Pseudo Rehearsal using non photo-realistic images

Suri Bhasker Sri Harsha,
Dr. Yeturu Kalidas.

Computer Science Department
Indian Institute of Technology Tirupati

Catastrophic forgetting problem

R. M. French, “Catastrophic forgetting in connectionist networks,”Trends in cognitive sciences, vol. 3, no. 4, pp. 128–135, 1999.

Continual Learning

[1] H. Shin, J. K. Lee, J. Kim, and J. Kim, “Continual learning with deep generative replay,” in Advances in Neural Information Processing Systems, pp. 2990–2999, 2017
[2] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, “Continual lifelong learning with neural networks: A review,”Neural Networks,2019.
[3] C. Atkinson, B. McCane, L. Szymanski, and A. Robins, “Pseudo-recursal: Solving the catastrophic forgetting problem in deep neural networks,”arXiv preprint arXiv:1802.03875, 2018.

Changing weights leads to *distorted* decision boundary

“Distortion of the decision boundary leads to forgetting.”

Pseudo Rehearsal ...

PSEUDO REHEARSAL

$M_t = \text{Train}\ (B_1{*} \cup B_2{*} \cup B_3{*} \cup \ldots B_{t-1}{*} \cup B_t, M_{t-1})$

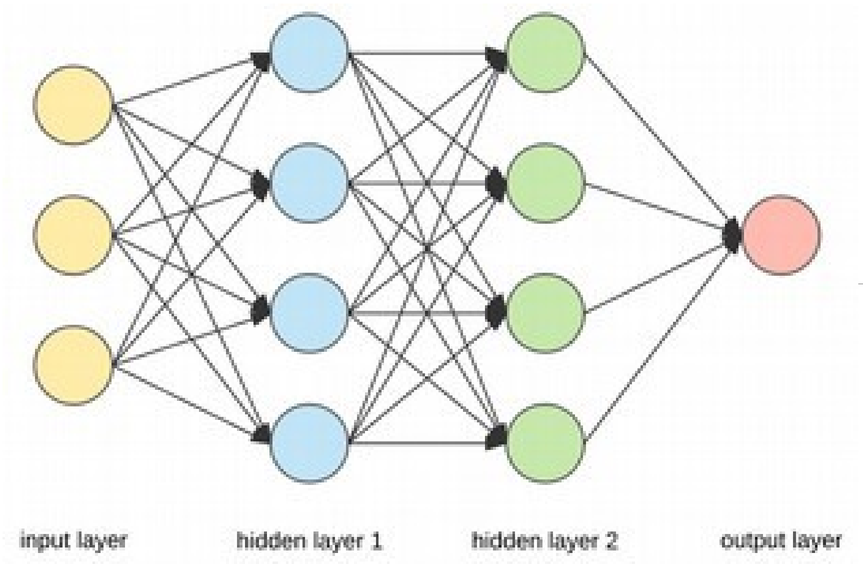
here, B_i is the training data for the task "T_i".

and $B_i{*} = G(B_i)$

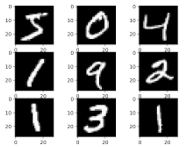
where B_i^* is the synthetic version of B_i , generated using the data generator G(). M_i represents the neural network being trained on task "T_i".

Ref: A. Robins, "Catastrophic forgetting, rehearsal and pseudorehearsal," Connection Science, vol. 7, no. 2, pp. 123–146, 1995.

GENERATIVE REPLAY

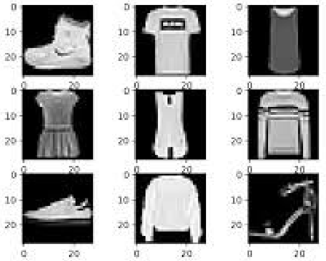


Generative Adversarial Network



MNIST Digits

MNIST Fashion



Reference: H. Shin, J. K. Lee, J. Kim, and J. Kim, “Continual learning with deep generative replay,” in Advances in Neural Information Processing Systems, pp. 2990–2999, 2017.

Visually complex images

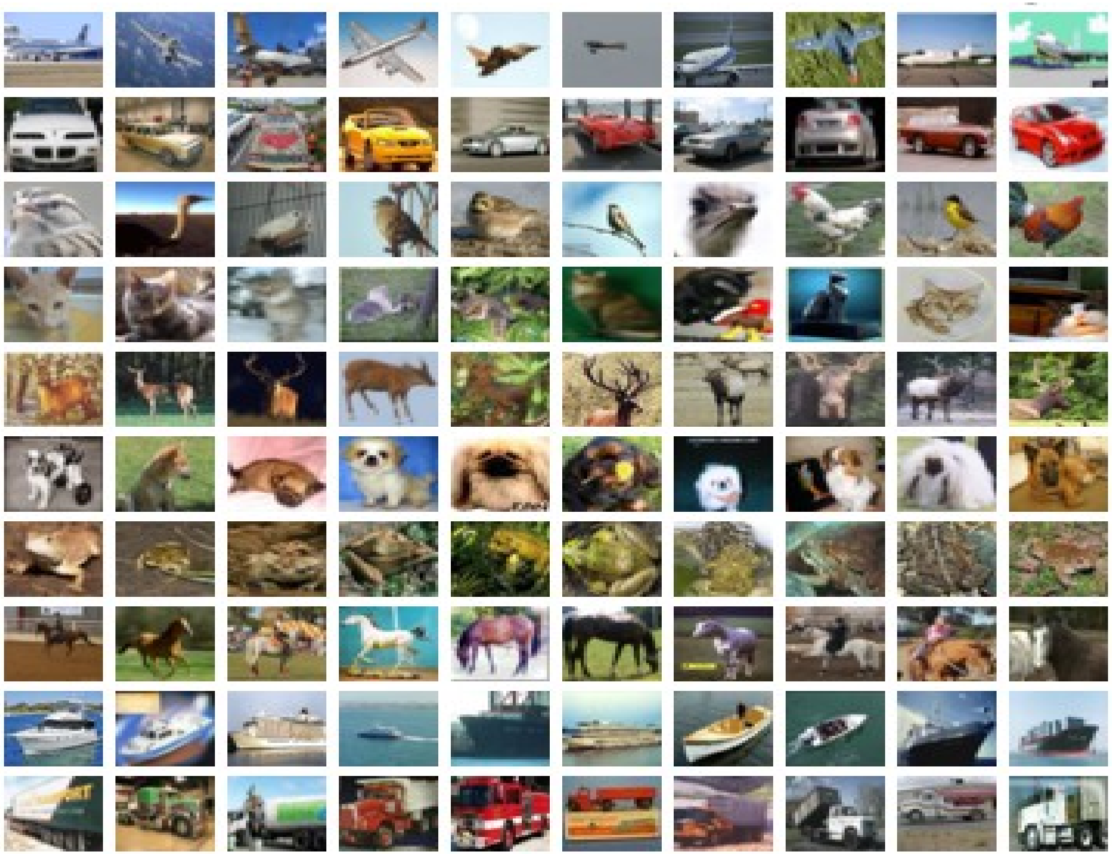
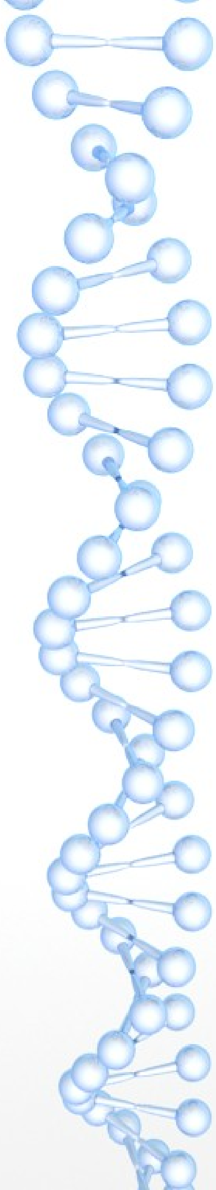


