Image Classication

Jing Liu
College of Information Science and Engineering
Ocean University of China
Email: liujing9395@stu.ouc.edu.cn

***Abstract*—With the extensive application of Convolution Neural Networks (CNNs), CNNs have been applied to image classication issues and achieve remarkable performance. Image classication is one of the most important tasks in the computer vision eld. In this paper, which focuses on the application of CNNs to image classication tasks, we design four different network structures from easy to complex in order to help you better understand the process of image classication using CNNs. Along the way, we analyze (1) the function of networks each layer, (2) the comparison among different network designs, (3) several improvement attempts by learning from the innovation of state-of-the-art CNNs.**

I. Introduction

Image classication, which can be dened as the task of categorizing images into one of several predened classes, is a fundamental problem in computer vision eld. It forms the basis for other computer vision tasks such as location, detection, and segmentation. Although the task can be considered as second nature of humans, it is much more challenging for a computer system. In traditional machine learning period, we need to do a dual-stage operation to solve the image classication problem. First, we need to extract handcrafted features from images using feature descriptors. Then put these features into a trainable classier. The major weakness of this approach is that the accuracy of the classication task is profoundly dependent on the design of the feature extraction stage, and this usually proved to be a challenging task [1].

In recent years, deep learning models that exploit multiple layers of nonlinear information processing, for feature extration and transformation as well as for pattern analysis and classication, have been shown to overcome these challenges. Among them, CNNs [2], [3] have become the leading architecture for most image recognition, classication, and detection tasks [4]. Despite some early successes, deep CNNs (DCNNs) were brought into the limelight as a result of the deep learning renaissance [5]–[7], which was fueled by GPUs, larger data sets, and better algorithms [8]–[10]. Several advances such as the rst GPU implementation [11] and the rst application of maximum pooling (max pooling) for DCNNs [12] have all contributed to their recent popularity.

The most signicant advance, which has captured intense interest in DCNNs, especially for image classication tasks, was achieved in the ImageNet Large Scale Visual Recognition Clallenge (ILSVRC) 2012 [13], when the winning entry, used a DCNN to classify approximately 1.2 million images into 1000 classes, with record-breaking results. Since then, DCNNs have dominated subsequent versions of the ILSVRC and, more specically, its image classication component.

The remainder of this paper is organized as follows: Section 2 briey introduces CNNs and acquaints readers with the key building blocks of their architecture. Section 3 introduces the four different CNNs in our experiment. Section 5 shows some experimental details and results.

II. Overview of CNN Architecture

CNN architectures come in several variations; however, in general, they consist of convolutional and pooling (or subsampling) layers, which are grouped into modules. Either one or more fully connected layers, as in a standard feedforward neural network, follow these modules. Modules are often stacked on top of each other to form a deep model. Fig. 1 illustrates typical CNN architecture. An image is input directly to the network, and this is followed by several stages of convolution and pooling. Thereafter, representations from these operations feed one or more fully connected layers. Finally, the last fully connected layer outputs the class label.

A. Convolutional Layers

The convolutional layers serve as feature extrators, and thus they learn the feature representations of their input images. The neurons in the convolutional layers are arranged into feature maps. Each neuron in a feature map has a receptive eld, which is connected to a neighborhood of neurons in the previous layer via a set of trainable weights, sometimes referred to as a lter bank. Inputs are convolved with the learned weights in order to compute a new feature map, and the convolved results are sent through a nonlinear activation function. All neurons within a feature map have weights that are constrained to be equal; however, different feature maps within the same convolutional layer have different weights so that several features can be extracted at each location. More formally, the kth output feature map k can be computed as

$$\gamma_k = f(W_k * x), \tag{1}$$

where the input image is denoted by x; the convolutional filter related to the kth feature map is denoted by W_k; the multiplication sign in this context refers to the 2D convolutional operator, which is used to calculate the inner product of the lter model at each location of the input image; and f$(\cdot)$ represents the nonlinear activation function [14]. Nonlinear activation functions allow for the extraction of nonlinear features. Traditionally, the sigmoid and hyperbolic tangent

