AutoML-Zero: Evolving Machine Learning Algorithms From Scratch

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

Presenter: Joel Zylberberg

www.jzlab.org

The "usual" Neural Nets Approach

<u>User specifies:</u>

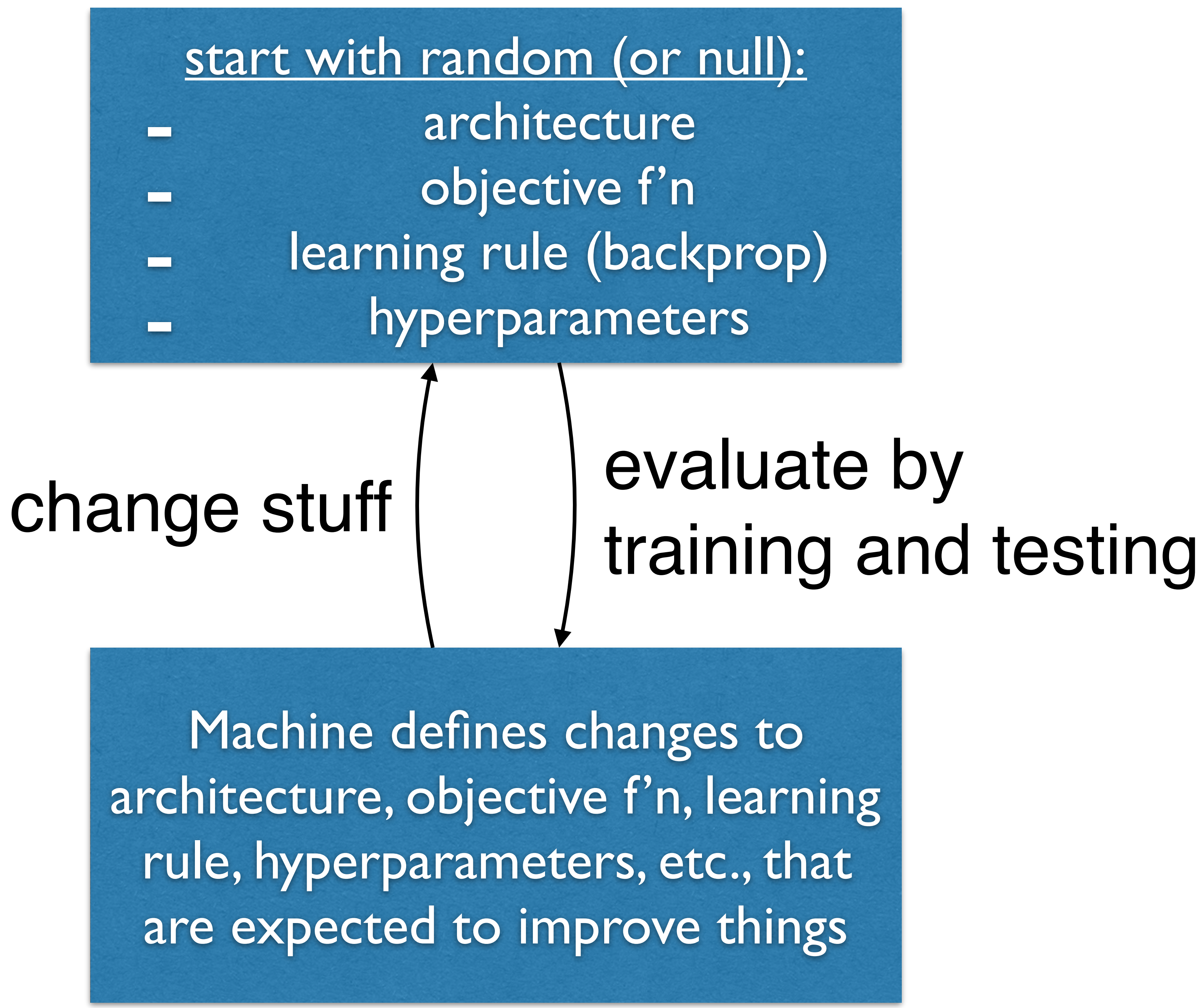
- architecture
- objective f'n
- learning rule (backprop)
- hyperparameters

