EAI Endorsed Transactions
on Industrial Networks and Intelligent Systems
Research Article EAI.EU

SCM-Net: A Lightweight AI-Based Sea Ice Classification for Climate Change

Nazanin Baramaki[1], Quang Nhat Le[1,*], Bradley D. E. McNiven[1], and Muhammad Fahim[2]

[1]Memorial University, St. John's, NL A1B3X5, Canada
[2]Queen's University Belfast, Belfast BT7 1NN, UK

Abstract

Sea ice is considered one of the most valuable sources of information for maintaining the balance of Earth's climate system and preventing excessive warming. This study introduces a lightweight yet accurate model designed for real-time sea ice classification to provide an accessible and practical tool for operational use. We propose the SCM-Net, a deep learning model for sea ice classification applications, and compare its performance against other state-of-the-art models, including MobileNet, Residual Network (ResNet), Visual Geometry Group Network (VGGNet), Vision Transformers (ViT), and Shifted Window Transformers (SwinT). The Swin Transformer Convolutional Hybrid model (SCM-Net) is a lightweight model with around 45 times less parameters, enabling usage in real time applications. The results demonstrate that the proposed SCM-Net model achieves a comparable and even better accuracy in comparison to other models. Moreover, the proposed model significantly reduces the number of parameters while improving inference efficiency. These results show that the proposed model is well suited for real time sea ice classification applications.

Received on 15 March 2026; accepted on 07 June 2026; published on 10 June 2026

Keywords: Light-weight deep learning models, real-time applications, SCM-Net, sea ice classification, swin transformer

doi:10.4108/eetinis.132.12246

1. Introduction

Recently, sea ice in the polar regions has been melting rapidly due to the greenhouse effect and global warming. This melting of the sea ice has major impacts across many regions. Scientists believe that sea ice formation and melting directly affect ocean currents and regulate global weather. Atmospheric temperatures are also rising, leading to thinner sea ice and a shorter freezing season. Further, Arctic sea ice is declining at approximately 13% per decade [1]. Changes in global temperatures cause sea ice to break up, which increases ocean storms and waves, thus accelerating its breakage. Another effect of sea ice breakage is the opening of new Arctic navigation routes, which are high-risk due to highly variable and unpredictable sea ice conditions. As a result, identifying and classifying sea ice is of great importance for navigating these areas and maintaining global equilibrium.

One of the most reliable sources for analyzing sea ice is satellite imagery. Synthetic Aperture Radar (SAR) images are among the most reliable sources, as they are accessible in all kinds of weather and all day long, making them a valuable source for analyzing sea ice formation and melting and for navigation through polar areas [2]. One of the most important techniques for analyzing sea ice is classifying the types of ice and water in these areas. Many techniques for classifying sea ice have been proposed based on SAR image analysis. The task of SAR image analysis is complex due to the physical properties of these images and the fact that we cannot treat them as in traditional RGB image analysis methodologies. On the other hand, performing these analyses manually is time-consuming, requiring expertise in remote sensing and polar regions. Due to recent advances in computer vision, efficient methods have been proposed for classifying sea ice [3–6]. Satellite images of sea ice have become a requirement for sea ice analysis, leading to another important area

*This paper has been submitted in part for presentation at 2026 ASEAN AI Summit, Da Nang, Vietnam, July 2026. Corresponding author is Quang Nhat Le, e-mail: qnle@mun.ca

of study: the collection and labelling of sea ice images as datasets [6], [7]. Studies on the dataset in [6], which have been used for classification in this paper, indicate that there is still room for improvement in sea ice classification.

Although there are significant advancements in sea ice classification [3, 4], there are challenges such as limited sea ice images, unbalancing in the datasets, models with long training and inference time, and the need for a lightweight model that could run on small devices with limited computational capabilities. Another challenge is the visual similarities in satellite imagery, which require complex models to separate different types of sea ice. Furthermore, due to the evolving dynamics of sea ice classification, there is a need for real-time classification with minimal computational cost. There are studies on sea ice images that address the aforementioned limitations and set accuracy as the primary goal, but they require substantial computational resources [4], [8]. On the other hand, it is quite important to select appropriate performance metrics for evaluating the model, as accuracy independently may not reflect the model's usability in real-world problems. Furthermore, studies in [9] and [10] evaluate model performance in terms of memory usage and Floating Point Operations (FLOPs) to meet the efficiency requirements mentioned above, using state-of-the-art image classification models, but they did not work on SAR images.

Based on the above requirements, this study seeks to develop a classification model on sea ice SAR images that maximizes accuracy while minimizing model parameters, memory usage, and execution time. This paper proposes a deep learning model architecture for sea ice image classification. The proposed model is benchmarked against state-of-the-art models in terms of classification performance metrics, as well as model parameters, memory usage, and inference time, while also analyzing the complexity of the proposed model. The key findings of the paper are demonstrated as follows:

- We propose a lightweight image classification model (SCM-Net) which reduces model parameters and operations while decreasing the inference time.
- We provide a comprehensive comparison of the proposed model against other state-of-the-art classification models and demonstrate the proposed model's superiority in terms of model accuracy, number of parameters, training and inference time, and overall effectiveness of a multi-class sea ice classification.
- We calculate the model complexity in terms of real additional and real multiplications, and discuss the overall effectiveness of the proposed model for real-time applications.

1.1. Related Works

Sea ice classification is the ability to distinguish between various types of ice and water. In this study, the classification is among six categories including open water, leads with water, brash/pancake ice, thin ice, thick ice-flat, and thick ice-ridged. Mathematical, machine learning and deep learning methods are generally used to make a classification between ice and water, and determine the types of ice.

A proposed model by the authors in [8] utilized a mini sea ice residual convolutional network, named MSI-ResNet, for a two class, ice and water classification and their overall accuracy was 91.09%. The work in [4] used a Multiscale Dual Attention Network (MSDA-Net) for a two class sea-ice classification and MSDA-Net has an overall accuracy of 96.1%. In [11], the authors proposed an interactive sea-ice classification pipeline based on a graph convolutional network (IceGCN) to classify SAR imagery in a 4-class classification problem. This model achieves the classification performance with an overall accuracy of 95.54%. The study in [3] used a SwinT U-Net for classifying imagery into four classes, and their result shows the overall accuracy about 97%.

There are important metrics to consider when evaluating a classifier's power. The most important metric in classification problems is accuracy, as the main goal of a classifier is to be accurate. Efficiency is another metric, as the most important objective of a classifier is that it must be usable in real devices. Even if a model achieves high accuracy, excessive memory usage or long inference time can limit its applicability to real-world problems. Another objective of the proposed model is to be computationally efficient and lightweight, thus enabling training within a reasonable time and ensuring short inference time for real-time deployment and daily operational use.

