Journal of Big Data

RESEARCH Open Access

Unsupervised feature learning-based encoder and adversarial networks

Endang Suryawati, Hilman F. Pardede*, Vicky Zilvan, Ade Ramdan, Dikdik Krisnandi, Ana Heryana, R. Sandra Yuwana, R. Budiarianto Suryo Kusumo, Andria Arisal and Ahmad Afif Supianto

*Correspondence: hilm001@lipi.go.id
Research Center for Informatics, Indonesian Institute of Sciences, Bandung, Indonesia

Abstract

In this paper, we propose a novel deep learning-based feature learning architecture for object classification. Conventionally, deep learning methods are trained with supervised learning for object classification. But, this would require large amount of training data. Currently there are increasing trends to employ unsupervised learning for deep learning. By doing so, dependency on the availability of large training data could be reduced. One implementation of unsupervised deep learning is for feature learning where the network is designed to "learn" features automatically from data to obtain good representation that then could be used for classification. Autoencoder and generative adversarial networks (GAN) are examples of unsupervised deep learning methods. For GAN however, the trajectories of feature learning may go to unpredicted directions due to random initialization, making it unsuitable for feature learning. To overcome this, a hybrid of encoder and deep convolutional generative adversarial network (DCGAN) architectures, a variant of GAN, are proposed. Encoder is put on top of the Generator networks of GAN to avoid random initialisation. We called our method as EGAN. The output of EGAN is used as features for two deep convolutional neural networks (DCNNs): AlexNet and DenseNet. We evaluate the proposed methods on three types of dataset and the results indicate that better performances are achieved by our proposed method compared to using autoencoder and GAN.

Keywords: Generative adversarial network, Unsupervised feature learning, Autoencoder, Convolutional neural networks

Introduction

Interests in applying machine learning technologies for object recognition have increased greatly in recent years [1–11]. The advancements of deep learning technologies are the drivers of the progress in the field [12]. Convolutional neural networks (CNNs) [13–15] are the dominant deep learning architectures for image data. Studies have shown CNN is better than traditional machine learning methods for many fields of object recognition, such as face recognition [16, 17], character recognition [4, 18], vehicle number detection [19, 20], vehicle surveillance in autonomous driving [21], medical imaging [22], identification of plant varieties [23–26], quality inspection of agricultural products [27], and detection of plant diseases [28–37]. This is due to their ability to

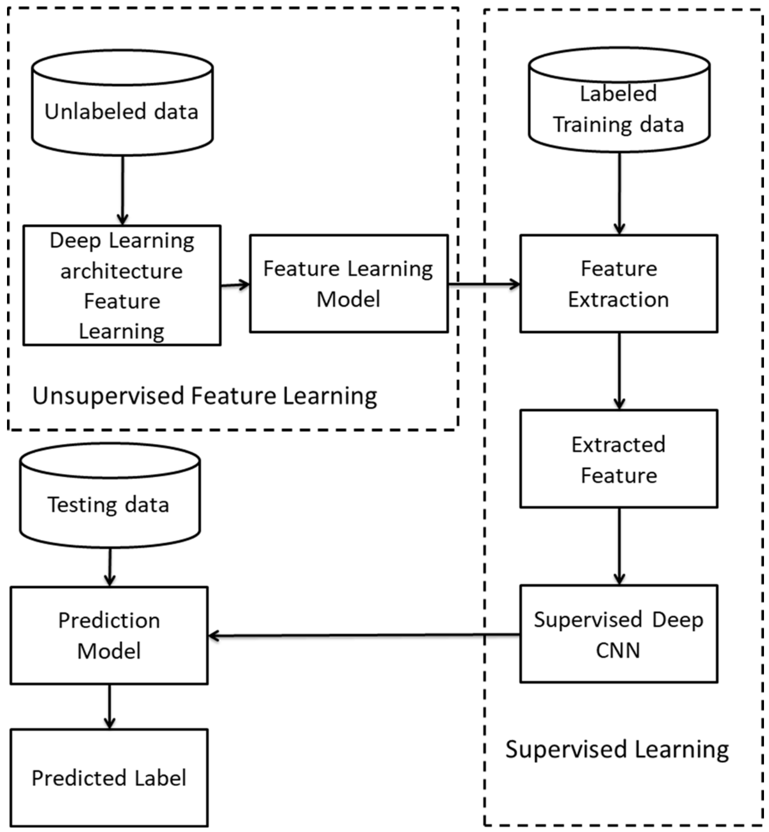

Fig. 1 Development and evaluation feature learning model

model complex relations between data with their target classes which are not easily be done by traditional shallow machine learning methods.

Many implementations of CNNs for object recognition employ them in supervised learning. The networks' parameters are optimized to map the data to the corresponding labels. Due to the great amount of the CNNs' parameters, a great amount of data training is needed. Otherwise, the networks are prone to over-fit, making them sensitive when they are tested with data from different distributions as in the training. However, collecting large amount of labelled data is expensive and requires huge efforts.

Training CNNs in unsupervised learning is one of the solutions for the above mentioned problem. Collecting unlabelled data is much easier and less expensive. Feature learning is one implementation of unsupervised learning for deep learning [38]. The implementation of deep learning as feature learning is illustrated in Fig. 1. In feature learning, the deep learning architectures are trained such that they to "learn" features automatically from unlabelled the data. After finishing the training, the resulting model is used to extract "features" from labelled data. These features are then be used as inputs to classifiers for object classifications. Several implementations of feature learning could refer to [25, 36, 37, 39–45].

Autoencoders and generative adversarial networks (GANs) are examples of deep learning methods that are applied for feature learning. An autoencoder is a network that comprises of two sub-networks: encoder and decoder. An encoder is designed so that the number of output nodes is greatly larger than the number of the input nodes. The aim is to compress the data into their latent variables. Meanwhile, a decoder is the mirror network of the encoder. The aim is to decompresses data back to the original dimensions. In [36], autoencoders are applied for the detection of plant diseases. In [42], they are applied for detection of epilepsy using EEG signals. Bottleneck features are introduced using autoencoders in [25] to reduce the data dimensionality. But, autoencoders have major limitations. Autoencoders are often merely "memorizing" the data during training, making them prone to over-fit. Several variants of autoencoders have been proposed to improve them. They are denoising autoencoders [43, 44], variational autoencoder (VAE) [45], and their combinations [37].

Meanwhile, GANs [46] are originally designed to generate new data from blindly presumed distributions of real data. They comprise two networks: generator (G) and discriminator (D). G aims to generate fake data that can fool D without ever really sees the distributions of real data, while D aims to discriminate fake images that are generated by G and the real ones. After the training is finished, G models are then used to generate additional data. Several implementations of GANs have been proposed in literature, for instance for generating more training data [47, 48], for image translation [49, 50], for creating super-resolution images [51], and for predicting the succeeding frames in a video [52]. There have been many variants of GAN that have been proposed. One of them is deep convolutional GAN (DCGAN) [39]. In DCGAN, CNNs are used to replace multi-layer perceptrons in the original GAN.