Image source:
Google image search



Genetic Algorithms

Difference in loss functions ...

GAN based approaches	Ours
minmax(Generator, Discriminator) + binary cross entrophy	Genetic Algorithm + Softmax confidence of target class

Our approach ...

Our approach ...

“ *Instead of trying to generate photo realistic images*, we try to generate images, which when trained upon have the ability to preserve the boundary that is responsible for retention of the previous task.”

Our approach ...

“ Instead of trying to generate photo realistic images, we try to generate images, which when trained upon have the ability to *preserve the boundary* that is responsible for retention of the previous task.”

Algorithm 1 Synthetic data generation for a class c

P: $\{x_0, x_1, x_2 ... x_{m-1}\}$ // Random population
$|P| = m$
while $(\exists x \in P) : f_c(x) \leq \tau$ **do**
  $P^{'} = \{< x, f_c(x) > |\forall x \in P\}$
  Let, L be a list in descending order of $f_c(x), \forall x \in P^{'}$
  $P^{*} = L[0 \ldots m * 0.25]$ //top 25% of elements
  $C = [crossover(P^{*}[j], P^{*}[j+1])|\forall j \in [0 \ldots |P^{*}| - 1]]$
  $M = [mutation(x)|\forall x \in P^{*}]$
  $M_C = [crossover(M[i], M[i+1])|\forall i \in [0, \ldots, |M| - 1]]$
  $P_{new} = P^{*} \cup C \cup M \cup M_C$
  $P = P_{new}$
end while

Here $f_c(x) = \frac{e^{z_c}}{\sum_{j=1}^{K} e^{z_j}}$
where z is vector of scores for each of the classes $1 \ldots K$, c is the given target class, $f_c(x)$ is softmax score for class c on input x and K is the total number of classes.

Algorithm 1 Synthetic data generation for a class c

P: $\{x_0, x_1, x_2 ... x_{m-1}\}$ // Random population
$|P| = m$
while $(\exists x \in P) : f_c(x) \leq \tau$ **do**
  $P^{'} = \{< x, f_c(x) > |\forall x \in P\}$
  Let, L be a list in descending order of $f_c(x), \forall x \in P^{'}$
  $P^{*} = L[0 \dots m * 0.25]$ //top 25% of elements
  $C = [crossover(P^{*}[j], P^{*}[j+1])|\forall j \in [0 \dots |P^{*}| - 1]]$
  $M = [mutation(x)|\forall x \in P^{*}]$
  $M_C = [crossover(M[i], M[i+1])|\forall i \in [0, \dots, |M| - 1]]$
  $P_{new} = P^{*} \cup C \cup M \cup M_C$
  $P = P_{new}$
end while

Here $f_c(x) = \frac{e^{z_c}}{\sum_{j=1}^{K} e^{z_j}}$
where z is vector of scores for each of the classes $1 \dots K$, c is the given target class, $f_c(x)$ is softmax score for class c on input x and K is the total number of classes.

Algorithm 1 Synthetic data generation for a class c

P: $\{x_0, x_1, x_2 ... x_{m-1}\}$ // Random population

$|P| = m$

while $(\exists x \in P) : f_c(x) \leq \tau$ **do**

  $P^{'} = \{< x, f_c(x) > | \forall x \in P\}$

  Let, L be a list in descending order of $f_c(x), \forall x \in P^{'}$

  $P^{*} = L[0 \ldots m * 0.25]$ //top 25% of elements

  $C = [crossover(P^{*}[j], P^{*}[j+1]) | \forall j \in [0 \ldots |P^{*}| - 1]]$

  $M = [mutation(x) | \forall x \in P^{*}]$

  $M_C = [crossover(M[i], M[i+1]) | \forall i \in [0, \ldots, |M| - 1]]$

  $P_{new} = P^{*} \cup C \cup M \cup M_C$

  $P = P_{new}$

end while

Here $f_c(x) = \frac{e^{z_c}}{\sum_{j=1}^{K} e^{z_j}}$

where z is vector of scores for each of the classes $1 \ldots K$, c is the given target class, $f_c(x)$ is softmax score for class c on input x and K is the total number of classes.

Algorithm 1 Synthetic data generation for a class c

P: $\{x_0, x_1, x_2 ... x_{m-1}\}$ // Random population
$|P| = m$
while $(\exists x \in P) : f_c(x) \leq \tau$ **do**
  $P^{'} = \{< x, f_c(x) > |\forall x \in P\}$
  Let, L be a list in descending order of $f_c(x), \forall x \in P^{'}$
  $P^{*} = L[0 \ldots m * 0.25]$ //top 25% of elements
  $C = [crossover(P^{*}[j], P^{*}[j+1])|\forall j \in [0 \ldots |P^{*}| - 1]]$
  $M = [mutation(x)|\forall x \in P^{*}]$
  $M_C = [crossover(M[i], M[i+1])|\forall i \in [0, \ldots, |M| - 1]]$
  $P_{new} = P^{*} \cup C \cup M \cup M_C$
  $P = P_{new}$
end while

