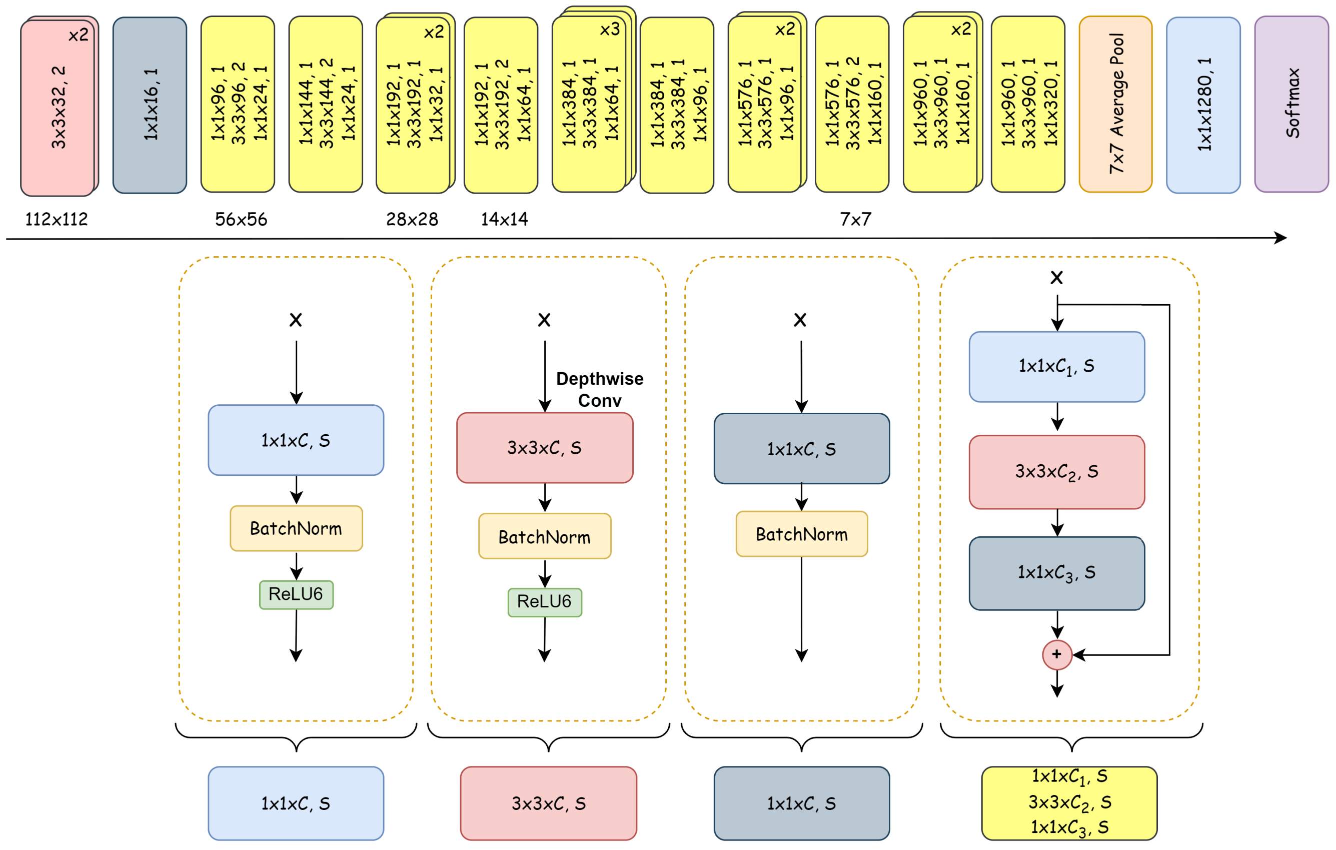

**FIGURE 2.** **High-level architecture of ResNet18 algorithm.**

### *B. MobileNetV2*

MobileNetV2, introduced by Google, is a CNN-based classification algorithm [35]. It consists of 15 inverted residual bottlenecks (IRBs). Each IRB includes expansion layers, depthwise separable convolutions, and projection layers, respectively. The expansion layer is a stack of

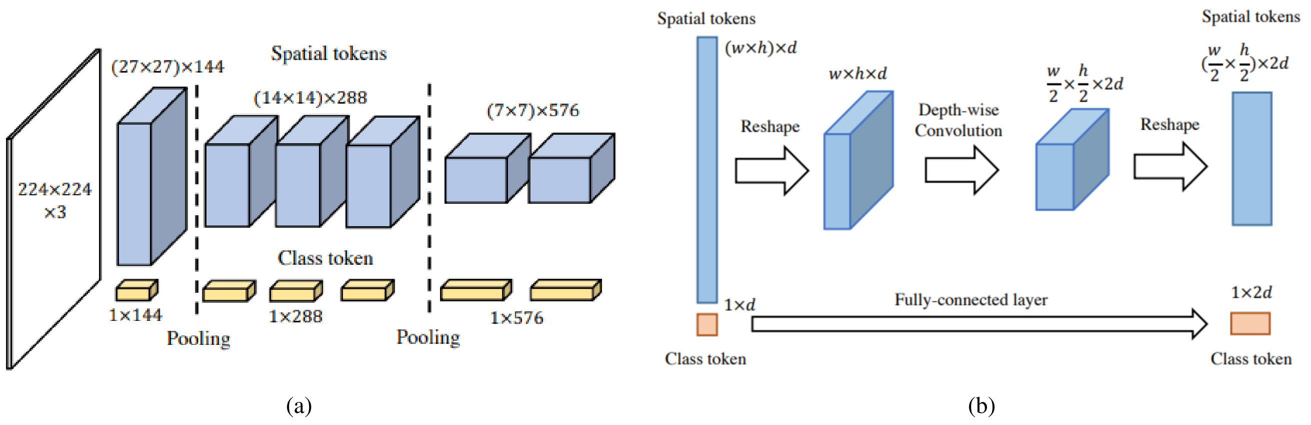

**FIGURE 3.** **(a) shows high-level architecture of PiT algorithm. In (b), PiT pooling layer is displayed [32].**

1 × 1 convolutions with high filter sizes and a ReLU6 activation function, designed to expand the feature vector channel size. Following the expansion layer, a depthwise separable 3 × 3 convolution layer is applied. This depthwise convolution reduces computational complexity by preventing inter-channel information exchange. The projection layer, which follows the depthwise separable convolution layer, applies a 1 × 1 convolution with a low channel size to reduce the feature vector channel size. In the final stage, a global average pooling layer is applied to the feature vector with a size of 1280, and the result is fed into a linear layer for classification. We illustrated high-level architecture of MobileNetV2 in Figure 2.

## C. PiT

Heo et al. propose a CNN and Transformer-based hybrid model that is called PiT [32]. Depthwise convolution layers are applied to given input image before patch embedding of vanilla ViT algorithm. Convolution layers increase channel dimension and decrease spatial dimension of inputs. They also state that scaling channel dimension up has impact on performance of the model.

## D. DeiT

Transformer-based DeiT, developed by Facebook FAIR group, is an improved form of ViT algorithm [36]. Touvron et al. proposes distillation token and label distillation in addition to vanilla ViT algorithm to decrease data and hardware requirements. DeiT algorithm concatenate a distillation token with patch embeddings unlike vanilla ViT. For label distillation, soft and hard label distillation approaches is introduced in DeiT algorithm like in Equation 8 and 9, respectively. In Equation 8 and 9, $\mathcal{L}_{CE}$ is cross-entropy, KL is Kullback-Leibler (KL) loss, $\lambda$ is coefficient to balance KL and cross-entropy, $\Psi$ is softmax function, $Z_s$ is student model logits, $\tau$ is distillation temperature, $Z_t$ is teacher model logits, and $y_t$ is decision of teacher model.

$$\mathcal{L}_{global} = (1-\lambda)\cdot\mathcal{L}_CE(\psi(Z_s), y) + \lambda\cdot\tau^2\cdot KL(\psi\left(\frac{Z_s}{\tau}\right), \psi\left(\frac{Z_t}{\tau}\right) \qquad (8)$$

$$\mathcal{L}_{global}^{hardDistill} = \frac{1}{2}\cdot\mathcal{L}_CE(\psi(Z_s), y) + \frac{1}{2}\cdot\mathcal{L}_{CE}\,(\psi(Z_s), y_t) \qquad (9)$$

## E. ResNet18

He et al. proposed ResNet architecture that has residual connections unlike other many CNN-based architectures [17]. ResNet architecture brings solution to gradient vanishing and exploding which makes the model more easy to train. ResNet18 is consist of stack of 18 convolution layers with residual connections. Each layer has different stride and filter sizes. It's high level architecture is illustrated in Figure 4.

