International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Exploring the Generalization Capacity of Over-Parameterized Networks

[1]Eshan Pandey, [2]Santosh Kumar

Submitted: 21/10/2022 **Revised**: 18/12/2022 **Accepted**: 07/01/2023

Abstract: Most over-parameterized deep neural networks can generalize to true data. Even over-parameterized networks can learn randomly shuffled bytes in true data, which is far better than just random guessing. In such a scenario where, localized relations are lost to a huge extent, over-parameterized networks show generalization capability. In this paper, multiple models have been trained on various subsets and derivations of CIFAR10 data. Then the models have been trained on true images and true labels, randomly shuffled bytes, and true labels, ordered shuffled bytes and true labels, and randomly generated noise. Shuffling of bytes instead of pixels can distort the image to a great extent, yet over-parameterized models are able to generalize the data. After experimentation, it has been observed that generalization in true pixels is easier in comparison to randomly shuffled bytes data, and generalization in randomly shuffled bytes data is similar to ordered shuffled pixel data. However, memorization is easier and is not affected by any relationships in data.

Keywords: *Deep learning, over-parameterized deep neural networks, generalization, over-fitting, memorization, CIFAR-10, VGG, Manifold learning, PCA*

1. Introduction

The question of what can a deep model generalize and what it cannot was highlighted by [1, 2]. They demonstrated that over-parameterized neural networks have a huge capacity and can memorize an entire dataset. Most modern deep neural networks (where the number of parameters far exceeds the number of training samples) can fit random labels and/or complete random noise and reduce the training error to almost zero. The work in [1, 2, 3, 20] provides an experimental study on deep neural networks and suggests that optimization remains an easy process even when generalization is not possible, e.g., in cases such as where the labels are randomized. The work in this paper is hugely inspired by [1,2].

Multiple models on various subsets and derivations of CIFAR10 data have been trained. Models on true image and true labels randomly shuffled bytes and true labels, ordered shuffled bytes and true labels, and randomly generated noise were trained. Throughout the study, labels were never randomized. Randomizing labels shatter the relation in training and test data, and so there is no question of generalization in such cases. Doing this provides more distortion in the localized relation in a sample and through the dataset. Random shuffling implies the permutation of shuffling of bytes in an image is random and ordered shuffling implies a single permutation is applied to all the images in the dataset (test and train). Even in such settings, over-parameterized deep neural networks show some learning that is far better than just random guessing. In this paper, the following observations have been found:

i. Some data are easier to generalize than others; generalization in over-parameterized networks is not solely the property of architecture.
ii. The ease of generalization in over-parameterized convolutional networks depends on the semantic and localized relation in the dataset, but not completely. Generalization is the combined property of data and architecture (implicit regularization).
iii. Over-parameterized networks have a great tendency to generalize before completely over-fitting (this could be because of over-parameterization, implicit regularization, and high dimensional data being mapped to few classes, and so initial generalization is easy, as long as it is possible).
iv. However, memorization remains easy and is not affected by any existence of relation/meaning in data. Memorization in over-parameterized networks is solely the property of the architecture (remains fairly similar for data of comparable size). Explicit regularization can prevent memorization, but it is not guaranteed.
v. Over-parameterized networks are extremely powerful tools to extract features from the data. (This relation may or may not be semantic and/or local)

2. Related Works

Numerous studies try to provide a framework to control generalization error. VC dimensions [4], Rademacher complexity [5], uniform stability [6,7,8]. Numerous studies try

1 eshan.19m101004@abes.ac.in, 2santoshg25@gmail.com
1Department of Computer Science & Engineering, ABES Engineering College, Ghaziabad, U.P., India
2School of Computing Science and Engineering, Galgotias University, Greater Noida, U.P., India
** Corresponding Author Email: santoshg25@gmail.com*

to formulate some bounds on the network design and learnability of deep neural networks. It is known that a two-layer network can optimize the regularized loss to a global minimum in polynomial iterations, using noisy gradient descent, if it has infinite width [9]. The expressive power of the shallow network is less than that of a deep network [10]. The universal approximation theorem defines the upper bound on the approximation capability of a two-layer network. Any continuous and bounded function can be approximated by a two-layer network having nonlinear activation [11, 12, 13]. A two-layer network can optimize the regularized loss to a global minimum in polynomial iterations, using noisy gradient descent, if it has infinite width [14]. The problem with such theoretical bounds is that such upper bounds fail to capture any meaningful real-life application. Under such assumptions, hypothesis space is infinite, and so is the width of the network

.

SGD can optimize to global minima with zero classification error in polynomial time [15]. If the labels are true, SGD learns and generalizes, but when the labels are random, SGD finds a network that memorizes the data. Over-parameterized networks can improve generalization by leveraging larger sizes and more complex activations. The learned network distributes the information evenly among the neurons [16].

3. Dataset Description

Notations used:

Ɗ ($images_m^n$): Dataset of true images and true labels, having total class n and m samples/ class.
Ɗ ($images_m'^n$): Dataset of images with randomly shuffled bytes, having total class n and m samples/ class.
Ɗ ($image_m''^n$): Dataset of images with ordered shuffled bytes, having total class n and m samples/ class. The order of the shuffle remains the same for all images of the dataset.
N: Noise (Randomly generated meaningless data)

Description of datasets:

A three-channel image of dimension 32x32 has 1024 pixels and 3072 bytes. Shuffling of bytes instead of just shuffling of pixels further distorts/modifies the localized information as even the color information is lost/modified. The transformation of randomly shuffling the pixels/bytes is a lossy transformation (there is no way to retrieve the original image back easily), but the transformed dataset can also be treated as a standalone dataset with images and labels pairs. This assumption is crucial while interpreting the results obtained. Note that the ordered shuffled is not the lossy transformation, it only modifies the localized information.
Ɗ ($image_m^n$) →Ɗ ($image_m'^n$) and Ɗ ($image_m^n$) →Ɗ ($image_m''^n$) can be considered a data transformation. However, Ɗ ($images_m^n$), Ɗ ($images_m'^n$), Ɗ ($images_m''^n$) and Ɗ (N_m^n) can also be considered a separate dataset.

CIFAR-n data have been used for the experiments, where 10 <= n < 1 (1 not inclusive), n being the total number of classes in the dataset. CIFAR-n data was also shuffled bytes wise, in an ordered and unordered manner. Also, models were trained on randomly generated noise.
While training the various networks, 5000 training samples and 1000 test samples were used per class. For all other purposes, 1000 samples per class were considered for ease of computation and better visibility of the plots.

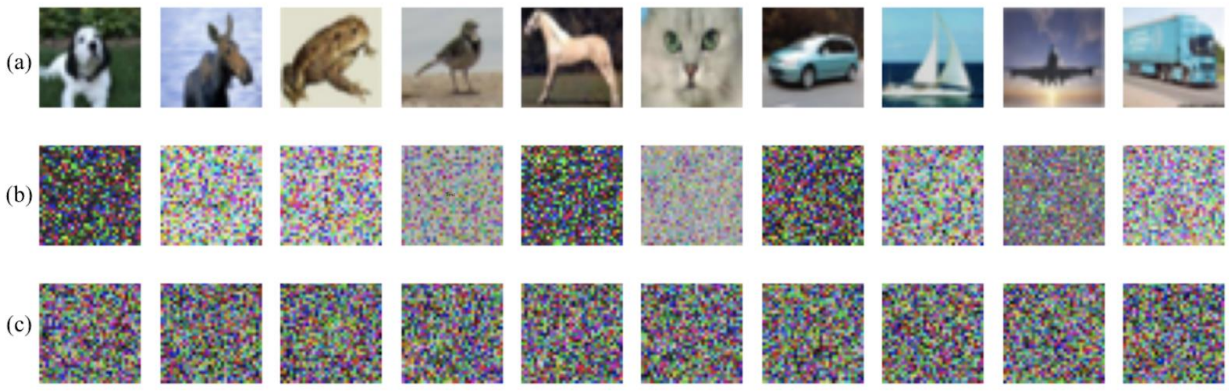
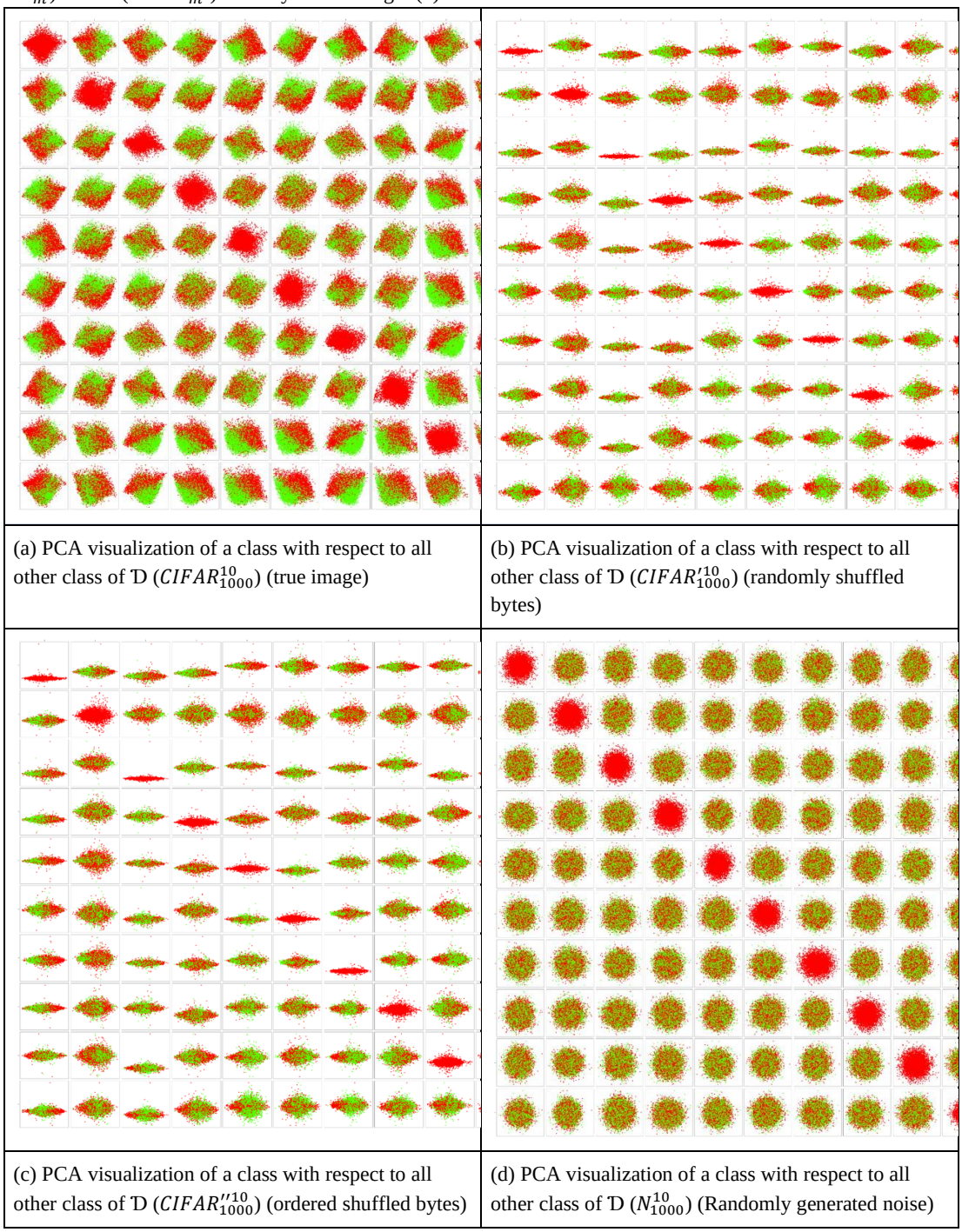


