Massachusetts Institute of Technology

CSAIL

Re-purposing Heterogeneous Generative Ensembles with Evolutionary Computation

The Genetic and Evolutionary Computation Conference (**GECCO 2020**)
July 8th-12th 2020, Cancun Mexico

JAMAL TOUTOUH
toutouh@mit.edu

ERIK HEMBERG
hembergerik@ csail.mit.edu

UNA-MAY O'REILLY
unamay@csail.mit.edu

ALFA
ANYSCALE LEARNING FOR ALL

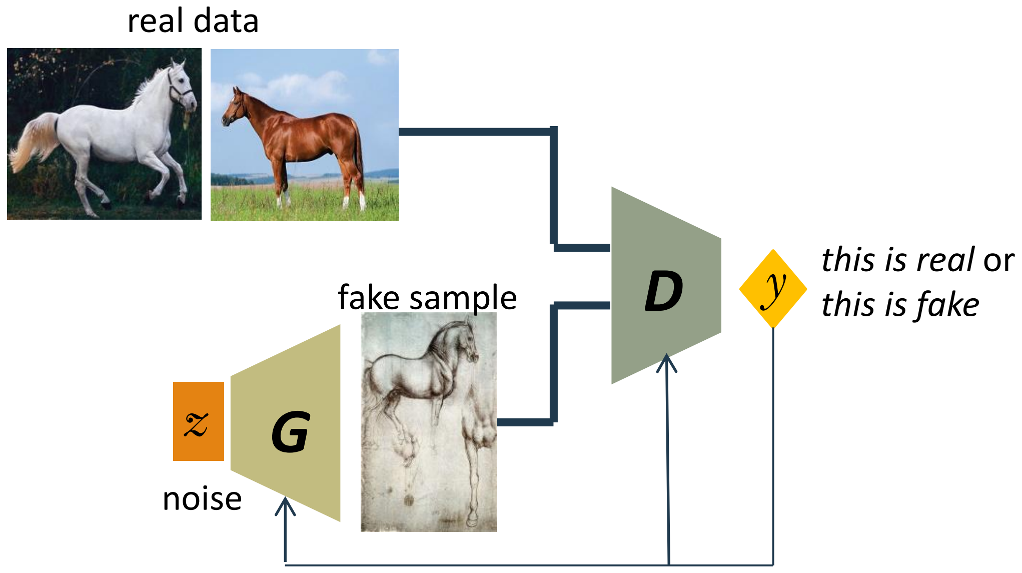
Agenda

- Introduction and Motivation
- Output Diversity GAN Optimization
- Genetic Ensemble Creation
- Experimental Setup
- Results
- Conclusions and Future Work

Introduction and Motivation

- **Generative models** are neural networks able to read random vectors (z) from a latent space and generate synthetic samples of a given data distribution

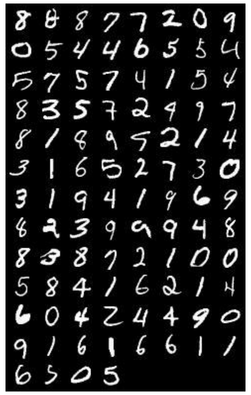
  - Generative Adversarial Networks (**GANs**) construct a generative models by raising an arms race between two neural networks, a **generator** (*G*) and a **discriminator** (*D*)



Introduction and Motivation

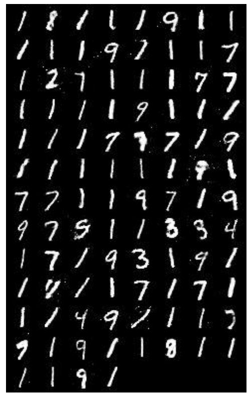
- Training GANs is difficult since the adversarial dynamics may give rise to different convergence pathologies
  - **Mode collapse** happens when the training converges to a local optimum, i.e. the generator produces realistic fake images that only represent a portion of the real data distribution

