Multi-layer PCA Network for Image Classification

Mubarakah Alotaibi(✉) and Richard C. Wilson(✉)

University of York, York, UK
{mmma512,richard.wilson}@york.ac.uk

Abstract. PCANet is a simple deep learning baseline for image classification, which learns the filters banks by PCA instead of stochastic gradient descent (SGD) in each layer. It shows a good performance for image classification tasks with only a few parameters and no backpropagation procedure. However, PCANet suffers from two main problems. The first problem is the features explosion which limits its depth to two layers. The second issue is the binarization process which leads to discriminative information loss. To handle these problems, we adopted CNN-like convolution layers to learn the PCA filter-bank and reduce the number of dimensions. We also used second-order pooling with z-score normalization to replace the histogram descriptor. The late fusion method is used to combine the class posteriors generated each layer. The proposed network has been tested on image classification tasks including MNIST, Cifar10, Cifar100 and Tiny ImageNet databases. The experimental results show that our model achieves better performance than standard PCANet and is competitive with some CNN methods.

Keywords: PCANet · PCANet+ · PCANet II · CNN · Spatial pyramid pooling · Second-order pooling · Fusion Neural Network

1 Introduction

Convolutional neural networks (CNNs) have witnessed immense success in image classification since Alexnet won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 [19]. Some of the famous CNNs include VGGNet [23], ResNet [24] and in general, architectures have become complicated with more layers. For example, one common realization of ResNet has 152 layers. The filters learning process in CNNs relies on optimization via gradient descent using backpropagation, making the whole process unexplainable and computationally expensive, particularly with more layers.

As a response to this, PCANet [2] was proposed in 2015 as a simple baseline for image classification problems. PCANet is trained using a closed-form non-iterative and unsupervised procedure and is an order of magnitude faster to train than a traditional CNN. Although the performance is not state-of-the-art, it is remarkably effective for such a simple architecture. PCANet has led to a family

A. Torsello et al. (Eds.): S+SSPR 2020, LNCS 12644, pp. 292–301, 2021.
https://doi.org/10.1007/978-3-030-73973-7_28

of related techniques, and the network proposed in this paper is broadly part of this family. While PCANet achieved good performance in various benchmarks, it suffers from some problems:

- The network is shallow, potentially causing a loss of performance on more complex datasets.
- Due to the way PCANet convolves the images and uses histogram-pooling, there is an explosion in the number of features with more layers or filters.
- Feature binarization restricts the filter responses to only 1's, and 0's, leading to discriminative-information loss.
- Since filter learning is unsupervised, later layers may not preserve important classification information from earlier layers.

In this paper, we propose a new network with some innovations. Firstly, we generate feature maps from all network layers and use *late fusion* to combine class posteriors. This introduces an element of supervised learning while preserving the single-pass strategy of PCANet. We are then able to preserve essential features from early network layers. Secondly, to improve the amount of discriminative information, we use *second-order pooling* which has recently become popular in CNNs and is used in PCANet-II [11]. We refined the approach by using z-score normalization to provide more stable and informative features. Finally, we adopt multi-channel convolution layers typically used in CNNs (but not in PCANet) and PCANet+ [1]. This helps to reduce the size of the feature maps. The performance of this new network is good and comparable to old NNs-based architectures but not the recent architectures.

2 Related Work

PCANet [2] is a simple feedforward network which does not require backpropagation to learn the filter-bank. Instead, they adopted the PCA eigenvectors learnt from stacking patches of images to be their candidate filters. After several convolutional layers, the features are encoded employing a binary hashing followed by block-wise histograms. In fact, this network generates (unsupervised) features, and the classification is left to a final stage. The filters were applied on a one-channel basis, i.e. each filter operates on one channel and produces one new grayscale image. As a result, the number of channels at the next level is channels $\times$ filters and rises quickly. The histogram pooling also generates many features, leading to a feature explosion if more layers are added.

The design of PCANet has led to other related variants. For example, DCT-Net [20], LBP-Net [21] and ICANet [22] changed the type of filters used by PCANet while pursuing with the same structure. PCANet-II [11] also used the same PCANet architecture but replaced the histogram-pooling with the second-order pooling to reduce the number of features. PCANet+ [1] tried to change the PCANet filters' topology by proposing PCA filter ensemble learning. Other research such as [23] and [31] attempted to learn a non-linear representation of the PCA filters using kernel methods.

