electronics

MDPI

Brief Report

Contrastive Learning via Local Activity

He Zhu [1,2,*], Yang Chen [1,*], Guyue Hu [1] and Shan Yu [1,2]

[1] Brainnetome Center, National Laboratory of Pattern Recognition (NLPR), China Academy of Sciences Institute of Automation (CASIA), Beijing 100190, China
[2] School of Future Technology, University of Chinese Academy of Sciences (UCAS), Beijing 101408, China
* Correspondence: he.zhu@nlpr.ia.ac.cn (H.Z.); yang.chen@ia.ac.cn (Y.C.)

Abstract: Contrastive learning (CL) helps deep networks discriminate between positive and negative pairs in learning. As a powerful unsupervised pretraining method, CL has greatly reduced the performance gap with supervised training. However, current CL approaches mainly rely on sophisticated augmentations, a large number of negative pairs and chained gradient calculations, which are complex to use. To address these issues, in this paper, we propose the local activity contrast (LAC) algorithm, which is an unsupervised method based on two forward passes and locally defined loss to learn meaningful representations. The learning target of each layer is to minimize the activation value difference between two forward passes, effectively overcoming the limitations of applying CL above mentioned. We demonstrated that LAC could be a very useful pretraining method using reconstruction as the pretext task. Moreover, through pretraining with LAC, the networks exhibited competitive performance in various downstream tasks compared with other unsupervised learning methods.

Keywords: unsupervised; representation learning; non-backpropagation

Citation: Zhu, H.; Chen, Y.; Hu, G.; Yu, S. Contrastive Learning via Local Activity. *Electronics* **2023**, *12*, 147. https://doi.org/10.3390/electronics12010147

Academic Editor: Tomasz Trzcinski

Received: 7 November 2022
Revised: 24 December 2022
Accepted: 25 December 2022
Published: 29 December 2022

1. Introduction

Unsupervised visual representation learning methods aim to learn better visual representations from a large number of images without human intervention. In the past, supervised pretraining was the mainstream approach for various computer vision tasks. However, with the explosive growth in the number of images, learning to perform visual tasks requires increasingly higher manual labeling costs. In recent years, contrastive learning methods [1–3] have been explored as the potential key for next-generation visual task frameworks [4–6], which maximize the similarity between image enhancements. However, to implement CL is still a complex process. On one hand, CL needs various forms of data augmentation and needs to sort many negative training instances. On the other hand, the conventional training algorithm for CL needs sophisticated computation for chained gradient calculation and propagation. The motivation of the current work is, therefore, to find a method for unsupervised learning that can avoid (1) data augmentation, (2) negative training instances, and (3) chained gradient calculation and propagation, while maintaining competitive performance in downstream tasks.

Thus, in this paper, we propose a simple and effective way to achieve unsupervised visual representation learning, rather than relying on various auxiliary modules. We first propose a novel local activity contrastive (LAC) algorithm for training an autoencoder (AE). Furthermore, based on LAC, we propose an unsupervised pretraining framework for deep networks to learn better visual representations. Overall, this research contributes the following:

- We propose a new autoencoder training algorithm, named the local activity contrast (LAC). The LAC algorithm can train the AE/CAE without using gradient backpropagation (BP); each layer of the network has its locally defined loss function.

- We demonstrate that the LAC algorithm is a useful unsupervised pretraining method, and does not require using negative pairs, momentum encoders, or complex augmentation treatment. Our experiments demonstrate that deep convolutional networks pretrained with the LAC show improved classification/detection performance on standard datasets (e.g., MNIST, CIFAR-10, CIFAR-100, Tiny-ImageNet, ImageNet, and COCO) after fine-tuning.
- We show that the LAC algorithm can effectively learn representations based on the image reconstruction pretext task. In contrast, gradient back-propagation could lead to over-fitting the pretext task and limiting the quality of learned representation.

2. Related Work

Contrastive Learning. Many recent studies on unsupervised pretraining [1–3,7–9] have focused on contrastive learning [10], which is a framework for learning similar/dissimilar representations from data organized in pairs. If the inputs are data-enhanced versions of the same image, the pair is considered positive; otherwise, the pair is considered negative. Networks trained by contrastive learning will learn similar representations from positive pairs and discriminative representations from negative pairs. Some studies [1–3,11,12] attract the positive (similar) pairs and repulses negative (different) pairs to learn the representations. Other studies [13,14] also provide a self-distillation framework, which only matches positive samples to learn the representations. Moreover, Zbontar et al. [15] propose to minimize the redundancy between the vector components of positive pairs' vectors, which can achieve good representation learning. However, the CL still heavily relies on data augmentations, chained gradient calculation and propagation. To address these issues, here we propose the LAC algorithm that can learn to give similar responses to a pair composed of an original image and its reconstruction, which can be applied without using data augmentation, negative pairs or gradient back-propagation.

