Genetic Programming Approach to Deep Neuroevolution

Ricardo H. R. de Lima
Aurora Pozo

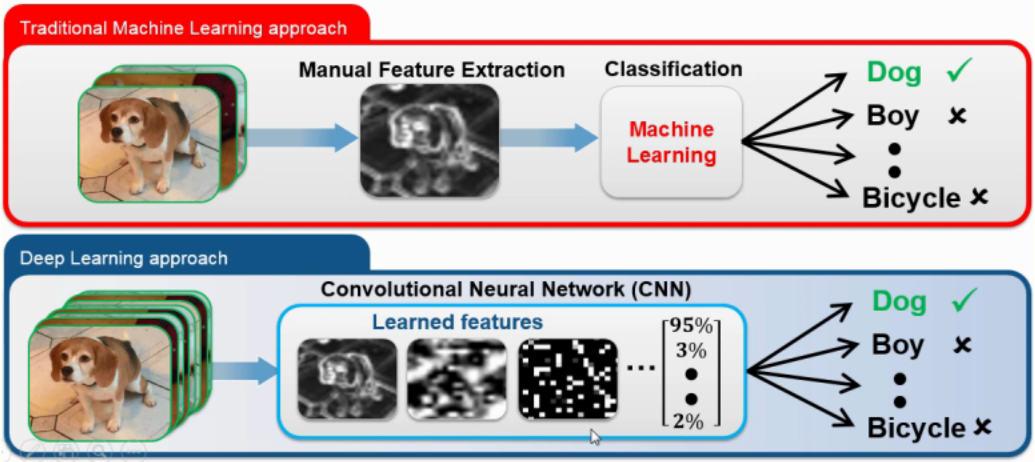
Conference, Month startDay–endDay, Year, City, Country

October 29, 2018

1. Deep Learning
   Exploring the topic
2. Genetic Programming
   Why using it?
3. Deep Neuroevolution
   Pros and Cons
4. Experiments
   Initial experiments
5. Conclusions
   Next steps

