THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

*PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,*
*Volume 23, Number 2/2022, pp. 199–208*

# AUTOMATIC SKIN LESION CLASSIFICATION BASED ON CXception

Pufang SHAN[1], Chong FU[1,2,3], Jialei CHEN[1], Ming TIE[4], Hongfeng MA[5]

[1] Northeastern University, School of Computer Science and Engineering, Shenyang 110819, China
[2] Ministry of Education, Northeastern University, Key Laboratory of Intelligent Computing in Medical Image, Shenyang 110819, China
[3] Engineering Research Center of Security Technology of Complex Network System, Ministry of Education, China
[4] Science and Technology on Space Physics Laboratory, Beijing 100076, China
[5] Dopamine Group Ltd., Auckland 1542, New Zealand
Corresponding author: Chong FU, E-mail: fuchong@mail.neu.edu.cn

**Abstract**. Automatic classification of skin lesions remains a challenging task due to the insufficient training data, the morphological diversity of skin lesions, and the existence of artefacts and intrinsic cutaneous features in dermoscopy images. We propose a novel convolutional neural network termed CXception to tackle these challenges. CXception is constructed by plugging a coordinate attention (CA) block into the basic Xception architecture. CA block is a simple yet efficient attention mechanism, which can encode both inter-channel relationships and long-range dependencies that preserve precise positional information. Integrating CA into Xception enables the model to learn more expressive representations, hence improving the diagnostic performance of skin lesions effectively and significantly. The proposed method is supposed to handle the outliers that are not in the training set. We tackle this problem by using an efficient data-driven approach. Besides, this multi-classification task comes with the problem of heavy class imbalance. We deal with this issue by adopting an optimized loss function called the class-weighted cross-entropy loss. The experimental results on the public benchmark dataset (ISIC 2019 dataset) demonstrate the superior performance of the proposed method relative to that of the baselines (backbone network and classical classification models) and state-of-the-art approaches. Code of the proposed method is available at https://github.com/shanpufang/skin-lesion-nine.

***Key words**:* skin lesion classification, CXception, coordinate attention, class balancing, outlier class.

## 1. INTRODUCTION

Automated skin lesion classification in dermoscopy images is still a challenging task due to the following three challenges:

**Challenge 1.** Skin lesions vary significantly in many aspects, such as color, size, shape, and location in the image. Artefacts and intrinsic cutaneous features are present in dermoscopy images. The contrast between the lesion and its surrounding skin is low.

**Challenge 2.** Task 1 of the ISIC 2019 Challenge (https://challenge2019.isic-archive.com) aims to classify nine categories of skin lesions, one of which (outlier class) is not presented in the training set.

**Challenge 3.** As can be seen from Table 1, the class distributions of the dataset are highly imbalanced.

To tackle Challenge 1, we construct a novel classification model called CXception, which has strong representation learning capability. In recent years, due to their powerful feature learning capabilities, convolutional neural networks (CNNs) have been widely used in the task of skin lesion classification. Yu *et al*. [1] utilized a fully convolutional residual network (FCRN) to segment skin lesions from dermoscopy images in the first stage and classify the lesions with a very deep residual network (DRN) in the second stage. In [2, 3, 4], the authors proposed a novel approach based on the ensemble of different CNN models for automatic skin cancer classification. Hosny *et al*. [5] applied transfer learning technique to the AlexNet model for application to skin lesion diagnosis. Although these methods perform well on skin lesion

classification tasks, there are still some disadvantages: complex system [1], high computational cost [2, 3, 4], and insufficient capability of representing features [5]. Hence, the aim of this paper is to *develop a CNN model with more powerful feature extraction capabilities while maintaining a low computational burden*. Xception [6] has demonstrated remarkable performance on image classification tasks [7, 8]. It models the channel-wise and spatial feature interdependencies completely separately by using depthwise separable convolutions, successfully learning richer representations to obtain better classification results. It also introduces the residual connection into its architecture to accelerate the convergence speed and improve model performance. Moreover, Xception enjoys the advantages of low model complexity and low computational cost. Accordingly, we consider adopting Xception as the backbone network in this paper.

Moreover, attention mechanisms have been proven helpful for a wide range of computer vision tasks [9, 10]. Integrating the attention module into CNN allows the network to emphasize meaningful features and suppress unnecessary ones, thereby improving the representational power of CNN. The most popular attention mechanisms are Squeeze-and-Excitation (SE) block [11], Bottleneck Attention Module (BAM) [12], Convolutional Block Attention Module (CBAM) [13], and CA block [14]. The SE block only models the inter-channel interdependencies while neglecting the positional information, which plays a vital role in capturing the structure of an object. BAM and CBAM encode the local position information but cannot capture the long-range dependencies that are critical for vision tasks. Stronger than them, the CA block can capture both the channel-wise feature dependencies and the long-range dependencies that preserve precise position information. Besides, the CA block is a computationally lightweight unit. It can bring significant performance improvements for CNNs at a slight additional computational cost. Thus, we suggest integrating a CA block into the Xception architecture to strengthen the representational power.

