

# GAN-Tree: An Incrementally Learned Hierarchical Generative Framework for Multi-Modal Data Distributions

Supplementary Material

## Introduction

We present a cluster assignment algorithm which is used in the *GAN-Tree* framework to assign cluster labels to new test samples. We also show further splits of *GAN-Tree* trained on the ImageNet [4] dataset and discuss the flexibility of *GAN-Tree* which allows it to model even outlier clusters. Following this, we show a step by step implementation of the *iGAN-Tree* incremental training algorithm (Algorithm 4 of the main paper). To encourage reproducible research, we will make the code available publicly.

## Unsupervised Classification using *GAN-Tree*

After obtaining a fully-trained *GAN-Tree*, the hierarchical tree framework is used to assign cluster label or object label category for a test sample  $x$  satisfying the input training distribution. The entire tree is traversed, starting from the root node to one of the leaf nodes, following the maximum inference probability given by the encoder network  $E_{post}^{(n)}(x)$  at each tree node  $n$  (see Algorithm 1). The nodes traversed by  $x$  at different levels of the *GAN-Tree* is also utilized to assign hierarchical categorization of the sample  $x$ . This demonstrates the advantage of the inherent top-down divisive clustering setting of *GAN-Tree* as compared to other multi-modal [2; 5] or multi-generator [1; 3] approaches. We achieve 96% accuracy on the MNIST test-set classification in a fully unsupervised setting.

---

**Algorithm 1** *Assign Label* procedure for cluster label assignment of a new sample  $x$ , given a *GAN-Set*  $G$  from *GAN-Tree*  $T$

---

```
1: input:  $x, G, T$ 
2:  $k \leftarrow 0$  ▷ index of the root node of  $T$ 
3: while  $k \notin G$  do
4:    $z \leftarrow E_{post}^{(k)}(x)$ 
5:    $k \leftarrow \operatorname{argmax}(p(i|z))$  ▷  $i \in \operatorname{children}(k)$ 
6: return  $k$ 
```