3 Multi-layer PCA Network Structure

Assume we have a set of N training samples $X^{(0)}$ that feed into the network, where $X^{(0)} = \{\{X_i\}_1^N : X_i \in \mathbb{R}^{m \times n \times d}\}$ and $d \in \{1, 3\}$ represents the grayscale and the coloured images respectively. Figure 1 describes the architecture of the network. The first layers are convolution layers, where the input to each convolution layer ($X^{(L-1)}$) is the previous layer's output. The following equation defines the output of each convolution layer (defined in the next section):

$$X^L = W^L * X^{(L-1)} \tag{1}$$

Where X^L represents the output of the layer (L), W^L is the PCA-based filters learned for the current layer (L), and ($*$) is the convolution operator.

For every convolution layer's output, we extract the features using the second-order pooling and then, optionally, apply multi-level spatial pyramid pooling (SPP) as in [2]. Finally, we run a classifier for each layer which outputs the class posteriors and send the fused posteriors to a classifier for final prediction. Details about the network structure are discussed in the following sections.

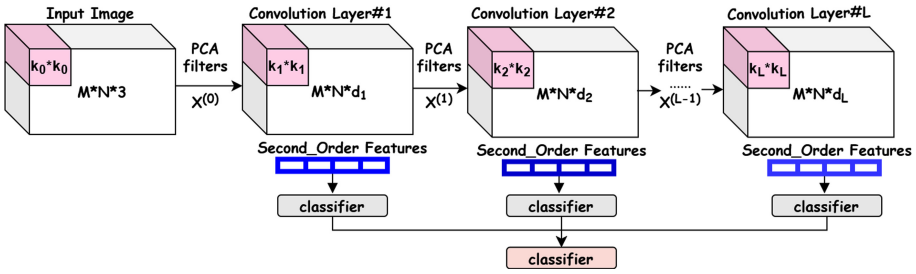

Fig. 1. Multi-layer PCA network architecture

3.1 PCA Convolution Layers

The PCA filters are calculated as in [1]. For N samples $X^{L-1} = \{X_i\}_1^N$, where $X_i \in \mathbb{R}^{m \times n \times d_{L-1}}$, and d_{L-1} is the number of filters in the layer (L-1), or $d_{L-1} \in \{1, 3\}$ for the input layer, we find the PCA filters as the following:

1. Extract all overlapping patches of size $k_L \times k_L \times d_{L-1}$ in each image X_i and subtract the mean-patch, where k_L represents the filter's size in layer (L).
2. Calculate $\bar{X}_i \in \mathbb{R}^{k_L^2 d_{L-1} \times N \widetilde{m} \widetilde{n}}$ by concatenating all the vectorized zero-mean patches from all sample images, where $\widetilde{m} = (m - k_L) + 1$ and $\widetilde{n} = (n - k_L) + 1$, m and n are the width and the height of the image.
3. Solve the following equation to find d_L principle components of ($\bar{X}^{L-1}$ $\bar{X}^{L-1^T}$).

$$\min_{V \in R^{(k_L \times k_L) \times d_{L-1}}} ||\bar{X}^{L-1} - VV^T \bar{X}^{L-1}||_F^2, \; V^T V = I_{d_L - 1} \tag{2}$$

Where $I_{d_{L-1}}$ is the identity matrix of size $d_{L-1} \times d_{L-1}$.

4. The PCA filters can be expressed as the following:

$$W_s^L = \underset{k_L \times k_L \times d_{L-1}}{\text{mat}} q_s, \; s = 1, 2, \ldots, d_L. \tag{3}$$

Where $\underset{k_L \times k_L \times d_{L-1}}{\text{mat}}(v)$ is the function that maps the vector $v \in \mathbb{R}^{k_L^2 d_{L-1}}$ to tensor $W \in \mathbb{R}^{k_L \times k_L \times d_{L-1}}$, q_s is the s^{th} principal eigenvector of $\bar{X}^{L-1}\bar{X}^{L-1^T}$ and d_L is the number of filters chosen for the layer (L).

5. The output of each convolution layer is obtained by convolving each filter with the sample images:

$$X_i^L = \bar{X}_i^{L-1} * W_s^L \in \mathbb{R}^{m \times n \times d_L} \tag{4}$$

where $s = 1, 2, \ldots, d_L$ and $\bar{X}_i^{L-1}$ is zero-padded to get the same image size.