In recent years, GANs and their variats are also used for features learning [53]. DCGAN and deep neural networks (DNN) are combined for feature learning in [40]. BiGAN is introduced in [41], where GAN is designed to learn the latent representation of the data. For various implementations of GANs for feature learning could refer to [54, 55].

Unfortunately, the use of random noise as input for G network makes GAN's learning projection to be unpredictable. To overcome this, we propose to use hybrid networks of encoders and GANs. We adopt the DCGAN architecture and place the encoder at the top of the G network. We call it Encoder-GAN (EGAN). By doing so, the encoder could learn the latent variables of the data and then passes them to the G network of GAN. So, instead of using random distributions as inputs, G uses latent variables of real data. Hence, the learning projection of G could be more predicted.

The use of an encoder with GAN has been proposed previously. They are RDCGAN [40] and BiGAN [41]. However, there are differences between them. Their differences are illustrated in Fig. 2. In the RDCGAN, G accepts two types of inputs. They are that of the encoder and random distributions. Meanwhile, for BiGAN, the inputs of G are still from random distributions. Encoder is used in parallel with the G networks where both outputs are used as inputs for the D network. Meanwhile, G in EGAN only accepts outputs of the encoder as inputs and the D network only accept output of G as inputs. By doing so, G is expected to have more knowledge about the data. In addition to the new architecture, we also introduce nested training schemes for training the EGAN and DCNN networks. Usually, when GAN is used for feature learning, it is trained separately with the classifier. The GAN networks are trained first, and

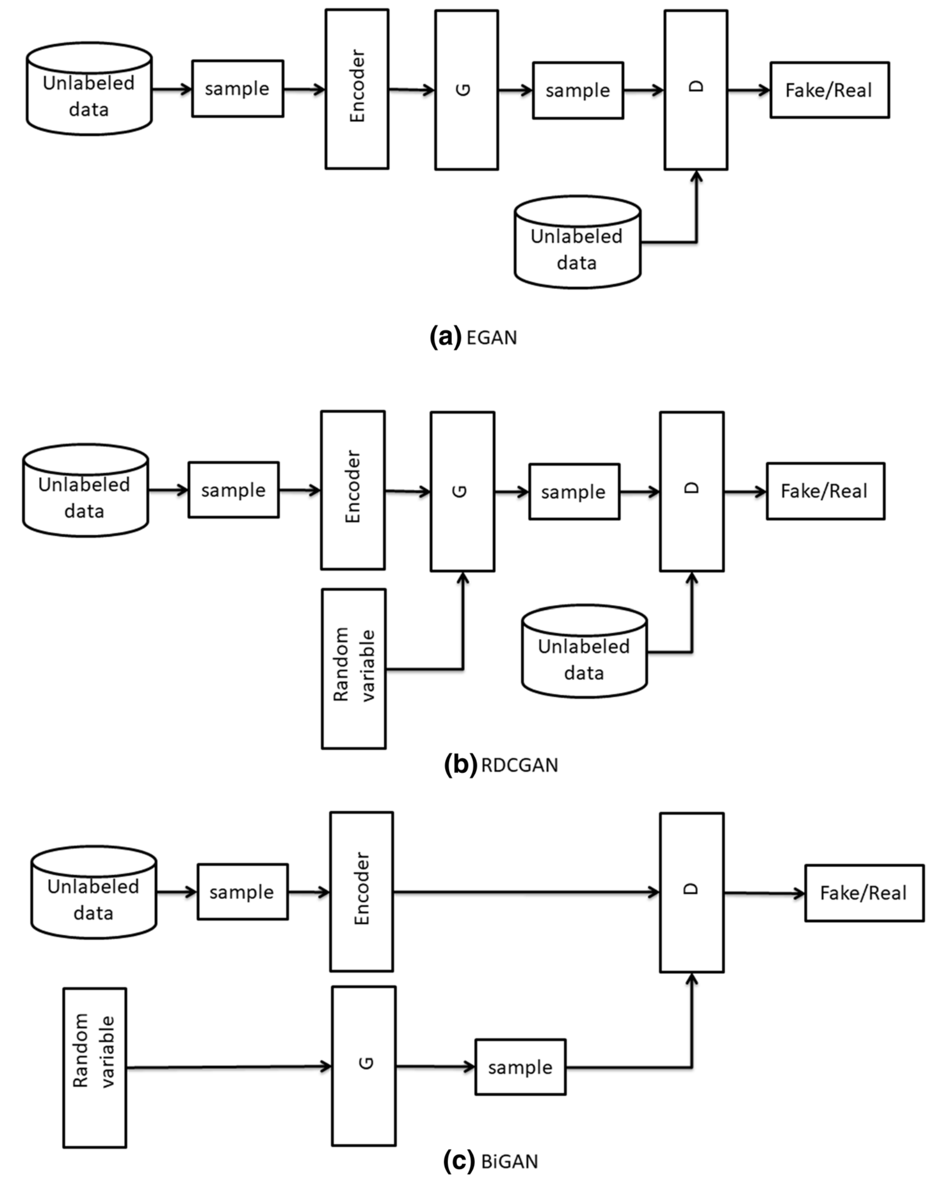

Fig. 2 Comparison between EGAN, RDCGAN, and BiGAN

then after the training is finished, it is used to extract features of labelled training data to train the classifiers. Here, we train EGAN and the classifier together. Hence, the results of classification are also used as feedbacks for EGAN.

This work is an extension of our work in [24]. In that paper, EGAN is applied to multi-layer perceptron classifiers, while in this paper, we extend it as inputs to deep CNN architectures: Alexnet and DenseNet. Furthermore, we evaluate the generalization capability of EGAN to wider tasks. In this paper, we evaluate EGAN with two publicly available datasets: PlantVillage, a dataset for plant diseases detection, and MNIST dataset, a popular dataset for benchmarking methods for object classification, in addition to Tea clone dataset. Comparisons with autoencoder, GAN, and BiGAN show that EGAN is more effective for all three datasets.

The remainder of the paper is organized as follow. In "A review of deep convolutional neural network (DCNN)" section, we briefly explain about CNN. We also briefly explain about GAN and autoencoder. We explain our proposed method in "The proposed method" section. The experimental setup is described in "Experimental setup" section and the results are explained in "Results and discussions" section. The paper is concluded in "Conclusions" section.

A review of deep convolutional neural network (DCNN)

In this section we first briefly explain about the use of DCNN for supervised learning. A typical composition of DCNN is explained. Then, we describe the use of DCNN for unsupervised learning. We explain both autoencoder and GAN briefly.

Figure 3 shows a typical structure of DCNN. DCNN has two main components. They are the feature extraction and the classification sections. The feature extractor usually consists of stacks of convolutional layers and the pooling layers. Weighted sum operations between the inputs and the kernels are conducted in the convolutional layers while the pooling layers "summarize" the output of convolutional layers, typically by taking the maximum or the average values and reduce the dimensionality of the inputs. The classification section usually consists of fully-connected layers that maps the outputs of feature extractors to the target labels.

DCNN could also be trained in an unsupervised way. The DCNN architecture is designed such that the network learns automatically the "important" underlying pattern of the data. Therefore, we can train DCNN to learn the features using unlabelled data. This is called feature learning. Then, after training the DCNN models in unsupervised way, they are used to extract features from a small amount of labelled data which are used to train classifiers in a supervised way. Two common architectures for feature learning are autoencoder and GAN.

