We have provided a detailed description of the four networks and a comparison between them in a supplementary file. Our code and the labelled data are also released in Github, which would be helpful to readers and could be applied in other studies.

Below are the details of our methods, which are provided in the supplementary file.

ResNet. We know that the deeper the network is, the more information can be obtained. However, as the network deepens, the optimisation becomes less effective and the accuracy of the test and training data would decrease. This is due to the problem of gradient explosion and gradient disappearance caused by deeper networks. It is common to normalise the input data and the data in the intermediate layers, which ensures that the network uses stochastic gradient descent (SGD) in back propagation, thus allowing the network to converge. However, this is only useful for networks of a few dozen layers and is useless when the network goes further down. ResNet was proposed in 2015 to address the problem of overfitting in deep neural networks. In ResNet's neural network, the problem of gradient disappearance is mitigated by proposing a residual structure (Figure 1of He et al., 2016), allowing the neural network to build ultra-deep network structures with more than 1000 layers. In this neural network, the dropout (randomly deactivated partial neurons) method is abandoned and the batch normalisation method is used.

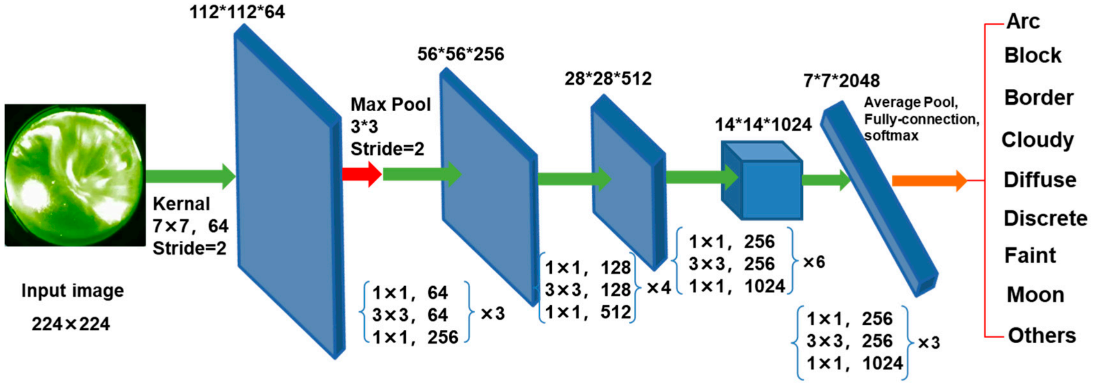
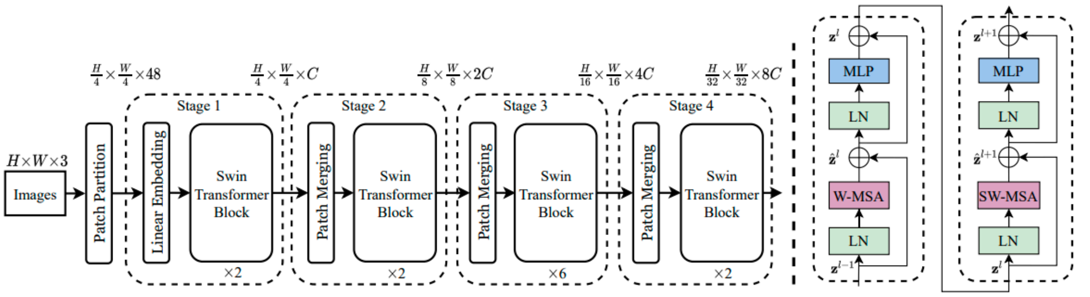


Figure S1. CNN structure of ResNet-50

Figure S1 shows the main structure of a 50-layer ResNet neural network. As can be seen from the figure, the length and width of the extracted feature layers become smaller as the network progresses, but the number of channels increases, indicating that deep learning neural networks can extract more abstract and complex features that are likely to extract the essential features of the data.

