This is the introduction of the 94th solution for RSNA Screening Mammography Breast Cancer Detection. The objective of this competition is to detect breast cancer in screening mammograms acquired during routine screenings. The dataset utilized for training, validation, and testing consists of mammograms in DICOM format, along with accompanying information such as laterality, age, biopsy results, and more. The submissions are assessed based on the probabilistic F1 score (pF1). To attain the final outcome, we combine the SEResNeXt and ConvNeXtV2 models.

Summary:

- Images: Kaggle train data
- Preprocess: ROI cropping in CV Canny Detection cv2.drawContours
- Model: SEResNeXt and ConvNext v2

Extracting Row Data:

In order to minimize data extracting times, we use DALI for decoding docom using GPU. The details can be obtained from this website. DALI can be used to accelerate and save the space of training and test data. The procedure is like the following Fig. 1:

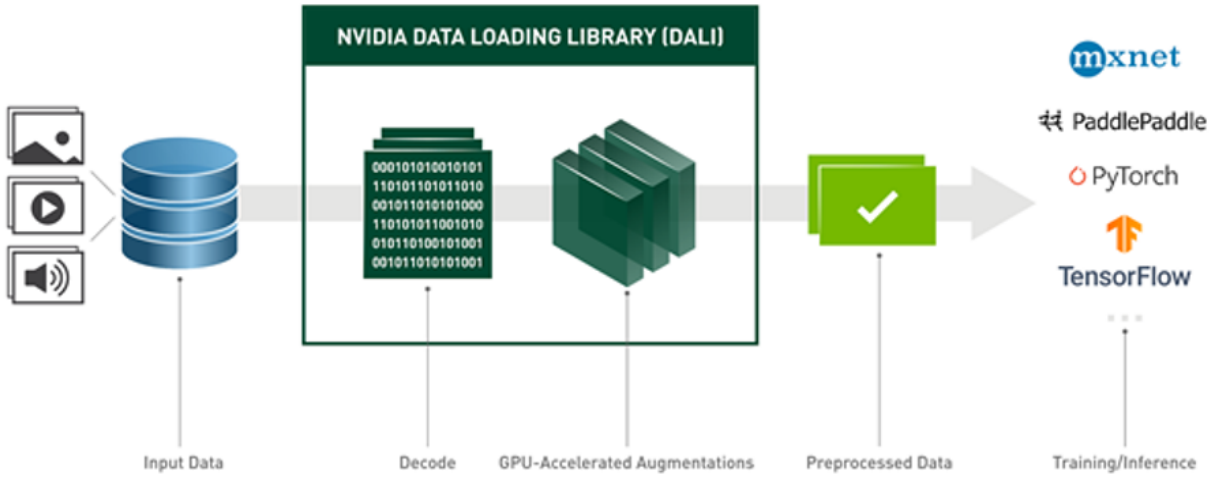

Fig. 1 Processing of DALI

Crop Data

Next, we will crop data to augment the image. The crop can be based on the offset of pixels as shown in Fig.2. The center points of the cropped image are overlapping areas of horizontal and vertical peaks. Fig.3 presents the comparison of original and cropped images. The reference of this section is RSNA: Cut Off Empty Space from Images. YOLO can also be a choice to detect the object, namely the breast in images. The reference of this section is Breast Cancer - ROI (brest) extractor. Finally, Open CV has its library to get the edge of objects. The reference of this section is NextVIT TensorRT Inference | Pytorch. We apply the third method to crop the objects.