The architecture of an autoencoder is shown Fig. 4. A typical autoencoder consists of two sub-networks: encoder and decoder. In the encoder, the dimensions of the input data are forcedly reduced to obtain compressed features. Then, the decoder reconstructs the data from compressed features to get the original data. The compressed features could be seen as a latent representation of data. By forcing the number of the outputs to be hugely reduced from the inputs, the encoder

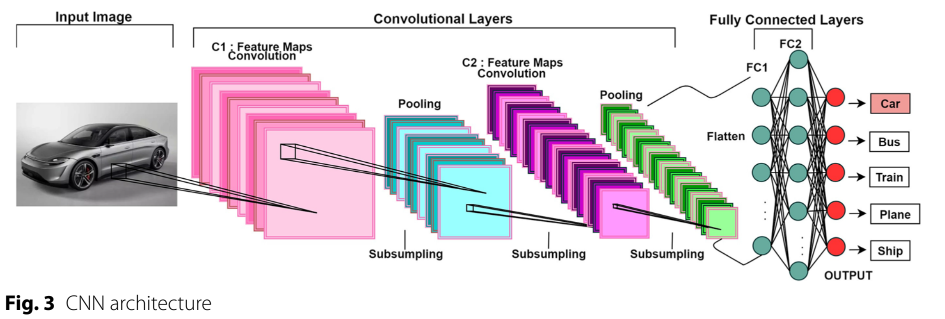

Fig. 3 CNN architecture

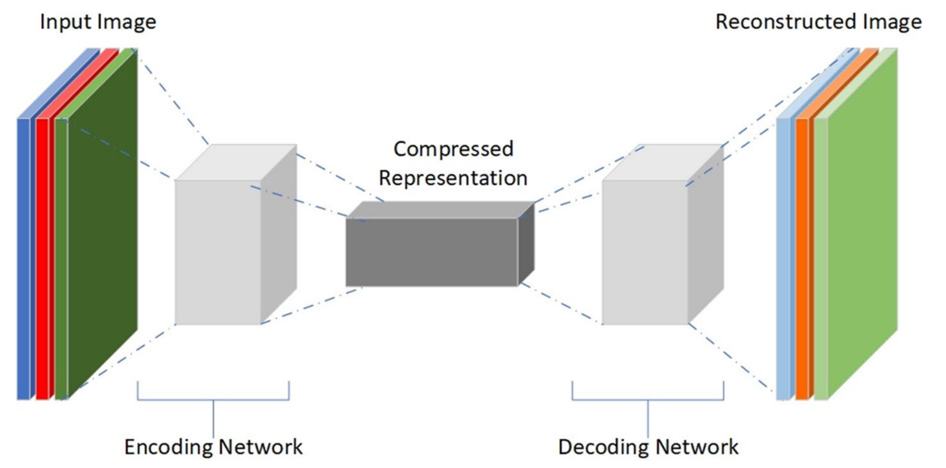

Fig. 4 A typical architecture of an autoencoder

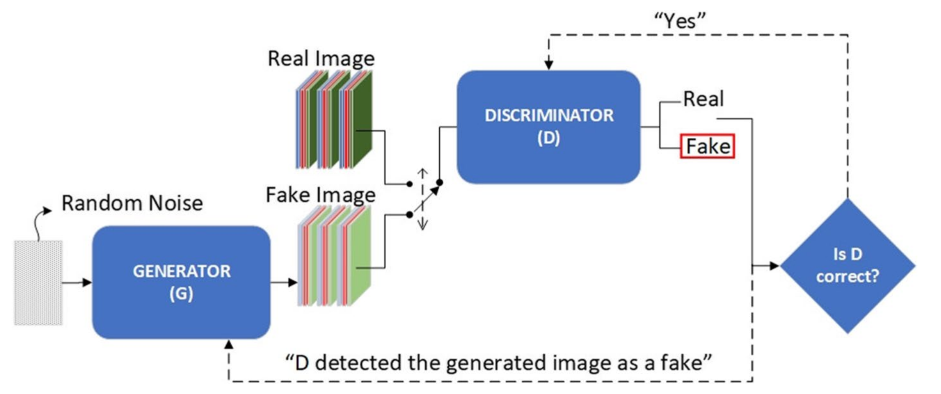

Fig. 5 The architecture of a generative adversarial network

learns underlying latent information about the data that are relevant and discard the unuseful information. In other words, the encoder could be seen as a nonlinear principal components analysis of the data.

GAN is another deep learning method that employ unsupervised learning. A structure of GAN is shown in Fig. 5. GAN consists of two networks. They are Generator (G) and Discriminator (D) networks. GANs work by contending both G and D. Generator learns to generate images through the presumptive distributions that it has never seen before. Meanwhile, D assesses the generated images, whether they are real or fake images. The D's assessment results would be the reference for G to optimize the presumptive distributions and generate images closer to real ones to fool D. This adversarial training continues until D cannot distinguish between real and fake images. GANs cannot remember input data like the autoencoder since the real data are never seen. Therefore, GANs can avoid the overfitting problem on the networks. However, GANs are sensitive to initialization since the presumptive distributions for G are usually random, making the learning trajectories to be unpredicted.

The proposed method

The grand scheme of using deep learning for feature learning have been explained in Fig. 1. In our implementation, we design an architecture, called EGAN as feature learning where it is trained with unlabelled data to obtain the feature learning model. This model is used to extract features from the labelled that, where the features are used to train Deep CNN (DCNN).

Figure 6 show the architecture of EGAN. EGAN is a hybrid between encoder and DCGAN. We put the encoder on the top of the Generator of the DCGAN network. Traditionally, the G networks of GAN accept random distribution (usually Gaussian) as inputs. But, this would make the learning trajectory of GAN to be unpredictable. So, the purpose of using encoder is to avoid that.

Let $p_{\mathbf{x}}$ is the target distribution of real data $\mathbf{x}$. Let p_z is noise distribution that G would map to data $\mathbf{z}$. G aim to learn distribution $p_g(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[p_g(\mathbf{x})|\mathbf{z})$ so that $p_g(\mathbf{x}) = p_{\mathbf{x}}$. To do this, GAN plays minimax game of function V between G and D as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log (1 - D(G(\mathbf{z})))] \tag{1}$$