---

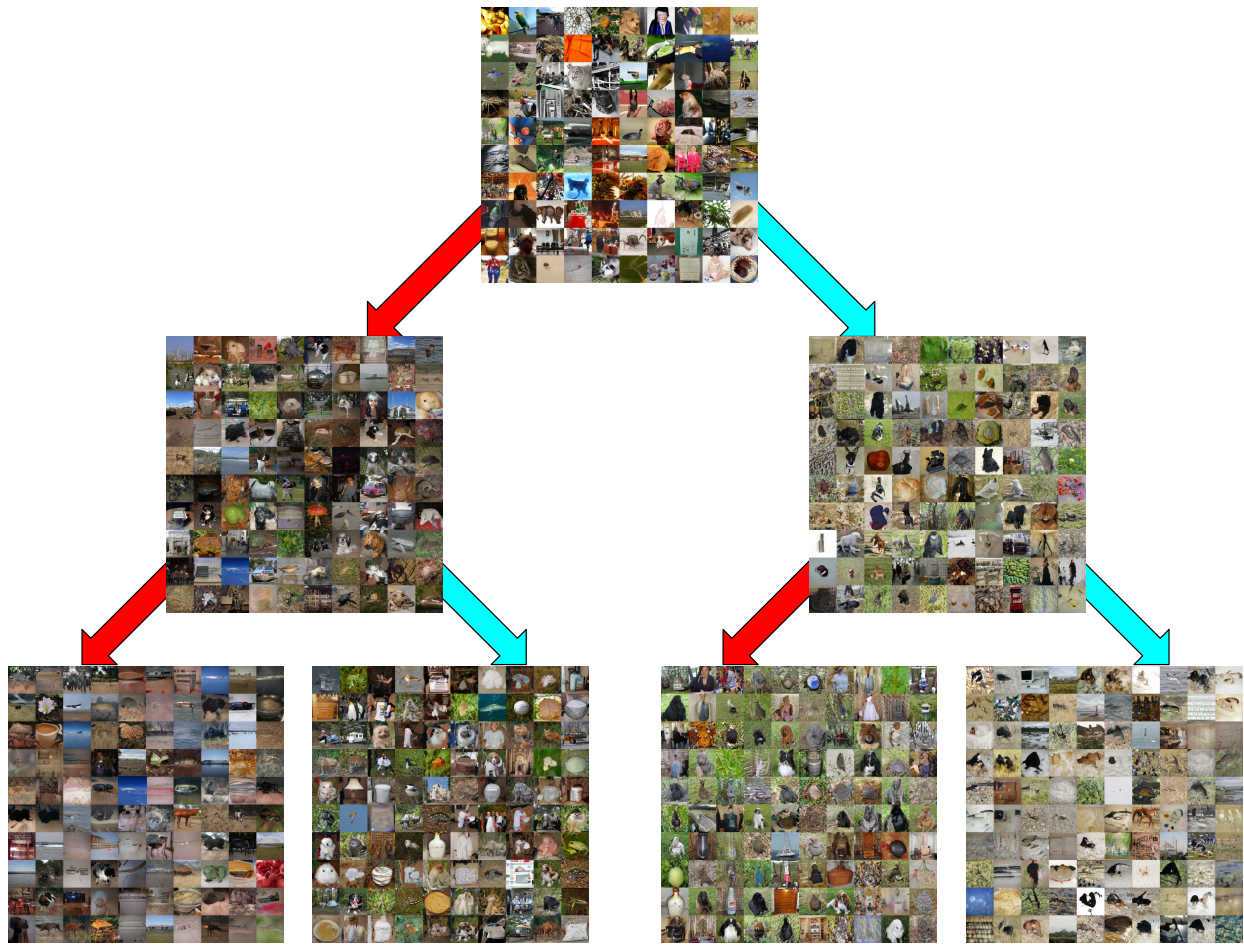


Figure 1: Splits of *GAN-Tree* trained on ImageNet.

## Further Splits of *GAN-Tree* Trained on ImageNet

Figure 1 shows the splits obtained by *GAN-Tree* trained on the ImageNet [4] dataset. The root node is split into images portraying scenery or having objects on a darker background (left child node) and into images portraying greenery or objects on a lighter background (right child node). A further split of the left child node of the root is primarily into images with scenery (left) and those with animals (right). A further split of the right child node of the root is into images with greenery (left) and those with objects on a light-colored background (right).

## Flexibility in Prior Probability Assignment

The prior probabilities defined for each node  $n$  can be updated manually depending on the requirement. For instance, consider a curated MNIST dataset with an equal number of samples for all digit classes except a single class consisting of 50% fewer samples as compared to others. In such

a case, the proposed *GAN-Tree* framework has a leaf node dedicated to that particular class with a reduced number of samples. However, during generation, samples are taken from the leaf nodes with uniformly distributed prior probabilities, even in the presence of data imbalance in the actually given training set. We observe decorated quality of generation for samples of that particular class despite the unavailability of many samples to approximate the corresponding distribution. This demonstrates the effectiveness of *GAN-Tree* to model outlier clusters in data distribution even with very few numbers of input samples.

## Step-by-Step Implementation of *iGAN-Tree*

We show a step by step implementation of the *iGAN-Tree* Incremental Training Algorithm (Algorithm 4 of the main paper, copied to Algorithm 2 for reference) for adding the digit class 5 to a *GAN-Tree* trained on MNIST digit classes 0-4. The procedure is portrayed in Figure 2.

- The Base *GAN-Tree*  $T$ , which has been trained on MNIST digit classes 0 to 4, is given along with the new Data Sample set  $\mathcal{D}'$ , which consists of the MNIST digit class 5 (see Fig. 2A).