## F. VGG16

VGG16, developed by visual geometry group from oxford, has thirteen consecutive convolution layers, five max pooling layer, and three linear layer at end of the network [37]. VGG16 uses small sized convolution layers that helps model to learn the data more efficiently. Its high level architecture was displayed in Figure 5.

## G. EfficientNetB1

EfficientNetB1 is a member of EfficientNet CNN family. EfficientNet family models were developed by Tan et al. This models were optimized to achieve the best performances unlike other CNN models. Depth, width and resolution properties of EfficientNet models are scaled together with a coefficient instead of one by one like other CNN models. We shared high-level architecture of EfficientNetb1 model in Figure 6.

## H. OBTAINING FEATURE VECTORS FROM MODELS

Last linear layer that is used to classify image is removed from deep learning models. Last layers of the new models give embeddings of the input image. This embedding size is displayed in Table 1 **output size** column. This process was repeated for all models and all datasets for different rotation degrees.

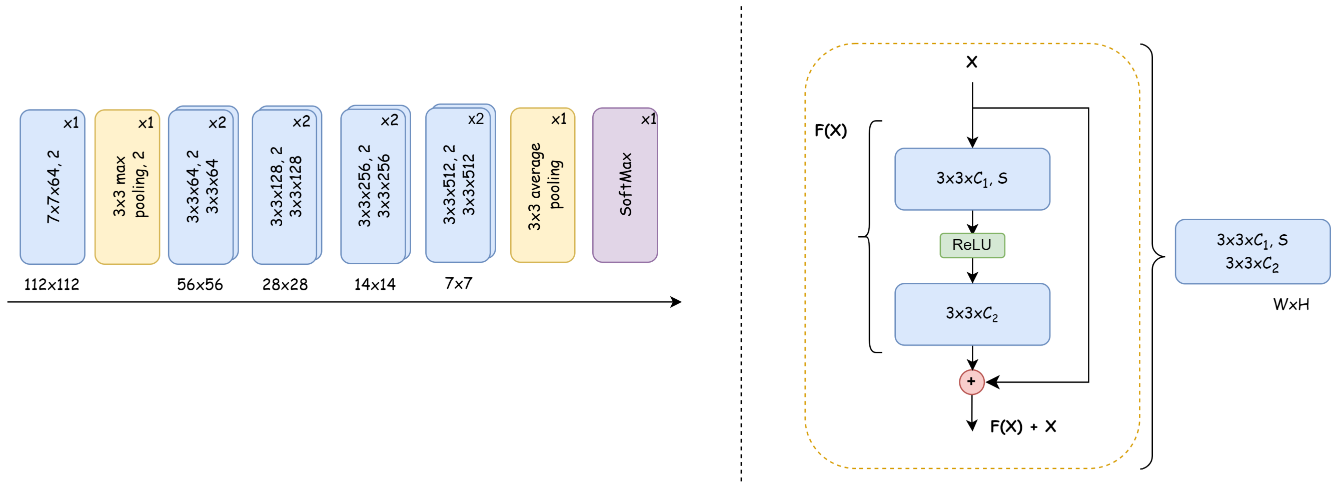

**FIGURE 4.** High-level architecture of ResNet18 algorithm.

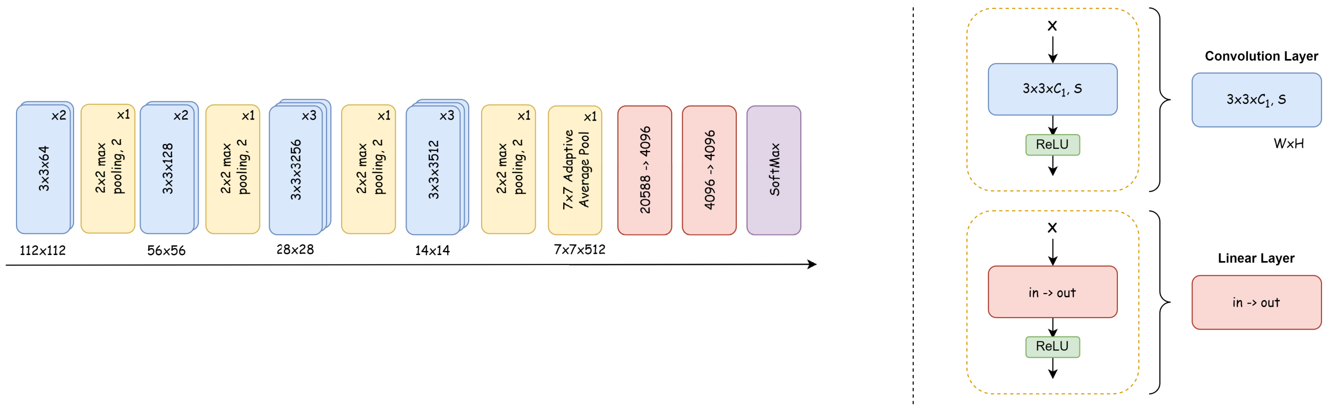

**FIGURE 5.** High-level architecture of VGG16 algorithm.

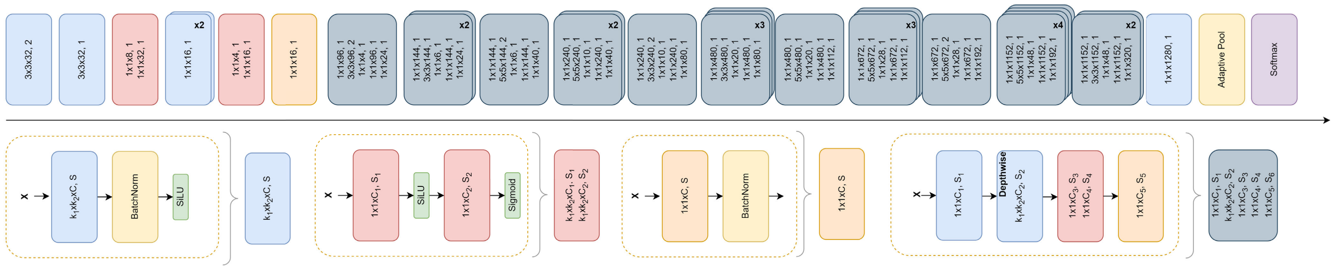

**FIGURE 6.** High-level architecture of EfficientNetB1 algorithm.

### I. INCREASING ROBUSTNESS OF MODELS BY CONCATENATING ROTATED FEATURES

In the proposed approach, each deep learning model is trained using original train datasets. Embeddings of train, test and their rotated forms were extracted as mentioned in Section III-I. An SVC is trained in two stages using extracted train dataset embeddings. In first stage, the classifier is trained using only original train dataset and it is tested on original test dataset and rotated test dataset. In second stage, embeddings of original dataset and its rotated forms, which are 10, 50 and 90 degrees, were concatenated and a SVC is trained with this dataset. All test datasets were evaluated with this trained SVC. We illustrated proposed approach in Figure 7.

## IV. EXPERIMENTAL RESULTS

### A. SETUP

Some properties of used models were given in Table 1. All experiments were conducted on commonly used three different datasets that are CIFAR100, CIFAR10, and Caltech256.