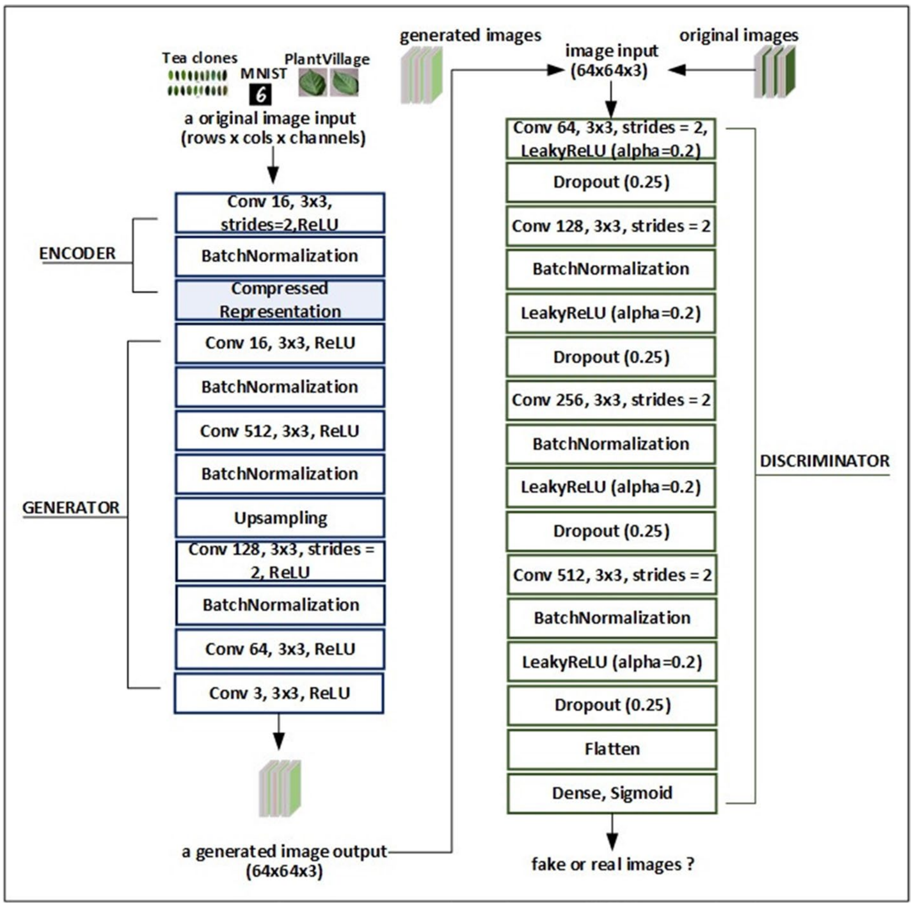

Fig. 6 Architecture of the proposed EGAN

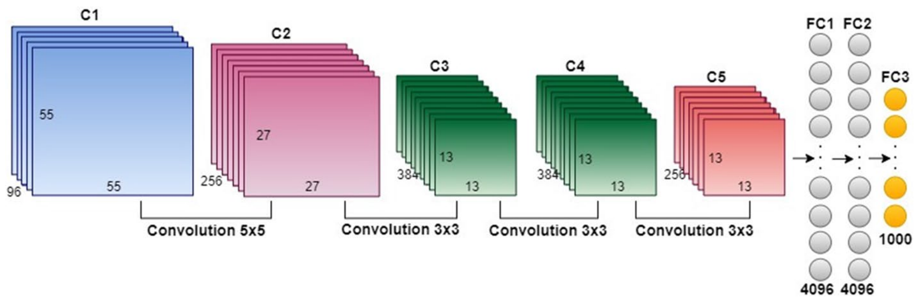

Fig. 7 Architecture of AlexNet

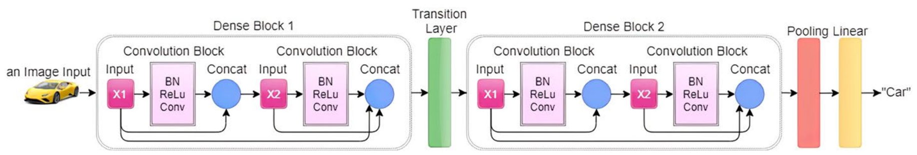

Fig. 8 Architecture of DenseNet

However, since G never sees actual data, the learning trajectories of p_g could be very far from $p_{\mathbf{x}}$. It makes the Generator to be very sensitive to initialization. To avoid this, we employ encoder E, so now E introduces distribution $p_E(\mathbf{z}|\mathbf{x}) = p_z(\mathbf{z})$ that map the real data $\mathbf{x}$ to latent variables $\mathbf{z}$. Therefore, (1) becomes:

$$\min_{G,E}\max_{D} V(D,G,E) = \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_E(\mathbf{z}|\mathbf{x})}[\log(1 - D(G(\mathbf{z})))] \tag{2}$$

By doing so, G does not learn from random initialisation. By putting encoder on top of G networks, we expect the learning process of G to be more predictable. Instead of blindly learn the distribution of the real data, the G network of GAN has prior knowledge of the data.

Here, we use DCGAN [39] as the backbone for GAN. It is a variant of GAN that employ convolutional layers instead of multilayer perceptrons as in GAN. DCGAN is found to be more stable in the training process than GAN. Meanwhile, the encoder is built by using one convolutional layer and one BatchNormalization process. We use encoder with convolutional layer with stride 2 and reduce the RGB input dimensions to a half of the original image's dimensions. This is the optimum setting for the encoder based on our observation. We only use a single convolutional layer and BatchNormalization for the encoder since we found that having more convolutional layers are not effective for the tasks and the performance tends to get worse. We do not use the decoder in our design as we aim to find the underlying latent variables of the data as input to GAN. Therefore, we need an architecture that compress the data into much smaller dimensions. Hence, we employ only encoder for this as adding decoder would reconstruct the data to the original dimensions.

For supervised DCNN, we use two architectures: AlexNet [13] and DenseNet [56]. The architectures of AlexNet and DenseNet are shown in Figs. 7 and 8 respectively. AlexNet comprises of stacked five convolutional layers that are followed by three fully connected layers. Meanwhile, DenseNet is much deeper than AlexNet. To avoid gradient vanishing problems that occur in very deep CNN, DenseNet apply multi-skip connections that carry information, pass it over for several layers, and then concatenate them, adding more width to the networks. Due to its intense concatenation operation to all previous layers, DenseNet requires larger memory usage.

To train EGAN and DCNN, we modify gradient based optimization as in [46]. We apply nested training for training EGAN and DCNN classifiers. In this paper, we use epoch $L = 200$ to train EGAN, and for each epoch, we train DCNN using sub-epoch $K = 5$. The detailed algorithm for training is described as follows:

Algorithm 1: Minibatch stochastic gradient descent for EGAN. There are number of sub-iteration, denoted as K is applied. Here $K = 5$ is applied

```
initialization;
for  $L$ steps do
    • Sample minibatch $m$ from data $\mathbf{x}$ to encoder generating distribution $p_E(\mathbf{z}|\mathbf{x})$
      and obtain $m$ samples $\{\mathbf{z}^{(1)}, ..., \mathbf{z}^{(m)}\}$ for generator;
    • Sample minibatch $m$ from data $\{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(m)}\}$, generating $p_{\mathbf{x}}(\mathbf{x})$;
    • Update Discriminator by using gradient descent
    • Sample minibatch $m$ from data $\mathbf{x}$ to encoder generating distribution $p_E(\mathbf{z}|\mathbf{x})$
      and obtain $m$ samples $\{\mathbf{z}^{(1)}, ..., \mathbf{z}^{(m)}\}$ for generator;
    • Update Generator using gradient descent;
    for  $K$ steps do
        • Sample minibatch $n$ samples from data $\mathbf{x}$ to generator and obtain features
          $\{\mathbf{v}^{(1)}, ..., \mathbf{v}^{(n)}\}$ for generator;
        • Update DCNN classifiers using gradient descent
    end
end
```

Experimental setup

The dataset

We use three datasets for evaluating our method. They are the Tea Clone, PlantVillage, and MNIST datasets. We develop a tea clone dataset for tea clones recognition tasks [24]. PlantVillage and MNIST are public datasets. PlantVillage contains images of leafs with various diseases, whereas MNIST is used for handwriting character recognition.

