Large scale visual search

Wu, S.

Chapter 5

Comparison of Information Loss Architectures in CNNs

Recent advances in image classification have been achieved by the application of deep Convolutional Neural Networks (CNNs). Pooling and sub-sampling operations in the CNNs lead to invariance to local transformations, but result in loss of accuracy. In this chapter, we propose a novel deep neural network called the "Weighted Integration Architecture Network" (WIAN) that can effectively recover the information loss due to the pooling operations in the CNNs. The proposed WIAN reuses the information from the previous layers in the network and assigns a weight matrix to each layer; and then integrates them to further enhance the image classification performance. Two weight value generation schemes are investigated: the first one is calculated according to the responses or entropy in the layer, and the second one is an adaptive learning scheme. Exhaustive experiments on four standard benchmark datasets (CIFAR-10, CIFAR-100, MNIST and SVHN) demonstrate the effectiveness and improved performance of the proposed WIAN.

5.1 Introduction

Prior to convolutional neural networks, the commonly and widely used approaches in image classification were using the Bags-of-Words (BoW) model [159]. This type of model first encodes the local features from the salient regions in the image as a histogram of quantized visual words, and then feeds the histogram into a SVM classifier [160]. This method is a type of orderless statistics that incorporates spatial geometry into the BoW representation. Lazebnik et al. [77] integrated a spatial pyramid framework into the BoW feature generation, that counts the number of visual words inside a set of image sub-regions instead of the whole image region. This procedure was further improved by using sparse coding optimization for the construction of a visual vocabulary [161], obtaining the best performance on the ImageNet 1000-class classification problem. The approaches based on the visual word model can be viewed as zero order statistics (i.e., counts of visual words), and discard a lot of valuable information of the image. The Fisher Vector image representation introduced by Perronnin et al. [162] overcame this issue and extracted first and second order statistics by employing the Fisher Kernel [163], achieving state-of-the-art image classification results.

Recently, a significant performance gain on the task of image classification has been made with deep convolutional neural networks (CNNs) [164, 165]. This is mainly due to their ability to learn rich high level image representations as opposed to hand-designed low-level features, as well as the availability of very large and more comprehensive training data.

Traditional convolutional neural networks used for image classification consist of several stacked convolutional layers (optionally followed by a normalization layer and a pooling layer), fully connected layers and a softmax layer (a classifier) on the top. Convolutional layers take the inner product of a linear filter and the underlying receptive field followed by a nonlinear activation function at every local region of the input. The outputs from each convolutional layer are called feature maps. The fully connected layer has connections to all individual activations in the feature maps from the previous layer and the resulting vector can be fed into the softmax layer for classification (as shown in Figure 5.1). Variants

of this basic design are proposed to improve the performance of the network. Most recent methods increase the depth of the CNN architecture as well as the width of each layer to enhance the performance [47, 166]. However, increasing the depth of CNNs brings the issue of vanishing gradients and over-fitting during the optimization of the network, especially if the number of labeled examples in the training set is limited. Several useful technologies are employed to address the issue of over-fitting: data augmentation which increases the number of training samples when using a small dataset, pre-training which initializes the networks with pre-trained parameters rather than randomly set parameters, and dropout which randomly omits half of the feature detectors, aims to prevent complex co-adaptations on the training data and enhance the generalization ability. The architecture of GoogleNet [42] is designed such that the depth and width of the network is increased while the computational budget is keep constant. The Network-in-Network (NIN) is an approach proposed by Lin et al. [47] that replaces the linear convolution by a nonlinear convolution function to enhance the abstraction ability of the neural network. Deeply supervised networks [167] focus on the importance of minimizing the output classification error while reducing the prediction error of each individual layer. A Siamese network [168] is trained with a pairwise loss function that minimizes the distance between the same class and maximizes the distance between different classes. A similar triplet network [169] employs the triplet ranking loss function to preserve relative similarity relations.