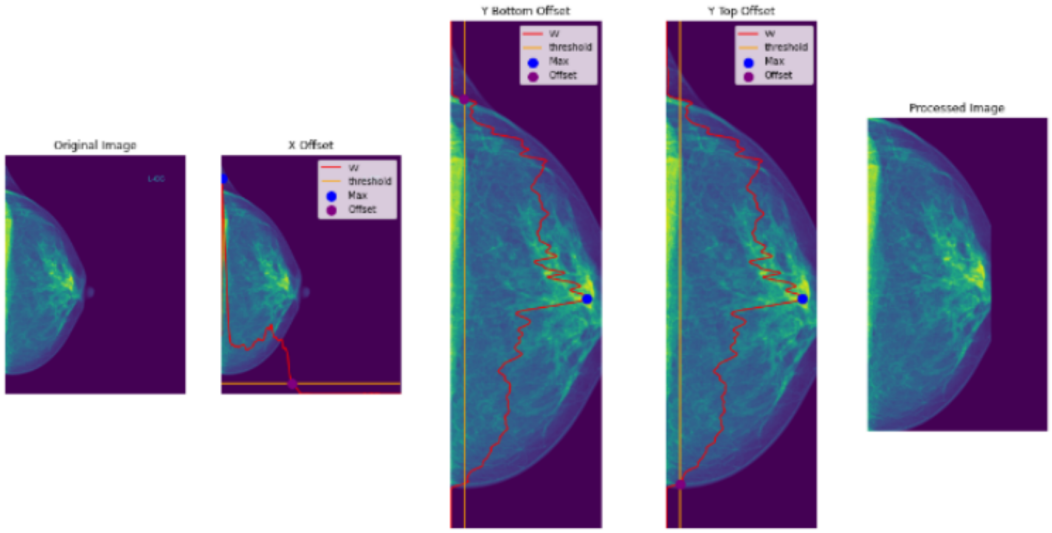

Fig 2. Offset of each image

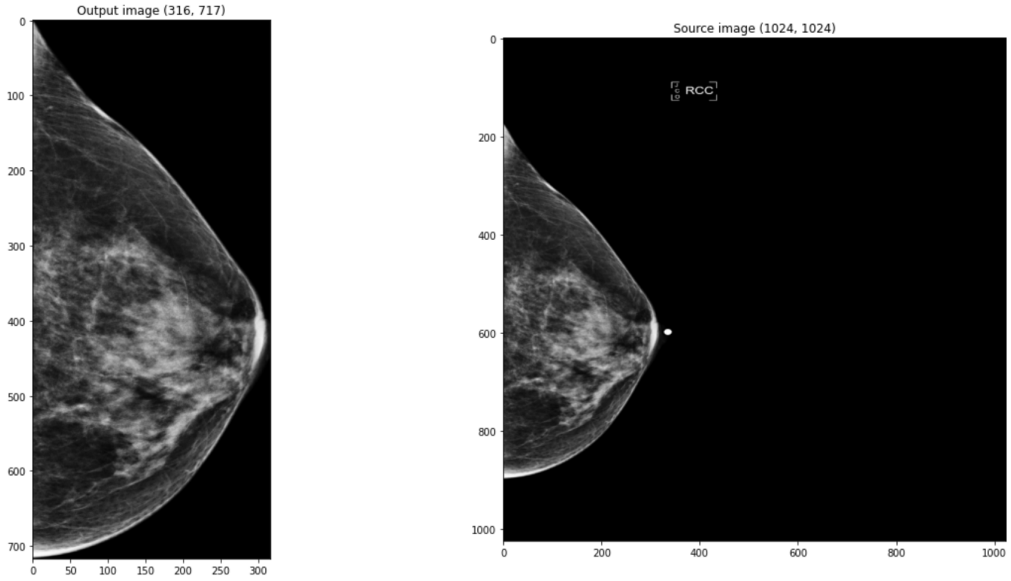

Fig 3. Comparison between original and cropped images

Sample Submission

Due to the imbalanced nature of the data, training the model on an unaltered training set may result in prediction biases. Fig. 4 illustrates the distribution disparity between positive and negative outcomes

