JAISCR, 2023, Vol. 13, No. 1, pp. 51 – 74
10.2478/jaiscr-2023-0004

FEATURE MAP AUGMENTATION TO IMPROVE SCALE INVARIANCE IN CONVOLUTIONAL NEURAL NETWORKS

Dinesh Kumar[1,*], Dharmendra Sharma[2]

[1]*School of Technology, Engineering, Mathematics and Physics, University of the South Pacific, Laucala Bay Road, Suva, Fiji*

[2]*Faculty of Science and Technology, University of Canberra, Canberra, ACT, 2617, Australia*

[*]*E-mail: dinesh.i.kumar@usp.ac.fj*

Submitted: 21st February 2022; Accepted: 19th October 2022

Abstract

Introducing variation in the training dataset through data augmentation has been a popular technique to make Convolutional Neural Networks (CNNs) spatially invariant but leads to increased dataset volume and computation cost. Instead of data augmentation, augmentation of feature maps is proposed to introduce variations in the features extracted by a CNN. To achieve this, a rotation transformer layer called Rotation Invariance Transformer (RiT) is developed, which applies rotation transformation to augment CNN features. The RiT layer can be used to augment output features from any convolution layer within a CNN. However, its maximum effectiveness is shown when placed at the output end of final convolution layer. We test RiT in the application of scale-invariance where we attempt to classify scaled images from benchmark datasets. Our results show promising improvements in the networks ability to be scale invariant whilst keeping the model computation cost low.

Keywords: Convolutional Neural Network, Feature Map Augmentation, Global Features, Scale-Invariant, Vision System

1 Introduction

The human vision system can easily filter information from the environment, and has the ability to recognise objects despite their changes in size, orientation and position. Whilst physiological studies explain this phenomena as automatic invariance encoding within the ventral stream in the vision system [1], there is limited evidence to explain how invariance encoding happens within the ventral stream. Similarly, Convolutional Neural Networks (CNNs) remain limited in their ability to be a truly invariant class of algorithms despite having surpassed performance of the vision system in some targetted and specific tasks. Given the lack of theoretical and empirical evidence it therefore remains a difficult problem to make CNNs invariant to object detection or classification.

To solve the problem of invariance in CNNs, research work has mostly focused on finding solutions for individual invariance problems separately such as rotation invariance, translation invariance and scale invariance. Introducing variation in the training dataset through data augmentation has been a popular technique to make CNNs spatially invariant but this technique leads to increased dataset volume and computation cost of the models. Based on the understanding that CNNs employ a *local-to-global* feature extraction strategy, and that it is the high-level global features that are used for final prediction suggests that creating variations in these high-level global features instead of input data may be useful to make CNNs recognise variations in the object brought about by changes in their size, orientation or position in the scene. Though there is no physiological evidence to suggest a similar process happens in the ventral stream, the theory of data augmentation applied to feature maps in a CNN provides an opportunity to make CNNs more invariant to image classification and object detection. While we note there are many classes of invariance problems to solve such as rotation invariance, translation invariance and scale invariance, in this work we present the case study of the augmentation of feature for improving scale invariance in CNNs.

Motivated by the above opportunities, Kumar & Sharma [2] proposed a novel method that allows CNNs to improve scale invariance by augmentation of feature maps. It is achieved by creating a transformer layer called Rotation Invariance Transformer (RiT) that can be placed at the output end of a convolution layer. This method is initially trialled using only one form of augmentation for the feature maps - rotation since other forms of transformations such as scaling and translation would cause loss of feature *pixel* values from the feature maps. Unlike data augmentation methods, this technique does not involve any manipulation of input data. Furthermore, feature map augmentation is suggested in the deeper end of the CNN feature extractor pipeline, allowing the classifier to learn variations of the input data in terms of its transformed high-level global features. The end-to-end CNN model containing the RiT layer is called the Rotation Transformer Network (RTN). In [2], the authors tested the model on rotation invariance. We extend their work by demonstrating and assessing the application of the RiT layer for improving scale invariance in CNNs.

Extensive experiments are conducted to evaluate scale invariance performance of RTNs on widely used CNN architectures and benchmark datasets. The results are presented for LeNet5 ([3]), VGG-16 ([4]) and ResNet-18 [5] networks trained on CIFAR-10 ([6]), FMNIST ([7]), Tiny ImageNet ([8]) and ImageHoof ([9]) datasets. First, the datasets are trained on models LeNet5, VGG-16 and ResNet-18 models to establish benchmark results and for comparison. Then, RiT layer is added to these backbone CNNs and the networks are then retrained on the same datasets. Whilst the RiT is not trained within the networks due to having no trainable parameters, it plays the role of augmenting feature maps generated within the network. The effect of transforming feature maps (by applying rotations) on the network's ability to classify test images subjected to varying scales is studied and compared with the benchmark results. The experimental results show promising improvements in the networks ability to classify scaled images over benchmark networks used without requiring changes to the rest of the model architecture.

This paper aims to contribute to the body of knowledge towards finding effective solutions to classification of scaled images by:

1. showing the usefulness of augmenting feature maps as an alternative to data augmentation to solve scale invariance classification,
2. showing that augmentation of feature maps applied through transformation of rotation has benefits towards promoting scale invariance,
3. demonstrating through ablation studies that augmentation of feature maps from the deeper end of the CNN module are more effective towards scale invariance task then augmentation of feature maps from earlier layers of the CNN architecture, and
4. showing the effectiveness of the RTN model [2] towards scale invariance classification in addition to rotation invariance, through additional extensive experiments on differently sized datasets and benchmark models.

The rest of the paper is organised as follows: Section 2 reviews related work while Section 3 introduces the RTN model. Section 4 describes the experiment design and results are presented in Section 5. A summary of outcomes are provided in Section 6.

2 Background

We summarise below the main advances in research in vision systems providing the basis for the current research and the presented outcomes.

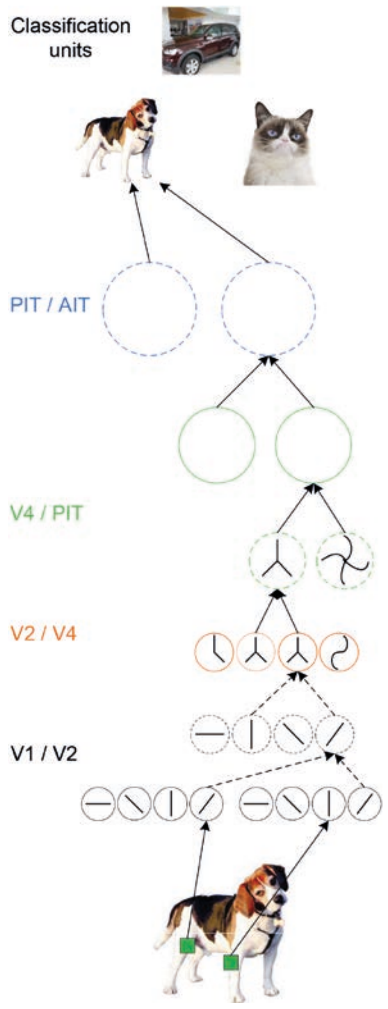


Figure 1. HMAX hierarchical model of the visual cortex ([10, 11]). Primary, secondary and quaternary visual areas are represented by acronyms V1, V2 and V4 (Secondary Visual Cortex) respectively. Posterior and anterior inferotemporal areas are represented by PIT and AIT respectively.

2.1 Information processing in the Vision System

The visual system employs a one-shot invariance encoding within the ventral system ([1]). The process of encoding object invariances begins as visual stimuli captured on the retina propagates via low-level cells to complex cells ([12, 13, 14, 15]).

CNNs model HMAX (Hierarchical MAX-pooling) hierarchical computational model of the visual cortex (see Figure 1)) through successive layers of convolution layers in the feature extractor part of the network, and thus, use the *local-to-global* feature extraction strategy to recognise objects. In CNNs, *global features* are represented as high-level features (complex features) generated by an aggregation of low-level features (local features) extracted in the early convolution layers of a CNN. At this stage in a CNN, global features tend to represent specific discriminative parts of an object or image that the network uses to identify objects during classification. These global features are then forward propagated to the classifier in the CNN for learning. This means CNNs only use features from the final convolution layer in the feature extractor. [16] further point out that high-level global features in the deeper layers of CNN are more discriminative for classification task and appropriate to characterize objects with complex characteristics; hence, useful for handling invariant object classification or detection.

Motivated by a) findings of physiological studies that invariance happens within the ventral stream, b) the property of high-level global features in the CNN to be more discriminative for classification task and appropriate to characterize objects with complex characteristic, and c) the potential benefits of the application of data augmentation, [2] proposed a novel method that allows CNNs to improve scale invariance by augmentation of feature maps. It is achieved by creating a transformer layer called Rotation Invariance Transformer (RiT) that can be placed at the output end of a convolution layer. This method is initially trialled using only one form of augmentation for the feature maps - rotation. In RiT, features are rotated by a given set of rotation parameters which are then passed to the next layer. Unlike data augmentation methods, this technique does not involve any manipulation of input data. Furthermore, feature map augmentation is suggested in the deeper end of the CNN feature extractor pipeline, allowing the classifier to learn variations of the input data in terms of its transformed high-level global features. The end-to-end CNN model containing the RiT layer is called the Rotation Transformer Network (RTN). In [2], the authors tested the model only on rotation invariance. As this paper focuses on scale-invariant image classification, we extend their work by demonstrating and assessing the application of the RiT layer for improving scale invariance in CNNs.

2.2 Augmentation methods to encode invariance in CNNs

One of the techniques used to improve on scale invariance that is relevant in this paper is the technique of augmentation. Various random transformations are applied on the input data which is then trained on the CNN. To make CNNs recognise scaled images is to train the model with multiple scales. This approach is referred to as *scale jittering* [17]. Initial reasons for the application of scale jittering for training CNN models has been to increase the volume of training samples in the train dataset in order to reduce overfitting and in some instances to overcome the problem of class imbalance. In addition, due to the increase in training samples potentially increases the chances of CNNs to recognise more scale-variant features, and thus, learn scale-invariant features.