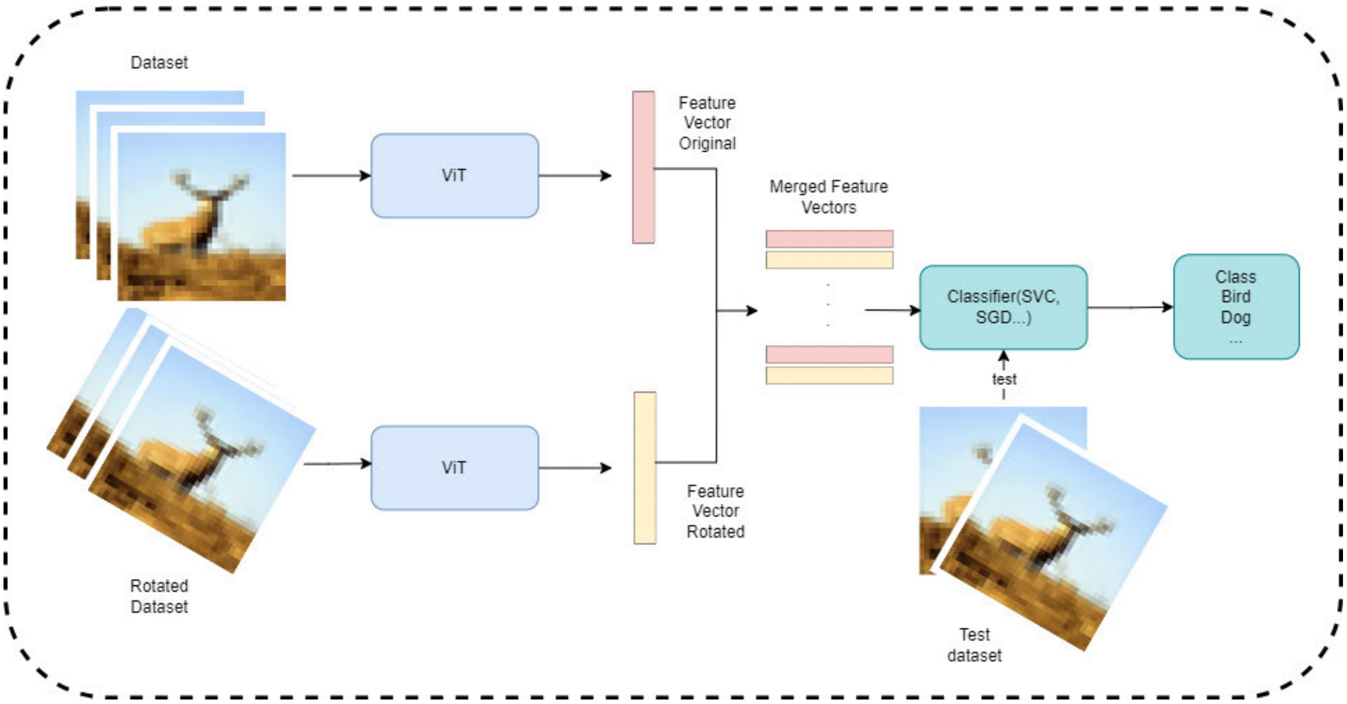

**FIGURE 7.** High-level architecture of proposed approach. Feature vectors of original dataset and rotated dataset are extracted using model, which is ViT in this Figure. These feature vectors are concatenated and given to a classifier.

We used PyTorch and timm library to implement models and use pretrained weights that is trained on ImageNet1k dataset. ImageNet1k dataset has 1000 classes and contains almost 1.3M training images. We used Google Colab for training and testing steps for all experiments. Google Colab has the following hardware properties: 16GB Random Access Memory(RAM), Intel(R) Xeon(R) CPU @ 2.20GHz and Tesla T4 GPU with 16gb Memory. We selected same model parameters for all deep learning models. Batch size were set 32, Adam optimizer were used with 0.0001 learning rate. We trained the all models until their train accuracy reach to 0.98. We used sklearn library SVC implementation with Radial Basis Function (RBF) kernel. We shared detailed parameters of Transformer-based algorithms below. Input image size, output embedding size, and parameter size for all models are shared in Table 1.

*ViT Parameters:* We used tiny architecture of ViT model that includes 224 $\times$ 224 input image size, 16 $\times$ 16 patch size, 192 patch embedding size, 12 layer, 3 head, and 192 output embedding size. This model has 5.7M learnable parameters.

*PiT Parameters:* We used tiny architecture of PiT model that includes 224 $\times$ 224 input image size, 16 $\times$ 16 patch size, 8 convolution stride, [32,32,32] base dimension, [2,6,4] layers, and [2,4,8] heads with 4.8M total learnable parameters. This model has 4.8M learnable parameters.

*DeiT Parameters:* We used tiny architecture of DeiT model that includes 224 $\times$ 224 input size, 16 $\times$ 16 patch size, 192 patch embedding size, 12 layers, 3 heads, and 192 output embedding size. This model has 5.7M learnable parameters.

**TABLE 1.** Input size, output size, total learnable parameters, and FLOPs of used models.

| Model | Input size | Output size | Parameters (M) | FLOPs (G) |
|---|---|---|---|---|
| ViT (tiny) [8] | 224x224 | 1x192 | 5.7 | 1.258 |
| VGG16 [37] | 224x224 | 1x4096 | 138.4 | 15.47 |
| PiT [32] | 224x224 | 1x256 | 4.8 | 0.702 |
| MobileNetV2 [35] | 224x224 | 1x1280 | 3.5 | 0.334 |
| EfficientNetB1 [18] | 256x256 | 1x1280 | 7.8 | 0.616 |
| DeiT [36] | 224x224 | 1x192 | 5.7 | 1.258 |
| ResNet18 [17] | 224x224 | 1x512 | 11.6 | 1.826 |

*CIFAR10 dataset:* CIFAR10 dataset is used to evaluate many performance metrics such as accuracy by majority of literature [38]. This dataset was created by **Canadian Institute for Advanced Research** between 2008-2009 years. CIFAR10 dataset has 50000 train images and 10000 test images with 10 classess. These classes are airplane, automobile, bird, cat,deer, dog, frog horse, ship and truck. Each class has 5000 train and 1000 test images with 32 $\times$ 32x3 image size. Fig. 8 shows some example images for CIFAR10 dataset.

*CIFAR 100 dataset:* CIFAR100 dataset [38] was also created by **Canadian Institute for Advanced Research** and has similar properties to CIFAR10. There are 100 class in CIFAR100 and each class has 500 train and 100 test image which means there are 50000 train images and 10000 test images. Each image has 32 $\times$ 32x3 image size. CIFAR100 examples are illustrated in Figure 9.

*Caltech256 dataset:* Caltech256 dataset is introduced by **California Institute of Technology** in 2022 [39]. Caltech256 dataet has 256 classes with 30607 images. There is no train or test separation in Caltech256 dataset. Images do not have fixed size and fixed image channel. We split the dataset into 30000 and 5607 images for train and test dataset, respectively. There are example images in Figure 10.

### B. EVALUATION METRICS

In classification tasks, many metrics can be used to evaluate the machine learning models' performance. Some of the well known performance metrics are accuracy, precision, recall, and f1-score. Accuracy evaluates whether model is capable to predict correct classes. However, accuracy may mislead especially on unbalanced datasets. Precision measures how many model's true positives are correct prediction. Recall detects how much model is good to predict true positives. In this study, f1-score metric is use. F1-score is harmonic average of the precision and recall values. Morever, using f1-score in classification tasks is more objective rather than using only accuracy, precision, or recall.

#### 1) PRECISION

Precision is a measure of how many of the predicted positive instances are actually correct. In classification tasks, it is defined as in 10. It evaluates the accuracy of positive predictions, making it particularly useful in cases where false positives are costly. We used macro precision value.

$$P = \frac{TP}{TP + FP} \tag{10}$$

#### 2) RECALL

Recall measures how many of the actual positive instances were correctly predicted. It is defined as in 11. It indicates how well the model captures all the positive cases, making it especially important in scenarios where missing a positive instance is costly, such as in medical diagnoses. We used macro recall value.

$$R = \frac{TP}{TP + FN} \tag{11}$$

#### 3) F1-SCORE

F1-score is acquired by dividing two times True Positive (TP) into sum of TP, False Positive (FP) and False Negative (FN) values. Its mathematical equation was given in Eq. 12. We used macro f1-score.