Deep Learning

Introduction

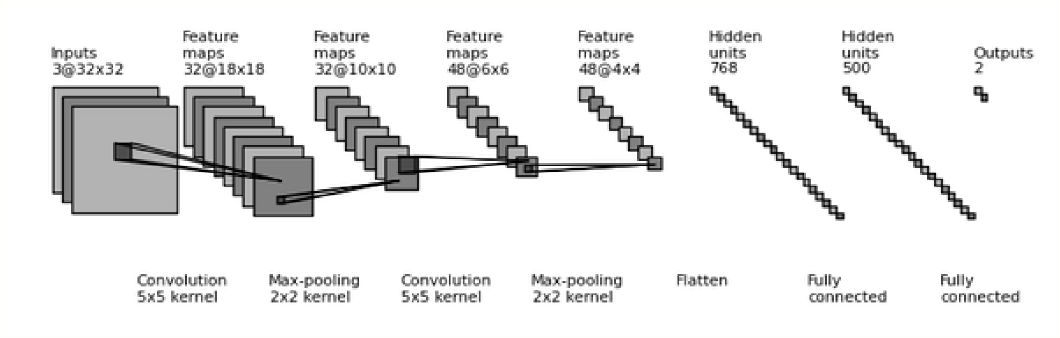


Introduction

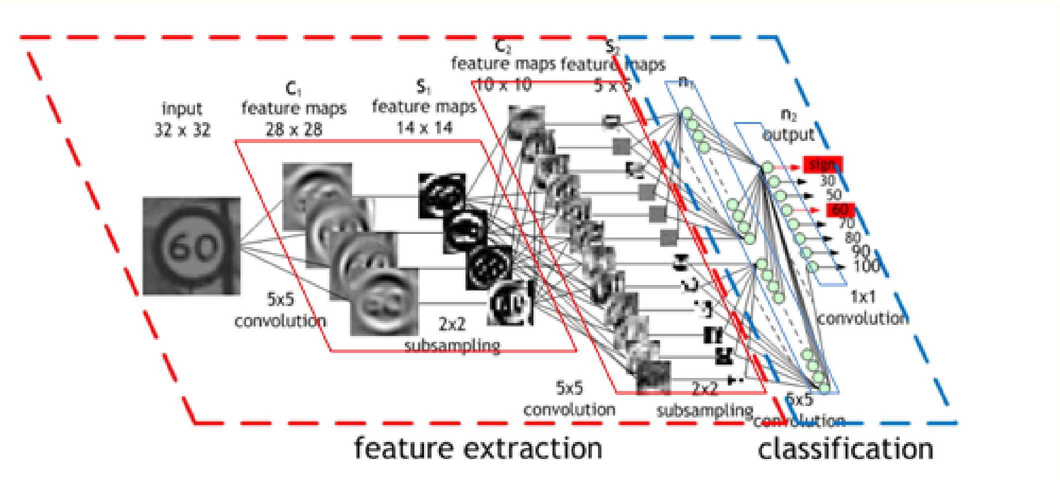
Deep Learning is a machine learning technique that can learn **useful representations** or features directly from images, text and sound

- Machine Learning approach
- Learns useful representations or features
- Reduces the use of preprocessing

Convolutional Neural Networks



Convolutional Neural Networks



Convolutional Neural Networks

- Used for image recognition
- Performs both generative and descriptive tasks

- Require large number of examples
- High number of trainable parameters
- High computational cost

Genetic Programming

Features

- Evolve programs
- Combines and modifies solutions to create new ones

- Classic Tree-based GP
- Grammatical Evolution
- Cartesian GP

Grammatical Evolution

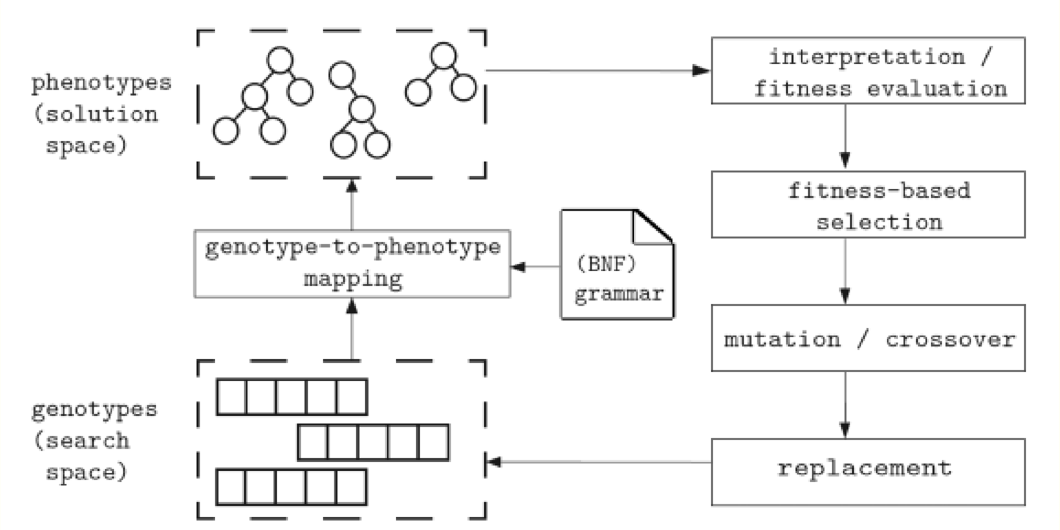
- BNF Grammar

  Define the programs structure
- Mapping process

  Translate genotype to phenotype
- Search Engine

  Genetic Algorithm

Grammatical Evolution



Deep Neuroevolution

Proposed Approach

Neuroevolution is a form of artificial intelligence that uses **evolutionary algorithms** to **generate artifial neural networks**, parameters, topology and rules. Studies on how to evolve Deep Neural Networks are called **Deep Neuroevolution**

- Fitness function
  - Accuracy/Error Rate
- Architecture
  - Define smallest/biggest valid model

Approach: Evaluating the Models

The CNNs are evaluated using the **error rate**, obtained from training and test, on well known datasets.