Non-backpropagation (Non-BP) Algorithms. Non-BP algorithms are approaches aimed to avoid the complex chained gradient calculation and propagation of training. Non-BP algorithms have drawn considerable attention in recent years. In [16,17], target-propagation is further introduced to backpropagate some predefined target values rather than gradients. Similar ideas are explored in [18], with a different definition of targets for each layer. Unlike the above algorithms with errors backpropagating in an indirect and top-down manner, some authors have proposed methods to send errors from the output layers directly to the hidden layers. In [19,20], the errors are propagated through fixed random feedback connections from the output layer to each hidden layer. In [21], the Hilbert–Schmidt independence criteria for the input, hidden, and output layers are proposed to update the corresponding weights, based on the information bottleneck theory [22,23].

Contrastive/Recirculation Training Algorithm. The LAC algorithm uses the recirculated forward signals to calculate the contrastive loss, which is similar to the previously suggested training algorithms for the restricted Boltzmann machine (RBM) [24,25]. Specifically, Hinton et al. [26] propose to train RBM by reducing the reconstruction error of visible and hidden units through a closed loop. Then, Hinton further proposes to train the RBM by minimizing the contrastive divergence [27], with the training process carried out layer by layer. However, symmetrical feedforward and feedback weights between two adjacent layers are necessary for RBM, and BP-based fine-tuning is still required in training to obtain good performance. The weight symmetry problem has been tackled in [28], in which the authors stated that training through the error backpropagated by random feedback connections can achieve competitive performance compared with the traditional BP. These two methods [27,28] achieve contrastive/recirculation learning based on short inner loops between adjacent layers in the network; however, the LAC method we propose depends on the long feedback loop in the hidden layer of a network, which therefore could be applied to more network structures and allow all layers of the system to be trained simultaneously. We note that in parallel with our work, Hinton [29] proposes the

forward–forward (FF) algorithm, which trains networks based on the idea of two forward passes similar to the LAC.

3. Method

In this section, we introduce the LAC algorithm in the context of training an autoencoder (AE) with a reconstruction task; then, we extend its usage to other network structures and tasks, treating the LAC method as a more general approach to learn the proper representation of an image.

For a well-trained AE, its output is an almost perfect copy of its input. Therefore, for each neuron in the network, the response to the original image should be close to the response to the reconstructed image. The LAC algorithm is inspired by these inherent characteristics of an AE, and training is achieved by learning the activity contrast, i.e., the difference in activities between two forward passes of the original images (OI) and reconstructed images (RI). Specifically, for each neuron, the loss function is defined by the square of the activity contrast, and the key point of the LAC method is that the responses in the second pass should learn to approach those in the first pass, rather than the opposite. Therefore, only the output of the second pass should be used to calculate the gradient.

Specifically, the network is trained using this algorithm by first calculating the activity contrast based on the responses of the individual neurons in two forward passes: the first one uses the original image as the input of the network, and the second one uses the reconstructed image. The activity contrast of each neuron is then defined as the local loss to guide the updating of the weights using the stochastic gradient descent (SGD) approach (Figure 1). Intuitively, each neuron updating its own weight according to the locally defined loss may lead to the difficulty in reducing the global reconstruction error. Surprisingly, we found that the LAC algorithm can successfully train a network to correctly reproduce the inputs only through local information.

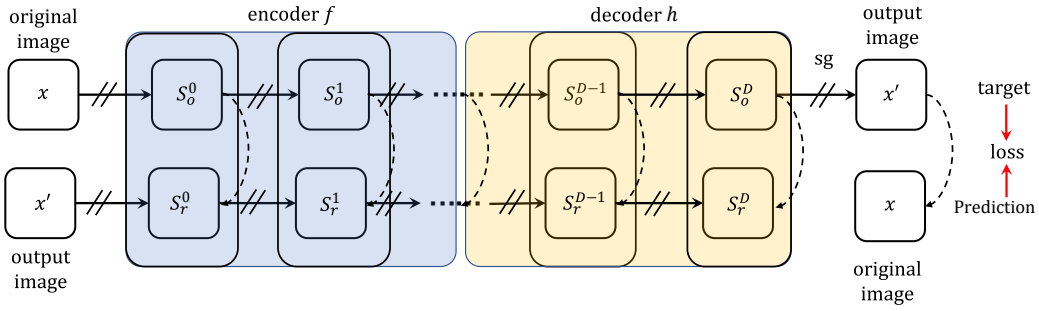


Figure 1. The local activity contrast (LAC) algorithm trains an autoencoder by reducing the local activity contrast of individual neurons in two forward passes of the original and reconstructed images. Specifically, in the first forward pass with the original image input, we record the activation value of each layer as the "target", and we use the activation value of each layer in the second forward pass with the reconstructed image input (coming from the first forward pass) as the "prediction". Each layer has its own independently defined LAC losses with which the gradients are calculated. Note that the gradient does not flow across layers (stop gradient: sg).