High quality
High diversity



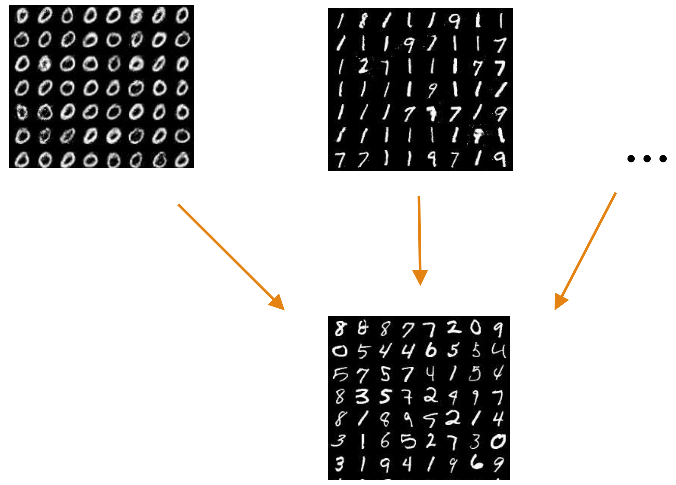
Mode collapse → limited diversity

High quality
Low diversity



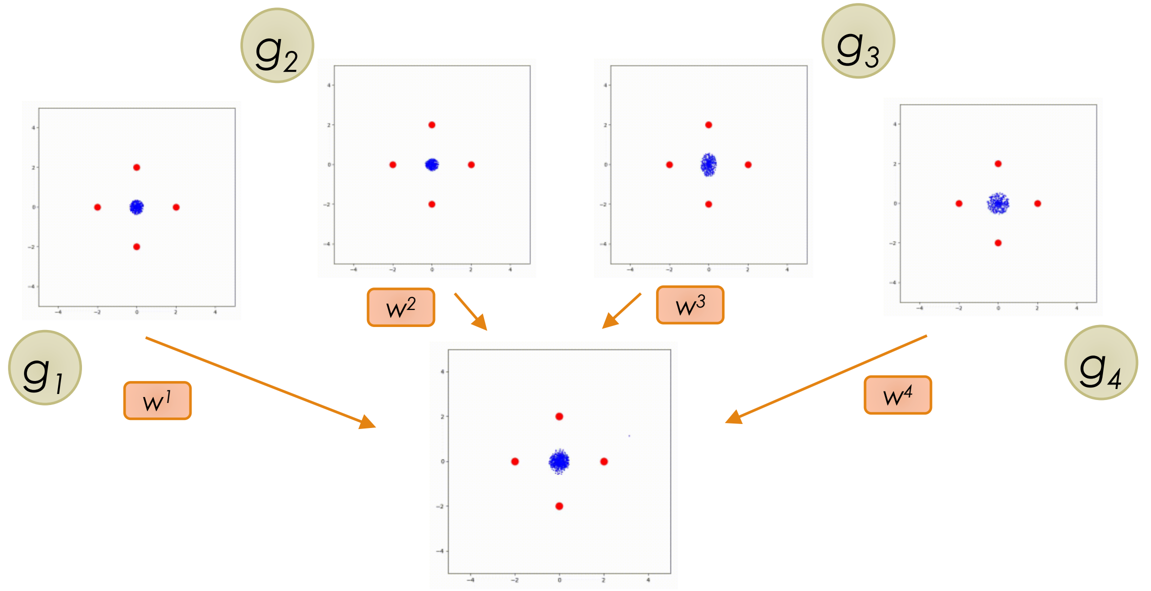
Introduction and Motivation

- A promising approach to deal with GAN training issues is training **ensembles of networks**
- Evolutionary Ensemble Learning (**EEL**) is the application of Evolutionary Computing to learn ensemble



Introduction and Motivation

- **Main idea:** Giving a set of **heterogeneous generators** previously trained to optimize a given objective learn mixtures (ensembles) of them to generate samples according to a new objective function
  - We focus on **samples diversity** to deal with mode collapse
  - Creating a generative model with a mixture of generators and randomly drawing samples from any of them according to a giving probability/weight w^i



Output Diversity GAN Optimization

- **Problem definition:**
  - We have a set of *n* generators **G**={g_0, …, g_{n-1}} optimized to get the best sample (image) possible
  - We want to get generative models that maximizes the output diversity in terms of Total Variation Distance (**TVD**), maximizing diversity is minimizing TVD

$$TVD = \frac{1}{2} \sum_{c \in Classes} | freq(real_c) - freq(fake_c) |$$

  - We define the **ensemble optimization problem** as to find the n-dimensional mixture weight vector ***w***

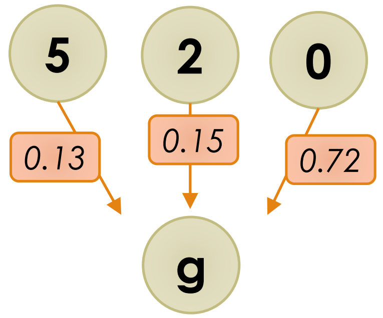
$$w^* = argmin\, TVD\left(\sum_{i=0}^{n-1} w_i G g_i\right)$$

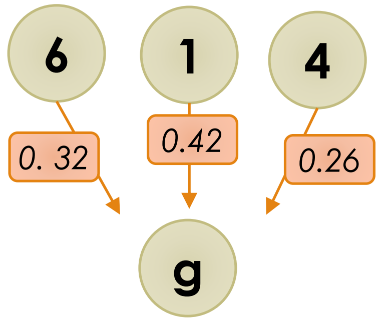
    - w_i represents the probability that a data sample comes from the *ith* generator in **G**
    - Gg_i represents the samples created by g_i
    - with $\sum_{i=0}^{n-1} w_i = 1$

Output Diversity GAN Optimization

- Restricted Ensemble Optimization of GENenrators (**REO-GEN**)
  - restriction: ensemble size *s* is known, $|w_i \neq 0| = s$

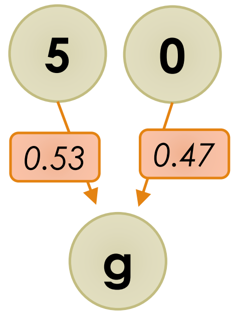
s=3

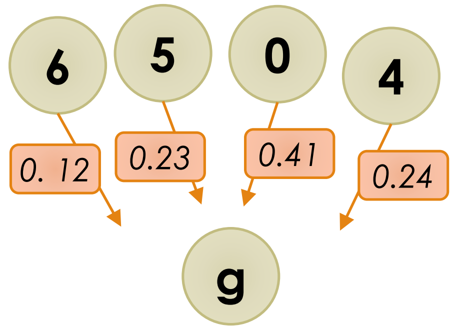




- Non-Restricted Ensemble Optimization of GENenrators (**NREO-GEN**)
  - restriction: an upper-bound for the ensemble size, $|w_i \neq 0| \leq s$