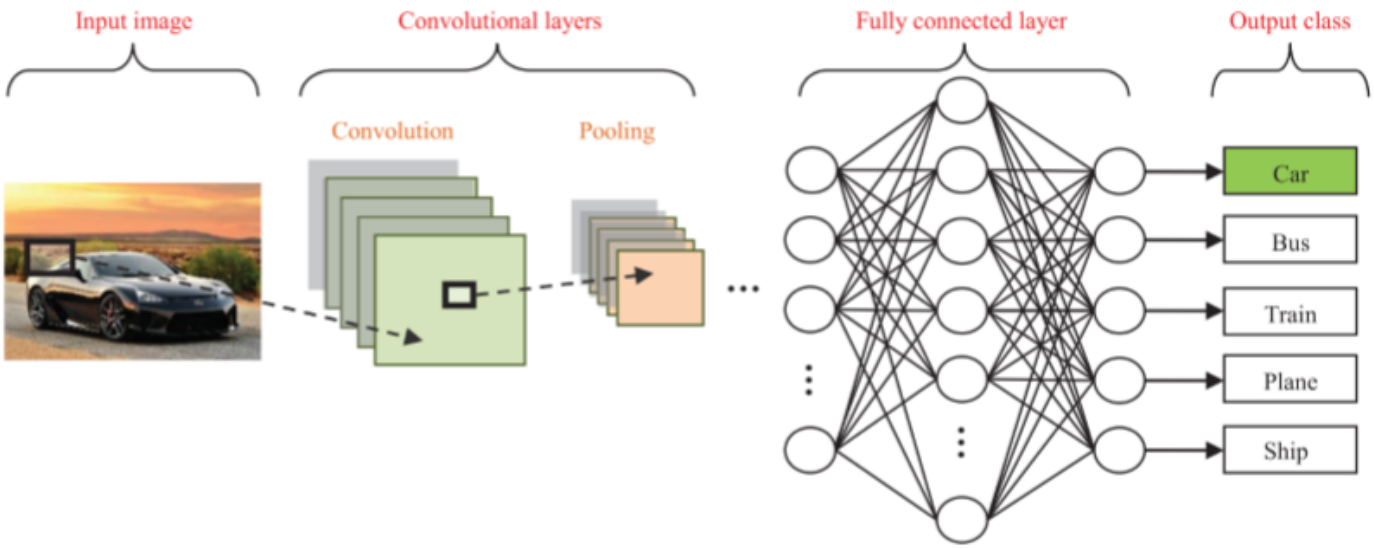


Fig. 1. Basic version of CNN architectures.

functions were used; recently, rectied linear units [15] have become popular.

B. Pooling Layers

The purpose of the pooling layers is to reduce the spatial resolution of the feature maps and thus achieve spatial invariance to input distortions and translations [2], [3]. Initially, it was common practice to use average pooling to prapagate the average of all the input values. However, in more recent models [8], [16]–[20], max pooling aggregation layers propagate the maximum value within a receptive eld to the next layer [12]. Formally, max pooling selects the largest element within each receptive eld such that

$$\gamma_{kij} = \max_{(p,q)\in\Upsilon_{ij}} x_{kpq}, \tag{2}$$

where the output of the pooling operation, associated with the kth feature map, is denoted by γ_{kij}, x_{kpq} denotes the element at location (p, q) contained by the pooling region Υ_{ij}, which embodies a receptive field around the position (i, j) [21]. Fig. 2 illustrates the difference between max pooling and average pooling.

