applied sciences

MDPI

Article

Local Contrast Learning for One-Shot Learning

Yang Zhang †, Xinghai Yuan †, Ling Luo, Yulu Yang, Shihao Zhang and Chuanyun Xu *,†

College of Computer and Information Sciences, Chongqing Normal University, Chongqing 401331, China; zhangyang@cqnu.edu.cn (Y.Z.)

* Correspondence: xcy@cqnu.edu.cn

† These authors contributed equally to this work.

Abstract: Learning a deep model from small data is an opening and challenging problem. In high-dimensional spaces, few samples only occupy an extremely small portion of the space, often exhibiting sparsity issues. Classifying in this globally sparse sample space poses significant challenges. However, by using a single sample category as a reference object for comparing and recognizing other samples, it is possible to construct a local space. Conducting contrastive learning in this local space can overcome the sparsity issue of a few samples. Based on this insight, we proposed a novel deep learning approach named Local Contrast Learning (LCL). This is analogous to a key insight into human cognitive behavior, where humans identify the objects in a specific context by contrasting them with the objects in that context or from their memory. LCL is used to train a deep model that can contrast the recognized sample with a couple of contrastive samples that are randomly drawn and shuffled. On a one-shot classification task on Omniglot, the deep model-based LCL with 86 layers and 1.94 million parameters, which was trained on a tiny dataset with only 60 classes and 20 samples per class, achieved an accuracy of 98.95%. Furthermore, it achieved an accuracy of 99.24% at 156 classes and 20 samples per class. LCL is a fundamental idea that can be applied to alleviate the parametric model's overfitting resulting from a lack of training samples.

Keywords: one-shot learning; artificial neural networks; deep learning; local cognitive context; contrasting

check for updates

Citation: Zhang, Y.; Yuan, X.; Luo, L.; Yang, Y.; Zhang, S.; Xu, C. Local Contrast Learning for One-Shot Learning. *Appl. Sci.* **2024**, *14*, 5217. https://doi.org/10.3390/app14125217

Academic Editors: Nektarios A. Valous and Douglas O'Shaughnessy

Received: 30 March 2024
Revised: 30 May 2024
Accepted: 12 June 2024
Published: 15 June 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Currently, deep learning is always tightly related to big data. It has become a unanimous agreement that the success of deep learning is the result of the big labeled datasets [1,2]. Deep learning provides a kind of end-to-end approach for object recognition that is more generalizable and robust than the traditional methods based on feature extraction, but it is difficult to implement because of its need for large annotated datasets [3]. Building a large training set for deep learning is sometimes so extremely expensive and not acceptable, which hinders the construction of more practical applications.

The machine learning methods of learning from one sample or few samples (which are usually called one-shot learning or few-shot learning [4]; in order to simplify, we call them one-shot learning) attract broad attention. One-shot learning means learning a novel concept from one sample or only a few samples. That is still an open challenge for object recognition due to a lack of training samples to tune the parameters of a learning model, which might result in a serious overfitting problem. Some learning techniques based on different theories have been proposed for tackling the problem based on different theories, such as transfer learning [5], meta-learning [6], metric learning [7], and data augmentation-based learning [8,9]; however, it is still ongoing. Many scholars have conducted relevant research in this field. For instance, Sung Whan Yoon et al. [10] proposed the Neural Network Augmented with a Task-Adaptive Projection method, which utilizes a scenario-based meta-learning strategy and linear projection for image classification. Zhiping Wu et al. [11] extracted features from different layers based

on coarse/fine-grained relation networks, where coarse-grained classification was applied to shallow features followed by fine-grained classification on deep features.

Although few-shot learning enables learning from a limited number of labeled data and recognizing unseen data, it still requires a relatively large number of training data categories during the training process. This still poses a significant challenge in specific specialized domains like medicine [12] and industrial fields [13], where obtaining or creating numerous data categories is frequently arduous and expensive. For instance, in the case of certain rare diseases, the availability of sample data may be extremely limited, posing significant challenges in constructing datasets of substantial scale [14]. In scenarios with limited training sample categories, samples typically occupy only a tiny portion of the high-dimensional space, presenting a significant sparsity issue. However, by treating a sample category as a "reference point" and comparing other category samples against it, the challenge of sample sparsity can be effectively overcome. Within the local space formed around a sample, serving as a reference point, the distribution of samples is relatively dense compared to that in the global space, thereby alleviating the sparsity issue present in the global space. By comparing the relationships between samples within the local space, particularly by observing other category samples relative to a category sample used as a reference point, requiring only a few features for discrimination, sample classification and recognition can be facilitated. Taking Figure 1 as an example (all facial images in the figure are generated by AI), individuals of the same race form a local space, while individuals of all races form a global space. When Asians observe other Asians, they can easily distinguish each other. However, when Asians attempt to differentiate Africans or Americans, they encounter greater challenges. This is due to the fact that, in the cognitive domain of human perception, there is greater familiarity and understanding of the facial features and skin tones of unfamiliar but same-race individuals. However, for unfamiliar individuals of other races, the differences in facial features and skin tones are less obvious due to lack of experience and exposure, resulting in difficulties in differentiation. In the global space, one needs to observe the features of all individuals to differentiate them, whereas, in the local space, differentiation can be achieved with fewer features.

To address the issue of sample sparsity in high-dimensional spaces, we propose a method called Local Contrast Learning (LCL), which is based on the idea of using a sample category as a "reference point" to construct a local space and compare and recognize other category samples. In high-dimensional spaces, it is necessary to identify suitable mapping spaces from multiple low-dimensional spaces for feature extraction. Constructing a low-dimensional projection space based on a specific sample is relatively straightforward, as this low-dimensional space is associated with the reference point. Additionally, many features among samples are similar, such as facial contours or strokes of letters, leading to redundancy when mapping samples from a high-dimensional space to a single low-dimensional space. However, by constructing local spaces, only a small subset of distinct features needs to be mapped to the low-dimensional space, enabling the capture of more subtle information from the samples.

As illustrated in Figure 2, in a three-dimensional space, there exist two sets of local sample spaces, where the reference samples correspond to distinct local mapping spaces. If both sets of samples are mapped to the same global projection space, denoted as α, a significant amount of redundant information will be present among the samples. However, if local spaces are constructed based on specific reference samples, resulting in local projection spaces β_1 and β_2, the two sets of samples will be mapped to distinct local projection spaces. In such local spaces, the differing features among the samples are more easily distinguishable. Furthermore, within a local space, positive contrast samples tend to be more similar to the reference sample, thereby being closer in distance; conversely, negative contrast samples are further away from the reference sample.

In the LCL method, a set of contrastive pairs is inputted into the network together, with one sample serving as the reference point (referred to as the recognized object) to form a local space, and the other samples (referred to as contrastive objects) are compared for recognition. The key insight about LCL is shown in Figure 3. In this paper, LCL constructed an implementation based on Resnet [15,16]; it is called a Local Contrast Neural

Network (LCNN). The LCNN has been shown to achieve higher recognition accuracy even in scenarios with limited data categories.

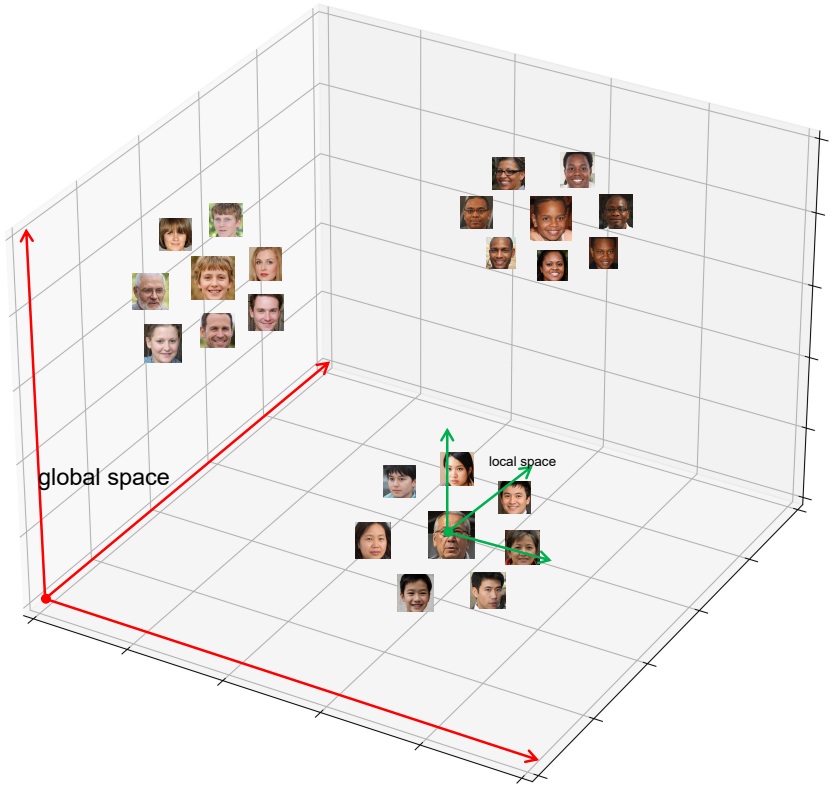


Figure 1. All facial images in the figure are generated by AI. Individuals of the same race form a local space, while individuals of all races form a global space. In the global space, the distribution of individuals exhibits sparsity. However, centering on an individual forms a local space, within which the distribution of individuals becomes denser.

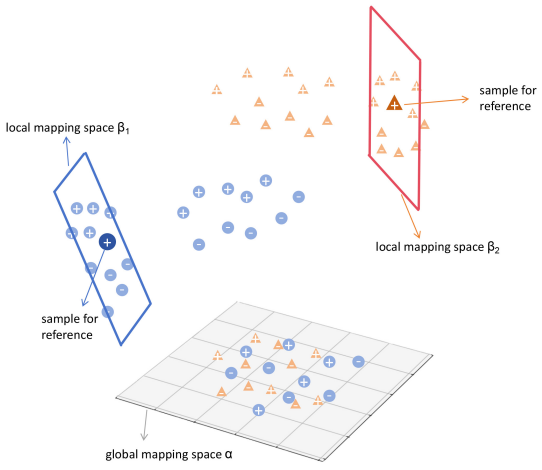


Figure 2. This is a three-dimensional space that includes a global mapping space α, two sets of local space samples, each set with its own reference samples, resulting in two local mapping spaces β_1 and β_2. Blue and yellow represent two groups of samples that are mapped into different local Spaces, each containing positive and negative samples.