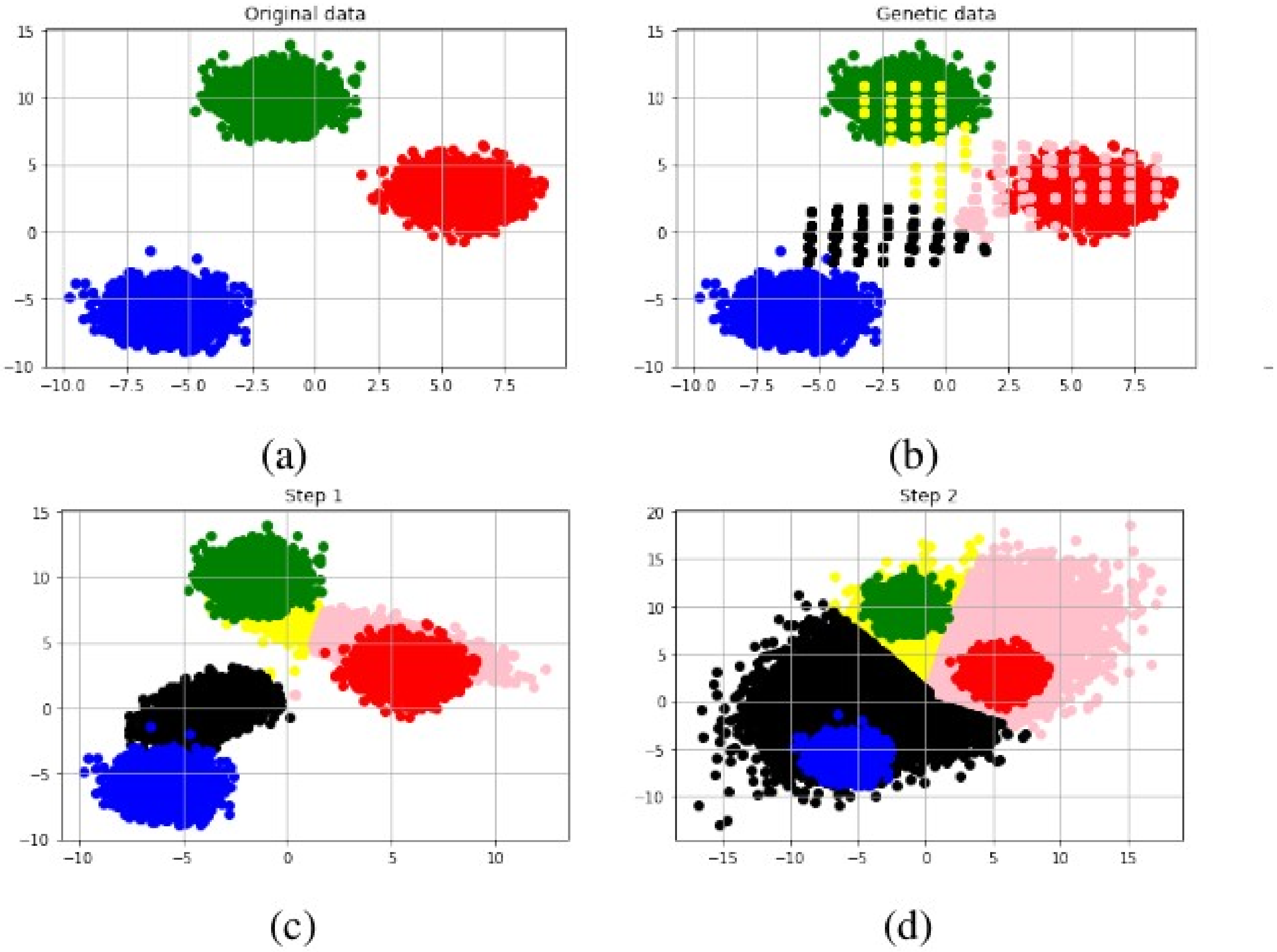
Here $f_c(x) = \frac{e^{z_c}}{\sum_{j=1}^{K} e^{z_j}}$
where z is vector of scores for each of the classes $1 \ldots K$, c is the given target class, $f_c(x)$ is softmax score for class c on input x and K is the total number of classes.

Algorithm 1 Synthetic data generation for a class c

P: $\{x_0, x_1, x_2 ... x_{m-1}\}$ // Random population
$|P| = m$
while $(\exists x \in P) : f_c(x) \leq \tau$ **do**
  $P^{'} = \{< x, f_c(x) > |\forall x \in P\}$
  Let, L be a list in descending order of $f_c(x), \forall x \in P^{'}$
  $P^{*} = L[0 \ldots m * 0.25]$ //top 25% of elements
  $C = [crossover(P^{*}[j], P^{*}[j+1]) | \forall j \in [0 \ldots |P^{*}| - 1]]$
  $M = [mutation(x) | \forall x \in P^{*}]$
  $M_C = [crossover(M[i], M[i+1]) | \forall i \in [0, \ldots, |M| - 1]]$
  $P_{new} = P^{*} \cup C \cup M \cup M_C$
  $P = P_{new}$
end while

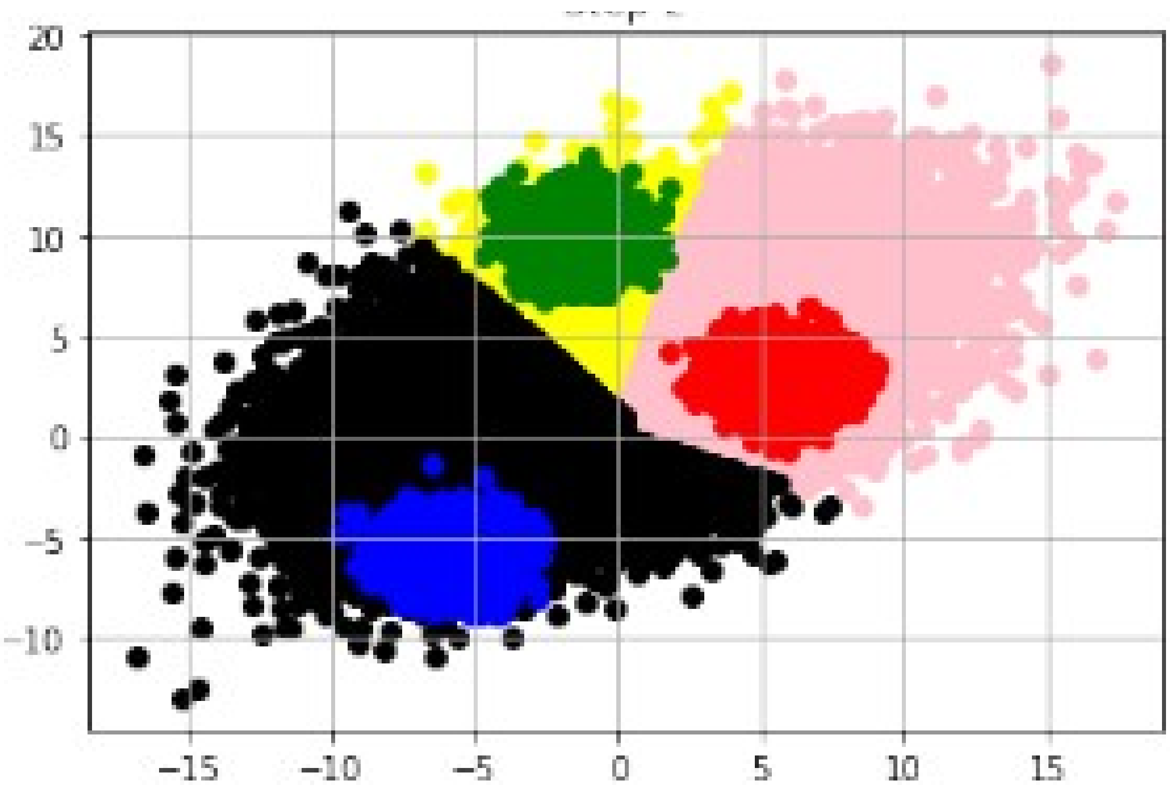
Here $f_c(x) = \frac{e^{z_c}}{\sum_{j=1}^{K} e^{z_j}}$
where z is vector of scores for each of the classes $1 \ldots K$, c is the given target class, $f_c(x)$ is softmax score for class c on input x and K is the total number of classes.