There are no specific studies on sea ice classification that compare or propose classifiers based on efficiency and computational complexity. The papers that actually did comparison mainly used ImageNet-1K [12], which is a large-scale image classification dataset. For more comparison of the model's complexity and efficiency, below are some of the existing works that compare some deep learning models alongside their improved models from the efficiency perspective on top of the ImageNet-1K dataset. In [13], the authors introduced four main groups of efficiency techniques: compact architectural design, pruning, knowledge distillation, and quantization. Multiple efficiency metrics, including accuracy, number of parameters, FLOPs, latency, and energy efficiency, were used to compare the results. The work in [14] presented

a large-scale, systematic comparison of efficient Vision Transformer (ViT) architectures focusing not only on accuracy but also on computational efficiency, including runtime, throughput, memory usage, and number of parameters. A Lightweight Adaptive Feature De-Drifting Module (AFD-Module) was proposed in [15] to improve image classification performance on JPEG-compressed images while maintaining high computational efficiency. In [16], the authors presented LMS-Net as a highly efficient lightweight model, and provided a clear quantitative comparison against classical and lightweight convolutional neural networks (CNNs), and the Transformer-based architectures. Their model contains only 2.01 million parameters and requires 0.38 GFLOPs, 7.63 MB of memory, number of parameters equalling to 2.01M and the cup latency of approximately 42.7 ms. In [10], the paper designed a compact multi-task CNN with fewer layers, reduced filter sizes, batch normalization, and a hybrid pooling strategy, combined with robust face detection, alignment, and extensive data augmentation. The authors provided a direct comparison of model efficiency, showing that the proposed network used approximately 2.7 million parameters, leading to lower memory usage and reduced computational cost. In [9], the study compared five ViT architectures, evaluated them against CNN baselines (U-Net and FCN) for electromagnetic field (EMF) strength estimation in 5G environments. The authors reported and analyzed the number of model parameters (in the best model 5.9M), highlighting the trade-off between model capacity and accuracy. Computational efficiency and FLOPs are also qualitatively discussed.

Building on insights from previous studies, this paper aims to achieve strong classification accuracy while reducing parameter count and computational cost. Runtime comparisons demonstrate that, in the proposed model, despite its compact size, the model satisfies real-time inference requirements, offering a favorable trade-off among accuracy, memory footprint, and execution speed. This work focuses on developing a hybrid architecture that addresses the aforementioned goals and tries to fill this gap that is not explicitly explored in the sea ice classification field. Such models can be effectively deployed in real-time applications and improve their performance while remaining comparable to existing state-of-the-art approaches.

1.2. Problem Statement and Research Objectives

Image classification is the most important problem in the Arctic and sub-Arctic regions, as it influences the safe transportation and living in these areas. Climate change, melting ice, and the shallowing of ice in these regions highlighted the need for a real-time and accurate classifier. A real-time classifier that predicts the type of ice encountered could ensure safe navigation and improve weather predictions. Another challenge in the field of sea ice classification is that the best data for analyzing and working with sea ice classification are SAR images, which are accessible in all weather conditions and all day long, making them the best choice for weather analysis. These SAR images have some issues, including high cost, difficulty of interpretation, and thermal noise. Even after creating a dataset, it is usually imbalanced, with an unequal number of data points per class. In addition, given their special nature, basic data augmentation is not useful for them. Another important challenge in this field is that state-of-the-art deep learning methods achieving the highest accuracies often have high computational complexity and require large memory usage, even during the inference phase. Therefore, it is important to develop methods that maintain high accuracy while enabling real-time inference with low memory requirements, making them suitable for deployment as applications on devices such as smartphones.

The main research objectives of our study are illustrated as follows:

- **O1: Data Augmentation:** Making the given dataset balanced and useful for training and testing a sea ice classification problem.
- **O2: Performance benchmarking:** Evaluating and comparing the SCM-Net, the proposed classifier, with the widely used state-of-the-art deep learning classifiers for sea ice multi-class classification.
- **O3: Lightweight model comparison:** Comparing the SCM-Net efficiency in terms of model parameters, runtime, model complexity, and inference time, thus showing its superiority in comparison with the other state-of-the-art models.

1.3. Contributions and Paper Organization

The main contributions of this paper are summarized as follows:

- The paper proposes SCM-Net, which is a lightweight novel deep learning architecture that achieves the accuracy of 90% on 6 class sea ice classification problem.
- SCM-Net is benchmarked against a diverse set of state-of-the-art classification models, including CNN-based models such as ResNet-50, MobileNet, and VGG-Net, and attention-based models such as ViT and SwinTransformer, and compares them using consistent evaluation and efficiency metrics.

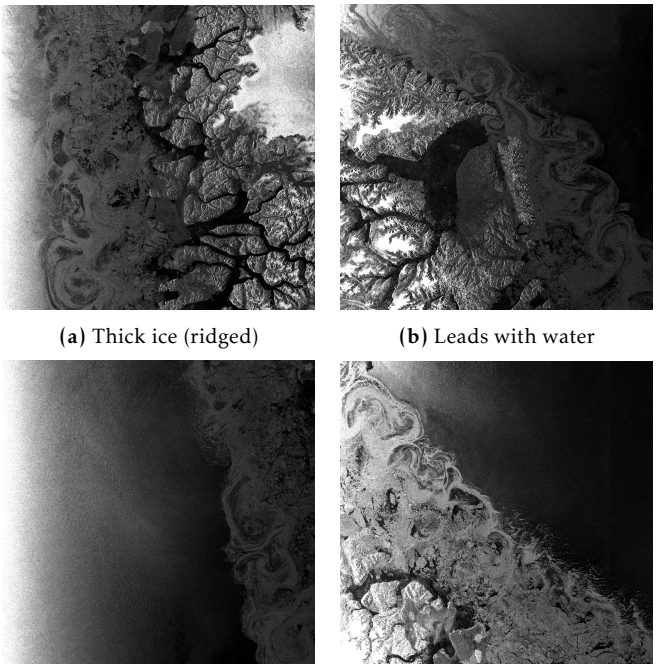
- The proposed SCM-Net model significantly reduces the number of model parameters. The proposed model's runtime and inference efficiency are much faster than those of other models, and its computational complexity is much lower than that of popular deep learning models used in image classification. These results show that the proposed model is lightweight and well-suited for real-time sea ice classification applications.

The organization of this paper is as follows: Section 2 describes the dataset and preprocessing pipeline, then provides the formal problem formulation. Section 3 introduces the proposed model's architecture, providing a detailed breakdown of model core components and an introduction to other models that have been used to compare against the proposed model. Section 4 details the experimental setup and evaluation metrics. Section 5 illustrates the classification results along with a performance analysis of the proposed model using important key performance metrics and comparison analysis against other models. Moreover, memory usage, number of parameters, complexity, and inference time are the other metrics that are described and compared in this section. Finally, Section 6 concludes the paper by summarizing the main findings and highlighting new directions for future research.

2. Dataset and Problem Formulation

2.1. Dataset Description

To understand how accurately and well the models will perform, we are using a SAR based ice types/ice edge dataset for deep learning analysis [6]. The dataset is created based on 31 Sentinel-1A Extended Wide (EW) Level-1 Ground Range Detected (GRD) scenes, with a spatial resolution of 40 m × 40 m, that were acquired north of the Svalbard archipelago during the winter months between September and March from 2015 to 2018. Four types of SAR images are shown in Fig. 1 and presented in grayscale for visualization purposes. In practice, the model utilizes a three-channel input composed of HH polarization, HV polarization, and the incidence angle. Fig. 1 illustrates a full SAR image with spatial dimensions of approximately 40 m × 40 m that contain different sea-ice types, including (a) thick ice (ridged), (b) leads with water, (c) open water, and (d) thin ice. To construct the dataset, the original images were divided into smaller patches (e.g., 46 × 46 pixels) which were subsequently manually annotated to generate labeled samples for training and evaluation. For more description, the dataset contains six classes, including open water, leads with water, brash/pancake ice, thin ice, thick ice-flat, and thick ice-ridged. The data records, called patches, were extracted from inside each polygon with a stride of 10 in different sizes: 10×10, 20×20, 32×32, 36×36, and 46×46 pixels for each class. Each patch contains three channel, co-polarization (HH), cross-polarization (HV), and incidence angle in shape of (3, patch_size, patch_size). The ranges for the HH, HV, and the incidence angle are from -30 to 0 dB, from -35 to -5 dB, and from 19 to 46 degrees, respectively.



