

Received 30 November 2023, accepted 17 December 2023, date of publication 25 December 2023,
date of current version 25 January 2024.

Digital Object Identifier 10.1109/ACCESS.2023.3347331

RESEARCH ARTICLE

Deep Residual Networks With a Flask-Like Channel Structure

DONGYAO LI¹, YUNHUI PAN¹, SHUHUA MAO²,
MINGSEN DENG¹, (Member, IEEE), AND HUJUN SHEN^{1,3}

¹School of Information, Guizhou University of Finance and Economics, Huaxi, Guiyang, Guizhou 550025, China

²School of Science, Wuhan University of Technology, Wuhan 430070, China

³Guizhou Provincial Key Laboratory of Computational Nano-Material Science, Guizhou Education University, Guiyang 550018, China

Corresponding authors: Mingsen Deng (msdeng@mail.gufe.edu.cn) and Hujun Shen (hujun.shen@gznc.edu.cn)

This work was supported in part by the Guizhou Provincial Key Technology Research and Development Program under Grant QKHZC[2023]YB185, in part by the Program for Innovative Research Team of Guizhou Province under Grant QKHPTRC(2020)5023, and in part by the National Natural Science Foundation of China under Grant 22167007.

ABSTRACT The development of deep residual network (ResNet) has contributed significantly to the progress of computer vision and image classification, expanding the applicability of convolutional neural networks to different fields. Researchers continue to improve the classification accuracy of ResNet by increasing parameter sizes or model complexity. However, enlarging the network parameter size would significantly increase the training workload. In this work, we adjusted the scale and variation pattern of ResNet18 channel numbers, evaluated their performance differences using different datasets, and designed a flask-like channel structure, which enabled ResNet18 to reduce model parameters while maintaining accuracy. Then, we use MNIST and STL10 datasets validate the effectiveness of FLC structure. Finally, we extend the FLC structure to other ResNet models with different layers, such as ResNet34, ResNet50, ResNet101, and ResNeXt. By testing these ResNet models on the CIFAR10 dataset, our experiments showed that the ResNet models with the FLC structure (namely ResNet_FLC) can maintain or improve the accuracy of the model by approximately 1% while reducing the number of model parameters and FLOPs.


INDEX TERMS Deep residual network, deep learning, channel size, parameter size, image classification accuracy.

I. INTRODUCTION

Convolutional Convolutional Neural Networks (CNNs) are the most popular deep learning neural networks used in computer vision [1], [2], [3], [4], natural language processing [5], [6], audio processing [7], and video processing [8]. Multiple convolutional and pooling layers in CNNs facilitate the better capturing of image features, making them a popular choice for image classification [1], [2], target detection, face recognition [3], and image segmentation [4]. The earliest CNN architecture, named LeNet [9], was proposed by LeCun et al. in 1998. The LeNet's structure contains an input layer, a convolutional layer, a fully connected layer, and an output layer, where the convolutional layer is specific to CNNs. Researchers have investigated the convolution, activation,

and pooling structures of the convolutional layer to develop a variety of complicated and high-performing CNNs. In 2012, Krizhevsky et al. [10] introduced AlexNet, which contains five convolutional layers and three fully connected layers, and achieved significant success in image classification. ZFNet [11], proposed in 2013, improved AlexNet by employing deconvolutional visualization. In 2014, GoogleNet [12] utilized its Inception module core for multi-scale feature extraction and fusion, which is more effective in image characterization. Furthermore, VGGNet [13] attained better performance by incorporating more layers.

The development of CNNs has shown that a model's performance would positively correlate with the network's depth within a specific range. Nevertheless, the model's accuracy would be deteriorated with increasing depth when the depth exceeds a specific range, leading to network degradation. In 2016, He et al. [14] provided a deep

The associate editor coordinating the review of this manuscript and approving it for publication was Zhan-Li Sun .

residual network (ResNet) to address the problems of gradient explosion, gradient disappearance, and network degradation caused by increasing depth. ResNet uses residual links for cross-layer information propagation by adding “jump links” in convolutional neural networks (CNNs). These links avoid information decay and distortion of data in deep networks, improving the training effectiveness and accuracy of the network. As a result, ResNet makes it possible to train extremely deep networks, which is essential for image classification. Inspired by ResNet, researchers have significantly improved representation ability, classification accuracy, and parameter size by increasing network depth, expanding network width [15], and refining network structure [16], [17]. In particular, image classification accuracy, defined as the ratio of the number of images correctly predicted by the classifier to the total number of images, is the most intuitive indicator for evaluating classification performance [18].

Huang et al. [19] proposed a training method called stochastic depth (SD) to enhance the image classification accuracy of ResNet models. In the training process, the deep network randomly dropouts some layers, thereby becoming a shallow network, while the original deep network is still present in the testing process. This treatment effectively reduces training time while enhancing model accuracy. Convolutional Residual Memory Networks (CRMNs) [20] use long short-term memory (LSTM) algorithm to train a residual network with deep layers, which showed excellent performance in testing on the CIFAR-100 dataset. To enhance ResNet performance, Targ et al. [21] presented the RiR network by adding a small ResNet to each residual block, thereby increasing the network depth and nonlinear capability. The Wide Residual Network (WRN) [15] improves network expression and accuracy by increasing channel sizes. ResNeXt network, developed by Xie et al. [22], improves model accuracy by augmenting the number of paths with the same topology. Meanwhile, Gao et al. [23] proposed the Res2Net network, which constructs hierarchical residual-like connections to represent multi-scale features within a single residual block and increase the receptive field range of each network layer, thus achieving strong feature representation and high prediction accuracy. Shen et al. [24] presented the Weighted Residual Network (WRResNet) by introducing trainable weight parameters to boost ResNet’s performance. Likewise, the ResNeSt method proposed by Zhang et al. [25] decomposes feature maps into subspaces, which allows each subspace to learn its features before concatenating the features from all subspaces. These improvements of ResNet can help to extract rich and distinctive features. However, the complexity of these models requires highly demanding network parameters, resulting in a significant computational workload.

In order to reduce the parameter size, Huang et al. [26] proposed Densely Connected Convolutional Networks (DenseNet) by connecting each layer to other layers in a feedforward manner. For each layer, the feature maps of all preceding layers are used as inputs, and its feature

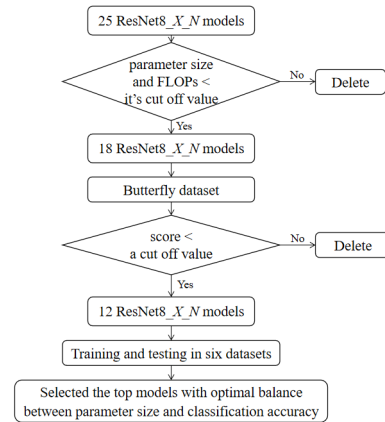


FIGURE 1. Flow chart for selecting the top models with optimal balance between parameter size, FLOPs and classification accuracy.

maps are used as inputs into all subsequent layers, which enhances the reuse of features and significantly reduces the number of parameters. Chen et al. [27] developed the Dual Path Network (DPN) by integrating the strengths of ResNet and DenseNet. DPN employs dual paths to explore new features, considerably reducing parameter size while ensuring the extraction of rich features. Han et al. [28] presented Pyramid Residual Networks (PyramidNet), which incrementally increases the feature map dimension of all ResNet units in a pyramid-like model. Appropriate feature map dimensions for each unit can curtail the parameter size in the model. Nonetheless, it is necessary to perform more systematic research on how the channel structure in ResNet impacts model accuracy and parameter size.