3.2 Second-Order Features

Figure 2 illustrates the mechanism of calculating second-order features from each convolution layer's output. So, for each output $X_i^L \in \mathbb{R}^{m \times n \times d_L}$, where $i = [1, 2, \ldots, N]$, N is the number of samples, and d_L is the number of responses in layer L. We first divide the tensors into patches of the same size and normalize each patch using z-score normalization. Then, we find the channel-wise covariance matrix for each patch, representing the second-order features related to that position in the image (determined by the patch size). Details about covariance calculation have been discussed in the following subsection.

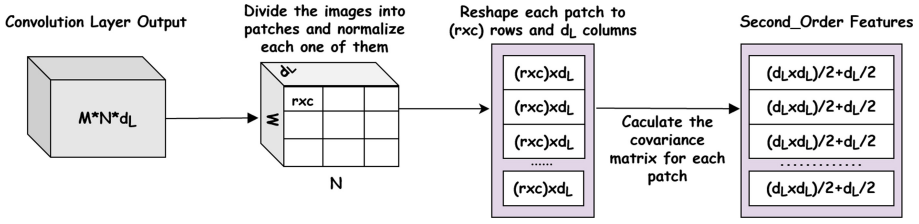

Fig. 2. Second order features for every convolution output

Covariance Computation. Assume $X \in \mathbb{R}^{M \times d}$ is sampled from the normal distribution, where M is the number of instances, and d is the number of dimensions. The following equation defines the covariance matrix of X.

$$\Sigma = \frac{1}{M} \sum_{k=1}^{M} (x_k - \mu)^T (x_k - \mu) \tag{5}$$

Where μ is the sample mean.

The covariance matrix Σ is a symmetric positive definitive matrix that forms a Riemannian manifold [5]. Most of the classifiers, such as SVM, deals with Euclidean data so direct use of covariance matrix entries for features is not ideal. We should first map Σ to the Euclidean tangent space using log matrix $\log \Sigma$ [5], or square-root matrix $\Sigma^{1/2}$ [3,4,6]. Empirically, the square-root matrix seems to produce better performance than logarithmic matrix [3].

The covariance matrix is difficult to estimate robustly when the number of samples is small compared to the number of dimensions [4,7], and so the square-root covariance cannot be accurately calculated directly. Therefore, we used the following equation as in [7] to estimate the covariance matrix.

$$\widetilde{\Sigma} = U \mathrm{diag}(\delta_i : i = 1, 2, \ldots, d) U^T \ , \ \delta_i = \sqrt{\left(\frac{1-\alpha}{2\alpha}\right)^2 + \frac{\lambda_i}{2\alpha}} - \frac{1-\alpha}{2\alpha} \tag{6}$$

Where U is the orthogonal matrix consisting of the eigenvectors, $(\lambda_i, i = 1 \ldots d)$ are the eigenvalues in decreasing order, α is regularizing parameter and set to be $\frac{1}{2}$ in all experiments as in [4]. This gives $\widetilde{\Sigma}$ as a regularized estimate of $\Sigma^{1/2}$. After computing the estimated covariance matrix, we combine the mean and the estimated covariance using the following positive-definitive matrix as in [7].

$$\mathcal{N}(\mu, \widetilde{\Sigma}) \sim \begin{pmatrix} \widetilde{\Sigma} + \mu\mu^T & \mu \\ \mu^T & 1 \end{pmatrix} \tag{7}$$

Where $\widetilde{\Sigma}$ and μ are the estimated covariance and the sample mean, respectively. Because this matrix is symmetric, the number of features for each convolution layer is $(d_L + 1)(d_L + 2)/2 - 1\times$ the number of patches.

3.3 Late Fusion

The intermediate activations' outputs could provide informative clues about the images, including local parts, boundaries, and low-level textures. Therefore, integrating information from all layers is essential for better performance and reliable prediction [8]. Generally, there are two standard fusion methods, namely early fusion and late fusion [9,10]. The difference between the two methods is explained in Fig. 3. The early fusion works in the features level, where we fuse the features first using one of the methods discussed in Table 1 and send the fused features to a classifier to predict the classes' labels. On the other hand, the late fusion works by running a classifier each level, combine the posteriors using one of the methods described in Table 1 and send them to the primary classifier for the final prediction. The researchers [9,10] showed that the late fusion method could provide comparable or better performance than the early fusion. We used the late fusion method in our experiments, where the class posteriors were averaged to predict the final results. This produces a large reduction in the number of features used in the final classifier.