However, this technique has predominantly been applied to augmentation of training data. This is achieved in two ways. One way is to create multiple instances of the training data prior to model training (*offline data augmentation*) such as in the work of [4, 18, 19]. Another way is to train the models with additional instances of the training data during model training (*online data augmentation*) such as in the work of [20, 21].

In general the application of data augmentation has shown promise in improving classification results [4]. Though good classification results are obtained using data augmentation, it is seldom practical particularly when working with large datasets. This technique also contradicts with how the natural vision system works which does not need to be exposed to variations of the same scene to learn invariance. Furthermore, explicit and conclusive work that tests the robustness of data augmentation on invariance problems is limited.

2.3 Augmentation of feature maps and filters

Manipulating feature maps in CNNs has been studied to some extent. For example, [22] propose augmentation of feature maps at intermediate layers of a CNN to regularise the network. In their work, the authors created augmentation layers (AugLayer) that can be dropped after convolution layers. In AugLayer, contrast and brightness adjustment augmentation are applied to outputs of some of the convolution layers of the ResNet ([5]) model trained on CIFAR-10 dataset. The modified output is used as input to the next convolution layer. The RiT layer differentiates from this work in three ways. Firstly, augmentation is applied by rotating the feature maps while preserving the original map. Secondly, the augmented feature maps are concatenated with the original feature map resulting in an increase in number of feature maps. Lastly, augmentation of high-level low-resolution feature maps is proposed instead of at arbitrary levels in the CNN structure.

In another work, [23] propose performing transformation in the learned feature space instead of in input space. They argue this technique to be also effective in unsupervised representational learning in addition to supervised learning and works for both static and sequential data. In RiT, each feature map is treated as an image and rotation transformation is applied on the entire spatial dimension of the feature map.

Furthermore, in [24] the Augmented Convolutional Feature Maps (ACFM) approach applies two feature extractors (a constrained convolutional layer and a non-linear residual feature extractor) in parallel on input data. The output features from both the feature extractors are concatenated for subsequent convolution layers. The authors denote the concatenated features as augmented features. In RTN, the concatenated feature maps are obtained via a direct transformation on the original feature map.

Rotating convolutional filters instead of feature maps have been studied in the work of Diego *et al.* [25] in which the authors reported encoding rotation invariance in the CNN by rotating filters. Their work demonstrated having achieved rotation invariance on texture classification. We note here that filters were only rotated losing the original state of the filters in the process. In RiT, we rotate feature maps instead of filters and both the original and rotated feature maps are preserved.

2.4 Transformer based methods to address scale invariance

An influential piece of work in this domain by [26] introduced an end-to-end trainable module called the Spatial Transformer Network (STN) that spatially transforms feature maps by passing them through the transformer's localisation network, grid generator and sampler in succession. The heart of the STN is the localisation network that contains a feedforward network which generates and learns the parameters of the spatial transformation that should be applied to the input feature map. This architecture is reported to learn several invariances such as translation, scale, rotation and generic warping. However, the drawback of this technique is that it limits the number of objects that can be modelled in a feedforward network. In addition, since STNs perform a purely spatial transformation, the feature maps of the transformed image cannot be re-aligned with its original ([27]). In RiT layer, the simple augmentation technique of rotation is applied. The generated augmented feature maps can be mapped back to its original via inverse rotation easily.

2.5 Filter Pyramid based methods

The works of [11, 12] claim that visual processing is hierarchical and that along the hierarchy, receptive field sizes of the neurons increases. This structure of the vision system is claimed to build invariance - first to position and scale and then to viewport and other transformations. Inspired by these models the approach of using differently sized convolutional filters in parallel to capture more context is increasingly being explored by researchers [28, 29]. Google's INCEPTION family of models uses this approach [17, 30, 31]. The design of the INCEPTION module [17] is based heavily on the intuition that visual information should be processed at various scales and then aggregated so that subsequent levels can extract abstract features from different scales. A prominent model using the INCEPTION module is GoogLeNet which won the ILSVRC in 2014 [17].

Similarly, based on the concept of large kernels and pyramid based methods, [32] propose a distributed information integration CNN model called D-Net by combining local and global features from images. In D-Net, global features are extracted using multi-scale filters. The authors argued that by combining CNN features with global features contributes to improving scale invariance in CNN networks. Following the design of D-Net which combines local and global features, [33] proposed another model called Stacked Filter CNN that used multi-scale filters to first extract global features in the beginning of a CNN pipeline and then processed by a benchmark backbone CNN such as the LeNet5 network. Both these approaches use a similar approach for testing the developed models on scaled images from the CIFAR-10 and FMNIST datasets. This paper follows the same approach for testing the RTN network and results are compared with D-Net and SFCNN for CIFAR-10 and FMNIST datasets in Section 5.2.

The advantage of using filter pyramids or multi-scale filters such as in the INCEPTION module, D-Net and SFCNN is that it allows the network to make larger spatial views of the image or feature maps. This allows the network to draw in more semantic information and discriminative features from the input than the conventional small filter sizes in standard CNNs. The features obtained from multi-scale filters aggregated with locally obtained features using smaller filters (via pooling or upsampling) allows participation of spatial global features for improving the performance of the network.

Limitations of this approach are that larger filter sizes mean more parameters for the network to update, and thus, take longer for network to converge. Further, convolution operations are computationally expensive; hence, adding further multi-scale filter units slows network training and puts strain on the computing environment. These problems are further compounded by the lack of standards that govern appropriate initialisation of network weight parameters and the training parameters.

3 RTN Model

The design of RiT layer proposed in [2] is inspired by the work of [1] that provides an in-depth analysis on the internal architecture of the visual system. Their studies suggest that invariance is learnt automatically within the vision system rather than it being exposed to variations of the same image. The end-to-end model consisting of convolu-

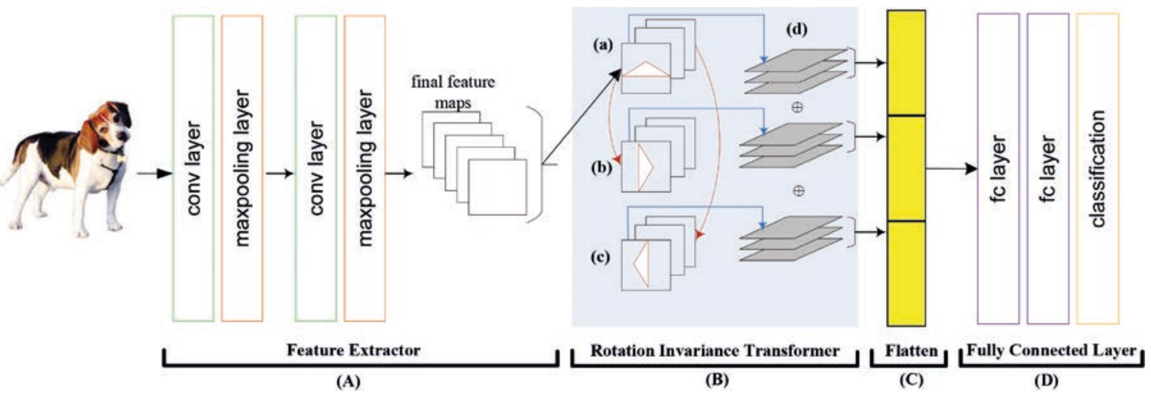


Figure 2. Architecture of RTN (adopted from [2]). The network comprises of (A) standard convolution, ReLU and maxpooling layers, (B) RiT layer, (C) flatten layer and (D) fully connected classifier layer. Features extracted through sequence of convolution layers (a) are fed to RiT layer and rotated using predefined rotation parameters which in this case, are 90° clockwise (b) & 90° anti-clockwise (c). The input features (a) and the rotated features (b) and (c) are stacked and returned as output. These are then reshaped into a vector form by the flatten layer (C) and forwarded to the classifier for learning (D).

tion layers and RiT layer is referred to as Rotation Transformer Network (RTN). The RiT layer and RTN architecture are described in [2], however, we describe the main components and processes in this section. RTN comprises four main parts as shown and described in Figure 2.

3.1 Rotation Invariance Transformer (RiT) layer

The RiT layer forms an integral part of the RTN model. This layer does not require any trainable parameters and its main operations are to apply rotation transformations to the input feature maps and stack them for forward passing. In addition to the input feature maps, the RiT layer accepts a list of rotation transformation parameters to apply to the input feature maps. These parameters are set at compile time of the model. In this work, two rotations are specified from the list $rot\_list = [90°, 270°]$ where 90° is a clockwise rotation and 270° represents 90° anti-clockwise rotation. The motivation behind choosing these rotation parameters to be in multiples of 90° is to allow a full rotation without losing any parts of a feature map that would otherwise be truncated from the edges. Since there are no parameters to be learnt within RiT layer, it executes fast and does not add significant computation time to the network. However, given that the RiT layer makes copies of the input feature maps increases the output depth dimension of the layer multiplied by a factor of n which is equal to the number of transformation supplied in rot_list. This increases the input dimension for the feedforward classifier but the number of parameters in the hidden layer remains unchanged. Hence, the reason the operation of rotation of 180° is left out at this stage is to keep the output depth of feature map to be within the processing capabilities of the available hardware. To have increased the depth of the feature four-fold would drastically increase the parameters in the fully connected layers of the classifier.

The process of RiT layer can be described mathematically as follows:

$$C^l = I^{l-1} \quad \oplus \quad T_i(I^{l-1}) \quad \oplus \quad T_{i+1}(I^{l-1}) \qquad i \in \theta \quad (1)$$

where l is the current RiT layer, l^{l-1} is the input matrix (feature map) of the previous layer of dimensions d-depth, h-height, w-width, T_i defines a rotation transformation, $\oplus$ represents feature map concatenation in d dimension and C^l is the output feature map of size $((3*d) \times h \times w)$. T is a list of θ rotation parameters in degrees given by:

$$\theta = \begin{bmatrix} \theta_1 = 90° \\ \theta_2 = 270° \end{bmatrix} \qquad (2)$$

The main objective of the RiT layer is to provide feature map augmentation via rotation transformation. Rotation as an augmentation technique is selected on these feature maps over other forms of techniques such as translations and scaling due to the following reasons:

(a) Rotations of 90° and 270° preserves all the features of the feature maps without loss of any information.