change stuff

evaluate by training and testing

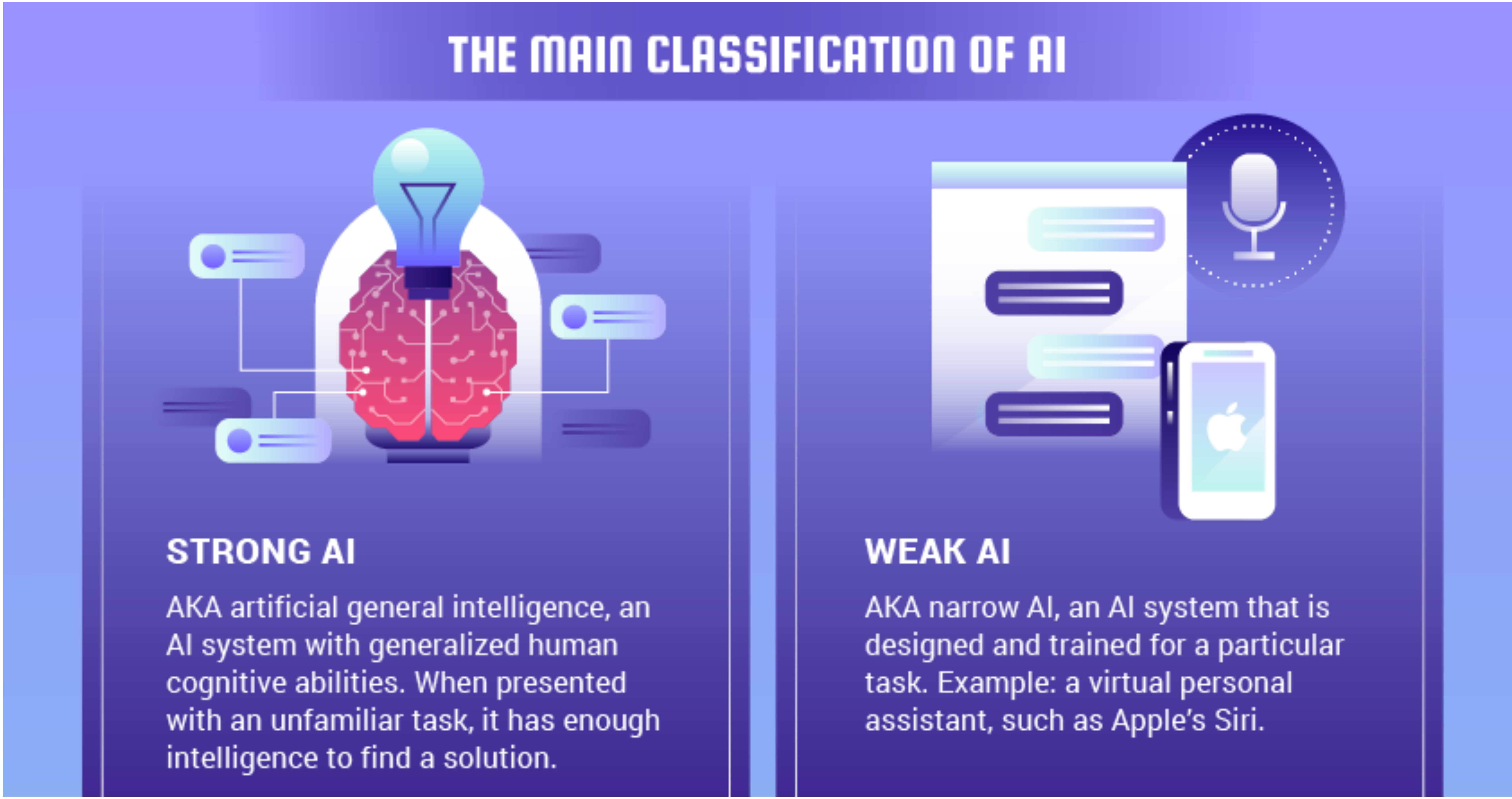
User uses intuition, experience, brute force experimentation, or conference papers to guess at better NN implementation