We adopt an efficient data-driven approach to detect the outlier samples (see Challenge 2). We create an additional dataset as the outlier class (*i.e.*, UNK). The images in this dataset are selected from the ISIC Archive (https://www.isic-archive.com), and we are certain that none of the images belong to any category of the training set. The additional dataset (195 dermoscopic images) includes images of lentigo NOS, lentigo simplex, lichenoid keratosis, angioma, and other benign skin lesions. We also try to address this issue by applying a thresholding approach. An image is regarded as an outlier when its highest predicted probability is lower than the threshold value. However, the thresholding approach performs slightly worse than the data-driven approach, so we choose the data-driven approach for handling the outliers.

We apply a class-weighted cross-entropy loss function to deal with the class imbalance problem mentioned in Challenge 3. In recent years, lots of efforts have been devoted to overcoming the problem. The most common methods are random under-sampling and random over-sampling. The two approaches bring slight improvements in performance but suffer from the loss of valuable information (random under-sampling) and the problem of over-fitting (random over-sampling). Different from the two above methods that solve the problem from the perspective of data pre-processing, we balance the class distribution of the dataset more efficiently from the perspective of weighting the loss function. Specifically, we utilize a class-weighted cross-entropy loss to train the model. Each class' loss is multiplied by its inverse frequency in this optimized loss function. Furthermore, we define a frequency factor $k$ to control the degree of balance. The impact of different settings of $k$ on the performance will be discussed in the experiment section.

**Contribution.** Our main contribution is four-fold.

1. We construct a novel convolutional neural network termed CXception to classify skin lesions accurately. CA block, which is an efficient attention mechanism, is integrated into the Xception architecture to obtain CXception. This block enables CXception to learn more powerful representations by capturing both the inter-channel information and precise positional information, boosting the diagnostic performance of skin lesions effectively and significantly.

2. We propose an efficient data-driven approach to handle the outliers. Specifically, we create an additional dataset with external data and use the dataset as the outlier class.

3. An optimized loss function called class-weighted cross-entropy loss is introduced to mitigate the class imbalance problem of the dataset.

4. We conduct extensive experiments on the public benchmark dataset (ISIC 2019 dataset) and confirm that the proposed method outperforms the baselines (backbone network and classical classification models) and state-of-the-art approaches.

## 2. METHOD

### 2.1. CXception

In this subsection, the proposed CXception model is described in detail. We start by introducing the design idea of CXception. The Inception-style models (Inception V1, Inception V2, Inception V3, Inception-ResNet, etc.) can be considered as the stack of Inception modules. The typical Inception module first exploits the cross-channel correlations by performing 1∗1 convolution and then maps the spatial cross-correlation via regular 2D convolution. The underlying assumption behind the typical Inception module is that mapping the spatial correlation and cross-channel correlation separately performs much better than mapping them simultaneously. The simplified Inception module utilizes only 3∗3 convolution filters and abandons the pooling operation. An equivalent version of the simplified Inception module is obtained through further transformation: perform a unified 1∗1 convolution followed by three 3∗3 convolutions. Note that these three 3∗3 convolutions take part of the output channels of the unified 1∗1 convolution as the input. Based on the above observations, a question naturally arises: can we make the stronger assumption that the interdependencies in spatial and channel dimensions are modelled separately completely. Motivated by this assumption, an "extreme" version of the Inception module first perform a 1∗1 convolution to model the channel-wise feature dependencies and then captures the spatial interdependence of each output channel of the 1∗1 convolution separately. It is observed that the "extreme" Inception module is highly similar to the depthwise separable convolution [6], but the latter enjoys stronger representation learning capability. Hence, the Inception modules in Inception V3 are replaced with depthwise separable convolutions to construct Xception with more powerful representations.

To further strengthen the representational power of Xception, we consider integrating a CA block into Xception, and the obtained model is called CXception. We give a complete description of the proposed CXception architecture in Fig. 1. CXception consists of three parts (Entrance part, Middle part, and Exit part) and is structured into 14 modules (*module* 1-*module* 14).

**Entrance part (*module* 1 – *module* 4).** We implement *module* 1 with two convolution layers and ReLU activation functions. The initial convolution layer comprises 32 convolutions of size 3×3 with stride 2. The second convolution layer comprises 64 convolutions of size 3×3. *Module* 2, *module* 3, and *module* 4 mainly consist of two depthwise separable convolution layers and a 3×3 max-pooling layer with stride 2, respectively. The depthwise separable convolution layers in *module* 2, *module* 3, and *module* 4 comprise 128, 256, and 728 depthwise separable convolutions, respectively.

**Middle part (*module* 5 – *module* 12).** *Module* 5 is repeated eight times in this part. We implement *module* 5 with three depthwise separable convolution layers and ReLU activation functions. Each depthwise separable convolution layer comprises 728 depthwise separable convolutions.

**Exit part (*module* 13 – *module* 14).** *Module* 13 is mainly composed of two depthwise separable convolution layers and a 3×3 max-pooling layer with stride 2. The two depthwise separable convolution layers comprise 728 and 1,024 depthwise separable convolutions, respectively. *Module* 14 begins with a CA block and two depthwise separable convolution layers, followed by a regular convolution layer (kernel size of 3×3) and a global average pooling layer, end with a softmax layer. The two depthwise separable convolution layers comprise 1,536 and 2,048 depthwise separable convolutions, respectively. Note that the original fully connected layer is replaced with a 2D convolution layer, successfully reducing the total number of parameters in CXception. A CA block is plugged into this part to improve the representation quality effectively.