“make_blobs” dataset from Sklearn



(a) (b) (c) (d)

“*make_blobs*” dataset from Sklearn



= Real data

= Synthetic data

Experiment on learning and retention behaviour ..

Experiment on learning and retention behaviour ..

- Step 1:: M_1 = Train (B_1)

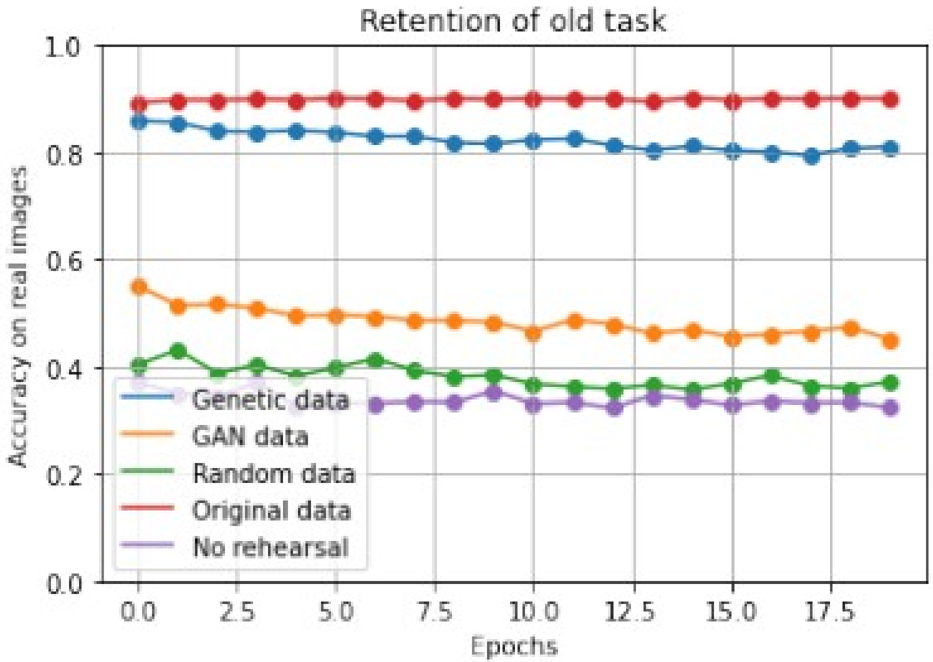
Experiment on learning and retention behaviour ..

- Step 1:: M_1 = Train (B_1)

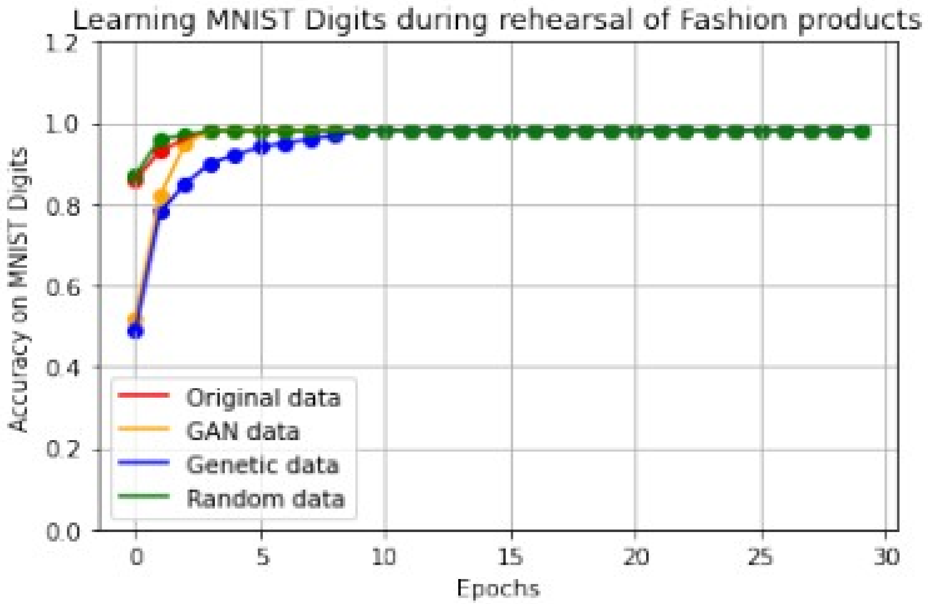
- Step 2:: M_2 = Train (B_1^* U B_2)

here B_1, B_2 are the training data for tasks "T_1" and "T_2" and B_1^* is the synthetic version for B_1.

Experiment on learning and retention behaviour ..



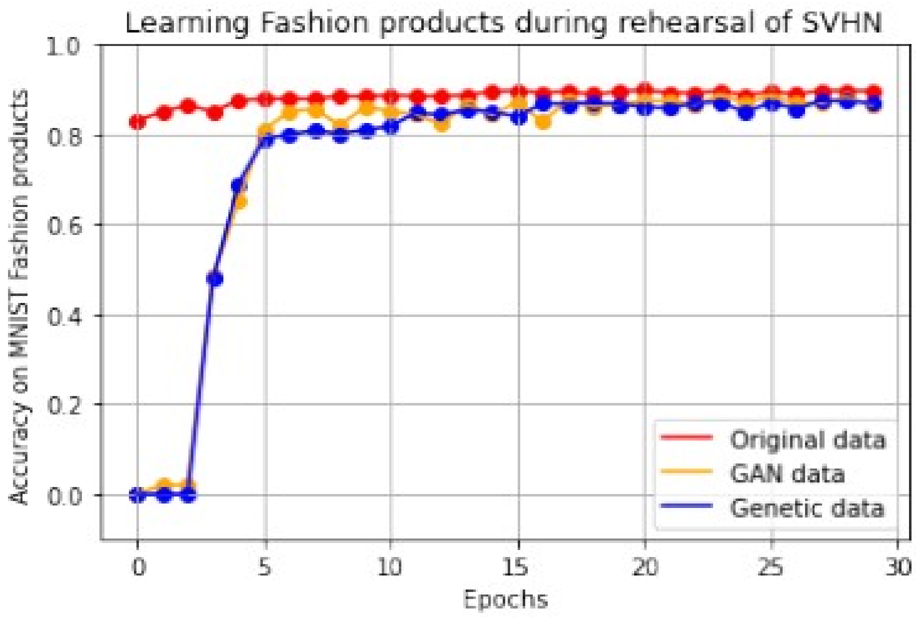
(a) Retention of Fashion dataset



(a) Learning of MNIST digits during rehearsal of Fashion dataset.

