Evolutionary Computation and Evolutionary Deep Learning for Image Analysis, Signal Processing and Pattern Recognition

Stefano Cagnoni[1], Ying Bi[2], and Yanan Sun[3]
[1]University of Parma, Italy
[2]Zhengzhou University, China
[3]Sichuan University, China
cagnoni@ce.unipr.it; yingbi@zzu.edu.cn; ysun@scu.edu.cn
http://gecco-2024.sigevo.org/

GECCO '24 Companion, July 14–18, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0495-6/24/07.. $15.00
https://doi.org/10.1145//3638530.3648410

Instructors

Stefano Cagnoni is an Associate Professor at the University of Parma. His research is mainly focused on EC applications to Image Analysis, Signal Processing and Pattern Recognition. Editor-in-chief of the "Journal of Artificial Evolution and Applications" from 2007 to 2010. For more than 10 years since 1999, he has chaired EvoIASP, an event dedicated to evolutionary computation for image analysis and signal processing, now merged with other events into the EvoApplications conference. At GECCO, he has co-chaired MedGEC, workshop on medical applications of EC and is currently co-chairing ECXAI on EC and Explainable AI. Co-editor of journal special issues dedicated to EC for Image Analysis and Signal Processing and Explainable AI. Member of the Editorial Board of the journals "Evolutionary Computation" and "Genetic Programming and Evolvable Machines".

Ying Bi is a professor at Zhengzhou University, China. Her research focuses mainly on evolutionary computer vision and machine learning. She has published an authored book on genetic programming for image classification and over 50 papers in fully refereed journals and conferences. She is currently the Vice-Chair of the IEEE CIS Task Force on Evolutionary Computer Vision and Image Processing, and a member of the IEEE CIS Task Force on Evolutionary Computation for Feature Selection and Construction. She is serving as the workshop chair of IEEE CEC 2024, organizer of the EDMML workshop in IEEE ICDM 2023, 2022, and 2021, and co-chair of the special session on ECVIP at IEEE CEC 2023, 2022 and IEEE CIMSIVP at IEEE SSCI 2023, 2022. She is serving as AEs for seven international journals.

Yanan Sun is a professor at Sichuan University, China. He has been a research postdoc at Victoria University of Wellington, New Zealand. His research focuses mainly on evolutionary neural architecture search. He has published >70 papers in fully referred journals and conferences, including IEEE TEVC, IEEE TNNLS, IEEE TCYB, NeurIPS, CVPR, ICCV, GECCO, and CEC. 12 out of the published papers have been selected as ESI Hot Paper, ESI Highly Cited Paper, IEEE CIS Chengdu Section Best Paper, AJCAI2024 Spotlight Paper, and MLMI2022 Best Paper. He is the funding chair of the IEEE CIS Task Force on Evolutionary Deep Learning and Applications. He is the leading chair of the special session on EDLA at IEEE CEC 2019, 2020, 2021, 2022, and 2024, and the symposium on ENASA at IEEE SSIC 2019-2023. He is an associate editor of IEEE TEVC, an associate editor of IEEE TNNLS, and an editorial member of Memetic Computing.

2

Tutorial Agenda

- Introduction: Evolutionary Image Analysis and Signal Processing
- Part I: A basic pathway to GP-based Deep Learning
- Part II: Evolutionary Neural Architecture Search for IASP and Pattern Recognition
- Part III：Evolutionary Deep Learning based on Genetic Programming for IASP and PR
- Summary

3

Introduction: Evolutionary Image Analysis and Signal Processing

Computer Vision

- The "art" of making computers see (and understand what they see)
- Computer vision vs image processing
- Sub-topics:
  - Image acquisition
  - Image enhancement
  - Image segmentation
  - 3D-information recovery/feature extraction
  - Image understanding
  - Object tracking
  - Edge detection
  - Segmentation
  - Motion detection
  - Object/digit recognition

5

Computer and Human Vision

LOW-LEVEL VISION

HUMAN	COMPUTER
Perception	Image acquisition
Selective information extraction	Feature enhancement (signal/image processing)
Grouping by 'similarity'	Segmentation
Extraction of spatial relationships	3D-information Recovery
Object recognition and semantic interpretation	Image Understanding

HIGH-LEVEL VISION

6

Application Taxonomy

- EC techniques
  - GA, GP, ES, EP, PSO, DE, LCS, EMO, EDA, etc.
- *EC as an optimization tool*
  - Optimisation of parameters of specific solutions (using GA, ES, PSO…)
    - Related with a well-defined task or for a whole system
  - Generation of solutions from scratch (GP, …)
    - Performance optimization based on specific objective functions
    - It is difficult to choose a model with reasonable assumptions
- *EC as THE solution (or a relevant part of it)*
  - Interactive qualitative comparisons between solutions
  - Generation of emergent collective solutions
    - Achievement of higher-level and complex tasks from collective use of trivial, local, hard-wired behaviours: generation of full EC-based solutions, NOT just parameter optimization tasks

7

Part I: A Basic Pathway to GP-based "Deep" Learning

8

Goals and constraints

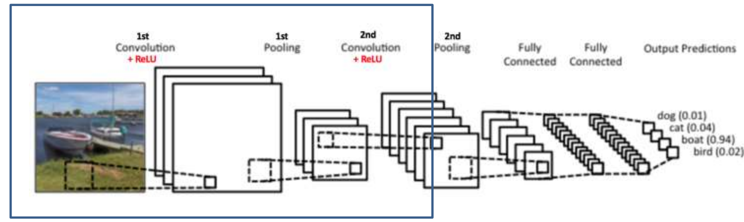
- Develop functionalities that can substitute (deep) neural networks' layers
- Using the higher representation power of symbolic function representation to synthesize more compact (and possibly interpretable) functions
- Evaluate what can be obtained by using the simplest and most direct possible approach, as close as possible to

one function = one GP tree

9

Deep Learning for Images
Typical Convolutional Neural Network structure

Input | Feature Extraction | Classification

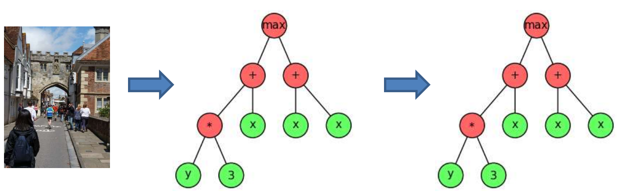


X Input | **Y=fe(X) Embedding layer** | **Z=c(Y) Classifier**

10

Deep Learning using GP

Input | Feature Extraction | Classification



X Input | **Y=fe(X) Embedding layer** | **Z=c(Y) Classifier**

11

GP-based "Deep" Learning

Observations:

- Y = fe(X) is a *vector of features*; therefore, fe(X) may be either a multi-output tree or a set of traditional GP trees. Its input may be a whole image (pattern) or it may be seen as a *filter or a set of non-linear kernels*, applicable to a small region, to be convolved with the input image.
- Z = c(Y) may be either a *single output* or, as happens in neural networks, *a set of N binary classifiers* or, possibly, *a softmax layer*, where N is the number of possible classes.

12

GP-based “Deep” Learning

Let’s make the simplest assumption: single-output GP trees composed of simple functions

and focus on the *embedding layer*:

- If we have real numbers as outputs, an M-dimensional embedding could be obtained:
  - by evolving M trees in parallel
  - by sequentially (iteratively) evolving single trees

GP-based “Deep” Learning

Regarding the *classifier*:

- One may set the classifier type and use a wrapper method to evolve an optimal embedding for that specific classifier...
- … but one may also consider a co-evolutionary wrapper where both the embedding and the classifier are “concurrently” evolved by two GP populations (termed P1 and P2, respectively)
  - Cooperative coevolution
  - Competitive co-evolution (GAN ?)

GP-based “Deep” Learning
Real vs Binary input data

Real input data and traditional GP (input data type same as output data type)

- The dimension of the embedding must be much larger than 1!

Binary input data and traditional GP

- A 64-bit embedding may be enough to generate discriminative features

SUB-MACHINE CODE GENETIC PROGRAMMING (Poli, Langdon 1998)

Inputs: Unsigned long (32 or 64 bit words) that encode arrays of binary inputs. The bit string may encode consecutive samples of a temporal sequence, a row or a window within a binary image, etc.

Function set: bitwise logical operators + circular shifts

A whole block of data is affected by a single bit-wise Boolean operation (SIMD paradigm).

Output: a 32/64 bit string, that may represent 32/64 possible outputs of a binary classifier.

So, 32/64 (non-independent) solutions are evaluated for each individual.

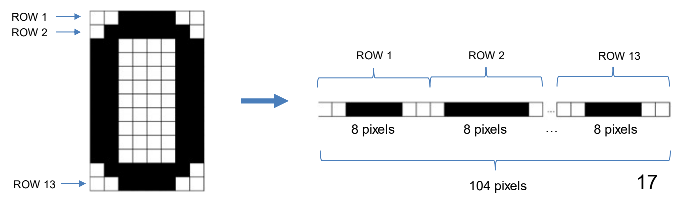
SmcGP example: Character Recognition

Real-world dataset collected by Società Autostrade SpA at highway toll booths

- 11034 binary patterns representing the ten digits from 0 to 9

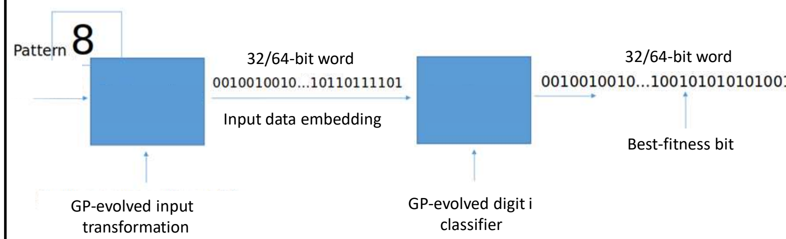
  6024 in the training set
  5010 in the test set (exactly 501 per class)

- Size: 13x8 pixels → strings of 104 binary features



17

New architecture: SmcGP + Embedding (a tiny step towards Deep GP)



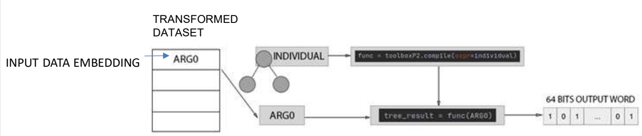
18

Co-evolutionary algorithm

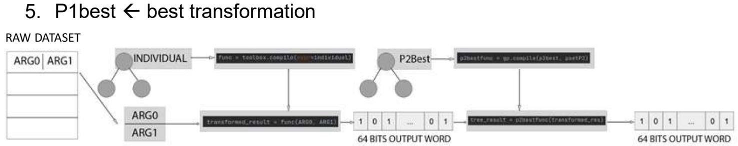
Generate a random initial embedding (P1Best)

Repeat

1. Compute the training set transformed by P1best
2. (*P2 evolution*) Evolve the classifier
3. P2best ← best classifier



4. (*P1 evolution*) Evolve the input data transformation using P2best as a classifier
5. P1best ← best transformation



Until the termination condition is reached

19

Parameters

Evolution of an embedding/classifier pair for one digit:

- Population: 1000
- Max number of generations: 1000
  (20 generations for each GP x 25 iterations)
- Termination condition/overfitting control: 40 consecutive generations without fitness improvement on the validation set (*if selected*)
- 5 runs
- Evolution parameters same as in the original paper
- Training set: 4218 patterns (*almost balanced*)
- Validation set: 1806 patterns (*almost balanced*)
- Test set: 5010 patterns (501 per digit)

20

Configurations

- Original SmcGP implementation
- Original SmcGP + overfitting control *(validation set)*
- SmcGP + Embedding
- SmcGP + Embedding + overfitting control

The method has been implemented in Python using DEAP (Distributed Evolutionary Algorithms in Python)