Fig 1: samples from dataset (a) Sample images from cifar10 (b) corresponding images from (a) with randomly shuffled bytes (c) randomly generated noise

The purpose of the experiment was to study the ability of various architectures to generalize and memorize the data. Under these circumstances, it also becomes important to understand the data and the effect of shuffling the bytes. The transition Ɗ ($CIFAR_m^n$) →Ɗ ($CIFAR_m'^n$) is not a lossy transformation as the transition Ɗ ($CIFAR_m'^n$) →Ɗ ($CIFAR_m^n$) is possible. The transition Ɗ ($CIFAR_m^n$) →Ɗ ($CIFAR_m''^n$) is a lossy transformation as the transformation Ɗ ($CIFAR_m''^n$) →Ɗ ($CIFAR_m^n$) is not possible. This is because in Ɗ ($CIFAR_m'^n$) bytes are shuffled in fixed permutation. In transition Ɗ ($CIFAR_m^n$) →Ɗ ($CIFAR_m'^n$) and Ɗ ($CIFAR_m^n$) →Ɗ ($CIFAR_m''^n$) Localized relation is distorted/modified.

Principal component analysis (PCA) decomposes a high dimensional dataset in a set of successive orthogonal components that explain a maximum amount of the variance.
PCA Visualizations can highlight important patterns in the dataset. Figure 2 plots PCA visualization of a class with respect to all others in the dataset [18]. Each block in the figure represents a pair of classes from an n-class dataset (except the diagonals, which represent only 1 single class). While for extremely high dimensional data, a 2D PCA plot may not be sufficient to explain the dataset, however, it may provide some visual information about the separability of classes. The clusters in Fig. 2(a) are more dispersed than the clusters in Fig 2(b), implying the samples of Ɗ ($CIFAR_m'^n$) are more similar

than Ɗ ($CIFAR_m^n$), and localized relations are lost in shuffling. Fig. 2(c) is remarkably similar to Fig. 2(b) implying the data sets Ɗ ($CIFAR_m^n$) and Ɗ ($CIFAR_m^{\prime n}$) are very similar. Fig. 2(d) represents evenly distributed samples that are randomly generated.



(a) PCA visualization of a class with respect to all other class of Ɗ ($CIFAR_{1000}^{10}$) (true image)

(b) PCA visualization of a class with respect to all other class of Ɗ ($CIFAR_{1000}^{\prime 10}$) (randomly shuffled bytes)

(c) PCA visualization of a class with respect to all other class of Ɗ ($CIFAR_{1000}^{\prime\prime 10}$) (ordered shuffled bytes)

(d) PCA visualization of a class with respect to all other class of Ɗ (N_{1000}^{10}) (Randomly generated noise)

Fig. 2. 2D PCA plot of the data Ɗ ($CIFAR_{1000}^{10}$), Ɗ ($CIFAR_{1000}^{\prime 10}$), Ɗ ($CIFAR_{1000}^{\prime\prime 10}$) and Ɗ ($N_{1000}^{10}$)

4.1 Data Visualization

To better understand the results that are obtained, it is important to understand the dataset that is used for training. The generalization achieved for each dataset is different. It is a challenge for us to measure the loss of information in the dataset achieved from shuffling in a meaningful manner. Mapping high dimensional data to a 2D plot may still not be a particularly good idea [32]; however, this might give some indication of the change in information the true data undergoes after random shuffling of bytes.

The figures in the section show various visualizations of the three-class subset of the data, Ɗ ($CIFAR_{1000}^{3}$), Ɗ ($CIFAR_{1000}^{\prime 3}$), and Ɗ (N_{1000}^{3}). A 3-class subset of the data has been selected for better visibility of the Figures. Figures 3-9 shows manifold embedding [19], Random projection, truncated SVD, isomap, standard LLE, modified LLE, MDS, random trees, spectral, t-SNE, and NCA embedding for 3 class data.

Random projection is an efficient dimensionality reduction technique, where the pairwise distance between any two samples is preserved [21, 22].

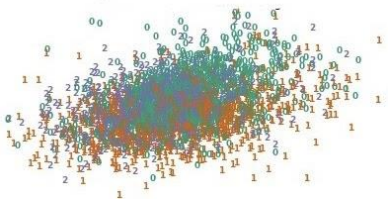
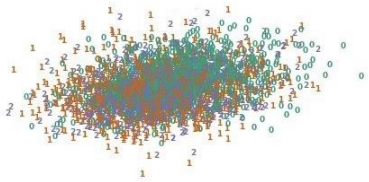
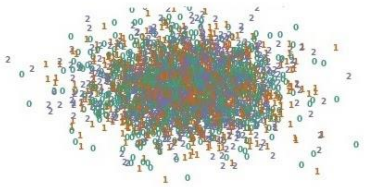
		
3(a) Ɗ ($CIFAR_{1000}^{3}$**)**	**3(b) Ɗ (**$CIFAR_{1000}'^{3}$**)**	**3(c)** Ɗ (N_{1000}^{3})

Fig. 3. Random projection embedding of the data sets. **(a)** Random projection embedding of 3 class true images **(b)** Random projection embedding of 3 class ordered shuffled bytes. **(c)** Random projection embedding of 3 class randomly generated noise.

Truncated singular value decomposition (Truncated SVD) is very similar to PCA and produces a low rank approximation of the data [23].

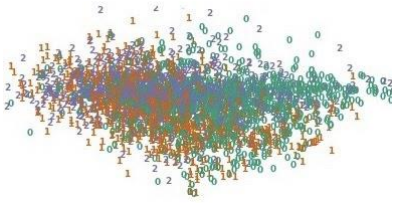
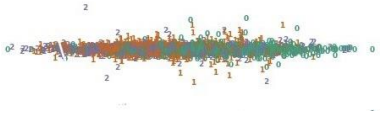
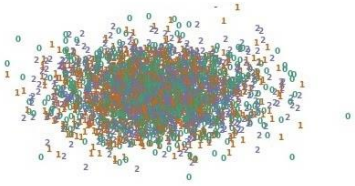
		
4(a) Ɗ ($CIFAR_{1000}^{3}$)	4(b) Ɗ ($CIFAR_{1000}'^{3}$)	4(c) Ɗ (N_{1000}^{3})

Fig. 4: truncated SVD embedding of the data sets. **(a)** truncated SVD embedding of 3 class true images **(b)** truncated SVD embedding of 3 class ordered shuffled bytes. **(c)** truncated SVD embedding of 3 class randomly generated noise.

Isometric Mapping seeks a lower-dimensional embedding that maintains geodesic distances between all points. It can be considered an extension of multidimensional scaling [24].