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{12}$$

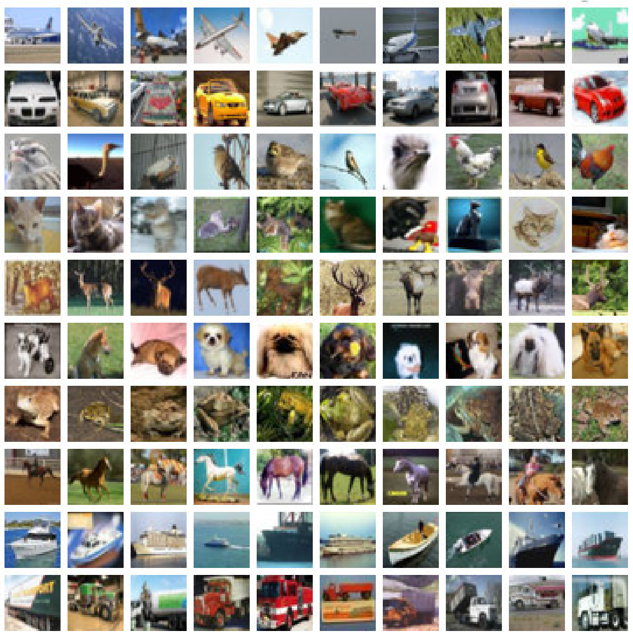
FIGURE 8. Example images from CIFAR10 dataset.

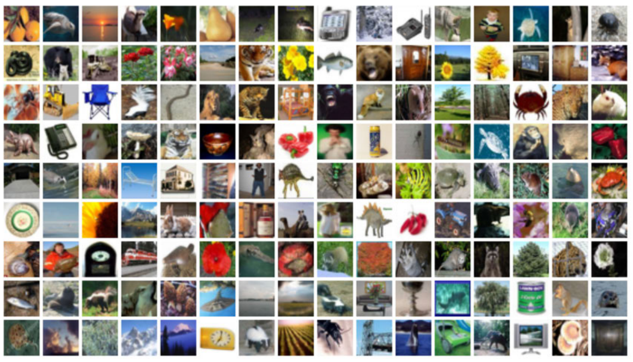
FIGURE 9. Example images from CIFAR100 dataset.

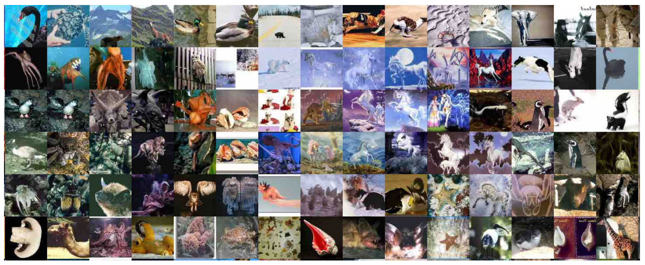
FIGURE 10. Example images from Caltech256 dataset.

### C. RESULTS

In this paper, we compared the robustness of seven models based on Transformer and CNN architectures. We used vanilla ViT, PiT, and DeiT for Transformer-based classification algorithms, and VGG16, ResNet18, MobileNetV2, and EfficientNetB1 for CNN-based algorithms.

To evaluate the models' performances, we used three publicly available datasets: CIFAR10, CIFAR100, and Caltech256. We used the f1-score metric to compare model performances, as it is more robust to the class imbalance problem, such as in Caltech256, and is widely utilized in many papers. The obtained f1-scores are shared in Table 2.

Each row in the first column represents the train data embeddings used to train the SVC. For example, **"raw + 50 degrees"** means that the SVC is trained with a concatenation of the original train data embeddings and 50 degrees rotated train data embeddings. Each row in the second column indicates which model is used to extract embeddings from the dataset.

Starting from the second column, each row each column (except for the **"average"** column and row) display which test dataset is used to evaluate the models. For instance, the **"50 degrees"** column represents that the trained SVC is tested using 50 degrees rotated data embeddings. The **"average"** column shows the performance mean of the same training data but with rotated test datasets across the models. For example, the first row and **"average"** column indicates that the models are trained on raw data and tested on the raw dataset, 10 degrees rotated datasets, 50 degrees rotated datasets, and 90 degrees rotated datasets. The "average" column is the mean of all these test results.

The **"average"** row (excluding the last column) shows that the test dataset is the same, but the training dataset is different. For example, the first column of **"average"** row means that models are trained on different datasets, like the raw dataset and the raw + 50-degree rotated dataset, and tested on just the raw dataset.

We colored cells of Table 2 to blue and red colors to indicate maximum and minimum values, respectively. These values are calculated with respect to each column and are separate for all used training datasets.

Table 2 shows that when we train the classifier with original train dataset embeddings and test on original and rotated datasets, the PiT model achieves the best average f1-scores of 0.59976 and 0.74515 for CIFAR100 and CIFAR10 datasets, respectively. On the other hand, EfficientNetB1 model achieves the best average f1-score 0.67576 for Caltech256 dataset. However, EfficientNetB1 model exhibit its the worst performance on CIFAR100 and CIFAR10 datasets. MobileNetV2 and ResNet18 outperform other CNN models for CIFAR100 and CIFAR10 datasets, respectively. There are 12.928%, 6.808%, and 7.687% performance reduction between the best Transformer-based model (PiT) and the best CNN-based models (MobileNetV2 and ResNet18 for CIFAR10, EfficientNetB1 for Caltech256) for CIFAR100, CIFAR10 and Caltech256 datasets, respectively. However, when the classifier is trained with original train dataset and tested with the original test dataset, this performance differences are 10.165%, 2.320%, and 8.742% for CIFAR100, CIFAR10, and Caltech256 respectively. It is seen that when test images are rotated, performance reduction is increased between PiT, MobileNetV2, and ResNet18 for CIFAR100 and CIFAR10 datasets. On the other hand, performance differences are decreased between PiT and EfficientNetB1 from 8.742% to 7.684%. It means that as rotation degrees increase, Transformer-based models perform better than CNN-based models.

Table 2 highlight following inferences for Transformer and CNN-based algorithms, respectively.

When we consider only Transformer-based models:

- PiT model achieves 0.67481 best f1-score for CIFAR100 dataset when it is trained with 50 + 90 degrees rotated dataset.
- PiT model achieves 0.84772 best f1-score for CIFAR10 dataset when it is trained with 10 + 50 degrees rotated dataset.
- ViT model achieves 0.68722 best f1-score for Caltech256 dataset when it is trained with 10 + 50 degrees rotated dataset.
- The best models achieve average f1-scores of 0.59976, 0.74515, and 0.62136 when they are trained on just original datasets for CIFAR100, CIFAR10, and Caltech256, respectively.

When we consider only CNN-based models:

- MobileNetV2 model achieves the best f1-score 0.57696 for CIFAR100 dataset when it is trained with raw data + 50 degrees rotated dataset.
- ResNet18 model achieves best f1-score 0.80511 for CIFAR10 dataset when it is trained with 50 + 90 degrees rotated dataset.
- EfficientNetB1 model achieves best f1-score 0.76369 for Caltech256 dataset when it is trained with 50 + 90 degrees rotated dataset.
- The best CNN models achieve f1-scores of 0.52222, 0.69429, and 0.67576 when trained on just the original datasets for CIFAR100, CIFAR10, and Caltech256, respectively.

Results show training the classifier with concatenation of different rotated dataset embeddings increase the performance of all models. However, even in this case, CNN-based models cannot outperform Transformer-based models except EfficientNetB1 for only Caltech256 dataset. All the best results were obtained from experiments that included 50 degrees rotated train images in the classifier's training dataset. This suggests that to make the developed algorithm more robust to rotation, incorporating 10 and 50 degrees rotated images into the dataset is beneficial.

The reason behind this may be that most of the algorithms struggle to classify 50 degrees rotated images. Most models find it more challenging to classify 50 degree rotated images compared to other rotation degrees.