For the tea clones dataset, we collected 4520 tea leaf images to build the dataset from the Research Institute for Tea and Cinchona (RITC) in Gambung, West Java, Indonesia. There are 11 types of Gambung tea clones. They are called the GMB (Gambung) Clone series. This research focuses only on two tea clones, and they are GMB 3 and GMB 9. We used 14 different types of cameras, DSLR cameras, and smartphones. We intend to enrich the data variation in the way of taking pictures through the various cameras. We capture images indoor and ignore the lighting arrangement and the distance of the leaf to the camera. We use the autofocus feature when capturing the image. The image samples of GMB 3 and GMB 9 are shown in Fig. 9.

Penn State University develops the PlantVillage (PV) dataset. This dataset is created based on the premise that food growth knowledge should be accessible openly to everyone. We choose three plants (corn, potato, and apple) of this dataset. Some of the samples are shown in Fig. 10.

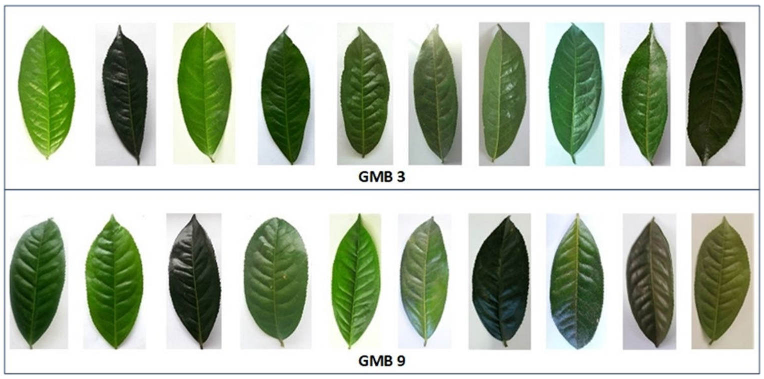

Fig. 9 The samples of GMB 3 and GMB 9

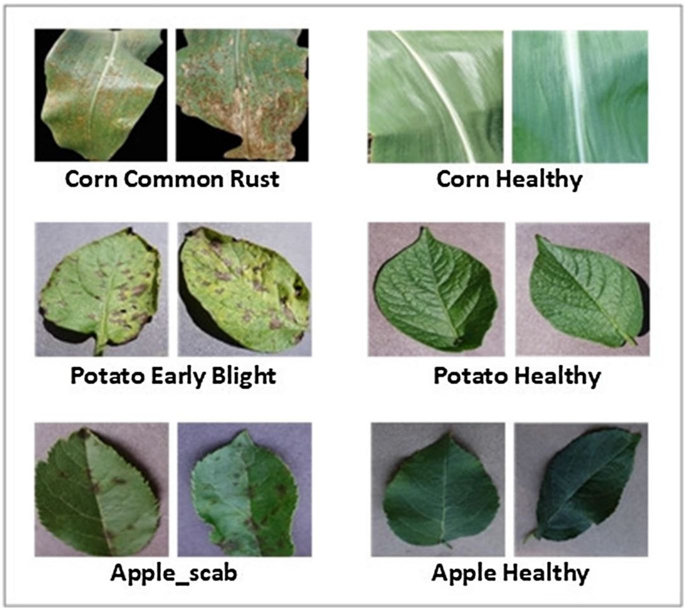

Fig. 10 The samples of PlantVillage dataset

MNIST (Modified Nasional Institute of Standards and Technology) is a dataset of handwritten digits 0 to 9. It is released in 1999 and become the benchmarking standard for classifications. Fig. 11 show the sample of the MNIST dataset. There are 70,000 images in the MNIST dataset. It is consists of 60,000 training images and 10,000 testing images. The training and testing images contain greyscale images.

System setup

Each dataset is divided into three parts: training, validation, and testing sets. The detailed distribution of data for all datasets is summarized in Table 1. We train EGAN using three datasets separately. For tea clones and PlantVillage datasets, each dataset is divided into three parts; 80% of training, 10% of validation, and 10% of testing. Meanwhile, we use 23,000 images in MNIST consisting of 20,000 training images and

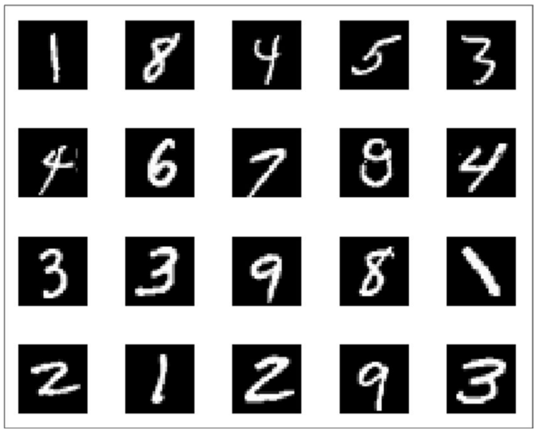
Fig. 11 The sample of MNIST dataset

Table 1 The data distribution in the experiments

Dataset	Class labels	Training	Validation	Testing	Total
Tea clones	GMB 3	1832	232	231	2295
	GMB 9	1784	220	221	2225
PV corn	Cercospora leaf spot	449	32	32	513
	Common rust	1031	81	80	1192
	Northern leaf blight	849	68	68	985
	Healthy	1017	73	72	1162
PV potato	Early blight	875	63	62	1000
	Late blight	875	62	63	1000
	Healthy	131	10	11	152
PV apple	Apple scab	552	39	39	630
	Black rot	544	39	38	621
	Cedar apple rust	241	17	17	275
	Healthy	1440	103	102	1645
MNIST)	0–9	16,000	4000	3000	23,000

3000 testing images. For validation, we separate 20% of training data. Here, we use the same data to train EGAN and DCNN classifiers: AlexNet, and DenseNet.

The capability of EGAN is evaluated as follows. First, EGAN's evaluation as the feature learning model using three datasets (the tea clones, PlantVillage, and MNIST) and two popular classifiers (AlexNet and DenseNet). This is evaluated using the accuracies of the classifier. We only use accuracy since the data is fairly balanced. Here, we compare EGAN with other feature learning methods: GAN, BiGAN, and autoencoder. For the referenced methods, we use GAN architecture as in [46], BiGAN as in [41] and whereas for autoencoder, we use the architecture DCNN-based autoencoder as in [36]. We denote this method CNN-AE from now on. For GAN, we apply the same training strategies as EGAN, while for CNN-AE, we train CNN-AE with the same number of the epoch as EGAN.