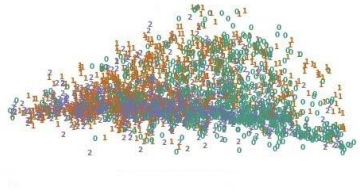
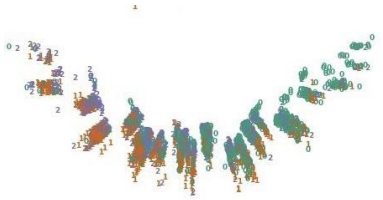
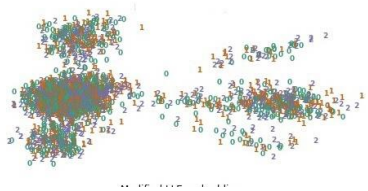
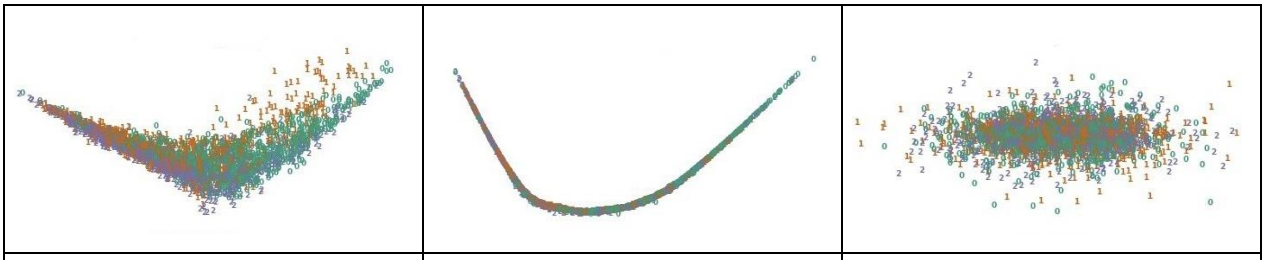
		
5(a) Ɗ ($CIFAR_{1000}^{3}$**)**	**5(b) Ɗ (**$CIFAR_{1000}'^{3}$**)**	**5(c)** Ɗ (N_{1000}^{3})

Fig. 5. Isomap embedding of the data sets. **(a)** Isomap embedding of 3 class true images **(b)** Isomap embedding of 3 class ordered shuffled bytes. **(c)** Isomap embedding of 3 class randomly generated noise.

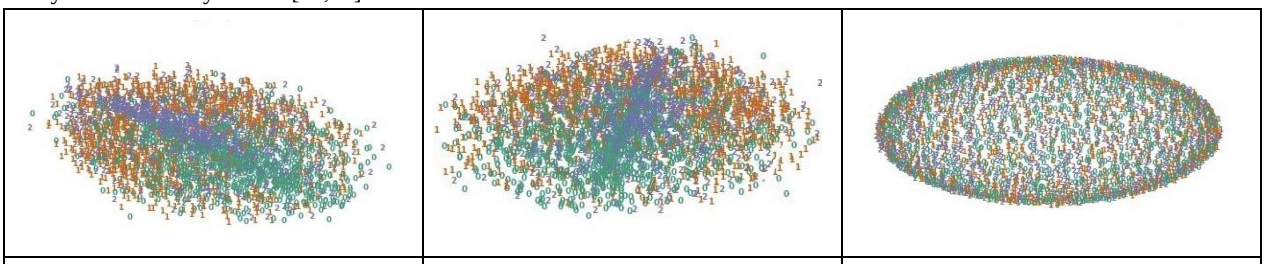
Locally linear embedding (LLE) is a dimensionality reduction technique; it preserves distances within local neighborhoods. It can be thought of as a series of local PCA which are globally compared to find the best nonlinear embedding [25].



6(a) Ɗ ($CIFAR^3_{1000}$)	**6(b) Ɗ (**$CIFAR'^3_{1000}$)	**6(c)** Ɗ (N^3_{1000})

Fig. 6. LLE embedding of the data sets. **(a)** LLE embedding of 3 class true images **(b)** LLE embedding of 3 class ordered shuffled bytes. **(c)** LLE embedding of 3 class randomly generated noise.

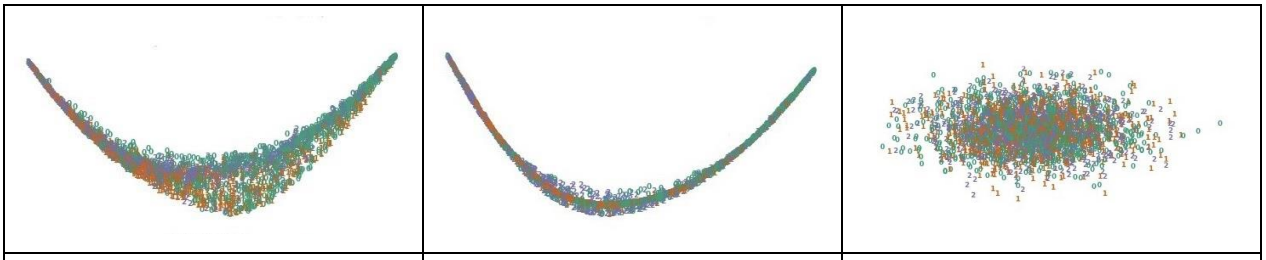
Multidimensional scaling (MDS) is a technique that creates mappings of items based on distance. It is used for analyzing similarity or dissimilarity in data [26,27].



7(a) Ɗ ($CIFAR^3_{1000}$)	**7(b) Ɗ (**$CIFAR'^3_{1000}$)	**7(c)** Ɗ (N^3_{1000})

Fig. 7. MDS embedding of the data sets. **(a)** MDS embedding of 3 class true images **(b)** MDS embedding of 3 class ordered shuffled bytes. **(c)** MDS embedding of 3 class randomly generated noise.

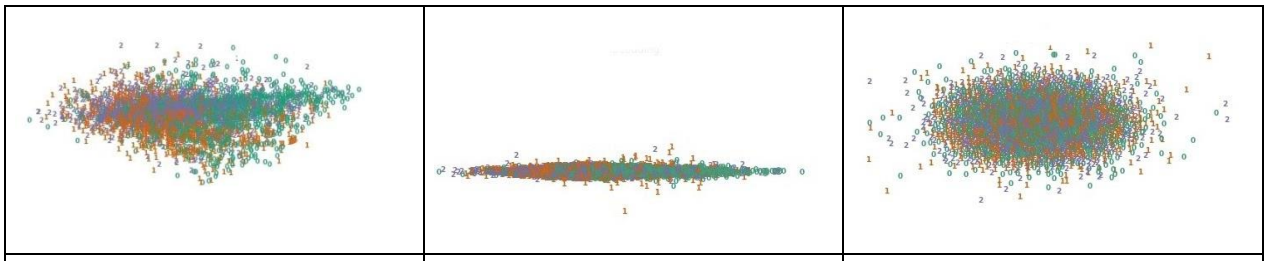
Spectral clustering results in a low-dimension embedding of the similarity matrix between the samples, followed by clustering [28,29,30,31].



8(a) Ɗ ($CIFAR^3_{1000}$)	**8(b) Ɗ (**$CIFAR'^3_{1000}$)	**8(c)** Ɗ (N^3_{1000})

Fig. 8. Spectral embedding of the data sets. **(a)** MDS embedding of 3 class true images **(b)** MDS embedding of 3 class ordered shuffled bytes. **(c)** MDS embedding of 3 class randomly generated noise.

T-distributed Stochastic Neighbor Embedding (t-SNE) is a technique to visualize high-dimensional data. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data.



9(a) $\mathcal{D}$ $(CIFAR^3_{1000})$ | **9(b)** $\mathcal{D}$ $(CIFAR'^3_{1000})$ | **9(c)** $\mathcal{D}$ (N^3_{1000})

Fig. 9. t-SNE embedding of the data sets. **(a)** t-SNE embedding of 3 class true images **(b)** t-SNE embedding of 3 class ordered shuffled bytes. **(c)** t-SNE embedding of 3 class randomly generated noise.

5. Architecture and Training

Four architectures VGG (modified) [17], small net, tNet, convNet have been trained for experimentation purposes. In the VGG network, the dense layer has been modified to 512 parameters. The VGG architecture contains no normalization and dropout. Small net has four convolutional layers and one dense layer. Small net has no regularization but has three dropouts. TNet is the largest network of all with a total of 67,716,666 trainable parameters. Normalization is used with each layer. No dropout and maxpool are used in tNet. ConvNet is a convolution network with convolutional layers along with batch normalization, dropout, and max pooling. In Fig. 10. shows four architectures used for the experimentation. Each architecture is trained on CIFAR-n data: true images, randomly shuffled images, and randomly generated noise, where 10 <= n < 1 and n is the total number of classes in the dataset.