The main contributions of this paper are as follows:

1. In response to the issue of sample sparsity in high-dimensional spaces, we propose the idea of constructing local spaces by using a single sample category as the reference center and comparing it with other category samples for recognition.
2. Building upon the concept of local spaces, we further introduce the Local Contrast Learning method. This method first establishes a local contrastive context, and then groups the recognized object with each contrastive object in the local context to form contrastive pairs. Finally, these contrastive pairs are iteratively fed into the network model for contrastive learning.
3. We have provided an implementation of LCL called Local Contrast Neural Network (LCNN). The LCNN demonstrates strong capabilities in few-shot learning. LCNN is suitable for training on small datasets, exhibiting outstanding generalization and overfitting resistance. It achieved an accuracy of 99.24% on the small Omniglot dataset containing 156 classes and up to 98.95% accuracy on the Omniglot dataset with only 60 classes.

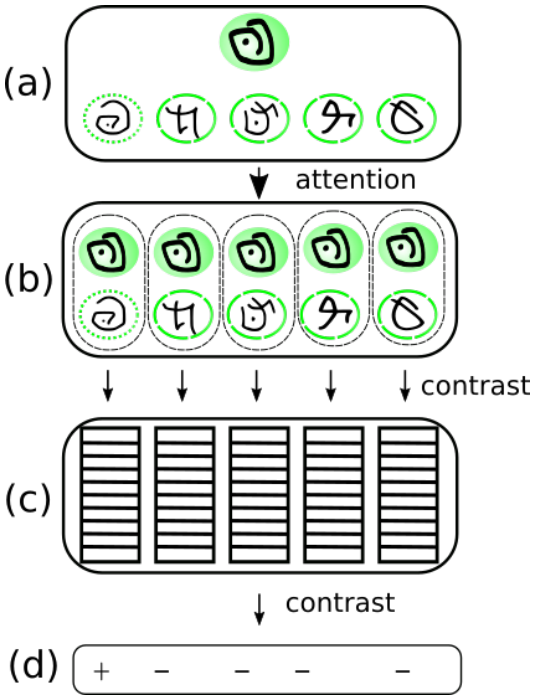


Figure 3. The key insight about Local Contrast Learning. (**a**) The local contrast context with one recognized object and five contrastive objects. (**b**) The contrastive pairs are built by focusing attention on every contrastive object. (**c**) The difference representations are created by contrasting every contrastive pair. (**d**) Recognition results obtained by contrasting the different representations. Specifically, the representation in the comparison object is the same as the recognition object, denoted by "+", while the recognition results of the other comparison objects are denoted by "−".

2. Related Work

2.1. One-Shot Learning

It is a fundamental challenge to learn to recognize a novel category from one sample or a few samples. The key challenge of one-shot learning is how to tackle the overfitting problem due to the lack of training samples [17]. In particular, it is more serious when the recognition model is a parametric model, such as a deep neural network. The one-shot learning problem has been widely addressed by transfer learning, meta-learning, metric learning, and data augmentation-based learning.

2.1.1. Transfer Learning

Transfer learning leverages knowledge from related tasks to aid new tasks, using abundant source domain data to learn general features applicable to small-sample tasks

in the target domain. Yuan Tai et al. [18] introduced a method using a non-attentive module for feature transfer. On the other hand, Moslem Yazdanpanah et al. [19] used feature normalization in the source domain and batch normalization during target domain adaptation. Nan Sun et al. [20] introduced a two-stage learning method that combines metric-based learning with a deep adaptation module, thus achieving seamless integration.

2.1.2. Meta-Learning

Meta-learning is a method that enables models to learn how to learn, by training on multiple tasks, allowing the model to quickly learn new tasks from a large pool of tasks. Xingyou Song et al. [21] proposed an evolutionary strategy-based algorithm to address second-order derivative estimation issues. Additionally, Sungyong Baik et al. [22] introduced the Learning to Forget (L2F) method to handle task conflicts by selectively forgetting initialized parameters. They also proposed a framework with adaptive loss functions for different tasks [23].

2.1.3. Metric Learning

Metric learning focuses on transforming samples into feature vectors and classifying them by distances or similarities between these vectors. Zhili Qin et al. [24] developed a multi-instance multi-head attention module to transform the problem into a multiple-sample learning issue. Jiangtao Xie et al. [25] proposed the deep brown distance covariance method, which learns image representations by measuring the similarity between the joint distribution and the marginal distribution of embedding features of query and support images. Additionally, Arman Afrasiyabi et al. [26] introduced the SetFeat method for set feature extraction, utilizing set-to-set matching metrics for training and inference of small-sample image classification.

2.1.4. Data Augmentation

Data augmentation increases the quantity of training data and enhances data diversity. Basic data augmentation techniques include translation, rotation, scaling, and flipping. However, these low-level augmentation methods may not be sufficient for small-sample tasks. To address this issue, Kai Li et al. [27] proposed adversarial feature hallucination networks, which employ conditional Wasserstein generative adversarial networks to generate additional samples. Zhiwu Lu et al. [28] introduced a novel data synthesis strategy that generates new data by perturbing semantic class prototypes. Recently, Uche Osahor et al. [29] proposed low-displacement rank regularization to mitigate overfitting in small-sample models and established three types of data augmentation strategies: support, query, and task.

2.2. Multi-Column Neural Network Shared Weights

A Local Contrast Neural Network (LCNN) is a multi-column neural network with shared weights (MNNW) among all columns. MNNW is distinguished from the multi-column neural network [30] that does not share weights among all columns. MNNW consists of multiple columns, which means that it has multiple dataflow paths from input to output, and the columns share weights. There are two representative architectures of MNNW; one is Siamese architecture [31–34] and another is triplet architecture [35–37]. MNNW is good at learning an embedding for metric learning [31,38–40] by comparing input samples of all columns. In order to train the embedding function, the contrastive loss function is designed in such a way that the minimization will decrease the energy of genuine pairs and increase the energy of impostor pairs. In this paper, we use a similar way of designing the loss function. Mahmoud Assran et al. [41] proposed a masked Siamese network for learning image representations by integrating invariance-based pre-training and mask denoising techniques.

Unlike Siamese networks with two columns, triplet networks have three columns, feeding three samples: one object and two contrast samples (one genuine, and one im-

postor). Triplet networks have been used in various tasks, such as remote sensing image retrieval [42], camouflaged object detection [43], and face detection using a Swin transformer [44].

LCNN does not learn similarity metrics like the approaches above, because we argue that it is difficult to learn a global similarity metric in a one-shot setting. LCL inputs pairs of samples into a branch and performs feature mapping by using one of the samples in the pair as a reference. This approach is distinct from Siamese and triplet networks, which input individual samples into a branch and perform feature extraction in the global feature space. In the global feature space, all samples are mapped to a single low-dimensional space. However, with a reference sample, the mapped low-dimensional space changes with each different reference sample. The purpose of inputting a pair of samples into the network is to construct a local space for feature mapping based on the reference sample. Thus, even for the same sample, the mapped space will change if the reference sample changes because the projection feature plane is associated with the reference sample.

3. Methodology

3.1. Local Contrast Learning

Local Contrast Learning is an approach used to learn a parameterized model to identify objects in a local context by contrasting objects one by one. Recognition is always carried out in a local context that must be a small scenario only with a few objects (the number of objects is usually less than seven) [45]. LCL draws on two key ideas from human cognitive behavior: (1) local context—recognition must be based on a local context, and only depend on the local context. If the context cannot provide enough information, the new local context must be built by achieving new information or recalling new memories. (2) Contrast—in order to identify a recognized object, the recognition is iteratively executed by contrasting the recognized object and a contrastive object; the contrast results based on the different contrastive objects are compared again among them.

LCL firstly generates a large number of contrastive sample groups by randomly sampling classes and exemplars from a training set and shuffling them and then feeds the sample groups into a parametric model to encode the differences; lastly, it outputs the activations of the contrastive samples by contrasting the differences. LCL can learn recognization in different local contexts rather than a single global context. The workflow of LCL is shown in Figure 4. LCL consists of three major components: Contrast Cognitive Context Constructor, Difference Embedding Generator, and Difference Perceptron.

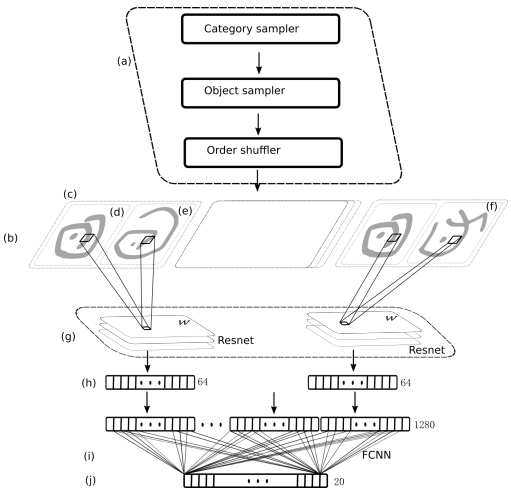


Figure 4. Workflow of local contrast learning. (**a**) Contrasting cognitive context constructor including Category Sampler, Object Sampler, order Shuffler; (**b**) Local Cognitive Context; (**c**) contrastive pair; (**d**) recognized object; (**e**) contrastive positive object; (**f**) contrastive negative object; (**g**) Difference Embedding Generator that is a parametric model with weights shared; (**h**) difference embedding; (**i**) Difference Perceptron; (**j**) contrastive pair local activation.

3.1.1. Contrasting Cognitive Context Constructor

The contrasting cognitive context constructor is responsible for constructing the Local Cognitive Context (LCC) from the labeled support set (training data), denoted as $S = \{S_1, \ldots, S_c, \ldots, S_{SC}\}$ and $S_c = \{(x_{ci}, y_c)\}_{i=1}^{K}$, where K is the number of training images in each category (in order to simplify the analysis, in this paper, it is supposed there is the same number of training images in every category; however, LCL allows different numbers). Here, $x_{ci} \in \mathbb{R}^{W \times H}$ is the ith training image (or sample) in the cth category, W is the width of an image, and H is the height of an image. Here, $y_c \in \mathbb{C}_S = \{c_1, \ldots, c_c, \ldots, c_{SC}\}$ is the corresponding label and SC is the number of the categories. The Local Cognitive Context $LCC = \left((x_c, y_c), \{(x_i, y_i, z_i)\}_{i=1}^{L}\right)$ is defined by a recognized object (x_c, y_c), and a set of contrastive objects $\{(x_i, y_i, z_i)\}_{i=1}^{L}$. Here, L is the number of contrastive objects in an LCC. In the set of contrastive objects, there is one and only one sample whose category is equal to y_c and that is called a contrastive positive object (the z_i is set to 0), but there is not a sample that is equal to x_c. The contrastive objects, except for this contrastive positive object, are called contrastive negative objects (the z_i is set to 1). There are not two samples with the same category in the contrastive objects.