- Number of epochs
- Time spent
- Loss

Approach: Architecture

Smallest CNN

The smallest cnn that we can think of is:
inputs » convolution » dense » outputs

- What defines a CNN is the use of convolutions
- The NN needs a fully connected layer to process correctly the inputs

Biggest CNN

How many layers is enough? 30, 50, 100?

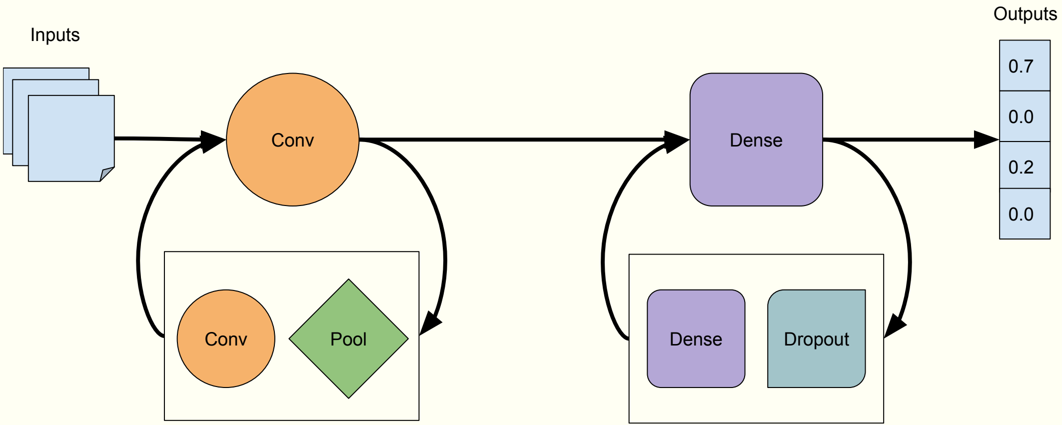
- Define a upper bound

Approach: Grammar

```
<cnn> ::= <conv> <c_layer> <d_layer> <dense>


<c_layer> ::= <c_layer> <c_layer> | <c_node> | '&'

<d_layer> ::= <d_node> <d_node> | <d_node> | '&'


<c_node> ::= <conv> | <maxpool> | <avgpool>

<d_node> ::= <dense> | <dropout>


<conv> ::= 'class_name' 'Conv2D' 'filters' <filters> 'kernel_size' <k_size> 'activation' <activation>

<dense> ::= 'class_name' 'Dense' 'units' <units>

<dropout> ::= 'class_name' 'Dropout' 'rate' <rate>

<maxpool> ::= 'class_name' 'MaxPooling2D' 'pool_size' <p_size> 'padding' <padding>

<avgpool> ::= 'class_name' 'AveragePooling2D' 'pool_size' <p_size> 'padding' <padding>
```

Figure: Part of the proposed grammar for designing CNNs

Approach: Grammar



Approach: Grammar

To avoid invalid models, some rules were created:

1. All models start with a convolution node
2. All models end with a dense node
3. The last node has the number of units set to the number of classes
4. The last node has the activation function set to 'softmax'
5. The first dense node is preceded by a 'flatten' node

Invalid models are still possible, in these cases, we penalize the solution.

Approach: CNN Parameters

- Convolution: Filters, Kernel Size, Activation
- Max/Average Pooling: Pool size, Padding
- Dense: Units, Activation
- Dropout: Drop rate
- Filters: 32, 64
- Kernel Size: (3, 3), (5, 5)
- Activation: Relu, Tahnm, Linear
- Pool size: (2, 2), (4, 4)
- Padding: Valid, Same
- Units: 32, 64
- Drop rate: [0, 1]

Tools

- Tensorflow/Theano

  Powerful tools to build and run cnns
- Keras

  High level framework that make easier to build the networks that run over tensorflow/theano
- Python

Experiments

Experiments: GE Parameters

Parameter	**Method**	**Value**
Population	-	20
Evaluations	-	400
Selection	Random	2
Crossover	One point	0.8
Mutation	Point	0.1
Prune	-	0.1
Duplication	-	0.1
Epochs	-	5

Experiments: MNIST

The MNIST dataset has 50000 for training and 10000 for validation and test. The images are 28x28 greyscale, 10 classes.