21

Results *(best highlighted in yellow)*

		0		5		6		7		8	
		TNR(%)	TPR(%)	TNR(%)	TPR(%)	TNR(%)	TPR(%)	TNR(%)	TPR(%)	TNR(%)	TPR(%)
Standalone Classifier	Mean	99.45	93.30	99.62	88.66	99.13	86.83	99.65	93.25	99.30	89.62
	St. Dev.	0.41	4.77	0.13	1.07	0.57	5.10	0.08	**0.39**	0.29	1.64
	Best	99.78	97.01	99.76	89.42	99.51	**95.61**	99.67	94.01	99.45	**91.62**
	Worst	98.76	85.43	99.38	86.63	98.23	80.64	99.49	93.01	99.38	86.63
Standalone Classifier + Overfitting Control	Mean	99.25	88.70	99.29	87.90	98.64	83.91	99.60	93.90	99.41	86.35
	St. Dev.	0.45	8.98	0.21	1.32	0.38	4.48	0.14	0.43	0.14	**0.96**
	Best	99.71	**97.41**	99.27	90.02	98.71	92.22	99.67	94.41	99.60	88.02
	Worst	98.78	88.02	99.07	86.03	97.98	79.04	99.31	**93.21**	99.33	85.63
Embedding	Mean	99.26	91.89	99.61	89.30	98.97	85.75	99.61	93.61	99.34	89.14
	St. Dev.	0.37	4.11	**0.09**	1.62	0.34	3.00	**0.07**	1.42	0.09	1.18
	Best	99.76	96.21	99.56	91.82	99.47	90.62	99.67	95.21	99.51	91.02
	Worst	99.05	86.03	**99.49**	87.62	98.54	82.63	99.53	91.62	99.38	87.62
Embedding + Overfitting Control	Mean	**99.82**	**96.41**	**99.65**	**90.86**	**99.40**	**92.41**	**99.66**	**94.61**	**99.57**	**89.74**
	St. Dev.	**0.07**	**1.06**	0.22	**0.94**	**0.15**	**0.43**	0.15	1.15	**0.08**	1.07
	Best	**99.93**	97.21	**99.84**	**92.02**	**99.69**	93.01	**99.89**	**96.21**	**99.65**	91.22
	Worst	**99.87**	**95.81**	99.20	**89.62**	**99.45**	**91.82**	**99.60**	92.61	**99.58**	**88.02**

How can we move further on?

- Similar ideas could be applied to more traditional GP trees with continuous inputs and outputs: a single embedding would not suffice, though.

  Possible solutions:

  - Incremental evolution of embeddings: first embedding as shown, subsequent embeddings added and optimized with respect to the classifier AND the embedding elements previously computed.
  - GP-based autoencoders
  - Use of convolutional layers where GP play the role of non-linear convolutional kernels/functions, multiple trees, multiple layers, etc.

23

GP2SO: modeling and embedding 1-D signals using parametric symbolic regression

Magnani, G., Mordonini, M., Cagnoni S. *Hybrid GP/PSO Representation of 1-D Signals in an Autoencoder Fashion* (Proc. WIVACE 2023, Springer CCIS, vol. 1977, 2024)

GP-based parametric symbolic regression actually generates autoencoders for function/signal families sharing a common parametric model:

- Each function/signal instance I_i is encoded into the parameters P_i (ENCODER)
- Plugging the parameters P_i into the model reconstructs the original instance I_i (DECODER)

24

GP2SO: modeling and embedding 1-D signals using parametric symbolic regression

Consider a set of 1D functions (e.g., time series) resulting from deforming, shifting, scaling, etc… a basic function that can be expressed as a single parametric model.

A very familiar case: a set of Gaussians having different mean, standard deviation and amplitude, i.e.,

$$\{ K_i\, G(x,\mu_i,\sigma_i),\ i = 1..N \}$$

where $\kappa_i \equiv (K_i, \mu_i, \sigma_i)$ are constants and G a Gaussian kernel

Infinite possible Gaussians share the same equation and parameters, whose values characterize their shape and position.

25

GP2SO: modeling and embedding 1-D signals using parametric symbolic regression

One could think of adapting GP such that it finds A SINGLE symbolic representation for the whole family of functions, i.e., what is called a *basis function*, having a few free parameters, setting whose values one could obtain a good approximation of any function in the data set.

This means we want to find f_{GP} such that

$$f_{GP}(x, \kappa_1, \kappa_2, .. \kappa_n)) \approx K\, G(x,\mu,\sigma)$$

NB In the most ideal case (a noiseless signal), a general model could be evolved as the GP-based symbolic regression of a single instance from the function family

26

GP2SO: modeling and embedding 1-D signals using parametric symbolic regression

This way, we can represent each function in our data set only by the values $\{\kappa_i\}$ of the free parameters.

Thus, the parameters that, given the model, solve the regression problem for a function in the family:

- represent an embedding for that function
- Permit recovering the function starting from the GP encoding

This is exactly what autoencoders do!

- *Could GP hybridized with GA, PSO, or any other handy parameter optimizer (gradient descent?) be the solution?*
- *Could we apply it to any set of functions, having as a secondary goal to minimize the embedding size?*

27

GP2SO implementation

- GP is good at performing symbolic regression
- PSO is good and fast at parameter fitting (and VERY GPU-friendly!)

We can consider a hybrid autoencoder-like approach in which

- GP finds the function family expression and
- PSO fits the free parameters to represent/reconstruct each specific function
- The code can be easily and effectively optimized for GPUs

28

GP2SO implementation

$f_{GP}(x, \kappa_1, \kappa_2, \dots \kappa_n)) \approx K\, G(x,\mu,\sigma) = f(x,\mu,\sigma)$

This means:

- The terminal set T includes the independent variable x and the free parameters
- The fitness of a tree representing a possible f_{GP} is computed by an external optimizer (PSO) that finds the values of $\kappa_1, \kappa_2, \dots, \kappa_N$ minimizing the target function

$$\text{Fitness} = d\,(f_{GP}, f)$$

that can be expressed, for instance, as the total squared error over a sampling of f

29

GP2SO: Preliminary results

Using the sampling of three Gaussians as input, and adding to the terminal set 6 free parameters, $\kappa_{1, \dots,} \kappa_6$, that underwent PSO, we evolved the following model:

$$f_{GP}(x, \boldsymbol{\kappa}) = \frac{\kappa_1}{\kappa_2 + \kappa_3 (x-\kappa_4)^2 + \kappa_5 (x-\kappa_4)^4 + \kappa_6 (x-\kappa_4)^8}$$

TEST RESULTS

Generating the sampling of 300 Gaussians G_i (200 samples each, with x in [-10, 10]) and reconstructing (representing) such functions as the optimal vector $\boldsymbol{\kappa}_i$ computed by PSO, we obtained approximations with a total squared error TSE always < 0.13, with TSE < 0.001 for 95.34% of the instances)

30

Analysis of the parameter space

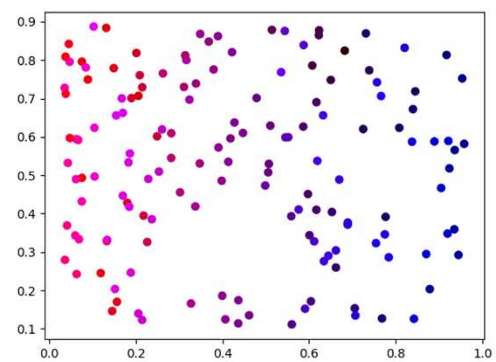
The embedding is optimal with respect to signal reconstruction.

- Is it also good for classification?
- Can we use this approach for feature extraction/construction?

We plotted the parameter sets obtained for the 300 Gaussians on the (μ,σ) coloring the dots according to the first T-sne component (Red channel) and the second T-sne component (Blue channel) of each Gaussian.

31

Clustering of Gaussians



(x,y) Coordinates

X = Mean

Y = St. Dev.

Color

R: 1st T-sne coordinate

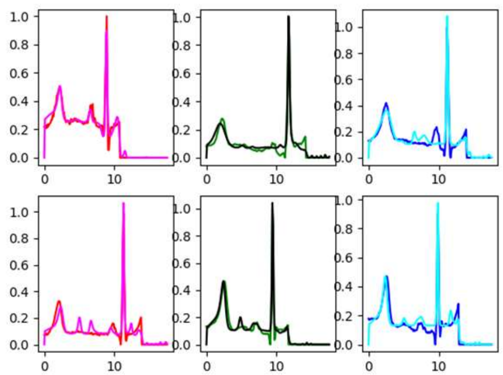
B: 2nd T-sne coordinate

NB All values normalized between 0 and 1

Near points have similar colors, i.e., they are similarly embedded

32

A real-world example: embedding of ECG signals



R, G, B: real ECG
M,B,C: corresponding reconstructed ECG

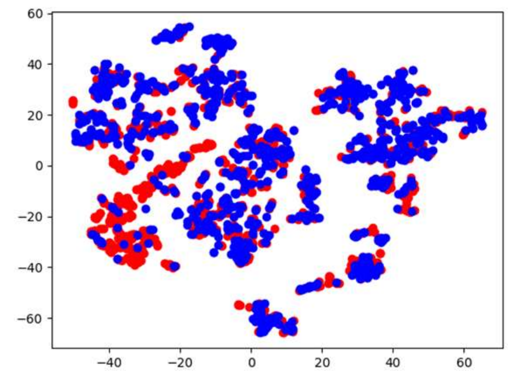
Both training and test ECG signals are reconstructed well.

Is this result equally good for classification?

Same model, different parameters for each signal

33

A real-world example: embedding of ECG signals



Unfortunately NOT!

(x,y) Coordinates

X = 1st T-sne coordinate

Y = 2nd T-sne coordinate

Color

R: class A

B: class B

34

Possible problems: Redundancy

With more extended function sets and more complex trees (including, for instance, periodic functions) some problems with parameter redundancy (existence of several parameter sets mapping the same function) may occur.

Remedy:
add a regularization term W_R (sum of the parameters' absolute values) and a size penalty W_S to the fitness function measuring the reconstruction error:

Fitness' = Fitness + W_S * size + W_R * S_i abs(k_i)

35

Possible problems: Reconstruction-oriented representation

Representations optimal for reconstruction may not be as good for classification (e.g., PCA).

- Often true in medicine when signal anomalies' energy is much smaller than the total signal energy.

Possible remedies:

- Different, more scale-independent distance measures as fitness (e.g., cosine similarity)
- In classification problems, adding statistics about the residuals of the reconstruction, followed by further feature selection to identify the most relevant components.

36

Possible problems: Computational Efficiency

GP2SO is not intrinsically efficient, requiring an entire PSO run for each GP fitness case evaluation during training, and another for encoding each new unseen data instance during inference

Remedy:
PSO search is both fast and easy to implement on GPUs with speed-ups easily reaching two orders of magnitude over a single-thread implementation

Conclusions (GP2SO)

GP2SO can obtain latent representations of complex data.

Pros and cons:

+ Possibility of learning from very few examples:
  very appealing for problems like medical diagnosis, etc., in which only few data are available.

- The need to run PSO for each pattern to be learned or transformed is computationally heavy

- Some problems (by now…) with classification.

Credits

The work described has been developed as their B.Eng. theses or Machine Learning project by

Fabrizio De Santis, M.Eng.
Dario Cavalli, M.Eng.
Giorgia Tedaldi, M.Eng.
Federico Brandini, M.Eng.,
Federico Sello, B.Eng.

Many thanks also to

Andrea Bettati, M.Eng,
Marco Carraglia, M.Eng,
Natalia Teresa Mazzara, M.Eng.
Leonardo Miccoli, M. Eng.

for contributing to the development/debugging of SmcGP in DEAP.

Part II: Evolutionary Neural Architecture Search for IASP and Pattern Recognition

Introduction

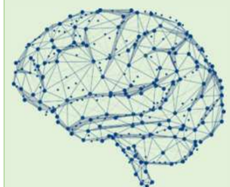
Artificial Intelligence

Mimicking the Intelligence or behavioural pattern of humans or any other living entity.

Machine Learning

A technique by which a computer can learn from data, without using a complex set of different rules.