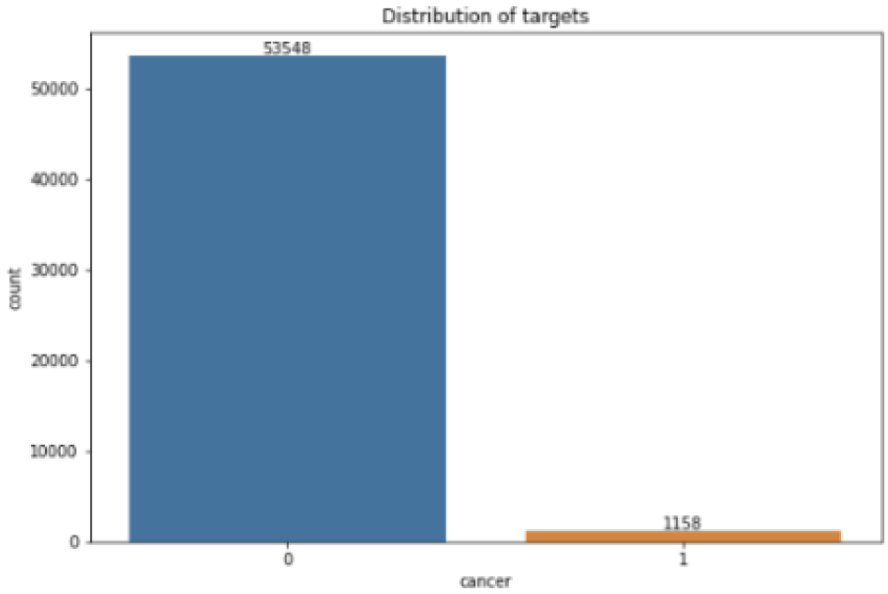

Fig.4 distribution of targets

To mitigate this bias, we performed sampling on the negative data. Specifically, we selected only 10% of the negative data for training purposes.

Model1

In the case of model 1, we utilize ConvNext v2 as our chosen approach. In the following sections, we will delve into the novel aspects and advancements introduced in both ConvNext and its updated version, ConvNext v2.

ConvNext

ConvNext is an innovative computer vision algorithm developed by Facebook AI Research (FAIR). It builds upon the foundation of ResNet50 or ResNet200 models. Several modifications have been introduced to enhance the ResNet architecture, such as Macro Design, ResNeXt-ify, inverted bottleneck, large kernel, and other specialized design elements. The optimized structure of ConvNext, as shown in Fig. 5, incorporates these modifications to improve its performance and efficiency.