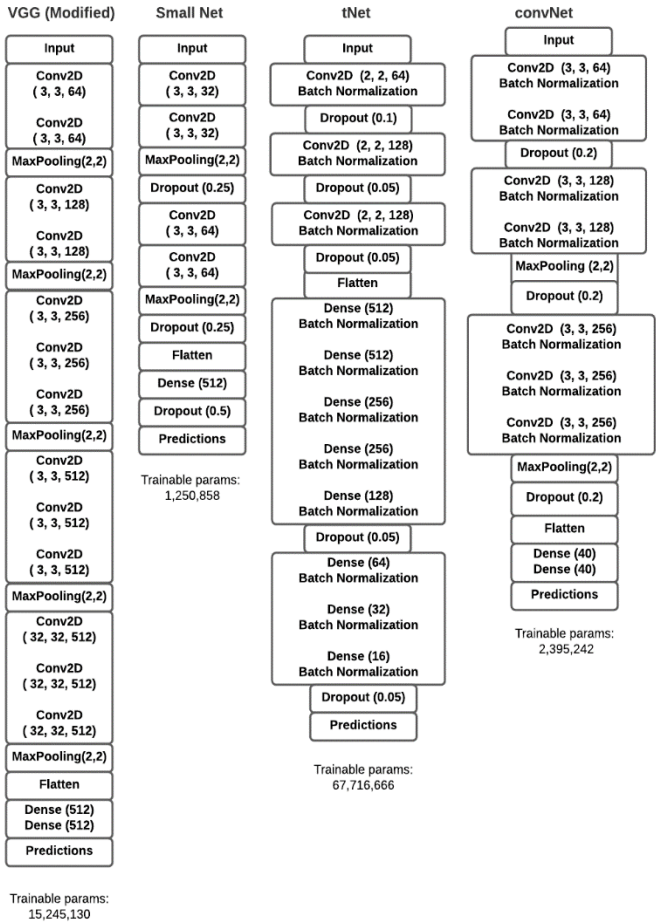


Fig. 10. VGG (modified), small net, tNet, convNet Architecture

Extraordinarily little to no extra efforts were made to train the model better. Each model was trained for 250 epochs. Categorical cross entropy for loss function. SGD and ReLU are used for VGG (modified), tNet and convNet Learning rate of 0.01 SGD and no decay while small net use RMSprop and ReLU for training with a learning rate of 0.0001, decay equals 1e-06. The batch size of 512 was used for training.

6. Performance Analysis of Models

Table 1 shows the best accuracy of VGG (modified), small net, tNet and convNet for the dataset Ɗ ($CIFAR^{n}_{5000}$), Ɗ ($CIFAR'^{n}_{5000}$) and Ɗ (N^{n}_{5000}). Ɗ (N^{n}_{5000}) is evenly and randomly generated noise, and there is nothing to be learned in sthe data. Ɗ (N^{n}_{5000}) is used to test the memorization ability of the model, and the test accuracy remains equal to just random guessing. E.g., for a 2-class classification task, just random guessing will result in 50% accuracy. *For*Ɗ ($CIFAR'^{n}_{5000}$) each model shows some learning greater than random guessing, implying some form of learning (initial generalization). Near-perfect train accuracy with low test accuracy implies memorization. Similar test and train accuracy mean good generalization. Models, when trained on Ɗ ($CIFAR^{n}_{5000}$), show good generalization. Models, when trained on Ɗ ($CIFAR'^{n}_{5000}$), show initial generalization before over-fitting.

Table 1: best train and test accuracy of VGG (modified), small net, tNet and convNet for the dataset Ɗ ($CIFAR^{n}_{5000}$), Ɗ ($CIFAR'^{n}_{5000}$) and Ɗ(*Nn5000)*.

Dataset	**Variant**	**Best accuracy**				
		VGG (Modified)	**Small Net**	**tNet**	**convNet**	**Random Guess**
CIFAR2	True Image	Train: 100% Test: 95.90%	Train: 99.67% Test: 97.05%	Train: 100% Test: 94.70%	Train: 100% Test: 97.05%	50.00%
	Shuffled Bytes	Train: 100% Test: 75.80%	Train: 92.77% Test: 74.80%	Train: 100% Test: 73.95%	Train: 99.87% Test:75.30%	
	Noise	Train: 100% Test: 50.00%	Train: 91.49 Test: 50.00%	Train: 100% Test: 50.00%	Train: 50.65 Test: 50.00%	
CIFAR3	True Image	Train: 100% Test: 91.53%	Train: 98.29% Test: 92.80%	Train: 100% Test: 88.27%	Train: 100% Test: 93.30%	33.33%
	Shuffled Bytes	Train: 100% Test: 57.43%	Train: 76.10% Test: 53.53%	Train: 99.99% Test: 54.83%	Train: 99.10% Test: 57.17%	
	Noise	Train: 100% Test: 33.33%	Train: 78.39% Test: 33.33%	Train: 99.99% Test: 33.33%	Train: 34.12% Test: 33.33%	
CIFAR4	True Image	Train: 100% Test: 84.70%	Train: 95.00% Test: 87.80%	Train: 99.98% Test: 82.07%	Train: 99.98% Test: 89.23%	25.00%
	Shuffled Bytes	Train: 100% Test: 42.72%	Train: 61.23% Test: 38.50%	Train: 99.97% Test: 40.83%	Train: 98.26% Test: 42.97%	
	Noise	Train: 100% Test: 25.00%	Train: 60.23% Test: 25.00%	Train: 99.97% Test: 25.00%	Train: 25.12% Test: 25.00%	
CIFAR5	True Image	Train: 100% Test: 81.32%	Train: 90.88% Test: 84.58%	Train: 99.98% Test: 76.10%	Train: 99.96% Test: 87.02%	20.00%

	Shuffled Bytes	Train: 100% Test: 37.12%	Train: 55.24% Test: 34.54%	Train: 99.96% Test: 35.66%	Train: 97.00% Test: 36.86%	
	Noise	Train: 100% Test: 20.00%	Train: 45.45 Test: 20.00%	Train: 99.95% Test: 20.00%	Train: 19.89% Test: 20.00%	
CIFAR 6	True Image	Train: 100% Test: 75.28%	Train: 86.44% Test: 81.15%	Train: 99.98% Test: 69.87%	Train: 99.92% Test: 84.08%	16.66%
	Shuffled Bytes	Train: 100% Test: 23.78%	Train: 45.40% Test: 28.85	Train: 99.95% Test: 28.35	Train: 97.34% Test: 31.33%	
	Noise	Train: 100% Test: 16.66%	Train: 41.74% Test: 16.66%	Train: 99.96% Test: 16.66%	Train 16.84% Test: 16.66%s	
CIFAR7	True Image	Train: 100% Test: 75.64%	Train: 85.5% Test: 80.54%	Train: 99.97% Test: 69.23%	Train: 99.93% Test: 84.59%	14.28%
	Shuffled Bytes	Train: 99.85% Test: 27.19%	Train: 37.83% Test: 25.59%	Train: 99.96% Test: 23.79%	Train: 97.11% Test: 27.04%	
	Noise	Train: 99.87% Test: 14.28%	Train: 36.43% Test: 14.28%	Train: 99.87% Test: 14.28%	Train: 14.14 Test: 14.28%	
CIFAR8	True Image	Train: 100% Test: 74.00%	Train: 82.69% Test: 78.95%	Train: 99.97% Test: 68.59	Train: 99.98% Test: 85.05%	12.50%
	Shuffled Bytes	Train: 99.74% Test: 24.28%	Train: 29.33% Test: 21.30%	Train: 99.96% Test: 23.39%	Train: 97.35 Test: 24.01	
	Noise	Train: 99.81 Test: 12.50%	Train: 27.12 Test: 12.50%	Train: 99.95% Test:12.50%	Train: 12.37% Test: 12.50%	
CIFAR9	True Image	Train: 100% Test: 75.80%	Train: 83.16% Test: 79.90%	Train: 99.97% Test: 69.16%	Train: 99.86 Test: 85.13%	11.11%
	Shuffled Bytes	Train: 99.61% Test: 21.48%	Train: 32.08% Test: 19.40%	Train 99.95%: Test: 20.47%	Train: 96.55%: Test: 21.06%	
	Noise	Train: 99.86% Test: 10%	Train: 24.10% Test:10%	Train: 99.94% Test:10%	Train: 11.06 Test:10%	

CIFAR10	True Image	Train: 100% Test: 74.94%	Train: 79.79% Test: 78.28%	Train: 99.96% Test: 68.76%	Train: 99.90% Test: 85.35%	10.00%
	Shuffled Bytes	Train: 99.64% Test: 20.85%	Train: 27.68% Test: 18.15%	Train: 99.94% Test: 19.27	Train: 96.69% Test: 20.67%	
	Noise	Train: 99.77% Test:10%	Train: 20.95% Test: 10%	Train: 99.87% Test: 10%	Train: 9.98% Test: 10%	

6.2 Training Plots

The test/train and loss plotted against epoch maintain a similar trend for the number of classes, as shown in Fig. 11, 12, 13, and 14.