The algorithm is illustrated in the pseudocode shown in Algorithm 1. Specifically, we consider the j-th neuron in the i-th layer in the AE, for which the LAC loss of a specific neuron can be calculated as

$$\mathcal{L}_{ij} = (\mathcal{S}_o^{ij} - \mathcal{S}_r^{ij})^2 \tag{1}$$

where $\mathcal{S}_o^{ij}$ denotes the reaction of the neuron corresponding to OI and $\mathcal{S}_r^{ij}$ to RI, respectively. The reactions are defined as

$$\mathcal{S}_o^{ij} = BN_{\gamma,\beta}(\mathcal{A}_o^{i-1}\mathcal{W}^{i-1,j}) \tag{2}$$

$$\mathcal{A}_o^{i} = \mathcal{F}([S_o^{i1}, \cdots, S_o^{ij}, \cdots]) \tag{3}$$

$$\mathcal{S}_r^{ij} = BN_{\gamma,\beta}(\mathcal{A}_r^{i-1}\mathcal{W}^{i-1,j}) \tag{4}$$

$$\mathcal{A}_r^{i} = \mathcal{F}([S_r^{i1}, \cdots, S_r^{ij}, \cdots]) \tag{5}$$

where $BN_{\gamma,\beta}(\cdot)$ denotes the operation of batch normalization, and $F(\cdot)$ is the activation function. Specifically, in our method, the output is connected to the input layer. For unifying our expression, we define:

$$\mathcal{S}_o^0 = \mathcal{O}I \tag{6}$$

$$\mathcal{S}_r^0 = \mathcal{R}I \tag{7}$$

and the same loss as defined in Equation (1) can be calculated.

Algorithm 1 Pseudocode of LAC in a Pytorch-like style.

```
# model: the networks output is :
# h_i,h_j: concatenation of all the hidden outputs
# o_i,o_j: outputs of the output layer
# h_opt: the optimizer of the hidden layers
# o_opt: the optimizer of the output layer
# detach() : stop the gradient flow into the other layers

for x in dataloader: #load a minibatch-size data x
h_i, o_i = model(x) # target activation
h_j, o_j = model(o_i) # prediction activation

# Compute hidden layers losses
h_loss = torch.sum((h_i.detach()-h_j)**2)

# Compute output layer loss
o_loss = torch.sum((o_i-x)**2)

# update: hidden layers
h_opt.zero_grad()
h_loss.backward(retain_graph=True)
h_opt.step()

# update: output layer
o_opt.zero_grad()
o_loss.backward()
o_opt.step()
```

Unlike other training methods, in LAC, we send the reconstructed image back to the network as input and then re-perform the feedforward calculation. The activity contrast calculations are then localized to each neuron in the network, and the parameters are updated independently through the gradient descent algorithm, without the need to explicitly propagate the gradient to other neurons, thus avoiding the chained gradient calculations in the traditional BP. According to Equation (1), the weights of the *j*th neuron in the *i*th layer should be updated to

$$\begin{aligned} \mathcal{W}_{t+1}^{ij} &= \mathcal{W}_{t}^{ij} - \eta \frac{\partial \mathcal{L}_{ij}}{\partial \mathcal{W}_{t}^{ij}} \\ &= \mathcal{W}_{t}^{ij} - \eta'(\mathcal{S}_{o}^{ij} - \mathcal{S}_{r}^{ij})\dot{\mathcal{S}}_{o}^{ij} A_{o}^{i-1} \end{aligned} \tag{8}$$

For LAC, the parameter update depends on the activity contrast between the two feedforward runs. The neurons in different layers thus carry out weight modification based on local information. Surprisingly, we found that this apparently uncoordinated update carried out for individual neurons can effectively reduce the global reconstruction loss of the network.

In addition to using the LAC algorithm to train AEs with reconstruction tasks, we also explored the possibility of using this algorithm as a more general method for training networks in representation learning.