---

### Algorithm 2 Incremental *GAN-Tree* Training

---

```

1: input: GAN-Tree  $T$ , New Data Sample set  $\mathcal{D}'$ 
2:  $i \leftarrow \text{index}(\text{root}(T))$ 
3: while  $i$  is NOT a leaf node do
4:    $l \leftarrow \text{left}(i)$ ;  $r \leftarrow \text{right}(i)$ 
5:   if  $\text{Avg}(p_x^{(l)}(\mathcal{D}')) \geq p_z^{(l)}(\mu^{(l)} + d_{\sigma_0})$  then  $i \leftarrow l$ 
6:   else if  $\text{Avg}(p_x^{(r)}(\mathcal{D}')) \geq p_z^{(r)}(\mu^{(r)} + d_{\sigma_0})$  then  $i \leftarrow r$ 
7:   else break
8: Here,  $i$  is the current node index
9:  $j \leftarrow \text{NewId}()$  (new parent index)
10:  $k \leftarrow \text{NewId}()$  (new child index)
11:  $\text{par}(k) \leftarrow j$ ;  $\text{par}(j) \leftarrow \text{par}(i)$ ;  $\text{par}(i) \leftarrow j$ 
12: if  $i = \text{index}(\text{root}(T))$  then
13:    $\text{root}(T) \leftarrow j$ ;  $\text{left}(j) \leftarrow i$ ;  $\text{right}(j) \leftarrow k$ 
14: else if  $i$  was the left child of its previous parent then
15:    $\text{left}(\text{par}(j)) \leftarrow j$ ;  $\text{left}(j) \leftarrow i$ ;  $\text{right}(j) \leftarrow k$ 
16: else
17:    $\text{right}(\text{par}(j)) \leftarrow j$ ;  $\text{left}(j) \leftarrow k$ ;  $\text{right}(j) \leftarrow i$ 
18: Create networks  $G^{(k)}$ ,  $D^{(k)}$  with random initialization
19: Train  $E^{(j)}$ ,  $G^{(k)}$  and  $D^{(k)}$  with  $\mathcal{L}_{recon}$  and  $\mathcal{L}_{adv}$ 
20:  $E^{(\text{par}(j))} \leftarrow \text{copy}(E^{(\text{par}(i))})$ 
21:  $G^{(j)} \leftarrow \text{copy}(G^{(i)})$ ;  $D^{(j)} \leftarrow \text{copy}(D^{(i)})$ ;  $i \leftarrow \text{par}(i)$ 
22: while  $GN^{(i)}$  is not root( $T$ ) do
23:   IncrementalNodeTrain( $i$ );  $i \leftarrow \text{par}(i)$ 
end while
24: Train  $GN^{(i)}$  with GAN Training Procedure on  $\mathcal{D}'$  and generated samples from Terminal GAN-Set.