This work aims to systematically evaluate the impact of various channel structures on the performance of ResNet models by following the flow chart in Figure 1. This paper is organized as follows. In section II, we presented our 25 ResNet18_X_N models and the selected models used to test the Butterfly50 dataset’s performance. In Sections III and IV, six publicly available datasets (Print0-9, CIFAR10, Vegetable, Sign-50, Sign-85, and Time datasets) were used to validate and analyze the performance of the selected models. In Sections V, we verify the effectiveness of the FLC structure. Finally, in Section VI, we present our conclusions and outlook.

II. BUILDING AND SCREENING MODELS

A. BUILDING RESNET18_X_N MODELS

The original ResNet18 model (Figure 2a) consists of 17 convolutional layers and one fully connected layer. The 18 layers can be divided into five channel regions: channel-1 (green), channel-2 (blue), channel-3 (orange), channel-4 (yellow), and channel-5 (purple). One can modify the ResNet18 model’s channel structures by adjusting the channel size in each channel region. For instance, we generated different ResNet18_X_N models (Figure 2b) by altering the channel sizes in five regions (channel-1 through channel-5). In these

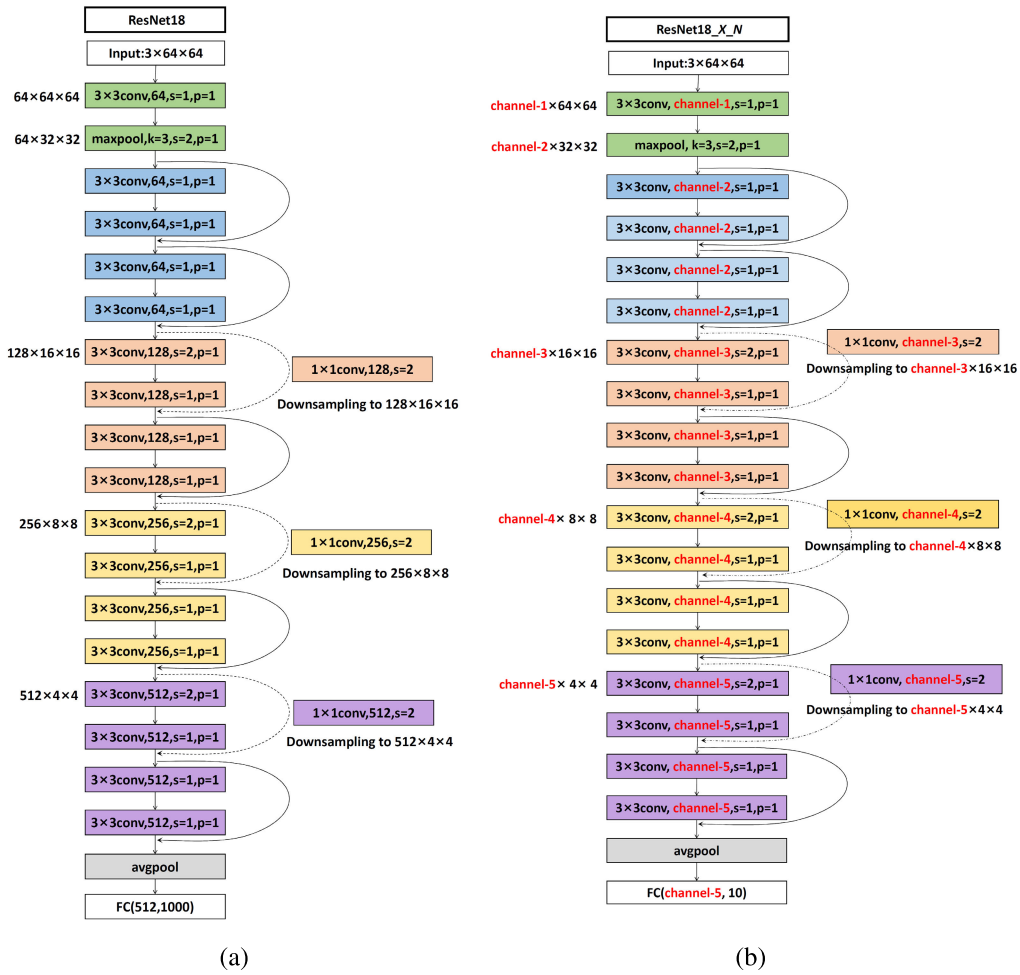


FIGURE 2. The framework of (a) original ResNet18 model and (b) ResNet18_X_N models with different channel structures. The 18 layers of the ResNet18 model are divided into five regions (channel-1 through channel-5), each of which is indicated by green, blue, orange, yellow, and purple colors, respectively. The variable X represents the type of the first channel transformation (channel1) while the variable N corresponds to the type of the subsequent four channel transformations (channel-2 through channel-5).

ResNet18_X_N models, the variable X defines the first channel type (channel-1), such as $X = C8, C32, C64,$ and $C128$. It should be noted that the stacking of multiple layers of small convolutions can have the same receptive field as the large convolution kernel, and can introduce more nonlinearity. Therefore, we use the 3×3 convolution kernel at all positions in ResNet18. The variable N corresponds to the type of the subsequent four channel regions (channel-2 through channel-5). Figure 3 illustrates the five different types of channel structures with $N = 1, 2, 3, 4,$ and 5 . For example, when $N = 1$, the variation of the ratio of the channel number in the five regions concerning the first channel region is $1 \rightarrow 11 \rightarrow 12 \rightarrow 14 \rightarrow 18$ (funnel-like channel structure). When $N = 2$, the variation is $1 \rightarrow 11 \rightarrow 12 \rightarrow 14 \rightarrow 12$ (flask-like channel structure). When $N = 3$, the variation is $1 \rightarrow 12 \rightarrow 14 \rightarrow 18 \rightarrow 116$ (pyramid-like channel structure). When $N = 4$, the variation is $1 \rightarrow 12 \rightarrow 14 \rightarrow 12 \rightarrow 11$ (diamond-like channel structure). When $N = 5$, the variation is $1 \rightarrow 12 \rightarrow 14 \rightarrow 18 \rightarrow 14$ (vase-like channel structure).

These 5 channel transformations belong to the category of channel quantity arrangement and combination. Finally, we would have 25 types of ResNet18_X_N models, in which the channel numbers, parameter sizes and the floating point operations (FLOPs) are given in Table 1.

B. SCREENING RESNET18_X_N MODELS

Among the 25 models for ResNet18, the original ResNet18 model is equivalent to the ResNet18_C64_1 model (shown as a black and bold style in Table 1). From Table 1, it is seen that the parameter size of ResNet18_C64_1 is 42.63MB and FLOPs is 2.23G. In this work, we selected the models with smaller parameter sizes and smaller FLOPs, and deleted those models where parameter sizes and FLOPs are all greater than ResNet18_C64_1's (shown as a red style in Table 1), which yielded 18 models.

We tested the 18 models and compared their classification accuracy (Acc), F1 value (F1), 95% confidence interval

TABLE 1. ResNet18_X_N Model channel numbers, parameter size and floating point operations (FLOPs). ResNet18_C64_1 is the original ResNet18 model (black bold), the models with both parameters size and FLOPs greater than ResNet18_C64_1 are marked in red, and these models will be removed.