(b) Translation of these feature maps is not recommended as this will cause loss of critical information due to truncation.

(c) The spatial dimension of the final feature maps limit scaling.

3.1.1 Forward propagation

The goal of RiT layer is to create variations of the feature maps from the final convolution layer for learning by the classifier. To achieve the forward pass function, the RiT layer first accepts a set of input feature maps. The incoming input feature maps are retained while copies of it are rotated according to the rotation parameters supplied (Figure 2 (a), (b) and (c)). The input feature maps plus rotated feature maps are then concatenated to form the final output map for the next layer (Figure 2 (d)). This stacked output is returned as augmented feature maps exiting the RiT layer.

3.1.2 Backward propagation

There are no trainable parameters in RiT layer. This makes the implementation of the backward function simple and straight forward. The backward function receives gradients from the network and unstacks or slices the gradients in the exact same dimensions of the input feature maps it received during the forward pass. It finally returns the gradient slice corresponding to the input feature map which is then backpropagated to the previous layers.

Table 1 summarises the details of the datasets.

3.2 Descriptions of other layers in RTN

(A) Feature Extraction Layer:

The convolutional network of RTN is built with the standard convolution, ReLU and maxpooling layers. For this work, feature formation layers of LeNet5, VGG-16 and ResNet-18 network are used. The primary role of the convolutional network is to extract features using convolution by small filters. Having extracted local features through earlier convolution layers, the model finally outputs global features with the help of network's deeper convolution layers. It is these global features that become the main input for the RiT layer.

(C) Flatten layer:

The role of the flatten layer is unchanged from standard convolutional models. This layer operates on the incoming $2D$ feature maps and reshapes to form a single dimension feature vector.

(D) Fully Connected layer:

Here, a fully connected neural network (NN) is used. In this work, the architecture of the NN is same as those defined in the LeNet5, VGG-16 and ResNet-18 models respectively. The learning of rotated features are highly dependent on the structure of this NN; hence, the design of this layer is important with respect to the number of hidden layers and neurons in those hidden layers. It is suggested by [34] that two hidden layers are capable of representing functions with any kind of shape and that the optimal size of the hidden layer is recommended to be between the size of its input data and size of its output. LeNet5 and VGG-16 satisfy the later criteria.

4 The Experiments

This section a) summarises the materials used in this paper that include the benchmark datasets and CNN structures, b) describes the RTN component architectures and selected hyper-parameters, c) describes the training process for RTN, and d) provides a summary of the process to prepare scaled images for testing.

4.1 Materials

4.1.1 Dataset descriptions

RTN is tested on both color and greyscale images. For this, CIFAR-10, FMNIST, Tiny ImageNet and ImageHoof benchmark datasets are selected to train and test benchmark CNNs and RTNs on scaled images. The datasets contain images of different sizes and are categorised into small scale, medium scale and large-scale datasets. Training and testing RTN on these datasets helped evaluate:

(i) whether the network behaviour in terms of test performance is consistent when trained on different image resolutions compared to benchmark CNNs, and

Table 1. Details of datasets used in this paper.

	Datasets			
	CIFAR-10	**FMNIST**	**Tiny ImageNet**	**ImageHoof**
Category	small-scale	small-scale	medium-scale	large-scale
Type	RGB Color (3-channel)	Grey-scale (1-channel)	RGB Color (3-channel)	RGB Color (3-channel)
Classes	10	10	200	10
Train samples	50000	60000	100000	10800
Tests samples	10000	10000	10000	2700
Samples per train class	5000	6000	500	1080
Samples per test class	1000	1000	50	270
Image resolution	$32x \times 32$	28×28	64×64	224×224

(ii) whether the network behaviour in terms of test performance on scaled images sourced from each dataset is consistent compared to benchmark CNNs.

4.1.2 Benchmark CNN architectures

For benchmarking and feature extractor part of RTN - LeNet5, VGG-16 and ResNet-18 CNNs are used as described below:

LeNet5:

Proposed by [3], the LeNet5 network used in this work, comprises of three sets of convolution layers and two maxpooling layers. The network is trained on CIFAR-10 and FMNIST datasets due to a) the relatively small sizes of the images which are 32×32 and 28×28 pixels respectively, and b) the feasibility of using LeNet5 on small sized image datasets as shown in research such as [35].

VGG-16:

Proposed by [4], the VGG-16 network used in this work, comprises of five blocks of convolution and maxpooling layers for the feature extractor part of the network. Attached to the feature extractor is the classifier block consisting of three fully connected layers. Hence, there are a total of 16 weight layers comprising of 13 convolution and 3 fully connected layers (configuration D of the VGG network).

ResNet-18:

To overcome the vanishing gradient problem in large networks He *et al.* [5] proposed the Residual Network (ResNet) CNN architecture. The main building blocks of a ResNet CNN called the Residual Block are used to create multiple layers that are initially not used, and skips them, reusing activation functions from previous layers, thus allowing ’skip connections’ in the network. Several variants of ResNet CNN have emerged depending on the number of layers of convolutional layers, however for the purpose of this research we have used the ResNet-18 architecture that contains an initial convolutional layer followed by 4 residual blocks comprising of 4 convolutional layers and a fully connected layer.

The architecture is of the LeNet5, VGG-16 and ResNet-18 is illustrated in Figure 3.

4.2 RTN components and hyper-parameter settings

In the experiments, RiT layer is placed at the tail end of the feature extraction pipeline and before the classification layer (Figure 2). The location of the RiT layer to be at the end of all convolution and maxpooling layers is to allow final extracted features to be fed into the RiT layer for transformation prior to being sent to the classification layer. Figure 3 shows the placement of RiT layer in RTN using VGG-16, LeNet5 and ResNet-18 CNNs as backbone.

There is only one hyper-parameter for RiT layer. This is a list of rotation degrees to apply to the input feature maps. In the initial experiments, this is set to $[90°, 270°]$. Furthermore, for RiT layer to be effective, the input feature maps must satisfy the criteria where its dimensions (f_w, f_h) should be > 1. This is to ensure rotation of these feature maps is possible in the RiT layer. Table 2 shows the input feature map sizes in the RiT layer from the respective backbone CNNs trained on the different datasets and the corresponding output feature map sizes.

Table 2. Input and output of the RiT layer when embedded in the LeNet5, VGG-16 and ResNet-18 backbone CNNs as illustrated in Figure 3. Column (e) shows the increase in parameter space in the first layer of the classifier (*fc1*). Only the first layer in the classifier is affected. Column (f) shows the increase in total network parameters as a result of inclusion of the RiT layer.

(a) **Backbone CNN**	**(b)** **Dataset**	**(c)** **Input feature map** $(d \times h \times w)$	**(d)** **Output feature map** $((3 \times d) \times h \times w)$	**(e)** **Shape of *fc1* in classifier** ***(before)*** $\rightarrow$ ***(after)***	**(f)** **Total parameters in network** ***(before)*** $\rightarrow$ ***(after)***
LeNet5	CIFAR-10	$120 \times 2 \times 2$	$360 \times 2 \times 2$	(480, 84) $\rightarrow$ (1440, 84)	(92246) $\rightarrow$ (172886)
LeNet5	FMNIST	$120 \times 2 \times 2$	$360 \times 2 \times 2$	(480, 84) $\rightarrow$ (1440, 84)	(91946) $\rightarrow$ (172586)
VGG-16	Tiny ImageNet	$512 \times 2 \times 2$	$1536 \times 2 \times 2$	(2048, 4096) $\rightarrow$ (6144, 4096)	(40708104) $\rightarrow$ (57485320)
VGG-16	ImageHoof	$512 \times 7 \times 7$	$1536 \times 7 \times 7$	(25088, 4096) $\rightarrow$ (75264, 4096)	(134301514) $\rightarrow$ (339822410)
Res-18	Tiny ImageNet	$512 \times 2 \times 2$	$1536 \times 2 \times 2$	(512, 200) $\rightarrow$ (1536, 200)	(11279112) $\rightarrow$ (11483912)
Res-18	ImageHoof	$512 \times 7 \times 7$	$1536 \times 2 \times 2$	(512, 10) $\rightarrow$ (1536, 10)	(11181642) $\rightarrow$ (11191882)

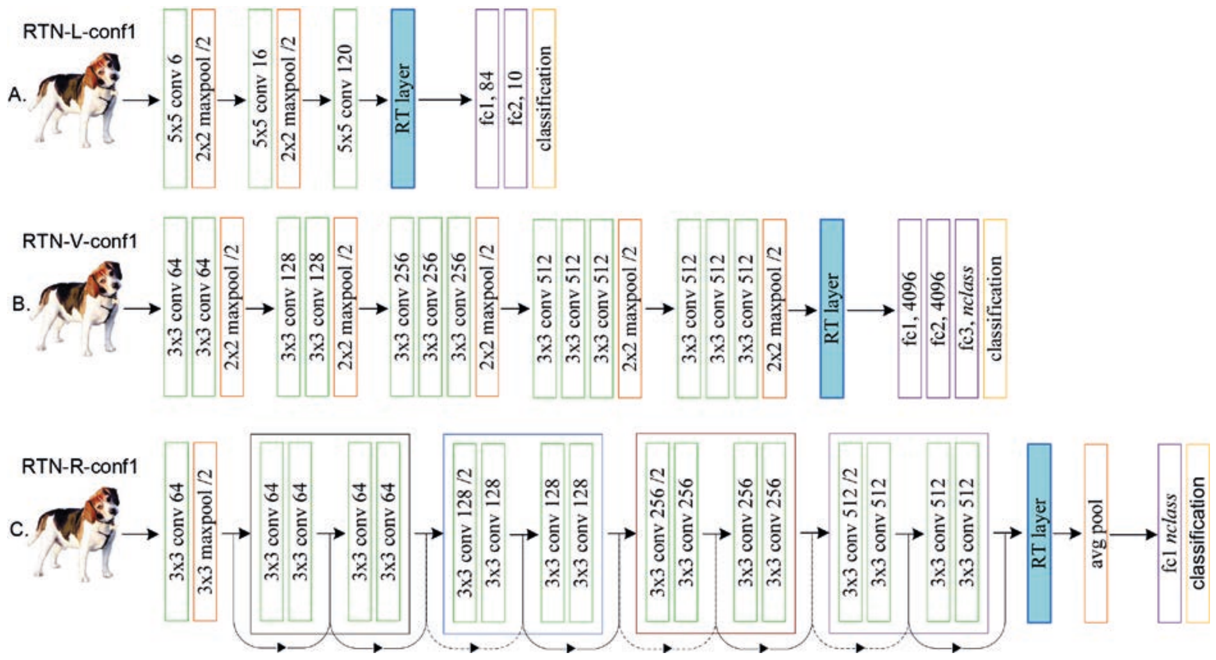


Figure 3. Placement of RiT layer in RTN using LeNet5, VGG-16 and ResNet-18 networks. The RiT layer is placed after the final feature extractor layers - denoted as (A) RTN-L-conf1 (RTN-LeNet5-configuration 1), (B) RTN-V-conf1 (RTN-VGG-16-configuration 1) and (C) RTN-R-conf1 (RTN-ResNet-18-configuration 1) respectively. RTN-L-conf1 is trained on CIFAR-10 and FMNIST datasets, while RTN-V-conf1 and RTN-R-conf1 is trained on Tiny ImageNet and ImageHoof datasets.