Several other design strategies for the CXception architecture are listed below.

– All depthwise separable convolutions in CXception are implemented with a kernel size of 3×3. Besides, the depth multiplier for all depthwise separable convolution layers is set to 1.

– Skip connection is extensively used in CXception, except at the very beginning and end of the model. Skip connection can avoid the problem of vanishing gradients and mitigate the degradation problem.

– We add a batch-normalization (BN) layer after each convolution layer and separable convolution layer. The BN layer is capable of solving the vanishing gradient problem, regularizing the model, and reducing the need for dropout.

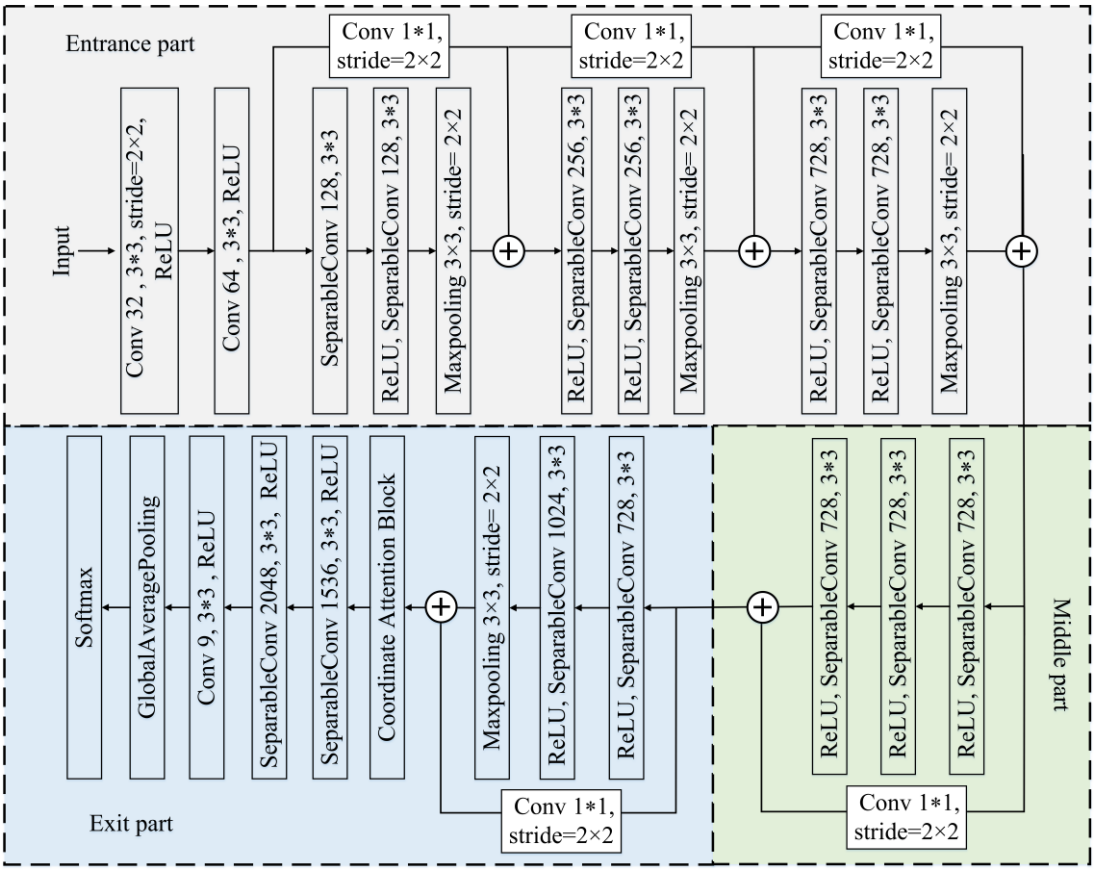


Fig. 1 – The overall architecture of CXception.

### 2.2. Coordinate attention block

The execution process of the CA block (see Fig. 2) is divided into three stages: coordinate information extraction, coordinate attention generation, and re-weighting. Algorithm 1 provides the pseudo-code of CA in a PyTorch-like style.

**Coordinate information extraction.** In the first stage, CA block applies two parallel 1D global pooling operations (*Avgpool_h* and *Avgpool_w*) to the input $X \in \mathbb{R}^{C\times H\times W}$. *Avgpool_h* and *Avgpool_w* allow the CA block to encode long-range dependencies along one spatial direction and capture the precise positional information along the other spatial direction. The outputs of *Avgpool_h* and *Avgpool_w* can be formulated as

$$H_{pool} = Avgpool\_h(X), \;\; W_{pool} = Avgpool\_w(X). \tag{1}$$

Here, *Avgpool_h*(·) is the global average pooling operation along the horizontal coordinate, and *Avgpool_w*(·) is the global average pooling operation along the vertical coordinate. $H_{pool} \in \mathbb{R}^{C\times H\times 1}$ and $W_{pool} \in \mathbb{R}^{C\times 1\times W}$ are the output feature maps produced by *Avgpool_h* and *Avgpool_w*.

