Supplementary material:

Here, we added more information about Convolutional Neural Network in Section"**<u>MATERIALS AND METHODS</u> 2.2 Convolutional Neural Network** ".

<u>MATERIALS AND METHODS</u>

2 Deep Learning Technique

2.2 Convolutional Neural Network

In the following paragraphs, we discuss these layers in sequence: first, the convolutional layer, then, the BN layer, the ReLU layer, the max-pooling, and average-pooling layers, and finally, the fully connected layer. Then, we present the hyperparameter settings for our model.

2.2.1 Convolutional Layer

The convolutional layers takes a 3-D volume $X \in \mathbb{R}^{W_X \times H_X \times D_X}$ as input. For example, if the input is an image, W_X and H_X are its width and height, and D_X is the depth or the number of channels of the image: $D_X$is 1 for a grayscale image and 3 for an RGB color image. The output of each convolutional layer is also a 3-D volume with reduced width and height but increased depth[1].

A convolutional layer uses filters (or kernels) to perform convolution operations on the input volume. A filter can be regarded as a neuron with learnable weights that can learn to capture visual features, e.g., edges and contours, from the input. When performing a convolution operation, each filter is connected to only a local region of the input, i.e., that neuron will only try to capture a specific visual feature from a local region. The filter size, which can be regarded as the receptive field of the neuron, is the same as the size of the local region. Specifically, in our convolutional layers, the filter depth was equal to the depth D_x of the input volume, but the width and height of the local region were usually smaller than W_x and H_x, respectively. A convolution operation is the dot product of a filter and a local region of the input volume followed by a bias $b \in \mathbb{R}$ offset. Given a filter with weights $W \in \mathbb{R}^{f \times f \times D_X}$ and a local region $R \in \mathbb{R}^{f \times f \times D_X}$, the convolution operation between them is as follows:

$$Conv(W, R) = \sum_{k=0}^{D_X} \sum_{j=0}^{f} \sum_{i=0}^{f} W_{ijk} R_{ijk} + b,,$$

where W_{ijk} is the weight of the filter and R_{ijk} is the pixel value of the local region of the input. Intuitively, the resulting convolution value $\mathrm{Conv(W, R)}$ measures the probability that the visual feature matching the one the filter has learned appears in the local region.

A convolution operation between a filter and a local region can only capture whether the local region contains a particular visual feature. To look for the visual feature in the entire image, we first performed zero-padding around the border of the input volume to preserve the information at the border during convolution. Then, we slid the filter over the image horizontally and vertically and conducted convolution operations on all the possible local regions. The step size of the slide was specified by a hyperparameter named "stride." This process produces a matrix of convolution values, called a *kernel map*. Additionally, a convolutional layer usually uses a batch of filters to capture different visual features. The *kernel maps* of all the filters form a *feature map*, which is the final output of the convolutional layer.

2.2.2 Batch Normalization and ReLU Layers

The BN layer was adopted to cope with the *Internal Covariate Shift* problem in deep neural network models[2]. It also helps avoid overfitting and accelerates deep neural network training. This layer takes the *feature map* output by the preceding convolutional layer as input and learns to normalize the *feature map*.

The input to the ReLU layer is the normalized *feature map*. It applies an elementwise activation function (in this case, the max function with a zero threshold) to the input as follows:

$$ReLU(V) = max(0, V).$$

The ReLU layer's output dimensions are the same as those of its input.

2.2.3 Pooling Layer

The pooling layer was used to progressively reduce the spatial size of the input volume to reduce the number of parameters and the computation required in the subsequent model layers, which helps control overfitting. To reduce the spatial size, the pooling layer leveraged a filter to summarize the local regions in each depth slice of the input. In contrast to convolutional layer filters, the filter in a pooling layer implements a fixed function instead of a function with learnable weights. In each depth slice, the pooling layer also slid the filter over the width and height of the input volume with a stride of S and summarized every local region to form a *kernel map*. The *kernel maps* of all the depth slices are used to construct an output *feature map*. Because the pooling layer operates independently in each depth slice, the depth dimensions of its output are the same as those of its input.

Our model included two types of pooling layers: a max-pooling layer and an average-pooling layer, whose filters implemented the max function and the average function, respectively. This meant that given a local region $R \in \mathbb{R}^{W \times H}$ in a depth slice, these filters' outputs for this region were the maximum and average value of this region, respectively.

2.2.4 Fully Connected Layer

The layers before the fully connected layer were expected to extract sufficient features from an image to perform classification. The fully connected layer is responsible for projecting the extracted features to a probability distribution for classification. The fully connected layer includes n neurons, where n is the number of classes, and each neuron corresponds to a specific class. When receiving an input volume, this layer first flattens it into a feature vector $V \in \mathbb{R}^{m}$. Then, it performs the following linear transformation on V:

$$Y = WV + B,$$

where $W \in \mathbb{R}^{\mathrm{m}\times\mathrm{m}}$ and $B \in \mathbb{R}^{\mathrm{m}}$ are learnable weights. Finally, the softmax function was applied to normalize Y so that Y_i could represent the probability of the input image belonging to class C_i. This softmax process is as follows:

$$P(C_i) = softmax(y_i) = \frac{exp^{y_i}}{\sum_{j=1}^{n} exp^{y_i}}.$$

2.2.5 Hyperparameter Settings

The convolutional layers and the pooling layers require several hyperparameters whose settings are shown in **Supplementary Table 1**. We selected these values based on the deep residual network architecture that won the first place in the ILSVRC 2015 classification competition[3].

Reference

1. Zeiler MD, Fergus R. Visualizing and understanding convolutional networks, In European conference on computer vision, Springer, 2014.
2. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 2015.
3. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition, In Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.

Supplementary Table 1 The Hyper-parameters in our model

Layer	#Filters	Filter Size	Stride	Padding
Convolutional layer 1	32	7x7x3	2	3
Convolutional layer 2	64	3x3x32	2	1
Convolutional layer 3	128	3x3x64	2	1
All max-pooling layers	/	3x3	2	1
Average-pooling layer	/	8x8	1	/