(a) Thick ice (ridged) **(b)** Leads with water

(c) Open water **(d)** Thin ice

Figure 1. Example SAR input cube used in the dataset. The HH and HV polarization channels were extracted from Sentinel-1A Extra Wide (EW) Level-1 Ground Range Detected (GRD) scenes with a pixel spacing of approximately 40 m × 40 m. The images were processed and visualized in grayscale using the ESA SNAP software for better representation

The dataset classes information, name and number of images is shown in the table 1. The distribution of images across classes indicates significant class imbalance, while some classes experience substantial data scarcity.

Table 1. Number of samples for different classes

Class Name	Number of Images
Open Water	6,642
Leads with Water	708
Brash/Pancake Ice	105
Thin Ice	109
Thick Ice (Flat)	171
Thick Ice (Ridged)	3,036

For each experiment, 20% of the data was randomly reserved as a test set to evaluate the performance

of the model. The remaining 80% was used for data augmentation and subsequent model training. To ensure robust evaluation, 10-fold cross-validation strategy was applied on the training set.

2.2. Problem Formulation

Sea ice classification is a problem of categorizing images of the dataset to a set of predefined classes as mentioned in Table 1 . In this paper, the dataset is defined as

$$\mathcal{D} = \{(X_i, y_i)\}_{i=1}^{N}, \tag{1}$$

where $X_i \in \mathbb{R}^{H\times W\times C}$ denotes a multichannel SAR image. Height, width, and channel are denoted by H, W, and C, respectively. SAR images contain 3 channels, namely HH (Horizontal-Horizontal), HV (Horizontal-Vertical) and incident angle. $y_i \in \{1, \ldots, K\}$ represents the corresponding class label among $K = 6$ ice categories.

The objective is to learn a following parametric mapping function:

$$f_\theta : \mathbb{R}^{H\times W\times C} \to \mathbb{R}^K, \tag{2}$$

parameterized by θ, which outputs a probability distribution over the K classes. The model parameters are optimized by minimizing the error over the training set, using the following cross entropy loss:

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\mathbf{1}(y_i = k)\log\left(f_\theta(X_i)_k\right). \tag{3}$$

This formulation shows a six-class sea ice classification problem. In the following sections, the proposed SCM-Net model is described and its capabilities are compared with those of other deep learning algorithms. Performance is reported using metrics appropriate for multi-class classification, e.g., receiver operating characteristic (ROC) and confusion matrix, as well as overall accuracy. Then, the runtime and inference time of SCM-Net are compared with the stated models. Next, the computational complexity of SCM-Net is investigated.

2.3. Data Preprocessing

In this paper, preprocessing mainly involves handling class imbalance in certain classes. Standard methods of image augmentation involve cutting and flipping the original samples to obtain augmented images, which are mainly used in SAR images [17]. However, the work in [18] clearly stated that traditional augmentations such as flipping, rotation, and cropping are not physically meaningful for SAR images. The authors in [18] mentioned that SAR images are aspect angle sensitive, meaning that flipping or arbitrary rotation changes the physical scattering geometry and does not correspond to a real radar observation, potentially distorting azimuth dependent scattering characteristics. Therefore, the authors suggested that GAN-based augmentation is more suitable than simple geometric transformations. In [19], the authors explained that simple interpolation or traditional augmentation methods cannot correctly model the continuous variation of scattering with aspect angle. Since SAR image appearance changes nonlinearly with azimuth and elevation, the authors explicitly motivated their work by stating that naive augmentation cannot capture the accurate and angle-dependent SAR physics. The authors in [20] explained that standard data augmentation methods are insufficient fo r mu lti ca tegory SA R im ages, since SAR signatures are strongly dependent on depression angle, azimuth angle, and electromagnetic scattering mechanisms. Based on the above information, they decided to design a generative adversarial network (GAN)-based SAR image generator that produces class specific S AR i mages, w hich c an b e d irectly u sed to improve target recognition accuracy. In [21], the authors proposed the GAN for learning a data distribution and generating new samples. The study in [22] surveyed the usage of GAN in image augmentation. Based on the above considerations, a standard conditional GAN is used for data augmentation, which is implemented in PyTorch [23]. It augmented the dataset as follows: the number of images for classes 0 (open water: 5,313 images) and 5 (thick ice-ridged: 3,036 images) remain unchanged. In contrast, the other classes are augmented as in Table 2.

Table 2. Number of samples for different classes after augmentation in train data

Class Code	Class Name	Total	Original	Augmented
Class 0	Open Water	6,642	5,313	5,313
Class 1	Leads with Water	708	566	2,216
Class 2	Brash/Pancake Ice	105	84	3,440
Class 3	Thin Ice	109	87	3,870
Class 4	Thick Ice (Flat)	171	136	5,590
Class 5	Thick Ice (Ridged)	3,036	2,428	2,428

The data splitting process divides the dataset into 80% for training and 20% for testing. Initially, the test data is separated before applying augmentation. After augmentation, 80% of the augmented data is used for training, while 20% is reserved for validation. After 100 epochs of training, the learned model is applied to the test set, and the results are reported.

3. Proposed Model: SCM-Net

In the proposed architecture, two main components are used: the SwinT block and the convolutional (Conv) block. As the name suggests, the SwinT block, as shown in Fig. 2, is adapted from the initial Swin Transformer architecture [24]. As described earlier, it follows a

vertical structure comprising layer normalization, a shifted window multi-head self-attention mechanism, and a residual connection. The output then passes through another layer normalization step, a MultiLayer Perceptron (MLP), and a second residual connection.

The Conv block, as its name implies, is a convolution-based module, which includes two key operations: depthwise and pointwise convolutions. These operations form the foundation of MobileNet, EfficientNet-Lite, and many lightweight sea ice models. In a standard convolution, all channels are mixed simultaneously using kernels that span the entire channel dimension. In this paper, depthwise convolution was used, which processes each channel independently, avoids inter-channel mixing, and applies one filter per channel, thereby dramatically reducing computational cost. The process of a depthwise convolution is shown in Fig. 2.

After the depthwise step processes each channel separately, the point-wise convolution (a 1×1 convolution) serves as the channel mixer. In the 1×1 filter, the model does not consider spatial neighborhoods; it only combines channel information at each location of the pixels. This step generates new feature channels and allows the network to learn higher-level representations. The process of a point-wise convolution is shown in Fig. 2.

The Conv block in SCM-Net design follows a vertical structure: a 3×3 depthwise convolution, batch normalization, a pointwise convolution, and a Rectified Linear Unit (ReLU) activation. In the complete model, the input data, consisting of three channels (HH, HV, and incident angle), first passes through a Conv Block. The output is then fed into a Swin Block, followed by another Conv block. The resulting features are processed by an attention module that includes a spatial attention layer and a pointwise convolution layer. Finally, the classification head consists of a global average pooling layer and a fully connected layer that produces the final output. This architecture is shown in Fig. 2.

3.1. Shifted Window Transformer Block

The Shifted Window Transformer Block is a core building unit of the Shifted window (swin) Transformer architecture. It follows the standard Transformer block design with pre-normalization and residual connections, but replaces global self-attention with window-based Multi-head Self-Attention (MSA).