ResNet18_X_N models	Channel size Channel-1	Channel-2	Channel-3	Channel-4	Channel-5	#.Param.(MB)	FLOPs
ResNet18_C8_1	8	8	16	32	64	0.67	0.04G
ResNet18_C8_2	8	8	16	32	16	0.22	0.03G
ResNet18_C8_3	8	16	32	64	128	2.67	0.14G
ResNet18_C8_4	8	16	32	16	8	0.22	0.07G
ResNet18_C8_5	8	16	32	64	32	0.85	0.11G
ResNet18_C16_1	16	16	32	64	128	2.68	0.14G
ResNet18_C16_2	16	16	32	64	32	0.85	0.11G
ResNet18_C16_3	16	32	64	128	256	10.65	0.54G
ResNet18_C16_4	16	32	64	32	16	0.86	0.29G
ResNet18_C16_5	16	32	64	128	64	3.37	0.42G
ResNet18_C32_1	32	32	64	128	256	10.67	0.56G
ResNet18_C32_2	32	32	64	128	64	3.39	0.44G
ResNet18_C32_3	32	64	128	256	512	42.56	2.16G
ResNet18_C32_4	32	64	128	64	32	3.43	1.14G
ResNet18_C32_5	32	64	128	256	128	13.47	1.67G
ResNet18_C64_1	64	64	128	256	512	42.63	2.23G
ResNet18_C64_2	64	64	128	256	128	37.93	3.31G
ResNet18_C64_3	64	128	256	512	1024	170.12	8.62G
ResNet18_C64_4	64	128	256	128	64	13.7	4.53G
ResNet18_C64_5	64	128	256	512	256	53.81	6.67G
ResNet18_C128_1	128	128	256	512	1024	170.38	8.89G
ResNet18_C128_2	128	128	256	512	256	54.07	6.94G
ResNet18_C128_3	128	256	512	1024	2048	680.24	34.42G
ResNet18_C128_4	128	256	512	256	128	54.75	18.07G
ResNet18_C128_5	128	256	512	1024	512	215.12	26.61G

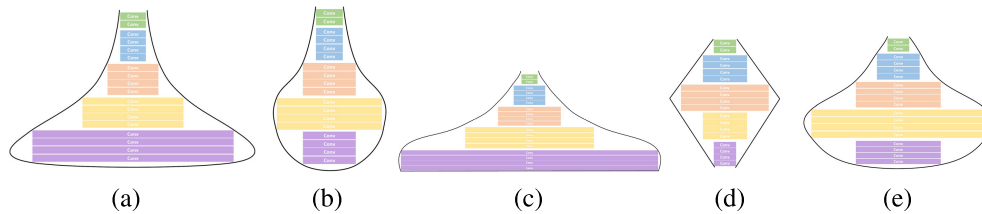


FIGURE 3. The ResNet18_X_N models with different values of N (channel structure): (a) funnel-like channel structure when N = 1, (b) flask-like channel structure when N = 2, (c) pyramid-like channel structure when N = 3, (d) diamond-like channel structure when N = 4, and (e) vase-like channel structure N = 5.

of AUC (CI_AUC) and confidence interval size of AUC (CI_Width) using various iterations (e.g., 20, 40, and 60). In our tests, we used 7,433 images downloaded from the Butterfly50 dataset, including 7,183 images for the training data and 250 images for the test data, all with an image size of $3 \times 224 \times 224$, as shown in Figure 4.

To evaluate a single category, we define TP (True Positive) as the number of instances where the model correctly predicts the positive class, FP (False Positive) as the number of instances where the model incorrectly predicts the positive class when the actual class is negative, TN (True Negative) as the number of instances where the model correctly predicts the negative class, and FN (False Negative) as the number of instances where the model incorrectly predicts the negative class when the actual class is positive. Then, we calculated Acc and F1 values using the following formula.

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \quad (1)$$

$$F1 = \frac{2PR}{P + R} \quad (2)$$

$$P = \frac{TP}{TP + FP} \quad (3)$$

$$R = \frac{TP}{TP + FN} \quad (4)$$

Calculate the AUC value for various iterations(e.g., 20, 40, and 60) and calculate the CI_AUC based on the AUC of all iterations. The calculation formula for the CI_AUC and CI_Width are as follows.

$$CI_AUC = (mean_auc - z \times \frac{std}{\sqrt{n}}, mean_auc + z \times \frac{std}{\sqrt{n}}) \quad (5)$$

$$CI_Width = 2 \times z \times \frac{std}{\sqrt{n}} \quad (6)$$

Among them, $mean_auc$ is the mean of AUC, std is the standard deviation of AUC, and n is the number of AUCs, which is the epoch value. z is the quantile of the standard



FIGURE 4. Representative images obtained from the Butterfly50 dataset.

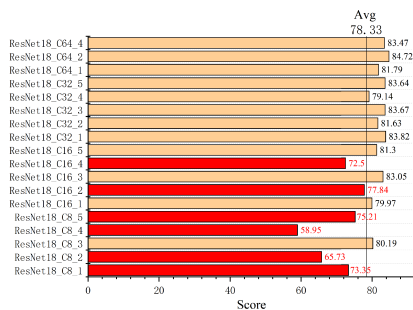


FIGURE 5. The calculated scores of 18 ResNet18_X_N models used in the testing of Butterfly50 dataset. The average score of 18 models is 78.33. It is seen that the scores of six models are below the average score.

normal distribution. When the confidence level is 95%, $z = 1.96$, which can be found in the normal distribution table.

The calculation steps for each model’s rating are shown in the following formula. We believe that the smaller the confidence interval, the better the model, so we assign a negative weight to CI_Width , as shown in formula 7. calculate the $Score_epoch_i$ (Formula 7, epoch = 20, 40, and 60) of each model based on CI_Width_i and acc_avg_i, fl_avg_i in Table 2. Finally, calculate the final score for each model as shown in formula 8.

$$Score_epoch_i = \frac{acc_avg_i}{2} + \frac{fl_avg_i}{2} - CI_Width_i \tag{7}$$

$$Score_i = \frac{Score_20_i}{6} + \frac{Score_40_i}{3} + \frac{Score_60_i}{2} \tag{8}$$

We display the scores of the model in Figure 5. Six models scored below the average (Figure 5) and will not be used in further testing. Consequently, we selected the top 12 models

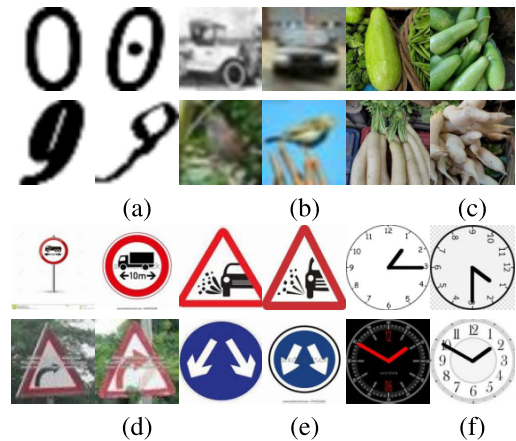


FIGURE 6. Representative images of the six public datasets downloaded from Kaggle: (a)Print0-9, (b)CIFAR10, (c)Vegetable, (d)Sign-50, (e)Sign-85, and (f)Time.