$s \leq 5$



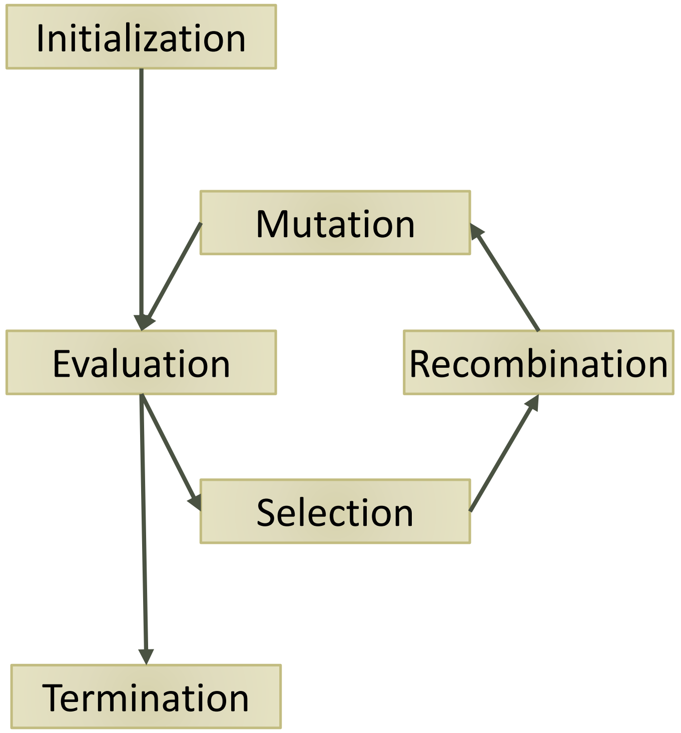


Genetic Ensemble Generation

- REO-GEN and NREO-GEN are addressed by using a **Genetic Algorithm**

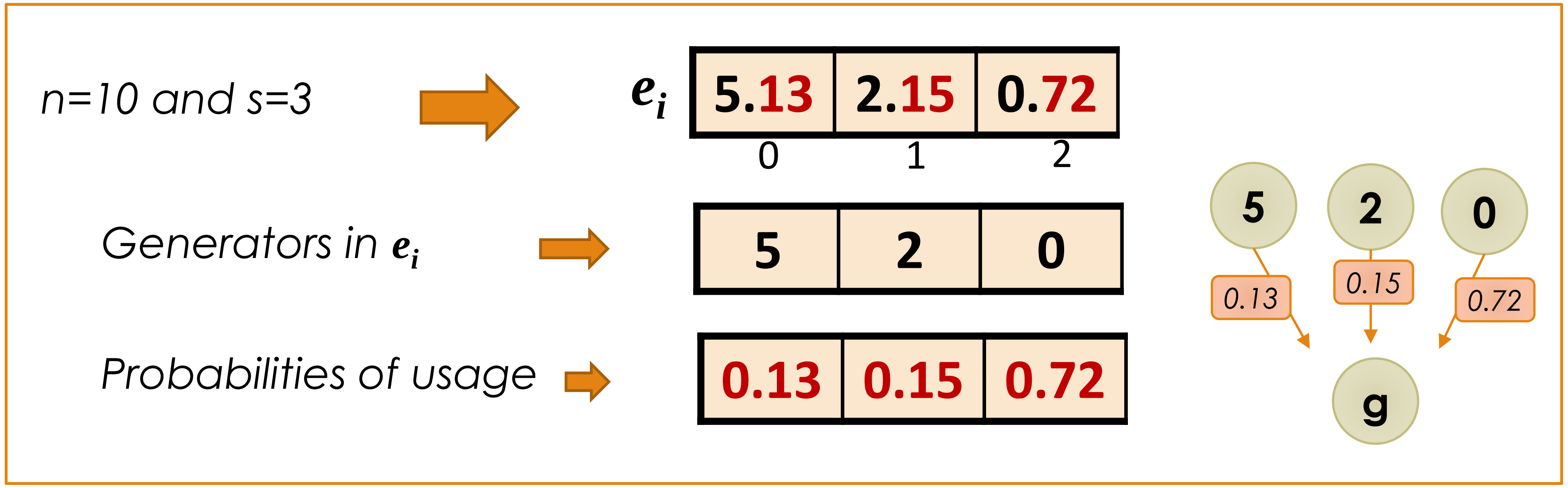
Genetic Algorithm

```
population = Create initial population
while not stop criteria do
    evaluate(population)
    parents = selection(population(generation))
    offspring = recombine(parents, recombination_prob)
    offspring = mutate(parents, mutation_prob)
    new_population = replace(offspring, population)
    update(population)
```



REO-GEN. Encoding

- A solution is represented as a **vector of real numbers** with two decimals of precision, having length s $[g^0.w^0, \ldots, g_{s-1}.w^{s-1}]$
  - *g^i is the **generator index***
  - *w^i represents the **probability** of using g^i to draw a sample*
  - *Restrictions:*
    - $g^i \in \{0, 1, \ldots, n-1\}$
    - $g^i \neq g^j$ for all $i \neq j$
    - $w_i > 0$
    - $\sum_{i=0}^{s-1} w_i = 1$