To simplify cross-window interaction while maintaining linear computational complexity, the architecture continuously uses two transformer blocks. The first block uses regular window partitioning (W-MSA), while the second uses shifted window partitioning (SW-MSA). This shift is inspired by the sliding window capability in the convolutional layer, then combining it with the attention mechanism. It is defined by $(\lfloor \frac{M}{2} \rfloor, \lfloor \frac{M}{2} \rfloor)$ pixels from the preceding layer's windows.

Formally, given the input feature z^{l-1}, the computations for two successive blocks are defined as

$$\begin{aligned} \text{W-MSA:} \quad \hat{z}^{l} &= \text{W-MSA}(\text{LN}(z^{l-1})) + z^{l-1}, \\ z^{l} &= \text{MLP}(\text{LN}(\hat{z}^{l})) + \hat{z}^{l}, \\ \text{SW-MSA:} \quad \hat{z}^{l+1} &= \text{SW-MSA}(\text{LN}(z^{l})) + z^{l}, \\ z^{l+1} &= \text{MLP}(\text{LN}(\hat{z}^{l+1})) + \hat{z}^{l+1}, \end{aligned} \tag{4}$$

where W-MSA and SW-MSA correspond to blocks l and $l+1$, respectively. The symbol $\hat{z}^{l}$ denotes the output feature after the self-attention module and residual connection. In contrast, z^{l} represents the output feature after the MLP module and the residual connection for block l, and the + operator shows the residual (skip) connection.

The MLP is built as a two-layer feed-forward network using Gaussian Error Linear Unit (GELU) activation. By alternating the window shifts between blocks, the model bridges the gaps between neighbouring windows. This allows for a much broader receptive field, similar to global attention, but without the heavy computational cost usually required. We also include relative position bias in each attention head to help the model better understand how pixels relate to one another spatially.

3.2. Depthwise Convolution

Depthwise convolution is one of the two key components of the MobileNet architecture, designed to reduce computational cost and model size compared to the standard convolutions. In a standard convolution, each output channel is computed as a weighted combination of all input channels using a $D_k \times D_k$ kernel. Given an input feature map of size $D_f \times D_f \times M$ and N output channels, the computational cost of standard convolution based on [25] is given by

$$D_k \cdot D_k \cdot M \cdot N \cdot D_f \cdot D_f. \tag{5}$$

Depthwise separable convolution consists of two layers: depthwise convolutions and pointwise convolutions. The depthwise separable convolution decouples the number of output channels from the kernel size. Thus, instead of combining all channels during filtering, each channel is convolved separately.

The computational cost depends multiplicatively on the number of input channels M, the kernel size $D_k \times D_k$, and the feature map size $D_f \times D_f$. MobileNet models address each of these terms, and their interactions [25]. In depthwise convolution, a single $D_k \times D_k$ filter is applied independently to each input channel. The computational cost becomes

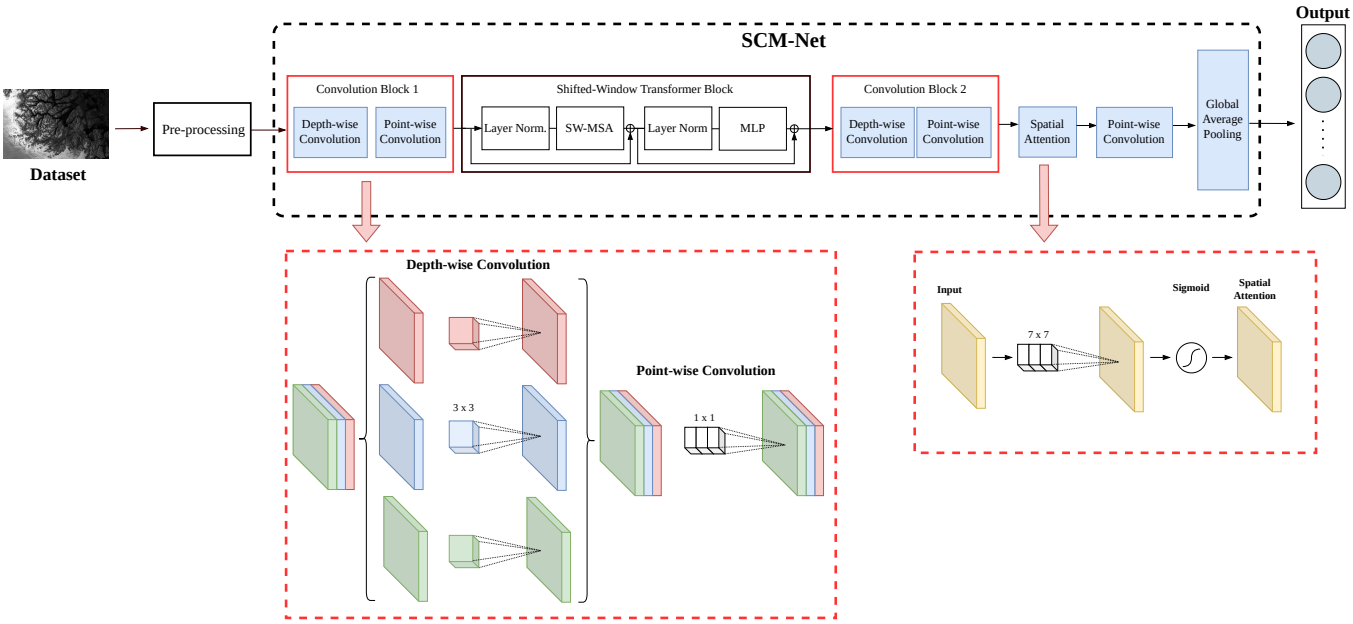


Figure 2. The proposed SCM-Net model architecture

$$D_k \cdot D_k \cdot M \cdot D_f \cdot D_f. \tag{6}$$

Unlike standard convolution, depthwise convolution does not produce new channel combinations; Instead, it performs spatial filtering separately for each channel. To recover cross-channel interaction, MobileNet follows a depthwise convolution with a *pointwise convolution*, which is a 1×1 convolution applied across channels to create a linear combination of the depthwise layer's output. The computational cost of the pointwise convolution is defined as

$$M \cdot N \cdot D_f \cdot D_f. \tag{7}$$

Therefore, the total computational cost of depthwise separable convolution becomes

$$D_k \cdot D_k \cdot M \cdot D_f \cdot D_f + M \cdot N \cdot D_f \cdot D_f. \tag{8}$$

Compared to standard convolution, this factorization decreases computation approximately by a factor of

$$\frac{D_k D_k M D_f D_f + M N D_f D_f}{D_k D_k M N D_f D_f} = \frac{1}{N} + \frac{1}{D_k^2}. \tag{9}$$

For standard values such as $D_k = 3$, the computational savings are substantial. This efficiency makes depthwise separable convolution as the fundamental building block of MobileNet architectures, enabling lightweight and low-latency neural networks for mobile and embedded applications.

4. Experimental Setup and Evaluation Metrics

This section describes the experimental protocol used to compare all models and the evaluation metrics reported in the results.

4.1. Experimental Setup

All experiments are implemented in Python using standard PyTorch libraries. Specifically, classical models and the preprocessing pipeline are implemented with scikit-learn. For evaluation, stratified hold-out split, where 80% of the instances are used for training and the remaining 20% are reserved for testing was adopted. In this work, the memory of the models has been compared using the number of parameters. Also, the runtime has been compared for each model over 100 epochs on the given dataset. All experiments are conducted on a laptop with an NVIDIA GEFOTCE RTX 3070 Graphics Processing Unit (GPU) and an AMD Ryzen 9 5900HX processor.