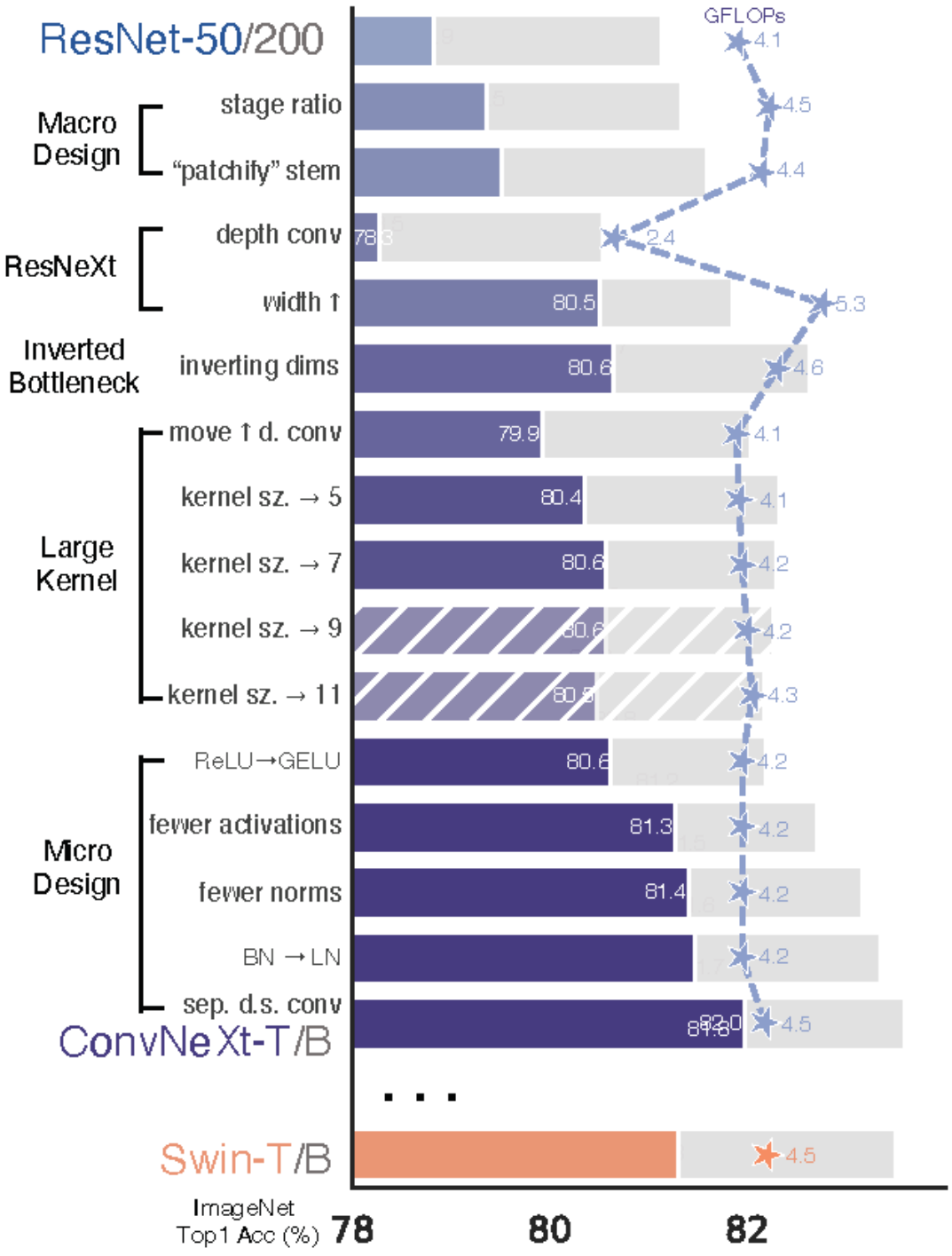

Fig.5 Improvement of ConvNext

Macro Design:
The ResNet architecture consists of stages with the following distribution: (3, 4, 6, 3). In the Swin-Transformer, the ratio of stages is either 1:1:3:1 or 1:1:9:1, with the latter being employed in larger models. On the other hand, ConvNeXt utilizes a stage configuration of 3:3:9:3.

ConvNext introduces a modification to the initial processing of input images by replacing the stem with Patchify. The stem cell design focuses on the initial processing of input images within the network. In traditional ConvNets and vision Transformers, a typical stem cell aggressively reduces the size of input images to generate an appropriate feature map size, taking advantage of the inherent redundancy present in natural images.

ResNeXt-ify:
The fundamental principle of ResNeXt is to utilize grouped convolution, which divides the convolutional filters into multiple groups. By increasing the number of groups and expanding the width of the network, ResNeXt achieves a higher computational speed while significantly reducing the number of floating-point operations (FLOPs) required.

Inverted Bottleneck:
The hidden dimension of the MLP block in ConvNext is four times larger than the input dimension. The design of ConvNext is influenced by the inverted bottleneck design commonly used in ConvNets, which incorporates an expansion ratio of 4.

Large Kernel:
The kernel size used in ConvNext is 7×7. In theory, a larger kernel size leads to a broader receptive field. By increasing the kernel size from 3×3 to 7×7, the network's performance improves from 79.9% to 80.6%, while the computational operations (FLOPs) required by the network remain relatively unchanged.