In this chapter, we propose a novel architecture called Weighted Integration Architecture Network (WIAN) to boost the performance of image classification. WIAN starts by reshaping the convolutional layers to the same shape by a convolution operation, and normalizes each reshaped convolutional layers to the same scale. WIAN automatically learns a weight value matrix using an adaptive learning scheme or using the responses or entropy on each reshaped and normalized convolutional layer. Then these convolutional layers are multiplied by the assigned weight matrixes respectively, and finally combined into a single layer by element-wise summing, as illustrated in Figure 5.3. The main contributions of WIAN are as follows:

First, the weight matrix learning scheme for each layer is adaptive.

Second, the integration layer can effectively recover the spatial information loss caused by the pooling operation and improve the accuracy of image classification.

The remainder of this chapter is organized as follows: we make a review of related work on the recovery of spatial information loss in Section 5.2. Section 5.3 gives an overview of convolutional neural networks for image classification. Section 5.4 provides a detailed description of the proposed Weighted Integration Architecture Network (WIAN). Section 5.5 presents the experimental results, and conclusions are given in Section 5.6.

5.2 Related Work

In CNNs, a convolutional layer is usually followed by a pooling operation. The pooling operation reduces the spatial resolution by computing a summary statistic over a local spatial region (typically a max or average operation). The main motivation behind the use of pooling is to promote invariance to local input transformations (such as translation, occlusion and truncation of the local stimulus). This is mainly due to the fact that the resulting outputs after pooling show invariance to the spatial location within the pooling region. Hence, the pooling layer is particularly important for the performance of image classification where local image transformations may obfuscate the object identity. Additionally, the pooling layer plays a vital role in preventing over-training while reducing computational complexity for the task of image classification. However, these invariance achieved by pooling come at the price of loss of accurate spatial information. Several research efforts attempt to make up for the loss caused by the pooling operation. A commonly used method is cascaded convolutional neural network. Sun et al. [170] proposed to use cascaded convolutional networks to improve the accuracy of facial landmarks detection and Toshev et al. [71] applied the cascaded convolutional network to the human pose estimation. Tompson et al. [75] designed a heat-map regression model to refine the locations of human body joints. Yang

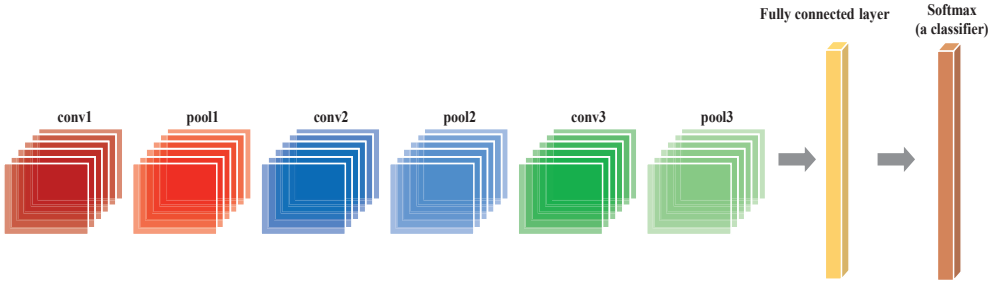


Figure 5.1: The architecture of a standard deep Convolutional Neural Network (CNN) used for image classification.

et al. [171] designed a DAG-CNN which extracts multi-scale features across each layer in the CNN and further integrates them for image classification. Inspired by the architecture of DAG-CNNs, the proposed WIAN automatically learns a weight matrix for each of previous layers and then integrates them for image classification. Thereby, the WIAN could improve the performance of the CNN.

