Paper Review :

AutoML-Zero: Evolving Machine Learning Algorithms From Scratch

Esteban Real , Chen Liang , David R. So, Quoc V. Le

INDEX :

OVERVIEW

AutoML-Zero: Evolving Machine Learning Algorithms From Scratch

Esteban Real [*1] **Chen Liang** [*1] **David R. So** [1] **Quoc V. Le** [1]

Abstract

Machine learning research has advanced in multiple aspects, including model structures and learning methods. The effort to automate such research, known as AutoML, has also made significant progress. However, this progress has largely focused on the architecture of neural networks, where it has relied on sophisticated expert-designed layers as building blocks—or similarly restrictive search spaces. Our goal is to show that AutoML can go further: it is possible today to automatically discover complete machine learning algorithms just using basic mathematical operations as building blocks. We demonstrate this by

field, ranging from learning strategies to new architectures [Rumelhart et al., 1986; LeCun et al., 1995; Hochreiter & Schmidhuber, 1997, among many others]. The length and difficulty of ML research prompted a new field, named *AutoML*, that aims to automate such progress by spending machine compute time instead of human research time (Fahlman & Lebiere, 1990; Hutter et al., 2011; Finn et al., 2017). This endeavor has been fruitful but, so far, modern studies have only employed constrained search spaces heavily reliant on human design. A common example is *architecture search*, which typically constrains the space by only employing sophisticated expert-designed layers as building blocks and by respecting the rules of backpropagation (Zoph & Le, 2016; Real et al., 2017; Tan et al.,

- Google 엔지니어 팀 발표(2020)
- ‘AutoML-Zero’ : 컴퓨터 프로그램이 스스로 머신러닝 예측 모델을 자동으로 생성할 수 있는 알고리즘
- 기존의 AutoML 자동화 연구는 신경망의 아키텍쳐에 초점. 그러나 여전히 인간 엔지니어의 많은 입력이 필요. 이는 필연적으로 인간의 편견과 한계가 기술에 녹아들어 있음을 의미
- 그렇다면 자체 머신러닝 알고리즘을 생성하는 시스템을 만들어 프로세스에 미치는 영향을 최소화할 수 있다면 어떨까?

OVERVIEW

Abstract

Machine learning research has advanced in multiple aspects, including model structures and learning methods. The effort to automate such research, known as AutoML, has also made significant progress. However, this progress has largely focused on the architecture of neural networks, where it has relied on sophisticated expert-designed layers as building blocks—or similarly restrictive search spaces. Our goal is to show that AutoML can go further: it is possible today to automatically discover complete machine learning algorithms just using basic mathematical operations as building blocks. We demonstrate this by

- 기존의 AutoML 연구 발전은 신경망의 아키텍처에 주로 초점. AutoML이긴 하지만 사람이 개입해서 신경망의 계층마다 옵션을 정해놓고, search 알고리즘을 통해 아키텍처를 설계하는 방식
  ▶ 사람이 설계해야하는 요소가 남아있음

- 본 논문의 목표는 AutoML이 더 나아갈 수 있음을 보여주는 것. 전체 ML 알고리즘을 설계하는 과정에서 사람의 개입을 최소화.

- ML과 관련 있는 요소들을 고등학교 교육과정 수준의 수학적인 연산 정도만 옵션으로 주고 다 설계할 수 있는지를 살펴보는 것을 중점으로 연구를 진행

METHODS

3.1. Search Space

We represent algorithms as computer programs that act on a small virtual memory with separate address spaces for scalar, vector and matrix variables (*e.g.* `s1`, `v1`, `m1`), all of which are floating-point and share the dimensionality of the task's input features (F). Programs are sequences of instructions. Each instruction has an operation—or *op*—that determines its function (*e.g.* "multiply a scalar with a vector"). To avoid biasing the choice of ops, we use a simple criterion: those that are typically learned by high-school level. We purposefully exclude machine learning concepts, matrix decompositions, and derivatives. Instructions have op-specific arguments too. These are typically addresses in the memory (*e.g.* "read the inputs from scalar address 0 and vector address 3; write the output to vector address 2"). Some ops also require real-valued constants (*e.g.* μ and σ for a random Gaussian sampling op), which are searched for as well. Suppl. Section S2 contains the full list of 65 ops.