To find the best parameters for EGAN, we tune the hyper-parameters of the networks as follow:

1. Optimizer : Adam optimizer is applied in this study. For initialization, we use Adam (0.0002, 0.5) for EGAN and Adam(lr=1e−5) for AlexNet and DenseNet.
2. Batch size : We varied 8, 16, 32, and 64 for training EGAN, AlexNet, and DenseNet for all datasets. In our case, the choice of small batch size due to our computing capability.
3. Epoch: We use 200 epochs for training EGAN and 5 epochs for sub-training each classifier. These values are selected based on our empirical observation.

For computation, we used Intel Xeon E5-2695 as the CPU, with 512 GB memory size and NVidia Tesla P100 16 GB as the GPU. All the architectures are developed with library from Keras [57] with Tensorflow [58] as the back-end.

Results and discussions

EGAN as feature learning model

Table 2 summarises the performance of EGAN as feature extractor and DCNN classifiers when batch-sizes are varied for all evaluated datasets. For all datasets, satisfactory performance could be achieved, and in most cases, 16 may have been suitable for batch-size. For that reason, we will use it in subsequent experiments. Need to be noted however, better performance could be achieved when a larger batch-size is applied. However, due to the limitation of our memory size, especially for DenseNet implementation, which requires a high memory load, implementation with a larger size than 32 could not be implemented.

In most cases, we found that DenseNet achieves slightly better accuracies than AlexNet. It is not surprising since DenseNet has more depth and width than AlexNet, which may be better in modeling nonlinear relations between features and their class targets. However, DenseNet has more training parameters than AlexNet and slower to train.

Table 2 Performance of EGAN when batch size is varied for all evaluated datasets

Dataset	Classifier	Batch size			
		8	16	32	64
Tea clones	AlexNet	92.92	**94.02**	92.47	91.15
	DenseNet	94.24	**94.46**	92.25	94.02
PV corn	AlexNet	91.26	**93.25**	91.66	92.85
	DenseNet	96.03	95.63	**96.42**	96.42
PV potato	AlexNet	94.85	**97.05**	94.11	**97.05**
	DenseNet	98.52	**98.52**	96.32	97.79
PV apple	AlexNet	88.26	**93.87**	91.83	92.85
	DenseNet	96.93	95.91	**97.95**	95.40
MNIST	AlexNet	**97.86**	97.70	97.63	97.03
	DenseNet	98.43	**98.80**	98.63	98.43

The best performance for each dataset and classifier is printed in bold

Table 3 Comparison of performance (Accuracy %) between EGAN , other features and feature learning methods

Dataset	Classifier	Feature extraction				
		RGB	GAN	CNNAE	BiGAN	EGAN
Tea clones	AlexNet	90.48	81.85	87.83	78.76	**94.02**
	DenseNet	80.97	75.22	80.75	82.30	**94.46**
PV corn	AlexNet	92.46	84.12	92.86	83.73	**93.25**
	DenseNet	92.46	78.57	90.08	81.74	**95.63**
PV potato	AlexNet	89.70	83.08	91.91	85.29	**97.05**
	DenseNet	94.85	80.14	88.97	85.29	**98.52**
PV apple	AlexNet	91.32	76.53	87.76	78.57	**93.87**
	DenseNet	89.28	69.38	87.76	77.55	**95.91**
MNIST	AlexNet	96.10	90.96	95.30	90.20	**97.70**
	DenseNet	97.00	83.63	96.07	86.70	**98.83**

The best performance for each dataset and classifier is printed in bold

Table 4 Comparison of running time between EGAN and other feature learning methods

Dataset	Classifier	GAN	CNNAE	BIGAN	EGAN
Tea clones	AlexNet	17.28	1.42	18.37	21.82
	DenseNet	109.71	2.13	114.62	117.39
PV corn	AlexNet	15.90	1.54	16.41	18.20
	DenseNet	100.62	2.02	102.88	106.25
PVpotato	AlexNet	9.21	5.97	9.05	10.45
	DenseNet	57.13	1.17	58.00	60.00
PV apple	AlexNet	13.12	1.21	13.42	15.28
	DenseNet	83.53	1.69	85.12	87.30
MNIST	AlexNet	69.71	5.97	67.57	70.09
	DenseNet	333.67	7.85	336.89	334.16

Comparison with other feature learning methods

Table 3 compares the performance of EGAN with other features and feature learning algorithms. They are RGB (without feature learning), GAN, BiGAN, and CNN-AE. It is clear that EGAN is superior to others for all datasets. Need to be noted that in this study, the classifiers are all trained without pre-training. On average, relative improvements of 4.72%, 19.92%, 6.86%, and 15.55% are achieved compared to RGB, GAN, CNN-AE, and BiGAN respectively. Interestingly, we found, RGB is largely better than GAN and BiGAN. It is slightly better than CNN-AE in most cases. This may be expected, as when GAN is applied, the presumed distributions of the data may completely different than the real data, and hence, may cause mismatch when they are used in training. BiGAN is generally better than GAN. But due to the use of random variables to input of G, the trajectory of the learning may not be as targeted as our proposed method.

Table 4 summarizes the running time of GANm CNN-AE, BiGAN, and EGAN. It is clear that EGAN is considerably slower than GAN and CNN-AE. This is as expected due to its much larger number of parameters than GAN and CNN-AE. Furthermore, the nested training requires more computation per epoch since each epoch requires five sub-epochs to train the classifiers. Meanwhile, the training time for EGAN and BiGAN is quite comparable. But EGAN has superior performance.

Conclusions

In this paper, we propose encoder-deep convolutional generative adversarial network (EGAN) as a solution for unsupervised feature learning. In EGAN, an encoder is put on top of GAN's Generator networks to avoid GAN learns completely different distributions from data training. We use DCGAN architectures and encoder with one convolutional layers for EGAN. For supervised learning, we employ two DCNN architectures. They are AlexNet and DenseNet. In addition, we use nested training to train both EGAN and DCNN.

Our evaluations of three types of datasets, including MNIST datasets, confirm that our proposed method is better than directly using RGB as features in two popular DCNN classifiers. It is also largely better than three feature learning methods: GAN, CNN-AE, and BiGAN. Our evaluation also shows that GAN performs much worse than directly using RGB. This strongly indicate how unpredictable the learning outcome of GAN due to random noise inputs.

During experiments, we found that number of layers for encoder may affect the performance. Having more layers may not produce significant improvements and the performance tend to get worse. We also found the selection of number of epoch for nested training may be influential to the performance.

In the future, we plan to evaluate the robustness of EGAN. Implementation of the encoder variants, data preprocessing, or data augmentation for improvements of our method also becomes our interest in the future.

Abbreviations
DCNN: Deep convolutional neural networks; DCGAN: Deep convolutional generative adversarial network; PV: PlantVillage.

Acknowledgements
The experiments are conducted in High Performance Computing Facilities of Research Center for Informatics, Indonesian Institute of Sciences.

Authors' contributions
HFP, ES made the contribution in the concept, design, analysist data and carried-out the experiments; VZ, AR, AAS, DK are participated in conducting experiments and the statistical analysist; RBSK, RSY, AH, AA are contributed in coding. All authors are contributed to the data collections. All authors read and approved the final manuscript.