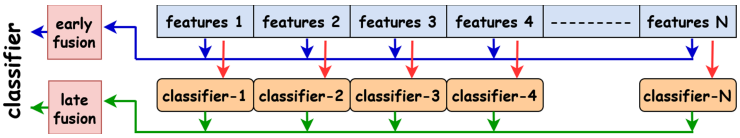

Fig. 3. Late fusion versus early fusion

Table 1. Fusion methodology

Fusion method	Definition
Concatenation	$f = [f_1, f_2, \ldots, f_N]$, where $f_i \in \mathbb{R}^{d_i}$ $f \in \mathbb{R}^{\sum_{k=1}^{N} d_k}$
Max	$f_k = \max_{i=1}^{d}(f_k^i)$, where $k = 1, \ldots, N$ and $f \in \mathbb{R}^N$
Sum	$f_k = \sum_{i=1}^{d}(f_k^i)$, where $k = 1, \ldots, N$ and $f \in \mathbb{R}^N$

4 Experiments and Results

We investigated several architectures to test our model in 4 benchmarks: CIFAR-10 [29], CIFAR-100 [29], MNIST [26] and Tiny ImageNet [30]. Table 2 presents the best configurations we found for the four datasets. Each entry gives the receptive field size and the number of output filters for that layer. The filter size for all of our experiments has fixed to 3×3. The classifier we ran for every convolution layer is the linear discriminant analysis (LDA), and the posteriors generated of the LDA classifiers were averaged and sent to SVM [28] to produce the final prediction. More details about these configurations and their accuracies have been discussed in the following subsections.

Table 2. Configurations for CIFAR-10/100, MNIST and Tiny ImageNet

CIFAR-10	CIFAR-100	MNIST	Tiny ImageNet
Input Image $32 \times 32 \times 3$		Input Image 28×28	Input Image $64 \times 64 \times 3$
3×3 conv-27		3×3 conv-9	3×3 conv-27
[3×3 conv-50] $\times 7$	[3×3 conv-50] $\times 4$	[3×3 conv-40] $\times 8$	2×2 max-pooling, stride $= 2$
			[3×3 conv-70] $\times 3$

4.1 Experiment on CIFAR-10 Database

CIFAR-10 database [29] consists of 50,000 coloured images for training and 10,000 images for testing. Each class contains objects that come with different angles and poses. The model used for testing CIFAR-10 as described in Table 2 consists of 8 layers with 27 filters for the first layer and 50 for the rest. We divided the feature maps of each layer into patches of size 8×8 with a stride $= 1$. Each convolution layer's second-order features are reduced using 3-levels SPP of 4×4, 2×2 and 1×1 subregions. The number of the first convolution layer features is 8508, and 27825 from the remaining layers.

Table 3 compares our model's accuracy with PCANet-2 and the current state of the art results (without data augmentation). The accuracy of our model is 4.58% better than PCANet-2. While the current state of the art methods achieved around 12% accuracy better than our model, our model is competitive with some simpler deep learning methods and learns features in a one-pass closed-form algorithm. So, the result is still promising.

Table 3. Comparison of the accuracy (%) of some methods on the CIFAR-10/CIFAR-100 database with no data augmentation

Method	CIFAR-10	CIFAR-100
Stochastic pooling [12]	84.87	57.49
Maxout network [13]	88.32	61.43
Network in network [14]	89.59	64.32
ALL-CNN [15]	90.2	–
Fractal network [16]	89.82	64.66
110 ResNet reported by [17,18]	86.82	55.26
ResNet stochastic depth [17]	–	62.20
164-ResNet(pre-activation) reported by [18]	–	64.42
Dense network(k = 24) [18]	94.08	76.58
Dense network-BC (k = 24)[18]	94.81	80.36
PCANet-2 [2]	77.14	51.62
PCANet-2 (combined) [2]	78.67	–
Multi-layer PCANet (ours)	81.72	57.86

4.2 Experiment on CIFAR-100 Database

CIFAR-100 database [29] is similar to CIFAR-10 but with 100 classes. The model we used in this experiment is identical to the one we used for CIFAR-10 but with 5 convolutional layers. We choose the number of layers to be five because there is no improvement in the accuracy when using eight layers as in CIFAR-10.