(shown as a bold style in Table 2) as candidates for the subsequent experiments.

III. EXPERIMENTS

A. DATASETS

The Kaggle datasets (<https://www.kaggle.com/datasets/>) offer a massive collection of free data that data scientists and machine learning competitions frequently employ. For our experiments, we chose five public datasets from Kaggle, namely, Print0-9, CIFAR10, Vegetable [29], Sign-50, Sign-85, and Time datasets. Table 3 provides a specific description of the six datasets, and the representative images used in our experiments are depicted in Figure 6.

B. EXPERIMENT SETTINGS

We preprocessed all six datasets by adjusting the image size, random graying, tensor conversion, and standardization. However, we did not augment the data in any way that could enhance image representation or the training models’ robustness. Between each convolution layer and the active layer, we adopted batch normalization (BN [30]). Then, we initialized weights for random samples from a normal distribution with a mean of 0 and a standard deviation of 0.1. Please note that we trained the plain/residual nets from scratch. In our experiments, we employed the cross-entropy loss function, set the batch size to 256, and used SGD as the optimizer with a learning rate of 0.1, a momentum of 0.9, and a weight decay of 0.0001 in training and testing. Because all the datasets were small classification datasets, only the top-1 error rate was evaluated. The algorithm for training and testing is presented as Algorithm 1.

IV. RESULTS AND DISCUSSION

A. THE IMPACT OF CHANNEL-1 ON THE PERFORMANCE OF RESNET18_X_N MODELS

Tables S2-S7(in Supplementary material) present the calculated results for the six datasets, including the averaged

TABLE 2. The average accuracy (acc_avg) and average F1 values (f1_avg) for five independent experiments, 95% Confidence Interval of AUC (CI_AUC) and 95% Confidence Interval Size of AUC (CI_Width) for one of the experiments, which obtained from the experiments using the Butterfly50 dataset at epoch = 20,40,60.

epoch	acc_avg (%)			f1_avg (%)			CI_AUC			CI_Width		
	20	40	60	20	40	60	20	40	60	20	40	60
ResNet18_C8_1	67.68	79.2	81.68	66.52	78.82	81.21	(86.93, 96.91)	(92.88, 97.94)	(94.96, 98.07)	9.98	5.06	3.11
ResNet18_C8_2	58.48	70.08	76.4	57.19	69.43	75.94	(84.66, 95.7)	(91.53, 96.85)	(94.44, 97.72)	11.04	5.32	3.28
ResNet18_C8_3	72.32	86.24	90.16	71.5	85.99	89.98	(85.33, 97.39)	(94.22, 98.99)	(95.43, 99.37)	12.1	4.77	3.94
ResNet18_C8_4	48.24	64.8	69.12	47.3	63.45	67.8	(84.06, 93)	(91.53, 96.32)	(93.47, 96.79)	8.94	4.79	3.32
ResNet18_C8_5	66	82.96	83.76	65.12	82.2	83.36	(85.89, 96.55)	(93.13, 98.07)	(95.33, 98.54)	10.66	4.94	3.21
ResNet18_C16_1	72.48	86.24	88.88	72.01	85.86	88.71	(87.15, 97.55)	(93.9, 99.23)	(95.9, 99.26)	10.4	5.33	3.36
ResNet18_C16_2	71.84	83.36	86.4	70.28	83.11	86.11	(86.68, 96.21)	(92.56, 97.83)	(95.72, 98.69)	9.53	5.27	2.97
ResNet18_C16_3	75.52	89.36	91.52	73.91	89.17	91.39	(87.56, 98.27)	(93.92, 98.97)	(96.67, 99.35)	10.7	5.05	2.68
ResNet18_C16_4	61.84	77.76	82.8	60.47	77.13	82.4	(87.95, 95.73)	(92.91, 97.45)	(94.08, 98.17)	7.78	4.54	4.09
ResNet18_C16_5	73.52	85.92	90.32	73.36	85.6	90.13	(88.83, 98.02)	(94.07, 98.67)	(95.55, 98.86)	9.19	4.6	3.31
ResNet18_C32_1	80.32	89.04	92.24	76.69	88.91	92.08	(87.30, 97.46)	(95.15, 99.36)	(96.19, 99.53)	10.2	4.21	3.34
ResNet18_C32_2	70.32	89.04	88.96	71.08	88.89	88.75	(89.32, 98.04)	(95.12, 99.25)	(96.06, 99.15)	8.72	4.13	3.09
ResNet18_C32_3	75.44	89.36	91.84	75.57	89.19	91.71	(88.19, 98.23)	(93.92, 99.13)	(96.98, 99.44)	10	5.21	2.46
ResNet18_C32_4	70.72	83.52	89.36	69.05	83.05	89.22	(86.88, 97.25)	(94.12, 98.47)	(95.58, 98.95)	10.4	4.35	3.37
ResNet18_C32_5	75.2	89.2	92.8	76.02	88.97	92.71	(88.93, 97.81)	(93.75, 98.99)	(95.29, 99.19)	8.88	5.24	3.9
ResNet18_C64_1	73.36	88.88	91.12	72.83	88.55	90.97	(87.72, 97.84)	(93.36, 98.67)	(94.93, 99.04)	10.1	5.31	4.11
ResNet18_C64_2	75.28	90.24	93.44	75.47	90.01	93.35	(88.67, 97.7)	(95.01, 99.39)	(95.94, 99.33)	9.03	4.38	3.39
ResNet18_C64_4	76.08	89.6	92.88	73.62	89.36	92.78	(90.60, 98.06)	(93.90, 99.2)	(94.92, 99.08)	7.46	5.3	4.16

TABLE 3. The six public datasets used for our experiments.

Datasets	category	Total	Training	Testing	resolution
Print1-9	10	10,160	9,140	1,020	28×28
CIFAR10	10	60,000	50,000	10,000	32×32
Vegetable	15	18,000	15,000	3,000	224×224
Sign-50	50	3,870	2,887	983	Unfixed
Sign-85	85	5,726	4,438	1,288	Unfixed
Time	144	12,960	11,520	1,440	224×224

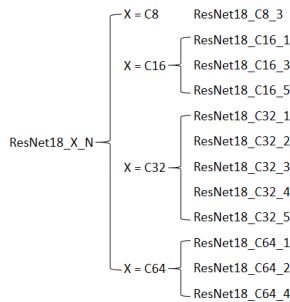


FIGURE 7. Four groups of the ResNet18 models: ResNet18_C8, ResNet18_C16, ResNet18_C32, and ResNet18_C64.

error (error_avg) and F1 value (f1_avg) obtained from five independent experiments. Based on variable X in channel-1, the selected ResNet18 models are divided into four groups (i.e., ResNet18_C8, ResNet18_C16, ResNet18_C32, and ResNet18_C64) shown in Figure 7. We computed the average error (X_error_avg) and F1 value (X_f1_avg) for each group and displayed them in Table 4. Additionally, Figure 8 depicts the impact of the number of iterations on X_error_avg.

In the calculated results of the Print0-9 dataset, it can be seen that the performance of the ResNet18_C16 and ResNet18_C64 groups is superior to the other two groups. However, for the three datasets (CIFAR10, Vegetable, and

Time), the ResNet18_C64 group significantly outperforms the ResNet18_C16 group. In the Sign-50 and Sign-85 datasets, the performance of the four groups is similar. Thus, our work suggests that the ResNet18_C64 group should be the best choice for the six testing datasets, and the channel size in the channel-1 region should preferably be set to 64 (or X = 64). In addition, based on Table 1 and Figure 7, we find that the average parameter size of the ResNet18_C64 group is 31.42 MB.