Run	Solution	Params	Validation	Test
1	[114 22 204 177]	506,058	0,989	0,990
2	[4 99 69]	181,194	0,991	0,993
3	too big	87,281	0,990	0,990
4	[213 121 215]	401,226	0,990	0,992
5	[37 141 108]	31,841	0,992	0,993
Mean	-	241,520	0,990	0,992

Experiments: notMNIST

The MNIST dataset has 50000 for training and 10000 for validation and test. The images are 28x28 greyscale, 10 classes.

Run	Solution	Params	Validation	Test
1	[217 21 150]	318,41	0,901	0,951
2	[51 193]	360,138	0,899	0,951
3	[63 85]	360,138	0,895	0,950
4	too big	360,138	0,897	0,948
5	[225 123 209 253 84 205]	360,138	0,897	0,949
Mean	-	351,792	0,898	0,950

Experiments: fashion MNIST

The MNIST dataset has 50000 for training and 10000 for validation and test. The images are 28x28 greyscale, 10 classes.

Run	Solution	Params	Validation	Test
1	[135 34]	406,218	0,918	0,917
2	[6 127 17]	1731,786	0,914	0,915
3	too big	361,354	0,917	0,920
4	[204 28]	599,754	0,915	0,914
5	[189 220]	406,218	0,917	0,918
Mean	-	701,066	0,916	0,917

Experiments: CIFAR 10

The MNIST dataset has 40000 for training and 10000 for validation and test. The images are 32x32 RGB, 10 classes.

Run	Solution	Params	Validation	Test
1	[235 220 235 13 210]	206,282	0,649	0,636
2	[63 209 120 138 130 117]	439,946	0,714	0,708
3	[166 165]	426,506	0,696	0,690
4	[94 117]	426,506	0,693	0,692
5	[34 171]	426,506	0,694	0,690
Mean	-	385,149	0,689	0,683

Experiments: CIFAR 100

The MNIST dataset has 50000 for training and 10000 for validation and test. The images are 32x32 RGB, 100 classes.

Run	Solution	Params	Validation	Test
1	[92 43 136 151 204 85 193 47 185 221]	159,332	0,322	0,329
2	[59 8 227 77 117 91 178 173]	318,564	0,350	0,350
3	[219 118 138 67]	5037,092	0,340	0,336
4	too big	1441,892	0,334	0,325
5	[13 37]	318,564	0,325	0,318
Mean	-	1455,089	0,334	0,332

Experiments: Running the best models

The best models were run now using 100 epochs. The values shown are the mean value across the five models obtained in the previous experiments.

Dataset	Training	Validation	Test
MNIST	0,979	0,976	0,976
notMNIST	0,946	0,904	0,904
Fashion MNIST	0,946	0,902	0,902
CIFAR-10	0,848	0,710	0,710
CIFAR-100	0,726	0,330	0,330

Conclusions

About the experiments

- Due to time limitations to execute experiments, each model was run for 5 epochs. Even though, the results for the MNIST datasets were all above 90% accuracy.
- The CIFAR 100 faced a serious case of overfitting, showing a high accuracy for the training, but a poor performance over the validation and test.
- All experiments could be improved if more time is given.

Next steps: Grammar

Optimize de grammar:

- filters
- units
- kernel size
- pool size
- activation
- optimizer
- loss

Next steps: Algorithm

The main issue with GE is dealing with fine tunning. Small changes on the genotype could produce major changes on the phenotype. A similar work of Suganuma 2017. Proposes the use of Cartesian GP to address the design of CNN architectures.

- Does not face the tunning issue
- simple representation
- flexible

Genetic Programming Approach to Deep Neuroevolution

Thank you!

Ricardo H. R. de Lima

Aurora Pozo