Table 3 compares the results achieved by our model with PCANet and other neural networks-based models (without data augmentation) including Residual network, Fractal network [16] network in network [14] and dense network [18]. The PCANet result has been achieved by running the same model used for CIFAR-10 [2], but with the CIFAR-100 database. The results show that our 5-layers PCANet accuracy is 6.24% better than the 2-layers PCANet and 2.60% better than ResNet with 110 layers, and about the same error rate as the stochastic pooling method [12]. The dense-network achieved the best performance with an error rate of 22% less than our model. Again our method improves on PCANet and is competitive with some older CNN-based methods, although it is not as good as more recent ones.

4.3 Experiment on the MNIST Database

The MNIST dataset [26] consists of 60,000 training examples and 10,000 test samples organized into 10 classes. Testing on this dataset was also performed without data augmentation. As described in Table 2, we used 9 convolutional layers, with 9 filters for the first layer and 40 for the rest. The performance reported here is the last convolution layer's accuracy, where we divided the 40 feature-maps into patches of size = 7×7 and stride = 1. The number of dimensions of the last layer has reduced using 3-level SPP of 16, 4 and 1 bins. The classifier used here is also the LDA, and the number of dimensions is 18060.

Table 4 compares the results obtained in this study with those obtained by PCANet [2]. The performance of our model is equivalent to those of PCANet-2, LDANet-2 and PCANet-1 (k = 13). Our interpretation of this situation is that the accuracy was good, and could not improve much by adding more layers.

Table 4. Comparison of cthe MNIST database with no data augmentation

Method	Accuracy(%)
PCANet-1 [2]	99.06
PCANet-2 [2]	99.34
LDANet-1 [2]	99.02
LDANet-2 [2]	99.38
PCANet-1 (k = 13) [2]	99.38
Multi-layer PCANet (ours)	99.40

4.4 Experiment on Tiny ImageNet

Tiny ImageNet database [30] consists of 100,000 training images divided into 200 categories. The validation and the test sets contain 10,000 images each, with 50 images per class. The test set is not labelled, and the results here are reported on the validation set. The model used for this experiment, as described in Table 2 consists of 4 convolution layers and one max-pooling layer with 27 filters generated from the first layer and 70 filters produced by the next layers. We introduced the max-pooling layer here to reduce the size of the output images. We divided the output images into patches of size 16×16 with stride = 1. The first layer's covariance features were pooled using 3-level SPP with 16, 4 and 1 bins. The next convolution layers have been connected to 2-level SPP with 4 and 1 bins. The number of features generated = 8508 from the first layer and 9450 from the remaining convolution layers.

Table 5 displays our model's accuracy compared to PCANet-2, ResNet34 and ResNet-50 reported by [27] without data augmentation. We tried to choose the best parameters to run PCANet-2 on 1TB memory. Therefore, PCANet-2 was trained with filter size k1 = k2 = 5, the number of filters L1 = 30, L2 = 8, and the block size = 16×16 with overlapping ratio = 0.5. To the best of our knowledge, we obtained the best error rate with no data augmentation.

Table 5. Comparison of the accuracy of some methods on the Tiny ImageNet database with no data augmentation

Method	ResNet-34 [27]	ResNet-50 [27]	PCANet-2	Multi-layer PCANet (ours)
Accuracy(%)	33.50	26.20	30.00	40.87

5 Conclusions

In this paper, we have presented some new refinements to the PCANet family of image classification methods to improve the performance and reduce the number of features. We introduced late fusion to preserve the information from all layers, adopted multi-channel convolutional layers and used second-order pooling with z-score normalisation. These substantially reduce the number of generated features and allow us to use deeper networks. We have shown that this offers improved performance over PCANet and results which are competitive with some simpler CNN architectures. We believe this is promising for a method where the features are unsupervised, but this is also a weakness of the architecture because we cannot learn which features are important for classification. In future work, we intend to study supervised convolutional layers where the filters are learnt with simple closed-form solutions in the same spirit as PCANet.