Specific Optimization:

1. Replacing ReLU with GELU
2. Fewer activation functions:
   In a Transformer block, there are several components, including key/query/value linear embedding layers, a projection layer, and an MLP block consisting of two linear layers. Notably, there is only a single activation function used within the MLP block. In contrast, it is customary to apply an activation function to each convolutional layer, including the 1×1 convolutions, in other contexts.
3. Fewer normalization layers
4. Replacing BN with LN
5. Separate downsampling layers:
   In ResNet, the spatial downsampling is achieved by the residual block at the start of each stage, using 3×3 conv with stride 2 (and 1×1 conv with stride 2 at the shortcut connection)

Performance:

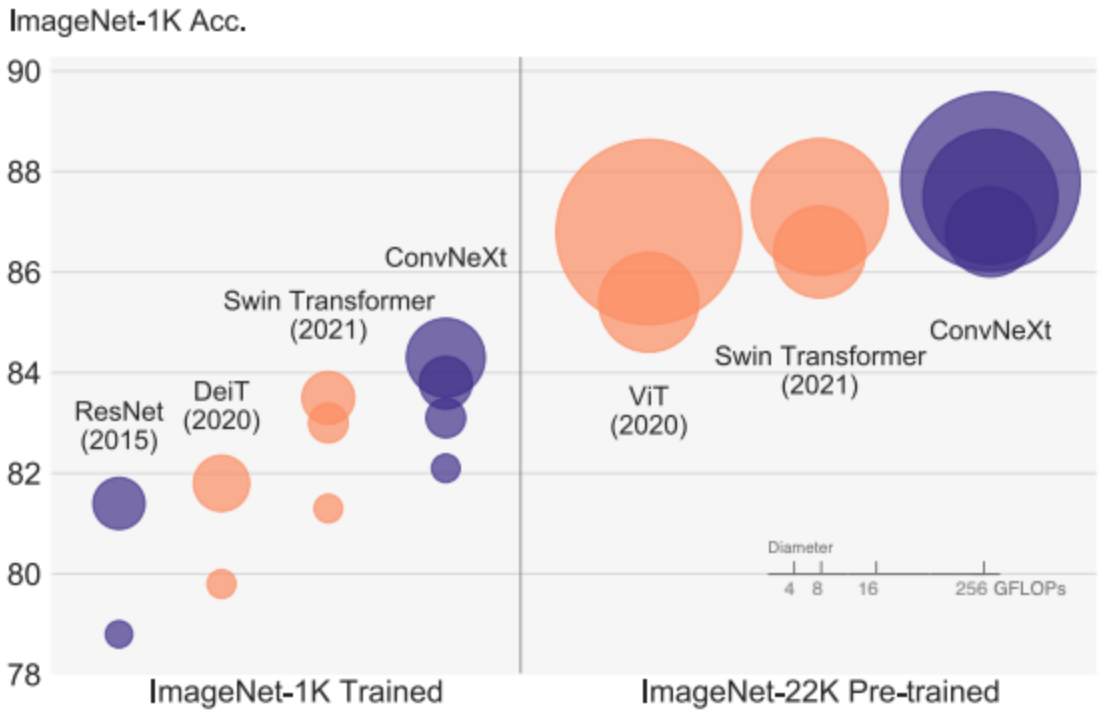

Fig.6 Performance of ConvNext

ConvNext v2

The update of ConvNext v2 are Fully Convolutional Masked Autoencoder and Global Response Normalization.

Fully Convolutional Masked Autoencoder:
Our approach is conceptually simple and runs in a fully convolutional manner. The learning signals are generated by randomly masking the raw input visuals with a high masking ratio and letting the model predict the missing parts given the remaining context. Fig. 7 provides a visual representation of this process.

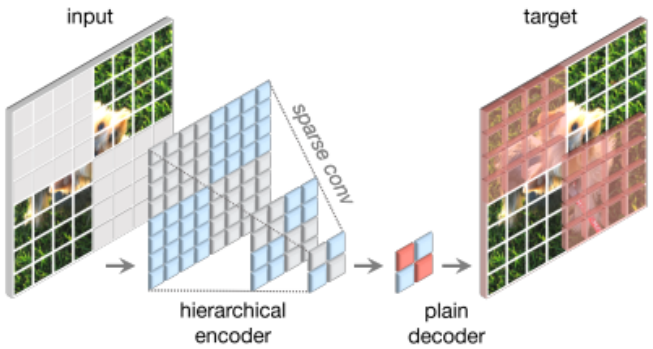

Fig.7 Fully Convolutional Masked Autoencoder

Global Response Normalization:
To enhance the effectiveness of FCMAE (Fully Convolutional Masked Autoencoder) pretraining in conjunction with the ConvNeXt architecture, we incorporate the Global Response Normalization (GRN) technique. This technique aims to optimize the performance of FCMAE by applying normalization across the entire network response.