5.3 Convolutional Neural Networks Classification

Considering a standard CNN architecture, as depicted in Figure 5.1, there are N convolutional layers, denoted as C^1,..., C^N. Each convolutional layer is followed by a pooling layer denoted as P^1,..., P^N, respectively. The objective of training a traditional CNN is to maximize the probability of the correct class, which is achieved by minimizing the softmax loss function. For a specific training set which includes m images: $\{(I^{(i)}, L^{(i)}); i = 1, ..., m\}$, where $I^{(i)}$ is the i^{th} image and $L^{(i)} \in \{1, ..., K\}$ is the class label. Let $\{x_j^{(i)}; j = 1, ..., K\}$ be the output of the activation j in the last fully connected layer, then the probability that the label of $I^{(i)}$ is j is given by

$$p_j^{(i)} = \frac{exp(x_j^{(i)})}{\sum_{j=1}^{K} exp(x_j^{(i)})} \tag{5.1}$$

The output of the fully connected layer is then fed into the softmax layer which aims to minimize the following loss function:

$$J_\theta = -\frac{1}{n}[\sum_{i=1}^{n}\sum_{j=1}^{K} 1\{L^{(i)} = j\} \log(p_j^{(i)})] \tag{5.2}$$

where $1\{.\}$ is the indicator function. Standard back-propagation is utilized to optimize the parameters of the network by computing the derivatives of the defined loss function.

Additionally, the success of AlexNet [48] suggests that the features emerging at the fully connected layers of a CNN trained for image classification can serve as good descriptors, when for example, using a SVM classifier for image classification.

5.4 Integration Architecture Network

As the architecture of standard CNNs did not take into account the information loss caused by the pooling operation, in this section, we explore several useful practices to integrate the information from the previous convolutional layers to recover the accuracy loss in CNNs. Performance evaluation results demonstrate that the integration of information from the previous convolutional layers could effectively increase the performance of image classification.

5.4.1 Concatenate Architecture Network

Inspired by the architecture of GoogleNet, a simple and effective way to train a high quality CNN is to concatenate the previous convolutional layer into a new layer. The illustration of the concatenate architecture network (CONCAT) is shown in Figure 5.2. In this architecture, we first reshape the convolutional layers in the CNN into the same shape by applying a convolution operation. These reshaped layers are normalized into the same scale and concatenated together. The fully connected layer takes all outputs of neurons in the concatenated layers

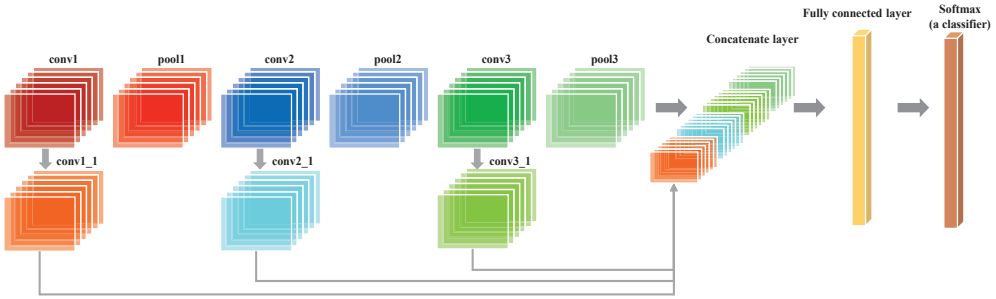


Figure 5.2: The illustration of the concatenate network (CONCAT). Each convolutional layer is reshaped and normalized, and concatenated in one layer for further processing.

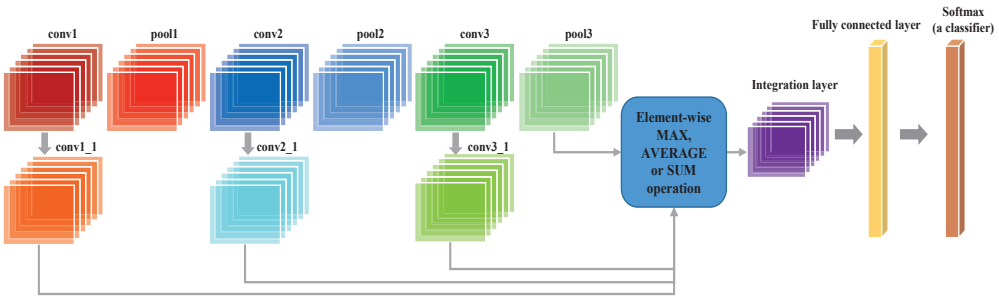


Figure 5.3: The architecture of the proposed Integration Architecture Networks. The layers are integrated by element-wise max, average or sum operations, respectively. The resulting network is called Max Integration Architecture Network (MIAN), Average Integration Architecture Network (AIAN) and Sum Integration Architecture Network (SIAN), respectively.

as input to every single neuron it has. Finally the output from the fully connected layer is fed into the softmax loss function optimizing classification.