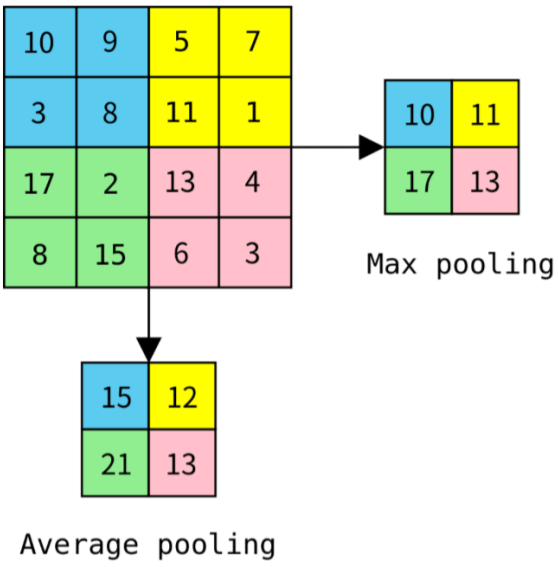


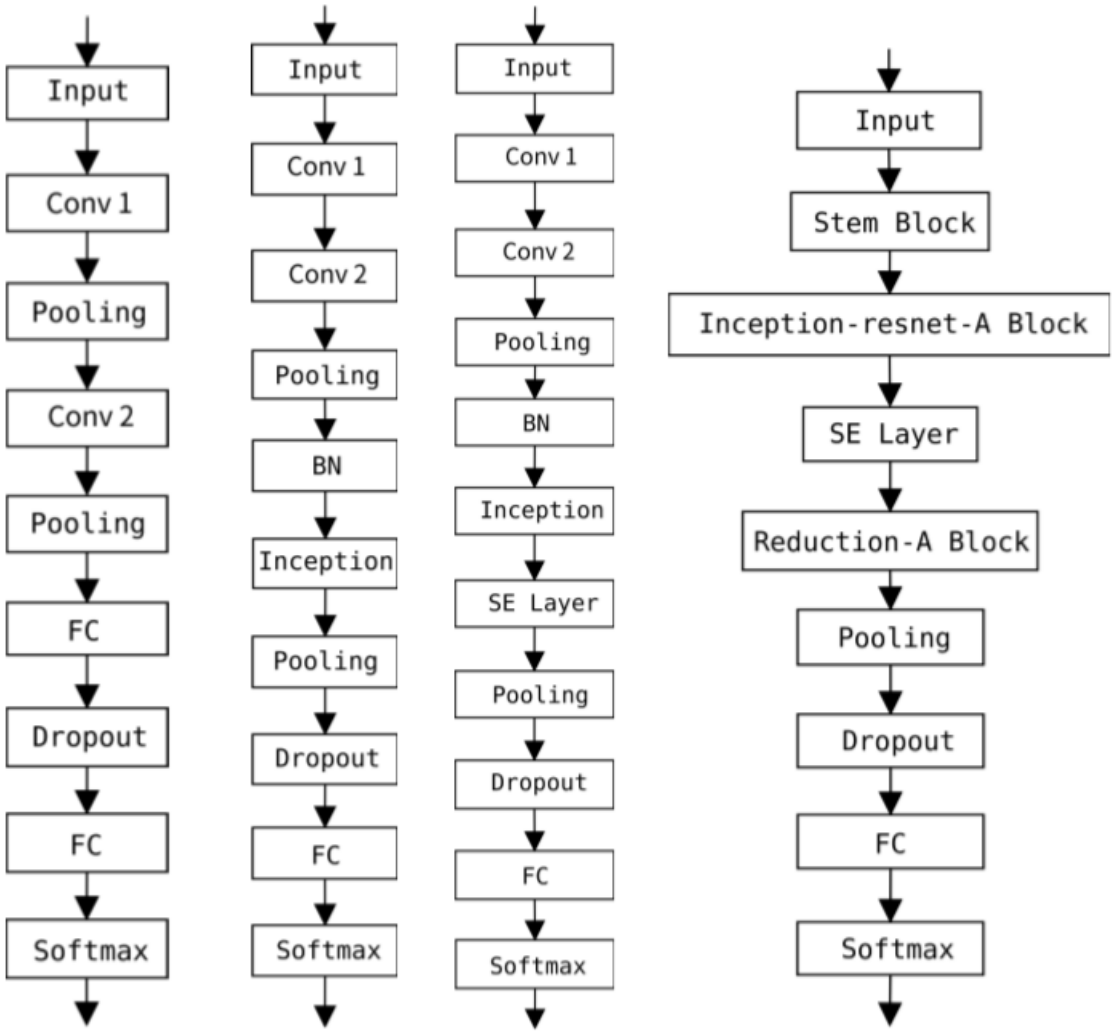
Fig. 2. Comparison between max pooling and average pooling.

C. Fully Connected Layers

Several convolutional and pooling layers are usually stacked on top of each other to extract more abstract feature representations in moving through the network. The fully connected layers that follow these layers interpret these feature representations and perform the function of high-level resoning [17], [18], [22]. For classication tasks, it is standard to use the softmax operator on top of a DCNN.

III. NETWORKS

In this section, we will introduce four different CNN architectures as shown in Fig. 3. In order to have a better understanding about the CNN implementation process, we arrange the four networks from simple to complex by adding some innovative blocks gradually. These innovative blocks are learned from some advanced CNNs such as Inception Networks [19], [23]–[25], ResNet [26] and SENet [27]. Also we conduct comparative experiments with different activation functions and different pooling operations. More specically, we will introduce every CNN separately in the following sections.



(a) a (b) b (c) c (d) d

Fig. 3. Illustration of four different CNNs in our experiments.

A.

Fig. 3a shows the network architecture used in our rst experiment. This is a relatively simple convolutional neural network structure with only two convolution layers (Conv) and two fully connected layers (FC). The main purpose of this experiment is to learn more about the basic structure and computation process of CNN. Meanwhile, we compared the different activation functions and pooling operations in this experiment, results are shown in Table II. For implementing the normalization operation, we choose batch normalization (BN) and dropout operations. BN and dropout all have effect on normalization and [28] conduct more comparison between them with deep theoretical analysis.