The Local Cognitive Context (LCC) is constructed from a labeled support set (training set) and is crucial for ensuring the effective training of the LCL method, especially when dealing with small datasets. By creating a diverse LCC, the LCL method enhances the robustness of the model, thereby improving its recognition and classification capabilities and overall performance in one-shot learning tasks.

Algorithm 1 can construct a large amount of different Local Cognitive Contexts by two random selections and one shuffle. The amount of Local Cognitive Contexts is extremely crucial for LCL. The number of contrastive categories combinations in Step 1 is

$$C_L^{SC} = \binom{SC}{L} = \frac{SC!}{L!(SC-L)!} \tag{1}$$

Algorithm 1: Build procedure for Local Cognitive Context

Data: A labeled support set $S = \{(x_i, y_i)\}_{i=1}^{K}$, and the number of contrastive objects L.

1. Randomly sample L categories from $\mathbb{C}_S$ as contrastive categories.
2. Select the first category as a contrastive positive category from the contrastive categories, and the other as contrastive negative categories.
3. Iteratively sample one image from each contrastive negative categories as a contrastive negative object.
4. Sample two images from the contrastive positive category; one is a recognized object (x_c, y_c) and another is a contrastive positive object.
5. Concatenate the contrastive positive objects and the contrastive negative objects as contrastive objects $\{(x_i, y_i, z_i)\}_{i=1}^{L}$.
6. Randomly shuffle $\{(x_i, y_i, z_i)\}_{i=1}^{L}$

The number of contrastive negative object combinations for each contrastive negative category is K. The number of contrastive positive object combinations for each contrastive positive category is Equation (2). So, the number of contrastive object combinations is Equation (3).

$$C_2^K = \binom{K}{2} = \frac{K!}{2!(K-2)!} = \frac{1}{2}K \times (K-1) \tag{2}$$

$$N_{co} = \frac{SC!}{L!(SC-L)!} \times \left((L-1) \times K + \frac{1}{2}K(K-1)\right) \tag{3}$$

In Step 6, the contrastive objects are randomly permutated, so the number of each contrastive object permutations is $L!$. Consequently, the size of the learning sample space for LCC is

$$N_{LCC} = N_{co} \times L! = \frac{SC!}{(SC-L)!} \times \left((L-1) \times K + \frac{1}{2} K(K-1) \right) \tag{4}$$

For example, in the small Omniglot [46], the number of training categories SC equals 136, the number of training images in each category K equals 20, and the number L of contrastive objects in an LCC equals 20; then, N_{LCC} is 1.18×10^{44}, which is an extremely large number.

It is a critical factor to ensure that LCL training does not easily lead to overfitting in the face of a small training set and that a contrasting cognitive context constructor can provide a large enough number of LCCs to train the LCL model.

3.1.2. Difference Embedding Generator

LCL is used to learn a parameterized model to embed the difference of the contrastive pair in a low-dimensional space. This is remarkably distinguished from the deep metric learning [47,48]. It is difficult to attempt to learn a global similarity metric to make the distance between the positive object and the cognitive object always larger than the distance between the negative object and the cognitive object, even if humans are almost incapable of ranking the similarities of more than seven objects.

The Difference Embedding Generator is responsible for comparing the identified object with each contrast object in the LCC and generating a vector that represents the differences between them. The Difference Embedding Generator first creates contrastive pairs $\{((x_c, y_c), (x_i, y_i, z_i))\}_{i=1}^{L}$ by grouping the recognized object with each contrastive object in the LCC; then, the contrastive pairs are iteratively fed into an identical neural network in order to obtain the difference embedding of contrastive pairs, theoretically, which may be any kind of neural network; in this paper, we used Resnet [16]. The difference embedding is a vector (in this paper, the vector dimensionality is 64). Here, we obtained a total of L difference embeddings, as described in Equation (5). Each difference embedding is derived through the comparison of the recognized object with each contrastive object by the Difference Embedding Generator, as detailed in Equation (6).

$$DE = (DE_1, \ldots, DE_i, \ldots, DE_L) \tag{5}$$

$$DE_i = deg((x_c, y_c), (x_i, y_i, z_i)) \tag{6}$$

Here, deg is the Difference Embedding Generator.

It has to be emphasized that (1) it is a pair of objects that is inputted into an embedding neural network; one is a recognized object and another is a contrastive object (positive or negative), and this is comparable to [34], while it is different from the Siamese network [49,50] or triplet network [35], in which the contrastive objects are inputted into the embedding network one by one; (2) the Difference Embedding Generator outputs an embedding in low-dimensional space representing the difference between a recognized object and contrastive object, instead of an embedding representing the object for feeding into the similarity function [35,51,52].

3.1.3. Difference Perceptron

Difference Perceptron is a response for mapping a group of difference embeddings to a vector whose elements represents Contrastive Pair Local Activation (CPLA) which means the relative similarity between each contrastive object and the recognized object in a Local Cognitive Context.

$$CPLA = (CPLA_1, \ldots, CPLA_i, \ldots, CPLA_L) = dp(DE) \tag{7}$$

Here, dp is the Difference Perceptron. DE is the difference embedding. The Difference Perceptron is a parametric model, such as a fully-connected neural network. It is emphasized that all DE are inputted into Difference Perceptron once, instead of one by one. $CPLA_i$ is not determined only by DE_i, but by all difference embeddings.

The dimensionality of the Contrastive Pair Local Activation is the same as contrastive objects and which element corresponds to each contrastive pair in a Local Cognitive Context. The activation of contrastive positive objects should be distinguished from contrastive negative objects and the rank of activation in all contrastive negative objects is meaningless. Local Contrast Learning does not aim to train a global similarity metric for contrastive objects, instead of locally identifying the contrastive positive objects in Local Cognitive Context. In this paper, we train the activation of contrastive negative objects to be near one and the activation of contrastive positive objects to be near zero.

3.1.4. Learning

There are two groups of trainable parameters in the LCC; one group is in the Difference Embedding Generator and another is in the Difference Perceptron. The goal of LCC training is to turn the parameters for expanding the activation gap between the contrastive negative object and the contrastive negative object in a Local Cognitive Context.

3.2. Local Contrast Neural Network

The Local Contrast Neural Network (LCNN) is an implementation for verifying the performance and analyzing the characteristics of the LCL. In this paper, an LCNN will be used to execute the one-shot learning task.

3.2.1. Model

In a one-shot learning task, a labeled support set (also called a training set), denoted as S, is given for training a parameterized model, and one test set for evaluation. Similar to Lake's [46] definition of one-shot learning, the categories of the support and the test sets are mutually exclusive.

The network architecture of LCNN is shown in Figure 5. The model firstly creates a Local Cognitive Context $LCC = \left((x_c, y_c), \{(x_i, y_i, z_i)\}_{i=1}^{L}\right)$ according to Algorithm 1, then constructs contrastive pairs $\{(x_c, x_i)\}_{i=1}^{L}$ by grouping the recognized object with each contrastive objects in LCC and outputs expected $Z = \{z_i\}_{i=1}^{L}$, where z_i equals 0 if the corresponding x_i is a positive object, otherwise z_i equals 1.

Contrastive pairs $\{(x_c, x_i)\}_{i=1}^{L}$ are iteratively inputted into a Resnet [16] to obtain the difference embedding. Resnet is, by nature, a deep convolutional neural network (CNN) [53–55], so a recognized object x_c and contrast object x_i are separate as an input channel of CNN, that is to say, the Resnet only has two channels. (Nevertheless, the multiple channels of the images can be simply concatenated together one by one if they are color images). The output of the Resnet is a difference embedding corresponding to a contrastive pair (x_c, x_i) that is a vector, the dimensionality of which is determined by the number of channels of the image and the number of the building blocks in the Resnet.

All difference embedding of an LCC are concatenated as a vector that is fed into the Difference Perceptron, which is a fully connected neural network, to produce the Contrastive Pair Local Activation denoted as $A = \{a_i\}_{i=1}^{L}, a_i \in \mathbb{R}$. The local activation a_i corresponds to the contrastive pair (x_c, x_i), yet the relation is built by training on the support set instead of a hard connection. The local activation a_i is activated by the sigmoid function.

The index of this contrastive positive object can be computed by the following equation:

$$i_{cpo} = arg_i min(CPLA_i) \tag{8}$$

LCNN uses a pre-activation Resnet as a Difference Embedding Generator, which is constructed by many pre-activation residual units [15]. The pre-activation residual unit consists of two activation functions (ReLU and BN [56]) and two 3 × 3 convolutional

layers. The last two layers of the Difference Embedding Generator are the activation function (ReLU and BN) and global average pooling [57] that outputs 64 dimension vectors. A residual unit is also introduced in the initial part of the Difference Embedding Generator. The LCNN model only has three hyperparameters: the steps of the learning rate decay d_1 and d_2, and maximum training steps m.