```

---

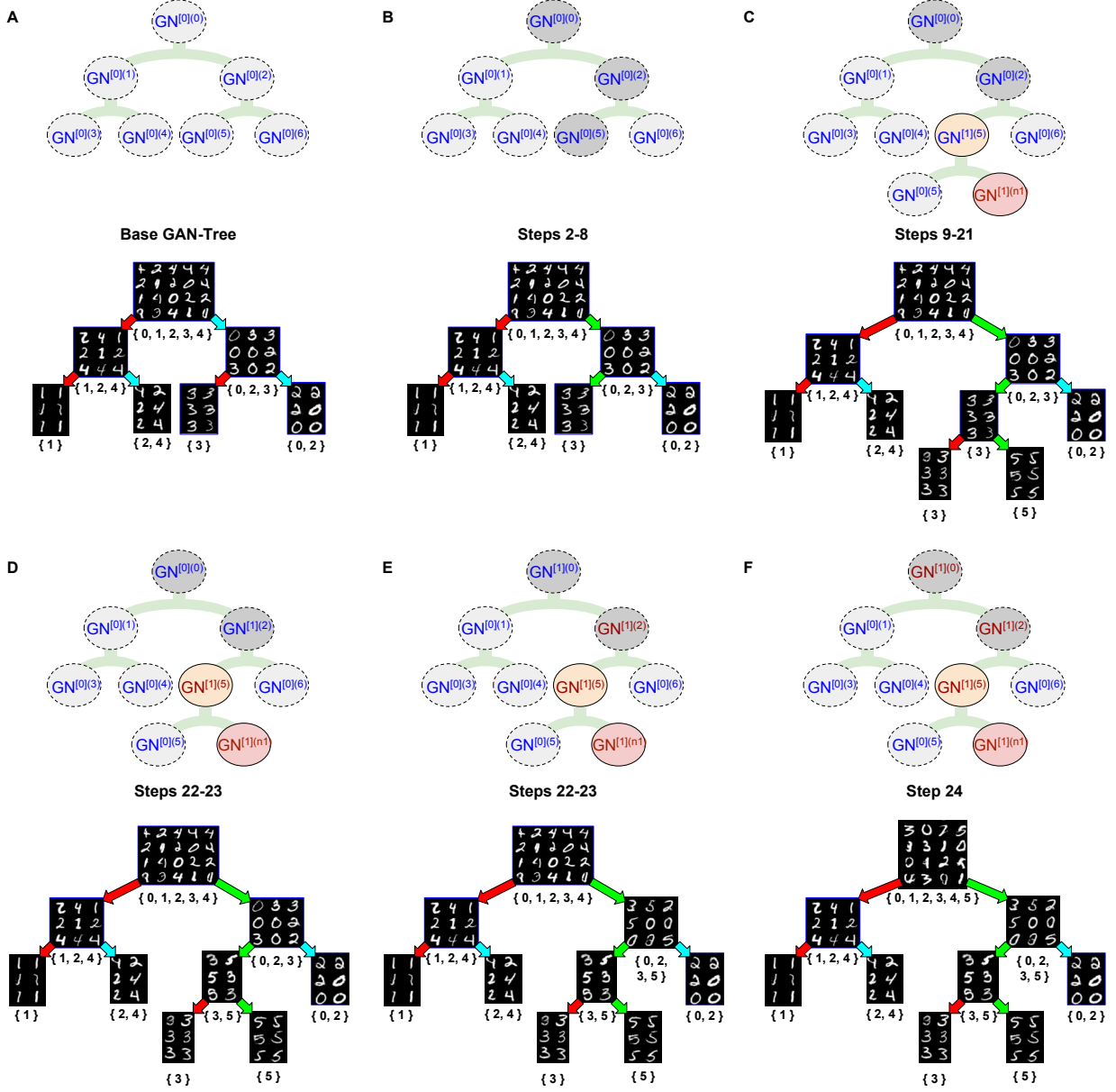


Figure 2: Incremental GAN-Tree training algorithm procedure for a Base *GAN-Tree*  $T$  trained on MNIST digit classes 0-4 and new Data Sample  $\mathcal{D}'$  which consists of MNIST digit class 5.

- The procedure for seeking the apt position to insert the new child node is run in a top-down fashion (lines 2-8 of Algorithm 2). It starts with the root node and checks which child node of the current node best models the data samples from  $\mathcal{D}'$ . After selecting a child node, this step is recursed on this selected node until a leaf node is reached or its children cannot model the new data samples appropriately. The position at which this seek procedure stops is where the new child node is to be inserted. The seek procedure path ends at  $GN^{[0](5)}$  for the case taken, and is highlighted in dark gray (see Fig. 2B).
- Once the position at which the new child node is to be inserted is found, two new nodes are created (lines 9-11 in Algorithm 2): i) a parent node with ID  $j$  ( $GN^{[1](5)}$  in the case taken), and ii) a child node with ID  $k$  which will model only  $\mathcal{D}'$  ( $GN^{[1](n1)}$  in the case taken). The node at which the seek procedure ends ( $GN^{[0](5)}$  for the case taken) becomes the child of the new parent node, according to whichever child it was of its previous parent, so as to maintain the same prior distribution (lines 12-17 in Algorithm 2). The child node is then trained after which the new parent node is assigned weights (lines 18-21 in Algorithm 2). The new parent node is highlighted in orange, the new child node in red, and the trained nodes in red text (see Fig. 2C).
- Once the new child node has been trained, the ancestor nodes are trained recursively (lines 22-23 of Algorithm 2), according to the Incremental Node Training Algorithm (given in Algorithm 3 of main paper). The recursive steps are shown in Fig. 2D and Fig. 2E. The node after retraining is highlighted in red text.
- Finally, the root is retrained with the GAN Training Procedure on samples from  $\mathcal{D}'$  and the samples generated from the GAN-Set formed by the leaf nodes (line 24 in Algorithm 2). The *GAN-Tree* structure after the root node training is shown in Fig. 2F.

## References

- [1] Arnab Ghosh, Viveka Kulharia, Vinay Namboodiri, Philip HS Torr, and Puneet K Dokania. Multi-agent diverse generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [2] Swaminathan Gurusurthy, Ravi Kiran Sarvadevabhatla, and R Venkatesh Babu. Deligan: Generative adversarial networks for diverse and limited data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [3] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. Mgan: Training generative adversarial nets with multiple generators. In *International Conference on Learning Representations*, 2018.
- [4] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.
- [5] Ilya O Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann Simon-Gabriel, and Bernhard Schölkopf. Adagan: Boosting generative models. In *Advances in Neural Information Processing Systems*, 2017.