5.4.2 Weighted Integration Architecture Network

The concatenate operation significantly increases the width of the integration layer, which means a larger number of parameters are stored in this layer. However, a large amount of parameters results in high storage requirements, and also it makes the network susceptible to over-fitting, especially if the amount of

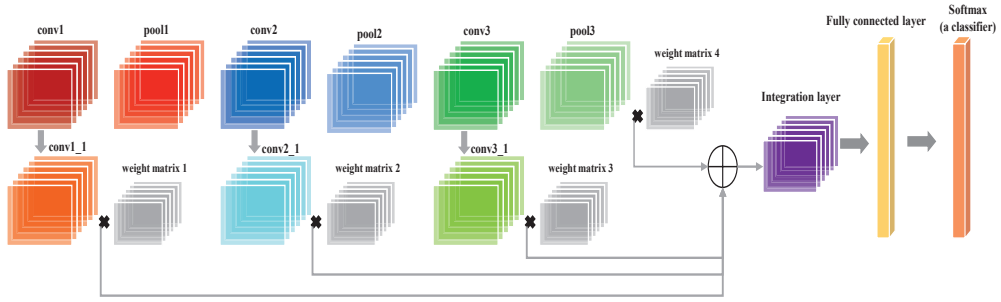


Figure 5.4: The architecture of the proposed Weighted Integration Architecture Network (WIAN). A weight matrix is assigned to each integration layer.

labeled data in the training set is small. Additionally, because of the existing redundant information between two adjacent layers, we propose to integrate the previous convolutional layers (reshaped to the same dimensions by a convolution operation and subsequently normalized) by applying element-wise max, average or sum operations, as depicted in Figure 5.3. The resulting network architectures are named the Max Integration Architecture Network (MIAN), the Average Integration Architecture Network (AIAN) and the Sum Integration Architecture Network (SIAN), respectively. Furthermore, we propose an adaptive method to integrate the previous convolutional layers, which assigns to each previous convolutional layer a weight matrix, respectively, and then combine them by element-wise summing (as shown in Figure 5.4). Two weight schemes are explored in this section, one relies on the responses or entropy of the convolutional layer and the other one is based on an adaptive weight learning method.

Responses based weight scheme: as shown in Figure 5.4, for the given N convolutional layers in the network, we denote the feature maps from layer C^n as $F^n, n = 1, ..., N$. These feature maps can be represented as a vector with dimension $w^n \times h^n \times c^n$, where w^n and h^n are the width and height of each individual feature map, and c^n denotes the number of feature maps of layer C^n. We further associate each unit of a feature map with a spatial coordinate (x, y) and the activation of this unit by $a(x, y)$. The response value of each feature map is calculated as $r^c = \sum_{x=1}^{w^n} \sum_{y=1}^{h^n} a(x, y), c = 1, ..., c^n$. The response value of each layer is computed as $R^n = \sum_{c=1}^{c^n} r^c$. The weight value in the weight matrix of

each layer is defined as:

$$weight_R^n = \frac{R^n}{\sum_{n=1}^{N} R^n} \tag{5.3}$$

Entropy based weight scheme: we further employ entropy information [172] on each convolutional layer to define a weight value. The activation of each unit $a(x, y)$ in the feature map can be treated as a state p_i, and the entropy of each feature map is computed by $e^c = \sum_{x=1}^{w^n} \sum_{y=1}^{h^n} (p_i \times \log p_i), c = 1, ..., c^n$. The entropy of each layer is computed as $E^n = \sum_{c=1}^{c^k} e^c$. The weight value in the weight matrix of each layer is then defined as:

$$weight_E^n = \frac{E^n}{\sum_{n=1}^{N} E^n} \tag{5.4}$$

For the responses or entropy based weight scheme, each unit in the weight matrix is assigned the same value of response or entropy calculated from each layer, thus, the weight matrix of responses or entropy based weight scheme can be reduced to one single weight value.