Table 3. Details of model training parameters, development and execution platform (reproduced from [36]). Benchmark CNNs and RTNs use the same parameters.

	Dataset					
	CIFAR-10	**FMNIST**	**Tiny ImageNet**	**ImageHoof**	**Tiny ImageNet**	**ImageHoof**
Backbone CNN	LeNet5	LeNet5	VGG-16	VGG-16	ResNet-18	ResNet-18
CNN activation function	relu	relu	relu	relu	relu	relu
FC activation function	sigmoid	sigmoid	relu	relu	relu	relu
Classification	softmax	softmax	-	-	-	-
Transfer learning	no	no	yes	yes	yes	yes
Training epochs	100	100	50	20	50	20
Learning rate	0.1 - epochs 1-2 0.01 - epochs 3-50 0.001 - epochs 51-100		0.0001 (epochs 1-50)	0.0001 (epochs 1-20)	0.0001 (epochs 1-50)	0.0001 (epochs 1-20)
Momentum	0.9					
Weight decay	0.0001					
Optimiser	Stochastic gradient decent					
Loss function	Cross-entropy					
Train batch size	4					
Test batch size	1					
DL library	Pytorch v1.2.0					
Hardware	Dell Precision T7910 64GB RAM Nvidia Geforce Titan X 12 GB GPU					

Table 4. Sample sizes and number of images in each scale category (reproduced from [36], while the original technique of scaling is adopted from [32, 37] and extended to Tiny ImageNet and ImageHoof datasets).

Dataset	**classes** (n)	**sample size** (x)	**Scale Categories (images per class, per scale)** ($s_i = n * x$)							**ense-mble**
			150 s_1	**140** s_2	**120** s_3	**100** s_4	**80** s_5	**60** s_6	**50** s_7	$\sum_{i=1}^{7} s_i$
CIFAR-10	10	100	1000	1000	1000	1000	1000	1000	1000	7000
FMNIST	10	100	1000	1000	1000	1000	1000	1000	1000	7000
Tiny ImageNet	200	20	4000	4000	4000	4000	4000	4000	4000	28000
ImageHoof	10	50	500	500	500	500	500	500	500	3500

Table 5. Reference names for the trained RTN models in this paper.

RTN model reference name	**Backbone CNN**	**Dataset**
RTN-L-conf1-Ci	LeNet5	CIFAR-10
RTN-L-conf1-Fm	LeNet5	FMNIST
RTN-V-conf1-Ti	VGG-16	Tiny ImageNet
RTN-V-conf1-Im	VGG-16	ImageHoof
RTN-R-conf1-Ti	Res-18	Tiny ImageNet
RTN-R-conf1-Im	Res-18	ImageHoof

4.3 Methodology

4.3.1 Training process

End-to-end training is performed for all RTN models based on LeNet5, VGG-16 and ResNet-18 backbone networks. The training parameters are summarised in Table 3 and are adopted from [36]. They are the same parameters that were used to train the benchmark CNNs on CIFAR-10, FMNIST, Tiny ImageNet and ImageHoof datasets to establish benchmark results against which the RTN results are compared with in Section 5 of this paper.

4.3.2 Scaled images for testing RTN

Scaled images are prepared from the test set of each dataset for testing respective RTNs. For this purpose, the technique described in [32] and [37], is used where for each dataset a random sample of images is scale transformed in 7 different scale categories, namely [150, 140, 120, 100, 80, 60, 50]. Each category value maps to a scale factor. For example 150 refers to an image scaled-up by a factor of 1.5, 50 refers to an image scaled-down by a factor of 0.5 and 100 refers to no scaling (is the original image). Each specific scale category therefore contains images enlarged or reduced of a specific size from the original. The images are scaled using the antialiasing technique [38]. In addition to testing on individual scale categories, RTN is tested on all scaled images combined in a special dataset referred to as the *ensemble* dataset. This is done to determine the average performance on all scale category images combined. Table 4 shows the details of scaled images generated for testing RTN. Table 6 shows an example of an image from each dataset and its scaled versions. Due to using RGB values $[0,0,0]$ for padding the borders around images that are scaled down, the scaled dataset is referred to as Black Border dataset.

Following this process, six trained RTN models are generated comprising of two RTN-L-conf1 networks trained on CIFAR-10 and FMNIST, two RTN-V-conf1 networks trained on Tiny ImageNet and ImageHoof datasets, and two RTN-R-conf1 networks trained on Tiny ImageNet and ImageHoof datasets. The names of these models are shown in Table 5. These names are used as reference in this paper.

5 Results and Discussion

Following the methods described in Section 4, the various RTNs developed for specific datasets are tested on their effectiveness in classifying scaled images. Here, the results are discussed in two specific areas, namely

– Effect of feature map augmentation on train and test statistics,

– Performance of RTN on scaled images.

First, the trained RTN models are compared with benchmark CNNs on model convergence by evaluating the training statistics such as loss and test accuracy. Second, the trained RTN models are rigorously tested on classification of scaled images in various aspects. In this work, the accuracy metric is used for evaluations and comparisons.

5.1 Effect of feature map augmentation on train and test statistics

A common approach to make CNNs generalise better and reduce overfitting is to apply data augmentation. In this paper, the augmentation of feature maps approach is applied instead. Whilst data augmentation causes an increase in number of input samples with no change in internal model architecture, this approach forces an increase in feature maps by a factor of 3 due to the applied transformations. Given the increase in feature maps, the convergence of the RTN models are investigated by analysing and comparing the RTN train and test statistics with the train and test statistics of standard benchmark CNNs trained on the same datasets.

Table 7 compares the train and test statistics (accuracy and loss) for all the networks. The results are for RTN-L-conf1, RTN-V-conf1 and RTN-R-conf1 networks using LeNet5 (L), VGG-16 (V) and ResNet-18 (R) backbone CNNs respectively (see Figure 3). The RTN-L-conf1 network is trained on CIFAR-10 and FMNIST datasets, while RTN-V-conf1 and RTN-R-conf1 networks are trained on Tiny ImageNet and ImageHoof datasets. These statistics are evaluations on the raw train and test images without any form of scale jittering. As can be seen, the RTN results are not only comparable, but have significantly performed better compared to

Table 6. Black-Border dataset. An example of scaled image from CIFAR-10, FMNIST, Tiny ImageNet and ImageHoof datasets (reproduced from [36]).

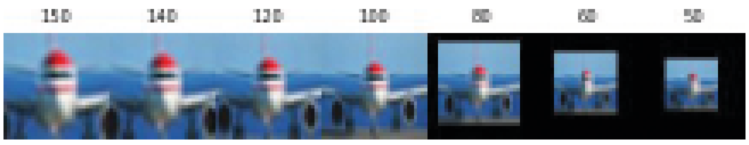
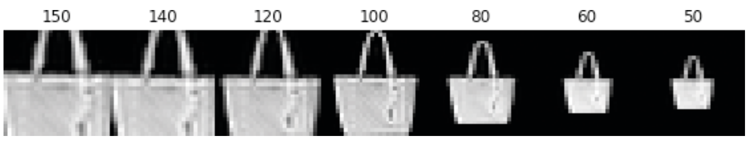
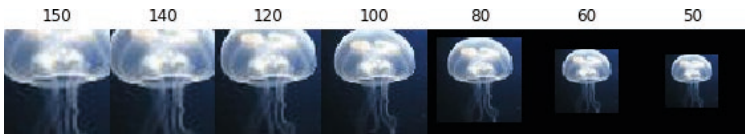
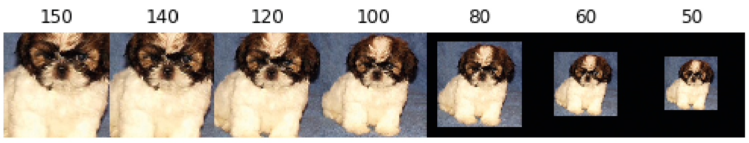
Dataset	Examples
CIFAR-10	
FMNIST	
Tiny ImageNet	
ImageHoof	

Table 7. Train and test statistics of RTN (L-conf1, V-conf1 and R-conf1) and benchmark CNNs.

Model	**train acc**	**train loss**	**test acc**	**difference (loss)**	**difference (test acc)**
			CIFAR-10		
LeNet5-Ci	0.737	1.734	0.568		
RTN-L-conf1-Ci	**0.856**	**1.623**	**0.638**	**-0.111**	**+7.0%**
			FMNIST		
LeNet5-Fm	0.934	1.535	0.899		
RTN-L-conf1-Fm	**0.962**	**1.505**	**0.910**	**-0.030**	**+1.1%**
			Tiny ImageNet		
VGG-16-Ti	0.995	0.018	0.576		
RTN-V-conf1-Ti	**0.995**	0.018	**0.587**	0.000	**+1.1%**
			ImageHoof		
VGG-16-Im	0.994	0.022	0.898		
RTN-V-conf1-Im	**0.999**	**0.004**	**0.905**	**-0.018**	**+0.7%**
			Tiny ImageNet		
ResNet-18-Ti	**0.961**	**0.144**	0.449		
RTN-R-conf1-Ti	0.938	0.211	**0.466**	+0.067	**+1.7%**
			ImageHoof		
ResNet-18-Im	**0.982**	**0.066**	0.902		
RTN-R-conf1-Im	0.960	0.134	**0.915**	+0.068	**+1.3%**

the standard CNN results on all datasets - color and greyscale (indicated in bold).