Deep Learning



A technique to perform machine learning inspired by our brain's own network of neurons.

1950's · 1960's · 1970's · 1980's · 1990's · 2000's · 2010's · 2017's · now

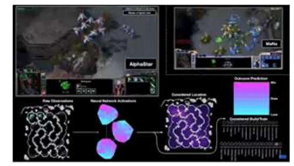
Introduction

FIELD

SPEECH	VISION	NATURE LANGUAGE PROCESSING
• Automatic Speech Recognition • Voice Assistant (Siri)	• Image Recognition • Automatic Drive • Face Recognition	• Text Analysis • Text Translation • Chatting Robot

APPLICATION

Object Detection



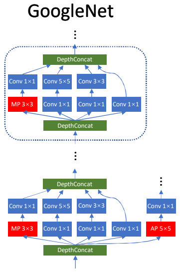
Alpha Star

Automatic Drive

42

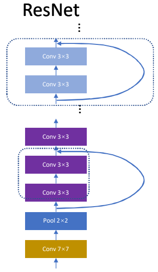
Introduction

GoogleNet



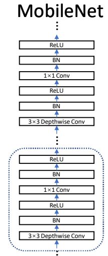
2012
- AlexNet
- NIN
- VGG
- GoogleNet
- …

ResNet



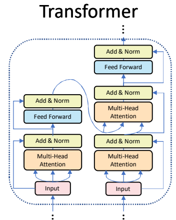
2015
- Highway Networks
- ResNet
- DenseNet
- Dual Path Network
- …

MobileNet



2017
- MobileNet(V1-V3)
- CondenseNet
- ShuffleNet(V1,V2)
- GhostNet
- EfficientNet
- …

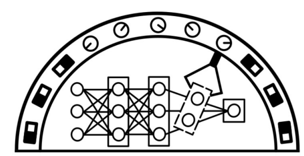
Transformer



2022
- Synthesizer
- Transformer-XL
- Longformer
- Reformer
- Axial Transformer
- …

43

Introduction



Concept diagram of NAS

NAS can be modeled by an optimization problem

$$\begin{cases} \arg\min_A = \mathcal{L}(A, \mathcal{D}_{train}, \mathcal{D}_{valid}) \\ s.t. \quad A \in \mathcal{A} \end{cases}$$

$\mathcal{A}$: the search space of the neural architectures
$\mathcal{L}(\cdot)$: measures the performance of the architecture A

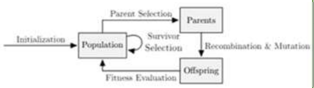
➢ Neural Architecture Search (NAS) aims to **automatically design** high-performance deep neural network architecture without relying on expert experience.

■ NAS algorithms can be broadly classified into three different categories based on **optimizer** methods:
- Reinforcement Learning-based NAS
- Gradient-based NAS
- Evolutionary Computation-based NAS (ENAS)

44

Introduction

- **Evolutionary Computation-based NAS(ENAS)**



- Insensitiveness to the local minima
- No requirement to gradient information

VS

- Reinforcement Learning-based NAS
  - Often find the ill-conditioned architectures
  - Not completely automatic
  - Usually requires much computational resources
- Gradient based NAS
  - Often find the ill-conditioned architectures
  - Not completely automatic
  - Often require to construct a supernet in advance, which also highly requires expertise

45

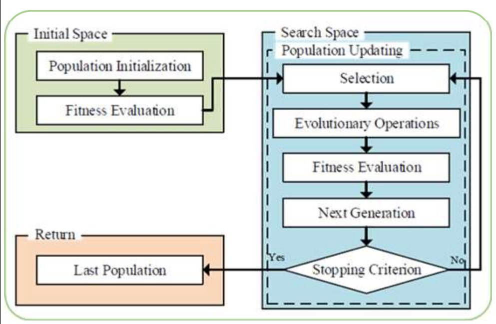
Introduction

History

- before1999: Evolutionary artificial neural networks (EANN)
  - Search for both the neural architectures and the optimal weight values
  - Apply to small-scale neuron networks
- 2000-2016: Neuroevolution
  - Search for both the neural architectures and the optimal weight values
  - Apply to median-scale neuron networks
- 2017-now: ENAS
  - Focus mainly on searching for the architectures of deep neural networks
  - Mainly convolutional neural networks

46

Background



The flowchart of a common ENAS algorithm.

Key issues in each step

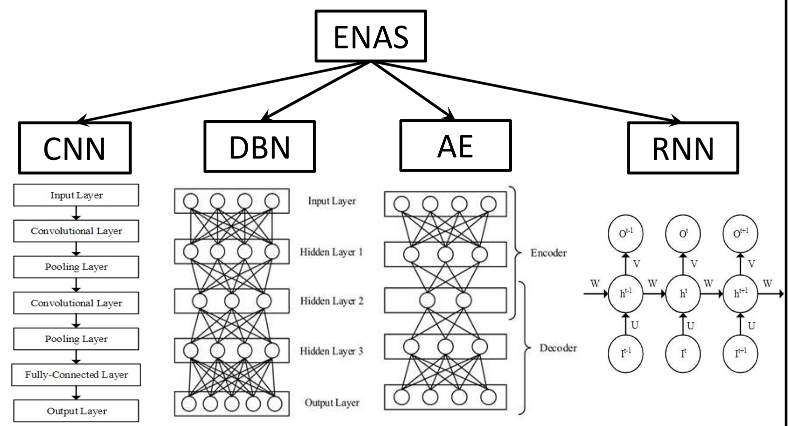
Population Initialization:
- Encoding space
- Encoding strategy

Population Updating:
- Evolutionary operators
- Selection strategy

Fitness Evaluation:
- Acceleration method

47

Background



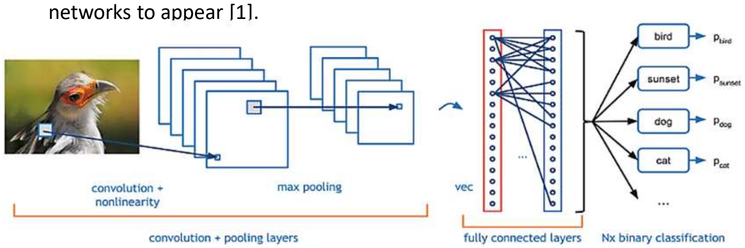
The categories of ENAS from neural network perspective

48

Convolutional Neural Network

■ The definition of Convolutional Neural Network (CNN)

– Convolutional Neural Network (CNN) is a class of Feedforward Neural Networks (FNN) with convolutional computation and deep structure.
– Research on convolutional neural networks started in the 1980s and 1990s, with time-delay networks and LeNet-5 being the first convolutional neural networks to appear [1].

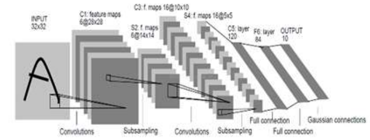


The application of CNNs

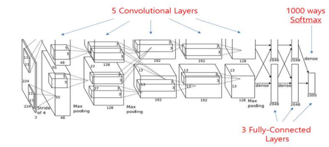
[1] LeCun Y, Bengio Y. Convolutional networks for images, speech, and time series[J]. The handbook of brain theory and neural networks, 1995, 3361(10): 1995.

Convolutional Neural Network

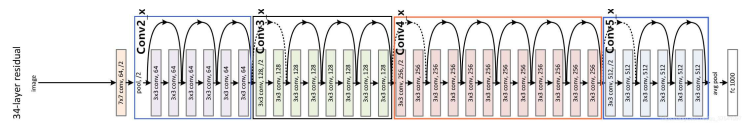
■Typical representative works of Convolutional Neural Networks



LeNet-5 1998



AlexNet 2012

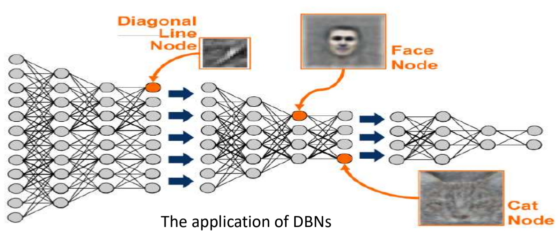


ResNet 2015

Deep Belief Network

■The definition of Deep Belief Network (DBN)

A DBN is a generative model which comprises of many layers of hidden units and is made up by stacking multiple Restricted Boltzmann Machines(RBMs). The belief neural network proposed by Neil in 1992 is different from the conventional FFNN. Hinton (2007) describes DBNN as "a probabilistic generative model consisting of multiple layers of random latent variables."



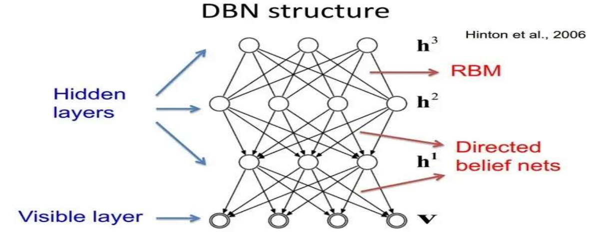
The application of DBNs

Deep Belief Network

■ The components of Deep Belief Network (DBN)

- The components of the DBN are Restricted Boltzmann Machines (RBM).
- The process of training a DBN is performed layer by layer.

DBN structure

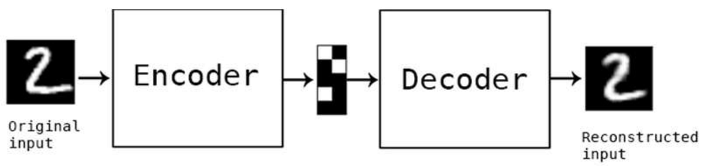


Auto-Encoder

■ The definition of Auto-Encoder (AE)

– Autoencoder (AE) is a class of Artificial Neural Networks (ANNs) used in semi-supervised and unsupervised learning.

– In 1994, Hinton and Richard S. Zemel constructed the first self-encoder-based generative model by proposing the "Minimum Description Length principle (MDL)".



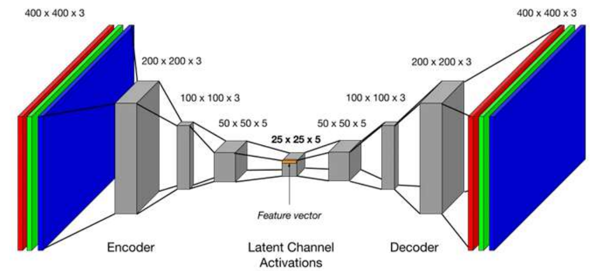
The application of AEs

Auto-Encoder

■ The components of Auto-Encoder (AE)

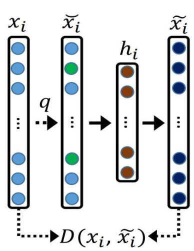
– An AE is typically composed of two symmetrical components: the encoder and the decoder.

– Common encoding parameters: number of hidden layers, neurons per layer.

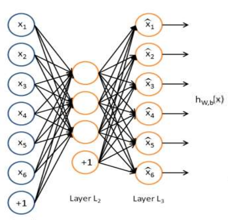


Auto-Encoder

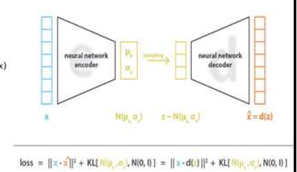
■ Typical representative works of Auto-Encoder (AE)



Denoising Autoencoder 2008



Sparse Autoencoder 2011

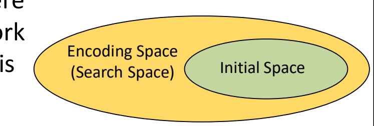


Variational Autoencoders 2016

Encoding Space and Initial Space

Initial space ⊆ Search space = Encoding space

- **Encoding space** means that where the potential deep neural network architecture will be searched, it is also called as "search space"
- **Initial space** means that where the first generation of population will be created



The relationship between encoding space, search space and initial space.