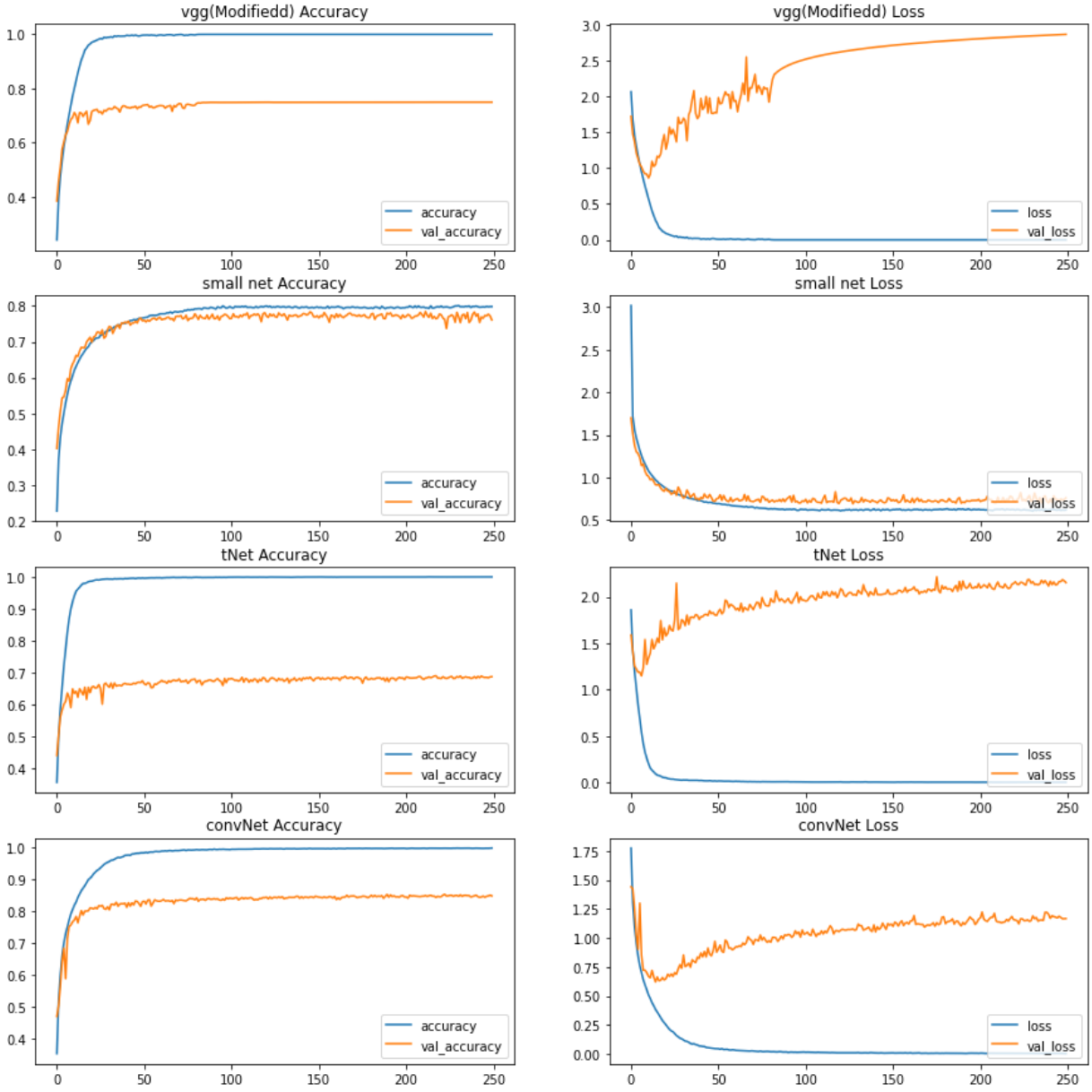


Fig. 11: Accuracy and Loss of VGG (Modified), small net, tNet and convNet trained on CIFAR10 true data

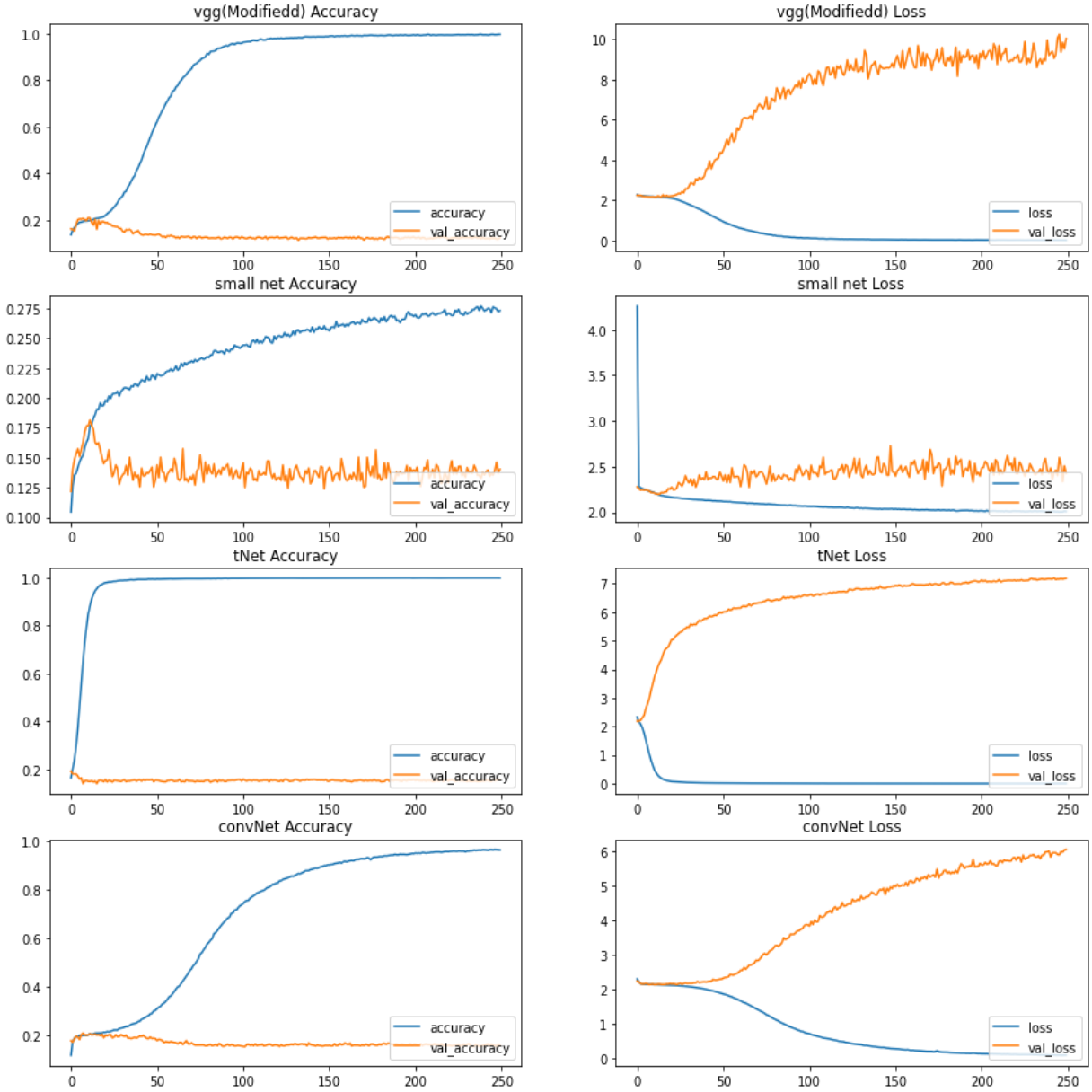


Fig 12: Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on CIFAR10 randomly shuffled bytes.