Significant results are obtained on small and medium scale datasets (FMNIST, CIFAR-10 and Tiny ImageNet) than on large-scale ImageHoof datasets indicated by the higher differences in test accuracy compared to the benchmark CNNs. However, this is coupled with an indication of declining test accuracy from small to large scale images trained on RTN (from 7.0% on CIFAR-10 dataset to 0.7% on ImageHoof dataset). One of the factors that the approach of feature map augmentation depends on is the output size of the feature extractor of the backbone CNNs. For example, in the experiments, the spatial dimensions ($h \times w$) of feature maps that are augmented for CIFAR-10 and FMNIST datasets are (2×2) while for ImageNet dataset it is (7×7). A bigger spatial dimension results in a higher input space for the classifier. This means the architecture of the classifiers need to be optimised to handle this increase in input. In the experiments the classifiers are unchanged and have the same architecture as they are in the standard CNNs.

Furthermore, RTN models are trained using the same training parameters as the standard CNNs and using the same number of training cycles. The results indicate the stability of the RTN during training despite the increase in feature maps for the classifier to process. Whilst the architecture of the feature extractor part of RTN remained unchanged, the increase in inputs to the classifier in the form of augmented feature maps fuelled the classifier with more data to process, thus minimising overfitting in the classifier layer and influencing better weight updates for the entire model.

The main conclusion derived from these results is that feature map augmentation in network training is useful in improving the classification of accuracy of the models, noting the augmentation of feature maps happens at the end of the feature extractor in CNNs.

5.2 Performance of RTN on scaled images

Finally, the consistency and performance of the RTN models on scaled images is evaluated. For this, improvements in individual scale category prediction accuracies and evaluations on the consistency of the results with results from benchmark models are investigated. The results of RTN and benchmark models on different scale categories on different datasets are outlined in Table 8. The accuracy metric is used to evaluate the test accuracy of the models from each scale category as well as on all combined scaled images in the ensemble dataset. The scores are highlighted bold in instances where the accuracy of RTN is higher than benchmark CNN for a scale category. The column *hit rate* is a count of scale categories the RTN model outperformed the benchmark CNN model for each dataset. In order to show promise in classification of scaled images, RTN is anticipated to perform better than benchmark models on as many scale categories as possible. Hence, a minimum threshold of 50% for *hit rate* is established. This means RTN should at least perform better on 50% of the scale categories over benchmark models on each dataset.

Compared with the benchmark CNNs, the performance of RTN on scaled images is analysed in three categories:

(i) Generalisation on multi-resolution datasets,

(ii) Performance on enlarged and reduced image scales, and

(iii) Evaluations on color and greyscale images.

(i) Generalisation on multi-resolution datasets:

In order to determine whether RTN performance is similar on scaled images from all datasets, the accuracy on all scale categories and on the ensemble dataset is evaluated. The hit rates of RTN on all datasets are $> 50\%$ indicating higher accuracy on majority of the scale categories than the benchmark CNNs. Furthermore, RTN outperformed the benchmark models on the ensemble dataset, with a difference of 5.6%, 1.7%, 0.6%, 1.7%, 1.0% and 1.9% on CIFAR-10, FMNIST, Tiny ImageNet and ImageHoof datasets trained on LeNet5, VGG-16 and ResNet-18 backbone CNNS respectively. This accounts for 392, 119, 168, 60, 280, and 67 more scaled images classified correctly from the ensemble dataset respectively. The difference in accuracy (in percentages) of RTNs and benchmark CNNs is shown in Table 8.

On the basis of classification accuracy results over various scale categories and the ensemble dataset, a promising performance of RTN on most

Table 8. Performance summarisation (accuracy) of RTN (L-conf1, V-conf1 and R-conf1) and benchmark CNNs on all the scale categories. The difference in accuracy between RTN and benchmark CNNs are shown as percentages in brackets. A positive figure indicates higher accuracy of RTN over the benchmark CNN on the respective scale category.

		scale categories (scale size in (%))								
Model	**met-ric**	**ense-ble**	**150**	**140**	**120**	**100**	**80**	**60**	**50**	**hit rate**
					CIFAR-10					
LeNet5-Ci	acc	0.381	0.449	0.478	0.531	0.577	0.265	0.217	0.149	
RTN-L-conf1-Ci		**0.437 (+5.6%)**	**0.491 (+4.2%)**	**0.542 (+6.4%)**	**0.610 (+7.9%)**	**0.631 (+5.4%)**	**0.396 (+13.1%)**	0.185 (-3.2%)	**0.206 (+5.7%)**	0.857 (6/7)
					FMNIST					
LeNet5-Fm	acc	0.611	0.575	0.654	0.785	0.895	0.703	0.373	0.295	
RTN-L-conf1-Fm		**0.628 (+1.7%)**	**0.579 (+0.4%)**	0.641 (-1.3%)	**0.788 (+0.3%)**	**0.915 (+2.0%)**	**0.705 (+0.2%)**	**0.423 (+5.0%)**	**0.348 (+5.3%)**	0.857 (6/7)
					Tiny ImageNet					
VGG-16-Ti	acc	0.351	0.398	0.450	0.534	0.580	0.307	0.114	0.055	
RTN-V-conf1-Ti		**0.357 (+0.6%)**	**0.402 (+0.4%)**	**0.464 (+1.4%)**	**0.550 (+1.6%)**	**0.594 (+1.4%)**	**0.318 (+1.1%)**	0.114 (0.0%)	**0.058 (+0.3%)**	0.857 (6/7)
					ImageHoof					
VGG-16-Im	acc	0.856	0.852	0.858	0.892	0.894	0.888	0.840	0.760	
RTN-V-conf1-Im		**0.873 (+1.7%)**	**0.878 (+2.6%)**	**0.896 (+3.8%)**	**0.916 (+2.4%)**	**0.916 (+2.2%)**	**0.902 (+1.4%)**	0.830 (-1.0%)	**0.790 (+3.0%)**	0.857 (6/7)
					Tiny ImageNet					
ResNet-18-Ti	acc	0.262	0.285	0.342	0.426	0.454	0.225	0.062	0.042	
RTN-R-conf1-Ti		**0.272 (+1.0%)**	**0.321 (+3.6%)**	**0.367 (+2.5%)**	**0.427 (+0.1%)**	**0.463 (+0.9%)**	0.217 (-0.8%)	**0.069 (+0.7%)**	0.042 (0.0%)	0.714 (5/7)
					ImageHoof					
ResNet-18-Im	acc	0.844	0.848	0.892	0.888	0.906	0.864	0.796	0.712	
RTN-R-conf1-Im		**0.863 (+1.9%)**	**0.894 (+4.6%)**	0.890 (-0.2%)	**0.922 (+3.4%)**	**0.920 (+1.4%)**	**0.874 (+1.0%)**	**0.798 (+0.2%)**	**0.744 (+3.2%)**	0.857 (6/7)

scale categories compared to the respective benchmark LeNet5, VGG-16 and ResNet-18 networks is confirmed.

(ii) Performance on enlarged and reduced image scales:

In this category, the performance of RTN is compared with benchmark CNNs on both upscaled and downscaled scale categories. RTN performance is observed to be higher on all scaled up color images (categories $150, 140, 120$) than on greyscale images (except on scale category 140 by the RTN network using the ResNet-18 backbone trained on ImageHoof dataset). Here, average accuracy on upscaled scale categories are 6.2%, 1.1%, 2.9%, 2.1% and 2.6% for CIFAR-10, Tiny ImageNet and ImageHoof datasets trained on LeNet5, VGG-16 and ResNet-18 backbone CNNS respectively, whereas average accuracy on FMNIST dataset is -0.2%.

There is promising performance of RTN over benchmark models when comparing accuracy scores on scaled down images (categories $80, 60, 50$). Considering LeNet5 and VGG-16 based RTN networks tested on color images (CIFAR10, Tiny ImageNet and ImageHoof datasets), performance is not the same as on scaled up images, where hit-rate is only noted to be better on at least 2 out of 3 downscaled scale categories. Here, a positive average accuracy of 5.2%, 0.5% and 1.1% is obtained on downscaled scale categories on CIFAR-10, Tiny Image and ImageHoof datasets respectively. However, the results of ResNet-18 based RTN networks tested on ImageHoof scaled color images indicate better performance on all downscaled scale categories. This indicates that while the results are generally promising on scaled up images, the performance of RTN networks may differ on scaled down images if built on top of different backbone CNNs.

Further, a higher average accuracy of RTN is noted on scaled down FMNIST images than on scaled up images and vice versa on color datasets. A probable explanation provided is that the padding of RGB [0,0,0] values around scaled color images further confuses the models into miss-classifying the images to its correct class, as the pixels are outside the normal color distribution of the dataset. On the other hand, the padded pixel values around greyscale scaled down images fall within the normal greyscale color distribution of the dataset, thus showing a different effect. From these results two conclusions are drawn:

1. The process of upscaling has enhanced some the features and that enough detail remains in the scaled up images for RTN to process, causing results to be better than on scaled down images.
2. The backbone CNN on which RTN networks are built has an influence on the overall performance of the network on scaled up and downscaled images.
3. The presence of noise in the form of RGB values [0,0,0] padded around scaled down images further confuses the model, forcing hit rates to be lower for scaled down images than on scaled up images particularly on color datasets.