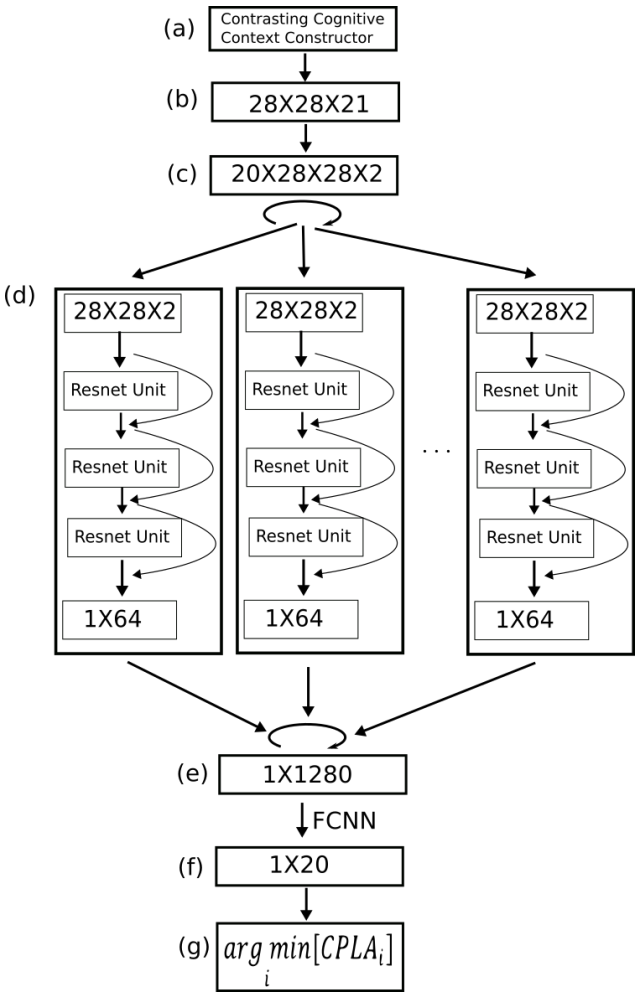


Figure 5. The network architecture of a Local Contrast Neural Network. (**a**) Contrasting cognitive context constructor. (**b**) Input included 20 contrastive objects and one recognized object from Local Contrast Context where 28 is the image size. (**c**) 20 contrastive pairs. (**d**) Difference Embedding Generator, which is a Resnet to be iteratively fed a contrastive pair. (**e**) Concatenating Difference Embedding. (**f**) Contrastive Pair Local Activation. (**g**) Minimum Contrastive Pair Local Activation as the positive object.

3.2.2. Loss Function

The LCNN is designed to identify an object that belongs to the same category as the recognized object in a local context, so it is expected that the LCNN's local activation at the corresponding position of the positive object is distinguished from the others. Therefore, according to the output expected, the expected local activation at the position of the positive object is zero, and the expected local activation at the position of the positive object is one. Based on cross entropy, the loss function is defined as

$$L_c(W) = \frac{1}{N}\sum_{1}^{N}\frac{1}{L}\left(\sum_{1}^{L} -(z_i \log a_i + (1 - z_i)\log(1 - a_i))\right) \quad (9)$$

where N is the number of LCCs in one mini-batch, and L is the number of the contrastive object in one LCC. W is the parameters of LCNN including the parameters W_{deg} of the Difference Embedding Generator and the parameters W_{dp} of the Difference Perceptron. The loss function can change to another form:

$$L_c(W) = L_N + L_P \tag{10}$$

$$L_P(W) = -\frac{1}{N}\sum_{1}^{N}\left[\log\left(1-a_i\right)|_{z_i=0}\right] \tag{11}$$

$$L_N(W) = -\frac{1}{N}\sum_{1}^{N}\frac{1}{L-1}\left[\sum_{1}^{L}\log\left(a_i\right)|_{z_i=1}\right] \tag{12}$$

where $z_i = 1$ means that the ith contrastive object in the LCC is negative and L_N penalizes its local activation if it is too small. $z_i = 0$ means that the ith contrastive object in the LCC is positive and L_P penalizes its local activation if it is too large. $L_c(W)$ is a contrastive loss function [58–60], and, by minimizing the $L_c(W)$, the local activation gap between the negative objects and the positive object can be enlarged. In order to improve the model's generalization and to speed up learning, a parameter regularization loss term is added to the $L_c(W)$, and the regularized loss is

$$L(W) = L_c + \frac{1}{2}\lambda \parallel W \parallel^2 = L_c + \frac{1}{2}\lambda \parallel W_{deg} \parallel^2 + \frac{1}{2}\lambda \parallel W_{dp} \parallel^2 \tag{13}$$

where λ is a regularization parameter that is set to 0.0002 in all experiments of this paper.

3.2.3. Weight Initialization

LCNN, in essence, is an end-to-end deep artificial neural network; it typically has more than 30 layers in this paper's experiments, so it is crucial to avoid exploding or diminishing by reaching the final layer. The variance scaling initializer [61] was used to initialize all weight parameters. It draws weight parameters from a uniform distribution within $[-limit, limit]$, where $limit$ is $\sqrt{6/fan_in}$ where fan_in is the number of input units.

3.2.4. Optimization

In order to minimize the loss $L(W)$, the stochastic gradient descent with momentum optimization [62–65] algorithm was used, as in [16]. The momentum of the optimized algorithm is fixed to 0.9. The learning rate starts from 0.1 and, after d_1 and d_2 iterations, the learning rate is separately set to 0.01, 0.001; d_1 and d_2 are two hyperparameters.

3.2.5. Few-Shot LCNN

One-shot recognition is an extreme case; however, in industrial applications, few-shot recognition is more practicable. It is easy to extend LCNN to few-shot recognition.

The architecture of a few-shot LCNN is shown in Figure 6. Firstly, the contrasting cognitive context constructor is used to create a few-shot Local Cognitive Context that is defined as Equation (14), where T is the number of shots and (x_{cj}, y_{cj}) is a recognized object. Then, LCC_{fs} is regrouped into multiple LCCs that are sequentially fed into identical LCNNs that output CPLA for every LCC. Finally, all CPLAs are summed up as one CPLA for LCC_{fs}.

$$LCC_{fs} = \left(\left\{\left(x_{cj}, y_{cj}\right)\right\}_{j=1}^{T}, \left\{\left(x_i, y_i, z_i\right)\right\}_{i=1}^{L}\right) \tag{14}$$

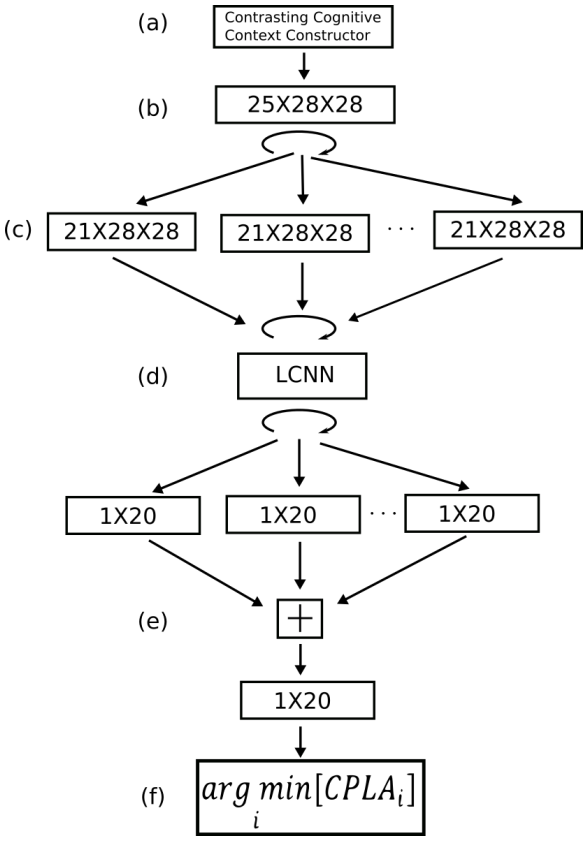


Figure 6. The network architecture of LCNN for few-shot recognition. The figure shows the architecture for a 20-way 5-shot classification. (**a**) Contrasting cognitive context constructor, which builds LCCs with few recognized objects. (**b**) The few-shot LCC with 5 recognized samples. (**c**) Five LCCs built by unstacking the few-shot LCC. (**d**) Each LCC is iteratively inputted into an LCNN to generate CPLAs. (**e**) All CPLAs are summed as CPLAs of the few-shot LCC. (**f**) The contrastive sample with a minimum CPLA is recognized as a positive sample.

4. Experiments and Results

A few comprehensive experiments on LCL and LCNN were carried out to evaluate its outstanding capabilities, including one-shot learning, small data learning, and transfer capability test.

4.1. Classification Evaluation

In order to evaluate the one-shot (including few-shot) classification performance of the LCNN, we carried out the classification experiments on two benchmark datasets: Omniglot [46], and CASIA-HWDB1.1 [66,67]. According to conventional one-shot test protocols [46], all categories of the datasets were divided into two subsets: a training set (background set) and test set, and the categories of the training set were disjoint with the categories of the test set. The model was trained on the categories of the training set and tested on the categories of the test set.

The learning rate decayed to 0.01 from 0.1 when the learning steps were larger than d_1, and then decayed to 0.001 when the learning steps were larger than d_2 (here, d_1 and d_2 are two hyperparameters). The maximum training steps m is a hyperparameter. If not specified, in the following experiments, the steps of the learning rate decay d_1 and d_2 were separately set to 44,800 and 51,200, and m was 57,600. The number of the layers of the LCNN was 86. The training mini-batch size was 40.

Every test result performed on the same test set might be different due to three kinds of randomness and computational errors, so we ran the test 100 times on the test set and we reported the mean classification accuracy with a 95% confidence interval.

4.1.1. Omniglot

The Omniglot dataset consists of images across 1623 classes with only 20 images per class, from 50 different alphabets in total. The dataset is divided into a background (training or support) set of 30 alphabets with 964 characters and an evaluation (test) set of 20 alphabets with 659 characters. According to the suggestion from Lake et al. [46], only the background set should be used for training, and one-shot learning results should be reported using alphabets from the evaluation set.

Except for the training set of 30 alphabets with 964 characters, Lake et al. [46] have provided smaller background sets, "background small 1" and "background small 2". Each of these contains just five alphabets and, respectively, contains 136 characters and 156 characters. Furthermore, to see how the models performed with more limited training classes, we built a tiny subset by picking up the first and second characters from each alphabet of the Omniglot background set, which was called "background tiny". The subset has 60 characters and 1200 samples.

There are a few variant tests proposed by Vinyals, O. et al. [68] and Santoro, A. et al. [69], which have different splits for the background set and evaluation set and different test trails from the BPL test protocol [46]. Vinyals randomly picked up 1200 characters from Omniglot's background set and evaluation set as the training set and 423 characters as a test set. In Vinyals's split, the training set and test set have identical alphabets but have different characters. However, in the BPL test protocol, the training set and test set have completely different alphabets, that is to say, a model trained on some alphabets was used to classify some other alphabets. It is more challenging to recognize the images in the novel alphabet than in novel characters in identical alphabets. Some research has also followed Vinyals's, such as such as Matching Nets [68], MAML [70], MetaNet [71], CNPs [72], TAML [73], Cross-way training on Prototypical Networks [74], EDANet [75], and MSRN [76]. All test trials were only constructed from the test set and each trial consisted of a few characters picked from the test set. Only one image of each character was picked as a contrastive object, and one or a few images were picked as recognized objects.