The performance is shown in Fig.8.

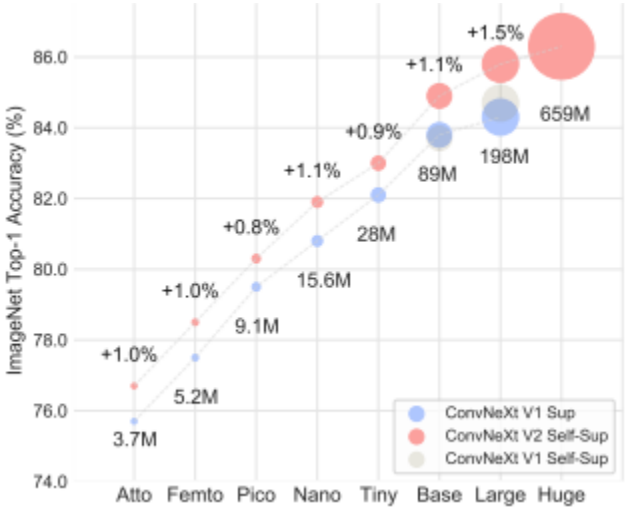

Fig.8 CovNext v2 performance

Model2

We also incorporated the SEResNeXt model into our study. Similar to its name suggests, SEResNeXt is a modified version of ResNext that incorporates squeeze-and-excitation blocks. These blocks allow the network to dynamically recalibrate channel-wise features, enhancing its performance.

ResNeXt:

ResNext introduces a novel fully connected layer that redefines the actions of neural networks as a combination of splitting, transforming, and aggregating. The splitting operation divides the input vector x into low-dimensional embeddings, while the transforming operation calculates wx. Finally, the aggregating layer combines these embeddings together.

The shortcut is also introduced into ResNext. The details can be seen in Fig.9.

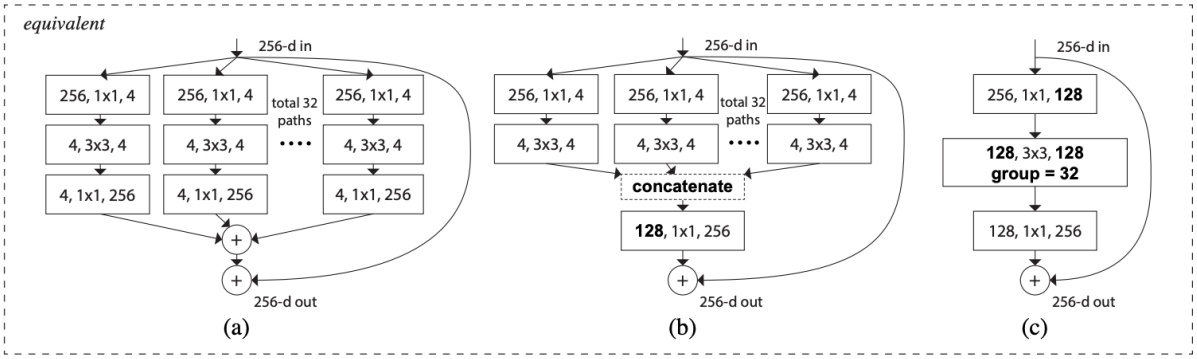

Fig.9. Equivalent building blocks of ResNeXt

Group convolution has other features. Splitting is essentially done by the grouped convolutional layer when it divides its input channels into groups. The grouped convolutional layer in Fig. 9(c) performs 32 groups of convolutions whose input and output channels are 4-dimensional. The grouped convolutional layer concatenates them as the outputs of the layer.