Finally, the activation value of each unit $a^{n+1}(x, y)$ in the integration layer is calculated using the following formula:

$$a^{n+1}(x, y) = \sum_{n=1}^{N} weight^n \times a^n(x, y) \tag{5.5}$$

Note that, in the specific case that the weight value from each layer is equal to $1/N$, the scheme becomes equal to the scheme of AIAN. If the weight from each layer is equal to 1, the scheme becomes equal to the scheme of SIAN.

Adaptive weight learning scheme: we further investigate an adaptive weight learning scheme in this chapter. Different from the responses or entropy based scheme where each unit in the weight matrix share the same weight value, the adaptive weight learning scheme assigns to each of the integrated layer a weight matrix. The initial values in each weight matrix are set to $1/k$, where k is the number of integrated layers. Then each unit in the weight matrix is automatically updated during each iteration of the CNN training. Let $weight_{(x,y)}^n$ be the value

of each unit in the weight matrix of the n^{th} layer, then the integrated value of each unit $a^{n+1}(x, y)$ in the integration layer is calculated as:

$$a^{n+1}(x,y) = \sum_{n=1}^{N} weight^{n}_{(x,y)} \times a^{n}(x,y) \tag{5.6}$$

5.5 Experimental Results

The proposed WIAN is implemented using Caffe [153]. The experimental environment is consisted of a computer with an i7 processor, 32GB RAM, and an NVIDIA TITANX. The network is trained using mini-batches of size 100 without data augmentation. The training process starts from the initial weights and learning rates, and it continues until the accuracy on the training set stops improving. Then the learning rates are lowered by a factor of 10 according to an epoch schedule determined on the validation set. The source code of WIAN is available at: http://press.liacs.nl/researchdownloads/.

5.5.1 Datasets

We evaluate the performance of WIAN on four benchmark datasets: CIFAR-10 [164], CIFAR-100 [164], MNIST [8] and SVHN [173].

CIFAR-10: the CIFAR-10 dataset is constructed for object recognition. It is composed of 10 object classes, with 6000 images per class. 50000 images are selected for training, and the remaining 10000 images are used for testing. Each image is given in the RGB-format with size 32×32 pixels.

CIFAR-100: the CIFAR-100 dataset is similar to the CIFAR-10 dataset (both use the same image size and format), except that the CIFAR-100 contains 100 classes with 600 images per class. CIFAR-100 also uses 50000 images for training and the remaining 10000 images for testing.

MNIST: the MNIST dataset consists of images of hand written digits which are 28×28 pixels in size. There are 60000 training images and 10000 testing images

in total. For this experiment, all images of the dataset have been resized to a fixed resolution of 32×32 pixels.

SVHN: the Street View House Numbers (SVHN) dataset is a collection of house numbers in the Google Street View images. It is composed of over 600000 color images with a fixed resolution of 32×32 pixels.

5.5.2 Details of Weighted Integration Architecture

The architecture of the network in the evaluation contains three convolutional layers, followed by Rectified Linear Unit (RELU) normalization and pooling operations, as well as a fully connected layer and a softmax classifier on top. Moreover, in order to integrate the previous convolution layers into one layer, we first convolute them to the same shape, normalize them into the same scale and then combine them.