Funding
This work is partially funded by INSINAS grant from Indonesian Ministry of Education, Culture, Research and Technology.

Availability of data and materials
The Plantvillage: https://github.com/spMohanty/PlantVillage-Dataset. The tea clones dataset that are used during the current study are not publicly available due to it is in the process of agreement between Research Center for Informatics and Research Institute for Tea and Cinchona but are available from the corresponding author on reasonable request. MNIST: https://www.kaggle.com/oddrationale/mnist-in-csv.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 3 May 2021 Accepted: 20 August 2021
Published online: 06 September 2021

References

1. Kasar MM, Bhattacharyya D, Kim T-H. Face recognition using neural network: a review. Int J Secur Appl. 2016;10:81–100. https://doi.org/10.14257/ijsia.2016.10.3.08.
2. Pandey S, Sharma S. Review: face detection and recognition techniques. Int J Comput Sci Inf Technol. 2014;5:4111–7.
3. Purohit A, Chauhan SS. A literature survey on handwritten character recognition. Int J Comput Sci Inf Technol. 2016;7:1–5.
4. Baldominos A, Sáez Y, Isasi P. A survey of handwritten character recognition with mnist and emnist. Appl Sci. 2019;2019:3169. https://doi.org/10.3390/app9153169.
5. Khaustov PA, Spitsyn VG, Maksimova EI. Algorithm for optical handwritten characters recognition based on structural components extraction. In: Proceedings of the 2016 11th international forum on strategic technology (IFOST); 2016. p. 355–8. https://doi.org/10.1109/IFOST.2016.7884126.
6. Fortunato S. An automated vehicle license plate recognition system. Computer. 2015;48:56–61.
7. Kumari R, Prakash S. A machine learning algorithm for automatic number plate recognition. Int J Comput Appl. 2017;174:6–9. https://doi.org/10.5120/ijca2017915297.
8. Mahendra O, Pardede HF, Sustika R, Kusumo BS. Comparison of features for strawberry grading classification with novel dataset. In: Proceedings of the 2018 international conference on computer, control, informatics and its applications (IC3INA); 2018. p. 7–12. https://doi.org/10.1109/IC3INA.2018.8629534.
9. Kusumo BS, Heryana A, Mahendra O, Pardede HF. Machine learning-based for automatic detection of corn-plant diseases using image processing. In: Proceedings of the 2018 international conference on computer, control, informatics and its applications (IC3INA); 2018. p. 93–7. https://doi.org/10.1109/IC3INA.2018.8629507.
10. Liming X, Yanchao Z. Automated strawberry grading system based on image processing. Comput Electr Agric. 2010;71:32–9. https://doi.org/10.1016/j.compag.2009.09.013.
11. Ramdan A, Sugiarto PD B.and Rianto, Prakasa E, Pardede HF. Support vector machine-based detection of pak choy leaves conditions using rgb and his features. In: Proceedings of the 2018 international conference on computer, control, informatics and its applications (IC3INA); 2018. p. 114–7. https://doi.org/10.1109/IC3INA.2018.8629540.
12. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521:436–44. https://doi.org/10.1038/nature14539.
13. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th international conference on neural information processing systems; 2012. p. 1097–105. https://doi.org/10.1145/3065386.
14. Szegedy C, Liu W, Jia Y, Sermanet P, Reed SE, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. In: Proceedings of the 2015 IEEE conference on computer vision and pattern recognition (CVPR); 2015. p. 1–9. https://doi.org/10.1109/CVPR.2015.7298594.
15. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556; 2015. https://doi.org/10.1109/ACPR.2015.7486599.
16. Wang M, Deng W. Deep face recognition: a survey; 2020. https://arxiv.org/abs/1804.06655.
17. Zhang Y, Guo M, Yang Y. Face recognition learning using data augmentation based on orthogonal experiments. Electronics. 2019;8:1088. https://doi.org/10.3390/electronics8101088.
18. Giardino D, Matta M, Silvestri F, Spano S, Trobiani V. FPGA implementation of hand-written number recognition based on cnn. Int J Adv Sci Eng Inf Technol. 2019;9:167–71. https://doi.org/10.18517/ijaseit.9.1.6948.
19. Masood SZ, Shu G, Dehghan A, Ortiz EG. License plate detection and recognition using deeply learned convolutional neural networks; 2017. https://arxiv.org/abs/1703.07330.
20. Puarungroj W, Boonsirisumpun N. Thai license plate recognition based on deep learning. Procedia Comput Sci. 2018;135:214–21. https://doi.org/10.1016/j.procs.2018.08.168.
21. Giovany C, Suescun P, Arenas OP, Moreno RJ. Detection of scratches on cars by means of cnn and r-cnn. Int J Adv Sci Eng Inf Technol. 2019;9:745–52. https://doi.org/10.18517/ijaseit.9.3.6470.
22. Huynh B, Li H, Giger M. Digital mammographic tumor classification using transfer learning from deep convolutional neural networks. Phys Rev E. 2016;3:034501. https://doi.org/10.1117/1.JMI.3.3.034501.
23. Ramdan A, Suryawati E, Kusumo BS, Pardede HF, Mahendra O, Dahlan FR, Fauziah F, Syahrian H. Deep cnnbased detection for tea clone identification. Jurnal Elektronika dan Telekomunikasi (JET). 2019;19:45–50. https://doi.org/10.14203/jet.v19.45-50.
24. Suryawati E, Zilvan V, Yuwana RS, Heryana A, Rohdiana D, Pardede HF. Deep convolutional adversarial network-based feature learning for tea clones identifications. In: Proceedings of the 2019 3rd international conference on informatics and computational sciences (ICICoS); 2019. p. 1–5. https://doi.org/10.1145/3065386.
25. Yuwana RS, Suryawati E, Heryana A, Zilvan V, Rohdiana D, Syahrian H. Bottleneck rgb features for tea clones identification. In: Proceedings of the 2019 international seminar on research of information technology and intelligent systems (ISRITI); 2019. p. 259–62. https://doi.org/10.1109/ISRITI48646.2019.9034667.
26. Dyrmanna M, Henrik Karstoftb H, Henrik Skov Midtibya HS. Plant species classification using deep convolutional neural network. Biosyst Eng. 2016;151:72–80. https://doi.org/10.1016/j.biosystemseng.2016.08.024.
27. Sustika R, Subekti A, Pardede HF, Suryawati E, Mahendra O, Yuwana S. Evaluation of deep convolutional neural network architectures for strawberry quality inspection. Int J Eng Technol. 2018;7:75–80.
28. Suryawati E, Rika Sustika R, Sandra Yuwana RS, Subekti A, Pardede HF. Deep structured convolutional neural network for tomato diseases detection. In: Proceedings of the 2018 international conference on advanced