**Coordinate attention generation.** In the second stage, we first perform a permutation operation of $H_{pool}$ to obtain $H'_{pool} \in \mathbb{R}^{C\times 1\times H}$. Next, we apply a concatenation operation to $H'_{pool}$ and $W_{pool}$ along the spatial dimension, yielding $HW \in \mathbb{R}^{C\times 1\times (W+H)}$. Then, we feed *HW* into a 1*1 convolution layer and a non-linear activation function. The output can be written as

$$t = \delta\left(Conv2d\left(Conca\left[Perm(H_{pool}), W_{pool}\right]\right)\right). \tag{2}$$

In Eq. (2), *Perm*(·), *Conca*[:,:], *Conv2d*(·), and δ(·) represent the operations of permutation, concatenation along the spatial dimension, 1*1 convolution, and non-linear activation function, respectively. The intermediate output is $t \in \mathbb{R}^{C/r\times 1\times (W+H)}$, which encodes the spatial information in horizontal and vertical directions. Here, *r* denotes the reduction ratio.

After that, we split $t$ into two tensors ($t^h \in \mathbb{R}^{C/r\times 1\times H}$ and $t^w \in \mathbb{R}^{C/r\times 1\times W}$) along the spatial dimension, and then perform 1*1 convolution and sigmoid activation on them separately, yielding

$$T^h = \sigma(Conv2d(t^h)), \quad T^w = \sigma(Conv2d(t^w)). \tag{3}$$

Here, $\sigma(\cdot)$ denotes the sigmoid activation function. $T^h \in \mathbb{R}^{C\times H\times 1}$ and $T^w \in \mathbb{R}^{C\times 1\times W}$ have the same channel dimension with the input $X$.

At last, we expand $T^h$ and $T^w$ into $T^H \in \mathbb{R}^{C\times H\times W}$ and $T^W \in \mathbb{R}^{C\times H\times W}$, which are regarded as the final attention weights. Note that the dimensions of $T^H$ and $T^W$ are the same as that of the input $X$.

The benefits of this stage are as follows: (1) it takes full advantage of the position information extracted from the first stage; (2) it is also capable of modelling the channel-wise relationships; and (3) the transformation is simple and the computational cost is low.

**Re-weighting.** We multiply the input $X$ by the attention weights $T^H$ and $T^W$ to produce the final refined output $X' \in \mathbb{R}^{C\times H\times W}$. The output can be written as

$$X' = X \otimes T^H \otimes T^W. \tag{4}$$

Here, $\otimes$ denotes the element-wise multiplication.

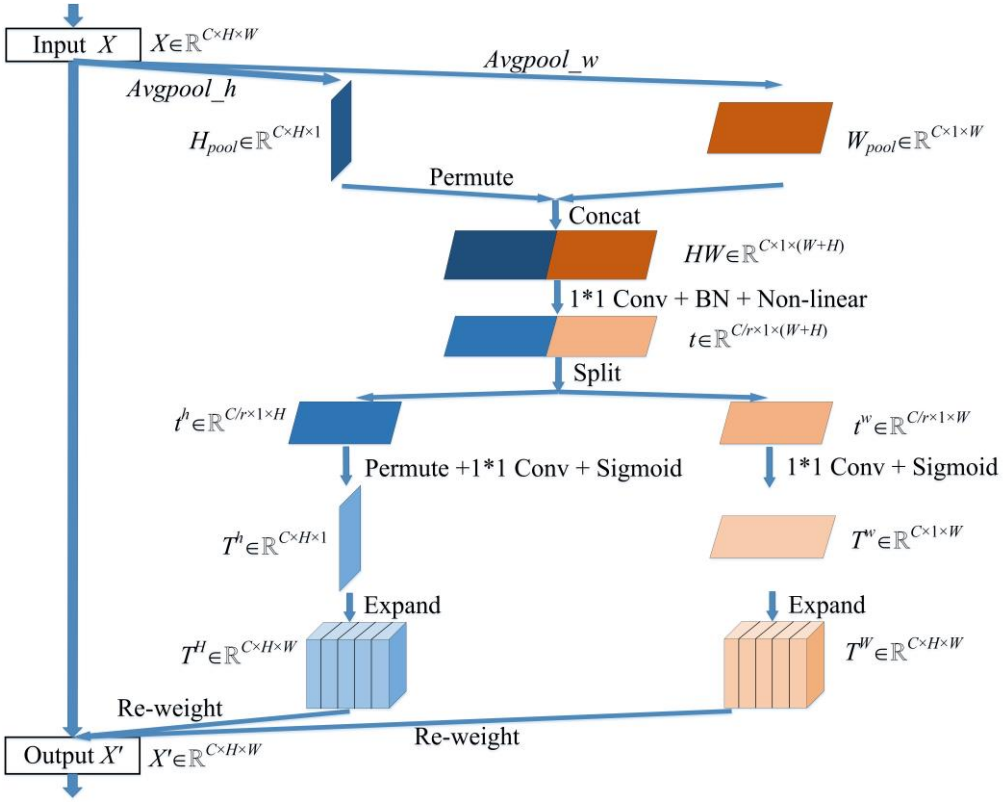


Fig. 2 – Diagram of the coordinate attention (CA) block.