MetaNet [71] was trained and tested on BPL's split of 30 training alphabets with 964 classes; however, it formed 400 trials from the evaluation classes to test the model, so it cannot completely match BPL's result, which is evaluated on standard 400 trials.

In order to extensively evaluate the performances of the LCNN and try to best match other approaches, in this section, we trained and tested the LCCN separately on the test protocol provided by Vinyals, O. et al. [68] with 1200 characters and on the test protocol provided by Lake [46] with 964 characters. Furthermore, in order to show the advantages of LCNN on small data, we trained the LCNN on the first 60 characters and the first 156 characters of the 1200 training characters. We also trained the LCNN on "background tiny" and "background small2", but formed trials from the evaluation set rather than the standard 400 trials provided by Lake.

In all experiments in this section, the test trials were created from the test set using Algorithm 1, and the number of test trials n_{trial} was computed by Equation (15).

$$n_{trial} = (EC \times KE)/(L + n_{shot}) \tag{15}$$

Here, EC is the number of characters in the test set. KE is the number of samples per character. L is the number of contrastive objects in an LCC, and n_{shot} is the number of shots.

We carried out five-way and twenty-way one-shot and five-shot classifications, and the comparisons with published results are shown in Table 1. The models labeled 60, 156, and 1200 were trained on the test protocol provided by Vinyals, O., with 60, 156, and 1200 training characters and 423 test characters, in which the training characters and test characters belong to the same alphabets. The models labeled 964, small2, and tiny were trained on the test protocol provided by Lake, respectively, with 964, 156, and 60 training characters and 659 test characters, in which the training characters and test characters belong to completely different alphabets.

From Table 1, it can be observed that the LCL method requires a smaller dataset, using only 156 categories. In the five-way one-shot task, its performance is slightly lower compared to Relation Nets, TAML, and MSRN, which utilize 1200 categories. However, in the 20-way one-shot task, our method shows a 0.4% improvement compared to Relation Nets and a 0.6% improvement compared to MSRN, surpassing all other methods in terms of accuracy. Moreover, when LCNN is used with 1200 categories, there is a 0.2% increase in accuracy in the 20-way one-shot task. Our method also exhibits smaller variance in accuracy. In the five-way one-shot task, the variance of accuracy is approximately 0.05%, while in the 20-way one-shot task, the variance is around 0.13%, both of which are lower than those of other methods. This reflects the superior stability of the models trained by our approach.

The advantage of the LCL method lies in its ability to achieve desirable results on very small datasets. The LCNN outperformed some prior approaches by only using 156 characters; furthermore, the results using 60 characters are comparable to published results, which used 1200 characters or 964 characters. However, the LCL method does not require extensive training data. Comparing the results using 156 characters and the results using 1200 or 964 characters, the latter is not obviously higher than the former, which demonstrates that, to train LCNN on Omniglot, 156 characters are almost enough, and more data cannot improve the performance much. That proves that the LCNN can learn enough knowledge from small data to discriminate objects. Conversely, more training data might import more noises that confuse the models and decline their performance; for instance, the five-way one-shot accuracy of 98.97% of the model LCNN 1200 is slightly lower than the accuracy of 99.24% of the model LCNN 156.

Comparing the models LCNN tiny, LCNN small2, LCNN 964, and models LCNN 60, LCNN 156, and LCNN 1200, the accuracies of the former are lower than the latter. That proves that it is more challenging to recognize the images in the novel alphabet than in novel characters in identical alphabets.

Table 1. Twenty-way one-shot and few-shot accuracy (%) on Omniglot based on variant test protocol. Results are accuracies averaged over 100-time runs with 95% confidence intervals where reported. '-': not reported. The bolded result indicates the best result in our experiment.

Model	Chars	5-Way Accuracy		20-Way Accuracy	
		1-Shot	5-Shot	1-Shot	5-Shot
Matching Nets [68]	1200	98.1	98.9	93.8	98.7
MAML [70]	1200	98.7 ± 0.4	99.9 ± 0.1	95.8 ± 0.3	98.9 ± 0.2
MetaNet [71]	1200	98.95	-	97	-
Relation Nets [77]	1200	99.63 ± 0.2	99.8 ± 0.1	97.6 ± 0.2	99.1 ± 0.1
CNPs [72]	1200	95.3	98.5	89.9	96.8
TAML [73]	1200	99.47 ± 0.25	99.83 ± 0.09	-	-
Cross-way training on Prototypical Networks [74]	1200	98.73 ± 0.19	99.64 ± 0.07	95.21 ± 0.17	98.77 ± 0.05
EDANet [75]	1200	98.61	99.13	-	-
MSRN [76]	1200	99.35 ± 0.25	99.70 ± 0.08	97.41 ± 0.28	99.01 ± 0.13
Attention mechanism and weight fusion [78]	1200	-	-	97.8	99.2
LCNN 60	60	98.95 ± 0.06	99.70 ± 0.03	96.84 ± 0.16	99.05 ± 0.10
LCNN 156	156	**99.24 ± 0.05**	**99.77 ± 0.03**	98.03 ± 0.13	99.31 ± 0.09
LCNN 1200	1200	98.97 ± 0.05	99.76 ± 0.03	**98.28 ± 0.12**	**99.46 ± 0.07**
MetaNet [71] 964	964	98.45	-	95.92	-
LCNN tiny	60	95.36 ± 0.09	99.04 ± 0.05	95.88 ± 0.20	98.85 ± 0.11
LCNN small2	156	98.54 ± 0.06	99.63 ± 0.04	97.01 ± 0.13	99.15 ± 0.08
LCNN 964	964	98.86 ± 0.09	99.70 ± 0.07	97.77 ± 0.22	99.33 ± 0.18

4.1.2. HWDB

Like the test on Omniglot, we evaluated LCNN on CASIA-HWDB1.1 [66,67], which is a Chinese handwritten character dataset for machine learning and is more diversified

and confusing than Omniglot. CASIA-HWDB1.1 is a widely used dataset that includes 3755 character classes and 300 images per class written by 300 calligraphers. Therefore, in order to extensively evaluate the performance and provide a benchmark for other one-shot classification approaches on the handwritten Chinese characters, we defined a one-shot classification of the handwritten Chinese characters. CASIA-HWDB1.1 includes a training subset with 240 images per character and a test subset with 60 images per character. However, we only used the first 20 images per character in the training subset to evaluate the performance of one-shot classification.

Figure 7 presents examples of recognition classification errors along with visual comparisons. These examples clearly demonstrate the potential classification errors that the model may encounter when faced with similar contrast objects. For example, in the example from the first row, the handwriting forms of the first pair of positive and negative contrast objects are visually similar and difficult to differentiate. Similarly, in the example from the third row, the handwriting forms of the second pair of positive and negative contrast objects also exhibit significant visual similarity.



Figure 7. CASIA-HWDB classification examples. The first and fourth columns are recognized objects. The second and fifth columns are positive contrastive objects, and the third and sixth columns are negative contrastive objects. The images were selected from the false classifications.

We picked the last 600 characters as a test set out of 3755 characters (the characters are ordered by GB2312 code) and, respectively, picked the first 60, 200, and 3155 characters as three different training sets: HWDB60, HWDB200, and HWDB3155. All images were resized to 64 × 64 and not augmented. In all experiments in this section, the test trials were created using Algorithm 1 on the test set, and the number of test trials n_{trial} was computed by the equation in the previous section.

To compare the performance of HWDB with other neural network approaches, we selected Matching Networks [68] as a baseline. The setup of Matching Networks is the same as the setup in the previous section; however, the maximum training steps is 315,500.

We carried out 20-way one-shot and five-shot classifications. The results comparing the baselines to the LCNN are shown in Table 2. In all experiments, the LCNN outperforms the baseline. The LCNN with 60 characters achieves an accuracy of 89.85%, which outperforms the accuracy of 85.83% of Matching Networks with 3155 characters. That demonstrates that the LCNN can achieve high performance using small data. Comparing the results of LCNN HWDB200 to LCNN HWDB3155, we can find that LCNN HWDB200 achieves almost the same one-shot and five-shot classification accuracy only using six percent of the samples of LCNN HWDB3155. That, again, demonstrates that the LCNN can learn enough knowledge from small data to discriminate objects.

Table 2. Twenty-way classification accuracy (%) on HWDB. Results are accuracies averaged over 100-time runs and with 95% confidence intervals where reported. The models labeled by HWDB60, HWDB200, and HWDB3155 were, respectively, trained on the first 60, 200, and 3155 characters of HWDB with 20 samples per character. '-' means no result.

Model	Chars	1-Shot	5-Shot
Matching Nets			
HWDB60	60	69.78	-
HWDB200	200	81.67	-
HWDB3155	3155	85.83	-
LCNN			
HWDB60	60	89.85 ± 0.26	97.70 ± 0.13
HWDB200	200	96.33 ± 0.13	99.41 ± 0.06
HWDB3155	3155	97.32 ± 0.16	99.60 ± 0.06

4.2. Small Data Deep Learning

In order to demonstrate the performances on a small training dataset, we designed a series of experiments on Omniglot [46] and CASIA-HWDB1.1 [66]. We constructed the different experiments by adjusting the parameters of training datasets, including the number of classes and the number of samples per class, the number of training steps, the number of network layers, and the number of shots.

In all experiments, if not specified, the number of the network layers was 26 and the number of parameters was about 488 k, d_1 and d_2 were separately set to 9600 and 12,800 m was 16,000, and the image size was 64 × 64.

In the Omniglot experiments, the BPL test protocol was used for one-shot classification and the Variant test protocol for five-shot classification and fifteen-shot classification. The selection of training data followed Lake's [46] suggestion. We also constructed two tiny training datasets based on Omniglot background small 1. One was based on the Latin alphabet with 26 characters and another was based on the Korean alphabet with 40 characters. All experiments used the standard BPL test protocol.

In the CASIA-HWDB1.1 experiments, the training characters were selected from the first 3155 characters; for example, the training data of the experiment "h1.1" in Table 3 selected the first 500 characters and drawn by the first 5 calligraphers.

Table 3. Results of small data experiments on Omniglot and HWDB with 20-way accuracy (%). ID with ′ means that the network has 86 layers, and ID with * means that the network has 46 layers; otherwise, the network has 26 layers. ID with + means that the image sizes are 28 × 28 and the images are augmented by 90, 180, and 270 degrees rotated and resized. ID with - means that the image sizes are 28 × 28 and the images are not augmented by rotating; otherwise, the image sizes are 64 × 64. The bold lines indicate that the model achieved the best results on different data sets in this set of experiments.