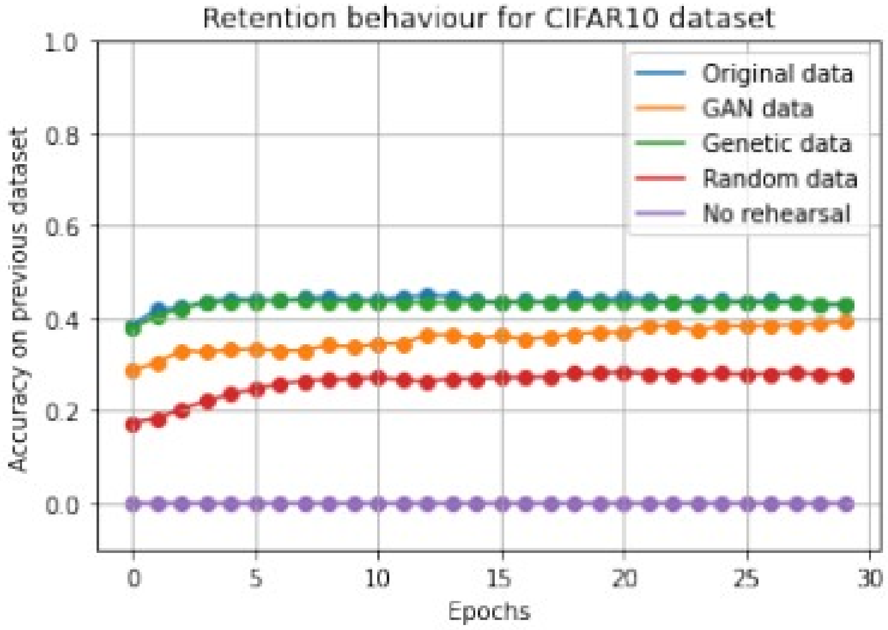
SVHN + Fashion datasets:

(b) Retention of SVHN dataset

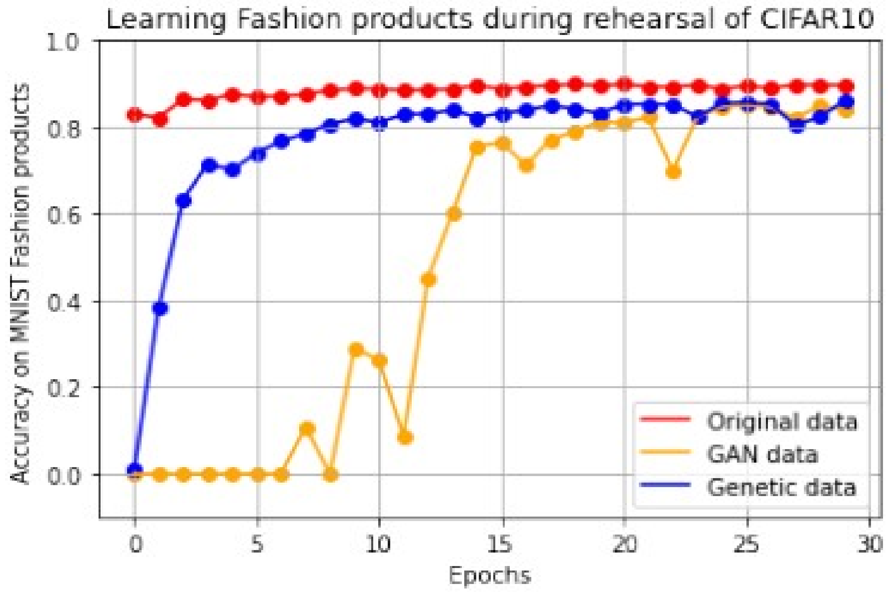


(b) Learning of MNIST Fashion during (
rehearsal of SVHN r

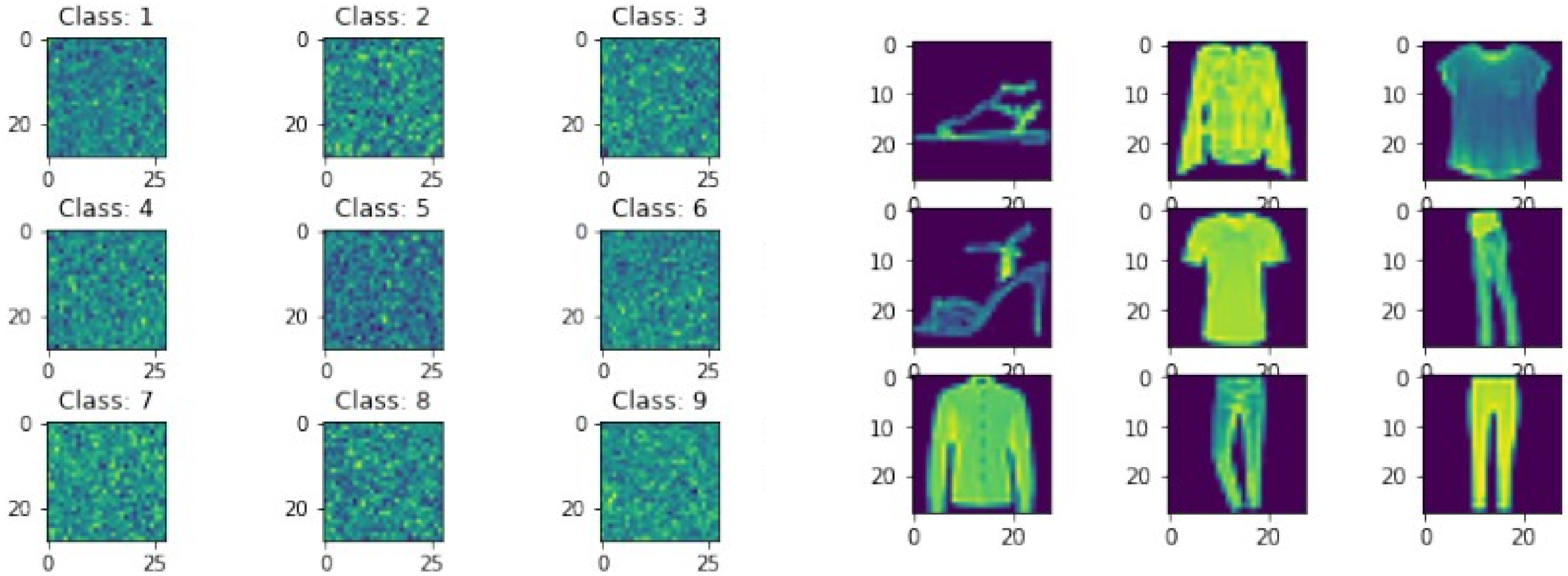
CIFAR10 + Fashion datasets:



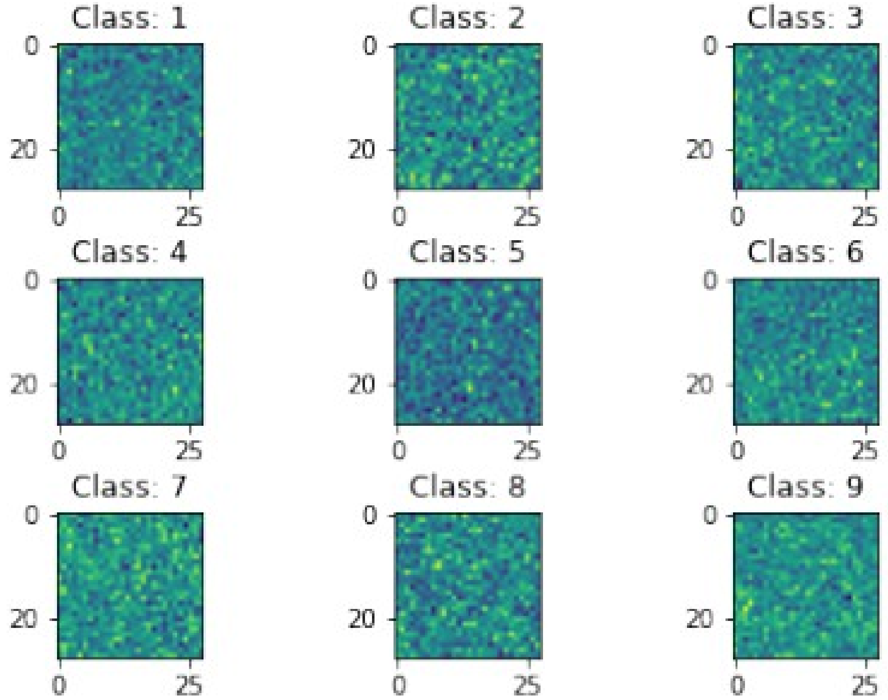
(c) Retention of CIFAR10

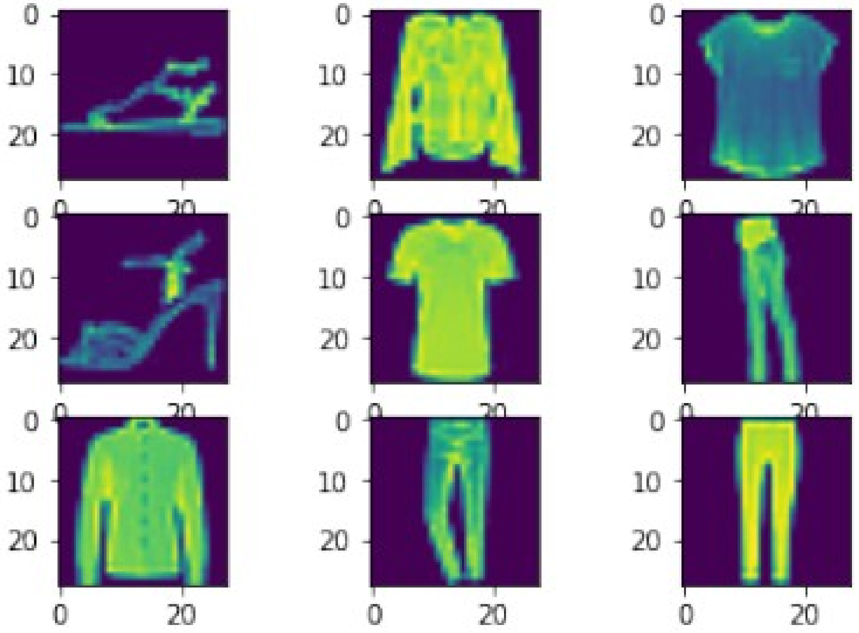


(c) Learning of MNIST Fashion during rehearsal of CIFAR10

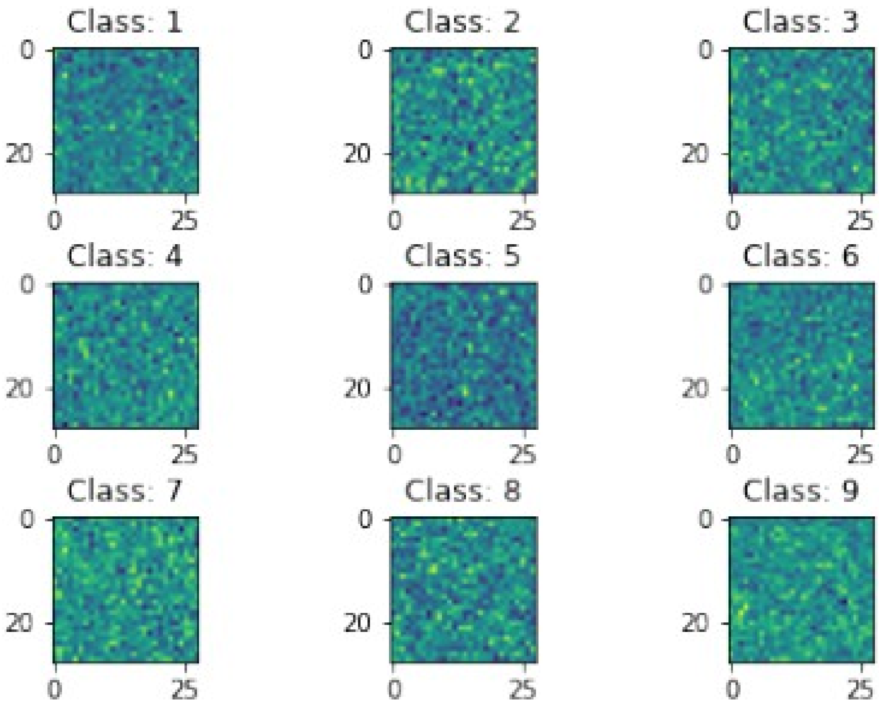

Class: 1
Class: 2
Class: 3
Class: 4
Class: 5
Class: 6
Class: 7
Class: 8
Class: 9

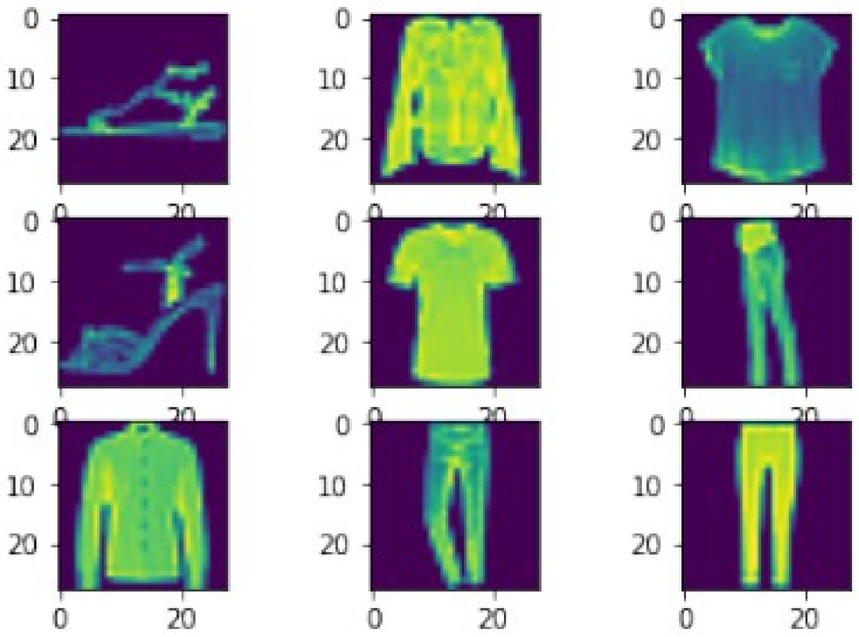