4. Experiments

4.1. Implementation Details

In the AE experiments with the network consisting of fully connected layers, the number of units in the hidden layers was [1000, 500, 250, 500, 1000], and the activation function was ReLU/Sigmoid (hidden/last layer), with batch normalization. During training, we used the RMSprop/Adam (hidden/reconstruction) optimizer with an initialized learning rate of 0.001/0.01 (hidden/reconstruction) and a weight decay of 0.001. The batch size was 128.

In the AE experiments with the network consisting of convolutional (Conv) layers, the kernel and stride for the Conv/Deconv layers were set to [(3 × 3, 1), (3 × 3, 1), (3 × 3, 2)/(3 × 3, 2)], and the activation function was Re-LU/Sigmoid (hidden/last layer), with batch normalization. During training, we used the RMSprop/Adam (hidden/reconstruction) optimizer with an initialized learning rate of $1 \times 10^{-6}/1 \times 10^{-4}$ (hidden/reconstruction). The batch size was 16.

In the convolutional neural network (CNN) pretraining experiments, we changed the kernel size of the first convolutional layer from 7 × 7 to 3 × 3 and removed the first maximum pooling layer in the original ResNet18 dataset to obtain a stronger classification performance baseline, named ResNet18-like CNN.

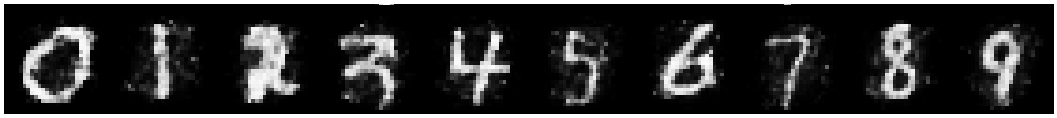
During pretraining, we used the RMSprop/Adam/SGD (hidden/reconstruction/regularization) optimizer with an initialized learning rate of $5 \times 10^{-7}/1 \times 10^{-5}/3 \times 10^{-5}$. By replacing the linear classifier with a deconvolution module, pretraining could be performed on the network with image reconstruction tasks.

In the downstream classification task, all deconvolutional layers were removed, and a linear classifier was added to the pretrained network. We used the SGD optimizer with an initialized learning rate of 0.01/0.1 (AE/ResNet) and adopted a warm-up learning rate strategy in the first two epochs, with a step decay of [60, 120, 160], gamma of 0.2 for fine-tuning, and weight decay of 5×10^{-4}. At this stage, the batch size was 128. Backpropagation was used to train the last linear layer or fine-tune the entire network.

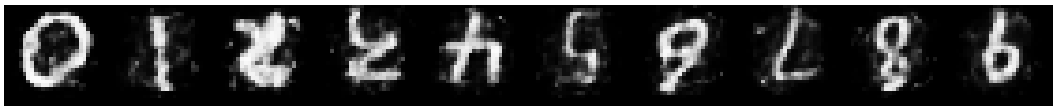
4.2. LAC for Training an Autoencoder

LAC for Fully Connected AE Normal AEs consist of fully connected layers, which are the basic structure for image reconstruction through encoding and decoding. We first explored using the LAC method to train a plain AE consisting of a three-layer encoder and a three-layer decoder. The results of the MNIST dataset are shown in Figure 2.

(**a**) Original Images



(**b**) Reconstruction



(**c**) Flipped Reconstruction

Figure 2. Examples of the reconstruction results on MNIST with an AE trained by the LAC.

LAC for Convolutional AE (CAE)

For more complex datasets with larger input sizes or RGB color channels, CNNs are the most popular architecture. Thus, we next verified the efficacy of LAC in training CNNs. To this end, we applied LAC to a CAE with a three-layer convolutional module as the encoder and a one-layer deconvolutional module as the decoder. We performed experiments on the ImageNet dataset. We found that LAC worked well in this task, with examples of the reconstructed images shown in Figure 3. Taken together, we verified the success of the LAC in training AEs under various conditions, including different network architectures, image sizes, and datasets.