ID	Training Dataset	Classes	Samples	Shots	Average Accuracy	Maximum Accuracy	Minimum Accuracy	Variance Accuracy	Amount
o1.1	omniglot_Latin	26	20	1	66.86	67.50	66.00	0.15	520
o1.2	omniglot_Latin	26	20	5	85.14	88.20	82.20	2.04	520
o1.3	omniglot_Latin	26	20	15	89.49	91.94	85.83	3.23	520
o1.4 *	omniglot_small2	156	5	1	90.51	91.00	90.00	0.08	780
o1.5 *	omniglot_small2	156	5	5	97.75	99.04	96.15	0.59	780
o1.6′ +	**omniglot_tiny**	**60**	**20**	**1**	**97.99**	**98.25**	**97.25**	**0.06**	**1200**
o1.7′ +	omniglot_small1	136	20	1	98.65	99.50	98.00	0.06	2720
o1.8′ +	**omniglot_small2**	**156**	**20**	**1**	**99.17**	**99.50**	**98.75**	**0.04**	**3120**
o1.9′ +	omniglot	964	20	1	98.92	99.50	98.50	0.04	19,280
h1.1 -	HWDB	500	5	1	79.42	82.00	76.36	2.63	2500
h1.2 -	HWDB	200	20	1	95.02	96.18	93.82	0.40	4000

By analyzing the experimental results, we can get the following observations.

4.2.1. Classification Accuracies

The models trained on the small data showed high few-shot classification accuracies. Table 3 shows the model's few-shot classification accuracies. Using 964 characters for training, LCNN achieved 98.92% (see experiment o1.9) one-shot classification accuracy and it outperforms the state-of-the-art accuracy of 96.7% [79] established by BPL and it is also better than the human identification accuracy of 95.5% [46]. Using Omniglot background small 1 and Omniglot background small 2 for training, LCNN, respectively, achieved accuracies of 98.65% and 99.17% (see experiments o1.7 and o1.8) that outperformed BPL, which, respectively, achieved 95.7%, 96.0% accuracy [79]. Similarly, on CASIA-HWDB1.1, the training on only 4000 samples (200 classes and 20 samples per class) output 95.02% one-shot classification accuracy. About 4000 samples can train a model with 26 layers and around 488k parameters and it does not fall into overfitting.

In order to further demonstrate the strength capability on small data, we designed a few experiments on very tiny training datasets. On the one-shot classification task on Omniglot, an LCNN with 86 layers and 1.94 million parameters, was trained on "background tiny" with only 60 classes and 20 samples per class, achieving an accuracy of 97.99% (see experiment o1.6). We only fed seven hundred and eighty samples (156 classes and five samples per class) into an LCNN with 46 layers and 975k parameters. The LCNN showed 90.51% one-shot classification accuracy, and 97.75% five-shot classification accuracy (see experiments o1.4, o1.5). We only fed an alphabet with 520 samples (26 classes and 20 samples per class) into the LCNN, and we obtained 66.86% one-shot accuracy, 85.14% five-shot accuracy, and 89.49% fifteen-shot accuracy (see experiments o1.1, o1.2 and o1.3).

4.2.2. Training Sample Amount

The number of training classes and the number of samples per class jointly determine the classification accuracy. Generally, the increase in the amount of training samples results in the improvement of the classification accuracy. However, the balance between the number of classes and the sample number per class is a crucial factor in the classification performance. We conducted the relevant experiments, which are presented in Table 4.

Table 4. Results of small data experiments on Omniglot and HWDB with 20-way accuracy (%). The network has 26 layers. ID with - means that the image sizes are 28 × 28 and the images are not augmented by rotating; otherwise, the image sizes are 64 × 64.

ID	Training Dataset	Classes	Samples	Shots	Average Accuracy	Maximum Accuracy	Minimum Accuracy	Variance Accuracy	Amount
o2.1	omniglot_Latin	26	20	1	66.86	67.50	66.00	0.15	520
o2.2	omniglot_Korean	40	20	1	52.39	53.50	51.50	0.26	800
o2.3 -	omniglot	964	2	1	83.42	85.67	80.83	1.55	1928
o2.4 -	omniglot	964	3	1	91.43	93.83	89.50	1.60	2892
o2.5 -	omniglot	964	5	1	95.22	96.17	93.33	0.69	4820
o2.6 -	omniglot	964	15	1	96.30	97.50	94.17	0.71	14,460
o2.7 -	omniglot	964	20	1	95.38	96.83	94.50	0.24	19,280
h2.1	HWDB	20	240	1	81.30	86.00	78.73	3.05	4800
h2.2	HWDB	200	20	1	91.74	93.64	88.91	1.85	4000
h2.3	HWDB	3155	2	1	62.25	66.55	57.45	4.08	6310
h2.4	HWDB	3155	5	1	94.75	96.18	92.18	0.87	15,775
h2.5	HWDB	3155	20	1	96.88	98.18	95.09	0.69	63,100
h2.6	HWDB	3155	240	1	97.82	98.05	97.45	0.02	757,200

As the class number is fixed, the classification accuracy increases with the sample number per class. However, when the sample number per class is too low, the accuracy sharply goes down; for example, on Omniglot, the accuracy of experiment o2.3 is about 12%

less than that of experiment o2.5. On HWDB, due to it being more diverse than Omniglot, the decrease is more severe; it is about 33% (see experiments h2.3 and h2.4). On the contrary, if the class number is too low, the large sample number cannot result in high accuracy; for instance, in experiment o2.2 on Omniglot, the result of 40 classes and 20 samples per class is lowered to 52.39%. For another example, in experiment h2.1 on HWDB, the result of 20 classes and 240 samples per class is 81.30%; however, in experiment h2.2, the result of 200 classes and 20 samples per class is rises to 91.74%.

Comparing the experiments o2.6 and o2.7 on Omniglot, with the increase of sample number per class and the amount, the classification accuracy declines. The reason for the decline is that more training data need more training steps; however, the steps of o2.6 and o2.7 are the same.

More training data (more classes and more samples per class) can make the one-shot classification more steady and robust. Comparing experiments o2.4 and o2.7, although the accuracy does not largely improve with the sample increase, the accuracy variance of o2.7 sharply comes down relative to o2.4, from 1.60% to 0.24%. There are similar tendencies in the HWDB; for example, from experiment h2.5 to h2.6, the accuracy variance ranges from 0.69% to 0.02%.

The diversity of the training data can improve the accuracy. Comparing the experiments o2.2 and o2.1 on Table 4, the accuracy with 40 classes in o2.2 is less than the accuracy with 20 classes in o2.1. The reason is that the letters in the Korean alphabet are more similar to each other in shape than the Latin alphabet. The Korean letters are mainly composed of vertical lines, horizontal lines, and points, so the shape variances are few.

4.3. Transfer Capability Test

The function of the LCL is to identify similar objects from a few contrasting objects instead of remembering the objects themselves. In other words, the LCL should learn to distinguish the similar objects and dissimilar objects, but it should not learn to remember the objects. Therefore, the discrimination capability trained on a dataset might be transferred to another dataset for distinguishing the contrast objects. In order to evaluate the transfer learning capability, we designed a few experiments that evaluated the classification accuracies on one dataset with the model trained on another dataset. For instance, we evaluated the classification accuracy on Omniglot with the model trained on HWDB.

Firstly, we designed the experiments about transferring learning from HWDB to Omniglot. The experiment results are shown in Table 5. In experiments a1 and a5, we directly evaluate the classification accuracies using the model trained on HWDB without the additional training. We obtain the average one-shot accuracy of 87.69% (using BPL one-shot evaluation standard [46]). The five-shot accuracy is high at 96.08%, which is greater than the human one-shot accuracy of 95.5% [46].

In order to demonstrate that the LCNN can continuously learn new knowledge from a new dataset, we trained new models using Omniglot based on the model that has been trained on HWDB. We have used three datasets to train three new models: Omniglot background small 1 with five samples per class, Omniglot background small 2 with five samples per class, and Omniglot background with twenty samples per class; the corresponding results are shown in Table 5. Figure 8 shows the classification accuracy comparison between cases with transfer learning and without transfer learning. On Omniglot small 1 and Omniglot small 2, the one-shot accuracies separately improve from 81.73% to 93.91%, and 84.41% to 94.71%, and the accuracy increases rise to 12.18% and 10.30%. With the help of the knowledge transferred from HWDB, the one-shot classification accuracy is high at 94.71%, which is near the human level of 95.5%, using only 780 training samples. However, on a training Omniglot background set with 964 classes, the accuracy did not improve (see experiments d1 and d5). The reason is that the transfer of the LCNN is only effective for small training sets, and it can alleviate the lack of training samples by transferring knowledge from other datasets.

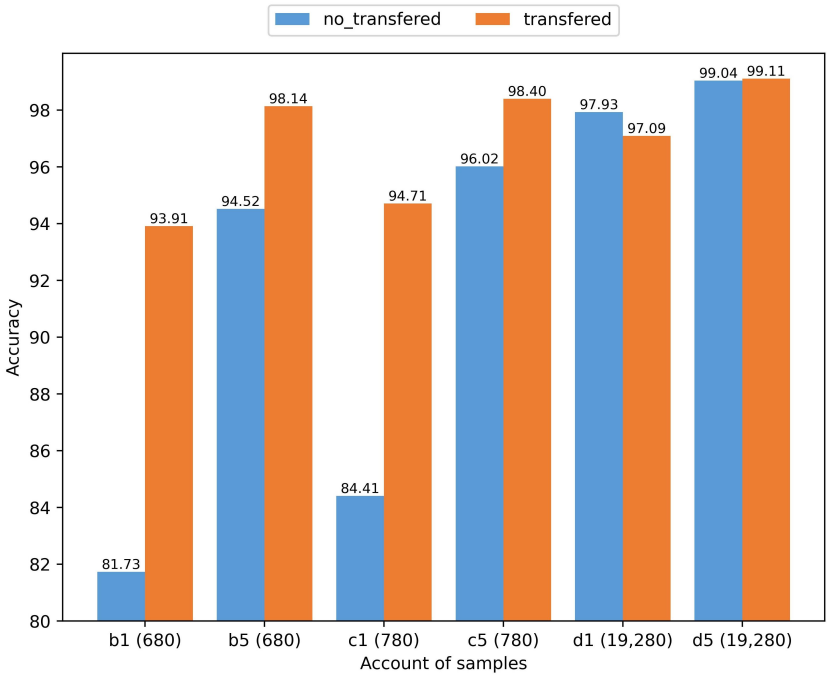


Figure 8. Transfer from HWDB to Omniglot.