### 2.3. Loss function

The proposed class-weighted cross-entropy loss function is applied to mitigate the class imbalance problem for the multi-class classification task. We write this loss function $L_{cw}$ as

$$L_{cw} = -t_i \cdot (y \cdot \log(y') + (1-y)\log(1-y')). \tag{5}$$

Here, $y$ is the true label, and $y'$ is the probability that an image belongs to the positive class. In the proposed loss function, each class' loss is multiplied by its corresponding inverse frequency to balance the class distribution on the ISIC 2019 dataset. The inverse frequency $t_i$ is defined as

$$t_i = (T / T_i)^k. \tag{6}$$

Here, $t_i$ denotes the inverse frequency of class $i$, and $k$ is a factor that controls the degree of balance. $T$ represents the total number of training samples, and $T_i$ is the number of samples of class $i$. We experiment with different values of $k$ and the model performs best when $k$=1.

**Algorithm 1** Pseudo-code of the CA block in a PyTorch-like style.

```
# X: input, C: number of channels, r: reduction ratio
---------------------------------------- initialization ----------------------------------------
Avgpool_h = nn. AvgPool2d((1, None))
Avgpool_w = nn. AvgPool2d((None, 1))
Conv1 = nn.Conv2d(C, C/r, 1)
Conv2 = nn.Conv2d(C/r, C, 1)
------------------------------------------ forward pass ------------------------------------------
# coordinate information extraction, Eq. (1)
H_pool = Avgpool_h(X)
W_pool = Avgpool_w(X)
# coordinate attention generation, Eq. (2) and Eq. (3)
H'_pool = H_pool .permute(0,1,3,2)
HW = concatenate([H'_pool, W_pool], dim=2)
t = sigmoid(Conv1(HW))
t^h, t^w = split(t, dim=2)
T^h = sigmoid(Conv2(t^h.permute(0,1,3,2)))
T^w = sigmoid(Conv2(t^w))
T^H = expand(T^h)
T^W = expand(T^w)
# re-weighting, Eq. (4)
X' = mul(X, T^H, T^W)
return X'
```

## 3. EXPERIMENTS

### 3.1. Dataset and experimental setup

The training dataset of the ISIC 2019 Challenge contains 25,331 dermoscopic images across nine categories: melanoma (MEL), melanocytic nevus (NV), basal cell carcinoma (BCC), actinic keratosis (AK), benign keratosis (BKL), dermatofibroma (DF), vascular lesion (VASC), squamous cell carcinoma (SCC), and unknown (UNK). The images come from three datasets: HAM10000 [15], BCN_20000 [16], and MSK [17]. The additional dataset used as UNK (195 images) is added to the training set of ISIC 2019 Challenge to obtain the entire training dataset (25,526 images). We give the class distribution of the entire training set in Table 1 and observe that the number of samples in the nine categories is highly imbalanced. The official test set of the ISIC 2019 Challenge contains 8,238 dermoscopic images. However, the test set is not available to us, so we split the entire training dataset to 80% for training and 20% for testing. Note that we do not use a validation set.

*Table 1*

Class distribution of the nine categories in the entire training set

| Class name | MEL | NV | BCC | AK | BKL | DF | VASC | SCC | UNK |
|---|---|---|---|---|---|---|---|---|---|
| Number | 4522 | 12875 | 3323 | 867 | 2624 | 239 | 253 | 628 | 195 |
| Ratio | 0.177 | 0.504 | 0.130 | 0.034 | 0.103 | 0.009 | 0.010 | 0.025 | 0.008 |

The experiments are carried out on an NVIDIA Tesla K40c card (12G memory). The proposed CXception model is implemented with Python based on the PyTorch library. In detail, we first implement the Xception model and CA block, and then insert the CA block into the Exit part of Xception to obtain CXception. To implement the proposed loss function, we multiply the formula of the cross-entropy loss by the inverse frequency $t_i$. A detailed description of the training and testing phases is provided below. During

the training phase, the learning rate is initialized to 0.0001 (warm-up step of 201) for training CXception from scratch. The stochastic gradient descent (SGD) algorithm is used as the optimizer. We set the number of epochs to 100, the momentum to 0.9, the weight decay to 0.0001, and the batch size to 16. Several preprocessing techniques are applied to the images. First, the images are normalized by subtracting the mean RGB values ([123.68, 116.779, 103.939]) of the ImageNet dataset. Then, we resize the short side of the image to 450 pixels while keeping the aspect ratio. Next, the image is resized to 3/4 of its size using the bilinear interpolation algorithm. Last, we expand the dataset by using several popular data augmentation strategies (*e.g.*, horizontal and vertical flipping). In the testing phase, to evaluate the proposed method quantitatively, we follow the widely adopted evaluation metrics: area under the curve (AUC), accuracy (ACC), precision (PR), recall (RE), specificity (SP), and F1-score (F1). These metrics can be formulated as:

$$ACC = (TP + TN) / (TP + FP + TN + FN), \quad PR = TP / (TP + FP), \quad SP = TN / (TN + FP),$$
$$RE = TP / (TP + FN), \quad AUC = \int_0^1 t_{pr}(f_{pr})\mathrm{d}f_{pr}, \quad F1 = 2TP / (2TP + FP + FN). \tag{7}$$

Here, *TN*, *FN*, *FP*, and *TP* denote the number of true negatives, false negatives, false positives, and true positives, respectively. $t_{pr}$ is the true positive rate and $f_{pr}$ is the false positive rate. We calculate the mean values of these metrics (*i.e.*, MAUC, MACC, MPR, MRE, MSP, and MF1) across the nine categories in the dataset.