Table 1 FLOPs column indicates floating point operations value that represents the complexity of the model calculation. Higher FLOPs means that the model needs high computational resources. VGG16 and MobileNetV2 models are the most resource-inefficient and resource-efficient models, respectively. ViT and DeiT have the highest FLOPs value compared to other transformer-based models. On the other hand, PiT has the least FLOPs value. There are 2.10 times difference between the most efficient Transformer -PiT- and CNN-based models -MobileNetV2- in terms of FLOPs. PiT has 1.37 times more learnable parameters than MobileNetV2. Table 2 also shows that MobileNetV2 exhibit worse performance than PiT model. ResNet18 is the most close model to PiT in terms of overall f1-score metric. ResNet18 has 2.6 times more FLOPs and 2.41 times more learnable parameters than PiT model. This show that the best CNN model can not outperform the best Transformer-based model

in terms of FLOPs. PiT has an exceptional model that should be pointed out. Although the number of parameters is close to each other, the FLOPs value is relatively lower than ViT. The main reason for this is the architecture of the PiT model.

Although the VGG16 models have more parameters than other CNN and Transformer-based models, it exhibit lower performance scores compared to other models. This inference highlights that model architecture is as important as model parameter size in improving performance and robustness to transformations such as rotation.

The PiT algorithm, which outperforms many models in most experiments, is a hybrid algorithm that combines CNN and Transformer architectures. This indicates that adding convolution layers helps the model learn more representative features and makes it more robust to image rotation perturbations.

## V. DISCUSSION

A representative dataset plays a crucial role in achieving desired results in machine learning. Training models with poorly representative data can prevent them from recognizing images in challenging conditions such as rotation and blurring. This can be particularly harmful in critical systems such as healthcare, where incorrect model outputs can have serious consequences. Often, results under these challenging conditions cannot be reversed. We investigated rotation robustness of two different architecture: CNN and Transformer. Observed results reveal that Transformer-based algorithms are more robust to rotation operations than CNN-based algorithms for CIFAR100 and CIFAR10 datasets. Although Transformer and CNN-based algorithms performed similarly, CNN-based algorithms are not able to outperform Transformed based algorithms in the most of the experiments.

Improving the quality and quantity of training data is essential if the dataset does not adequately represent the problem like in rotated image case. However, collecting images that reflect these challenging conditions is not always feasible due to unique situations or complex processes. Techniques such as data augmentation and oversampling are well-known strategies to enhance data quality and simulate the real-world scenarios. Nonetheless, these techniques may lead to longer processing times and increased hardware requirements. We added rotated images into classifier dataset instead of deep learning dataset to improve robustness of used model and mitigate hardware and computation requirements. These datasets helped all models to exhibit more robust results to rotation operation. However, even in this case, most of the Transformer-based algorithms outperform CNN-based algorithms.

We used CNN and Transformer-based seven different algorithms. CNN and Transformer-based architectures are different from each other. Even Transformer and CNN-based models differ in itself architectures. Classification performance is directly related to model's learnable parameters and architecture. It is desired to well-designed and big models perform best results. Lack of one of them may harm the performance. For example VGG16 and MobileNetV2 models both are CNN-based algorithm. Even VGG16 has near 40 times more learnable parameters than MobileNetV2, it cannot outperform MobileNetV2 model in the most of experiments. MobileNetV2 model has more modern layers such as normalization, residual, dropout, and $1 \times 1$ CNN layers. On the other hand, the VGG16 model has only CNN layers without regularization or normalization layers. Lack of modern layers may cause VGG16 to overfit or underfit the data. On the other hand, Transformer-based models outperform all CNN-based algorithms in many experiments even they have lesser learnable parameters. ViT and PiT algorithm has so similar architecture except PiT's convolution layers in patch embedding stage. PiT is a hybrid algorithm that is combination of CNN and Transformer architectures. PiT algorithm performs better results compared to other Transformer or CNN-based algorithms. It indicates that convolution layers are also beneficial to learn more representative embeddings of the dataset. This result encourages developing CNN and Transformer-based hybrid models due to obtain better performances. Transformer-based classification algorithms benefit from global information of the images. On the other hand, CNN-based models use local information to classify the images. Global information is expected to be more robust to local changes, such as image rotation. However, local features such as edge, corners, and so on are more prone to change image rotation since CNN extract embeddings pixel-based. Transformer and CNN hybrid models include both local and global embeddings from the image. This help model have more representative embedding vector.

Training used classifier with dataset that is rotated in many different angles. Each rotation degrees contribute to make model better. However, it is crucial to find most optimum rotation degrees due to obtain best results. Examined experiments show that training the classifier with 50 degrees rotated embeddings makes model to perform better results in terms of f1-score. We think the reason behind that is somehow all models suffer to recognize 50 degrees rotated images.

Dataset characteristics are an important factor to achieve desired results. We used three different datasets in all experiments. Table 2 show that even we use same model, there are large and small margins between performances with respect to used different datasets. For example, 2 avearage column clearly reval that while there are larger margin between ViT and VGG16 for CIFAR100 and Caltech256, the margin is smaller for CIFAR10 dataset. There may any reason behind this situation like class numbers, rotated images in datasets or other statistical properties. In next studies, this phenomena must be evaluated too.

In this study, we evaluated the rotation sensitivity of seven models, including both CNNs and Transformer-based models. We used vanilla ViT, PiT, and DeiT for the Transformer-based classifiers, and VGG16, ResNet18, MobileNetV2, and EfficientNetB1 for the CNN-based classifiers. It is well known that conventional CNN methods perform well

**TABLE 2.** F1-score performance of all models with respect to rotation degrees. First column indicate which dataset used to trian SVC. Used model name is displayed in second column. raw data, 10, 50, and 90 degree columns represent used test dataset and how rotation degrees are performed on it.