According to the parameter configuration of each layer, the architecture of the WIAN in the performance evaluation can be described concisely by layer notations with the following layer sizes (CONV denotes the convolutional layer, RELU denotes the rectified linear unit layer, POOL denotes the pooling layer, and FC denotes the fully connected layer):

$INPUT(32 \times 32 \times 3)$

$CONV1(32 \times 32 \times 32) \rightarrow RELU1 \rightarrow POOL1(16 \times 16 \times 32)$

$CONV2(16 \times 16 \times 32) \rightarrow RELU2 \rightarrow POOL2(8 \times 8 \times 32)$

$CONV3(8 \times 8 \times 64) \rightarrow RELU3 \rightarrow POOL3(4 \times 4 \times 64)$

$CONV1 \rightarrow CONV1\_1(4 \times 4 \times 64)$

$CONV2 \rightarrow CONV2\_1(4 \times 4 \times 64)$

$CONV3 \rightarrow CONV3\_1(4 \times 4 \times 64)$

$CONV1\_1 + CONV2\_1 + CONV3\_1 + POOL3 \rightarrow FC \rightarrow Softmax$

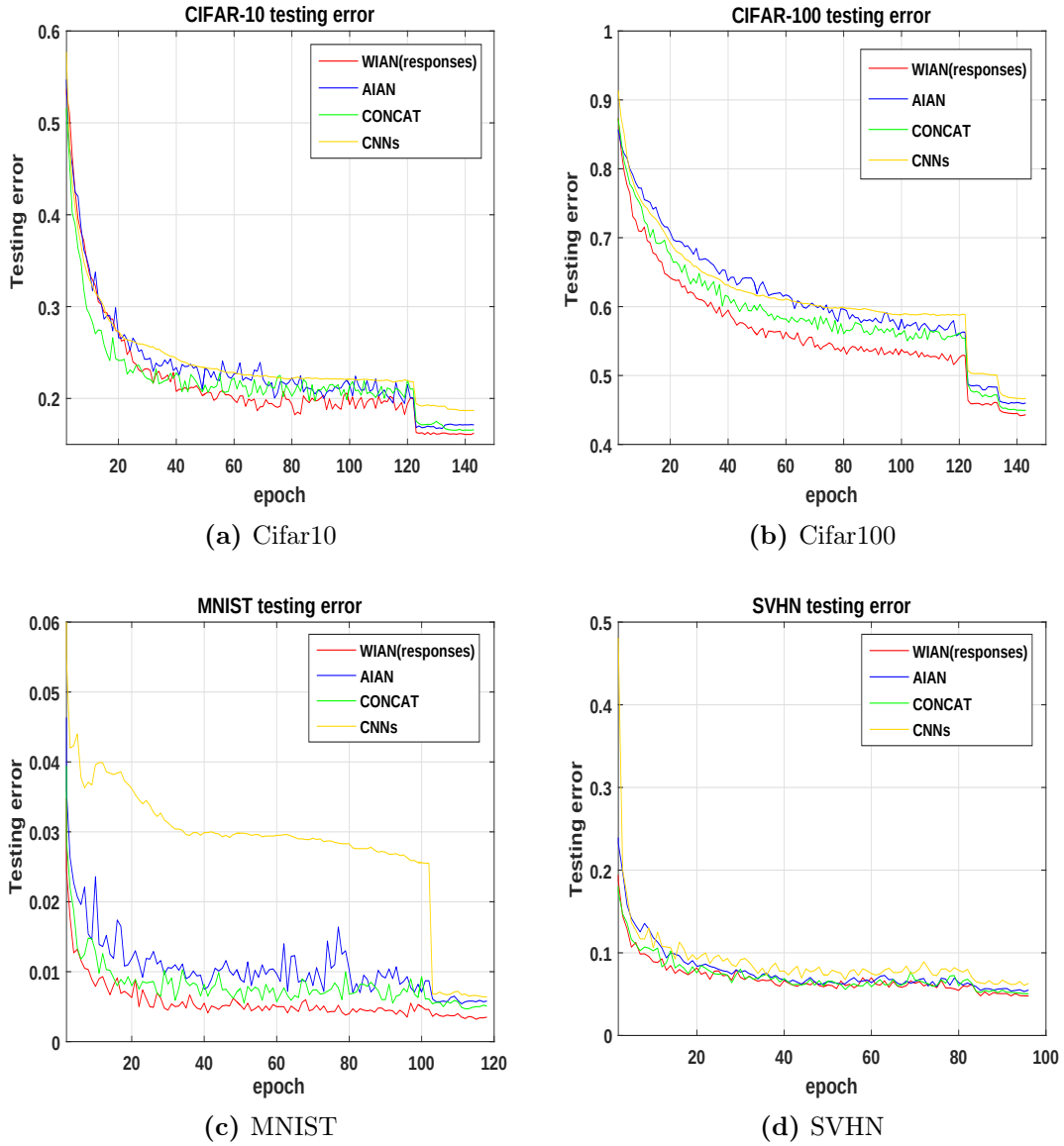
Methods	CIFAR-10	CIFAR-100	MNIST	SVHN
WIAN(responses)	**83.92**	**55.84**	**99.65**	**95.21**
WIAN(entropy)	83.86	55.25	99.58	94.95
WIAN(adaptive learning)	83.15	54.25	99.55	94.6
MIAN	82.75	54.2	99.4	94.55
AIAN	82.9	54.1	99.46	94.68
SIAN	82.6	53.9	99.3	94.52
CONCAT [42]	83.3	55.06	99.51	94.94
CNNs [48]	81.5	53.5	99.3	94.15