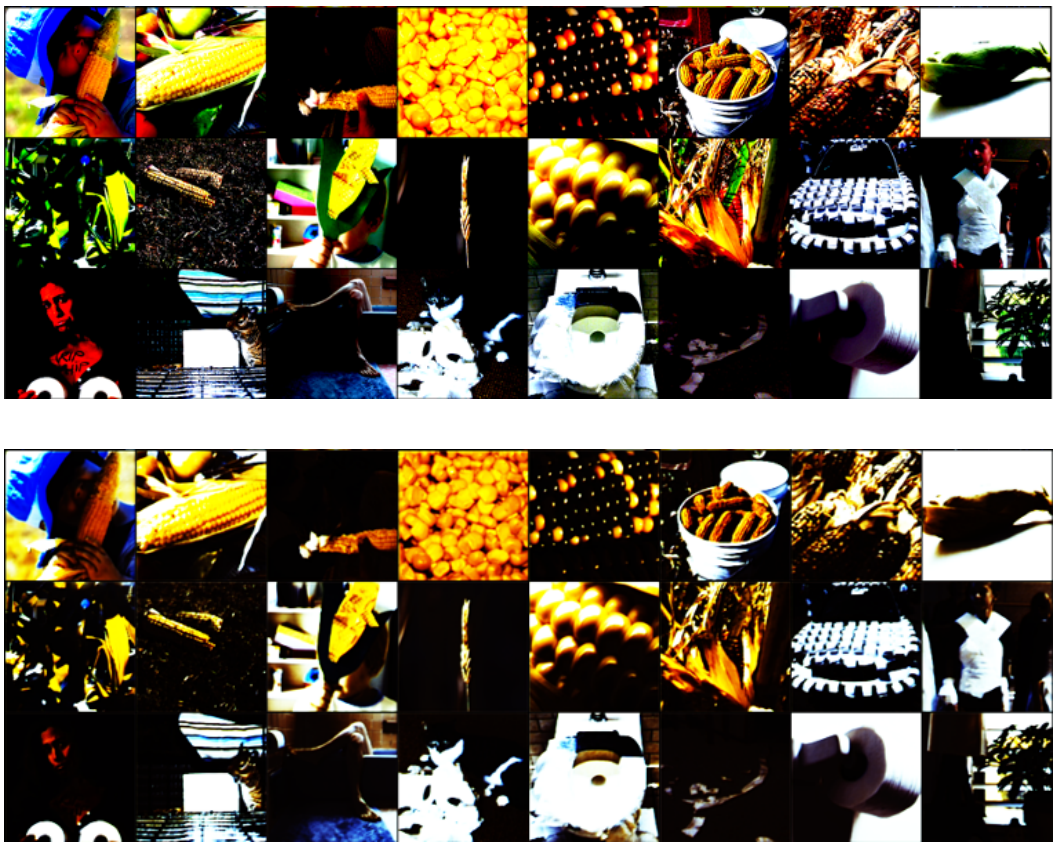


Figure 3. Examples of reconstructed images in the ImageNet dataset. The original images (256 × 256) were randomly clipped to 224 × 224. The top group contains original clipped images, and the bottom group contains the outputs of CAE trained by LAC.

4.3. LAC for Unsupervised Pretraining

4.3.1. MNIST Classification

Table 1 shows the results of applying different pretrained encoders to the classification task. Here we use the top-1 accuracy as the metric, which is the same as in [2].

Specifically, in the "Linear evaluation" experiment, the pretrained encoder was fixed, and only an additional linear classifier (a fully connected layer) was trained to classify the image category based on the output of the encoder. In the "fine-tuning" experiment, all the parameters, including those in both the pretrained encoder and the linear classifier, can be trained with the supervising signals for classification. "No pretraining" refers to the condition with a randomly network as the encoder combined with a fully connected layer as the linear classifier.

The accuracy obtained by the LAC method in the linear evaluation experiment was 88.3%, which is much better than the no pretraining condition (46.8%) and not so far away from the result achieved with the BP-pretrained encoder (93.8%). Noted that the key advantage of unsupervised learning methods, in comparison to supervised learning, is that they can be applied to very large datasets without manual labeling to learn useful feature representations, which has been very important for the recent development in various fields including computer vision, natural language processing, etc. Thus, the key value of the LAC algorithm is not that it can outperform the supervised learning with random initialized encoder.

Importantly, in Table 1, we show that LAC can achieve better results than BP in the fine-tuning condition. Previous papers [30,31] suggest that the image reconstruction pretext task might be detrimental to learn a more generally applicable feature representation and perform worse than training from scratch without any pretext task, which was consistent with BP-pretraining results in our experiments. However, fine-tuning the LAC-pretrained encoder leads to improvements in the downstream classification task. Thus, the reason that the image reconstruction is unsuitable as a pretext task to learn useful representation may come from the BP training method rather than the task itself. These results suggest another important advantage of the LAC method compared to the conventional BP, besides avoiding complex chained gradient calculations.