B. THE IMPACT OF VARIABLE N ON THE PERFORMANCE OF RESNET18_X_N MODELS

According to the variable N, the ResNet18 models can be divided into five groups (Figure 9), such as ResNet18_X_1, ResNet18_X_2, ResNet18_X_3, ResNet18_X_4, and ResNet18_X_5. Figure 3 illustrates that the five groups represent distinct channel structures, such as funnel-like, flask-like, pyramid-like, diamond-like, and vase-like channels. For each channel type, we calculated the average error (N_error_avg) and F1 value (N_f1_avg), as shown in Table 5. In addition, we determined the score of each group using the following algorithm (Algorithm 2).

By analyzing the final scores in Table 5, we observed that the ResNet18_X_2 group (with a flask-like channel structure) outperforms all other groups (or other channel structures). It can be seen from Table 1 and Figure 9 that the ResNet18_X_2 group has an average parameter size of 20.66 MB. In addition, our analysis reveals that the original ResNet18 model is equivalent to the ResNet18_C64_1 model (with a funnel-like channel structure), which has a parameter size of 42.63 MB. Table 1 indicates that the parameter size of the ResNet18_C64_2 model is 37.93 MB, a reduction compared to that of the original ResNet18 model (or ResNet18_C64_1 model).

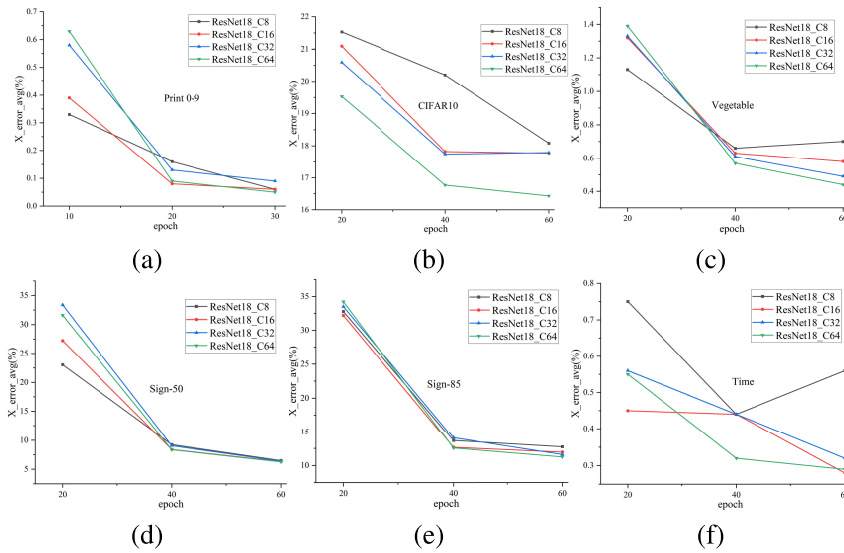


FIGURE 8. The impact of the number of iterations on X_error_avg using different datasets (a) Print0-9, (b) CIFAR10, (c) vegetable, (d) Sign-50, (e) Sign-85, (f) Time.

TABLE 4. The calculated results of the X_error_avg and X_f1_avg values for four groups of models, where X = C8, C16, C32 and C64.

Dataset		X_error_avg(%)				X_f1_avg (%)			
		X = C8	X = C16	X = C32	X = C64	X = C8	X = C16	X = C32	X = C64
Print0-9	epoch=10	0.33	0.39	0.58	0.63	99.67	99.61	99.42	99.37
	epoch=20	0.16	0.08	0.13	0.09	99.84	99.92	99.87	99.91
	epoch=30	0.06	0.06	0.09	0.05	99.94	99.94	99.91	99.95
CIFAR10	epoch=20	21.53	21.09	20.59	19.53	78.37	78.85	79.42	80.45
	epoch=40	20.2	17.82	17.74	16.77	79.84	82.14	82.21	83.18
	epoch=60	18.08	17.77	17.79	16.43	81.87	82.18	82.14	83.51
Vegetable	epoch=20	1.13	1.32	1.33	1.39	98.87	98.68	98.67	98.61
	epoch=40	0.66	0.63	0.61	0.57	99.34	99.37	99.39	99.43
	epoch=60	0.7	0.58	0.49	0.44	99.3	99.42	99.51	99.56
Sign-50	epoch=20	23.05	27.22	33.37	31.6	75.25	70.87	64.2	66.44
	epoch=40	9.24	8.39	9.05	8.35	90.33	91.31	90.61	90.4
	epoch=60	6.48	6.32	6.39	6.25	93.52	93.68	93.61	93.75
Sign-85	epoch=20	32.78	32.19	33.5	34.23	65.34	66.03	64.2	63.64
	epoch=40	13.66	12.65	14.07	12.56	86.1	87.08	85.69	87.18
	epoch=60	12.73	11.95	11.6	11.24	87.06	87.86	88.18	88.56
Time	epoch=20	0.75	0.45	0.56	0.55	99.25	99.55	99.46	99.46
	epoch=40	0.44	0.44	0.44	0.32	99.56	99.57	99.57	99.68
	epoch=60	0.56	0.28	0.32	0.29	99.45	99.72	99.68	99.72

V. VERIFICATION CONCLUSION

A. VERIFY THE EFFECTIVENESS OF FLC FRAMEWORK

We compared the classification error curves of ResNet18_Original (ResNet18_C64_1) and ResNet18_FLC models (ResNet18_C64_2), as shown in Figure 10. It is encouraging that the ResNet18_FLC model outperforms the ResNet18_Original model. Specifically, in the CIFAR10, Sign-50, and Sign-85 datasets, ResNet18_FLC performs significantly better than ResNet18_Original. Although the classification error of both models gradually converges as epochs increase, ResNet18_FLC consistently exhibits lower errors. In the other three datasets (Print0-9, Vegetable, and Time), one can see that the performance of the

ResNet18_Original and ResNet18_FLC models is similar. However, the error converges more rapidly using the ResNet18_FLC model than the ResNet18_Original model. Overall, the ResNet18_FLC model (with FLC structure) can improve the classification accuracy of the original ResNet18 model while substantially reducing the parameter size from 42.63 MB to 37.93 MB.

Then, we validated the effectiveness of the FLC framework using the MNIST, MNIST_balance, and STL10 datasets, the CINIC10 classification dataset, and the Caltech101 classification dataset. The introduction of each dataset is shown in Table 6. It should be noted that we have removed 100000 unsupervised data from the

TABLE 5. The calculated results of N_error_avg and N_f1_avg for four groups of models, where $N = 1, 2, 3, 4$ and 5 . The scores of the five groups were determined according to Algorithm 2.