Table 5.1: The performance comparison of different convolutional neural network architectures on the four benchmark datasets, CIFAR-10, CIFAR-100, MNIST and SVHN. The number in the table denotes the accuracy of image classification.

5.5.3 Evaluation Results

We present the performance of our proposed WIAN (three weight schemes are evaluated, the first one is based on responses, the second one relies on entropy information and the third one is an adaptive weight learning scheme) and make a comprehensive comparison with general CNNs, Max Integration Architecture Networks (MIAN), Average Integration Architecture Networks (AIAN), Sum Integration Architecture Networks (SIAN) as well as the directly concatenate (CONCAT) of the previous convolutional layers in the CNN architecture. The concatenation operation is similar to the inception module in GoogleNet [42]. A softmax loss function is employed to predict the classification accuracy. The evaluation results of the classification accuracy are listed in Table 5.1.

It turns out that the evaluated integration schemes (WIAN, MIAN, AIAN, SIAN and CONCAT) all achieve improved performance when compared to general CNNs. The WIAN (based on responses, entropy and adaptive weight learning on each layer in the CNN) show much better results than the other approaches. WIAN based on the weight calculated according to the responses on each layer shows the best performance on all the benchmarks. The integration schemes of MIAN, AIAN and SIAN show similar results on the test datasets.



(a) Cifar10 **(b)** Cifar100

(c) MNIST **(d)** SVHN

Figure 5.5: The comparison of the classification error among several possible architectures on the four benchmark datasets.

Additionally, we further investigate the behaviours of the testing error during each epoch in the CNN training. The performance of WIAN (responses), AIAN, CONCAT and the general CNNs are evaluated. The graphs depicted in Figure 5.5 show that WIAN (responses) reaches the smallest testing error faster than others.

This further demonstrates that the weighted integration of previous convolutional layers can boost the performance of the network.

5.6 Conclusions

In this chapter, we propose to reuse the information encoded in previous layers in the network to recover the precision loss due to the pooling operation in the CNN. We present a novel Weighted Integration Architecture Network (WIAN) to enhance the performance of CNN based image classification, where each layer is multiplied by a weight matrix generated according to the responses or entropy of the layer, adaptive learning and then element-wise summed together. The evaluation results demonstrated that the WIAN can yield high accuracy on image classification, and WIAN shows better performance than the scheme that employs direct concatenation, and the schemes employing max, average and sum integration of the previous convolutional layers in the CNN architecture. Moreover, WIAN based on the weight value calculated according to the responses on each layer is more robust than WIAN based on entropy value as well as the adaptive learning scheme.