From the perspective of the hidden layer representation learning, the LAC method has a similar mechanism to that in [2], as LAC can maximize the hidden layer activity consistency between the two feedforward passes via the MSE loss in latent space but without using BP. Importantly, the inputs of these two passes do not need complex augmentation, as was required in [2,3]. Instead, they simply use the reconstruction as one augmented view of the input. During the process of training an AE with the LAC algorithm, the reconstructed image may be far from a perfect copy of the original image at the early stages. Nevertheless, the LAC algorithm can guide the network to evolve toward learning useful representations of the input image.

To further examine the potential use of LAC in representation learning, we applied it to non-reconstruction tasks. The flipped image reconstruction experiment using MNIST was a simple test to check whether the framework could be used when the inputs of the first and second feedforward passes were different. This experiment was the same as the normal AE experiment, except that the output layer was forced to produce a flipped version of the original image. Figure 2c illustrates that the LAC algorithm could train the AE to complete the flipping task.

Table 1. Top-1 accuracy for classification of MNIST dataset after linear evaluation and fine-tuning. After pretraining AE to reconstruct MNIST images, the decoder is replaced by a fully connected layer to classify digits from 0 to 9. The best results are in bold.

Pretraining Method	Linear Evaluation	Fine-Tuning
No pretraining	46.75 ± 0.05	96.08 ± 0.06
BP pretraining	93.75 ± 0.06	94.10 ± 0.06
LAC pretraining	88.25 ± 0.08	**96.18 ± 0.06**

4.3.2. CIFAR-10/CIFAR-100 Classification

Next, we examine if the LAC algorithm can be applied in more sophisticated networks with more complex datasets. To this end, we trained ResNet [32] using LAC to learn the representation of images in CIFAR and ImageNet datasets. The CIFAR-10 dataset consists of 60,000 images categorized into 10 classes. In total, it has 6000 images per class, including 50,000 for training and 10,000 for testing. The CIFAR-100 is similar to the CIFAR-10, except that it has 100 classes, with 500 training images and 100 testing images per class. The pretraining/fine-tuning experiments were based on the CIFAR-10/100 training datasets, respectively, and the classification accuracy metrics were evaluated, accordingly, on the CIFAR-10/100 test datasets. In the pretraining phase, the last linear classification layer in the original ResNet was replaced by a deconvolutional layer to reproduce the reconstructed image. On the CIFAR 10 and 100 datasets, we found that the LAC algorithm was also effective in training the network for the reconstruction task. After fine-tuning the LAC-pretrained encoder and the classifier, the system exhibited significant performance improvements in the downstream classification task, as shown in Table 2. Table 2 also shows the experimental results of the LAC pretraining of ResNet, which achieved improved classification performance over both supervised end-to-end training by a sizable margin (1.0%) and some other CL pretraining methods using a large number of negative pairs.

Table 2. Top-1 accuracy (%) for the classification task of CIFAR-10/100 test datasets and Tiny ImageNet validation dataset after fine-tuning. "No pretraining" means supervised training from scratch. The best results are in bold.

Pretraining Method	CIFAR-10	CIFAR-100	Tiny ImageNet
No pretraining	95.38 ± 0.2	75.61 ± 0.2	62.86 ± 0.3
MOCO [1]	95.40 ± 0.1	76.02± 0.1	63.38± 0.2
SimCLR [2]	94.00 ± 0.1	76.50 ± 0.1	63.50± 0.2
LAC	**95.72 ± 0.1**	**76.59 ± 0.1**	**63.78 ± 0.2**

4.3.3. Tiny ImageNet Classification

The Tiny ImageNet (http://cs231n.stanford.edu/tiny-imagenet-200, accessed on 1 January 2022.) dataset is a subset of ImageNet, which contains 100,000 images for training and 10,000 images for validation across 200 classes. The pretraining/fine-tuning experiments were based on the training dataset, and the classification accuracy was calculated on the validation dataset. Table 2 shows that, similar to the results achieved in the CIFAR datasets, ResNet pretrained by the LAC algorithm achieved better classification performance on the Tiny ImageNet dataset compared to the baseline.

4.3.4. COCO Detection and Segmentation

We also use LAC to pretrain the ResNet-50 on the ImageNet dataset and evaluate its performance on more challenging downstream task, such as detection and segmentation. Following previous research [1], we use Mask R-CNN [33] with a C4 backbone, with batch normalization tuned and synchronized across GPUs. Table 3 shows the object detection

and semantic segmentation results for the COCO dataset [34], which has achieved better performance compared to MoCo v2 [3]. The results suggest that LAC-based pretraining has functional advantages across various downstream visual tasks.