B.

As shown in Fig. 3b, we construct the CNN architecture with the inception block [19] and adopt the BN layer for normalization operation. The inception block was rst proposed in [19] as shown in Fig. 4. The main idea of the Inception architecture is to consider how an optimal local sparse structure of a convolutional vision network can be approximated and covered by readily available dense components.

C.

In the third experiment, we add a novel module called SE block to compose the network as shown in Fig. 3c. The detail composition is shown as illustrated in Fig. 5. SE block [27] is a novel architecture unit that can adaptively recalibrates channel-wise feature responses by explicitly modelling interdependencies between channels. From the existing experiments using the SE block, we nd that SE block produce signicant performance improvements at minimal additional computational cost.

D.

In the nal experiment, from the network structure as shown in Fig. 3d, we design a simplied Inception-ResNet [25] con-

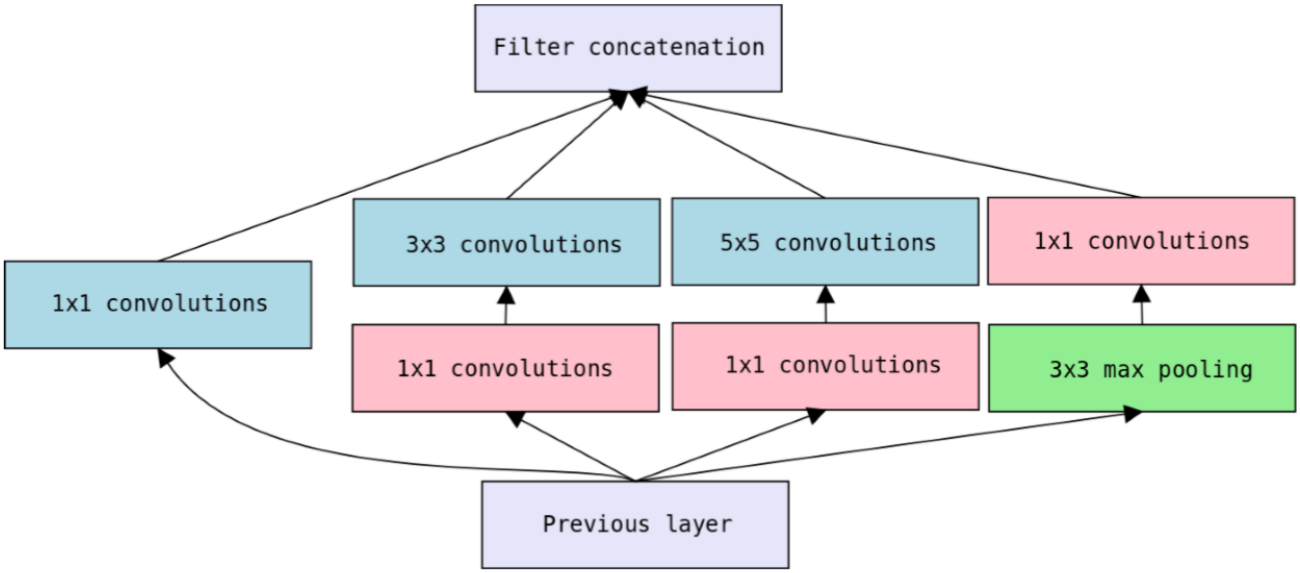


Fig. 4. Inception module.

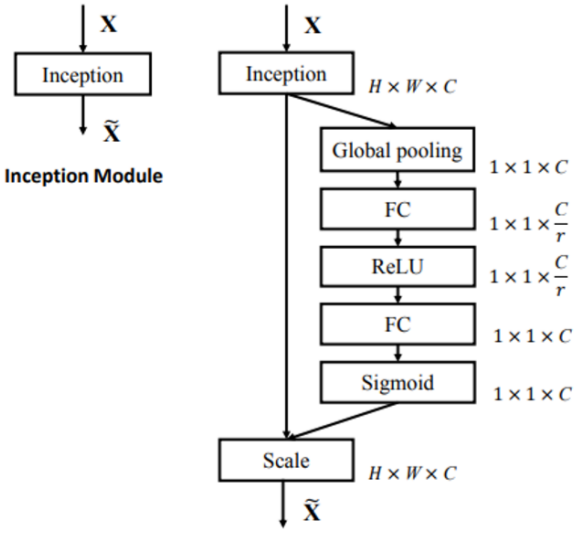


Fig. 5. SE-Inception module.

structed with SE block. By implementing this experiment, we can have a better understanding about the Inception module, ResNet module and the SE block. The Stem block, Inceptionresnet-A block and Reduction-A block structures are as shown in Fig. 6.