Can a neural network that is trained on images that look like random noise, classify real images?





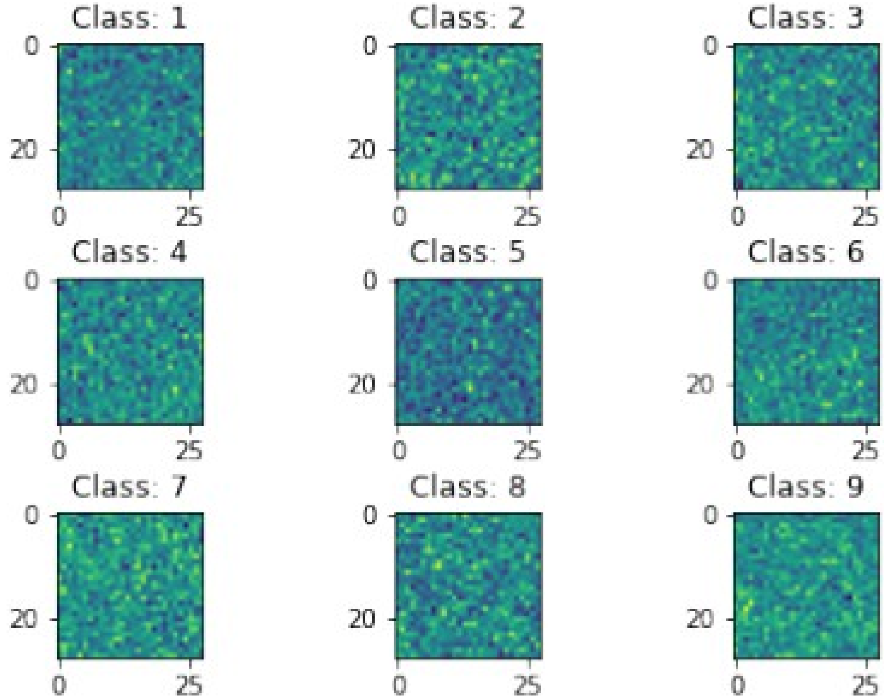
Can a neural network that is trained on images that look like random noise, classify real images?

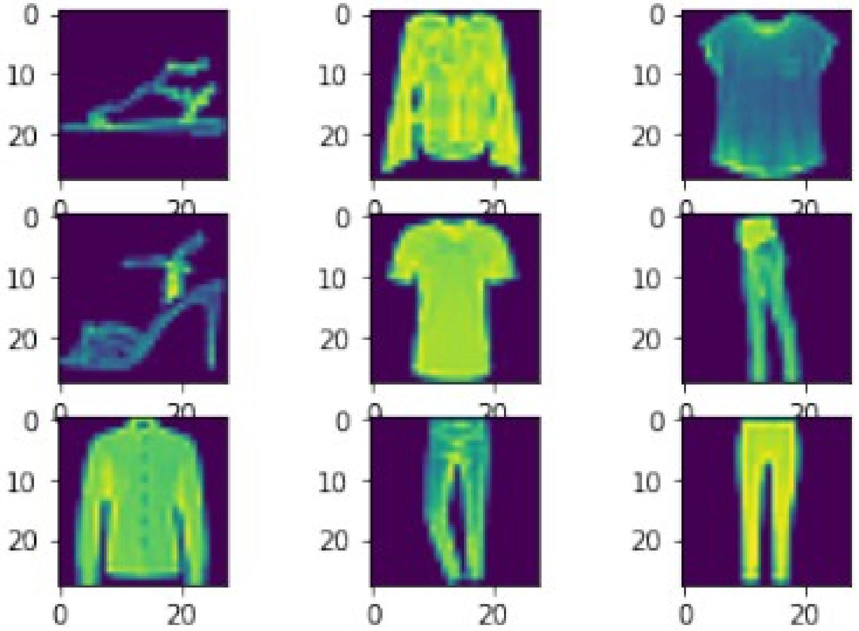




TRAIN

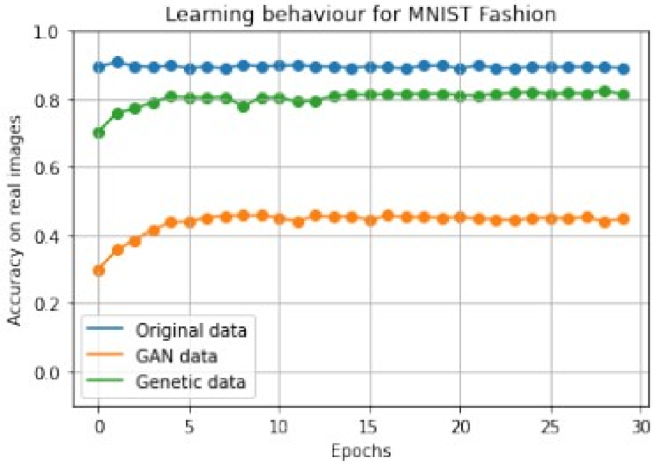
Can a neural network that is trained on images that look like random noise, classify real images?