Search Space

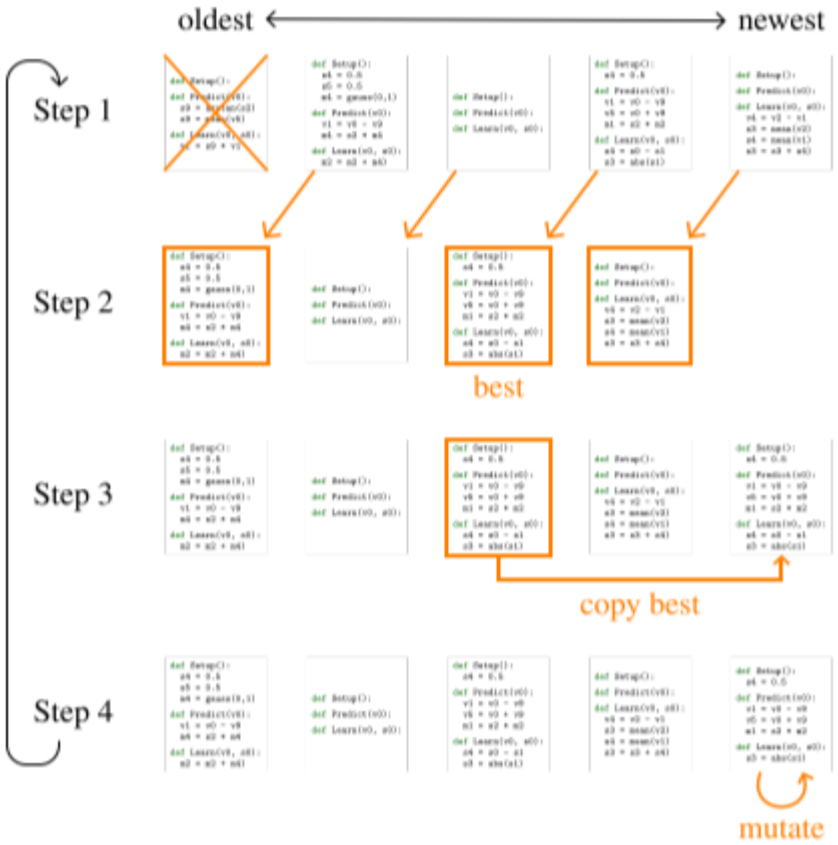
- 65가지의 고등학교 수준의 수학 연산들(Instruction)만 이용하여 아무것도 없는 상태에서 머신러닝 알고리즘을 만들고, 학습을 수행
- 기존에 이와 비슷한 수학 기호들을 search space로 하여 AutoML을 적용한 시도를 다룬 연구(Neural Optimizer Search with Reinforcement Learning)가 있었으나, optimizer를 제외한 나머지 영역은 사람이 개입했다는 차이점

METHODS

```
# (Setup, Predict, Learn) = input ML algorithm.
# Dtrain / Dvalid = training / validation set.
# sX/vX/mX: scalar/vector/matrix var at address X.
def Evaluate(Setup, Predict, Learn, Dtrain,
Dvalid):
  # Zero-initialize all the variables (sX/vX/mX).
  initialize_memory()
  Setup() # Execute setup instructions.
  for (x, y) in Dtrain:
    v0 = x # x will now be accessible to Predict.
    Predict() # Execute prediction instructions.
    # s1 will now be used as the prediction.
    s1 = Normalize(s1) # Normalize the prediction.
    s0 = y # y will now be accessible to Learn.
    Learn() # Execute learning instructions.
  sum_loss = 0.0
  for (x, y) in Dvalid:
    v0 = x
    Predict() # Only Predict(), not Learn().
    s1 = Normalize(s1)
    sum_loss += Loss(y, s1)
  mean_loss = sum_loss / len(Dvalid)
  # Use validation loss to evaluate the algorithm.
  return mean_loss
```

- Setup() : weight를 초기화
- Predict() : forward propagation을 구성하는 모든 요소
- Learn() : training에 관여하는 모든 요소를 포함하고 있으며 training loop에서만 수행.
- component function과 training set, validation set이 필요
- classification task의 경우, prediction 값을 0과 1사이로 normalize 시키기 위해 sigmoid(binary classification) 또는 softmax(multi-class)를 추가해 줌(사람의 개입)