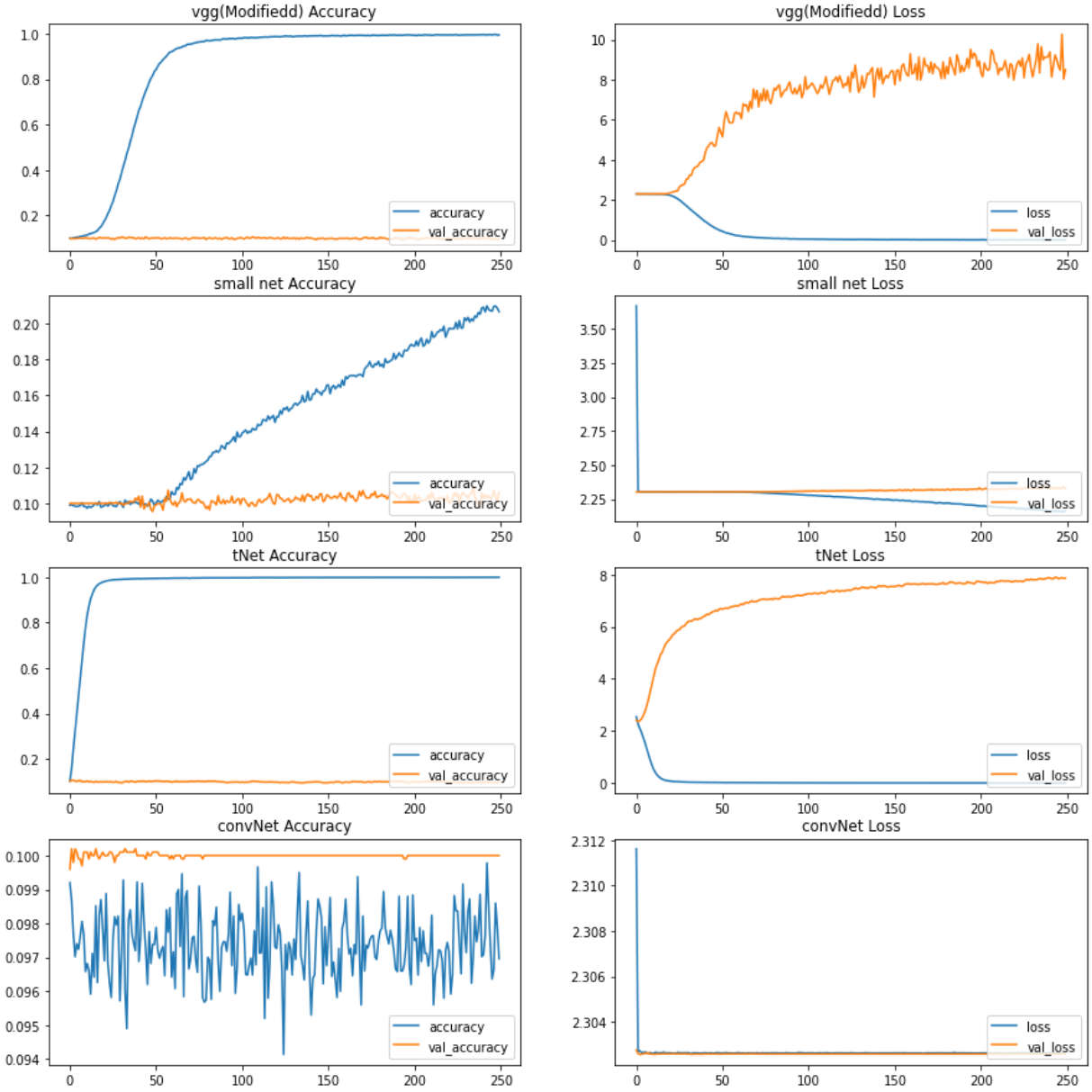
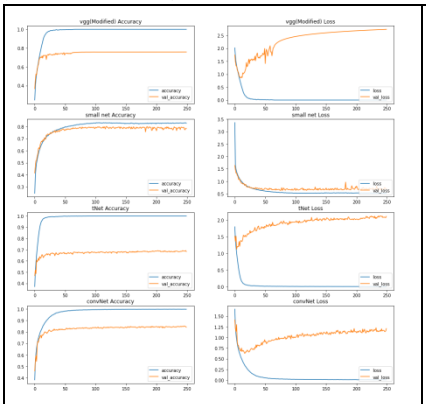
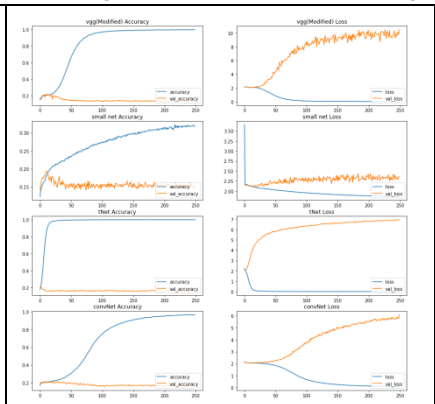
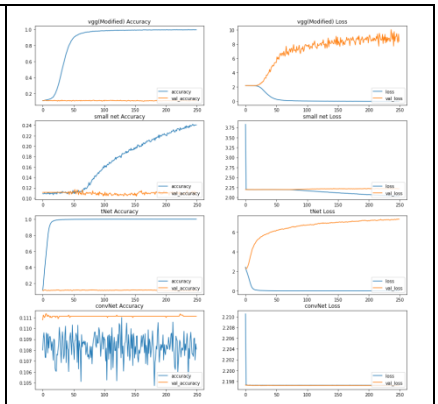
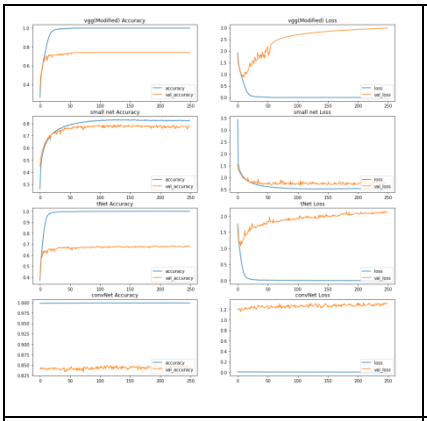
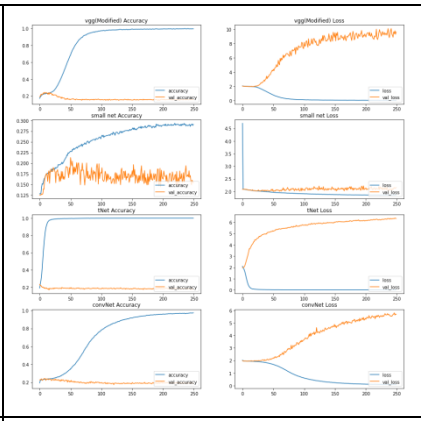
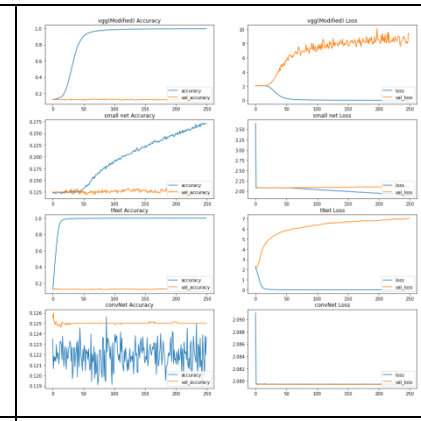
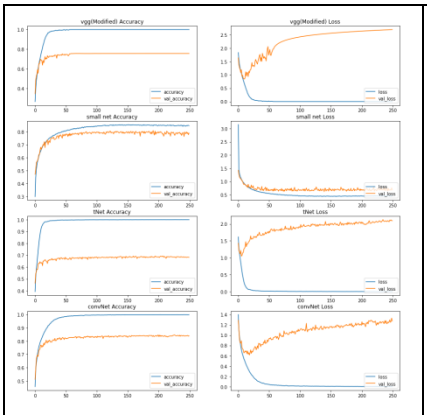
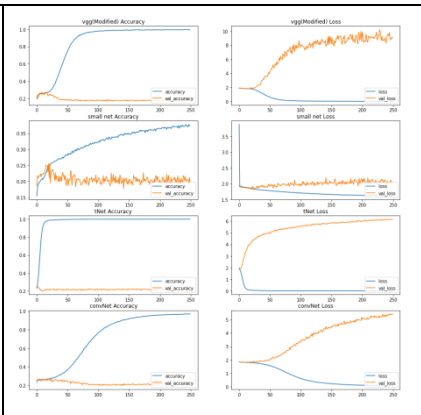
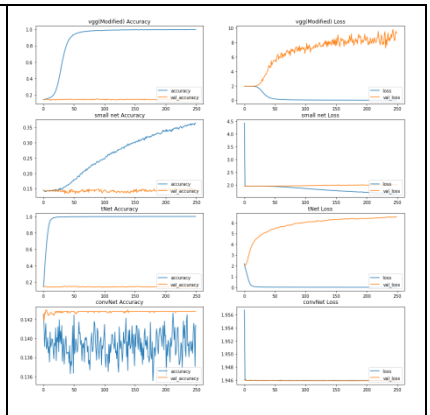
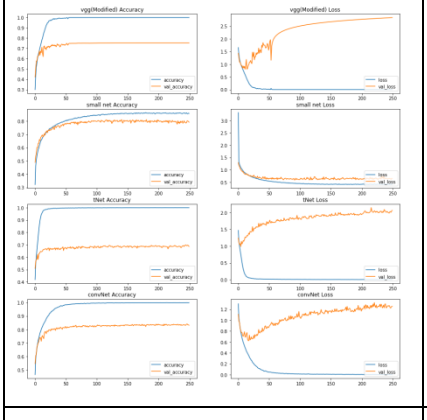
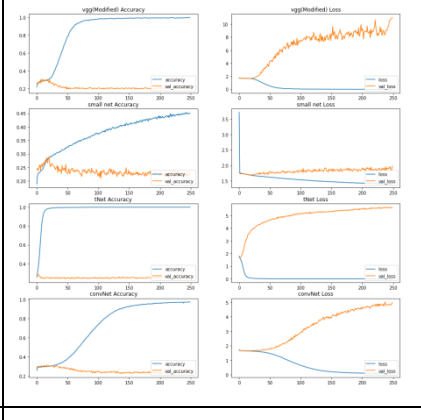
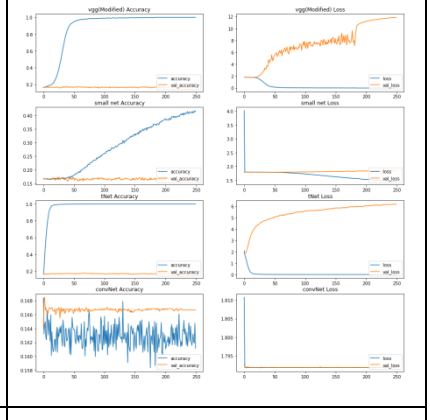
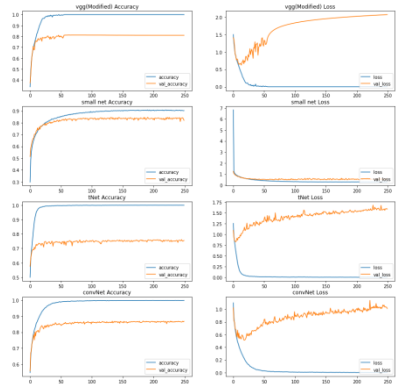
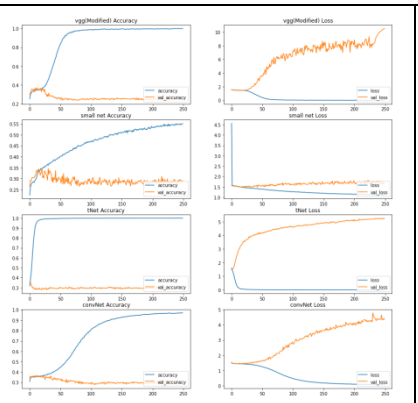
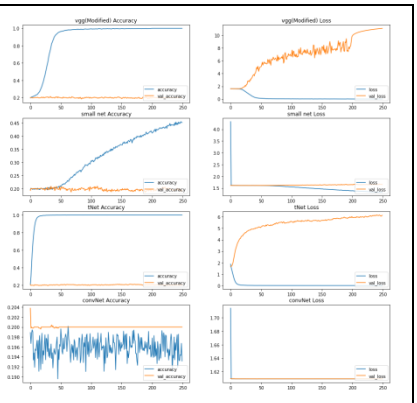
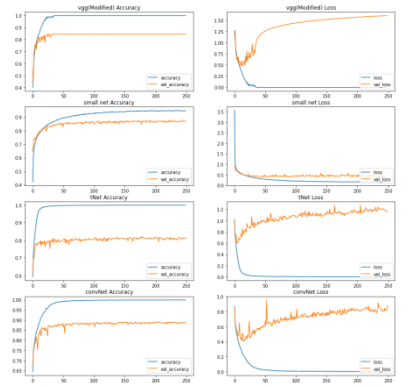
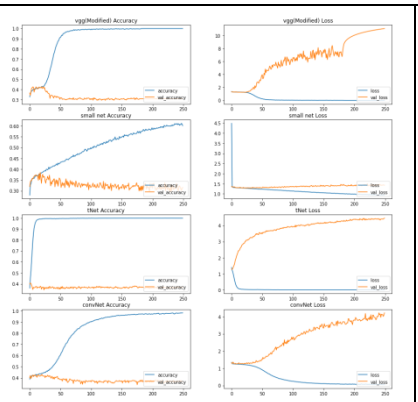
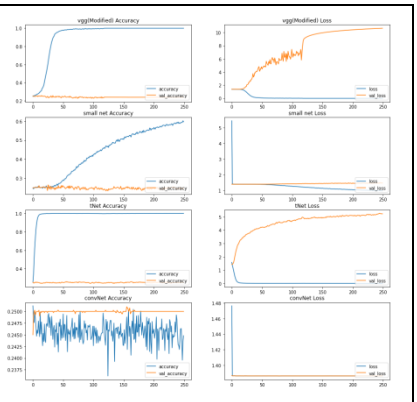
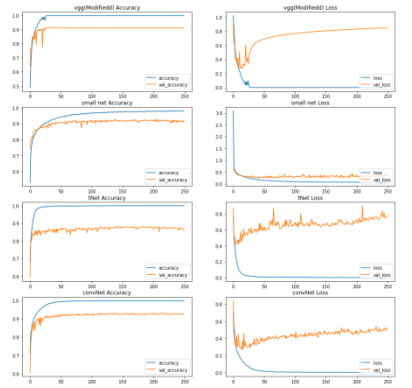
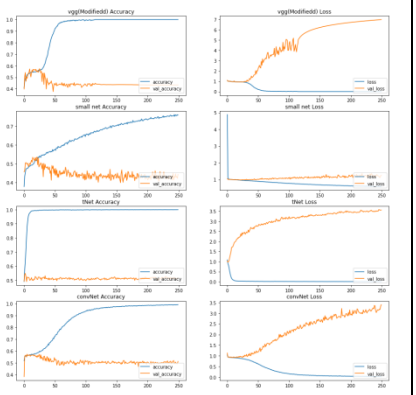
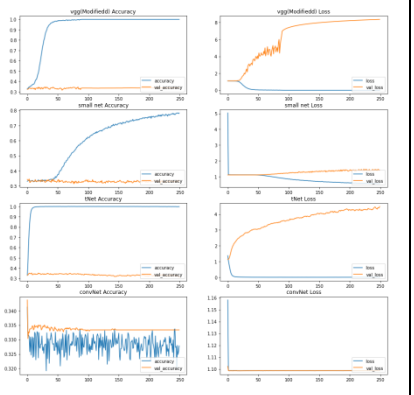