(iii) Evaluations on color and greyscale scaled images:

In this category, the performance of RTN on color and greyscale images is studied. Despite the fact that only a single greyscale dataset is used in the experiments, some interesting patterns are observed. For example, in-spite of similar image resolution in CIFAR-10 and FMNIST datasets, RTN is shown to perform better on CIFAR-10 color scaled images. A similar performance of RTN on scaled Tiny ImageNet and ImageHoof images is observed. From these results three conclusions are drawn:

1. CIFAR-10, Tiny ImageNet and ImageHoof all contain natural images; and, therefore, have more complex features than the greyscale FMNIST dataset. The presence of more discriminative features allows the models to perform better.
2. The presence of color promotes the diversity of the object colors such as dogs with different fur colors. This causes the network to become slightly tolerant to color invariance ([35]).
3. Despite truncation of images due to upscaling, the presence of color provides enough details for the network to extract features from. Greyscale images lack this advantage.

Table 9 compares the performance of RTN networks with D-Net [32] and SFCNN [33] networks on scaled images. The comparison is possible since

Table 9. Comparison of RTN (L-conf1) networks with benchmark LeNet5, D-Net [32] and SFCNN [33] networks. The values for RTN accuracies which are highlighted bold illustrates scale categories on which performance of RTN is higher than on D-Net and SFCNN networks.

		scale categories (scale size in (%))								
Model	**metric**	**ensemble**	**150**	**140**	**120**	**100**	**80**	**60**	**50**	**hit rate compared with LeNet5**
CIFAR-10										
LeNet5-Ci **[32, 33]**	acc	0.381	0.449	0.478	0.531	0.577	0.265	0.217	0.149	
D-Net-Ci **[32]**		0.419	0.481	0.532	0.594	0.637	0.277	0.214	0.195	0.857 (6/7)
SFCNN-Ci **[33]**		0.394	0.438	0.487	0.547	0.586	0.351	0.193	0.159	0.714 (5/7)
RTN-L-conf1-Ci (**this paper**)		**0.437**	**0.491**	**0.542**	**0.610**	0.631	**0.396**	**0.185**	**0.206**	0.857 (6/7)
FMNIST										
LeNet5-Fm **[32, 33]**	acc	0.611	0.575	0.654	0.785	0.895	0.703	0.373	0.295	
D-Net-Fm **[32]**)		0.629	0.570	0.685	0.804	0.922	0.712	0.410	0.303	0.857 (6/7)
SFCNN-Fm **[33]**		0.566	0.480	0.573	0.743	0.880	0.647	0.369	0.267	0.000 (0/7)
RTN-L-conf1-Fm (**this paper**)		0.624	**0.592**	0.650	0.791	0.914	0.696	**0.421**	**0.306**	0.714 (5/7)

Table 10. Statistical significance of the global accuracy (p_2 vs p_1) of RTNs with respective benchmark CNNs models trained on each dataset.

parameters	**Test on CIFAR-10**	**Test on FMNIST**	**Test on Tiny ImageNet**	**Test on ImageHoof**	**Test on Tiny ImageNet**	**Test on ImageHoof**
C_1	LeNet5-Ci	LeNet5-Fm	VGG-16-Ti	VGG-16-Im	ResNet-18-Ti	ResNet-18-Im
C_2	RTN-L-conf1-Ci	RTN-L-conf1-Fm	RTN-V-conf1-Ti	RTN-V-conf1-Im	RTN-R-conf1-Ti	RTN-R-conf1-Im
p_1	0.568	0.899	0.576	0.898	0.449	0.902
p_2	0.638	0.910	0.587	0.905	0.466	0.915
$\hat{p_1}$	0.577	0.895	0.580	0.894	0.454	0.906
$\hat{p_2}$	0.631	0.915	0.594	0.916	0.463	0.920
x_1	577	895	2320	447	1816	453
x_2	631	915	2376	458	1852	460
n	1000	1000	4000	500	4000	500
p	0.604	0.905	0.587	0.905	0.459	0.913
Z	**-2.469**	**-1.525**	**-1.272**	**-1.186**	**-0.808**	**-0.785**
Accepted hypothesis	H_1	H_1	H_1	H_1	H_1	H_1

our work uses a similar approach to test scaled images. Since D-Net and SFCNN are tested on scaled CIFAR-10 and FMNIST images, we compare their results with RTN networks on the same datasets. On CIFAR-10 dataset, RTN accuracies are higher on 6 out of the 7 scale categories compared to accuracies on the same scale categories for the LeNet5, D-Net and SFCNN networks. On FMNIST dataset, RTN performance is higher on 3 scale categories. Comparing hit rates, both RTN and D-Net hit rates are higher than SFCNN. However, multi-scale filters in the D-Net model makes it a computationally expensive model whereas no trainable parameters in the RiT layer in the RTN network makes it a lightweight model to use for the application of classifying scaled images.

5.3 Model validation

The statistical significance of the global accuracy of RTNs are compared with respective benchmark CNNs using the *Test of Hypothesis for the Difference of Two Proportions* method ([39]). The comparison is made for each pair of benchmark CNN and RTN model trained on each datasets.

A Test for the Difference of Two Proportions:

- Let p_1 and p_2 be global accuracy of classifiers C_1 and C_2 on the training set T. Here, C_1 and C_2 refer to benchmark CNN and RTN trained on a dataset.

- Let t be a sample from test set T of size n. Here, t is the sample corresponding to scale category [100]. This is because the images in this category are not manipulated by scaling; and, therefore, share the same distribution as the global test set of each dataset,

- Let $\hat{p}_1$ and $\hat{p}_2$ be the accuracies obtained from classifiers C_1 and C_2 on t respectively.

- Let x_1 and x_2 be the number of samples correctly classified from classifiers C_1 and C_2 on t respectively, such that:
  $\hat{p}_1 = \frac{x_1}{n}, \hat{p}_2 = \frac{x_2}{n}$

- The test statistic is given by:

$$Z = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{2p(1-p)/n}} \quad \text{where} \quad p = \frac{(x_1 + x_2)}{2n} \tag{3}$$

- The goal is to prove p_2 of classifier C_2 is better than p_1 of classifier C_1. Thus, the null hypothesis H_0 and alternative hypothesis H_1 is formulated as:

  $H_0 : p_1 = p_2$ (no statistically significant difference between accuracy of C_1 and C_2)
  $H_1 : p_1 \neq p_2$ (there is statistically significant difference between accuracy of C_1 and C_2)

  where the rejection region (RR) is defined by

H_1	RR	
$p_1 < p_2$	$Z < -z_\alpha$	(if true p_2 of classifier C_2 is better than p_1 of classifier C_1)
$p_1 > p_2$	$Z > z_\alpha$	(if true p_1 of classifier C_1 is better than p_2 of classifier C_2)
$p_1 \neq p_2$	$\lvert Z \rvert > -z_{\alpha/2}$	

  where z_α is obtained from a standard normal distribution for a given level of significance, α.

- In this work, $\alpha = 0.05$ for 5% level of significance giving $z_{0.05} = -1.645$. That is, when relation $Z < 1.645$, classifier C_2 is accepted as more accurate than classifier C_1 with 95% confidence level.

Table 10 summaries the Test of Hypothesis for the Difference of Two Proportions for each CNN and RTN model on each dataset. In all cases $Z < 1.645$ is obtained, indicating global accuracy p_2 of RTN models are better than p_1 of benchmark CNNs. Thus, this shows there is statistical significance of the global accuracies of the developed RTN models compared with the benchmark CNNs.

5.4 Ablation study

A technique borrowed from the field of neuroscience, *ablation* studies in Artificial Neural Network (ANN) examine the performance of a model or algorithm by removing, altering or swapping some of its parts or features ([40]). The placement of RiT layer within RTN is based on two premises:

1. That invariance happens within the ventral stream which is supported by physiological studies.

2. That high-level global features in the CNN tend to be more discriminative for classification task and appropriate to characterize objects with complex characteristic.