Table 5. Results of all experiments on Omniglot and HWDB with 20-way accuracy (%). In all experiments, the image size is 64 × 64. The experiments b1, b5, c1, c5, d1, and d5 have a constant learning rate of 0.001 and max learning steps of about 5000, and the corresponding non-transferred accuracies are given by the models with hyperparameters d_1 9600, d_2 12,800, and m 16,000. The "avg_acc" means average accuracy. The "no_trans_avg_acc" means no transferred average accuracy. The "var_accuracy" means variance average.

ID	Source	Target	Test	Target Classes	Target Samples	Test Shots	avg_ acc	no_trans_ avg_acc	Max Accuracy	Min Accuracy	var_ Accuracy
a1	HWDB		omniglot			1	87.69		88.75	86.50	0.39
a5	HWDB		omniglot			5	96.08		97.20	94.40	0.67
b1	HWDB	omniglot small1	omniglot	136	5	1	93.91	81.73	94.75	92.75	0.21
b5	HWDB	omniglot small1	omniglot	136	5	5	98.14	94.52	99.00	96.40	0.43
c1	HWDB	omniglot small2	omniglot	156	5	1	94.71	84.41	95.25	94.25	0.08
c5	HWDB	omniglot small2	omniglot	156	5	5	98.40	96.02	99.40	97.20	0.36
d1	HWDB	omniglot	omniglot	964	20	1	97.09	97.93	97.75	96.00	0.16
d5	HWDB	omniglot	omniglot	964	20	5	99.11	99.04	99.80	98.40	0.13

5. Discussion

The LCL method is based on the idea of using a sample as the center to establish a local space to contrast and identify other samples. Contrasting samples within a local context is a key factor for the success of the LCNN. Contrasting in local context forces the LCNN to adapt to different local contexts; thus, the LCNN cannot remember the context

information; so, in the case of tiny samples, the LCNN would not encounter overfitting. The other success of the LCL is partially attributed to the extremely large quantity of Local Cognitive Contexts (LCCs) and the three randomness factors: randomly selecting classes, randomly selecting samples, and randomly shuffling contrastive objects. They all enable the LCNN to learn to distinguish objects instead of representing and remembering the training object or its pattern. Of course, a super deep neural network, such as Resnet, is a very important factor, and its depth may provide enough flexibility to adapt to a large local context. However, the LCL method ensures that the super deep neural network avoids overfitting in the case of tiny samples. Therefore, the super deep neural network contributes to the high classification accuracy, but LCL contributes to its success in training. It is a fine design to use the Difference Perceptron to contrast the difference embedding instead of learning a similarity metric. That makes the contrast objects in the LCC become a list rather than a set and makes the order of the contrast objects become meaningful. The order is crucial to creating tremendously different LCCs.

The number of training classes and the number of samples per class are an essential factor to improve performance. More classes and samples per class can improve the test accuracy. However, the accuracy is only increased when the sample is smaller. In case the training sample is enough, the increase of sample size almost cannot improve the classification accuracy, which is similar to human beings' cognitive behavior, in that humans can learn to recognize objects from a few samples, but humans cannot learn more from redundant similar samples. In some cases, the increase of samples might worsen the accuracy, because more samples may bring in more noise, which can interfere with the training. For instance, in Table 1, the accuracy of 98.97% of 1200 characters is slightly less than the accuracy of 99.24% of 156 characters, because, in Omniglot, there are some similar characters in different alphabets, which is noise to the LCNN.

Our method generates a large number of LCCs, which can result in slower training speeds due to the significant amount of data involved. Furthermore, while our method performs well in one-shot learning scenarios, its application in other domains requires further exploration. For instance, applying the LCL method to areas such as speech recognition and natural language processing may present new challenges and opportunities that warrant investigation.

6. Conclusions

We proposed a novel deep learning approach named Local Contrast Learning to alleviate deep model overfitting resulting from a lack of training samples. In a local space centered around an object, the LCL enables the deep model to adapt to large local contexts and carefully capture the difference between contrastive objects and recognized objects. The LCL is able to stably and successfully train deep neural networks with about 100 layers using dozens of sample classes and tens of samples in each class.

We have conducted numerous experiments, and our deep model based on the LCL method achieved significant improvements on the Omniglot dataset. In the five-way one-shot task, the model achieved an accuracy of 99.24% when considering 156 classes, surpassing the accuracy of 98.97% achieved when considering 1200 classes. Additionally, when using only 60 classes, the model achieved a high accuracy of 98.95% in the five-way one-shot task, which is comparable to the accuracy of 98.97% achieved with 1200 classes. This result demonstrates the capability of the LCL method to enable deep models to be trained on very small datasets.

Author Contributions: Y.Z.: conceptualization, methodology, supervision, writing—original draft, and writing—review and editing. X.Y.: conceptualization, writing—original draft, methodology, investigation, formal analysis, and data curation. L.L.: data curation and writing—review and editing. Y.Y.: formal analysis and investigation. S.Z.: data curation, visualization, and investigation. C.X.: Project administration, conceptualization, writing—review and editing, supervision, funding acquisition, resources. All authors have read and agreed to the published version of the manuscript.

Funding: The project was funded by the Natural Science Foundation of Chongqing (Grant No. cstc2014jcyjA40034) for research on a service quality measurement model and its application based on preference ontology and the National Undergraduate Innovation and Entrepreneurship Training Program of China (202310637006).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The experiments were conducted on publicly open datasets. The datasets can be accessed in their corresponding published papers. The code was publicly available on GitHub at https://github.com/shepherd0/LCNN (accessed on 29 May 2024).

Acknowledgments: The work and writing of this thesis have received strong support and assistance from Xuanpeng Zhang of Soyea Technology Co., Ltd., Hangzhou, China.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

LCL	Local Contrast Learning
LCNN	Local Contrast Neural Network
LCC	Local Cognitive Context
MNNW	Multi-column Neural Network Shared Weights
CPLA	Contrastive Pair Local Activation

References

1. Li, X.; Yang, X.; Ma, Z.; Xue, J.H. Deep metric learning for few-shot image classification: A review of recent developments. *Pattern Recognit.* **2023**, *138*, 109381. [CrossRef]
2. He, K.; Pu, N.; Lao, M.; Lew, M.S. Few-shot and meta-learning methods for image understanding: A survey. *Int. J. Multimed. Inf. Retr.* **2023**, *12*, 14. [CrossRef]
3. Liu, L.; Zhou, T.; Long, G.; Jiang, J.; Yao, L.; Zhang, C. Prototype propagation networks (PPN) for weakly-supervised few-shot learning on category graph. *arXiv* **2019**, arXiv:1905.04042.
4. Fei-Fei, L.; Fergus, R.; Perona, P. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *28*, 594–611. [CrossRef]
5. Wu, J.; Zhao, Z.; Sun, C.; Yan, R.; Chen, X. Few-shot transfer learning for intelligent fault diagnosis of machine. *Measurement* **2020**, *166*, 108202. [CrossRef]
6. Hou, R.; Chang, H.; Ma, B.; Shan, S.; Chen, X. Cross attention network for few-shot classification. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 1–12.
7. Chen, H.; Li, H.; Li, Y.; Chen, C. Multi-level metric learning for few-shot image recognition. In Proceedings of the International Conference on Artificial Neural Networks, Bristol, UK, 6–9 September 2022; pp. 243–254.
8. Kim, H.H.; Woo, D.; Oh, S.J.; Cha, J.W.; Han, Y.S. Alp: Data augmentation using lexicalized pcfgs for few-shot text classification. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 22 February–1 March 2022; Volume 36, pp. 10894–10902.
9. Chen, Z.; Fu, Y.; Wang, Y.X.; Ma, L.; Liu, W.; Hebert, M. Image deformation meta-networks for one-shot learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 8680–8689.
10. Yoon, S.W.; Seo, J.; Moon, J. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 7115–7123.
11. Wu, Z.; Zhao, H. Hierarchical few-shot learning based on coarse-and fine-grained relation network. *Artif. Intell. Rev.* **2023**, *56*, 2011–2030. [CrossRef]
12. Yu, C.; Liu, J.; Nemati, S.; Yin, G. Reinforcement learning in healthcare: A survey. *ACM Comput. Surv. (CSUR)* **2021**, *55*, 1–36. [CrossRef]
13. Wang, T.; Chen, Y.; Qiao, M.; Snoussi, H. A fast and robust convolutional neural network-based defect detection model in product quality control. *Int. J. Adv. Manuf. Technol.* **2018**, *94*, 3465–3471. [CrossRef]
14. Suh, S.; Cheon, S.; Choi, W.; Chung, Y.W.; Cho, W.K.; Paik, J.S.; Kim, S.E.; Chang, D.J.; Lee, Y.O. Supervised segmentation with domain adaptation for small sampled orbital CT images. *J. Comput. Des. Eng.* **2022**, *9*, 783–792. [CrossRef]
15. He, K.; Zhang, X.; Ren, S.; Sun, J. Identity mappings in deep residual networks. In Proceedings of the Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Proceedings, Part IV 14; Springer: Berlin/Heidelberg, Germany, 2016; pp. 630–645.
16. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on cOmputer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.