		$N_error_avg(\%)$					$N_f1_avg(\%)$				
		N=1	N=2	N=3	N=4	N=5	N=1	N=2	N=3	N=4	N=5
Print0-9	epoch=10	0.67	0.27	0.74	0.47	0.29	99.33	99.73	99.26	99.53	99.71
	epoch=20	0.15	0.03	0.2	0.06	0.05	99.85	99.97	99.8	99.94	99.95
	epoch=30	0.07	0	0.12	0.06	0.04	99.93	100	99.88	99.94	99.95
CIFAR10	epoch=20	20.43	20.11	20.46	21.41	20.33	79.57	79.84	79.52	78.57	79.65
	epoch=40	17.21	17.06	17.78	19.4	17.38	82.75	82.89	82.2	80.54	82.58
	epoch=60	17.19	16.94	17.03	19.09	17.44	82.76	83	82.92	80.8	82.5
Vegetable	epoch=20	1.33	1.33	1.11	1.58	1.38	98.67	98.67	98.89	98.42	98.62
	epoch=40	0.62	0.61	0.66	0.59	0.53	99.38	99.39	99.34	99.41	99.47
	epoch=60	0.55	0.46	0.59	0.42	0.5	99.45	99.54	99.41	99.58	99.5
Sign-50	epoch=20	32.98	31.36	29.63	31.02	26.89	64.64	66.71	68.35	66.6	71.29
	epoch=40	9.35	8.02	8.95	8.36	8.54	90.41	91.71	90.68	91.31	91.09
	epoch=60	6.78	5.85	6.83	5.44	5.49	93.04	93.99	93.02	94.42	94.42
Sign-85	epoch=20	32.17	31.59	34.22	34.96	33.63	66.01	66.65	63.94	62.06	64.03
	epoch=40	12.24	12.93	14.03	13.4	12.6	86.54	86.88	85.7	86.26	87.16
	epoch=60	11.84	11.13	12.25	11.48	11.41	87.93	88.72	87.57	88.29	88.37
Time	epoch=20	0.62	0.49	0.68	0.41	0.42	99.38	99.52	99.33	99.62	99.58
	epoch=40	0.68	0.32	0.4	0.22	0.29	99.32	99.68	99.6	99.78	99.71
	epoch=60	0.4	0.17	0.43	0.24	0.28	99.6	99.83	99.58	99.76	99.72
Final score		4	42	2	18	6	0	40	2	18	16

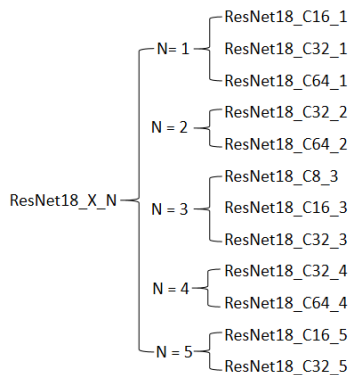


FIGURE 9. Five groups of the ResNet18 models: ResNet18_X_1, ResNet18_X_2, ResNet18_X_3, ResNet18_X_4, and ResNet18_X_5.

TABLE 6. The two public datasets used in this section.

Datasets	category	Total	Training	Testing	resolution
MNIST	10	70,000	60,000	10,000	28×28
STL	10	13,000	5,000	8,000	96×96
CINIC	10	180,000	90,000	90,000	32×32
MNIST_balance	47	131,600	112,800	18,800	28×28
Caltech101	101	8,677	6,074	2,603	Unfixed

STL10 dataset and only used the training and testing set of STL10.

In this section, we select the same experiment as the third section to train and test the model. We obtained the error, Confidence Interval and Confidence Interval width of AUC of ResNet18_Original and ResNet18_FLC models, and calculated the running time of each model in the experiment. As shown in Table 7, ResNet18_FLC has smaller error, CI_Width, and running time.

TABLE 7. Experimental results of training ResNet18_Original (Or) and ResNet18_FLC (FLC) models using two public datasets in Table 6.

		error	CI_AUC	CI_Width	Time(s)
MNIST	Or	0.39	(99.99,100)	0.01	3739
	FLC	0.34	(99.99,100)	0.01	3388
STL10	Or	39.95	(85.23,89.76)	4.53	623
	FLC	33.54	(88.68,92.30)	3.62	578
CINIC	Or	27.37	(94.45,95.86)	1.41	13108
	FLC	26.74	(94.74,95.73)	0.99	12908
MNIST_balance	Or	11.23	(99.22,99.62)	0.40	9339
	FLC	10.66	(99.62,99.67)	0.05	8698
Caltech101	Or	33.76	(87.27,93.23)	5.96	708
	FLC	29.09	(89.56,94.94)	5.38	675

B. COMPARATIVE STUDY OF VARIOUS RESNET MODELS WITH AND WITHOUT FLASK-LIKE CHANNEL STRUCTURES

We applied the flask-like channel (FLC) framework to other ResNet models with varying layers, such as ResNet34, ResNet50, ResNet101, and ResNeXt models. However, as the number of layers deepens, the number of parameters increases exponentially, significantly increasing the computational cost of the model. To reduce the computational resources of the model, we used a smaller batch size (show in table 8). Similarly, we calculated the average errors and F1 values of these models using the CIFAR10 dataset, presented in Table 8. A comparison (Figure 11) of the results obtained from the original ResNet models and the proposed FLC model revealed that the FLC models significantly reduce parameter sizes by 31.86% to 56.61%, reduce FLOPs approximately by 9.38% to 28.57% and improves classification accuracy by roughly 1%. This comparative analysis demonstrates the FLC framework’s transferability to other ResNet models.