Table 3. Instance segmentation and object detection results on COCO with the ×1 training schedule and a C4 backbone. * denotes reproduced results. AP^{mk} means mask average precision. AP^{bb} means the bounding-box average precision (AP^{bb}). AP^{*} is the average over 10 IoU levels on 80 categories (start from 0.5 to 0.95 with a step size of 0.05). AP^{*}_{50} means the AP with IoU = 0.50. AP^{*}_{75} means the AP with IoU = 0.75. The best results are in bold.

Method	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}
MoCo v2 * [3]	34.2	55.4	36.2	39.0	58.6	41.9
MoCHi * [35]	34.4	55.6	36.7	39.2	58.8	42.4
LAC	**34.6**	**56.0**	**36.9**	**39.6**	**59.1**	**42.8**

5. Conclusions

In this paper, we presented an algorithm that enables the network to learn through local activity contrast (LAC), which can be applied as an unsupervised, non-BP training method for AEs. Importantly, LAC can be used as an effective pretraining method using reconstruction as the pretext task. With subsequent fine-tuning, the networks pretrained by LAC exhibited a significant advantage in various downstream visual tasks compared with BP pretraining or end-to-end training approaches.

Author Contributions: Conceptualization, H.Z., Y.C. and S.Y.; methodology, H.Z. and Y.C.; software, H.Z.; validation, H.Z., Y.C. and G.H.; formal analysis, H.Z., Y.C. and G.H.; writing—original draft preparation, H.Z.; writing—review and editing, Y.C., G.H. and S.Y.; visualization, H.Z.; supervision, S.Y.; project administration, Y.C. and S.Y.; funding acquisition, Y.C. and S.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Key Research and Development Program of China under Grant 2021ZD0200402, the International Partnership Program of CAS under Grant 173211KYSB20200021, the Strategic Priority Research Program of the Chinese Academy of Sciences (CAS) under Grant XDB32040200, CAS Project for Young Scientists in Basic Research under Grant No. YSBR-041, and Young Scientists Fund of the National Natural Science Foundation of China under Grand 1190051195.

Data Availability Statement: Data available in a publicly accessible repository that does not issue DOIs. (1) MNIST can be found here: http://yann.lecun.com/exdb/mnist/. (2) CIFAR-10/100 can be found here: https://www.cs.toronto.edu/~kriz/cifar.html. (3) Tiny-ImageNet can be found here: http://cs231n.stanford.edu/tiny-imagenet-200. (4) ImageNet can be found here: https://www.image-net.org/. (5) COCO can be found here: https://cocodataset.org/. These dataset are accessed on 1 January 2022.

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

References

1. He, K.; Fan, H.; Wu, Y.; Xie, S.; Girshick, R. Momentum Contrast for Unsupervised Visual Representation Learning. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020.
2. Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G.E. A Simple Framework for Contrastive Learning of Visual Representations. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, Virtual Event, 13–18 July 2020; Volume 119, pp. 1597–1607.
3. Chen, X.; Fan, H.; Girshick, R.B.; He, K. Improved Baselines with Momentum Contrastive Learning. *arXiv* **2020**, arXiv:2003.04297.
4. Zhu, J.; Liu, S.; Yu, S.; Song, Y. An Extra-Contrast Affinity Network for Facial Expression Recognition in the Wild. *Electronics* **2022**, *11*, 2288 . [CrossRef]