| Train data | Model | Test data | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | raw data | | | 10 degree | | | 50 degree | | | 90 degree | | | average | | |
| | | cifar100 | cifar10 | caltech256 | cifar100 | cifar10 | caltech256 | cifar100 | cifar10 | caltech256 | cifar100 | cifar10 | caltech256 | cifar100 | cifar10 | caltech256 |
| raw data | ViT | 0,82724 | 0,95888 | 0,80488 | 0,72949 | 0,90507 | 0,74168 | 0,31858 | 0,49383 | 0,47043 | 0,42155 | 0,46225 | 0,46843 | 0,57422 | 0,70501 | 0,62136 |
| | VGG16 | **0,66570** | **0,91909** | **0,72109** | 0,59706 | 0,87273 | **0,62573** | 0,12579 | 0,39438 | **0,22178** | **0,23099** | **0,59148** | **0,27964** | 0,40488 | 0,69442 | **0,46206** |
| | ResNet18 | 0,69910 | 0,93511 | 0,73202 | 0,65940 | 0,88761 | 0,68961 | 0,27931 | 0,50857 | 0,32203 | 0,34729 | 0,44586 | 0,41975 | 0,49628 | 0,69429 | 0,54085 |
| | PiT | 0,81886 | 0,95732 | 0,79539 | **0,77775** | **0,94262** | 0,74615 | **0,41146** | **0,62841** | **0,50995** | 0,39097 | 0,45222 | 0,44376 | **0,59976** | **0,74515** | 0,62381 |
| | MobileNet | 0,73562 | 0,94936 | 0,75759 | 0,68464 | 0,78517 | 0,72179 | 0,29717 | 0,22135 | 0,37350 | 0,37144 | 0,44460 | 0,46202 | 0,52222 | 0,60012 | 0,57872 |
| | EfficientNetB1 | **0,84840** | **0,96868** | **0,87159** | **0,30479** | **0,15230** | **0,78988** | **0,00289** | **0,02005** | 0,48708 | **0,43969** | 0,48085 | **0,55448** | **0,39894** | **0,40547** | **0,67576** |
| | DeiT | 0,80423 | 0,94991 | 0,78240 | 0,73468 | 0,91408 | 0,74244 | 0,35130 | 0,59702 | 0,48007 | 0,38341 | **0,44003** | 0,42264 | 0,56841 | 0,72526 | 0,60689 |
| | | | | | | | | | | | | | | | | |
| raw data + 10 degree rotated | ViT | 0,82287 | 0,95736 | 0,81126 | 0,77550 | 0,93455 | 0,77320 | 0,37501 | 0,56877 | 0,51834 | 0,43253 | 0,48781 | 0,48178 | 0,60148 | 0,73712 | 0,64614 |
| | VGG16 | **0,66275** | **0,91379** | **0,71743** | 0,62215 | 0,88728 | **0,66857** | 0,14209 | 0,41974 | **0,24686** | **0,23715** | **0,60124** | **0,28964** | **0,41603** | 0,70551 | **0,48063** |
| | ResNet18 | 0,70776 | 0,93395 | 0,74651 | 0,68174 | 0,90878 | 0,70814 | 0,29426 | 0,52654 | 0,34978 | 0,35320 | 0,46267 | 0,43459 | 0,50924 | 0,70798 | 0,55976 |
| | PiT | 0,81581 | 0,95809 | 0,79714 | **0,78921** | **0,94662** | 0,76255 | **0,43460** | 0,64091 | 0,52814 | 0,40224 | 0,45637 | 0,45248 | **0,61047** | 0,75050 | 0,63508 |
| | MobileNet | 0,73959 | 0,94825 | 0,76185 | 0,70159 | 0,89075 | 0,74023 | 0,31287 | 0,31297 | 0,38595 | 0,37675 | **0,45604** | 0,46869 | 0,53270 | 0,65200 | 0,58918 |
| | EfficientNetB1 | **0,84272** | **0,96727** | **0,86976** | **0,58474** | **0,68152** | **0,82635** | **0,00359** | **0,01987** | **0,54755** | **0,44539** | 0,47765 | **0,57461** | 0,46911 | **0,53658** | **0,70457** |
| | DeiT | 0,80386 | 0,95058 | 0,78324 | 0,76228 | 0,93187 | 0,75171 | 0,38953 | **0,65693** | 0,50498 | 0,39113 | 0,46275 | 0,43354 | 0,58670 | **0,75053** | 0,61837 |
| | | | | | | | | | | | | | | | | |
| raw data + 50 degree rotated | ViT | 0,81465 | **0,95675** | 0,80296 | 0,74545 | 0,92425 | 0,75432 | 0,58130 | 0,79827 | 0,62370 | **0,49631** | 0,60779 | 0,57818 | 0,65943 | 0,82177 | 0,68979 |
| | VGG16 | **0,64733** | 0,91178 | **0,70512** | 0,58511 | 0,87648 | **0,62118** | 0,37025 | 0,68336 | **0,47120** | **0,25453** | 0,65273 | **0,37486** | 0,46430 | 0,78109 | **0,54309** |
| | ResNet18 | 0,69051 | 0,93032 | 0,72775 | 0,65698 | 0,88851 | 0,68685 | 0,44668 | 0,75244 | 0,49077 | 0,39475 | 0,55437 | 0,48786 | 0,54723 | 0,78141 | 0,59831 |
| | PiT | 0,80992 | 0,95159 | 0,79107 | **0,78204** | **0,94039** | 0,75088 | **0,60293** | **0,83535** | 0,62336 | 0,49288 | **0,66376** | 0,55643 | **0,67194** | **0,84778** | 0,68044 |
| | MobileNet | 0,71534 | 0,94737 | 0,75320 | 0,67369 | 0,79901 | 0,72721 | 0,49329 | 0,70179 | 0,53925 | 0,42552 | 0,48604 | 0,53748 | 0,57696 | 0,73355 | 0,63928 |
| | EfficientNetB1 | **0,84123** | **0,01818** | **0,86709** | **0,28398** | **0,01818** | **0,81676** | **0,08226** | **0,15361** | **0,69064** | 0,44058 | **0,01818** | **0,67407** | **0,41201** | **0,05204** | **0,76214** |
| | DeiT | 0,79599 | 0,94844 | 0,77798 | 0,74450 | 0,92634 | 0,74292 | 0,56791 | 0,80490 | 0,60125 | 0,46281 | 0,59509 | 0,51526 | 0,64280 | 0,81869 | 0,65935 |
| | | | | | | | | | | | | | | | | |
| raw data + 90 degree rotated | ViT | 0,81304 | 0,95291 | 0,80541 | 0,71703 | 0,89396 | 0,73425 | 0,39932 | 0,57245 | 0,56058 | 0,62649 | 0,81273 | 0,64051 | 0,63897 | 0,80801 | 0,68519 |
| | VGG16 | **0,63410** | **0,90917** | **0,69459** | 0,56220 | 0,86959 | **0,58326** | 0,14714 | 0,44185 | **0,27833** | **0,42661** | 0,78097 | **0,52167** | **0,44251** | 0,75040 | **0,51946** |
| | ResNet18 | 0,68309 | 0,92505 | 0,72532 | 0,64380 | 0,86710 | 0,67689 | 0,31790 | 0,57311 | 0,37303 | 0,50483 | **0,75442** | 0,57475 | 0,53740 | 0,77992 | 0,58750 |
| | PiT | 0,80539 | 0,95032 | 0,78786 | **0,76661** | **0,93403** | 0,74135 | **0,50018** | **0,73692** | 0,58755 | 0,61026 | 0,82171 | 0,61298 | **0,67061** | **0,86075** | 0,68243 |
| | MobileNet | 0,71770 | 0,94119 | 0,74350 | 0,66473 | 0,75764 | 0,71931 | 0,34219 | 0,26836 | 0,42056 | 0,55350 | 0,80659 | 0,62609 | 0,56953 | 0,69345 | 0,62736 |
| | EfficientNetB1 | **0,83862** | **0,96426** | **0,87022** | **0,29464** | **0,15529** | **0,79479** | **0,00196** | **0,03786** | **0,58899** | **0,66981** | **0,85781** | **0,75652** | 0,45126 | **0,50381** | **0,75263** |
| | DeiT | 0,79338 | 0,94172 | 0,77020 | 0,72158 | 0,90064 | 0,73122 | 0,42347 | 0,68317 | 0,55038 | 0,58804 | 0,80227 | 0,58872 | 0,63162 | 0,83195 | 0,66013 |
| | | | | | | | | | | | | | | | | |
| 10 + 50 degree rotated | ViT | 0,80659 | 0,94699 | 0,79581 | 0,77126 | 0,93572 | 0,76406 | 0,58157 | 0,79676 | 0,62223 | **0,48502** | 0,59268 | 0,56680 | 0,66111 | 0,81804 | 0,68722 |
| | VGG16 | **0,63222** | 0,90475 | **0,66451** | 0,61669 | 0,89036 | **0,66095** | 0,36942 | 0,69112 | **0,47903** | **0,25622** | 0,64917 | **0,33997** | 0,46864 | 0,78385 | **0,53612** |
| | ResNet18 | 0,68110 | 0,92235 | 0,72668 | 0,66468 | 0,91012 | 0,69327 | 0,44651 | 0,75282 | 0,49319 | 0,39031 | 0,56327 | 0,48324 | 0,54565 | 0,78714 | 0,59909 |
| | PiT | **0,81050** | **0,95317** | 0,78833 | **0,78889** | **0,94440** | 0,76300 | **0,60166** | **0,83459** | 0,62107 | 0,48490 | **0,65872** | 0,54927 | **0,67149** | **0,84772** | 0,68042 |
| | MobileNet | 0,71032 | 0,92344 | 0,74409 | 0,68423 | 0,89357 | 0,72753 | 0,49001 | 0,70640 | 0,53916 | 0,42243 | 0,48441 | 0,53343 | 0,57675 | 0,75196 | 0,63605 |
| | EfficientNetB1 | 0,73007 | **0,01818** | **0,85092** | **0,57184** | **0,01818** | **0,82282** | **0,08235** | **0,15432** | **0,68872** | 0,37984 | **0,01818** | **0,65998** | **0,44103** | **0,05222** | **0,75561** |
| | DeiT | 0,79261 | 0,94324 | 0,78087 | 0,76333 | 0,93201 | 0,75201 | 0,56658 | 0,80316 | 0,59801 | 0,45229 | 0,59903 | 0,51146 | 0,64370 | 0,81936 | 0,66059 |
| | | | | | | | | | | | | | | | | |
| 10 + 90 degree rotated | ViT | **0,80997** | 0,94961 | 0,80094 | 0,76316 | 0,92960 | 0,75850 | 0,42537 | 0,61206 | 0,57885 | 0,61971 | 0,81077 | 0,63631 | 0,65456 | 0,82551 | 0,69365 |
| | VGG16 | **0,62904** | 0,90811 | **0,67629** | 0,60565 | 0,88609 | **0,65243** | 0,15291 | 0,45651 | **0,29609** | **0,41556** | 0,77978 | **0,51163** | **0,45079** | 0,75762 | **0,53411** |
| | ResNet18 | 0,67456 | 0,91629 | 0,72211 | 0,65806 | 0,90035 | 0,68634 | 0,32852 | 0,58155 | 0,38135 | 0,50486 | **0,75398** | 0,57763 | 0,54150 | 0,78805 | 0,59186 |
| | PiT | 0,80803 | **0,95268** | 0,78703 | **0,78411** | **0,93878** | 0,75533 | **0,50687** | **0,74370** | 0,59498 | 0,60023 | 0,81952 | 0,60781 | **0,67481** | **0,86367** | 0,68628 |
| | MobileNet | 0,71460 | 0,91090 | 0,74233 | 0,67734 | 0,88916 | 0,72067 | 0,35215 | 0,31834 | 0,42796 | 0,54652 | 0,81315 | 0,62470 | 0,57265 | 0,73289 | 0,62891 |
| | EfficientNetB1 | 0,80393 | **0,80393** | **0,86118** | **0,57901** | **0,68041** | **0,81833** | **0,00365** | **0,02781** | **0,61739** | **0,66649** | **0,86006** | **0,75550** | 0,51327 | **0,59305** | **0,76310** |
| | DeiT | 0,79299 | 0,94217 | 0,77508 | 0,75371 | 0,92658 | 0,74580 | 0,43986 | 0,71454 | 0,55460 | 0,58006 | 0,80181 | 0,58251 | 0,64166 | 0,84628 | 0,66450 |
| | | | | | | | | | | | | | | | | |
| 50 + 90 degree rotated | ViT | 0,75319 | 0,90581 | 0,76933 | 0,69703 | 0,89065 | 0,71681 | 0,58226 | 0,79811 | 0,63441 | 0,63070 | 0,81780 | 0,64343 | 0,66580 | 0,85309 | 0,69100 |
| | VGG16 | **0,43640** | 0,80460 | **0,50567** | 0,43930 | 0,81671 | **0,49277** | 0,36947 | 0,70370 | **0,47384** | **0,43010** | 0,80641 | **0,52356** | **0,41881** | 0,78285 | **0,49896** |
| | ResNet18 | 0,56802 | 0,87755 | 0,63357 | 0,55424 | 0,82268 | 0,60858 | 0,44731 | 0,75604 | 0,49321 | 0,51529 | 0,76417 | 0,58025 | 0,52121 | 0,80511 | 0,57890 |
| | PiT | **0,75621** | 0,90809 | 0,75098 | **0,73528** | 0,89577 | 0,71672 | **0,61537** | **0,84412** | 0,62443 | 0,61650 | **0,82195** | 0,61865 | **0,68084** | **0,86748** | 0,67770 |
| | MobileNet | 0,59799 | **0,69561** | 0,66044 | 0,56436 | 0,65010 | 0,64698 | 0,50109 | 0,70673 | 0,53572 | 0,56194 | 0,81944 | 0,63140 | 0,55635 | 0,71797 | 0,61864 |
| | EfficientNetB1 | 0,73209 | 0,87008 | **0,82949** | **0,26194** | **0,01818** | **0,77665** | **0,08374** | **0,15182** | **0,69149** | **0,66854** | **0,01818** | **0,75710** | 0,43658 | **0,26457** | **0,76369** |
| | DeiT | 0,73827 | **0,91539** | 0,74790 | 0,69139 | **0,90044** | 0,71590 | 0,57705 | 0,80681 | 0,60640 | 0,59324 | 0,80762 | 0,59238 | 0,64999 | 0,85757 | 0,66564 |
| | | | | | | | | | | | | | | | | |
| average | ViT | **0,80679** | 0,94690 | 0,79865 | 0,74270 | 0,91626 | 0,74897 | 0,46620 | 0,66289 | 0,57265 | **0,53033** | 0,65597 | 0,57363 | 0,63651 | 0,79551 | 0,67348 |
| | VGG16 | **0,61536** | 0,89590 | **0,66924** | 0,57545 | 0,87132 | **0,61498** | 0,23958 | 0,54152 | **0,35245** | **0,32159** | **0,69454** | **0,40585** | **0,43800** | 0,75082 | **0,51063** |
| | ResNet18 | 0,67202 | 0,92009 | 0,71628 | 0,64556 | 0,88359 | 0,67853 | 0,36578 | 0,63587 | 0,41476 | 0,43008 | 0,61411 | 0,50830 | 0,52836 | 0,76341 | 0,57947 |
| | PiT | 0,80353 | **0,94732** | 0,78540 | **0,77484** | **0,93466** | 0,74800 | **0,52473** | **0,75200** | 0,58421 | 0,51400 | 0,67061 | 0,54877 | **0,65427** | **0,82615** | 0,66659 |
| | MobileNet | 0,70445 | 0,90230 | 0,73757 | 0,66437 | 0,80934 | 0,71482 | 0,39840 | 0,46228 | 0,46030 | 0,46544 | 0,61575 | 0,55483 | 0,55816 | 0,69742 | 0,61688 |
| | EfficientNetB1 | 0,80530 | **0,65866** | **0,86004** | **0,41156** | **0,24629** | **0,80651** | **0,03721** | **0,08076** | **0,61598** | 0,53005 | **0,39013** | **0,67604** | 0,44603 | **0,34396** | **0,73964** |
| | DeiT | 0,78876 | 0,94164 | 0,77395 | 0,73878 | 0,91885 | 0,74029 | 0,47367 | 0,72379 | 0,55652 | 0,49300 | 0,64409 | 0,52093 | 0,62355 | 0,80709 | 0,64792 |