Swim Transformer. Since the Transformer's breakthrough in natural language processing (NLP) tasks, computer scientists have been trying to use the Transformer in the CV field. Most of the hyperparameters commonly found in CNN networks can also be manually adjusted in Swin Transformer, such as the number of network blocks, the number of layers in each block, the size of the input image, etc. The network architecture is cleverly designed and is a wonderful application of the Transformer to the image domain. the input to the Swin Transformer is the original size of the image, e.g. 224*224 for ImageNet. in addition the Swin Transformer uses the most common hierarchical network structure used in CNNs. A particularly important aspect of CNNs is that as the network hierarchy deepens, the receptive field also expands, and this is also satisfied in Swin Transformer.

Swin Transformer proposes four network frameworks, from small to large, namely Swin-T, Swin-S, Swin-B and Swin-L. In this study, Swin-T was used for training, and the structure of Swin-T is shown in Figure S2(a). Swin Transformer Block, which is shown in Figure S2(b).



(a) Architecture (b) Two Successive Swin Transformer Blocks

Figure S2. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (From Liu et al.,2021)

The network structure of the Swin Transformer is very simple, consisting of four stages and an output header, and is very easy to extend. the network framework of the four stages of the Swin Transformer is the same, and each stage has only a few basic super-references to adjust, including the number of hidden nodes, the number of network layers, the number of multi-headed self-attentive heads, the downsampling scale, etc. These hyperparameters will be shown in detail in our code.

MobileVit. It combines the advantages of CNN and ViT to build a lightweight, generic and mobile-friendly network model. Taking a different perspective allows Transformer to process information as a convolution. MobileViT significantly outperforms CNN and ViT based (e.g. Mobilenetv1, MNASNet and MixNet) networks on different tasks and datasets. It also has better generalisation capability, which is the gap between training and evaluation metrics. For two models with similar training metrics, the model with better evaluation metrics is more generalisable because it can better predict unseen datasets. Compared to CNN, which has poor generalisation ability even with extensive data augmentation, MobileViT shows better generalisation ability.

The aim of MobileViT Block is to model the local and global information in the input tensor with a small number of parameters. For a given input tensor$X \in R^{H\times w\times C}$, MobileViT first applies an n×n standard convolutional layer and then generates features$X_L \in R^{H\times w\times d}$ using 1×1 convolutional layer. n×n convolutional layers encode local spatial information, while point convolution projects the tensor into a higher dimensional space (d-dimensional, where d>c) by learning a linear combination of input channels.

ConvNeXt. ConvNeXt is an improved version of the classical ResNet50/200 network based on some of the advanced ideas of the Transformer network, which combines the advantages of both networks by introducing some of the latest ideas and techniques of the Transformer network into the existing modules of the CNN network. The main optimisations made are: (1) increase the number of training Epochs from 90 to 300; (2) change the optimizer from SGD to AdamW; (3) more sophisticated data expansion strategies, including Mixup, CutMix, RandAugment, Random Erasing, etc.; (4) add regular strategies, such as random depth, label smoothing, EMA, etc. More specific hyperparameters for pre-training and fine-tuning can be found in our code. The improvements made to ConvNeXt and the result of these improvements are shown in Figure S3.

Both the ResNet and Swin-T networks have four stages, and the ratio of stacked blocks in each stage is 1:1:3:1 for Swin-T and 1:1:9:3 for Swin-L. From this, we can see that the third layer of the Transformer network has more stacks, so the ConvNeXt network adjusts the number of stacks in each stage of ResNet from (3, 4, 6, 3) to (3, 3, 9, 3) in this ratio, and keeps the ratio at 1:1:3:1. The ConvNeXt network replaces the stem layer with the same convolutional kernel size of 4 and step size of 4 as the Swin-T network, with a small improvement in accuracy and GFLOPs. The MLP module in the Transformer network is similar to the Inverted Bottleneck module in MobileNet V2 in that it has a "thin end, thick middle" anti-bottleneck structure. Consequently, the ConvNeXt network is designed with a similar Inverted bottleneck structure in mind.