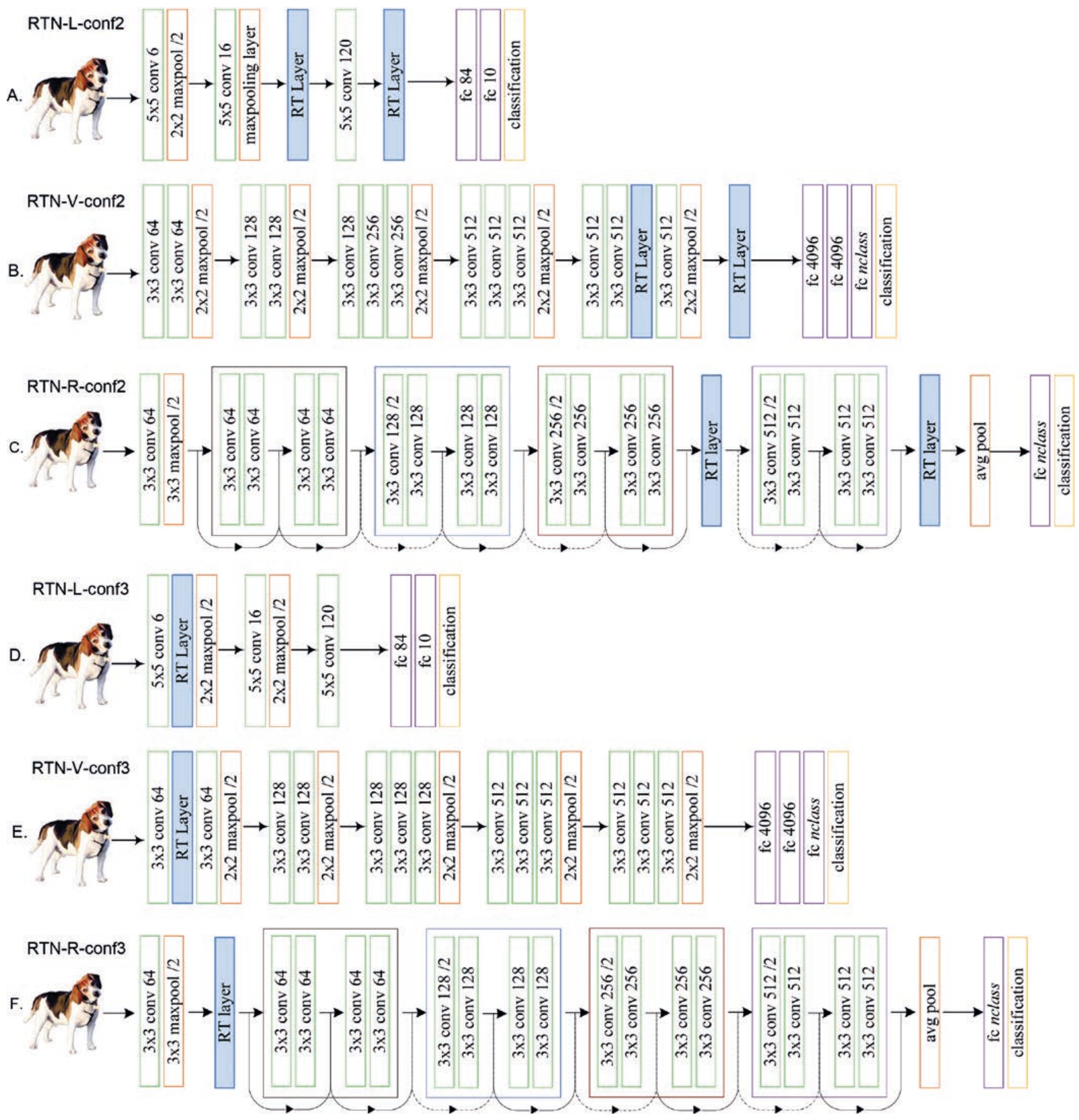


Figure 4. Placement of the RiT layer in LeNet5, VGG-16 and ResNet-18 networks in the ablation study. (A), (B) and (C) demonstrate augmentation of multiple high-level low-resolution feature maps. For this experiment, two RiT layers are placed after the final two convolution layers. (D), (E) and (F) demonstrate augmentation of low-level high-resolution feature maps. Here, the RiT layer is placed after the first convolution layer in LeNet5, VGG-16 and ResNet-18 networks.

Table 11. Performance summarisation (accuracy) of RTN (configuration 2 and 3) compared with benchmark CNNs and RTN (configuration 1) on all the scale categories.

		scale categories (scale size in (%))								
Model	**met-ric**	**ense-ble**	**150**	**140**	**120**	**100**	**80**	**60**	**50**	**hit rate**
CIFAR-10										
LeNet5-Ci		0.381	0.449	0.478	0.531	0.577	0.265	0.217	0.149	
RTN-L-conf1-Ci	acc	**0.437**	**0.491**	**0.542**	**0.610**	**0.631**	**0.396**	0.185	**0.206**	0.857 (6/7)
RTN-L-conf2-Ci		**0.431**	**0.476**	**0.543**	**0.603**	**0.647**	**0.367**	**0.218**	**0.160**	1.000 (7/7)
RTN-L-conf3-Ci		**0.384**	**0.459**	**0.495**	**0.536**	0.569	**0.347**	0.163	0.116	0.571 (4/7)
FMNIST										
LeNet5-Fm		0.611	0.575	0.654	0.785	0.895	0.703	0.373	0.295	
RTN-L-conf1-Fm	acc	**0.628**	**0.579**	0.641	**0.788**	**0.915**	**0.705**	**0.423**	**0.348**	0.857 (6/7)
RTN-L-conf2-Fm		**0.646**	**0.601**	**0.658**	**0.811**	**0.916**	**0.721**	**0.430**	**0.387**	1.000 (7/7)
RTN-L-conf3-Fm		**0.616**	0.552	0.622	0.771	**0.911**	0.692	**0.426**	**0.340**	0.429 (3/7)
Tiny ImageNet										
VGG-16-Ti		0.351	0.398	0.450	0.534	0.580	0.307	0.114	0.055	
RTN-V-conf1-Ti	acc	**0.357**	**0.402**	**0.464**	**0.550**	**0.594**	**0.318**	0.114	**0.058**	0.857 (6/7)
RTN-V-conf2-Ti		**0.352**	0.384	0.444	0.527	**0.592**	**0.328**	**0.121**	**0.062**	0.571 (4/7)
RTN_V-conf3_Ti		0.292	0.320	0.363	0.442	0.495	0.272	0.108	0.055	0.000 (0/7)
ImageHoof										
VGG-16-Im		0.856	0.852	0.858	0.892	0.894	0.888	0.840	0.760	
RTN-V-conf1-Im	acc	**0.873**	**0.878**	**0.896**	**0.916**	**0.916**	**0.902**	0.830	**0.790**	0.857 (6/7)
RTN-V-conf2-Im		**0.861**	**0.858**	**0.862**	**0.896**	**0.908**	**0.900**	**0.842**	**0.780**	1.000 (7/7)
RTN-V-conf3-Im		0.707	0.694	0.728	0.758	0.782	0.758	0.652	0.544	0.000 (0/7)
Tiny ImageNet										
ResNet-18-Ti		0.262	0.285	0.342	0.426	0.454	0.225	0.062	0.042	
RTN-R-conf1-Ti	acc	**0.272**	**0.321**	**0.367**	**0.427**	**0.463**	0.217	**0.069**	0.042	0.714 (5/7)
RTN-R-conf2-Ti		0.241	0.285	0.318	0.382	0.432	0.172	0.058	0.040	0.000 (0/7)
RTN-R-conf3-Ti		0.137	0.142	0.172	0.239	0.284	0.080	0.022	0.017	0.000 (0/7)
ImageHoof										
ResNet-18-Im		0.844	0.848	0.892	0.888	0.906	0.864	0.796	0.712	
RTN-R-conf1-Im	acc	**0.863**	**0.894**	0.890	**0.922**	**0.920**	**0.874**	**0.798**	**0.744**	0.857 (6/7)
RTN-R-conf2-Im		0.756	0.762	0.796	0.820	0.826	0.774	0.712	0.600	0.000 (0/7)
RTN-R-conf3-Im		0.285	0.304	0.330	0.392	0.420	0.236	0.184	0.132	0.000 (0/7)

Thus, to verify the above premises, ablation studies are conducted by varying the placement of the RiT layer within the CNN pipeline. They are described as follows:

5.4.1 Augmentation of multiple high-level low-resolution feature maps:

In this experiment, augmentation of feature maps are extended to other layers in the deeper of end the CNN pipeline. The aim is to expose the classifier with several variations of high-level features and to test whether augmentation of multiple high-level features maps in the deeper end of the CNN improves invariance, in particular classification of scaled images. To verify this, two RiT layers are inserted after the last two convolution layers in the LeNet5 and VGG-16 network (see Figure 4 A. and B.)), whereas in the ResNet-18 network RiT layers are added after the last two blocks (see Figure 4 C.)). These new RTN models are trained using the same training parameters as benchmark CNNs and tested on scaled images. Comparison is made with the results of benchmark CNNs on scaled images. RTNs from this ablation study are named as RTN-L-conf2, RTN-V-conf2 and RTN-R-conf2 models appended by the dataset initials.

Table 11 describes the results on scaled images. Analysing the hit rates, RTN configuration 2 models shows promising results when configured with LeNet5 and VGG-16 networks where hit rate is $> 50\%$. In addition, RTNs trained on CIFAR-10, FMNIST and ImageHoof have higher accuracies on all scale categories compared to the benchmark CNNs (hit rate = 100%). RTN configuration 2 architecture also produces better results on all FMNIST dataset scale categories compared with RTN configuration 1 models. However, when configured with ResNet-18 network as backbone, the RTN configuration 2 networks performance is poor when evaluated on Tiny ImageNet and ImageHoof scaled images. While on one hand the results confirm augmentation of multiple high-level feature maps applied on LeNet5 and VGG-16 networks using configuration 2 technique is beneficial towards classification of scaled images and that the benefits are higher on greyscale scaled images, the RiT layer placement after intermediate residual blocks in the ResNet architecture degrades the performance of the overall network.

Since the RiT layer augments the feature maps using fixed rotation parameters, adding subsequent RiT layers adjacent to each other is not required as this operation will only duplicate the augmented features from the previous RiT layers.

5.4.2 Augmentation of low-level high-resolution feature maps:

In this experiment, augmentation of feature maps in the early stages of the CNN are investigated. As such, the RiT layer is placed after the first convolution layer of the LeNet5, VGG-16 and ResNet-18 networks which are then trained using the same training parameters as benchmark CNNs (see Figure 4 (D, E and F)). RTNs from this ablation study are named as RTN-L-conf3, RTN-V-conf3 and RTN-R-conf3 models appended by the dataset initials. This architecture conflicts with the premises stated in Section 5.4 that suggest high-level features are more useful for classification task and characterize objects with complex shapes. Therefore, results from this ablation study are expected to be inferior to RTN configuration 1 and 2 models. Table 11 also shows results of configuration 3 models on scaled images compared with the benchmark CNNs (LeNet5, VGG-16 and ResNet-18). As anticipated, the hit rates are lower for RTN models on CIFAR-10 and FMNIST dataset and worse on Tiny ImageNet and ImageHoof datasets.

5.5 Performance of RTN on rotated images

Given that features are rotated in the RiT layer indicates such an operation could directly lead to promotion of rotation invariance. To answer this question, an investigation of the application of the RiT layer with CNNs on rotation invariance is presented in [2]. In this work, the RiT layer is inserted in the LeNet5 and modified versions of the VGG (MVGG) networks as the final layer in the feature extraction part. Experiments on classification of rotated images demonstrated improved results than using the CNN models without the RiT layer. The results presented in [2] and in this paper shows RTN's potential in promoting both rotation and scale invariance.

6 Conclusion

This paper has extended the work presented in [2] by demonstrating and assessing the application of the Rotation Invariance Transformer (RiT) layer for improving scale invariance in CNNs. RiT utilizes the technique of feature map augmentation within CNN models and can be dropped at the output end of a convolution or maxpooling layer in a CNN. In this work, a CNN using RiT layers is referred as a Rotation Transformer Network (RTN).

The design of RiT layer is motivated by a) findings by physiological studies that suggest the mammalian visual system encodes invariances to objects within the ventral stream as visual signals propagate from the retina to the cortex, and b) the potential benefits of the application of augmentation of input data to improve invariant object detection and classification. Experiments are conducted with the RiT layer placed at the end of the feature extraction layers of the CNN due to the knowledge that high-level features generated within CNNs are more useful for classification task and appropriate to characterize objects with complex shapes. These properties are important for invariant image classification.