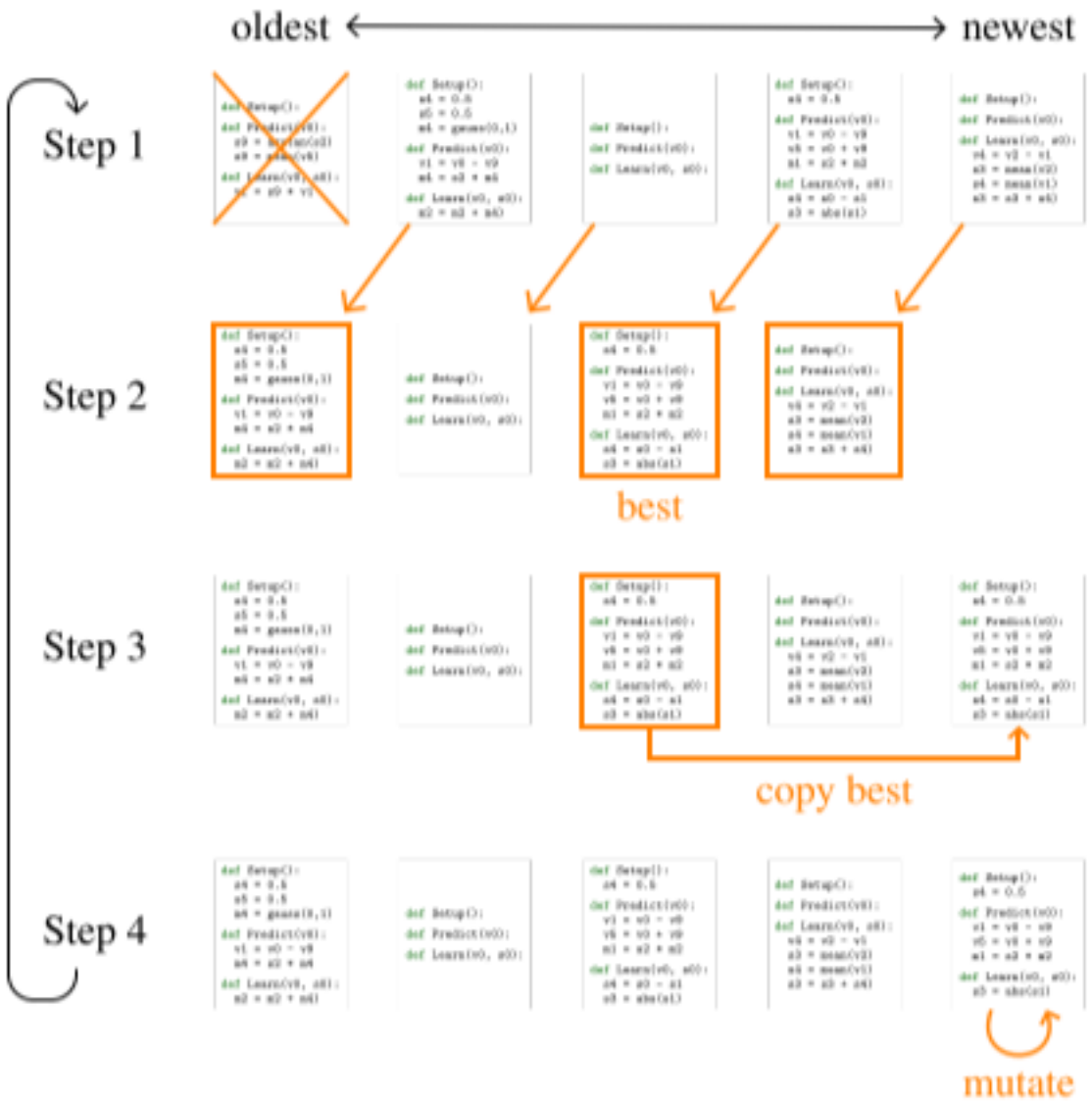
METHODS



Search Method

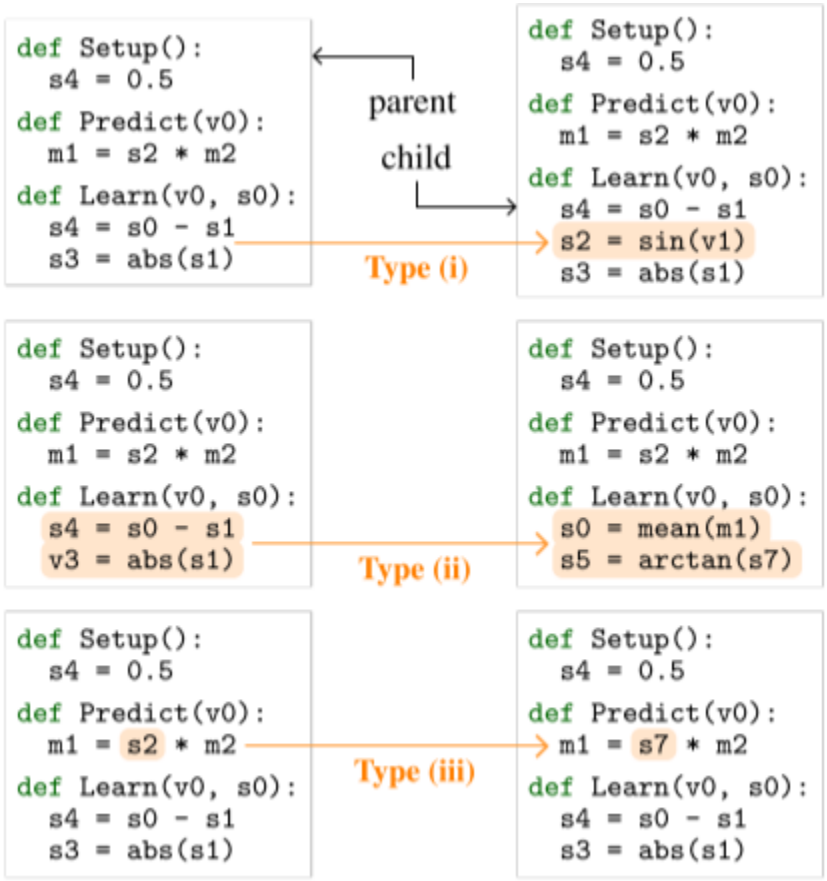
- AutoML-Zero에서는 evolutionary algorithm(진화 알고리즘)을 채택. (regularized evolution search method)
- 진화 알고리즘(evolutionary algorithm) : 생물의 진화를 흉내 낸 생식 돌연변이, 유전자 조작, 자연선택, 적자생존 등 진화의 구조에 영감을 얻은 탐색 기법. 즉 생물의 진화를 모방해 최적해를 찾아내는 알고리즘

METHODS



Step 1. 모수 집단에서 가장 오래된 알고리즘을 제거한다.
Step 2. 랜덤하게 P보다 적은 T개의 알고리즘을 선택하여 가장 좋은 성능을 가진 알고리즘을 부모(parent)로 삼는다.
Step 3. 선택된 알고리즘을 복사한다.
Step 4. 부모 알고리즘은 변이 되어 자식(child) 알고리즘이 되고, 이 알고리즘이 모수에 더해진다.

METHODS



MUTATION(변이)
- Insert or Remove random instruction :
임의의 instruction을 넣거나 제거하는 방법

- Randomize component function :
하나의 component function의 구성을 모두 바꿔버리는 방법

- Modify an argument :
하나의 instruction의 argument 하나만 바꾸는 방법

▶ MUTATION을 적용할 때는 부득이하게 사람이 개입

EXPERIMENT

4.1. Finding Simple Neural Nets in a Difficult Space

We now demonstrate the difficulty of the search space through random search (RS) experiments and we show that, nonetheless, interesting algorithms can be found, especially with evolutionary search. We will explore the benefits of evolution as we vary the task difficulty. We start by searching for algorithms to solve relatively easy problems, such as fitting linear regression data. Note that without the following simplifications, RS would not be able to find solutions.

"how difficult is searching the AutoML-Zero space?"

- evolutionary search를 통해 새로운 알고리즘을 찾을 수 있다는 결과
- 회귀분석 같은 간단한 케이스의 경우도 진화 알고리즘이 5배 효과적이었다는 결과, RS와의 비교에서 갭이 크게 나타남