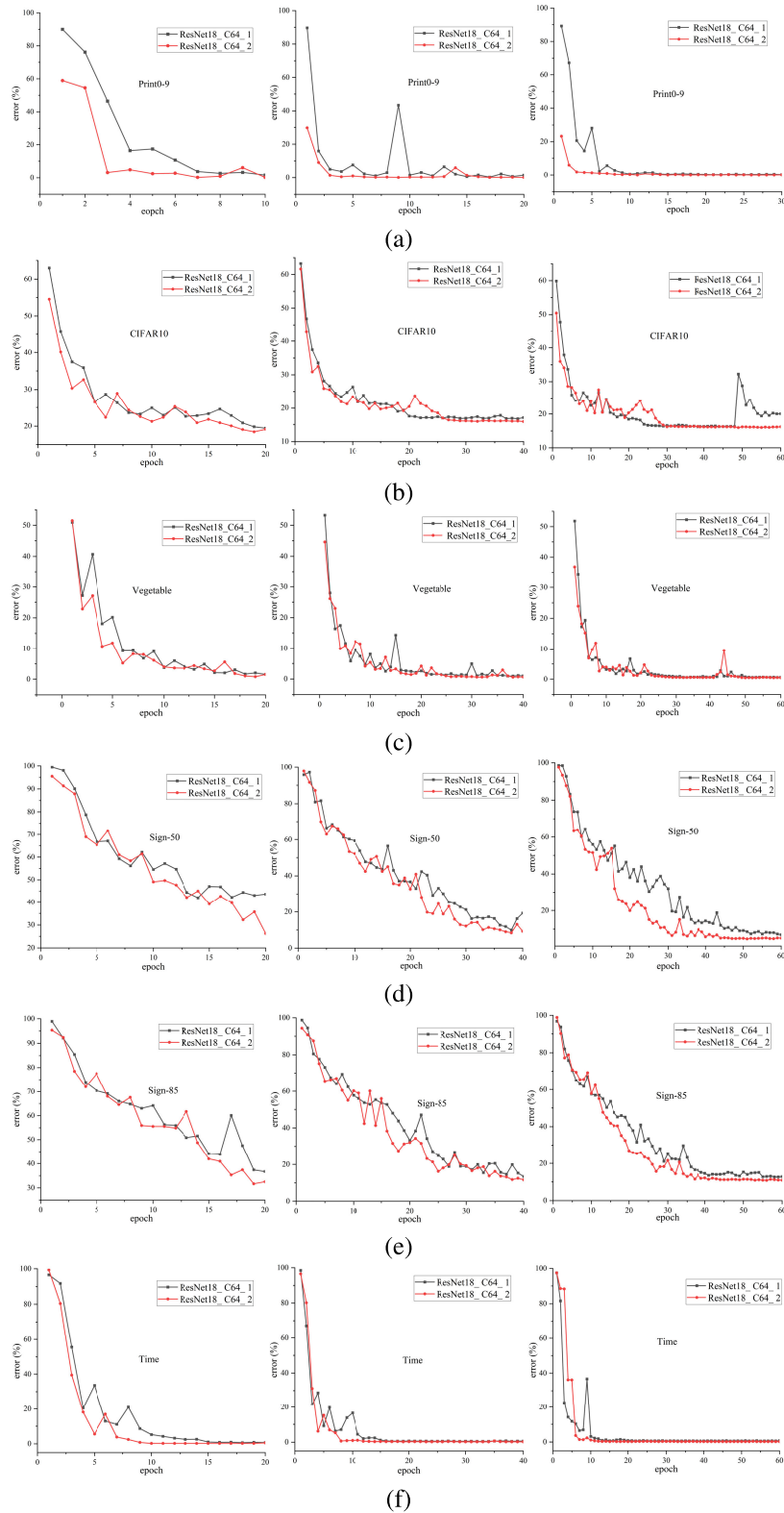


FIGURE 10. Comparison between the ResNet18_Original (ResNet18_C64_1) and ResNet18_FLC (ResNet18_C64_2) models in the calculations of classification accuracy (error) using different datasets (a) Print0-9, (b) CIFAR10, (c) Vegetable, (d) Sign-50, (e) Sign-85, (f) Time.

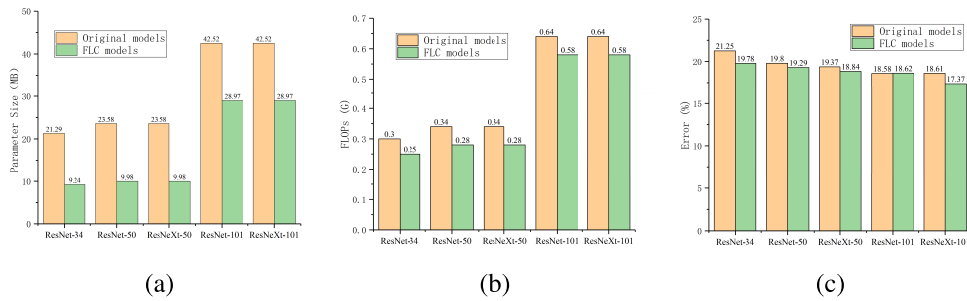


FIGURE 11. Comparison between the original ResNet models and our proposed ResNet_FLC models in (a) parameter size, (b) FLOPs and (c) calculated error, obtained from the experiments using the CIFAR10 dataset.

Algorithm 1 Training and testing with ResNet18_ X_ N models

```

Input: Epoch; model; optimizer; Loss function: criterion;
input data: dataloaders; hyper-parameters: opt
Output: epoch_acc; epoch_f1
1: epoch_acc = []
2: epoch_f1 = []
3: model.train()
4: for batchidx, data in enumerate(dataloaders['train'], 0):
5: inputs, target = data
6: inputs, target = inputs.to(opt.device), target.to(opt.device)
7: outputs = model(inputs)
8: loss = criterion (outputs, target)
9: optimizer.zero_grad()
10: loss.backward()
11: optimizer.step()
12: loss += loss.item() 13: pred = outputs.argmax(dim=1)
14: model.eval()
15: y_true = []
16: y_pred = []
17: with torch.no_grad():
18: for inputs, target in dataloaders['test']:
19: inputs, target = inputs.to(opt.device), target.to(opt.device)
20: outputs = model(inputs)
21: _, pred = torch.max(outputs, 1)
22: y_true.extend(target.cpu().numpy())
23: y_pred.extend(pred.cpu().numpy())
24: acc = accuracy_score(y_true, y_pred) * 100
25: f1 = f1_score (y_true, y_pred, average='weighted') * 100
26: epoch_acc.append(acc)
27: epoch_f1.append (f1)
28: return epoch_acc, epoch_f1
    
```

VI. CONCLUSION AND OUTLOOK

In this study, we aim to search for the optimal compromise between the ResNet complexity and performance by constructing different types of channel structures for ResNet models to systematically evaluate their impact on perfor-

Algorithm 2 Calculating the score in Table 5

```

Input: Weight s for each row in Table 11, s = [2, 4, 6, 2, 4, 6, 2, 4, 6, 2, 4, 6, 2, 4, 6, 2, 4, 6, 2, 4, 6]
Score 'score_i' for each category, initial value is 0, i = 1, 2, 3, 4, 5
error_avg:18 rows and 5 columns of data for N_error_avg in Table 5
Output: Score 'score_i' for each category, i = 1, 2, 3, 4, 5
1: for i in len(N_error_avg) do
2: h = error_avg[i]. index (min(error_avg[i]))
3: if h == 0 then
4: score_1 += s[i]
5: if h == 1 then
6: score_2 += s[i]
7: if h == 2 then
8: score_3 += s[i]
9: if h == 3 then
10: score_4 += s[i]
11: if h == 4 then
12: score_5 += s[i]
13: return score_1, score_2, score_3, score_4, score_5
    
```

TABLE 8. The calculated results obtained from ResNet_original (Or) models and our proposed ResNet_FLC (FLC) models in the experiments of CIFAR10 dataset.

Models		Batch size	#params.	FLOPs	Error (%)	F1(%)
ResNet-34	Or	128	81.21	0.30G	21.25	78.6
	FLC	128	35.24	0.25G	19.78	80.3
ResNet-50	Or	32	89.75	0.34G	19.8	80.2
	FLC	32	38.07	0.28G	19.29	80.62
ResNeXt-50	Or	32	89.75	0.34G	19.37	80.36
	FLC	32	38.07	0.28G	18.84	81.17
ResNet-101	Or	32	162.2	0.64G	18.58	81.57
	FLC	32	110.52	0.58G	18.62	81.3
ResNeXt-101	Or	32	162.2	0.64G	18.61	81.48
	FLC	32	110.52	0.58G	17.37	82.7

mance. We created 25 models with varying channel structures for ResNet18 and selected 18 models with smaller parameter sizes and smaller FLOPs than the original ResNet18 model. Using the Butterfly50 dataset, we tested these 18 models' performance, and the top comprehensive performance 12 models were chosen for subsequent studies. Secondly,

we applied twelve ResNet18 models to six publicly available image datasets, such as Print0-9, CIFAR10, Vegetable, Sign-50, Sign-85, and Time. After extensive experiments, the Flask-like Channel (FLC) structure demonstrated the best performance among the five channel structures. We validated the effectiveness of the model using MNIST and STL10 public datasets. Finally, we applied the FLC framework to various ResNet models with different layers, such as ResNet34, ResNet50, ResNet101, and ResNeXt. Then, we compared the performance of these models (with and without the FLC structure) in the CIFAR10 dataset. Our results reveal that the ResNet models using the FLC structure can maintain or improve the accuracy of the model by approximately 1% while reducing the number of model parameters and FLOPs.