SE:

In deep neural networks, different channels in different feature maps usually represent different objects. Channel attention adaptively recalibrates the weight of each channel, and can be viewed as an object selection process, thus determining what to pay attention to. Hu et al. first proposed the concept of channel attention and presented SENet for this purpose.

SE blocks are divided into two parts, a squeeze module and an excitation module. Global spatial information is collected in the squeeze module by global average pooling. The excitation module captures channel-wise relationships and outputs an attention vector by using fully-connected layers and non-linear layers (ReLU and sigmoid). Then, each channel of the input feature is scaled by multiplying the corresponding element in the attention vector. Details can be seen in Fig. 10 and Fig. 11.

The first is the Squeeze operation, which performs feature compression from the spatial dimension, turns the h*w*c feature into a 1*1*c feature, and obtains a vector with a global receptive field to some extent, and the number of output channels Matching the number of input feature channels, it represents the global distribution of responses on feature channels. The algorithm is very simple, it is a global average pooling.

The second is the Excitation operation, which generates weights for each feature channel by introducing the w parameter, where w is a multi-layer perceptron, which is learnable, and undergoes a dimensionality reduction in the middle to reduce the amount of parameters. And through a Sigmoid function to obtain the normalized weight between 0 and 1, to complete the explicit modeling of the correlation between feature channels.

The last is a Scale operation. The weight of the Excitation output is regarded as the importance of each feature channel after selection, and the channel width is multiplied and weighted to the previous features to complete the original feature in the channel dimension.

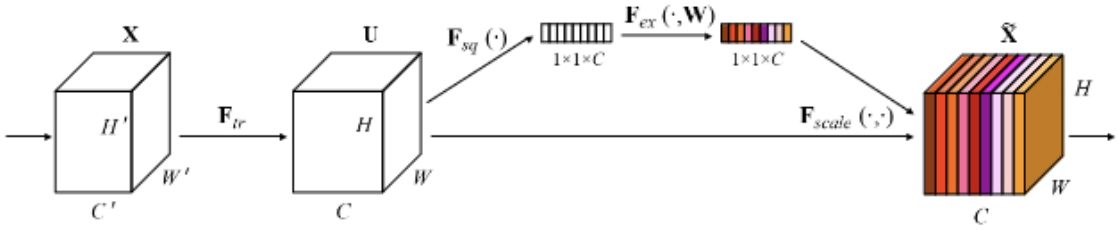

Fig.10 structure of SENet

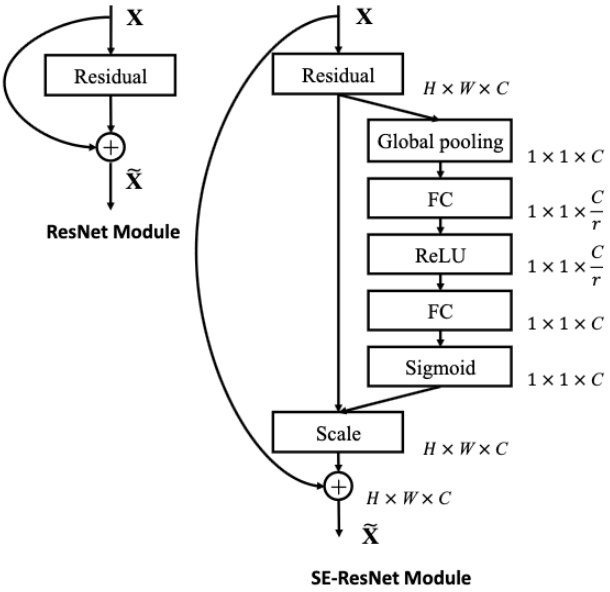

Fig.11 structure of residual

Merge Two Result

Lastly, we allocate 50% of the weights to each model and make predictions based on a threshold.