### 3.2. Ablation studies

In this section, we conduct comprehensive experiments and ablation studies to demonstrate the effectiveness of our proposed strategies: the integration of CA block into Xception, the class-weighted cross-entropy loss function, and the data-driven approach.

**CA block.** We first verify the effect of the CA block on classification performance. Extensive experiments are performed on the following models: baseline network (Xception), SE-integrated network (Xception+SE), BAM-integrated network (Xception+BAM), CBAM-integrated network (Xception+CBAM), and CA-integrated network (CXception). The experimental results are summarized in Table 2. CXception achieves better performance than SE-integrated network, BAM-integrated network, and CBAM-integrated network, demonstrating the superiority of the CA block over SE block, BAM and CBAM. This can be explained by the fact that the SE block only models the inter-channel interdependencies while neglecting the positional information. BAM and CBAM encode the local position information but cannot capture the long-range dependencies. Powerful than them, the CA block can capture both the channel-wise feature dependencies and the long-range dependencies that preserve precise position information. Accordingly, we choose to integrate the CA block into the Xception model. We also experiment with different placements of CA in Xception and find that the model works best when CA is integrated into the end of Xception.

*Table 2*

Result comparisons of Xception, SE-integrated network, BAM-integrated network, CBAM-integrated network, and CA-integrated network

| Model | MAUC | MRE | MPR | MSP | MACC | MF1 |
|---|---|---|---|---|---|---|
| Xception | 0.931 | 0.822 | 0.966 | 0.971 | 0.958 | 0.815 |
| Xception+BAM | 0.934 | 0.824 | 0.971 | 0.972 | 0.960 | 0.819 |
| Xception+CBAM | 0.935 | 0.825 | 0.973 | 0.974 | 0.962 | 0.821 |
| Xception+SE | 0.937 | 0.827 | 0.975 | 0.976 | 0.964 | 0.823 |
| CXception | **0.942** | **0.833** | **0.978** | **0.981** | **0.971** | **0.826** |

**Class balancing.** We tackle the class imbalance problem with three different approaches: random under-sampling (the first experiment), random over-sampling (the second experiment), and the proposed class-weighted cross-entropy loss function (the third experiment). In the first and second experiments, we train CXception models on the datasets using the random under-sampling method and random over-sampling method, respectively. In the third experiment, we train the CXception model with the proposed class-weighted cross-entropy loss rather than the standard cross-entropy loss. The experimental results are listed in

Table 3. We can clearly see that the model trained with the proposed loss function achieves consistent improvements on all metrics over the other models, demonstrating the effectiveness of this loss function in handling the class imbalance problem. The proposed loss function down-weights the loss assigned to the majority classes and up-weights the loss assigned to the minority class, successfully mitigating the negative influence of class imbalance on the multi-classification task. To further study the impact of different settings of $k$ in Eq. (6) on the model performance, we experiment with different values of $k$ (*i.e.*, 1, 2, 3, and 4) and see the performance change. Results are shown in Table 4, and we observe that the model yields the best results when $k=1$. Finally, the value of $k$ is set to 1 in this work.

*Table 3*

Result comparisons of CXception models under different class balancing strategies

| Methods | MAUC | MRE | MPR | MSP | MACC | MF1 |
|---|---|---|---|---|---|---|
| Without class balancing | 0.930 | 0.819 | 0.967 | 0.969 | 0.958 | 0.813 |
| Under-sampling | 0.934 | 0.820 | 0.972 | 0.973 | 0.961 | 0.816 |
| Over-sampling | 0.933 | 0.823 | 0.974 | 0.972 | 0.960 | 0.814 |
| Proposed loss function | **0.942** | **0.833** | **0.978** | **0.981** | **0.971** | **0.826** |

*Table 4*

Result comparisons of CXception models under different settings of $k$

| $k$ | MAUC | MRE | MPR | MSP | MACC | MF1 |
|---|---|---|---|---|---|---|
| 1 | **0.942** | **0.833** | **0.978** | **0.981** | **0.971** | **0.826** |
| 2 | 0.938 | 0.830 | 0.973 | 0.977 | 0.962 | 0.819 |
| 3 | 0.934 | 0.827 | 0.967 | 0.974 | 0.958 | 0.814 |
| 4 | 0.933 | 0.825 | 0.966 | 0.972 | 0.957 | 0.811 |

**Data-driven approach.** We apply two different methods to handle the outliers. One is the data-driven approach, and the other is the thresholding approach. In the thresholding approach, a probability threshold is set in the final predicting phase. We experiment with different threshold values (*i.e.*, 0.3, 0.35, and 0.4) and find that the model achieves the best result when the threshold value is set to 0.35. Experimental results are presented in Table 5. We observe that the model using the data-driven approach outperforms the model using the thresholding approach, showing the effectiveness of the data-driven approach in handling the outlier class. Finally, the data-driven approach is adopted to deal with the outlier problem.