Encoding Space and Initial Space

➢ Taxonomy on **initial space**

- **The trivial space** contains only a few primitive layers and can evolve to a competitive architecture.
- **For a random space**, all the individuals in the initial population are randomly generated in the limited space, and it has been adopted by many methods.
- **The well-designed space** contains the state-of-the-art architectures. In this way, a promising architecture can be obtained, whereas it can hardly evolve to other novel architectures.

➢ Taxonomy on **encoding space**

☐ According to the **basic units** they adopt:
- Layer-based
- Block-based
- Cell-based
- Topology-based

☐ According to the **common constraints**:
- Fixed depth
- Rich initialization
- Partial fixed structure
- Relatively few constraints

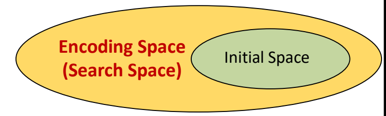
57

Encoding Space

The encoding space defines which architectures can be represented in principle.

☐ **Direct design**:
Pros. Encompass all network architectures
Cons. Be exponential and time-consuming
Application: Applied to unfamiliar scenarios

☐ **Combining prior knowledge:**
Pros. Effectively reduce the search space
Cons. Limits the network to learn structures
Application: Applied to familiar scenarios



The relationship between encoding space, search space and initial space.

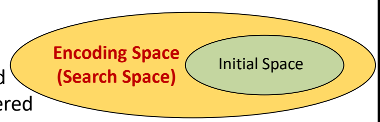
58

Encoding Space

The initial space is often a subspace of the encoding space and determines what kind of individuals may appear in the initial population.

☐ **Direct initialization:**
Pros. No relevant experience required
Cons. Novel structures can be discovered
Application: Applied to unfamiliar scenarios

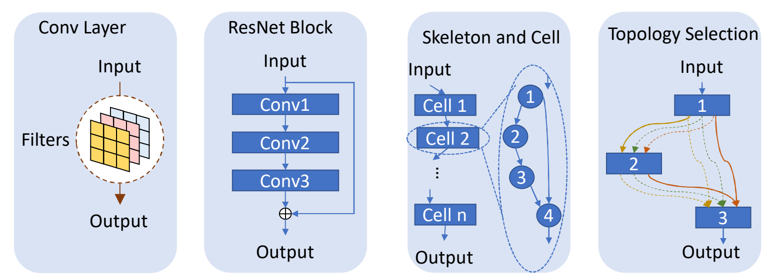
☐ **Combining prior knowledge:**
Pros. Contains the state-of-the-art architectures
Cons. Novel structures can not be discovered
Application: Applied to familiar scenarios



The relationship between encoding space, search space and initial space.

59

Basic Units of Encoding Space



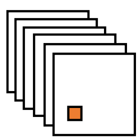
Layer-based	Block-based	Cell-based	Topology-based
Focus on the parameters in the layer (e.g. the filters)	Focus on the parameters of specific block (e.g. amount in ResNet Block)	The architecture is built by stacking the same cells	Concerned about the operations between nodes

60

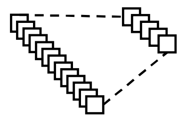
Basic Units of Encoding Space

The layer-based encoding space denotes that the basic units in the encoding space are the primitive layers, such as convolution layers and fully-connected layers.

- **Pros:** Lead to a huge search space, since it tries to encode so much information in the search space.
- **Cons:** Take more time to search for a promising individual because there are more possibilities to construct a well-performed DNN from the primitive layers



Convolution layer



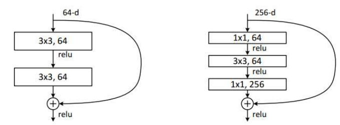
Fully-connected layer

61

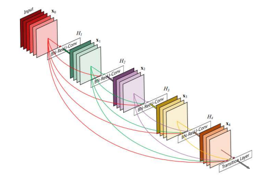
Basic Units of Encoding Space

The block-based encoding space is developed, where various layers of different types are combined as the blocks to serve as the basic unit of the encoding space.

- **Pros:** Have promising performance and often require fewer parameters to build the architecture.
- **Cons:** Still need to learn some parameters for each layer, which is time consuming.



ResBlock



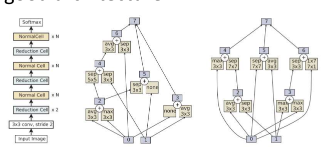
DenseBlock

62

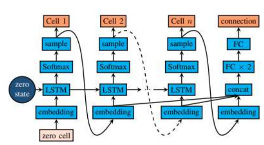
Basic Units of Encoding Space

The cell-based encoding space is similar to the block-based one, and it can be regarded as a special case of the block-based space where all the blocks are the same.

- **Pros:** Greatly reduce the size of the encoding space. All the basic units in the encoding space are the same and parameters in terms of constructing the promising DNN is much fewer.
- **Cons:** No theoretical basis for that the cell-based space can help to obtain a good architecture.



AmoebaNet-A architecture



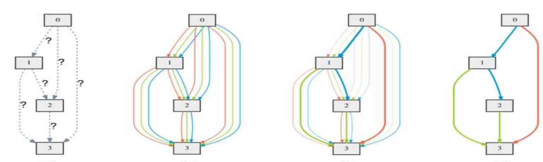
the micro and macro cell

63

Basic Units of Encoding Space

The topology-based space does not consider the parameters or the structure of each unit (layer or block), yet they are only concerned about the connections between units.

- **Pros:** Obtain a highly accurate and efficient neural network architecture and greatly reduce the search time and overhead
- **Cons:** Limitations on the representation of the neural network architecture



An overview of DARTS

64

Constraints on Encoding Space

The constraints on the encoding space are important, because the constraints represent the human intervention which restricts the encoding space and lighten the burden of evolutionary process.

- **Fixed depth**
  - A strong All the individuals in the population have the same depth.
  - constraint and largely reduces the encoding space.
- **Rich initialization**
  - A strong constraint with a lot of manual experience.
  - The initialized architectures are manually designed, which goes against the original intention of NAS.
- **Partial structure fixed**
  - The architecture is partially settled.

Constraints on Encoding Space

- **Fixed depth**

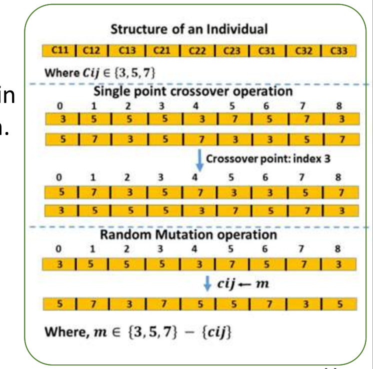
Keep the length of the chromosome unchanged, thus all the individuals in the population have the same depth.

- **Advantages:**

Conveniently employ the genetic operators of EC methods.

- **Disadvantages:**

Largely reduce the size of encoding space.



An example of fixed depth constraint [1].

[1] A. Singh, S. Saha, R. Sarkhel, M. Kundu, M. Nasipuri, and N. Das, "A genetic algorithm based kernel-size selection approach for a multi-column convolutional neural network," 2019, arXiv:1912.12405. [Online]. Available: http://arxiv.org/abs/1912.12405

Constraints on Encoding Space

- **Rich initialization**

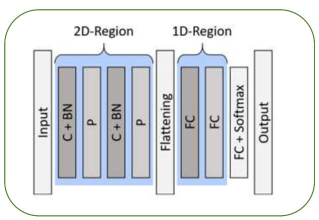
The well-designed encoding space, which usually requires a lot of expertise.

- **Advantages:**

Might achieve good performance on specific problems.

- **Disadvantages:**

It often require the algorithm designers have strong expertise of deep neural networks, and the designers may clear know the rough architecture for solving the problem at hand.



An example of encoding space with rich initialization constraint [1].

[1] F. M. Johner and J. Wassner, "Efficient evolutionary architecture search for CNN optimization on GTSRB," in Proc. 18th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA), Dec. 2019, pp. 56–61.

Constraints on Encoding Space

- **Partial structure fixed**

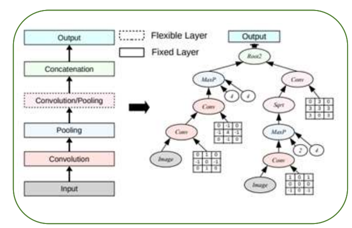
The architecture is partially settled.

- **Advantages:**

Allow algorithm designers to bring some of their expert knowledge into the Encoding.

- **Disadvantages:**

Partially reduce the size of encoding space.



An example of encoding space where partial structure are fixed[1].

[1] Y. Bi, B. Xue, and M. Zhang, "An evolutionary deep learning approach using genetic programming with convolution operators for image classification," in Proc. IEEE Congr. Evol. Comput. (CEC), Jun. 2019, pp. 3197–3204.

Encoding Strategy

The encoding strategy can be divided into two categories according to whether **the length of an individual changes or not** during the evolutionary process:

- **Fixed-length encoding strategy**
  - It is easy to take the use of standard evolutionary operations.
  - The maximal depth is limited in advance.
  - The maximal length is determined by experts because the optimal depth is unknown.
- **Variable-length encoding strategy**
  - It can contain more details of the architecture with more freedom of expression.
  - The neural architecture with the optimal depth which is unknown can be searched.
  - The evolutionary operators may be not suitable for this kind of encoding strategy and need to be redesigned.

69

Encoding Strategy

Fixed-length encoding strategy

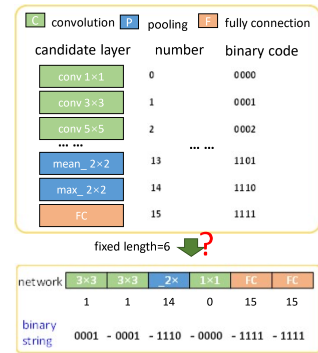
Generate individuals of fixed encoded length at initialization, and the individual lengths remain unchanged during the evolutionary process.

Advantage

- Facilitates the use of standard crossover and mutation operations
- Reduce search space size

Disadvantage

- Difficult to precisely predefine the optimal depth of the DNN
- Require rich domain knowledge from both encoding and DNN



An illustrative example of fixed-length encoding strategy.

70

Encoding Strategy

Variable-length encoding strategy

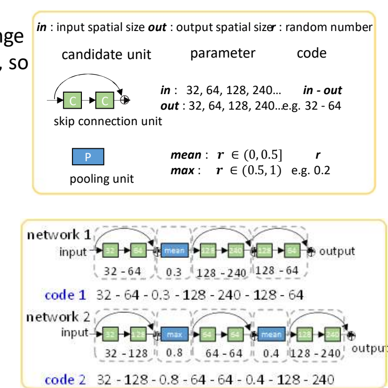
The coding length of individuals may change continuously during evolutionary process, so that its corresponding the network is not limited to a specific depth.

Advantage

- The optimal length of DNN can be obtained by searching
- Contain more details of the architecture with more freedom of expression

Disadvantage

- Redesign evolutionary operators which may not suitable for this kind of encoding strategy



An illustrative example of variable-length encoding strategy[1].

[1] Sun, Yanan, et al. "Automatically designing CNN architectures using the genetic algorithm for image classification." IEEE transactions on cybernetics 50.9 (2020): 3840-3853.

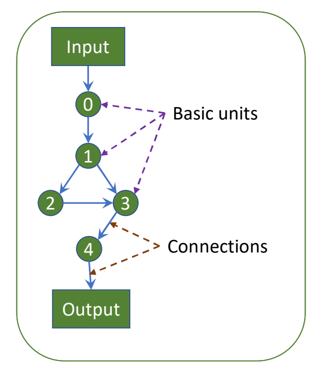
71

Encoding Strategy

Most of the neural architectures can be represented as **directed graphs**, which are made up of **different basic units** and **the connections** between the units.

The encoding for an architecture can be divided into two aspects:

- **Configurations of basic units**
  - Layers
  - Blocks
  - Cells
- **Connections**
  - Linear structure
  - Non-linear structure

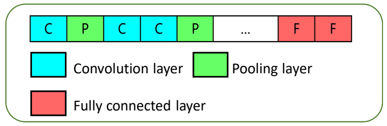


An illustrative example of a neural network represented as a directed graph.