5. Zhao, D.; Yang, J.; Liu, H.; Huang, K. Specific Emitter Identification Model Based on Improved BYOL Self-Supervised Learning. *Electronics* **2022**, *11*, 3485 [CrossRef]
6. Liu, B.; Yu, H.; Du, J.; Wu, Y.; Li, Y.; Zhu, Z.; Wang, Z. Specific Emitter Identification Based on Self-Supervised Contrast Learning. *Electronics* **2022**, *11*, 2907. [CrossRef]
7. Wu, Z.; Xiong, Y.; Yu, S.X.; Lin, D. Unsupervised feature learning via non-parametric instance discrimination. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018.
8. Hjelm, R.D.; Fedorov, A.; Lavoie-Marchildon, S.; Grewal, K.; Bachman, P.; Trischler, A.; Bengio, Y. Learning deep representations by mutual information estimation and maximization. *arXiv* **2018**, arXiv:1808.06670.
9. Ye, M.; Zhang, X.; Yuen, P.C.; Chang, S.F. Unsupervised embedding learning via invariant and spreading instance feature. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019.
10. 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.
11. Chen, T.; Kornblith, S.; Swersky, K.; Norouzi, M.; Hinton, G.E. Big Self-Supervised Models are Strong Semi-Supervised Learners. In Proceedings of the NeurIPS 2020, Virtual, 6–12 December 2020.
12. Chen, X.; Xie, S.; He, K. An Empirical Study of Training Self-Supervised Vision Transformers. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, 10–17 October 2021; pp. 9620–9629.
13. Chen, X.; He, K. Exploring Simple Siamese Representation Learning. In Proceedings of the CVPR 2021, Virtual, 19–25 June 2021; pp. 15750–15758.
14. Caron, M.; Touvron, H.; Misra, I.; Jégou, H.; Mairal, J.; Bojanowski, P.; Joulin, A. Emerging properties in self-supervised vision transformers. In Proceedings of the ICCV, Montreal, QC, Canada, 10–17 October 2021; pp. 9650–9660.
15. Zbontar, J.; Jing, L.; Misra, I.; LeCun, Y.; Deny, S. Barlow Twins: Self-Supervised Learning via Redundancy Reduction. In Proceedings of the ICML 2021, Virtual, 18–24 July 2021; Meila, M., Zhang, T., Eds.; Volume 139, pp. 12310–12320.
16. Lee, D.; Zhang, S.; Fischer, A.; Bengio, Y. Difference Target Propagation. In Proceedings of the Machine Learning and Knowledge Discovery in Databases, Porto, Portugal, 7–11 September 2015.
17. Bengio, Y.; Lee, D.; Bornschein, J.; Lin, Z. Towards Biologically Plausible Deep Learning. *arXiv* **2015**, arXiv:1502.04156v3.
18. Choromanska, A.; Cowen, B.; Kumaravel, S.; Luss, R.; Rigotti, M.; Rish, I.; Kingsbury, B.; DiAchille, P.; Gurev, V.; Tejwani, R.; et al. Beyond Backprop: Online Alternating Minimization with Auxiliary Variables. *arXiv* **2018**, arXiv:1806.09077.
19. Widrow, B.; Greenblatt, A.; Kim, Y.; Park, D. The No-Prop algorithm: A new learning algorithm for multilayer neural networks. *Neural Netw.* **2013**, *37*, 182–188. [CrossRef] [PubMed]
20. Nøkland, A.; Eidnes, L.H. Training neural networks with local error signals. *arXiv* **2019**, arXiv:1901.06656.
21. Ma, W.K.; Lewis, J.P.; Kleijn, W.B. The HSIC Bottleneck: Deep Learning without Back-Propagation. *arXiv* **2019**, arXiv:1908.01580.
22. Tishby, N.; Pereira, F.; Bialek, W. The information bottleneck method. *arXiv* **2000**, arXiv:physics/0004057.
23. Shwartzziv, R.; Tishby, N. Opening the Black Box of Deep Neural Networks via Information. *arXiv* **2017**, arXiv:1703.00810.
24. Hinton, G.E.; Osindero, S.; Teh, Y.W. A fast learning algorithm for deep belief nets. *Neural Comput.* **2006**, *18*, 1527–1554. [CrossRef] [PubMed]
25. Sacramento, J.; Costa, R.P.; Bengio, Y.; Senn, W. Dendritic cortical microcircuits approximate the backpropagation algorithm. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 8721–8732.
26. Hinton, G.E.; Mcclelland, J.L. Learning Representations by Recirculation. In *Neural Information Processing Systems 0 (NIPS 1987)*; American Institute of Physics: College Park, MD, USA, 1988; pp. 358–366.
27. Hinton, G.E. Training products of experts by minimizing contrastive divergence. *Neural Comput.* **2002**, *14*, 1771–1800. [CrossRef] [PubMed]
28. Lillicrap, T.P.; Cownden, D.; Tweed, D.B.; Akerman, C.J. Random feedback weights support learning in deep neural networks. *arXiv* **2014**, arXiv:1411.0247.
29. Hinton, G. The Forward-Forward Algorithm: Some Preliminary Investigations. Avaliable online: https://www.cs.toronto.edu/~hinton/FFA13.pdf (accessed on 20 December 2022).
30. He, K.; Chen, X.; Xie, S.; Li, Y.; Dollár, P.; Girshick, R. Masked autoencoders are scalable vision learners. *arXiv* **2021**, arXiv:2111.06377.
31. Xie, Z.; Zhang, Z.; Cao, Y.; Lin, Y.; Bao, J.; Yao, Z.; Dai, Q.; Hu, H. Simmim: A simple framework for masked image modeling. *arXiv* **2021**, arXiv:2111.09886.
32. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
33. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.

34. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014.
35. Kalantidis, Y.; Sariyildiz, M.B.; Pion, N.; Weinzaepfel, P.; Larlus, D. Hard Negative Mixing for Contrastive Learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 21798–21809.