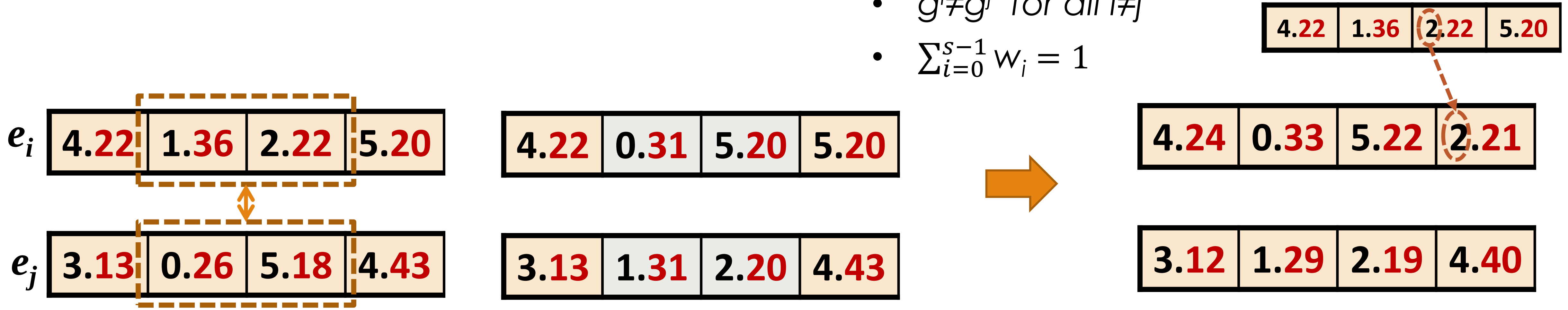


REO-GEN. Recombination

- The information of the genes located between two randomly picked points within the chromosome are exchanged
  - Swapping the integer part (generators identifiers)
  - Averaging the weights