IV. EXPERIMENT

In this section, we will show the experimental results of the four experiments.

TABLE I
EXPERIMENTAL RESULTS OF FIG. 3A NETWORK.

Activation function	Accuracy
Sigmod	65.94%
Tanh	66.40%
ReLu	65.18%

TABLE II
EXPERIMENTAL RESULTS OF FIG. 3B, FIG. 3C, FIG. 3D NETWORKS.

Network	Accuracy
Fig. 3b	70.13%
Fig. 3c	75.32%
Fig. 3d	78.98%

REFERENCES

[1] Y. LeCun *et al.*, "Generalization and network design strategies," *Connectionism in perspective*, pp. 143–155, 1989.

[2] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*, 1990, pp. 396–404.

[3] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[5] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[6] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[7] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007, pp. 153–160.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[9] L. Deng, D. Yu *et al.*, "Deep learning: methods and applications," *Foundations and Trends® in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[10] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[11] K. Chellapilla, S. Puri, and P. Simard, "High performance convolutional neural networks for document processing," in *Tenth International Workshop on Frontiers in Handwriting Recognition*. Suvisoft, 2006.

[12] F. J. Huang, Y.-L. Boureau, Y. LeCun *et al.*, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007, pp. 1–8.

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[14] W. Yu, K. Yang, Y. Bai, H. Yao, and Y. Rui, "Visualizing and comparing convolutional neural networks," *arXiv preprint arXiv:1412.6631*, 2014.

[15] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[16] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer vision and pattern recognition (CVPR), 2012 IEEE conference on*. IEEE, 2012, pp. 3642–3649.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[18] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich *et al.*, "Going deeper with convolutions." Cvpr, 2015.

[20] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[21] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed pooling for convolutional neural networks," in *International Conference on Rough Sets and Knowledge Technology*. Springer, 2014, pp. 364–375.

[22] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

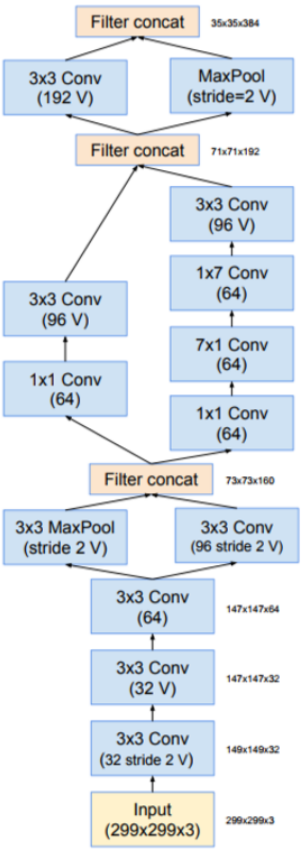
[24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[25] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." in *AAAI*, vol. 4, 2017, p. 12.

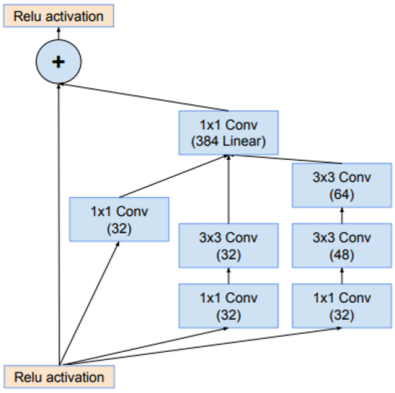
[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[27] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *arXiv preprint arXiv:1709.01507*, 2017.

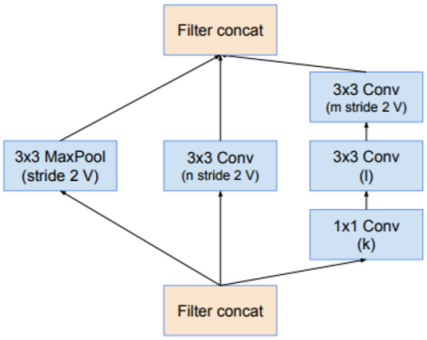
[28] X. Li, S. Chen, X. Hu, and J. Yang, "Understanding the disharmony between dropout and batch normalization by variance shift," *arXiv preprint arXiv:1801.05134*, 2018.



(a) scha



(b) schb



(c) schc

Fig. 6. The schemas of (a) Stem block, (b) Inception-resnet-A block, (c) Reduction-A block.