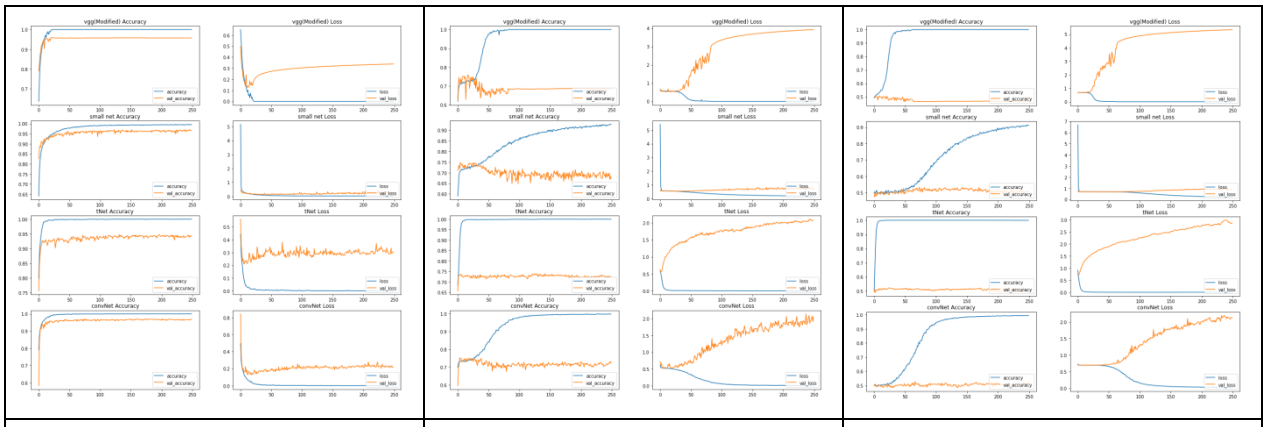
Fig 13: Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on noise, size = CIFAR10

Experiment has also been performed for datasets CIFAR-n, (n=9,8,7,6,5,4,3,2). And it has been seen that training and test plots of accuracy and loss are similar to the 10-class dataset as shown in Fig. 18.

		
Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on CIFAR 9 true data	Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on CIFAR 9 randomly shuffled Bytes.	Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on Noise, size = CIFAR 9

		
Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on CIFAR 8 true data	Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on CIFAR 8 randomly shuffled Bytes.	Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on noise, size = CIFAR 8
		
Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on CIFAR 7 true data	Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on CIFAR 7 randomly shuffled Bytes.	Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on noise, size = CIFAR 7
		
Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on CIFAR 6 true data	Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on CIFAR 6 randomly shuffled Bytes.	Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on noise, size = CIFAR 6

		
Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on CIFAR 5 true data	Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on CIFAR 5 randomly shuffled Bytes.	Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on noise, size = CIFAR 5
		
Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on CIFAR 4 true data	Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on CIFAR 4 randomly shuffled Bytes.	Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on noise, size = CIFAR 4
		
Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on CIFAR 3 true data	Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on CIFAR 3 randomly shuffled Bytes.	Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on noise, size = CIFAR 3



Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on CIFAR 2 true data	Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on CIFAR 2 randomly shuffled Bytes.	Accuracy and Loss of VGG(Modified), small net, tNet and cconvNet trained on noise, size = CIFAR 2

Fig. 14: Train and test accuracy and loss for datasets CIFAR-n, (n=9,8,7,6,5,4,3,2)

It can be observed from Fig. 17 and Fig. 18 that over-parameterized networks can generalize to true data easily, as in the case with CIFAR-n data. In the case of randomly shuffled bytes, over-parameterized networks show accuracy which is greater than just mere guessing, implying there is some form of learning that takes place. Considering randomly shuffled bytes of CIFAR-n dataset, a separate dataset CIFAR-n' and ordered shuffle bytes of CIFAR-n dataset another separate dataset CIFAR-n", It is observed that it is easier to generalize CIFAR-n than CIFAR-n' and CIFAR-n". The trainability and learnability of CIFAR-n' and CIFAR-n" remain similar. It is in this regard that most theoretical bounds do not capture the true generalization ability of deep neural networks.

An interesting observation is that results for randomly shuffled bytes and ordered shuffle bytes remained the same. This could be because CNNs extract localized information. Once the localized information is lost (or distorted), over-parameterized networks do not capture the difference between randomly shuffled and ordered shuffled bytes. This could indicate that over-parameterized networks tend to learn a broader relation in data than just simply (over) fitting the training data pairs.