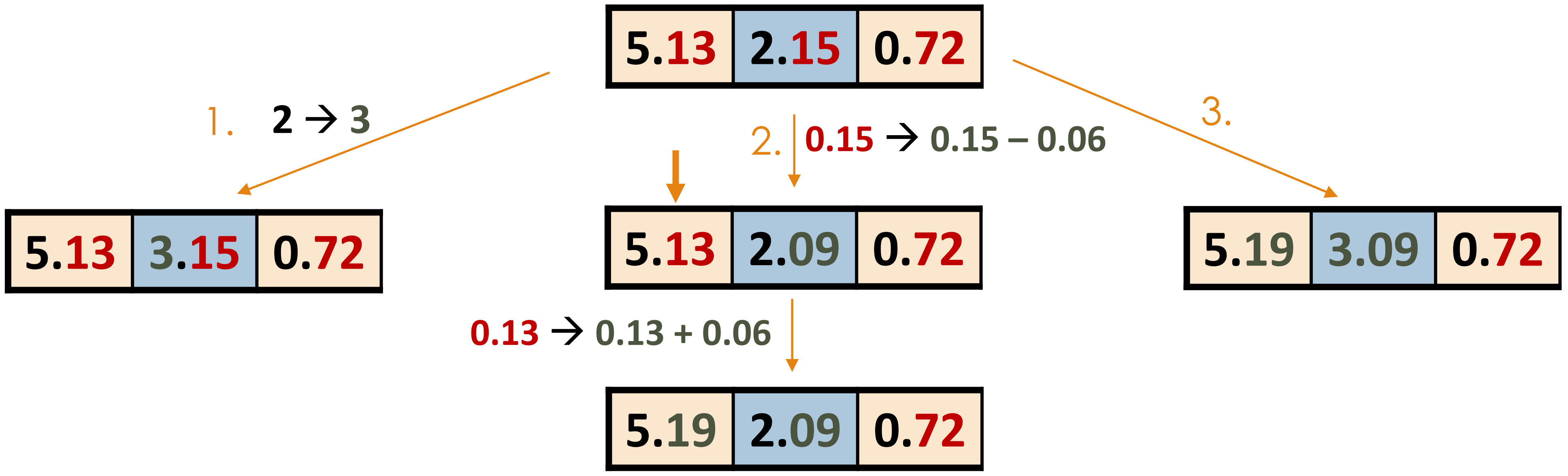
Fixing operator is applied

- *$g^i \neq g^j$ for all $i \neq j$*
- $\sum_{i=0}^{s-1} w_i = 1$



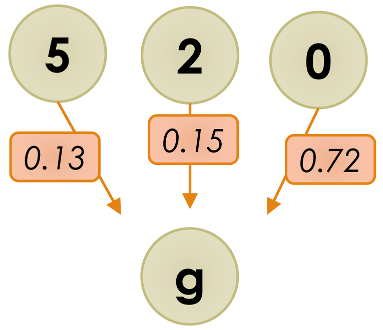
REO-GEN. Mutation

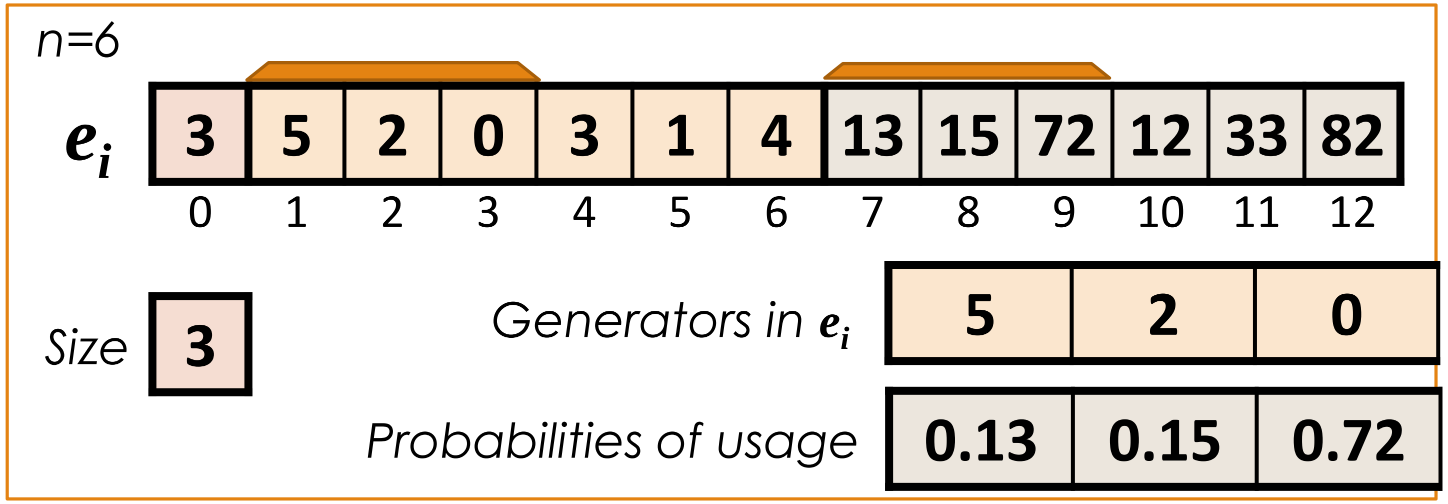
- Equally likely applies three variations on solutions
  1. randomly changing the generator
  2. randomly modifying the probability (addition or subtraction)
  3. combining both



NREO-GEN. Encoding

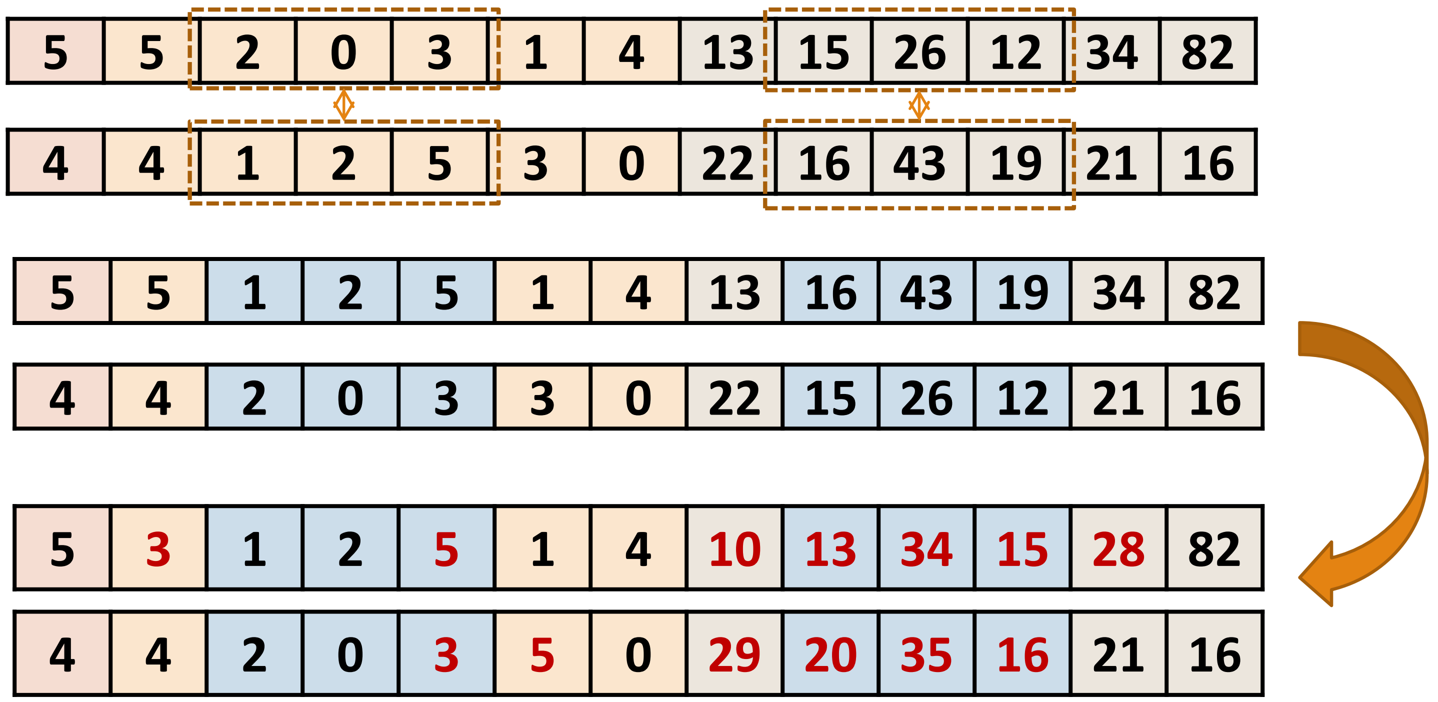
- A solution is represented as a **vector of integer numbers** of 2n+1 elements [e^0, e^1, …, e^n, e^{n+1}, …, e^{2n+1}] divided in three segments:
  - **ensemble size** (one element) → [e^0]
  - **generators** identification (n elements) → [e^1, …, e^n]
  - **probability** definition (n elements) → [e^{n+1}, …, e^{2n+1}]
  - *Restrictions:*
    - $0<e^0 \leq n$
    - $e^i \neq e^j$ *for all* $i \neq j$ *and* $i,j \in \{1, \ldots, n\}$
    - $e^i>0 \quad i \in \{n+1, \ldots, n+e^0\}$
    - $\sum_{i=n+1}^{n+e^0} e^i = 100$