The number of classifications in all datasets used in this study was below 144. Future experiments will be conducted to assess the robustness of the Flask-like Channel (FLC) framework by applying it to datasets with larger numbers of classifications (beyond 144), like ImageNet. In addition, we will attempt to apply the FLC structure to other fields [31], [32] of deep learning, in order to investigate the effectiveness of the FLC structure in other application scenarios in future work.

REFERENCES

- [1] R. Shang, J. Wang, L. Jiao, X. Yang, and Y. Li, "Spatial feature-based convolutional neural network for PolSAR image classification," *Appl. Soft Comput.*, vol. 123, Jul. 2022, Art. no. 108922.
- [2] G. Li, M. Zhang, J. Wang, D. Weng, and H. Corporaal, "SCWC: Structured channel weight sharing to compress convolutional neural networks," *Inf. Sci.*, vol. 587, pp. 82–96, Mar. 2022.
- [3] A. Sepas-Moghaddam, A. Etemad, F. Pereira, and P. L. Correia, "CapsField: Light field-based face and expression recognition in the wild using capsule routing," *IEEE Trans. Image Process.*, vol. 30, pp. 2627–2642, 2021.
- [4] J. Cheng, F. Zhang, D. Xiang, Q. Yin, and Y. Zhou, "PolSAR image classification with multiscale superpixel-based graph convolutional network," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5209314.
- [5] H. T. Phan, N. T. Nguyen, and D. Hwang, "Convolutional attention neural network over graph structures for improving the performance of aspect-level sentiment analysis," *Inf. Sci.*, vol. 589, pp. 416–439, Apr. 2022.
- [6] P. Bhuvaneshwari, A. N. Rao, Y. H. Robinson, and M. N. Thippeswamy, "Sentiment analysis for user reviews using bi-LSTM self-attention based CNN model," *Multimedia Tools Appl.*, vol. 81, no. 9, pp. 12405–12419, Apr. 2022.
- [7] C. Liu, H. Sun, J. Katto, X. Zeng, and Y. Fan, "QA-filter: A QP-adaptive convolutional neural network filter for video coding," *IEEE Trans. Image Process.*, vol. 31, pp. 3032–3045, 2022.
- [8] Z. Fang, B. Yin, Z. Du, and X. Huang, "Fast environmental sound classification based on resource adaptive convolutional neural network," *Sci. Rep.*, vol. 12, no. 1, p. 6599, Apr. 2022.
- [9] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Mar. 1998, doi: 10.1109/5.726791.
- [10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 84–90.
- [11] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 818–833.
- [12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
- [13] K. Simonyan and A. Zisserman, "Very deep convolution networks for large-scale image recognition," *Comput. Sci.*, vol. 18, no. 3, pp. 178–182, 2014.
- [14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
- [15] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*.
- [16] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.
- [17] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–19.
- [18] S. Feng and C. Chen, "Performance analysis of fuzzy BLS using different cluster methods for classification," *Sci. China Inf. Sci.*, vol. 64, pp. 1–12, Jan. 2021.
- [19] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, Sep. 2016, pp. 646–661.
- [20] J. Moniz and C. Pal, "Convolutional residual memory networks," 2016, *arXiv:1606.05262*.
- [21] S. Targ, D. Almeida, and K. Lyman, "ResNet in ResNet: Generalizing residual architectures," *Journal*, vol. 2, no. 5, pp. 99–110, 2016.
- [22] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5987–5995.
- [23] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr, "Res2Net: A new multi-scale backbone architecture," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 2, pp. 652–662, Feb. 2021.
- [24] F. Shen, R. Gan, and G. Zeng, "Weighted residuals for very deep networks," in *Proc. 3rd Int. Conf. Syst. Informat. (ICSIAI)*, Nov. 2016, pp. 936–941.
- [25] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, H. Lin, Z. Zhang, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, and A. Smola, "ResNeSt: Split-attention networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2022, pp. 2736–2746.
- [26] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.
- [27] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 4470–4478.
- [28] D. Han, J. Kim, and J. Kim, "Deep pyramidal residual networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6307–6315.
- [29] M. I. Ahmed, S. M. Mamun, and A. U. Z. Asif, "DCNN-based vegetable image classification using transfer learning: A comparative study," in *Proc. 5th Int. Conf. Comput., Commun. Signal Process. (ICCCSP)*, May 2021, pp. 235–243.
- [30] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
- [31] M. Muthuswamy and A. M. Ali, "Sustainable supply chain management in the age of machine intelligence: Addressing challenges, capitalizing on opportunities, and shaping the future landscape," *Sustain. Mach. Intell. J.*, vol. 3, pp. 33103.1–33103.14, Jun. 2023.
- [32] N. Nabeeh, "Assessment and contrast the sustainable growth of various road transport systems using intelligent neurosophic multi-criteria decision-making model," *Sustain. Mach. Intell. J.*, vol. 2, pp. 1–12, Mar. 2023, doi: 10.61185/SMIJ.2023.22102.



DONGYAO LI received the degree from the School of Medical Information Engineering, Heilongjiang University of Chinese Medicine, in 2020. She is currently pursuing the master's degree with the School of Information, Guizhou University of Finance and Economics. Her main research interests include deep learning, machine learning, and deep residual networks.



YUNHUI PAN was born in Shandong, China. He received the bachelor's degree in engineering from Jiangxi Normal University, in 2020. He is currently pursuing the master's degree with the School of Information Science and Engineering, Guizhou University of Finance and Economics. His research interests include natural language processing, sentiment analysis, and multimodal sentiment analysis.



MINGSEN DENG (Member, IEEE) is currently a Professor of computer science with the Guizhou University of Finance and Economics, China. He has published more than 100 papers in prestigious journals and international conferences. His research interests include parallel and distributed computing, electronic structure calculations, and network analysis for big data. He has been an Executive Member of the Technical Committee of High-Performance Computing of the China Computer Federation, since 2010. He received numerous awards, including the Outstanding Scientists and Technologists of the Chinese Institute of Electronics, in 2020, the One Hundred Person Project of the Guizhou Province, in 2016, and the Young Scientist Award of Guizhou Province, in 2018.



SHUHUA MAO received the Ph.D. degree in engineering from the Wuhan University of Technology, in 2011. From 2011 to 2015, he was engaged in postdoctoral research in shipbuilding and ocean engineering with the Wuhan University of Technology. He is currently a Professor of statistics and mathematics with the Wuhan University of Technology. His main research interests include intelligent optimization, deep learning, fractional grey system theory and its applications, statistical prediction, and decision modeling.



HUJUN SHEN received the Ph.D. degree in chemistry from Johns Hopkins University, in 2006. Since 2018, he has been the Deputy Director of the Guizhou Provincial Key Laboratory of Computational Nano-Material Science. He is currently a Professor with Guizhou Education University, China. His research interests include optimizing deep learning methods, material simulation based on active learning, and molecular dynamics simulation based on machine learning. He received numerous awards, including the Guizhou Provincial Natural Science Award, in 2020, the Guizhou Province Graduate Teaching Achievement Award, in 2019 and 2021, and the Outstanding Instructor of the Blue Bridge Cup Competition, in 2022.

...