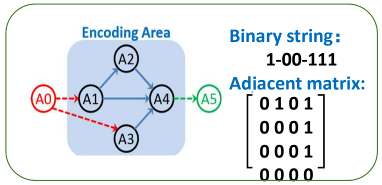
72

Encoding Strategy

- **Linear structure**: the basic units are stacked one by one to build up the skeleton of the architecture.
  - Its widespread use in ENAS stems from its simplicity.
- **Non-linear structure**: there are skip-connections or loop-connections in the architecture.
  - Adjacent matrix is the most popular way to represent the connections.



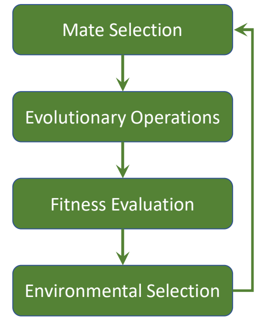
An illustrative example of linear structure[1].



An illustrative example of non-linear structure[2].

[1] Sun, Yanan, et al. "Evolving deep convolutional neural networks for image classification." *IEEE Transactions on Evolutionary Computation* 24.2 (2019): 394-407.
[2] Xie, Lingxi, and Alan Yuille. "Genetic CNN." *Proceedings of the IEEE international conference on computer vision*. 2017.

Population Updating - EAs



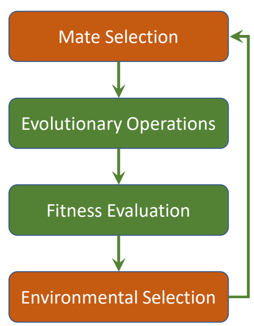
The flow chart of EAs in Population Updating.

- **Mate Selection**: the individuals with better fitness are selected by a selection algorithm to be the parents to produce offspring.
- **Evolutionary Operations**: the evolutionary operations, such mutation and crossover, are performed on the selected parents to produce new individuals.
- **Fitness Evaluation**: A fitness function is performed on the new generated individuals to compute their fitness.
- **Environment Selection**: A selection strategy is utilized like environment selection which chooses individuals based on their fitness to make up the next population.

Population Updating - EAs

Selection strategy:

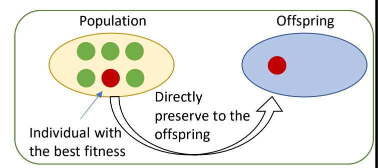
- mate selection and environmental selection
  - Elitism
  - Roulette
  - Discard the worst or the oldest
  - Tournament selection
  - Others



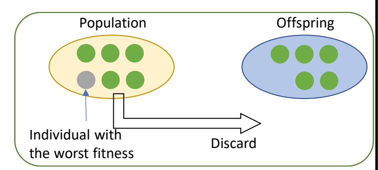
The selection strategy is used in mate selection and environmental selection.

Population Updating - EAs

- **Elitism**: the simplest strategy that keeps the individuals with higher fitness.
  - It can cause a loss of diversity in the population, which may lead the population falling into local optima.
- **Discard the worst or the oldest**: discarding the worst is similar to elitism, which removes the individuals with poor fitness values from the population.
  - Discards the oldest is also called aging evolution, which can explore the search space more, instead of zooming in on good models too early.



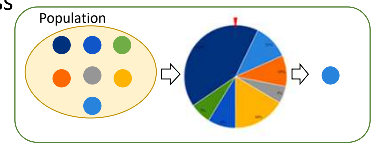
An illustrative example of Elitism.



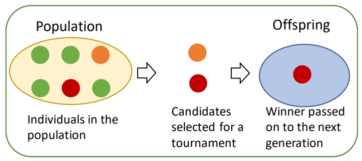
An illustrative example of Discard the worst.

Population Updating - EAs

- **Roulette**: gives every individual a probability according to its fitness among the population to survive (or be discarded), regardless it is the best or not.
  - The individuals with better fitness have a higher probability to be selected, and the individuals with low fitness also have a chance of being selected.
- **Tournament selection**: selects the best one from an equally likely sampling of individuals.
  - The worst individual never survives, while the best individual wins all tournaments in which it participates



An illustrative example of Roulette.



An illustrative example of Tournament selection.

77

Population Updating - EAs

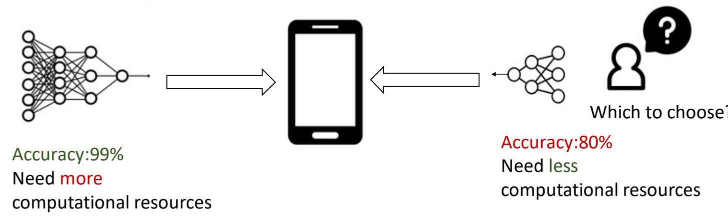
Single objective

Only consider one indicator of neural architecture such as the performance value

(e.g. only searching for the architecture with the highest classification accuracy)

Problems:

- Cannot find an architecture that can achieve the best in all objectives, some compromise architectures are need.



78

Population Updating - EAs

Multi-objective

Performance of the neural network and **the number of parameters** are considered simultaneously

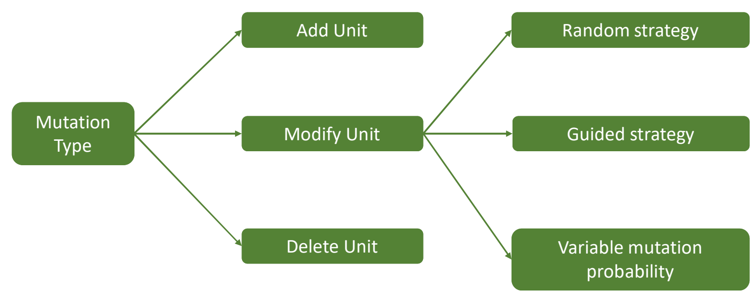
Solutions:

- converting it into a single objective optimization problem with weighting factors (i.e. the weighted summation method)
- directly address it through some famous multi-objective optimization algorithms
  - NSGA-II
  - NSGA-III
  - MOEA/D

79

Population Updating - EAs

Mutation and **Crossover** operators are two of the most commonly used evolutionary operations in EAs.

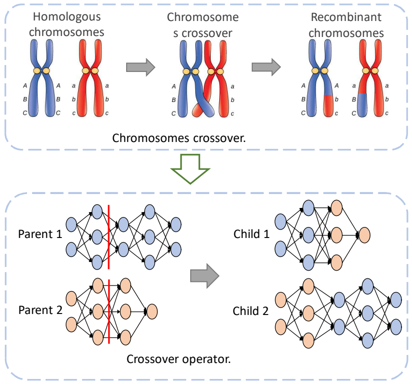


80

Population Updating - EAs

Crossover

- The crossover operation is inspired by the crossover phenomenon of chromosomes in biology. The chromosomes of two parents cross and exchange equal segments between non sister chromatids in the genetic process to generate two new chromosomes. At the same time, the probability of chromosomes crossover is generally high.



Chromosomes crossover.

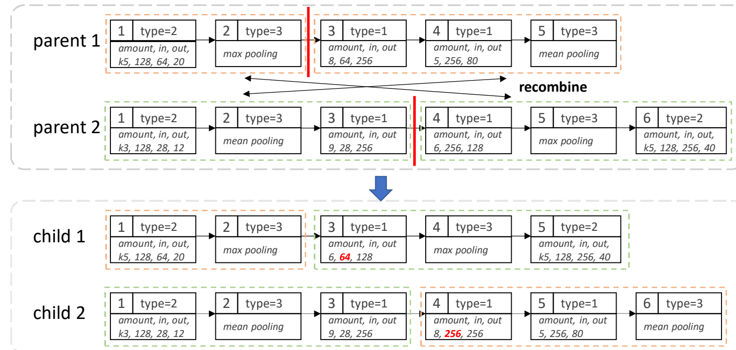
Crossover operator.

Common crossover operator

- Cluster crossover
- internal crossover

Population Updating - EAs

Cluster crossover: One or more cross points are randomly generated in the coding strings of two parent individuals, and then gene exchange is conducted at the control position of two individuals.



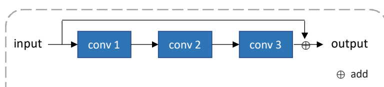
An example of cluster crossover [1].

[1] Sun, Yanan, et al. "Completely automated CNN architecture design based on blocks." IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 4, pp. 1242-1254, 2020.

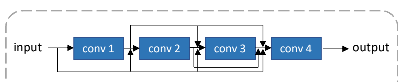
Population Updating - EAs

Detail of basic unit

- **number:** the position of unit in network
- **type:** different types of unit
  - 1: ResNet Block unit
  - 2: DenseNet Block unit
  - 3: Pooling layer unit
- **parameters:** parameter configuration for each unit



An example of RB.



An example of DB including four convolutional layers.

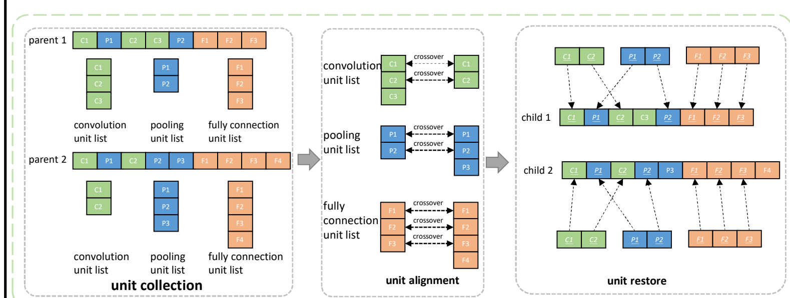
ResNet Block unit
- amount: the number of RBs
- in: input spatial size
- out: output spatial size

DenseNet Block unit
- amount: the number of DBs
- in: input spatial size
- out: output spatial size
- k: growth rate of spatial size per layer

Pooling layer unit
- one PU consisting of a single pooling layer
- pooling layer types: max/ mean pooling

Population Updating - EAs

Internal crossover: Each type of unit is collected from the individual and stacked in the order of the individual. Align the units of same type in two individuals, pair the units at the same and perform the crossover operation. Then restore the unit lists to generate two new individuals.



An example of internal crossover [1].

[1] Sun, Yanan, et al. "Evolving deep convolutional neural networks for image classification." IEEE transactions on evolutionary computation 24.2 (2017): 394-406.

Population Updating - EAs

Mutation

The mutation operation of genetic algorithm is inspired by chromosomes variation.

- Mutation operations may perform on each position of the units from one individual.

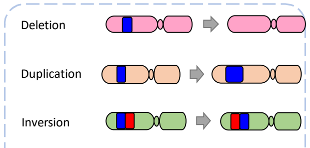
Mutation can be defined as small random adjustments in chromosomes to obtain new solutions.

Common mutation strategy

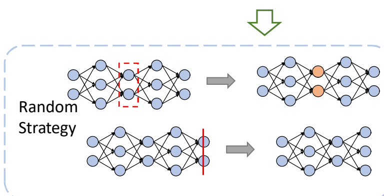
- Random strategy
- Gaussian strategy
- RNN based strategy

Advantage

- Make EAs have global random search ability
- Maintain population diversity



Chromosomes variation.



Mutation operator.

85

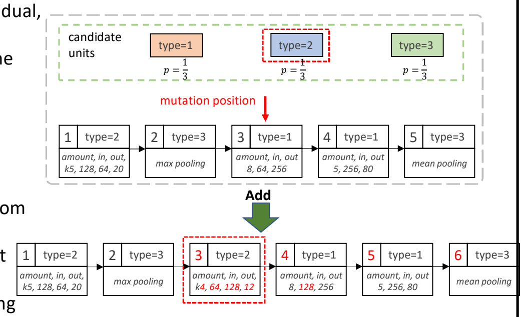
Population Updating - EAs

Random strategy

A mutation position is randomly selected in the current individual, and one particular mutation operation is selected from the mutation list with a identical probability.