4.2. Evaluation Metrics

Accuracy. One of the most important metrics in calculating the performance of a classification model is accuracy. If the predicted class matches the true class, the prediction is counted as correct. The predicted class is obtained by selecting the class with the highest logit value:

$$\hat{y} = \arg\max(\text{logits}, \text{dim} = 1).$$

The overall accuracy is then computed as

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}.$$

Due to the imbalance in the sea ice datasets even after augmentation, the evaluation metrics chosen for this study include the ROC curve. Other metrics used for evaluation include the AUPRC curve and the confusion matrix.

ROC Curve. We report multiclass ROC-AUC using the One-vs-Rest (OvR) strategy. For each class c, the class was treated as the positive class while all remaining classes are considered negative. The class-wise AUC value AUC_c is computed using the predicted probabilities, and the final score is obtained by macro-averaging across all classes:

$$\text{ROC-AUC}_{\text{OvR}} = \frac{1}{C}\sum_{c=1}^{C}\text{AUC}_c. \tag{10}$$

Confusion Matrix. The class-wise errors are analyzed using confusion matrices, where entry (r, c) counts the number of instances with true class r predicted as class c.

4.3. Hyperparameter Settings

SCM-Net uses CrossEntropyLoss (CEL) as the loss function, enabling comparisons between models in this study for multi-class image classification, since the compared models use the same loss function. CEL measures the difference between the predicted and ground-truth probability distributions and returns a scalar loss value. CLE uses the softmax activation function to convert logits to probabilities. CLE penalizes false positives heavily, which is of major importance for sea ice classification.

With softmax function, given logits $z_1, z_2, \ldots, z_C$, the probability for class j is given by

$$p_j = \frac{e^{z_j}}{\sum_{k=1}^{C} e^{z_k}}.$$

This makes all probabilities sum to 1. Since we have batches of data, we use CrossEntropyLoss with batches for a batch of size N:

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}\log(p_{i,y_i}).$$

Adam Optimizer (Adaptive Moment Estimation), which is popular in image classification using deep learning models, is used in this study. This optimizer combines the benefits of Momentum and RMSProp, which makes it capable of training faster and more stably. L2 regularization is applied to help mitigate overfitting by adding a penalty to the large weights.

5. Results

In this section, the accuracy and loss functions of the proposed model are compared with the state-of-the-art deep learning models. After that, the runtime and memory usage of these models are evaluated to demonstrate the superiority of the proposed approach.

In this paper, we applied several state-of-the-art models including EfficientNet, MobileNet, ResNet, VGG, Vision and Swin Transformers to the chosen dataset. A comparative analysis of the aforementioned performance on sea ice image classification was conducted, with particular focus on the Swin Transformer and hybrid CNN-based models. In this study, we propose SCM-Net, compare its accuracy with other models, and discuss its advantages and disadvantages. Furthermore, the paper demonstrated that the proposed lightweight models are suitable for real-time applications based on model inference time, parameter count, and other efficiency criteria.

Model comparison was conducted using ResNet, VGGNet, EfficientNet, MobileNet, and InceptionNet as CNN-based models. On the other hand, ViT and SwinT models are also considered for performance comparison with transformer-based models. Attention-based models illustrate higher accuracy across various vision tasks. In this study, we evaluate those claims by investigating the performance in terms of accuracy, memory usage, parameter count, and runtime against SCM-Net.

5.1. Overall Model Performance

In the Table 3, a comparison of the accuracies achieved by applying the PyTorch implementation of the *state-of-the-art* models and the implementation of the SCM-Net are presented. The other column of this table compares the loss functions of the aforementioned models. The table shows that while several baseline models, e.g., SwinWithBatchCNN and SwinWithCNN, achieve slightly higher accuracy, the proposed SCM-Net maintains competitive performance with an accuracy of 90% and a relatively low loss of 0.57. This indicates that SCM-Net offers a balanced trade-off between accuracy and loss, demonstrating its effectiveness compared to more complex state-of-the-art baseline models.

Table 3. Accuracy and loss comparison of different models on the dataset

Model Name	Accuracy (%)	Loss
SwinTWithCNN	94	0.38
SwinWithBatchCNN	95	0.35
RegularSwinT	94	0.40
MobileNet	90	0.61
ViT	90	0.59
EfficientNet	91.8	0.59
ResNet	92	0.56
DenseNet	90	0.61
VGG-16	85.8	0.68
ConvNet	85	0.67
SwinConvNext	75	0.78
SCM-Net	90	0.57

As shown in Fig. 3, the test accuracy improves throughout training with some fluctuations, ultimately reaching approximately 90% at convergence.

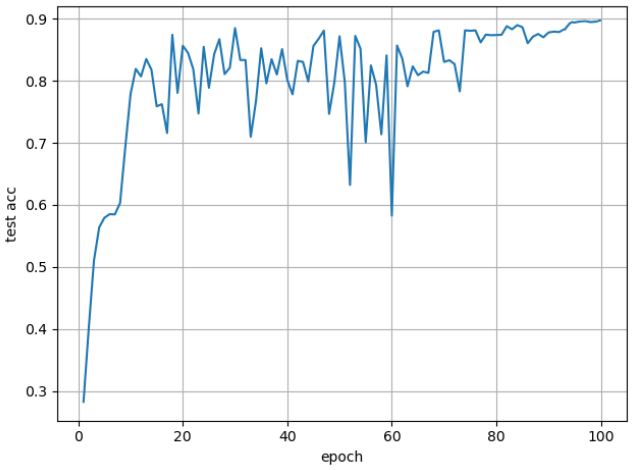


Figure 3. Accuracy of the SCM-Net model

As shown in Fig. 4, the proposed lightweight SCM-Net model achieves strong class-wise performance, with most predictions correctly aligned along the diagonal. This result demonstrates that reduced model complexity does not compromise classification accuracy.

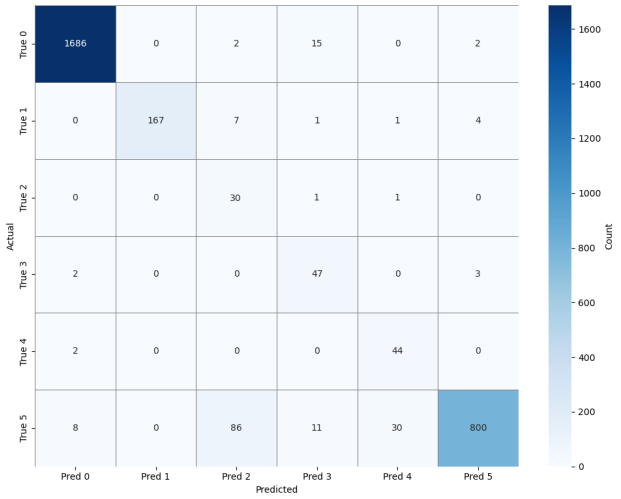


Figure 4. Confusion matrix of the SCM-Net model

Fig. 5 illustrates the ROC curves of the proposed SCM-Net model for all six classes. The images show that the area under the curve (AUC) values across classes are consistently higher in the proposed model, which demonstrates a strong discriminative capability. Most class-wise ROC curves are concentrated in the top-left corner, indicating high true-positive rates at low false-positive rates. Although Class 1 exhibits a comparatively lower AUC, its performance remains strong, while Classes 3 and 5 achieve near-perfect discrimination. Overall, these results demonstrate the robustness and reliability of the proposed model in multi-class classification.