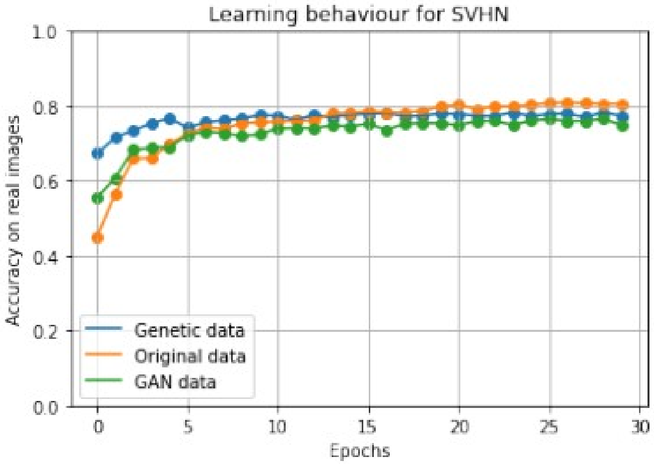


TEST ??

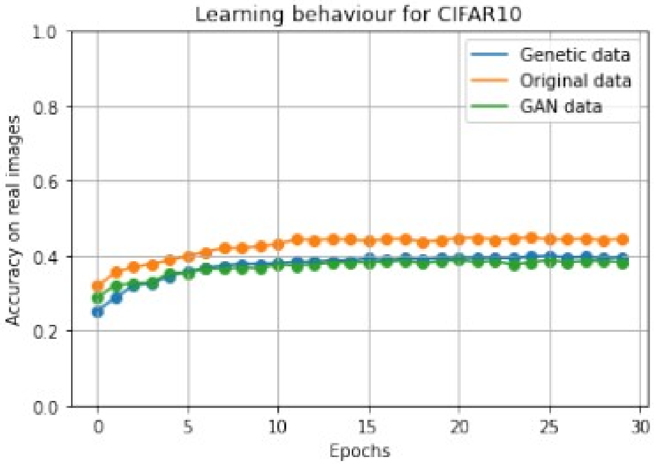
Experiment results



(d) Classification accuracy on real fashion images



(e) Classification accuracy on real SVHN images



(f) Classification accuracy on real images of CIFAR10

* By “near-perfect”, we mean accuracy reached when trained on original data.

Agreement score

Agreement score

$$\alpha(P_M, P_N) = \frac{\theta}{|T|} * 100$$

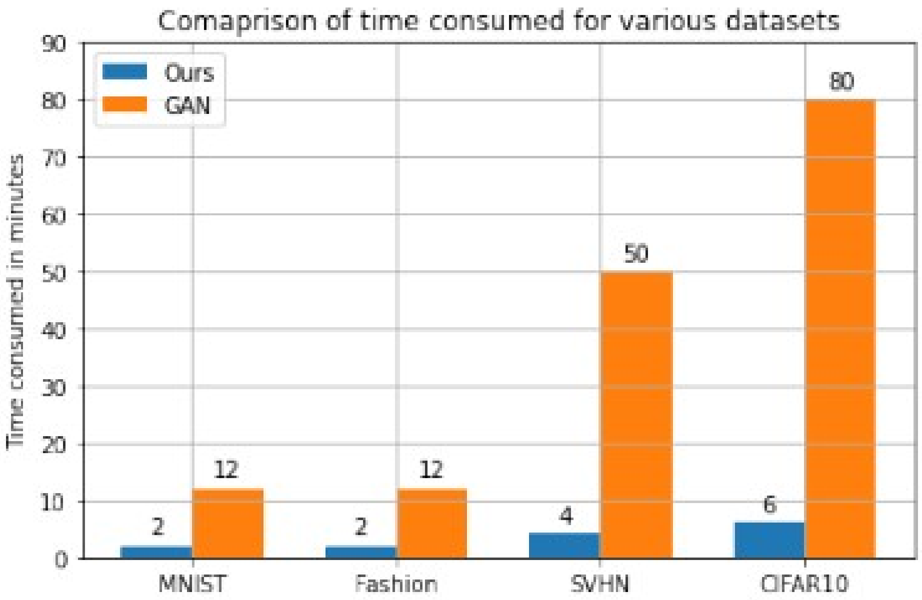
where P_M and P_N are the predictions of model M and model N on some test dataset. θ is the number of identical predictions, $|T|$ is the size of the test data.

Agreement score

Dataset	GAN Data	Ours
MNIST Handwritten digits	**95.346%**	83.675%
SVHN	**86.8%**	83.6%
MNIST Fashion products	52.459%	**80.977%**
CIFAR10	55.7%	**61.8%**

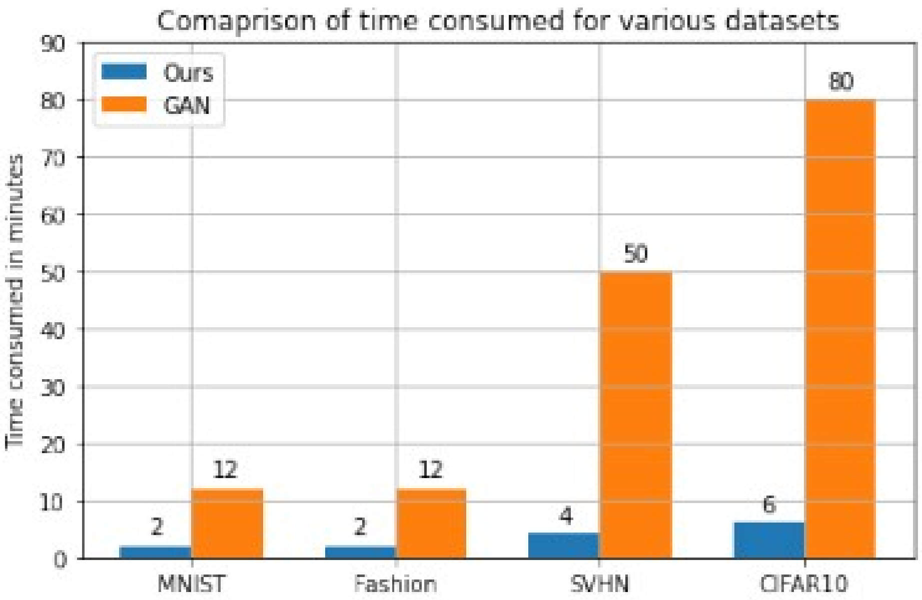
TABLE I: Results of Agreement score experiment

Computational requirements:



(a) Time required by all the methods

Computational requirements:



(a) Time required by all the methods

	GAN	Ours
Hardware	Tesla P100 GPU	Intel Xeon 2.5 Ghz Dual core CPU

Key differences ...

	GAN based approaches	Ours
Input	Original data	Model, Target labels
Output	Synthetic data	Synthetic data

Conclusions ..

- Neural networks can be rehearsed on non-photo realistic images.
- High retention and learning capacities can be achieved with non-photo realistic images as well.
- Forgoing photo-realism can result in efficient utilization of computational resources.

Thank you.