on images with local features such as edges and corners. However, these local features can be lost when images are rotated, making it crucial to include rotated images in the training dataset. To mitigate the hardware and time costs, we added rotated image feature vectors to the classifier dataset rather than the deep learning model training dataset. Our results clearly show that these feature vectors improve the robustness of all models. In this study we evaluated the rotation sensitivity of CNN and Transformer-based six models. We used vanilla ViT, PiT, and DeiT as Transformer-based algorithms, and VGG16, ResNet18, MobileNetV2, and EfficientNetB1 for CNN-based algorithms. It is well known that conventional CNN-based methods performs well on images that have local features such as edges, corners and more complex patterns. Since these kinds of local features are lost when the images are rotated, it is crucial to include rotated images to train dataset. To mitigate the hardware and time costs, we added rotated image feature vectors to the classifier dataset rather than the deep learning model training dataset. Our results clearly show that these feature vectors improve the robustness of all models. We also found that despite CNN-based models, Transformer-based models are less sensitive to image rotation since they focus on global dependencies instead of local features. We also observed that training the classifier with 10 and 50 degree rotated train dataset achieves best score in the most of experiments.

## VI. CONCLUSION

In this paper, we aimed to evaluate image rotation robustness of two different image classification architecture that are Transformer and CNN. We used Transformer-based vanilla ViT, PiT, and DeiT models. For CNN architecture, we used VGG16, ResNet18, MobileNetV2, and EfficientNetB1 models. We use three publicly available datasets that are CIFAR100, CIFAR10, and Caltech256. We trained all models with train datasets and extracted embeddings of original and rotated datasets. An SVC is trained with embeddings of original and rotated train dataset and performances evaluated on original and rotated test datasets. We used 10, 50 and 90 degree left rotation angles. Results show that

Transformer-based classification algorithms performs better results compared to CNN-based algorithms.