At the same time moving up the depth convolution layer and increasing the number of convolution kernels, and replacing the RELU with the more common GELU activation function, using only one activation function and one regularisation function in each block, with the regularisation function again BN replaced with LN, and a separate downsampling layer at the end designed to perform separate downsampling operations on the features.

With these five operations referenced above from the Swin-T network, the existing modules in the classic ResNet50 network were transformed into a new ConvNeXt network. The new network achieves a significant improvement in accuracy for the same parameter size.

We compared four algorithms, ResNet-50, ConvNeXt, Swim Transformer and MobileViT, with training times of 1.9h, 4.0h, 3.8h and 1.4h. In all of the above algorithms, we use transfer learning, which has the advantage of being able to improve the accuracy of the initial training and significantly reduce the training time. If we did not use transfer learning, we would need many more iterations and the training time would be much longer. We ran the above code on a laptop with a four-core Intel i7-1165G7 CPU @ 2.80GHz.

So our different algorithms have their own strengths. When we want to spend less training or the computer performance is insufficient, we can choose the MobileViT and

ResNet-50 algorithm. When we want to pursue higher accuracy, we can choose ConvNeXt. Or when we focus on different auroral subcategories, we choose different networks (Figure 3). For example, when we focus on moon, we can choose Swim Transformer, and when studying diffuse and discrete, we can choose ConvNeXt algorithm.

The F1-score combines the precision and recall of a classifier into a single metric by taking their harmonic mean. It is primarily used to compare the performance of two classifiers. Suppose that classifier A has a higher recall, and classifier B has higher precision. In this case, the F1-scores for both the classifiers can be used to determine which one produces better results. The maximum value of F1 is 1, and the minimum value is 0. The larger the value, the better the performance.

More details about the algorithm are shown in our released code.

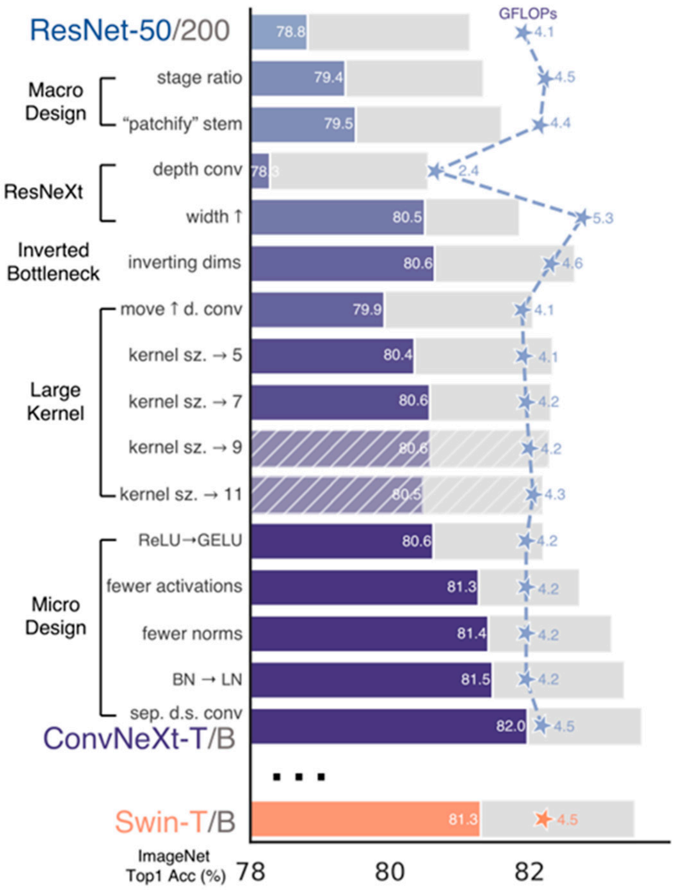


Figure S3. ConvNeXt algorithm improvements and results (from Liu et al.,2022)