computer science and information systems (ICACSIS); 2018. p. 385–90. https://doi.org/10.1109/ICACSIS.2018.8618169.
29. Krisnandi D, Pardede HF, Yuwana RS, Zilvan V, Heryana A, Fauziah F, Rahadi VP. Diseases classification for tea plant using concatenated convolution neural network. CommIT (Commun Inf Technol) J. 2019;13:67–77. https://doi.org/10.21512/commit.v13i2.5886.
30. Barbedo JGA. Factors in uencing the use of deep learning for plant disease recognition. Biosyst Eng. 2018;172:84–91. https://doi.org/10.1016/j.biosystemseng.2018.05.013.
31. Liu B, Zhang Y, He D, Li Y. Identification of apple leaf diseases based on deep convolutional neural networks. Symmetry. 2018;10:11. https://doi.org/10.3390/sym10010011.
32. Ferentinos KP. Deep learning models for plant disease detection and diagnosis. Comput Electr Agric. 2018;145:311–8. https://doi.org/10.1016/j.compag.2018.01.009.
33. Geetharamani G, Pandian JA. Identification of plant leaf diseases using a nine-layer deep convolutional neural network. Comput Electr Eng. 2019;76:323–38. https://doi.org/10.1016/j.compeleceng.2019.04.011.
34. Chen J, Liu Q, Gao L. Visual tea leaf disease recognition using a convolutional neural network model. Symmetry. 2019;11:343. https://doi.org/10.3390/sym11030343.
35. Yuwana RS, Suryawati E, Zilvan V, Ramdan A, Pardede HF, Faiziah F. Multi-condition training on deep convolutional neural networks for robust plant diseases detection. In: Proceedings of the 2019 international conference on computer, control, informatics and its applications (IC3INA); 2019. p. 30–5. https://doi.org/10.1109/IC3INA48034.2019.8949580.
36. Pardede HF, Suryawati E, Sistika E, Zilvan V. Unsupervised convolutional autoencoder-based feature learning for automatic detection of plant diseases. In: Proceedings of the 2018 international conference on computer, control, informatics and its applications (IC3INA); 2018. p. 158–62. https://doi.org/10.1109/IC3INA.2018.8629518.
37. Zilvan V, Ramdan A, Suryawati E, S, KRB, Krisnandi D, Pardede HF. Denoising convolutional variational autoencoders-based feature learning for automatic detection of plant diseases. In: Proceedings of the 2019 3rd international conference on informatics and computational sciences (ICICoS); 2019. p. 1–6. https://doi.org/10.1109/ICICoS48119.2019.8982494.
38. Bengio Y, Courville A, Vincent P. Representation learning: a review and new perspectives. IEEE Trans Pattern Anal Mach Intell. 2013;35(8):1798–828. https://doi.org/10.1109/TPAMI.2013.50.
39. Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks; 2016. https://arxiv.org/abs/1511.06434.
40. Mehralian M, Karasfi B. Rdcgan: Unsupervised representation learning with regularized deep convolutional generative adversarial networks. In: Proceedings of the 2018 9th conference on artificial intelligence and robotics and 2nd Asia-Pacific international symposium; 2018. p. 31–8. https://doi.org/10.1109/AIAR.2018.8769811.
41. Donahue J, Krähenbühl P, Darrell T. Adversarial feature learning. In: 5th international conference on learning representations, iclr 2017, toulon, france, april 24-26, 2017, conference track proceedings. OpenReview.net; 2017. https://openreview.net/forum?id=BJtNZAFgg.
42. Wen T, Zhang Z. Deep convolution neural network and autoencoders-based unsupervised feature learning of eeg signals. IEEE Access. 2018;6:25399–410. https://doi.org/10.1109/ACCESS.2018.2833746.
43. Li B, Xu K, Feng D, Mi H, Wang H, Zhu J. Denoising convolutional autoencoder based b-mode ultrasound tongue image feature extraction. In: Proceedings of the ICASSP 2019—2019 IEEE international conference on acoustics, speech and signal processing (ICASSP); 2019. p. 7130–4. https://doi.org/10.1109/ICASSP.2019.8682806.
44. Chao L, Tao J, Yang M, Li Y. Improving generation performance of speech emotion recognition by denoising autoencoders. In: Proceedings of the the 9th international symposium on chinese spoken language processing; 2014. p. 341–4. https://doi.org/10.1109/ISCSLP.2014.6936627.
45. Tulala P, Mahyar H, Ghalebi E, Grosu R. Unsupervised wafermap patterns clustering via variational autoencoders. In: Proceedings of the 2018 international joint conference on neural networks (IJCNN); 2018. p. 1–8. https://doi.org/10.1109/IJCNN.2018.8489422.
46. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. In: Proceedings of the 27th international conference on neural information processing systems (NIPS'14); 2014. p. 2672–80.
47. Efimov D, Xu D, Kong L, Nefedov A, Anandakrishnan A. Using generative adversarial networks to synthesize artificial financial datasets; 2020. https://arxiv.org/abs/2002.02271.
48. Yuwana RS, Fauziah F, Heryana A, Krisnandi D, Kusumo RBS, Pardede HF. Data augmentation using adversarial networks for tea diseases detection. Jurnal Elektronika dan Telekomunikasi. 2020;20(1):29–35.
49. Benaim S, Wolf L. One-sided unsupervised domain mapping; 2017. https://arxiv.org/abs/1706.00826.
50. Kim T, Cha M, Kim H, Lee JK, Kim J. Learning to discover cross-domain relations with generative adversarial networks. In: Proceedings of the 34th international conference on machine learning, PMLR; 2017. p. 1857–65.
51. Ledig C, Theis L, Huszar F, Caballero J, Cunningham A, Acosta A, Aitken A, Tejani A, Totz J, Wang Z, Shi W. Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the 2017 IEEE conference on computer vision and pattern recognition (CVPR); 2017. p. 105–14. https://doi.org/10.1109/CVPR.2017.19.
52. Tulyakov S, Liu M, Yang X, Kautz J. MoCoGAN: Decomposing Motion and Content for Video Generation; 2017. https://arxiv.org/abs/1707.04993. https://doi.org/10.1109/CVPR.2018.00165.
53. Bousmalis K, Silberman N, Dohan S, Erhan D, Krishnan D. Unsupervised pixel-level domain adaptation with generative adversarial networks. In: Proceedings of the 2017 IEEE conference on computer vision and pattern recognition (CVPR); 2017. p. 95–104. https://doi.org/10.1109/CVPR.2017.18.
54. Hong Y, Hwang U, Yoo J, Yoon S. How generative adversarial networks and their variants work: an overview. ACM Comput Surv. 2019;52:10–11043. https://doi.org/10.1145/3301282.
55. Gonog L, Zhou Y. A review: generative adversarial networks. In: Proceedings of the 2019 14th IEEE conference on industrial electronics and applications (ICIEA); 2019. p. 505–10. https://doi.org/10.1109/ICIEA.2019.8833686.

56. Huang G, Liu Z, Maaten LVD, Weinberger KQ. Densely connected convolutional networks. In: Proceedings of the 2017 IEEE conference on computer vision and pattern recognition (CVPR); 2017. p. 2261–9. https://doi.org/10.1109/CVPR.2017.243.
57. Chollet F, et al. Keras. GitHub; 2015. https://github.com/fchollet/keras.
58. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X. TensorFlow: large-scale machine learning on heterogeneous systems. Software available from tensor ow.org; 2015. https://www.tensorflow.org/

Publisher's Note