*Table 5*

Result comparisons of CXception models under different approaches for handling the outliers

| Methods | MAUC | MRE | MPR | MSP | MACC | MF1 |
|---|---|---|---|---|---|---|
| Thresholding | 0.934 | 0.826 | 0.967 | 0.974 | 0.960 | 0.818 |
| Data-driven | **0.942** | **0.833** | **0.978** | **0.981** | **0.971** | **0.826** |

### 3.3. Comparison with the baselines and state-of-the-art methods

To evaluate the proposed method, we compare CXception with the baselines (backbone network and classical classification models) and state-of-the-art algorithms ([18], [19], and [20]). The baselines include: (1) ResNet-101, (2) ResNeXt-101, (3) Vgg-16, (4) Vgg-19, (5) Inception-v3, (6) Inception-ResNet-v2, (7) SE-ResNeXt-50, (8) SE-ResNeXt-101, (9) SE-ResNet-50, (10) SE-ResNet-101, and (11) Xception (backbone network). As shown in Table 6, we observe significant improvements of the proposed CXception model over the baselines and state-of-the-art methods, demonstrating the superiority of our method on this skin lesion classification task. The performance gains can be attributed to three main reasons: (1) the integration of CA block into Xception enables the model to encode both the inter-channel information and the precise positional information for learning more powerful representations; (2) the class-weighted cross-entropy loss function down-weights the loss assigned to the majority classes (*e.g.*, NV) and up-weights the loss assigned to the minority class (*e.g.*, DF), effectively alleviating the class imbalance problem; and (3) we handle the outlier class with the efficient data-driven approach, leading to the improvement in diagnostic performance.

To further evaluate our method, the model complexity analysis of CXception and the baselines is provided. The model parameters, testing time per image, and Flops of these models are listed in Table 7. As we can see, the Flops and model parameters of the baselines are higher than CXception. Although the testing time per image of ResNet-101 is slightly lower than our model, our model achieves much better results than ResNet-101. In a brief conclusion, our method boosts the classification performance significantly while maintaining a low computational cost. We also plot the training loss vs. epoch graph of the proposed CXception model, see Fig. 3.

*Table 6*

Result comparisons of the proposed CXception model with the baselines and state-of-the-art methods

| Methods | MAUC | MRE | MPR | MSP | MACC | MF1 |
|---|---|---|---|---|---|---|
| ResNet-101 | 0.899 | 0.793 | 0.937 | 0.937 | 0.928 | 0.784 |
| ResNeXt-101 | 0.903 | 0.801 | 0.940 | 0.938 | 0.931 | 0.787 |
| Vgg-16 | 0.854 | 0.753 | 0.891 | 0.889 | 0.880 | 0.739 |
| Vgg-19 | 0.862 | 0.764 | 0.897 | 0.894 | 0.882 | 0.746 |
| Inception-V3 | 0.907 | 0.799 | 0.949 | 0.946 | 0.934 | 0.793 |
| Inception-ResNet-v2 | 0.886 | 0.775 | 0.924 | 0.923 | 0.913 | 0.770 |
| SE-ResNeXt-50 | 0.898 | 0.791 | 0.935 | 0.937 | 0.925 | 0.782 |
| SE-ResNeXt-101 | 0.894 | 0.786 | 0.931 | 0.930 | 0.920 | 0.779 |
| SE-ResNet-50 | 0.891 | 0.782 | 0.928 | 0.926 | 0.916 | 0.775 |
| SE-ResNet-101 | 0.883 | 0.773 | 0.920 | 0.921 | 0.911 | 0.766 |
| Xception | 0.910 | 0.814 | 0.946 | 0.952 | 0.939 | 0.803 |
| Ref. [18] | - | 0.798 | 0.804 | 0.970 | 0.949 | 0.801 |
| Ref. [19] | - | - | 0.963 | - | 0.963 | - |
| Ref. [20] | 0.910 | 0.650 | - | - | 0.950 | 0.640 |
| CXception | **0.942** | **0.833** | **0.978** | **0.981** | **0.971** | **0.826** |

*Table 7*

Model parameters, testing time per image, and Flops of the proposed CXception model and the baselines

| Methods | Model parameters | Flops | Testing time per image /(ms) |
|---|---|---|---|
| ResNet-101 | 42518601 | 25263206976 | 39.3 |
| ResNeXt-101 | 42147145 | 25817720640 | 76.2 |
| Vgg-16 | 33634121 | 46180163712 | 43.8 |
| Vgg-19 | 38943817 | 58669471872 | 45.3 |
| Inception-V3 | 24869682 | 10528293008 | 46.8 |
| Inception-ResNet-v2 | 54320297 | 22456773728 | 47.0 |
| SE-ResNeXt-50 | 25529337 | 13620662256 | 50.6 |
| SE-ResNeXt-101 | 46924857 | 25850967088 | 79.2 |
| SE-ResNet-50 | 26041417 | 13166089280 | 42.4 |
| SE-ResNet-101 | 47261769 | 25296418880 | 49.7 |
| Xception | 14516945 | 10060939178 | 39.1 |
| CXception | **14616546** | **10129967881** | **39.7** |

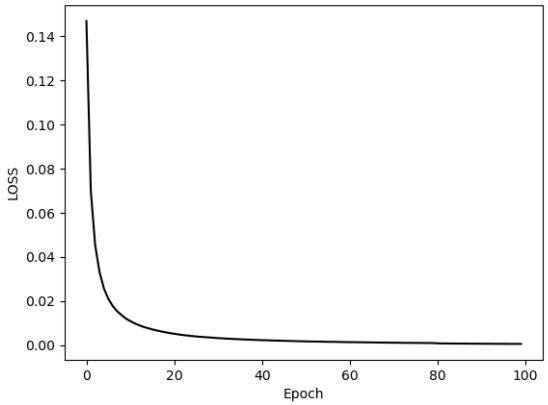


Fig. 3 – The training loss vs. epoch graph of the CXception model.