In a RTN, the RiT layer takes input feature maps from convolution or maxpooling layers and performs augmentation via rotation transformation. Within the RiT layer, full rotations of 90° and 270° are performed to prevent truncation of any part of the feature map and to preserve all the features. The final output from the RiT layer is then learnt by the classifier module in the CNN.

RTNs are trained on different datasets and their performance on classification of scaled images in seven different scale categories are examined. The results are compared with benchmark LeNet5, VGG-16 and ResNet-18 networks on the same task. The results show RTNs perform better in recognising scaled images in comparison to benchmark CNNs and demonstrate promising performance of RTNs on scaled up images than on scaled down images. Furthermore, results are more promising on color images than on greyscale images.

The test of hypothesis for the difference of two proportions is used to validate the significance of the global accuracy of RTNs compared to CNNs studied in this paper. The test reveals the global test accuracy of RTN is statistically significant.

The results of ablation studies conducted show RiT layers enhance the CNNs ability to handle scale invariance when used in the deeper end of the CNN feature extraction pipeline. The study also confirms exposing the network with more variants of high-level global features (via augmentation) improves scale-invariance classification, better than when exposing the network with augmented low-level local features.

The effect of rotating feature maps in the RiT layer on rotation invariance is adequately tested and presented in [2]. Combined with results presented in this paper indicates RTN's potential in promoting both rotation and scale invariance.

There are no trainable parameters in RiT layer, thus it executes fast and does not add significant computation time during network training. Furthermore, this method can easily be integrated into CNN based applications requiring scale invariance such as in skin lesion classification. However, given the RiT layer makes copies of the input feature maps internally increases output volume of the layer by a factor of three. Though this has the benefits of extra input to avoid overfitting and improve generalisation, the downside is that it may require more training time for the network to fine tune its weights. In addition, when the RiT layer is used as the final layer in the feature extraction part of the CNN, increases the input neurons of the fully connected layer. This in turn causes an increase in the number of weights required between the input and hidden layer. Furthermore, subsequent classifier layers may need to be restructured in terms of additional neurons per layer or additional layers to accommodate for the high volume of feature maps from the RiT layer.

Finally, the results show potential benefits of RTN when RiT layers are used on high-level feature maps. The goodness of the augmented feature maps are dependent on the goodness of the global high-level features extracted. The high-level features are in turn dependent on the goodness of the local low-level features extracted. In this process, the CNN pipeline begins with local features extracted first, thus ignoring global spatial relationships amongst features which may prove useful in improving spatial invariance problem in CNNs.

The sequential implementation of the convolution and maxpooling layers in LeNet5, VGG-16 and ResNet-18 networks allowed easy integration of the RiT layer into those networks. Whilst, improvements in classification of scaled images over these classic CNN architectures are promising, the following present as exciting future work:

– the integration and evaluation of the RiT layer within residual blocks of ResNet and against more advanced CNNs such as DenseNet ([41]);
– test on applications or domains other than image classification such as audio magnitude spectrograms where where data is represented in 1D space;
– expand the ablation study by inserting the RiT layer after all convolutional layers and performing a separate training for each RiT position in order to further validate correlation of RiT placement in the network to the biological human vision.

References

[1] J. Dicarlo, D. Zoccolan, and N. C Rust, How does the brain solve visual object recognition? Neuron, vol. 73, pp. 415–34, 02 2012.

[2] D. Kumar, D. Sharma, and R. Goecke, Feature map augmentation to improve rotation invariance in convolutional neural networks, in Advanced Concepts for Intelligent Vision Systems, J. Blanc-Talon, P. Delmas, W. Philips, D. Popescu, and P. Scheunders, Eds. Cham: Springer International Publishing, 2020, pp. 348–359.

[3] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner et al., Gradient-based learning applied to document recognition, Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

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

[5] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[6] A. Krizhevsky, G. Hinton et al., Learning multiple layers of features from tiny images, Citeseer, Tech. Rep., 2009.

[7] H. Xiao, K. Rasul, and R. Vollgraf, Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, arXiv, Tech. Rep., 2017.

[8] F. F. Li, A. Karpathy, and J. Johnson, Tiny ImageNet Visual Recognition Challenge, https://tiny-imagenet.herokuapp.com/, 2019, [Online; accessed 30-Dec-2019].

[9] A. Shaw, Imagehoof dataset, https://github.com/fastai/imagenette/blob/master/README.md, 2019, [Online; accessed 10-Dec-2019].

[10] R. Maximilian and P. Tomaso, Hierarchical models of object recognition in cortex, Nature Neuroscience, vol. 2, pp. 1019–1025, 1999.

[11] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, Robust object recognition with cortex-like mechanisms, IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, no. 3, pp. 411–426, Mar. 2007. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2007.56

[12] T. Serre, Hierarchical Models of the Visual System, in Encyclopedia of Computational Neuroscience, D. Jaeger and R. Jung, Eds. New York, NY: Springer New York, 2013, pp. 1–12.

[13] T. Poggio and T. Serre, Models of visual cortex, Scholarpedia, vol. 8, no. 4, p. 3516, 2013, revision #149958.

[14] P. M. Bays, A signature of neural coding at human perceptual limits, Journal of Vision, vol. 16, no. 11, pp. 4–4, 09 2016. [Online]. Available: https://doi.org/10.1167/16.11.4

[15] D. H. Hubel and T. N. Wiesel, Receptive fields of single neurons in the cat's striate cortex, J. Physiol, vol. 148, pp. 574–591, apr 1959.

[16] Q. Zhao, T. Sheng, Y. Wang, Z. Tang, Y. Chen, L. Cai, and H. Ling, M2det: A single-shot object detector based on multi-level feature pyramid network, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 9259–9266.

[17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, Going deeper with convolutions, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[18] R. Girshick, Fast r-cnn, in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.

[19] N. Van Noord and E. Postma, Learning scale-variant and scale-invariant features for deep image classification, Pattern Recognition, vol. 61, pp. 583–592, 2017.

[20] A. Kanazawa, A. Sharma, and D. W. Jacobs, Locally scale-invariant convolutional neural networks, CoRR, vol. abs/1412.5104, 2014.

[21] D. Marcos, B. Kellenberger, S. Lobry, and D. Tuia, Scale equivariance in cnns with vector fields, arXiv preprint arXiv:1807.11783, 2018.

[22] L. Ou, Z. Chen, J. Lu, and Y. Luo, Regularizing cnn via feature augmentation, in International Conference on Neural Information Processing. Springer, 2017, pp. 325–332.

[23] T. DeVries and G. W. Taylor, Dataset augmentation in feature space, arXiv preprint arXiv:1702.05538, 2017.

[24] B. Bayar and M. C. Stamm, Augmented convolutional feature maps for robust cnn-based camera model identification, in 2017 IEEE International Conference on Image Processing (ICIP). IEEE, 2017, pp. 4098–4102.

[25] D. Marcos, M. Volpi, and D. Tuia, Learning rotation invariant convolutional filters for texture classification, in 2016 23rd International Conference on Pattern Recognition (ICPR). IEEE, 2016, pp. 2012–2017.

[26] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, Spatial transformer networks, in Advances in Neural Information Processing Systems 28. Curran Associates, Inc., 2015, pp. 2017–2025.

[27] L. Finnveden, Y. Jansson, and T. Lindeberg, The problems with using stns to align cnn feature maps, arXiv preprint arXiv:2001.05858, 2020.

[28] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, Multi-scale orderless pooling of deep convolutional activation features, in European conference on computer vision. Springer, 2014, pp. 392–407.

[29] S. Zagoruyko and N. Komodakis, Learning to compare image patches via convolutional neural networks, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 4353–4361.

[30] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, Rethinking the inception architecture for computer vision, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.

[31] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, in Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[32] D. Kumar and D. Sharma, Distributed information integration in convolutional neural networks, in Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP,. SciTePress, 2020, pp. 491–498.

[33] D. Kumar and D. Sharma, Feature map upscaling to improve scale invariance in convolutional neural networks, in Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, vol. 5. Scitepress, Feb. 2021, pp. 113–122.

[34] J. Heaton, Introduction to Neural Networks for Java, 2Nd Edition, 2nd ed. Heaton Research, Inc., 2008.

[35] H. Hosseini, B. Xiao, M. Jaiswal, and R. Poovendran, On the limitation of convolutional neural networks in recognizing negative images, in 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, 2017, pp. 352–358.

[36] D. Kumar, Multi-modal information extraction and fusion with convolutional neural networks for classification of scaled images, Ph.D. dissertation, University of Canberra, Canberra, Australia, 2020.

[37] D. Kumar and D. Sharma, Multi-modal information extraction and fusion with convolutional neural networks, in 2020 International Joint Conference on Neural Networks (IJCNN). IEEE World Congress on Computational Intelligence (IEEE WCCI), 2020, pp. 1–9.

[38] P. P. Tanner, P. Jolicoeur, W. B. Cowan, K. Booth, and F. D. Fishman, Antialiasing: A technique for smoothing jagged lines on a computer graphics image—an implementation on the amiga, Behavior Research Methods, Instruments, & Computers, vol. 21, no. 1, pp. 59–66, 1989.

[39] T. G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural computation, vol. 10, no. 7, pp. 1895–1923, 1998.

[40] R. Meyes, M. Lu, C. W. de Puiseau, and T. Meisen, Ablation studies in artificial neural networks, arXiv preprint arXiv:1901.08644, 2019.

[41] R. Annunziata, C. Sagonas, and J. Calì, Destnet: Densely fused spatial transformer networks, arXiv preprint arXiv:1807.04050, 2018.

Dinesh Kumar received his Ph.D. degree in Computing Science specialising in Computer Vision from the University of Canberra, Australia. His research interests include pattern recognition, computer vision and multi-modal information extraction and integration with a focus on brain-inspired cognitive architectures.
https://orcid.org/0000-0003-4693-0097

Dharmendra Sharma has been an academic and a researcher in artificial intelligence and its applications. He has been a Dean, Chair-Academic Board, a Distinguished Professor, GAICD, SM-IEEE, and Fellow of ACS/SPACS. He has supervised to completion over forty higher degrees research students and has produced over 315 research articles.
https://orcid.org/0000-0002-9856-4685