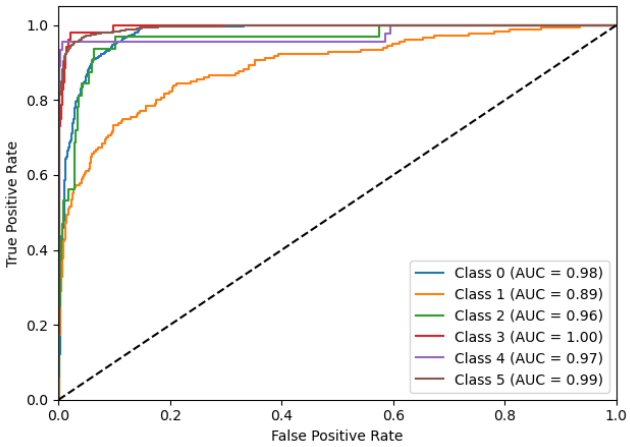


Figure 5. ROC of the SwinConv mixing hybrid model

5.2. Computational Complexity Comparison

When discussing model efficiency precisely, especially in the context of lightweight vision models such as the SCM-Net, it is important to distinguish between efficiency, computational complexity, memory complexity, and runtime performance. Computational complexity

is often used to divide memory consumption and runtime performance, which is often overlooked but critically important for deployment.

The amount of memory the GPU or CPU needs during both training and inference is often referred to as memory usage. This memory usage includes memory allocated for model parameters, intermediate feature maps, gradients, and optimizer states. Formally, this memory footprint for each model can be approximated as follows:

$$M_{\text{total}} = M_{\text{params}} + M_{\text{activations}} + M_{\text{gradients}} + M_{\text{optimizer}}, \tag{11}$$

where M_{params} denotes the memory required to store learnable parameters, $M_{\text{activations}}$ represents intermediate feature maps during forward propagation, $M_{\text{gradients}}$ corresponds to stored gradients during backpropagation, and $M_{\text{optimizer}}$ accounts for additional states maintained by the optimization algorithm (e.g., momentum terms in Adam).

In Table 4, the comparison includes both the static memory footprint of model parameters and the dynamic memory required for activations during training. It is shown that the proposed SCM-Net is significantly more lightweight than all baseline models, requiring only 49,768 parameters, 0.19 MB of Memory, and substantially fewer MACs.

Table 4. Comparison of model parameters, model size, MACs, and FLOPs across models

Model Name	Total Parameters	Model Size (MB)	MACs (M)	FLOPs (M)
SwinTWithCNN	21,351,378	81.96	36,990	73,970
SwinWithBatchCNN	25,059,922	96.12	43,670	87,330
SwinTransformer	27,587,184	105.63	2,990	5,990
MobileNet	2,231,558	8.64	6.66	13.33
ViT	87,459,846	333.63	2,950	5,900
EfficientNet	4,015,234	15.5	10.8	18.9
ResNet	23,520,326	90	84	170
VGG-16	134,256,126	512	433	865
ConvNet	2,472,454	9.9	42	84
SwinConvNext	92,809	0.37	10	20
SCM-Net	**49,768**	**0.19**	**12.77**	**20.54**

Another important factor to consider in memory usage comparisons is the number of FLOPs required by each model. FLOPs stands for the total number of floating-point operations a model performs to process a single input (e.g., one forward pass of an image). The types of operations included in this kind of counting are additions and multiplications (especially convolution operations), Matrix multiplication, and attention layers, which indicate the computational cost of a model and they are independent of hardware. Models with fewer FLOPs generally run faster and consume less energy per model, while actual runtime also depends on hardware efficiency and memory access patterns. Table 4 compares the FLOPs of various models and highlights the computational efficiency of the proposed model and emphasizes the ability of the model in terms of the FLOPs it uses compared to other models under the same hardware conditions.

Table 5 summarizes the architectural configuration in layers of the proposed SCM-Net model. It contains tensor dimensions, kernel sizes, and activation functions. Together, they provide insight into the model's structural design and overal computational complexity.

Table 5. Hyperparameters of SCM-Net model

	Layer Type	Tensor Size	Kernel Size	Activation
Stem	Input	$3 \times 32 \times 32$	-	-
	Conv3×3 (s=2)	$64 \times 16 \times 16$	3×3	ReLU
Block1	DWConv3×3	$64 \times 16 \times 16$	3×3	ReLU
	PWConv1×1	$64 \times 16 \times 16$	1×1	ReLU
Swin-T	LayerNorm	$64 \times 16 \times 16$	-	-
	SW-MHSA	$64 \times 16 \times 16$	$M_w = 4$	-
	LayerNorm	$64 \times 16 \times 16$	-	-
	MLP	$64 \rightarrow 128 \rightarrow 64$	1×1	GELU
Block2	DWConv3×3	$64 \times 16 \times 16$	3×3	ReLU
	PWConv1×1	$64 \times 16 \times 16$	1×1	ReLU
Head	Spatial Attention	$64 \times 16 \times 16$	7×7	Sigmoid
	PWConv1×1	$64 \times 16 \times 16$	1×1	-
	GAP	$64 \times 1 \times 1$	-	-
	FC	6	-	-

The input/output dimensions of the proposed model are shown in Table 6. Let H_0 and W_0 represent the input dimensions corresponding to the height and width, respectively, while the input channels are denoted by C_0. At every intermediate stage i of the network, the dimensions are represented by H_i and W_i, with C_i representing the channels corresponding to that stage.

Table 6. Complexity summary table

Comp.	Real Add. (C_A)	Real Mult. (C_M)
Conv3 × 3	$9H_1W_1C_1C_0$	$9H_1W_1C_1C_0$
Conv Block	$H_cW_c[12C_c + C'_cC_c + 3C'_c]$	$H_cW_c[11C_c + C'_cC_c + 2C'_c]$
Swin-T Block	$H_sW_s[(4+2r)C_s^2 + (2M_w^2 + r + 5)C_s + N_h(M_w^2 + 2)]$	$H_sW_s[(4+2r)C_s^2 + (2M_w^2 + 4)C_s + N_hM_w^2]$
Attention	$H_aW_a[5C_a + C'_aC_a + 16]$	$H_aW_a[2C_a + C'_aC_a + 18]$
Output	$C_o(H_oW_o + N_{out} - 1)$	$C_o(N_{out} + 1)$

Layer parameters which control the computational operations are denoted as follows: the kernel size is represented by k, where $k = 3$ corresponds to a 3×3 convolution, and s indicates the stride of the convolutional operation. For the Swin block, N_h denotes the number of attention heads in the SW-MHSA

mechanism, and $d_k = C/N_h$ represents the dimension per attention head. M_w denotes the window size of the Swin Transformer, and r illustrates the MLP layer expansion ratio. Final output of the classification layer, which is denoted by N_{out} classes.

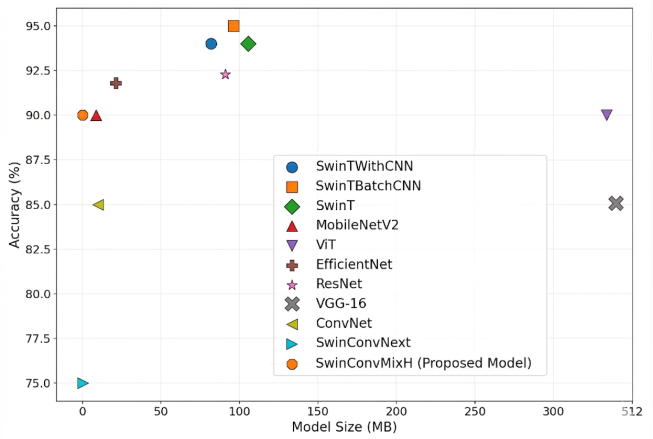


Figure 6. Model size vs. accuracy comparison

Figure 6 compares model accuracy against model size (in MB). This visualization highlights the trade-off between performance and memory footprint, showing that although larger models yield proportional gains in accuracy, the proposed models achieve good results despite their small model size. Results demonstrate that the proposed model (SwinConvMixH) achieves significantly l ower m emory u sage t han S win Transformer and MobileNet while maintaining competitive accuracy. This reduction comes primarily from the use of depthwise-pointwise convolutions, which replace heavy standard convolutions, and from the window-based self-attention mechanism in the Swin block, which reduces quadratic complexity and the number of parameters.