If pixel randomization is considered as a data transformation problem, ordered pixel shuffling and random pixel shuffling are two different transformations. But this data transformation is supplemented by loss in information (distortion in localized semantic relation). This implies localized semantic relation in data is of vital importance in the generalization of over-parameterized networks. However, even with complete distortion in localized information, networks capture some relation and show some learning, implying that localized information is not all that deep neural networks look for while learning.

Learning in shuffled bytes datasets shows that the over-parameterized deep networks are great tools for feature extraction. It can also be observed that memorization remains an easy task across architectures and datasets. However, convNet fails to memorize the randomly generated noise each time. But convNet achieves best performance for true images and at par results for shuffled bytes. By increasing the size of a fully connected layer or reducing the explicit regularization, by either approach convNet could memorize the randomly generated noise. ConvNet is the only architecture in the experiment with heavy explicit regularization and smaller fully connected layers. Using explicit regularization may help in not memorization but this is not necessary, as by just increasing the size of fully connected layers, the model could overcome the effect of regularization.

7. Conclusion

It is observed that generalization in Ɗ ($CIFAR^n$) is easier in comparison to Ɗ ($CIFAR'^n$) and generalization in Ɗ ($CIFAR'^n$) is similar to Ɗ ($CIFAR''^n$). If one considers Ɗ ($CIFAR^n$), Ɗ ($CIFAR'^n$), Ɗ ($CIFAR''^n$) a separate dataset, it can be concluded that some data can be generalized easier than others. Thus, generalization is the combined property of data and model.

Optimization remains easy even under scenarios where generalization is not possible [1-2]. The work presented in this paper concludes that generalization remains a possibility if there is some consistent relationship between train and test data (images and labels). Considering transformation Ɗ ($CIFAR^n$) →Ɗ ($CIFAR'^n$) and Ɗ ($CIFAR^n$) →Ɗ ($CIFAR''^n$), even after distortion/modification in localized relation, all models show some learning for Ɗ ($CIFAR'^n$) and Ɗ ($CIFAR''^n$). This implies that, even after complete distortion in localized relation in the data, over-parameterized deep networks can extract some information and show some learning. Over-parameterized deep networks have enormous potential for generalization.

It has been observed that convNet failed to memorize the randomly generated data, but by increasing the parameters or by decreasing the explicit regularization, convNet could fit the randomly generated noise, implying explicit regularization may work against over-fitting, but this is not guaranteed.

8. Future Possibilities

Studying generalization as a combined property of architecture and data could help develop better deep learning models. Most theoretical bounds only capture the extreme possibility of the data or the model in practical applications; this is not a

sufficient measure of the possible results. While localized information in data plays a vital role in the generalization of convolutional neural networks, this is not all the information that a network learns. A better understanding of what an over-parameterized network learns is to be developed.

It is difficult to say for sure when and how much effect various explicit regularization techniques have on generalization and memorization. Such design decisions are mostly hit and trail; this points to the lack of very fundamental understanding of the over-parameterized networks and regularization techniques.

Is it possible to develop a framework to understand what can and cannot be generalized? It remains a question for us, how much information is lost in the transformation from true pixels to shuffled bytes, and what is the best generalization that is possible against the loss of information due to this transformation.

Even after a huge distortion in information, over-parameterized networks can learn the data to some extent. This knowledge can be used to develop a method of training models that are more resilient against adversarial attacks.

Author contributions

Eshan Pandey: Conceptualization, Methodology, Experimentation **Santosh Kumar:** Writing, Validation, Reviewing and Editing.

Declaration of competing interest The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Zhang, Chiyuan & Bengio, Samy & Hardt, Moritz & Recht, Benjamin & Vinyals, Oriol. (2016). Understanding deep learning requires rethinking generalization. Communications of the ACM. 64. 10.1145/3446776.

[2] Zhang, Chiyuan, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. "Understanding deep learning (still) requires rethinking generalization." Communications of the ACM 64, no. 3 (2021): 107-115.

[3] Allen-Zhu, Zeyuan, Yuanzhi Li, and Zhao Song. "A convergence theory for deep learning via over-parameterization." In International Conference on Machine Learning, pp. 242-252. PMLR, 2019.

[4] Vapnik, Vladimir Naumovich. "Adaptive and learning systems for signal processing communications, and control." Statistical learning theory (1998).

[5] Bartlett, Peter L., and Shahar Mendelson. "Rademacher and Gaussian complexities: Risk bounds and structural results." Journal of Machine Learning Research 3, no. Nov (2002): 463-482.

[6] Mukherjee, Sayan, Partha Niyogi, Tomaso Poggio, and Ryan Rifkin. "Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization." Advances in Computational Mathematics 25, no. 1 (2006): 161-193.

[7] Bousquet, Olivier, and André Elisseeff. "Stability and generalization." The Journal of Machine Learning Research 2 (2002): 499-526.

[8] Poggio, Tomaso, Ryan Rifkin, Sayan Mukherjee, and Partha Niyogi. "General conditions for predictivity in learning theory." Nature 428, no. 6981 (2004): 419-422.

[9] Wei, Colin, Jason Lee, Qiang Liu, and Tengyu Ma. "Regularization matters: Generalization and optimization of neural nets vs their induced kernel." (2019).

[10] Delalleau, Olivier, and Yoshua Bengio. "Shallow vs. deep sum-product networks." Advances in neural information processing systems 24 (2011): 666-674.

[11] Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." Neural networks 2, no. 5 (1989): 359-366.

[12] Funahashi, Ken-Ichi. "On the approximate realization of continuous mappings by neural networks." Neural networks 2, no. 3 (1989): 183-192.

[13] Barron, Andrew R. "Approximation and estimation bounds for artificial neural networks." Machine learning 14, no. 1 (1994): 115-133.

[14] Wei, Colin, Jason Lee, Qiang Liu, and Tengyu Ma. "Regularization matters: Generalization and optimization of neural nets vs their induced kernel." (2019).

[15] Allen-Zhu, Zeyuan, Yuanzhi Li, and Zhao Song. "A convergence theory for deep learning via over-parameterization." In International Conference on Machine Learning, pp. 242-252. PMLR, 2019.

[16] Allen-Zhu, Zeyuan, Yuanzhi Li, and Yingyu Liang. "Learning and generalization in overparameterized neural networks, going beyond two layers." arXiv preprint arXiv:1811.04918 (2018).

[17] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[18] Mueller, A. (n.d.). Another look at MNIST. Https://Peekaboo-Vision. Retrieved August 1, 2021, from https://peekaboo-vision.blogspot.com/2012/12/another-look-at-mnist.html

[19] Pedregosa, F., Grisel, O., Blondel, M., & Varoquaux, G. (n.d.). Manifold learning on handwritten digits: Locally Linear Embedding, Isomap. . ¶. Https://Scikit-Learn.Org/. Retrieved August 1, 2020, from https://scikit-learn.org/0.15/auto_examples/manifold/plot_lle_digits.html#example-manifold-plot-lle-digits-py

[20] H. Salehinejad, S. Valaee, T. Dowdell, E. Colak and J. Barfett, "Generalization of Deep Neural Networks for Chest Pathology Classification in X-Rays Using Generative Adversarial Networks," 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 990-994, doi: 10.1109/ICASSP.2018.8461430.

[21] Sanjoy Dasgupta. 2000. Experiments with random projection. In Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence (UAI'00), Craig Boutilier and Moisés Goldszmidt (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 143-151.

[22] Ella Bingham and Heikki Mannila. 2001. Random projection in dimensionality reduction: applications to image and text data. In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '01). ACM, New York, NY, USA, 245-250.

[23] Halko, N., Martinsson, P. G., & Tropp, J. A. (2011). Finding structure with randomness: Probabilistic

algorithms for constructing approximate matrix decompositions. SIAM review, 53(2), 217-288.
[24] Tenenbaum, J.B.; De Silva, V.; & Langford, J.C. A global geometric framework for nonlinear dimensionality reduction. Science 290 (5500)
[25] "Nonlinear dimensionality reduction by locally linear embedding" Roweis, S. & Saul, L. Science 290:2323 (2000)
[26] Borg, I., & Groenen, P. J. (2005). Modern multidimensional scaling: Theory and applications. Springer Science & Business Media.
[27] Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika, 29(1), 1-27.
[28] Ng, A., Jordan, M., & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. Advances in neural information processing systems, 14.
[29] Zhuzhunashvili, D., & Knyazev, A. (2017, September). Preconditioned spectral clustering for stochastic block partition streaming graph challenge (preliminary version at arxiv.). In 2017 IEEE High Performance Extreme Computing Conference (HPEC) (pp. 1-6). IEEE.
[30] Meilă, M., & Shi, J. (2001, January). A random walks view of spectral segmentation. In International Workshop on Artificial Intelligence and Statistics (pp. 203-208). PMLR.
[31] Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. IEEE Transactions on pattern analysis and machine intelligence, 22(8), 888-905.
[32] Zhanxuan, Hu & Wu, Danyang & Nie, Feiping & Wang, Rong. (2021). Generalization Bottleneck in Deep Metric Learning. Information Sciences. 581. 10.1016/j.ins.2021.09.023.