NREO-GEN. Recombination

- The information of the genes located between two randomly picked points within the chromosome are exchanged
  - Swapping the generators identifiers and the probabilities



5	5	2	0	3	1	4	13	15	26	12	34	82
4	4	1	2	5	3	0	22	16	43	19	21	16

5	5	1	2	5	1	4	13	16	43	19	34	82
4	4	2	0	3	3	0	22	15	26	12	21	16

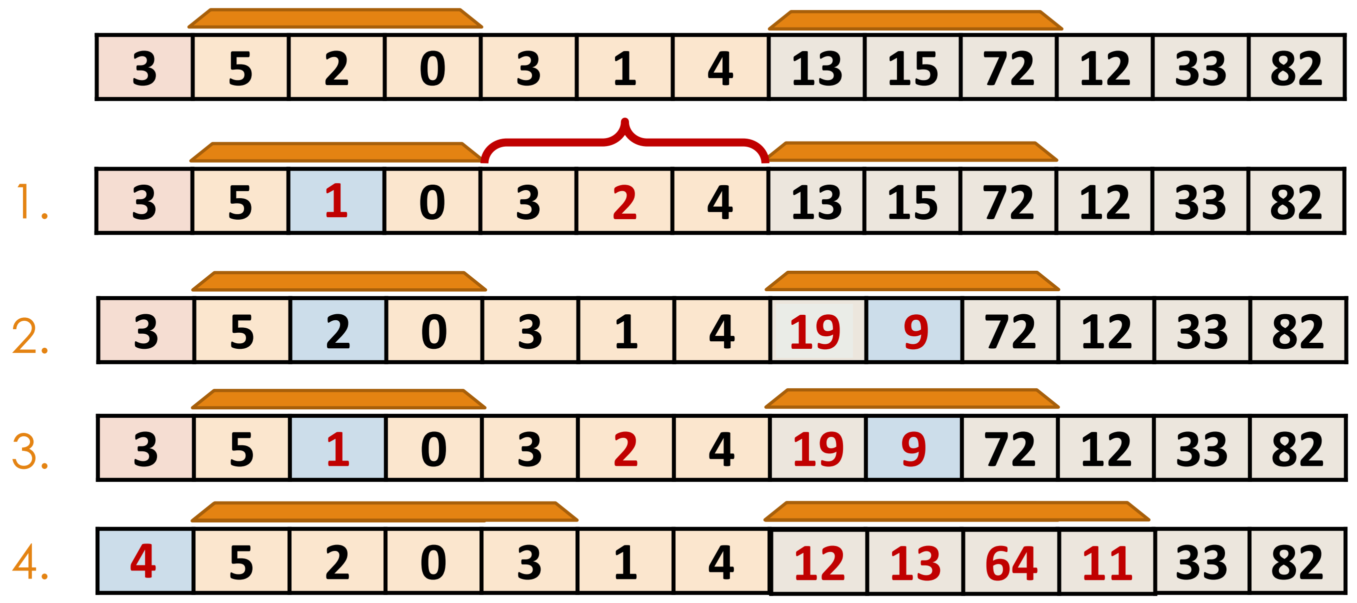
5	3	1	2	5	1	4	10	13	34	15	28	82
4	4	2	0	3	5	0	29	20	35	16	21	16

***Fixing* operator** is applied

- *$g^i \neq g^j$ for all $i \neq j$*
- $\sum_{i=n+1}^{n+e^0} e^i = 100$

NREO-GEN. Mutation

- Equally likely applies four variations on solutions
  1. randomly changing the generator
  2. randomly modifying the probability (addition or subtraction)
  3. combining both (1 and 2)
  4. randomly picking new generator size

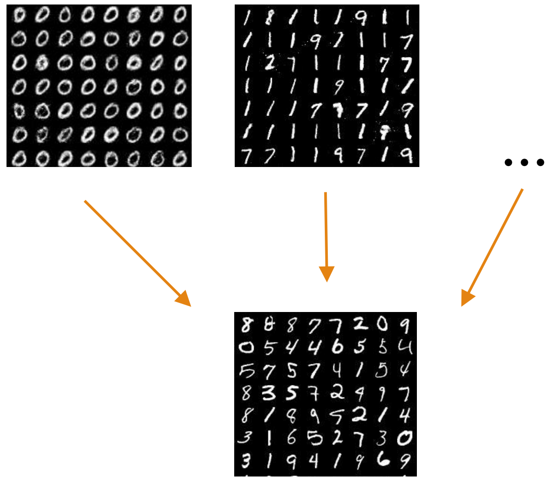


	3	5	2	0	3	1	4	13	15	72	12	33	82
1.	3	5	1	0	3	2	4	13	15	72	12	33	82
2.	3	5	2	0	3	1	4	19	9	72	12	33	82
3.	3	5	1	0	3	2	4	19	9	72	12	33	82
4.	4	5	2	0	3	1	4	12	13	64	11	33	82