EXPERIMENT

4.2. Searching with Minimal Human Input

Teacher datasets and carefully chosen ops bias the results in favor of known algorithms, so in this section we replace them with more generic options. We now search among a

```
# sX/vX/mX = scalar/vector/matrix at address X.
# "gaussian" produces Gaussian IID random numbers.
def Setup():
  # Initialize variables.
  m1 = gaussian(-1e-10, 9e-09) # 1st layer weights
  s3 = 4.1 # Set learning rate
  v4 = gaussian(-0.033, 0.01) # 2nd layer weights
def Predict():  # v0=features
  v6 = dot(m1, v0) # Apply 1st layer weights
  v7 = maximum(0, v6) # Apply ReLU
  s1 = dot(v7, v4) # Compute prediction
def Learn():  # s0=label
  v3 = heaviside(v6, 1.0) # ReLU gradient
  s1 = s0 - s1 # Compute error
  s2 = s1 * s3 # Scale by learning rate
  v2 = s2 * v3 # Approx. 2nd layer weight delta
  v3 = v2 * v4 # Gradient w.r.t. activations
  m0 = outer(v3, v0) # 1st layer weight delta
  m1 = m1 + m0 # Update 1st layer weights
  v4 = v2 + v4 # Update 2nd layer weights
```

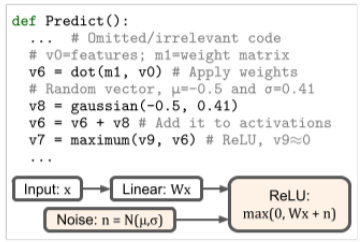
"can we use our framework to discover reasonable algorithms with minimal human input?"

- 최소한의 사람의 개입만으로도 납득할만한 알고리즘을 발견할 수 있는지?
- CIFAR-10과 MNIST을 이용하여 binary classification 문제해결하는 실험 결과 : 사람이 디자인한 2-layer MLP + gradient descent 모델보다 20번 중 13번 더 좋은 성능을 달성
- 전체 CIFAR-10 test set에 대해 성능을 비교해보니 AutoML-Zero로 찾은 모델이 logistic regression, 2-layer MLP보다 더 높은 정확도를 달성

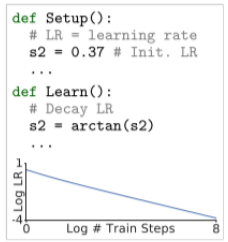
EXPERIMENT

4.3. Discovering Algorithm Adaptations

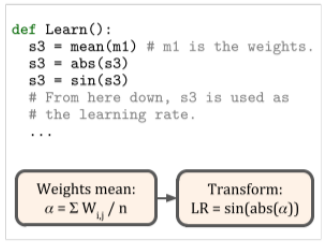
In this section, we will show wider applicability by searching on three different task types. Each task type will impose its own challenge (*e.g.* "too little data"). We will show that evolution specifically adapts the algorithms to meet the challenges. Since we already reached reasonable models from scratch above, now we save time by simply initializing the populations with the working neural network of Figure 5.



(a) Adaptation to few examples.



(b) Adaptation to fast training.



(c) Adaptation to multiple classes.

"can we discover different algorithms by varying the type of task we use during the search experiment?"

- 탐색 실험 중에 사용하는 작업 유형을 변경하여 다른 알고리즘을 발견할 수 있는지? 즉 문제상황이 바뀔 때 유연하게 적응할 수 있는지 확인
- 총 3가지 상황 실험에서 잘 적응한 결과를 보임
  - training example이 부족한 경우 : AutoML-Zero는 input data에 noise를 주입하여 augmentation을 하는 것을 스스로 배움
  - 빠른 학습을 하여야 하는 경우 : 스스로 learning-rate decay를 학습
  - class의 개수가 늘어나는 경우 : 사람이 설계하는 것과는 조금 다른 방식을 채택하였지만, 성능이 향상됨

▶ 즉 사람이 이해할 수 있는 방식을 제안하기도 하지만 이해할 수 없는 방식도 제안.

CONCLUSION

- 기존에 사람이 많이 개입하여 왔던 AutoML 분야에서 사람의 개입을 최소화한 채 밑바닥부터 evolutionary search method를 통해 수학 연산(instruction)들을 조합하여 ML 모델을 생성하는 과정을 제안
- AutoML-Zero의 궁극적인 목표는 새로운 기계 학습의 개념으로 이어지길 기대하며, 검색 공간에서 인간의 편견을 줄이는 것. 이번 결과에서는 인간이 설계하는 분량을 크게 절감했으나, 정교한 진화 계산, 강화학습 등의 미래 과제가 남음
- 기계 학습 알고리즘을 zero부터 만들어내는, AutoML 분야의 새로운 가능성을 보여준 논문

EOD.