The “autoML” Approach



Context on AI and Machine Learning

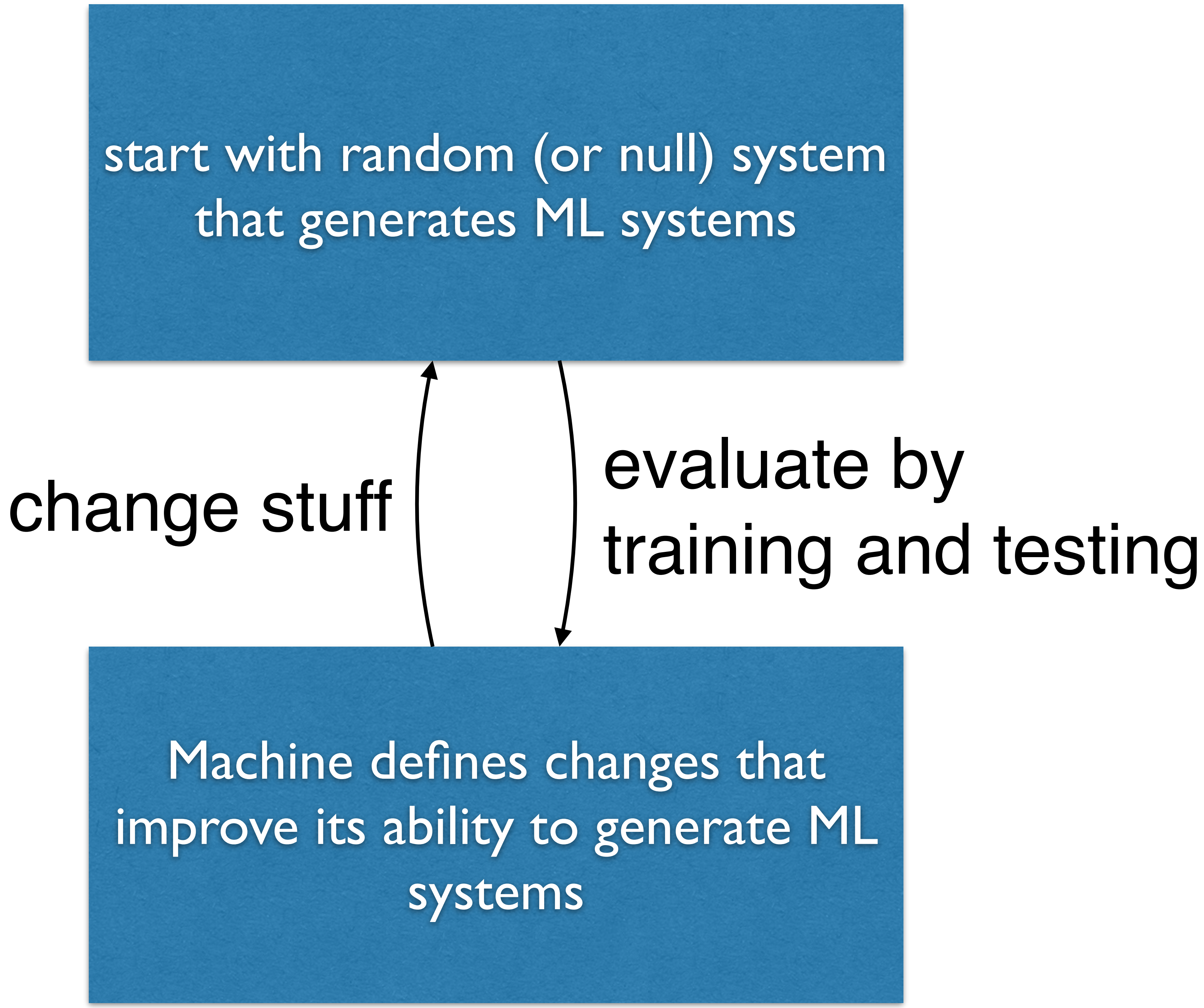
[VJ Manickam, 2020]



This doesn't exist yet
(but is desirable and/or dangerous)