REO-GEN and NREO-GEN. Evaluation

- The fitness evaluation for both problems is computing by drawing a number of samples by using the ensemble defined by the solution (e_i) and computing the TVD

$$fitness(e_i) = TVD(e_i)$$



Experimental Setup

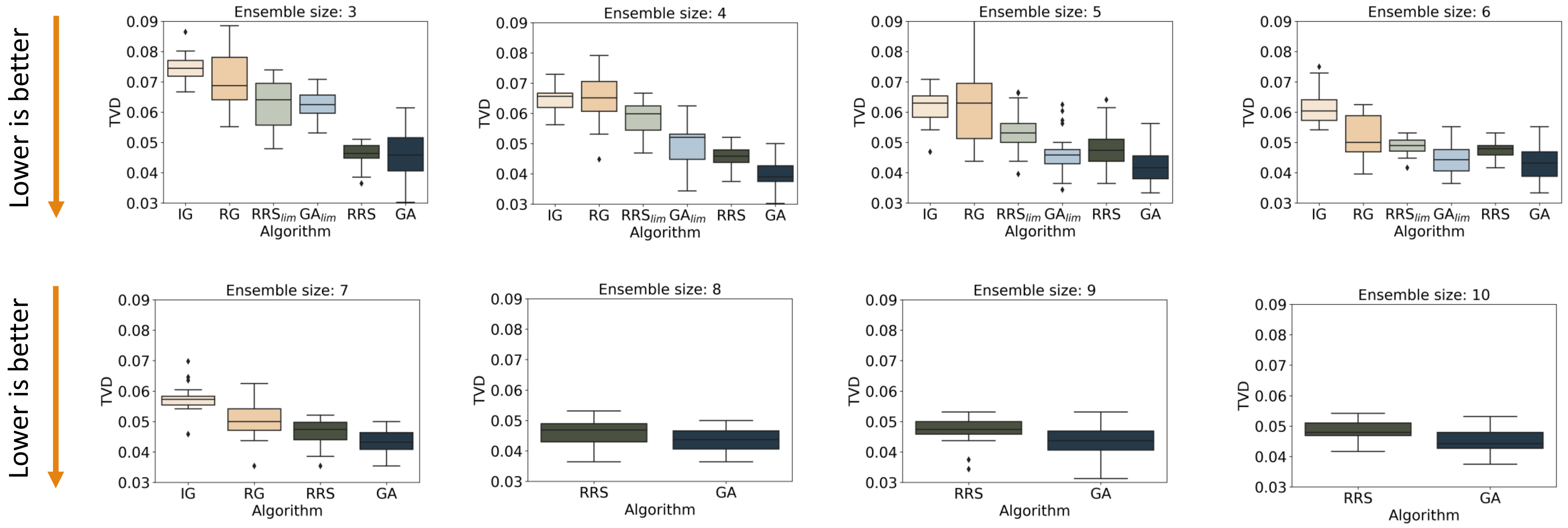
- Source code built by using Pytorch and DEAP
- Problem instance: MNIST
  - 3,000 pre-trained generators to create MNIST dataset samples
  - We keep (220) the ones that creates samples with a given quality FID<40
    - Average FID = 36.393 (**Quality**)   Average TVD = 0.113 (**Diversity**)
- We address:
  - REO-GEN with sizes from 3 to 10
  - NREO-GEN with two different max sizes: 8 and 100

Experimental Setup

- Methods:
  - REO-GEN: Iterative Greedy (**IG**), Random Greedy (**RG**), **GA**, Random REO Search (**RRS**), **GA$_{lim}$** (GA with same fitness evaluation than IG and RG) and **RRS$_{lim}$** (RRS with same fitness evaluation than IG and RG)
  - NREO-GEN: **GA** and Random NREO Search (**RNRS**)
- Experimental Setup:
  - Stop condition: performing 10,000 fitness (TVD) evaluations
  - Population size: 100 individuals (ensembles)
  - Fitness evaluation: TVD generating 50,000 MNIST samples

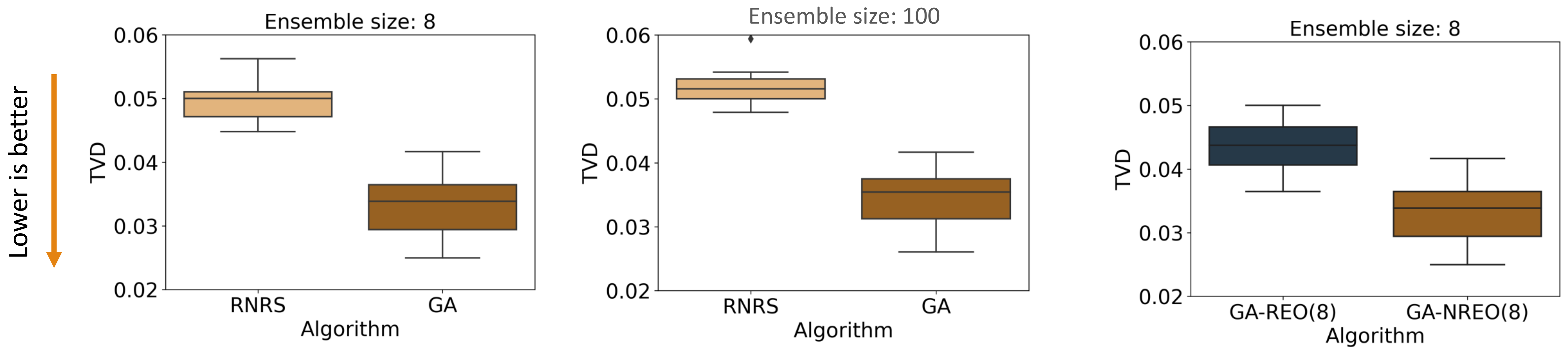
Results. REO-GEN Fitness

- With the same fitness evaluations GA_{lim} provides the best results
- GA finds the best ensembles