Fig. 7 compares the number of parameters of the proposed model with other state-of-the-art deep learning models. Results demonstrate that the proposed model (SCM-Net) achieves significantly lower amount of parameters than Swin Transformer and MobileNet while maintaining competitive accuracy.Replacing heavier standard convolutions with depthwise–pointwise convolutions and the window-based self-attention mechanism with a lightweight version of the Swin block reduces quadratic complexity and the number of parameters. In fact, Fig. 7 shows one of the strengths of the proposed SCM-Net model: despite requiring significantly f ewer p arameters, it demonstrates outstanding accuracy.

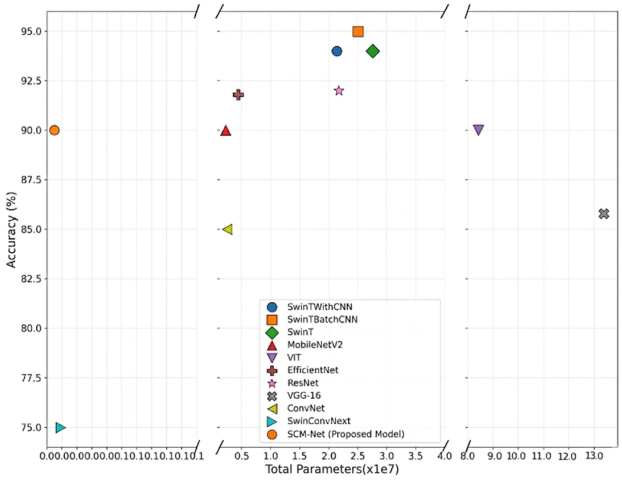


Figure 7. Accuracy vs. total parameters comparison

5.3. Runtime Comparison

Runtime performance is a critical factor in addition to memory efficiency, es pecially fo r re al-time or near-real-time applications. Runtime can be quantified as the inference time per sample or per batch, typically measured in milliseconds (ms), or as throughput in terms of frames per second (FPS). Mathematically, the average inference time can be expressed as

$$T_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} T_i, \tag{12}$$

where T_i denotes the processing time for the i-th sample and N is the total number of evaluated samples.

For further analysis between models, memory usage and runtime are also considered under the same hardware configuration. Models that consume less inference time and memory can be beneficial, as they can be used in limited-resource environments such as edge devices, real-time remote sensing applications, and embedded systems.

In this work, the runtime has been compared for each model over 100 epochs on the given dataset. Fig. 8 illustrates that the SCM-Net architecture achieves better runtime and memory usage than both deep CNN-based and attention-based models. All experiments are conducted on a laptop with an NVIDIA GeForce RTX 3070 GPU and an AMD Ryzen 9 5900HX processor, and the results are shown in minute-based.

As it is observable from the Fig. 8, there is a huge drop in the runtime of the proposed lightweight hybrid model. This reduction in runtime (completing 100 epochs in 20 minutes) can be attributed to the following main factors:

- This architecture contains fewer than 50K parameters, and fewer parameters directly translate to

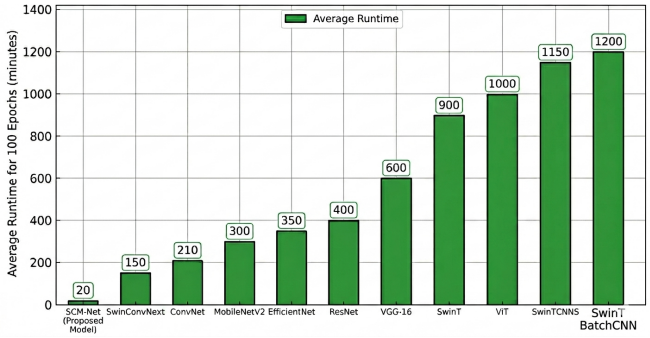


Figure 8. Runtime comparison

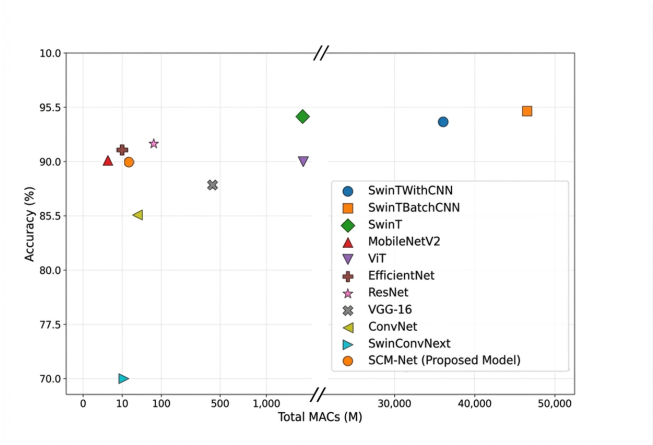


Figure 9. Accuracy vs. MACs

fewer multiplications. On the other hand, fewer memory transfers per iteration is the direct result of fewer multiplications, thereby the training time is hugely reduced.

- The core operators in the hybrid model are predominantly lightweight convolutional and mixing layers. These operations are highly optimized on GPUs. Compared to multi head self attention operations used in SwinT and ViT, they have far less computational overhead. The convolution operations scale linearly with input resolution, making them much more efficient for small inputs such as 32×32 images, while attention layers scale quadratically with input sequence length.

- The simplified structure of the hybrid models minimizes redundant operations such as repeated normalization and expensive matrix multiplications, thus reduces latency. Collectively, this hybrid model achieves a runtime that is an order of magnitude faster than conventional CNNs and transformers, while maintaining competitive accuracy.

Memory usage is important because deep learning models are often used in real-time applications and run on devices with limited resources (e.g., edge devices, mobile phones, embedded systems). It impacts deployability, scalability, and efficiency. Smaller memory usage may sometimes lead to lower accuracy, but it allows for better deployability. Multiply-Accumulate Operations (MACs) in millions is a measure of computation, i.e., how many multiplications and additions are needed for one forward pass of the model on a single input. It depends on both the model architecture and the input size. Generally, it was shown that the more computationally expensive a model is, the higher its MACs are. The model accuracy and MACs are compared for the model used in Fig. 9.

The ultimate goal of deep learning-based models is to be useful for real-time applications. When comparing deep learning-based methods, Runtime is one of the most important factors. In such settings, computational efficiency is ju st as critical as accuracy. Traditional convolutional networks such as VGG and ResNet often require more processing time due to their depth and high parameter count. On the other hand, lightweight architectures like MobileNet and EfficientNet we re de signed to reduce computational cost, enabling faster inference on limited hardware while maintaining competitive accuracy. Recent advances in image processing methods have introduced attention-based models, such as ViT and shifted window Transformers.