References

1. Low, C.Y., Teoh, A.B.-J., Toh, K.-A.: Stacking PCANet+: an overly simplified convnets baseline for face recognition. IEEE Signal Process. Lett. **24**(11), 1581–1585 (2017)
2. Chan, T.-H., et al.: PCANet: a simple deep learning baseline for image classification? IEEE Trans. Image Process. **24**(12), 5017–5032 (2015)
3. Yu, K., Salzmann, M.: Statistically-motivated second-order pooling. In: Proceedings of the European Conference on Computer Vision (ECCV) (2018)
4. Wang, Q., et al.: Deep CNNs meet global covariance pooling: better representation and generalization. arXiv preprint arXiv:1904.06836 (2019)
5. Carreira, J., Caseiro, R., Batista, J., Sminchisescu, C.: Semantic segmentation with second-order pooling. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7578, pp. 430–443. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33786-4_32
6. Mao, Y., Wang, R., Shan, S., Chen, X.: COSONet: compact second-order network for video face recognition. In: Jawahar, C.V., Li, H., Mori, G., Schindler, K. (eds.) ACCV 2018. LNCS, vol. 11363, pp. 51–67. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-20893-6_4
7. Wang, Q., et al.: RAID-G: robust estimation of approximate infinite dimensional Gaussian with application to material recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016)
8. Liu, Yu., Guo, Y., Georgiou, T., Lew, M.S.: Fusion that matters: convolutional fusion networks for visual recognition. Multimedia Tools Appl. **77**(22), 29407–29434 (2018). https://doi.org/10.1007/s11042-018-5691-4

9. Ergun, H., et al.: Early and late level fusion of deep convolutional neural networks for visual concept recognition. Int. J. Semant. Comput. **10**(03), 379–397 (2016)
10. Ebersbach, M., Herms, R., Eibl, M.: Fusion methods for ICD10 code classification of death certificates in multilingual corpora. In: CLEF (Working Notes), September 2017
11. Fan, C., et al.: PCANet-II: when PCANet meets the second order pooling. IEICE Trans. Inf. Syst. **101**(8), 2159–2162 (2018)
12. Zeiler, M.D., Fergus, R.: Stochastic pooling for regularization of deep convolutional neural networks. arXiv preprint arXiv:1301.3557 (2013)
13. Goodfellow, I., et al.: Maxout networks. In: International Conference on Machine Learning. PMLR (2013)
14. Lin, M., et al.: Network in network. arXiv preprint arXiv:1312.4400 (2013)
15. Springenberg, J.T., Dosovitskiy, A., Brox, T., Ried-miller, M.: Striving for simplicity: the all convolutional net. arXiv preprint arXiv:1412.6806 (2014)
16. Larsson, G., Maire, M., Shakhnarovich, G.: FractalNet: ultra-deep neural networks without residuals. arXiv preprint arXiv:1605.07648 (2016)
17. Huang, G., Sun, Yu., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 646–661. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_39
18. Huang, G., et al.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)
19. Krizhevsky, A., et al.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (2012)
20. Ng, C.J., Teoh, A.B.J.: DCTNet: a simple learning-free approach for face recognition. In: 2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA). IEEE (2015)
21. Xi, M., et al.: Local binary pattern network: a deep learning approach for face recognition. In: 2016 IEEE international conference on Image processing. IEEE (2016)
22. Zhang, Y., Geng, T., Wu, X., Zhou, J., Gao, D.: ICANet: a simple cascade linear convolution network for face recognition. EURASIP J. Image Video Process. **2018**(1), 1–7 (2018). https://doi.org/10.1186/s13640-018-0288-4
23. Wu, D., et al.: Kernel principal component analysis network for image classification. arXiv preprint arXiv:1512.06337 (2015)
24. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv: 1409.1556 (2014)
25. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 818–833. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_53
26. LeCun, Y., et al.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
27. Abai, Z., Rajmalwar, N.: DenseNet models for tiny imagenet classification. arXiv preprint arXiv:1904.10429 (2019)
28. Fan, R.-E., et al.: LIBLINEAR: a library for large linear classification. J. Mach. Learn. Res. **9**, 1871–1874 (2008)
29. Krizhevsky, A., et al.: Learning multiple layers of features from tiny images (2009)
30. https://tiny-imagenet.herokuapp.com/
31. Qaraei, M., et al.: Randomized non-linear PCA networks. Inf. Sci. **545**, 241–253 (2021)