Results. NREO-GEN Fitness

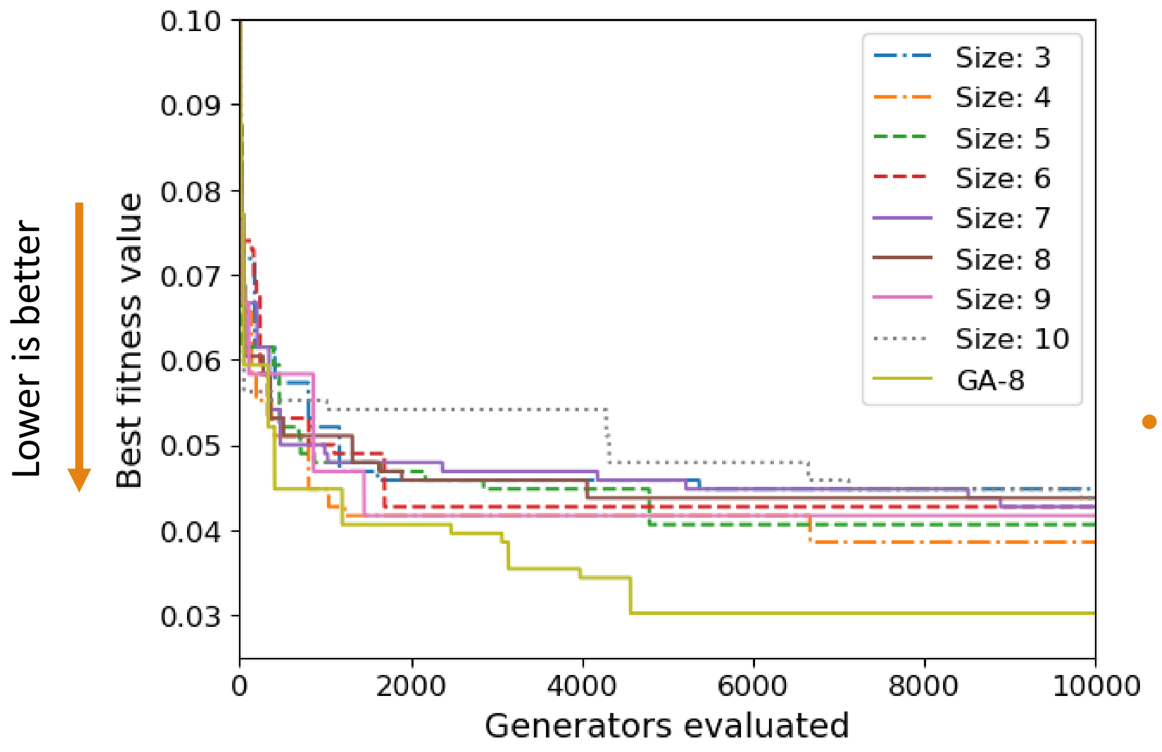
- GA provides the best ensembles
- There not improvements when allowing bigger ensembles



- When comparing REO-GEN (s=8) and NREO-GEN(s≤8), the second one finds more competitive generators

Results. Fitness Evolution

- Evolution of the median run



- REO-GEN:
  - when s=10, it converges slower
  - when s=4, it converges to the best results
  - for the other ensemble sizes the performance is similar
- NREO-GEN (GA-8):
  - converges faster
  - the best ensemble before the first 5,000 fitness evaluations

Results. Generative Models Quality

- Diversity and quality of all the computed generative models according to the ensemble size
- Created generative models (ensembles) critically improves TVD and FID
- NREO-GEN ensembles provides the best TVD
- Best REO-GEN ensembles are with s=4

Ensemble size (s)	TVD Mean±Stdev	FID Mean±Stdev
1	0.113±0.010	36.393±1.985
3	0.046±0.006	27.576±3.947
4	**0.043±0.005**	27.890±3.048
5	0.046±0.007	28.225±3.888
6	0.045±0.005	27.077±3.030
7	0.045±0.005	27.556±3.531
8	0.046±0.005	26.726±3.515
9	0.046±0.005	26.919±3.181
10	0.047±0.004	27.218±3.408
`GA(NREO(8))`	0.033±0.004	27.342±3.530

Percentage of NREO-GEN solutions given the size of the ensemble

Ensemble size	3	4	5	6	7	8
Percentage of solutions (%)	23.3	**46.7**	20.0	6.7	0.0	3.3

The most repeated size of a NREO-GEN final solution is 4

Conclusions and Future Work

Conclusions:

- We have empirically shown that EEL can be applied to re-purpose generative models by combining previously trained generators
  - optimize diversity of the samples generated (TVD)
- We have defined two different problems: REO-GEN and NREO-GEN
- GA constructs generative models that highly improve the diversity (TVD) and the quality of the generated samples (FID)
- The best results were obtained when we used the GA to address NREO-GEN
  - The degree of freedom of not searching for ensembles with specific sizes allows this method to move through the search space in the way that it converges to better results than the other methods

Conclusions and Future Work

Future Work:

- Evaluating other evolutionary approaches and encoding the weights in a different way such as a larger set of integers or continuous values
- Addressing the problem applying a MO approach to simultaneously optimize quality (FID), diversity (TVD), and network complexity
- Evaluating our evolutionary approach with other datasets that require much complex networks (e.g., CIFAR-10 or CelebA)
- Devising a algorithm that first will train a population of GANs (that tries to keep them as diverse as possible) and then it applies one of the defined methods to get high competitive generative model

Massachusetts Institute of Technology

ALFA
ANYSCALE LEARNING FOR ALL

GECCO

Thanks!

Comments?

Massachusetts Institute of Technology

ANYSCALE LEARNING FOR ALL

Lipizzaner 2.0

Comming soon