## 4. CONCLUSION

In this paper, we propose a novel convolutional neural network called CXception to diagnose skin lesions accurately. We train the CXception model with a class-weighted cross-entropy loss function to handle the imbalanced data. We introduce an efficient data-driven approach to deal with the outlier class. The proposed method is extensively evaluated on the public benchmark dataset. The experimental results demonstrate the superior performance of the proposed method relative to that of the baselines and state-of-the-art approaches.

## REFERENCES


1. L.Q. YU, H. CHEN, Q. DOU, J. QIN, P.A. HENG, *Automated melanoma recognition in dermoscopy images via very deep residual networks,* IEEE Transactions on Medical Imaging, **36**, *4*, pp. 994–1004, 2016.
2. A. NOZDRYN-PLOTNICKI, J. YAP, W. YOLLAND, *Ensembling convolutional neural networks for skin cancer classification,* International Skin Imaging Collaboration (ISIC) Challenge on Skin Image Analysis for Melanoma Detection, MICCAI, 2018.
3. J.X. ZHUANG, W.P. LI, S. MANIVANNAN, R. WANG, J. ZHANG, J. PAN, G. JIANG, Z. YIN, *Skin lesion analysis towards melanoma detection using deep neural network ensemble,* ISIC Challenge, **2018**, *2*, pp. 1-6, 2018.
4. A. MAHBOD, G. SCHAEFER, I. ELLINGER, R. ECKER, A. PITIOT, C.L. Wang, *Fusing fine-tuned deep features for skin lesion classification,* Computerized Medical Imaging and Graphics, **71**, pp. 19–29, 2019.
5. K.M. HOSNY, M.A. KASSEM, M.M. FOUAD, *Classification of skin lesions into seven classes using transfer learning with AlexNet,* Journal of Digital Imaging, **33**, *5*, pp. 1325–1334, 2020.
6. F. CHOLLET, *Xception: Deep learning with depthwise separable convolutions,* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1251–1258.
7. N. JINSAKUL, C.F. TSAI, C.E. TSAI, P. WU, *Enhancement of deep learning in image classification performance using xception with the swish activation function for colorectal polyp preliminary screening,* Mathematics, **7**, 12, art. 1170, 2019.
8. H.Y. CHEN, Y. YANG, S. ZHANG, *Learning robust scene classification model with data augmentation based on xception,* Journal of Physics: Conference Series, **1575**, *1*, art. 012009, 2020.
9. I. BELLO, B. ZOPH, A. VASWANI, J. SHLENS, Q.V. LE, *Attention augmented convolutional networks,* Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 3286–3295.
10. J. FU, J. LIU, H.J. TIAN, Y. LI, Y.J. BAO, Z.W. FANG, H.Q. LU, *Dual attention network for scene segmentation,* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 3146–3154.
11. J. HU, L. SHEN, G. SUN, *Squeeze-and-excitation networks,* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7132–7141.
12. J. PARK, S. WOO, J.Y. LEE, I.S. KWEON, *Bam: Bottleneck attention module,* arXiv preprint arXiv:1807.06514, 2018.
13. S. WOO, J. PARK, J.Y. LEE, I.S. KWEON, *Cbam: Convolutional block attention module,* Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 3–19.
14. Q.B. HOU, D.Q. ZHOU, J.S. FENG, *Coordinate attention for efficient mobile network design,* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 13713–13722.
15. P. TSCHANDL, C. ROSENDAHL, H. KITTLER, *The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions,* Scientific Data, **5**, *1*, pp. 1–9, 2018.
16. M. COMBALIA, N.C. CODELLA, V. ROTEMBERG, B. HELBA, V. VILAPLANA, O. REITER, C. CARRERA, A. BARREIRO, A.C. HALPERN, S. PUIG, et al., *BCN20000: Dermoscopic lesions in the wild,* arXiv preprint arXiv:1908.02288, 2019.
17. N.C. CODELLA, D. GUTMAN, M.E. CELEBI, B. HELBA, M.A. MARCHETTI, S.W. DUSZA, A. KALLOO, K. LIOPYRIS, N. MISHRA, H. KITTLER, et al., *Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (ISBI), hosted by the international skin imaging collaboration (ISIC),* 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018), 2018, pp. 168–172.
18. M.A. KASSEM, K.M. HOSNY, M.M. FOUAD, *Skin lesions classification into eight classes for ISIC 2019 using deep convolutional neural network and transfer learning,* IEEE Access, **8**, pp. 114822–114832, 2020.
19. M.K. MONIKA, N.A. VIGNESH, C.U. KUMARI, M. KUMAR, E.L. LYDIA, *Skin cancer detection and classification using machine learning,* Materials Today: Proceedings, **33**, pp. 4266–4270, 2020.
20. T.A. PUTRA, S.I. RUFAIDA, J.S. LEU, *Enhanced skin condition prediction through machine learning using dynamic training and testing augmentation,* IEEE Access, **8**, pp. 40536–40546, 2020.


*Received November 13, 2021*