We saw that when models are trained with original dataset and tested on original test datasets, performance of all models were decreased. Especially most models fail when test datasets are rotated 50 degrees. Concatenating original and rotated train dataset embeddings increase performance of all deep learning models. Results show that all Transformer-based algorithms exhibit better performance than CNN-based algorithms. In the most of experiments, Transformer-based PiT algorithm achieves qualitative and quantitative better f1-scores than other Transformer and CNN-based algorithms. Generally, MobileNetV2 and EfficientNetB1 model performs better than other CNN-based models. We saw that training SVC with 50 degrees rotated form of train dataset usually achieves better scores for many models. Results also revealed that although VGG16 model has more learnable parameters than other models, in most of the experiments it achieves worse performance which show that model architecture is also as much important as learnable parameters. In following studies, impact of other perturbations such as image blurring, noising or changing brightness may be investigated. More experiments can be done to identify which structure makes more robust Transformer-based image classification algorithms than CNN-based classification algorithms. PiT that is a CNN and Transformer-based hybrid classification algorithm performs well unlike other CNN-based algorithms. This show that using Transformer and Hybrid architecture together may increase performance and robustness to perturbations such as image rotations.

## CONFLICTS OF INTEREST

The authors declared that there is no conflict of interest.

## DATA AVAILABILITY

The CIFAR10 dataset is publicly available at
https://www.cs.toronto.edu/ kriz/cifar.html
The CIFAR100 dataset is publicly available at
https://www.cs.toronto.edu/ kriz/cifar.html
The CIFAR100 dataset is publicly available at
https://data.caltech.edu/records/nyy15-4j048

## REFERENCES

[1] G. Wang, J. C. Ye, and B. De Man, "Deep learning for tomographic image reconstruction," *Nature Mach. Intell.*, vol. 2, no. 12, pp. 737–748, Dec. 2020.

[2] Y. Kortli, M. Jridi, A. Al Falou, and M. Atri, "Face recognition systems: A survey," *Sensors*, vol. 20, no. 2, p. 342, Jan. 2020.

[3] I. V. Pustokhina, D. A. Pustokhin, J. J. P. C. Rodrigues, D. Gupta, A. Khanna, K. Shankar, C. Seo, and G. P. Joshi, "Automatic vehicle license plate recognition using optimal K-means with convolutional neural network for intelligent transportation systems," *IEEE Access*, vol. 8, pp. 92907–92917, 2020.

[4] E. Bertelli, L. Mercatelli, C. Marzi, E. Pachetti, M. Baccini, A. Barucci, S. Colantonio, L. Gherardini, L. Lattavo, M. A. Pascali, S. Agostini, and V. Miele, "Machine and deep learning prediction of prostate cancer aggressiveness using multiparametric MRI," *Frontiers Oncol.*, vol. 11, Jan. 2022, Art. no. 802964.

[5] F. B. Tek, "Mitosis detection using generic features and an ensemble of cascade adaboosts," *J. Pathol. Informat.*, vol. 4, no. 1, p. 12, Jan. 2013.

[6] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.

[7] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16×16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 6000–6010.

[10] X.-X. Niu and C. Y. Suen, "A novel hybrid CNN–SVM classifier for recognizing handwritten digits," *Pattern Recognit.*, vol. 45, no. 4, pp. 1318–1325, Apr. 2012.

[11] X. Sun, L. Liu, C. Li, J. Yin, J. Zhao, and W. Si, "Classification for remote sensing data with improved CNN-SVM method," *IEEE Access*, vol. 7, pp. 164507–164516, 2019.

[12] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. 13th Eur. Conf.*, Zurich, Switzerland. Cham, Switzerland: Springer, Sep. 2014, pp. 818–833.

[13] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[14] Y. Tang, "Deep learning using linear support vector machines," 2013, *arXiv:1306.0239*.

[15] D. Justus, J. Brennan, S. Bonner, and A. S. McGough, "Predicting the computational cost of deep learning models," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2018, pp. 3873–3882.

[16] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," 2020, *arXiv:2007.05558*.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[18] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[19] Z. Tianyu, M. Zhenjiang, and Z. Jianhu, "Combining CNN with hand-crafted features for image classification," in *Proc. 14th IEEE Int. Conf. Signal Process. (ICSP)*, Aug. 2018, pp. 554–557.

[20] E. Osherov and M. Lindenbaum, "Increasing CNN robustness to occlusions by reducing filter support," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 550–561.

[21] C. Pirlot, R. C. Gerum, C. Efird, J. Zylberberg, and A. Fyshe, "Improving the accuracy and robustness of CNNs using a deep CCA neural data regularizer," 2022, *arXiv:2209.02582*.

[22] R. R. Yedla and S. R. Dubey, "On the performance of convolutional neural networks under high and low frequency information," in *Proc. 5th Int. Conf.*, Prayagraj, India. Cham, Switzerland: Springer, Dec. 2021, pp. 214–224.

[23] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," 2017, *arXiv:1710.09412*.

[24] M. Alkaddour and U. Tariq, "Investigating the effect of noise on facial expression recognition," in *Proc. Comput. Vis. Conf. (CVC)*, vol. 2. Cham, Switzerland: Springer, 2020, pp. 699–709.

[25] K. De and M. Pedersen, "Impact of colour on robustness of deep neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2021, pp. 21–30.

[26] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "ImageNet-trained CNNs are biased towards texture; Increasing shape bias improves accuracy and robustness," 2018, *arXiv:1811.12231*.

[27] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.

[28] M. Z. Joel, S. Umrao, E. Chang, R. Choi, D. X. Yang, J. S. Duncan, A. Omuro, R. Herbst, H. M. Krumholz, and S. Aneja, "Using adversarial images to assess the robustness of deep learning models trained on diagnostic images in oncology," *JCO Clin. Cancer Informat.*, vol. 6, May 2022, Art. no. e2100170.

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[30] Y.-Y. Yang, C. Rashtchian, H. Zhang, R. R. Salakhutdinov, and K. Chaudhuri, "A closer look at accuracy vs. robustness," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp> 8588–8601.

[31] S. Bhojanapalli, A. Chakrabarti, D. Glasner, D. Li, T. Unterthiner, and A. Veit, "Understanding robustness of transformers for image classification," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 10231–10241.

[32] B. Heo, S. Yun, D. Han, S. Chun, J. Choe, and S. J. Oh, "Rethinking spatial dimensions of vision transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 11936–11945.

[33] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

[34] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.

[35] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[36] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 10347–10357.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[38] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2009, pp. 1–60, vol. 1, no. 1.

[39] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Inst. Technol. Pasadena, CA, USA, Tech. Rep., 2007.

**MUHAMMED CIHAD ARSLANOGLU** received the B.Sc. and M.Sc. degrees in electrical and electronics engineering from Dicle University, where he is currently pursuing the Ph.D. degree in electrical and electronics engineering. He has been engaged in programming since high school and developed various applications in image processing, artificial intelligence, and web technologies during his undergraduate studies. He has three years of full-time experience as an Artificial Intelligence and Computer Vision Specialist at Dicle Electricity Distribution Company. He is also a System Administrator with the Information Technology Department, Dicle University.

**ABDULKADIR ALBAYRAK** received the bachelor's, master's, and Ph.D. degrees in computer engineering and developed algorithms based on traditional and deep learning methods for the classification of biomedical images and the detection of cellular structures (cancerous or normal cells) within these images for his theses. He completed his postdoctoral training with the Department of Laboratory Medicine and Pathology, Mayo Clinic, and is currently with the Generative AI Division, Mayo Clinic. He has published many research articles in SCI-indexed journals and has presented papers at various conferences. He achieved second place worldwide in the competition titled "Mitosis Detection in Histopathological Images" organized by the International Conference Pattern Recognition (ICPR).

**HUSEYIN ACAR** received the B.Sc. degree in electronics engineering from Bursa Uludag University, Bursa, Türkiye, in 2006, and the M.Sc. and Ph.D. degrees in electrical-electronics engineering from Dicle University, Diyarbakır, Türkiye, in 2010 and 2020, respectively. He is currently an Assistant Professor with the Department of Electrical-Electronics Engineering, Dicle University. His research interests include machine learning, image processing, remote sensing, and embedded systems.