Common mutation operations

- **Add** (add a unit with random parameter settings)
- **Delete** (remove the unit at the selected position)
- **Modify** (randomly changing the parameter values of the unit at the selected position)

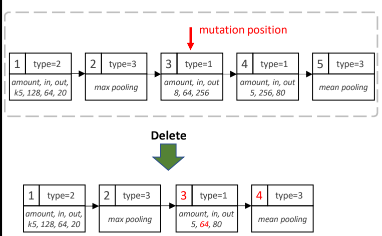


A example of add operator [1].

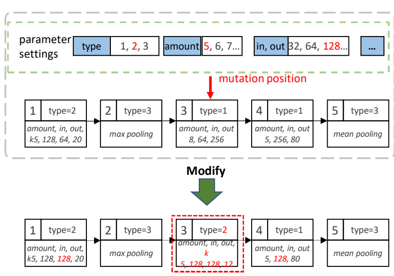
[1] Sun, Yanan, et al. "Completely automated CNN architecture design based on blocks." IEEE transactions on neural networks and learning systems 31.4 (2019): 1242-1254.

86

Population Updating - EAs



A example of delete operator [1].



A example of modify operator [1].

[1] Sun, Yanan, et al. "Completely automated CNN architecture design based on blocks." IEEE transactions on neural networks and learning systems 31.4 (2019): 1242-1254.

87

Efficient Evaluation

- It will take about 32 minutes to train a neural network to convergence on the TPU v2 accelerator which is the ultra high-performance hardware[1], not to mention training hundreds or thousands of neural networks in ENAS.
- Examples:
  - Large-scale Evo algorithm[2] use 250 GPUs for 11 days.
  - AmoebaNet[3] which takes the use of 450 GPUs for 7 days.

Such computational resources are not available for everyone interested in NAS.

[1] Ying, Chris, et al. "Nas-bench-101: Towards reproducible neural architecture search." International Conference on Machine Learning. PMLR, 2019.
[2] Real, Esteban, et al. "Large-scale evolution of image classifiers." International Conference on Machine Learning. PMLR, 2017.
[3] Real, Esteban, et al. "Regularized evolution for image classifier architecture search." Proceedings of the aaai conference on artificial intelligence. Vol. 33. No. 01. 2019.

88

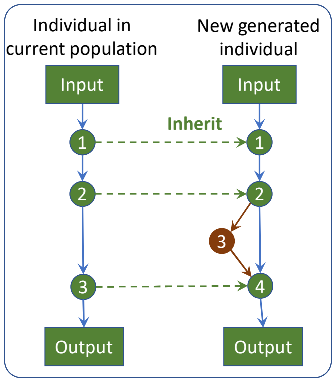
Efficient Evaluation

- Due to the evaluation is the most **time-consuming** stage, the strategies to improve the efficiency of evaluation will be discussed.
- Five of the most common **methods to shorten the time**:
  - Weight inheritance
  - Early stopping policy
  - Reduced training set
  - Population memory
  - Performance predictor

89

Weight Inheritance

- The evolutionary operators usually **do not completely disrupt** the architecture of an individual. → Some parts of the new generated individual are the same with previous individuals.
- The ultimate weight inheritance let the new individual **completely inherit the knowledge its parent learned** and training such an individual to convergence will save a lot of time.

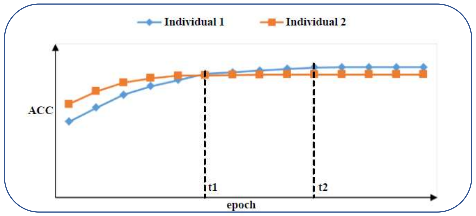


An example of weight inheritance.

90

Early Stopping Policy

- The simplest way is to set a **fixed** relatively small number of training epochs.
- Disadvantages: The early stopping policy can lead to **inaccurate estimation** about individual performance (especially the large and complicated architecture).



An example of inaccurate estimation about individual performance using early stopping policy.

91

Reduced Training Set

- Using a subset of that data has similar properties to a large dataset can also shorten the time effectively.
- The smaller dataset can be regarded as the proxy for the large one, e.g. CIFAR-10 and ImageNet.

CIFAR-10, 10 classes

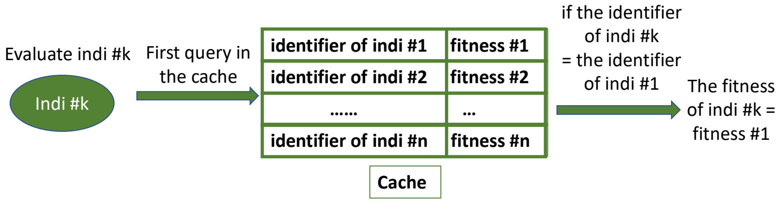
ImageNet, 1000 classes

CIFAR-10 and ImageNet.

92

Population Memory

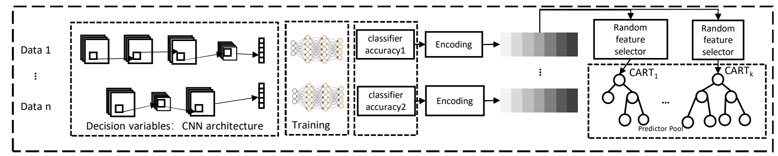
- The population memory is a unique acceleration method of ENAS.
- It works by **reusing the corresponding architectural information** that has previously appeared in the population.



An illustrative example of population memory.

Performance Predictor

- Performance predictor **directly maps the architecture and its performance** by using a regression model.
- Advantages: can **effectively evaluate the architecture**.



An illustrative example of performance predictor (E2EPP[1]).

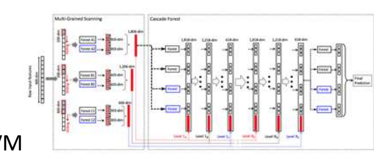
[1] Sun, Yanan, et al. "Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor." IEEE Transactions on Evolutionary Computation 24.2 (2019): 350-364.

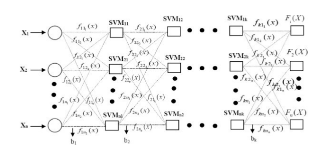
Part III: Evolutionary Deep Learning based on Genetic Programming for IASP and PR

Non-NN-based Deep Learning

Can $f(x)$ be other format to achieve deep learning?

- Deep Forest: decision tree
- PCANet: PCA filters
- Deep Support Vector Machine: SVM
- Genetic Programming based Deep Structures/Learning
- ......

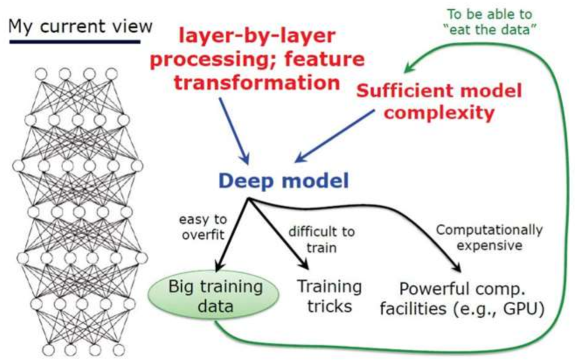




- Zhi-Hua Zhi, and Ji Feng. "Deep forest." *National science review* 6, no. 1 (2019): 74-86.
- Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma. "PCANet: A simple deep learning baseline for image classification?." *IEEE transactions on image processing* 24, no. 12 (2015): 5017-5032.
- Onuwa Okwuashi, and Christopher E. Ndehedehe. "Deep support vector machine for hyperspectral image classification." *Pattern Recognition* 103 (2020): 107298.

Deep Learning (Prof. Zhihua Zhou)

Layer-by-layer Processing | Feature Transformation | Sufficient model Complexity



97

Why Genetic Programming?

① **Flexible variable-length** representation

② **GP is a learning algorithm** that automatically learns model structures and coefficients

—a model can be a feature, a set of features, a classifier, a rule, or an ensemble ......

③ Perform multiple tasks using a single tree/program

④ Easy to have deep structures and complex functions as nodes

⑤ Potential interpretability (understandability)

Other advantages: population-based beam search, non-differential objective functions,
ease of cooperating with domain knowledge

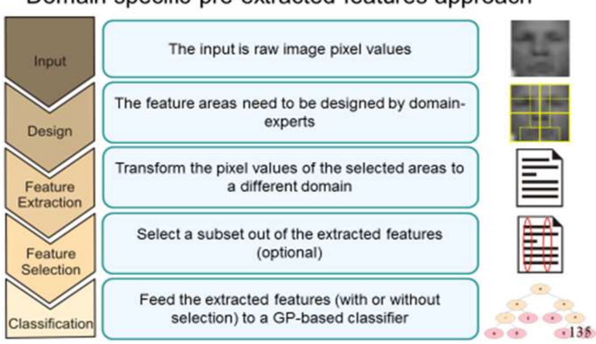
98

3-Tier/2-Tier GP

GP for Image Recognition/Classification

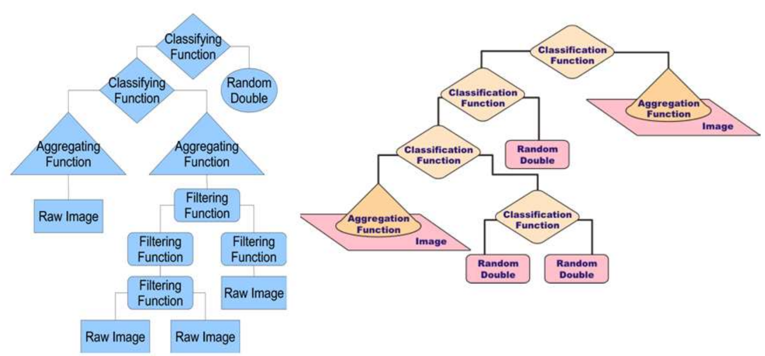
The traditional way

Domain-specific pre-extracted features approach



99

3-Tier/2-Tier GP

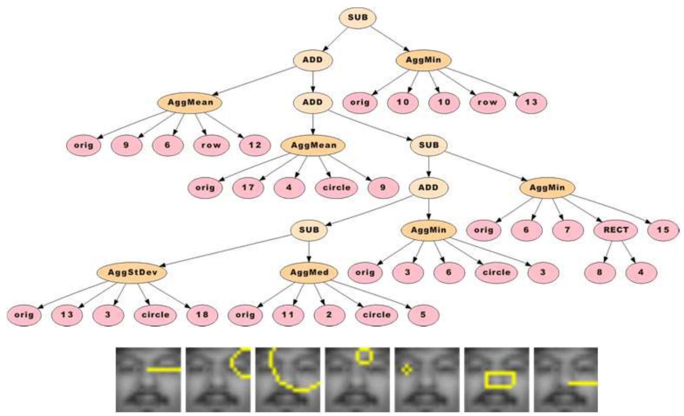


Daniel Atkins, Kourosh Neshatian and Mengjie Zhang. "A Domain Independent Genetic Programming Approach to Automatic Feature Extraction for Image Classification". Proceeding of the 2011 IEEE Congress on Evolutionary Computation. IEEE Press. New Orleans, USA. June 5-8, 2011. pp. 238-245.

Harith Al-Sahaf, Andy Song, Kourosh Neshatian, Mengjie Zhang. "Two-Tier Genetic Programming Towards Raw Pixel Based Image Classification". *Expert Systems With Applications*. **Vol. 39**, Issue 16. 2012. pp. 12291-12301

100

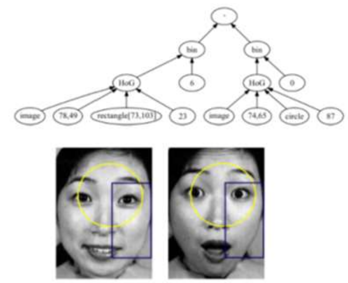
2-Tier GP (2012)



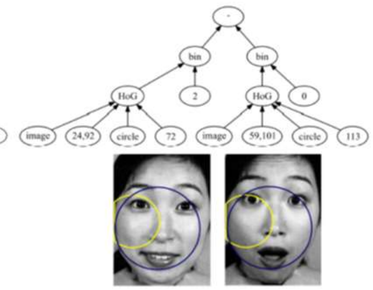
101

GP-HoG [2015-16]

- The below tree has 98% training and 95% test performance on the Jaffe dataset despite being very simple.