17. Jadon, S.; Jadon, A. An overview of deep learning architectures in few-shot learning domain. *arXiv* **2020**, arXiv:2008.06365.
18. Tai, Y.; Tan, Y.; Xiong, S.; Sun, Z.; Tian, J. Few-shot transfer learning for SAR image classification without extra SAR samples. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2022**, *15*, 2240–2253. [CrossRef]
19. Yazdanpanah, M.; Rahman, A.A.; Chaudhary, M.; Desrosiers, C.; Havaei, M.; Belilovsky, E.; Kahou, S.E. Revisiting learnable affines for batch norm in few-shot transfer learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 9109–9118.
20. Sun, N.; Yang, P. T2L: Trans-transfer Learning for few-shot fine-grained visual categorization with extended adaptation. *Knowl.-Based Syst.* **2023**, *264*, 110329. [CrossRef]
21. Song, X.; Gao, W.; Yang, Y.; Choromanski, K.; Pacchiano, A.; Tang, Y. Es-maml: Simple hessian-free meta learning. *arXiv* **2019**, arXiv:1910.01215.
22. Baik, S.; Hong, S.; Lee, K.M. Learning to forget for meta-learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 2379–2387.
23. Baik, S.; Choi, J.; Kim, H.; Cho, D.; Min, J.; Lee, K.M. Meta-learning with task-adaptive loss function for few-shot learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 9465–9474.
24. Qin, Z.; Wang, H.; Mawuli, C.B.; Han, W.; Zhang, R.; Yang, Q.; Shao, J. Multi-instance attention network for few-shot learning. *Inf. Sci.* **2022**, *611*, 464–475. [CrossRef]
25. Xie, J.; Long, F.; Lv, J.; Wang, Q.; Li, P. Joint distribution matters: Deep brownian distance covariance for few-shot classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 7972–7981.
26. Afrasiyabi, A.; Larochelle, H.; Lalonde, J.F.; Gagné, C. Matching feature sets for few-shot image classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 9014–9024.
27. Li, K.; Zhang, Y.; Li, K.; Fu, Y. Adversarial feature hallucination networks for few-shot learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 13470–13479.
28. Guan, J.; Lu, Z.; Xiang, T.; Li, A.; Zhao, A.; Wen, J.R. Zero and few shot learning with semantic feature synthesis and competitive learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *43*, 2510–2523. [CrossRef] [PubMed]
29. Osahor, U.; Nasrabadi, N.M. Ortho-shot: Low displacement rank regularization with data augmentation for few-shot learning. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 3–8 January 2022; pp. 2200–2209.
30. Ciregan, D.; Meier, U.; Schmidhuber, J. Multi-column deep neural networks for image classification. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 3642–3649.
31. Chopra, S.; Hadsell, R.; LeCun, Y. Learning a similarity metric discriminatively, with application to face verification. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–26 June 2005; Volume 1, pp. 539–546.
32. Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; Shah, R. Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 1993; Volume 6.
33. Chen, X.; He, K. Exploring simple siamese representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 15750–15758.
34. Grm, K.; Dobrisek, S.; Struc, V. Deep pair-wise similarity learning for face recognition. In Proceedings of the 2016 4th International Conference on Biometrics and Forensics (IWBF), Limassol, Cyprus, 3–4 March 2016; pp. 1–6.
35. Hoffer, E.; Ailon, N. Deep metric learning using triplet network. In Proceedings of the Similarity-Based Pattern Recognition: Third International Workshop, SIMBAD 2015, Copenhagen, Denmark, 12–14 October 2015; Proceedings 3; Springer: Berlin/Heidelberg, Germany, 2015; pp. 84–92.
36. Min, W.; Mei, S.; Li, Z.; Jiang, S. A two-stage triplet network training framework for image retrieval. *IEEE Trans. Multimed.* **2020**, *22*, 3128–3138. [CrossRef]
37. Wei, J.; Huang, C.; Vosoughi, S.; Cheng, Y.; Xu, S. Few-shot text classification with triplet networks, data augmentation, and curriculum learning. *arXiv* **2021**, arXiv:2103.07552.
38. Wang, R.; Wu, X.J.; Chen, Z.; Hu, C.; Kittler, J. Spd manifold deep metric learning for image set classification. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**. [CrossRef] [PubMed]
39. Liao, S.; Shao, L. Graph sampling based deep metric learning for generalizable person re-identification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 7359–7368.
40. Ermolov, A.; Mirvakhabova, L.; Khrulkov, V.; Sebe, N.; Oseledets, I. Hyperbolic vision transformers: Combining improvements in metric learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 7409–7419.
41. Assran, M.; Caron, M.; Misra, I.; Bojanowski, P.; Bordes, F.; Vincent, P.; Joulin, A.; Rabbat, M.; Ballas, N. Masked siamese networks for label-efficient learning. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; pp. 456–473.
42. Cao, R.; Zhang, Q.; Zhu, J.; Li, Q.; Li, Q.; Liu, B.; Qiu, G. Enhancing remote sensing image retrieval using a triplet deep metric learning network. *Int. J. Remote Sens.* **2020**, *41*, 740–751. [CrossRef]

43. Pang, Y.; Zhao, X.; Xiang, T.Z.; Zhang, L.; Lu, H. Zoom in and out: A mixed-scale triplet network for camouflaged object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 2160–2170.
44. Liang, B.; Wang, Z.; Huang, B.; Zou, Q.; Wang, Q.; Liang, J. Depth map guided triplet network for deepfake face detection. *Neural Netw.* **2023**, *159*, 34–42. [CrossRef]
45. Miller, G.A. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychol. Rev.* **1956**, *63*, 81. [CrossRef] [PubMed]
46. Lake, B.M.; Salakhutdinov, R.; Tenenbaum, J.B. Human-level concept learning through probabilistic program induction. *Science* **2015**, *350*, 1332–1338. [CrossRef]
47. Kaya, M.; Bilge, H.Ş. Deep metric learning: A survey. *Symmetry* **2019**, *11*, 1066. [CrossRef]
48. Cen, J.; Yun, P.; Cai, J.; Wang, M.Y.; Liu, M. Deep metric learning for open world semantic segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 15333–15342.
49. Koch, G.; Zemel, R.; Salakhutdinov, R. Siamese neural networks for one-shot image recognition. In Proceedings of the ICML Deep Learning Workshop, Lille, France, 6–11 July 2015; Volume 2.
50. Kumar BG, V.; Carneiro, G.; Reid, I. Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 5385–5394.
51. Ma, Y.; Bai, S.; An, S.; Liu, W.; Liu, A.; Zhen, X.; Liu, X. Transductive Relation-Propagation Network for Few-shot Learning. In Proceedings of the IJCAI, Yokohama, Japan, 11–17 July 2020; Volume 20, pp. 804–810.
52. Zhu, W.; Li, W.; Liao, H.; Luo, J. Temperature network for few-shot learning with distribution-aware large-margin metric. *Pattern Recognit.* **2021**, *112*, 107797. [CrossRef]
53. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation applied to handwritten zip code recognition. *Neural Comput.* **1989**, *1*, 541–551. [CrossRef]
54. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*. [CrossRef]
55. Khan, A.; Sohail, A.; Zahoora, U.; Qureshi, A.S. A survey of the recent architectures of deep convolutional neural networks. *Artif. Intell. Rev.* **2020**, *53*, 5455–5516. [CrossRef]
56. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, PMLR, Lille, France, 6–11 July 2015; pp. 448–456.
57. Lin, M.; Chen, Q.; Yan, S. Network in network. *arXiv* **2013**, arXiv:1312.4400.
58. Hadsell, R.; Chopra, S.; LeCun, Y. Dimensionality reduction by learning an invariant mapping. In Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), New York, NY, USA, 17–22 June 2006; Volume 2, pp. 1735–1742.
59. Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; Krishnan, D. Supervised contrastive learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 18661–18673.
60. Chen, T.; Luo, C.; Li, L. Intriguing properties of contrastive losses. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 11834–11845.
61. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1026–1034.
62. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
63. Sutskever, I.; Martens, J.; Dahl, G.; Hinton, G. On the importance of initialization and momentum in deep learning. In Proceedings of the International Conference on Machine Learning, PMLR, Atlanta, GA, USA, 16–21 June 2013; pp. 1139–1147.
64. Liu, Y.; Gao, Y.; Yin, W. An improved analysis of stochastic gradient descent with momentum. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 18261–18271.
65. Yuan, W.; Hu, F.; Lu, L. A new non-adaptive optimization method: Stochastic gradient descent with momentum and difference. *Appl. Intell.* **2022**, *52*, 3939–3953. [CrossRef]
66. Liu, C.L.; Yin, F.; Wang, D.H.; Wang, Q.F. CASIA online and offline Chinese handwriting databases. In Proceedings of the 2011 International Conference on Document Analysis and Recognition, Beijing, China, 18–21 September 2011; pp. 37–41.
67. Liu, C.L.; Yin, F.; Wang, D.H.; Wang, Q.F. Online and offline handwritten Chinese character recognition: benchmarking on new databases. *Pattern Recognit.* **2013**, *46*, 155–162. [CrossRef]
68. Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D. Matching networks for one shot learning. *arXiv* **2016**, arXiv:1606.04080.
69. Santoro, A.; Bartunov, S.; Botvinick, M.; Wierstra, D.; Lillicrap, T. Meta-learning with memory-augmented neural networks. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 20–22 June 2016; pp. 1842–1850.
70. Finn, C.; Abbeel, P.; Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the International Conference on Machine Learning, PMLR, Boston, MA, USA, 24–25 October 2017; pp. 1126–1135.
71. Munkhdalai, T.; Yu, H. Meta networks. In Proceedings of the International Conference on Machine Learning, PMLR, Boston, MA, USA, 24–25 October 2017; pp. 2554–2563.

72. Garnelo, M.; Rosenbaum, D.; Maddison, C.; Ramalho, T.; Saxton, D.; Shanahan, M.; Teh, Y.W.; Rezende, D.; Eslami, S.A. Conditional neural processes. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 1704–1713.
73. Jamal, M.A.; Qi, G.J. Task agnostic meta-learning for few-shot learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 11719–11727.
74. Huang, M.; Xu, Y.; Bao, W.; Xiang, X. Training few-shot classification via the perspective of minibatch and pretraining. In Proceedings of the CAAI International Conference on Artificial Intelligence, Hangzhou, China, 5–6 June 2021; pp. 650–661.
75. Cho, W.; Kim, E. Improving Augmentation Efficiency for Few-Shot Learning. *IEEE Access* **2022**, *10*, 17697–17706. [CrossRef]
76. Zheng, W.; Tian, X.; Yang, B.; Liu, S.; Ding, Y.; Tian, J.; Yin, L. A few shot classification methods based on multiscale relational networks. *Appl. Sci.* **2022**, *12*, 4059. [CrossRef]
77. Sung, F.; Yang, Y.; Zhang, L.; Xiang, T.; Torr, P.H.; Hospedales, T.M. Learning to compare: Relation network for few-shot learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 1199–1208.
78. Meng, X.; Wang, X.; Yin, S.; Li, H. Few-shot image classification algorithm based on attention mechanism and weight fusion. *J. Eng. Appl. Sci.* **2023**, *70*, 14. [CrossRef]
79. Lake, B.M.; Salakhutdinov, R.R.; Tenenbaum, J. One-shot learning by inverting a compositional causal process. *Adv. Neural Inf. Process. Syst.* **2013**, *26*.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.