Transformer and deep convolutional architectures are computationally expensive due to their self-attention mechanisms and the larger number of parameters they use. To bridge the gap between accuracy and Runtime, hybrid approaches were introduced that combine lightweight convolutional layers with transformer mechanisms. Overall, the runtime comparisons showed that hybrid models are the better option for resource-constrained environments. In contrast, transformation-based models trade increased Runtime for gains in accuracy.

5.4. Inference Time Comparison

In addition to runtime, which is important for training a deep learning model, inference time is another critical factor that must be considered. In embedded platforms that needed instant responses, the model must perform inference with limited processing power, memory bandwidth, and energy.

SCM-Net is benchmarked against the state-of-the-art deep learning models used in the study, and the results are shown in Table 7. Compared to other models, SCM-Net achieves better inference time due to its hybrid,

lightweight architecture. Moreover, transformer-based models have the highest inference time among deep CNN-based models.

Table 7. Average inference time per sample using PyTorch

Model Name	Inference Time (ms)
ResNet152	8.240
MobileNet	4.100
ConvNet	1.649
DenseNet	4.605
EfficientNet	4.971
VGG16	1.614
Swin Transformer	7.642
Vision Transformer	8.061
SCM-Net	**0.941**

6. Conclusion

In conclusion, precise sea ice classification is needed to navigate safely and efficiently in polar regions, particularly during warmer seasons, when changing ice conditions pose a high risk to vessels. In this study, the SCM-Net model for classifying a six-class sea ice dataset has been proposed. Although other state-of-the-art models, such as Swin Transformers, achieve accuracies above 95%, their large number of parameters and high runtime make them less suitable for deployment on devices that require real-time decision-making. The proposed SCM-Net model achieved approximately 90% accuracy and was trained in only 20 minutes; it has fewer than 50,000 parameters, making it both lightweight and efficient. For a quick comparison, MobileNet, which achieves the same accuracy as the proposed model, has more than 2 million parameters. These are the key strengths of the proposed model, highlighting the hybrid architecture's efficiency and effectiveness.

Although the proposed model shows promising results for sea ice classification, several directions remain open for future research. Proposing more advanced hybrid models to achieve higher accuracy with fewer model parameters could be a useful future direction. Additional optimization of the SCM-Net model could be explored to improve accuracy while maintaining low computational cost. Overall, these directions offer significant potential to further enhance the efficiency and applicability of the proposed architectures in real-time environmental monitoring applications.

References

[1] R. Lindsey and M. Scott, "Climate change: Arctic sea ice summer minimum," https://www.climate.gov/news-features/understanding-climate/climate-change-arctic-sea-ice-summer-minimum, 2020, accessed on 8 October 2024.

[2] N. Zakhvatkina, V. Smirnov, and I. Bychkova, "Satellite sar data-based sea ice classification: An overview," *Geosciences*, vol. 9, no. 4, p. 152, 2019.

[3] I. Sudakow, V. K. Asari, R. Liu, and D. Demchev, "Meltpondnet: A swin transformer u-net for detection of melt ponds on arctic sea ice," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 15, pp. 8776–8784, Oct. 2022.

[4] Z. Zhang, G. Deng, C. Luo, X. Li, Y. Ye, and D. Xian, "A multiscale dual attention network for the automatic classification of polar sea ice and open water based on sentinel-1 sar images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 17, pp. 5500–5516, Jan. 2024.

[5] J. Zhang, W. Zhang, X. Zhou, Q. Chu, X. Yin, G. Li, X. Dai, S. Hu, and F. Jin, "Cnn and transformer fusion network for sea ice classification using gaofen-3 polarimetric sar images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 17, pp. 1234–1245, Sep. 2024.

[6] S. Khaleghian, H. Ullah, T. Kræmer, N. Hughes, T. Eltoft, and A. Marinoni, "Sea ice classification of sar imagery based on convolution neural networks," *Remote Sensing*, vol. 13, no. 9, p. 1734, Apr. 2021.

[7] J. Lohse, A. P. Doulgeris, and W. Dierking, "Mapping sea-ice types from sentinel-1 considering the surface-type dependent effect of incidence angle," *Annals of Glaciology*, vol. 61, no. 83, pp. 260–270, Jun. 2020.

[8] T. Zhang *et al.*, "Deep learning based sea ice classification with gaofen-3 fully polarimetric sar data," *Remote Sensing*, vol. 13, no. 8, p. Art. no. 1452, Apr. 2021.

[9] D. Kim, D. Park, J. Seo, H. Cho, S. Ahn, N. Kang, and S. Kim, "Estimation of electromagnetic field strength: Experiments using vision transformers," *IEEE Access*, vol. 13, pp. 195 735–195 748, Dec. 2025.

[10] O. Agbo-Ajala and S. Viriri, "A lightweight convolutional neural network for real and apparent age estimation in unconstrained face images," *IEEE Access*, vol. 8, pp. 162 800–162 808, Sep. 2020.

[11] M. Jiang, X. Chen, L. Xu, and D. A. Clausi, "Icegcn: An interactive sea ice classification pipeline for sar imagery based on graph convolutional network," *Remote Sensing*, vol. 16, no. 13, Jun. 2024.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2009, pp. 248–255.

[13] L. Papa, P. Russo, I. Amerini, and L. Zhou, "A survey on efficient vision transformers: Algorithms, techniques, and performance benchmarking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 12, pp. 7682–7700, Dec. 2024.

[14] T. C. Nauen, S. Palacio, F. Rauer, and A. Dengel, "Which transformer to favor: A comparative analysis of efficiency in vision transformers," *arXiv preprint arXiv:2308.09372*, Aug 2023. [Online]. Available: https://arxiv.org/abs/2308.09372

[15] L. Peng, Y. Cao, Y. Sun, and Y. Wang, "Lightweight adaptive feature de-drifting for compressed image classification," *IEEE Transactions on Multimedia*, vol. 26, pp. 6424–6436, Jul. 2024.

[16] H. Zhang, J. Liu, D. Jiang, Z. Luo, W. Liang, S. Li, X. Chen, X. Yuan, and S. Gao, "Lms-net: A light-weight network for mungbean salt stress identification," *IEEE Access*, vol. 12, pp. 131 701–131 713, Sep. 2024.

[17] Y. Zhang, Z. Lei, H. Yu, and L. Zhuang, "Imbalanced high-resolution sar ship recognition method based on a lightweight cnn," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, Jan. 2022.

[18] J. Kong and F. Zhang, "Sar target recognition with generative adversarial network (gan)-based data augmentation," in *Proc. 13th Int. Conf. Advanced Infocomm Technology (ICAIT)*. IEEE, Oct. 2021, pp. 215–218.

[19] G. F. Araujo, R. Machado, and M. I. Pettersson, "Synthetic sar data generator using pix2pix cgan architecture for automatic target recognition," *IEEE Access*, vol. 11, pp. 143 369–143 386, Dec. 2023.

[20] S. Du *et al.*, "Multi-category sar images generation based on improved generative adversarial network," in *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. Brussels, Belgium: IEEE, Jul. 2021, pp. 4260–4263.

[21] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 27, Dec. 2014, pp. 2672–2680.

[22] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, Jan. 2018.

[23] PyTorch Contributors, "Pytorch," https://pytorch.org/, accessed: Mar. 10, 2026.

[24] Z. Liu, Y. Lin, Y. Cao, H. Hu, R. Wang, Q. Guo, L. Zhang, and D. Lin, "Swin transformer: Hierarchical vision transformer using shifted windows," *arXiv preprint arXiv:2103.14030*, Mar. 2021.

[25] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, Apr. 2017. [Online]. Available: https://arxiv.org/abs/1704.04861