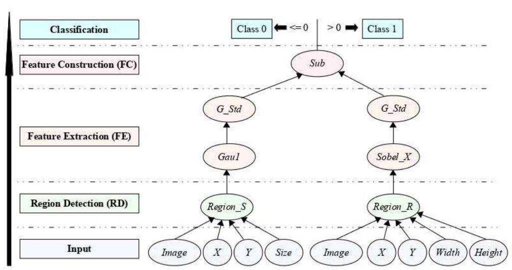


- The below tree has 95% training and 100% test performance on the Jaffe dataset.



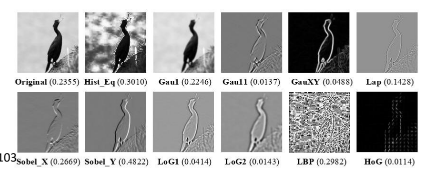
102

MLGP: An Automatic Feature Extraction Approach to Image Classification Using Genetic Programming



Five layers:

1. Input layer
2. Region detection layer
3. Feature extraction layer
4. Feature construction layer
5. Classification layer

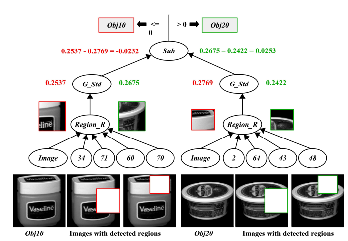


Ying Bi, Bing Xue, Mengjie Zhang. An Automatic Feature Extraction Approach to Image Classification Using Genetic Programming. Proceeding of the 21th Europen Conference on Applications of Evolutionary Computation (EvoApplications 2018). Lecture Notes in Computer Science. Parma, Italy. 4-6 April 2018. pp. 421-438.

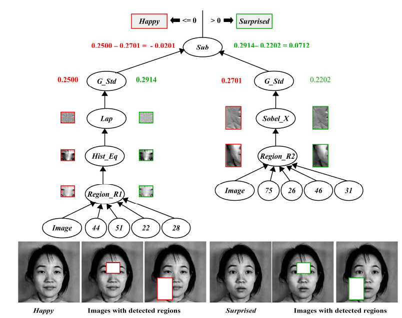
103

Example Solutions

- An example solution on object images

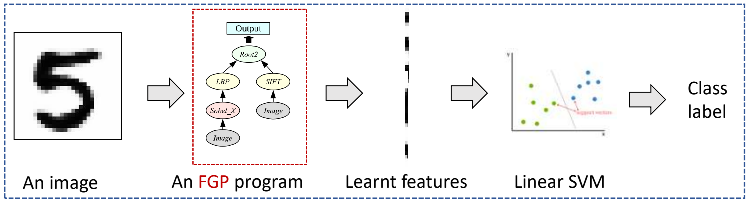


- An example solution on face images



104

FGP: Genetic Programming with A Flexible Program Structure and Image-Related Operators for Feature Learning to Image Classification



- The complexity of the FGP solutions for different tasks can be various
- The FGP method can learn various types and numbers of effective features from raw images
- FGP can be easily applied to different types of image classification tasks to achieve good classification performance
- The evolved solutions of FGP can be easily visualised, which provide more insights on the tasks

Ying Bi, Bing Xue, Mengjie Zhang. Genetic Programming with Image-Related Operators and a Flexible Program Structure for Feature Learning in Image Classification. *IEEE Transactions on Evolutionary Computation*. **Vol. 25**, Issue 1. 2021. pp. 87 - 101. DOI: 10.1109/TEVC.2020.3002229

Experimental Results

☐ Classification error rates of the proposed FGP method

Methods	MB	MRD	MBR	MBI	Rectangle	RI	Convex
SVM+RBF [30]	3.03(+)	11.11(+)	14.58(+)	22.61(+)	2.15 (+)	24.04(+)	19.13(+)
SVM+Poly [30]	3.69(+)	15.42(+)	16.62(+)	24.01(+)	2.15(+)	24.05(+)	19.82(+)
SAE-3 [29]	3.46(+)	10.30(+)	11.28(+)	23.00(+)	2.14(+)	24.05(+)	–
DAE-b-3 [29]	2.84(+)	9.53(+)	10.30(+)	16.68(+)	1.99(+)	21.59(+)	–
CAE-2 [29]	2.48(+)	9.66(+)	10.90(+)	15.50(+)	1.21(+)	21.54(+)	–
SPAE [44]	3.32(+)	10.26(+)	9.01(+)	13.24(+)	–	–	–
RBM-3 [29]	3.11(+)	10.30(+)	6.73(+)	16.31(+)	2.60(+)	22.50(+)	–
ScatNet-2 [27, 28]	1.27(+)	7.48(+)	12.30(+)	18.40(+)	0.01(+)	8.02(+)	6.50(+)
RandNet-2 [28]	1.25(+)	8.47(+)	13.47(+)	11.65(+)	0.09(+)	17.00(+)	5.45(+)
PCANet-2(softmax) [28]	1.40(+)	8.52(+)	6.85(+)	11.55(+)	0.49(+)	13.39(+)	4.19(+)
LDANet-2 [28]	**1.05**	7.52(+)	6.81(+)	12.42(+)	0.14(+)	16.20(+)	7.22(+)
NNet [30]	4.69(+)	18.11(+)	20.04(+)	27.41(+)	7.16(+)	33.20(+)	32.25(+)
SAA-3 [30]	3.46(+)	10.30(+)	11.28(+)	23.00(+)	2.41(+)	24.05(+)	18.41(+)
DBN-3 [30]	3.11(+)	10.30(+)	6.73(+)	16.31(+)	2.60(+)	22.50(+)	18.63(+)
FCCNN [25]	2.43(+)	8.91(+)	6.45	13.23(+)	–	–	–
FCCNN (with BT) [25]	2.68(+)	9.59(+)	6.97(+)	10.80(+)	–	–	–
SPCN [26]	1.82(+)	9.81(+)	**5.84**	9.55(+)	0.19(+)	10.60(+)	–
FGP(best)	1.18	**7.37**	6.54	**7.48**	**0.00**	**6.10**	**1.54**
FGP(mean)	1.30	8.44	7.34	10.35	0.12	7.34	1.84
FGP(std)	0.06	0.6	0.42	1.41	0.11	0.61	0.19
Rank	2/18	**1/18**	3/18	**1/18**	**1/15**	**1/15**	**1/10**

IEGP: Genetic Programming with A New Representation to Automatically Learn Features and Evolve Ensembles for Image Classification

Traditional Ensemble Methods for Image Classification

Raw Images → Feature Extraction → Base Classifiers → Class labels

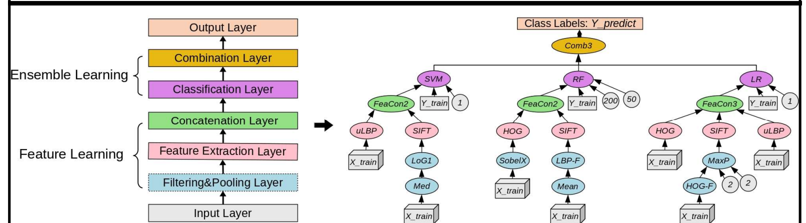
The New Approach for Image Classification

Raw Images → An IEGP solution → Class labels

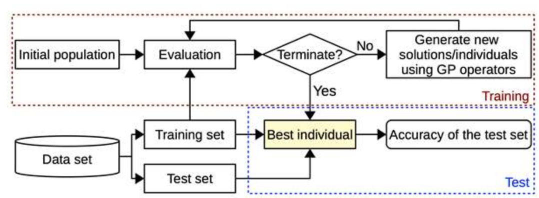
- A new multi-layer individual representation is developed in IEGP to allow it to automatically and simultaneously learn features and evolve ensembles for image classification
- IEGP can learn high-level features through multiple transformations
- IEGP can automatically select and optimise the parameters for the classification algorithms in the evolved ensemble
- IEGP can automatically address the diversity issue when building the ensembles

Ying Bi, Bing Xue, Mengjie Zhang. "Genetic Programming with A New Representation to Automatically Learn Features and Evolve Ensembles for Image Classification". IEEE Transactions on Cybernetics. **Vol. 51**, Issue 4. 2021. pp. 1769-1783. DOI:10.1109/TCYB.2020.2964566.

Multi-Layer Representation of IEGP



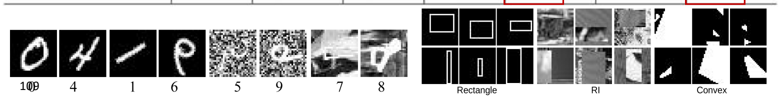
Overall Algorithm



Experimental Results

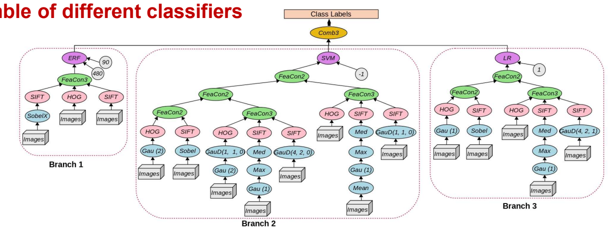
Classification accuracy of the proposed IEGP method

Method	MB	MRD	MBR	MBI	Rectangle	RI	Convex
SVM+RBF [51]	96.97(+)	88.89(+)	85.42(+)	77.39(+)	97.85(+)	75.96(+)	80.87(+)
SVM+Poly [51]	96.31(+)	84.58(+)	83.38(+)	75.99(+)	97.85(+)	75.95(+)	80.18(+)
SAE-3 [36]	96.54(+)	89.70(+)	88.72(+)	77.00(+)	97.86(+)	75.95(+)	–
DAE-b-3 [36]	97.16(+)	90.47(+)	89.70(+)	83.32(+)	98.01(+)	78.41(+)	–
CAE-2 [36]	97.52(+)	90.34(+)	89.10(+)	84.50(+)	98.79(+)	78.46(+)	–
SPAE [52]	96.68(+)	89.74(+)	90.99(+)	86.76(+)	–	–	–
RBM-3 [36]	96.89(+)	89.70(+)	93.27(+)	83.69(+)	97.40(+)	77.50(+)	–
ScatNet-2 [33, 34]	98.73(+)	92.52(+)	87.70(+)	81.60(+)	99.99(+)	91.98(+)	93.50(+)
RandNet-2 [34]	98.75(+)	91.53(+)	86.53(+)	88.35(+)	99.91(+)	83.00(+)	94.55(+)
PCANet-2 (softmax) [34]	98.60(+)	91.48(+)	93.15(+)	88.45(+)	99.51(+)	86.61(+)	95.81(+)
LDANet-2 [34]	98.95	92.48(+)	93.19(+)	87.58(+)	99.86(+)	83.80(+)	92.78(+)
NNet [51]	95.31(+)	81.89(+)	79.96(+)	72.59(+)	92.84(+)	66.80(+)	67.75(+)
SAA-3 [51]	96.54(+)	89.70(+)	88.72(+)	77.00(+)	97.59(+)	75.95(+)	81.59(+)
DBN-3 [51]	96.89(+)	89.70(+)	93.27(+)	83.69(+)	97.40(+)	77.50(+)	81.37(+)
FCCNN [35]	97.57(+)	91.09(+)	93.55(+)	86.77(+)	–	–	–
FCCNN (with BT) [35]	97.32(+)	90.41(+)	93.03(+)	89.20(+)	–	–	–
SPCN [32]	98.18(+)	90.19(+)	94.16	90.45	99.81(+)	89.40(+)	–
EvoCNN (best) [53]	98.82	94.78	97.20	95.47	99.99(+)	94.97	95.18(+)
EGP (best) [26]	97.19(+)	–	–	–	99.91(+)	–	93.97(+)
IEGP (best)	98.82	94.28	93.59	89.41	**100**	94.88	**98.26**
IEGP (mean)	98.69	93.78	92.65	88.42	99.94	89.02	97.76
IEGP (std)	0.08	0.24	0.35	0.64	0.05	2.1	0.26
Rank	2/20	2/19	3/19	3/19	**1/17**	2/16	**1/12**

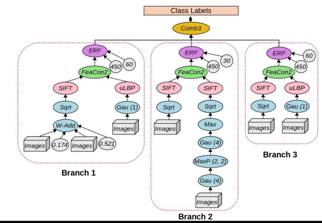


Example Solutions

1 Ensemble of different classifiers

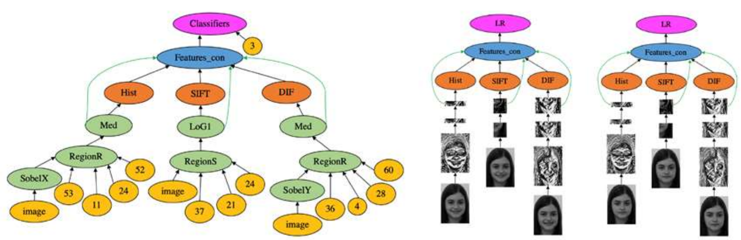


2 Ensemble of ensembles



GP-FR: with feature reuse

- A new GP approach with a new program structure, a new function set and a new terminal set to achieve flexible feature reuse
- Evolve programs/solutions that conduct region detection, image filtering, feature extraction, feature concatenation, and classification automatically and simultaneously

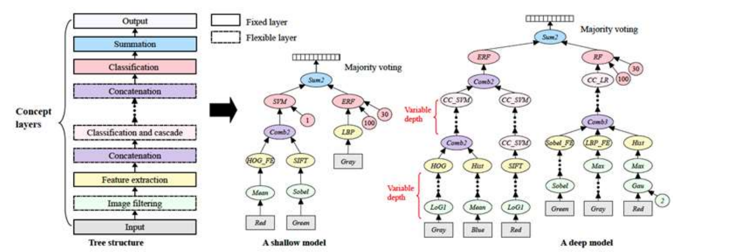


(a) example program. (b) "smile" class. (c) "natural" class.

Qinglan Fan, Ying Bi, Bing Xue, and Mengjie Zhang. "Genetic programming for image classification: a new program representation with flexible feature reuse." IEEE Transactions on Evolutionary Computation, DOI: 10.1109/TEVC .2022.3169490, 2022.

EDLGP

- Evolve variable-length tree based symbolic models, achieving promising classification performance in the data-efficient scenario
- A flexible multi-layer model representation to automatically evolve shallow or deep models for different image classification tasks



Ying Bi, Bing Xue, and Mengjie Zhang. "Genetic Programming-Based Evolutionary Deep Learning for Data-Efficient Image Classification." IEEE Transactions on Evolutionary Computation, DOI: 10.1109/TEVC.2022.3214503, 2022.

EDLGP

Classification Performance

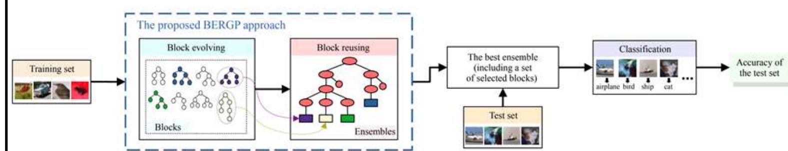
- Comparisons between the new approach and CNNs of varying complexity and dropout rate

Model	Dropout	sCIFAR10-10	sCIFAR10-20	sCIFAR10-40	sCIFAR10-80	sCIFAR10-160	sCIFAR10-320	sCIFAR10-640	sCIFAR10-1280
CNN-lc	0.0	27.1	32.4	36.1	41.8	45.3	50.1	54.1	59.4
CNN-lc	0.4	29.7	33.8	38.2	43.5	47.1	52.8	57.2	61.7
CNN-lc	0.7	29.7	34.9	40.0	44.9	49.4	53.9	58.1	62.3
CNN-mc	0.0	28.5	34.3	38.8	43.0	48.6	53.7	58.8	63.3
CNN-mc	0.4	29.7	34.9	39.9	45.4	50.9	55.8	61.0	66.2
CNN-mc	0.7	31.5	36.2	41.3	47.1	51.9	57.7	62.8	67.6
CNN-hc	0.0	30.1	34.2	39.1	44.7	50.6	56.1	61.2	65.9
CNN-hc	0.4	31.7	36.1	40.8	46.5	52.0	58.0	63.5	68.8
CNN-hc	0.7	**31.9**	37.0	42.5	48.1	53.9	**59.5**	**64.8**	69.8
ResNet-20	–	23.3	29.0	31.9	38.5	44.7	51.3	62.3	**71.5**
EDLGP(best)	–	**35.8**	**43.2**	**48.1**	**52.3**	**58.2**	**60.6**	**64.8**	64.5
EDLGP(mean)	–	31.7±3.8	**37.7±4.0**	**45.7±1.6**	**49.3±1.8**	**54.3±2.5**	58.7±1.5	61.6±2.0	61.1±3.4
Model	Dropout	sFMNIST-10	sFMNIST-20	sFMNIST-40	sFMNIST-80	sFMNIST-160	sFMNIST-320	sFMNIST-640	sFMNIST-1280
CNN-lc	0.0	71.1	74.0	77.8	81.0	83.6	85.4	86.7	88.2
CNN-lc	0.4	71.2	76.1	79.8	81.9	84.2	85.7	87.1	88.7
CNN-lc	0.7	71.3	75.6	78.7	81.6	83.3	85.3	87.0	87.9
CNN-mc	0.0	72.5	75.5	79.0	82.0	84.4	86.3	88.0	89.1
CNN-mc	0.4	72.4	76.1	79.6	82.9	84.7	86.6	88.1	89.6
CNN-mc	0.7	72.5	76.9	79.9	82.9	84.9	86.8	88.2	89.4
CNN-hc	0.0	71.9	75.9	80.1	82.3	85.1	86.8	88.6	89.5
CNN-hc	0.4	72.2	76.3	80.2	83.0	85.0	86.9	88.5	89.8
CNN-hc	0.7	**73.3**	77.4	80.5	**83.2**	**86.5**	**87.1**	88.8	89.9
ResNet-20	–	62.3	71.4	77.0	80.4	84.1	86.9	**89.2**	**90.5**
EDLGP(best)	–	73.2	**79.5**	**82.1**	**83.5**	85.0	**87.2**	88.6	89.0
EDLGP(mean)	–	71.8±1.0	**78.0±0.7**	**81.2±0.7**	83.0±0.4	84.5±0.5	86.1±0.7	86.8±0.6	88.0±0.6
Model	Dropout	sSVHN-10	sSVHN-20	sSVHN-40	sSVHN-80	sSVHN-160	sSVHN-320	sSVHN-640	sSVHN-1280
CNN-lc	0.0	25.3	37.5	50.5	64.1	73.0	77.7	80.9	83.6
CNN-lc	0.4	28.4	44.9	59.0	70.2	77.0	80.3	83.3	85.8
CNN-lc	0.7	28.8	46.7	60.6	72.1	77.7	81.3	83.2	86.2
CNN-mc	0.0	26.8	39.9	53.4	68.6	75.5	79.6	83.2	85.5
CNN-mc	0.4	29.6	43.3	64.3	72.1	78.3	82.2	84.8	87.3
CNN-mc	0.7	27.7	45.8	64.1	**74.6**	79.7	83.2	86.2	88.2
CNN-hc	0.0	24.9	37.5	55.5	67.7	74.5	80.1	84.0	86.1
CNN-hc	0.4	27.5	45.6	63.1	73.6	79.3	82.7	85.7	88.1
CNN-hc	0.7	28.8	44.8	64.7	74.4	79.5	84.0	86.3	88.6
ResNet-20	–	20.3	40.0	54.7	74.1	**83.5**	**86.7**	**89.5**	**92.2**
EDLGP(best)	–	**57.5**	**66.2**	**72.8**	**75.5**	79.7	82.1	82.1	85.0
EDLGP(mean)	–	**55.5±1.5**	**60.2±2.4**	**70.1±1.5**	73.6±1.1	76.9±2.2	79.0±1.5	80.5±0.8	82.3±0.9

Ying Bi, Bing Xue, and Mengjie Zhang. "Genetic Programming-Based Evolutionary Deep Learning for Data-Efficient Image Classification." IEEE Transactions on Evolutionary Computation, DOI: 10.1109/TEVC.2022.3214503, 2022.

Two-stage GP (BERGP)

- A two-stage approach (BERGP) based on GP with simple program structures is developed to automatically evolve and reuse blocks to construct solutions of ensembles for data-efficient image classification
- The first stage evolves a set of small and diverse blocks for image feature extraction
- The second stage makes effective reuse of the evolved blocks to construct ensembles for image classification

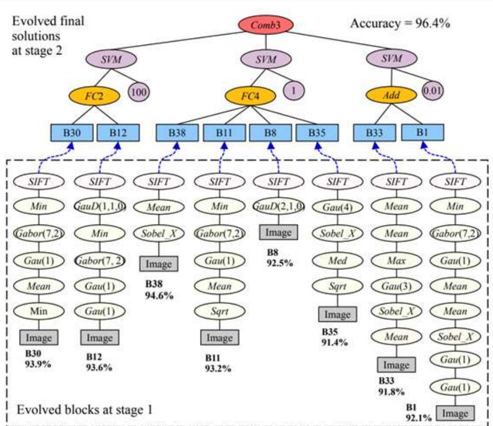


Ying Bi, Jing Liang, Bing Xue, and Mengjie Zhang. "A Genetic Programming Approach with Building Block Evolving and Reusing to Image Classification." IEEE Transactions on Evolutionary Computation, DOI: 10.1109/TCYB.2022.3174519, 2023.

Two-stage GP (BERGP)

Example Solutions

- Eight trees/blocks selected from Stage 1



Ying Bi, Jing Liang, Bing Xue, and Mengjie Zhang. "A Genetic Programming Approach with Building Block Evolving and Reusing to Image Classification." IEEE Transactions on Evolutionary Computation, DOI: 10.1109/TCYB.2022.3174519, 2023.

Summary

NN-based evolutionary deep learning has started to demonstrate great potential to outperform the manually designed state-of-the-art deep networks in image classification and analysis

GP based evolutionary deep learning has also started, and is expected to demonstrate the advantages in effectiveness, efficiency and interpretability in image analysis

Evolutionary deep learning is still in an early stage, but is expected to show the great accuracy, efficiency, small training set, and good interpretability of the deep models.

Acknowledgement

- Thanks GECCO 2043 Organizers
- Thanks CI lab at Zhengzhou University
- Thanks ECRG group at VUW

117

Summary

118

Concluding Remarks

- Evolutionary computer vision and image analysis is still a big and hot topic
  - Evolutionary deep learning will play a significant role
  - GP-based deep learning will have more developments
  - Interpretability and expandability will be a major focus
- EC techniques will be more popular in pattern recognition
  - Classification, Clustering
  - GP, GAs, PSO, DE,
  - EC will be in more mainstream conferences and journals
- GPU/FPGA will be popular tools

119

Future Events

- **IEEE SSCI Symposia**
  - CI in Feature Analysis, Selection and Learning in Image and Pattern Recognition (**IEEE FASLIP**))
  - CI for Multimedia Signal and Vision Processing (**IEEE CIMSIVP**)
- **EvoStar 2025**
  - Special Session on **Evolutionary Machine Learning**
  - EvoApplications including **Image Analysis and Pattern Recognition**
  - Paper submission**: 01 November 2024**
- **IEEE CEC 2025**
  - Special Session on **Evolutionary Computer Vision**
  - Paper Submission Deadline**: 31 Jan 2025 (tentative)**

120