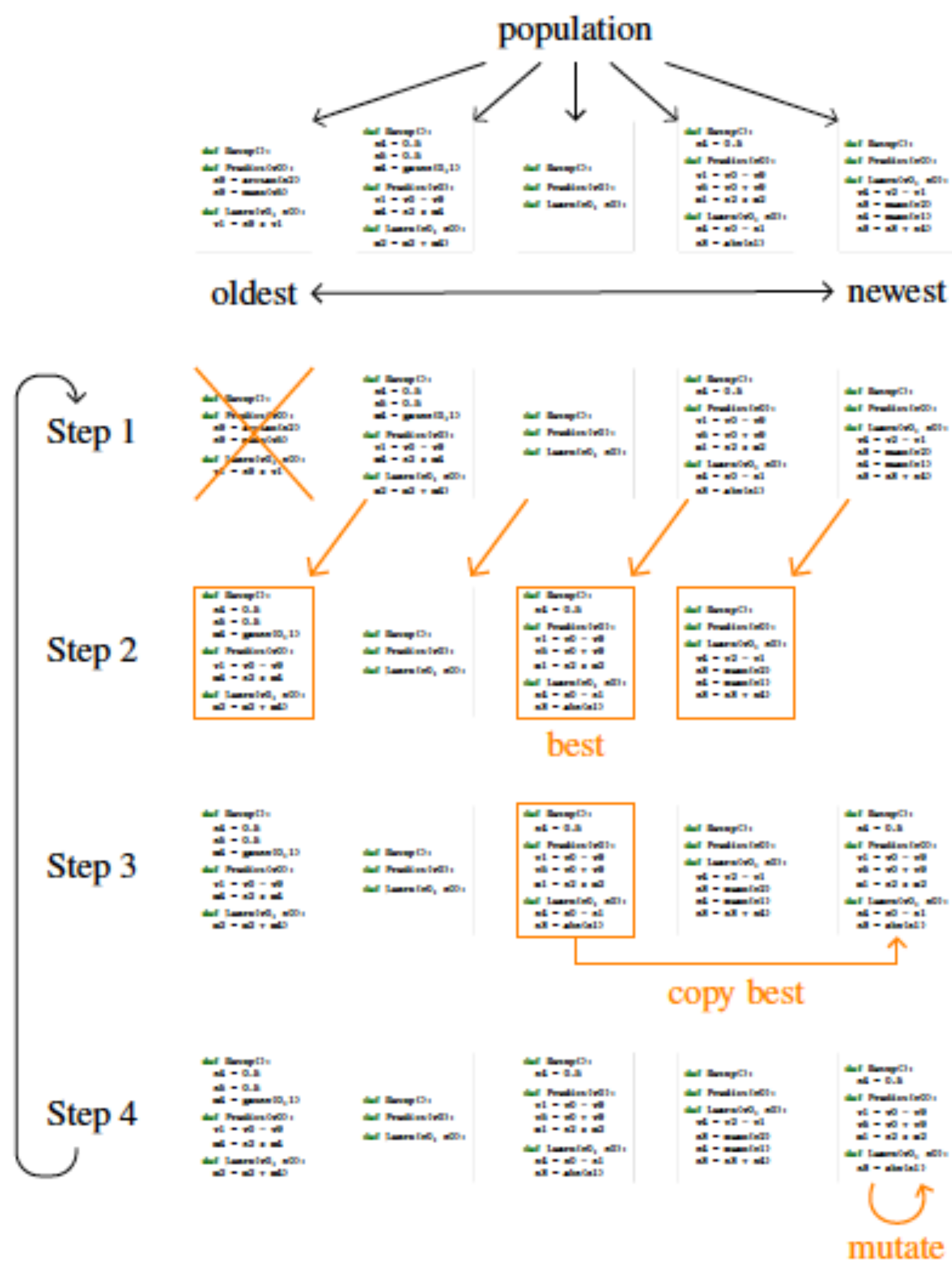
This is a product of hand-built NNs
(and is rapidly improving)

The dreams of field



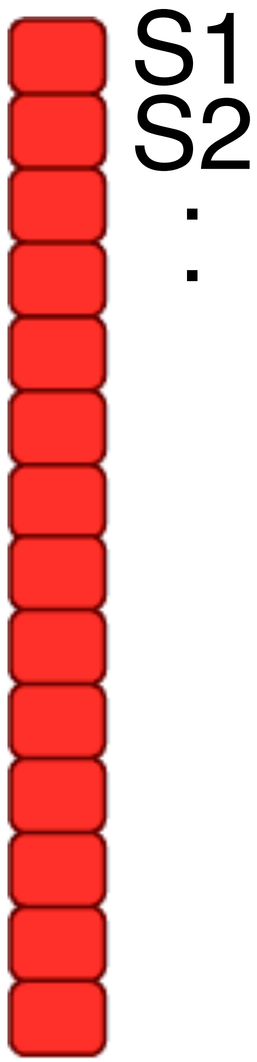
Basic idea of the paper:

1) define an algorithmic procedure for writing ML algorithms
2) use simulated evolution to evolve a population of these ML algorithms (best ones propagated forward with mutation)
3) spend **tons** of compute resources and wait awhile



Core pieces of AutoML-Zero programs

<u>Virtual Memory</u>



<u>Instructions</u>

operation
and associated
memory address(es)

E.g.,
Read input from S1,S2
multiply them together,
write answer to S3

<u>Component Functions</u>

setup, predict, learn

Each component
function is a series
of instructions

Example programs (the big population of these evolves)

```
def Setup():
  s4 = 0.5
  s5 = 0.5
  m4 = gauss(0,1)
def Predict(v0):
  v1 = v0 - v9
  m4 = s2 * m4
def Learn(v0, s0):
  m2 = m2 + m4)
```

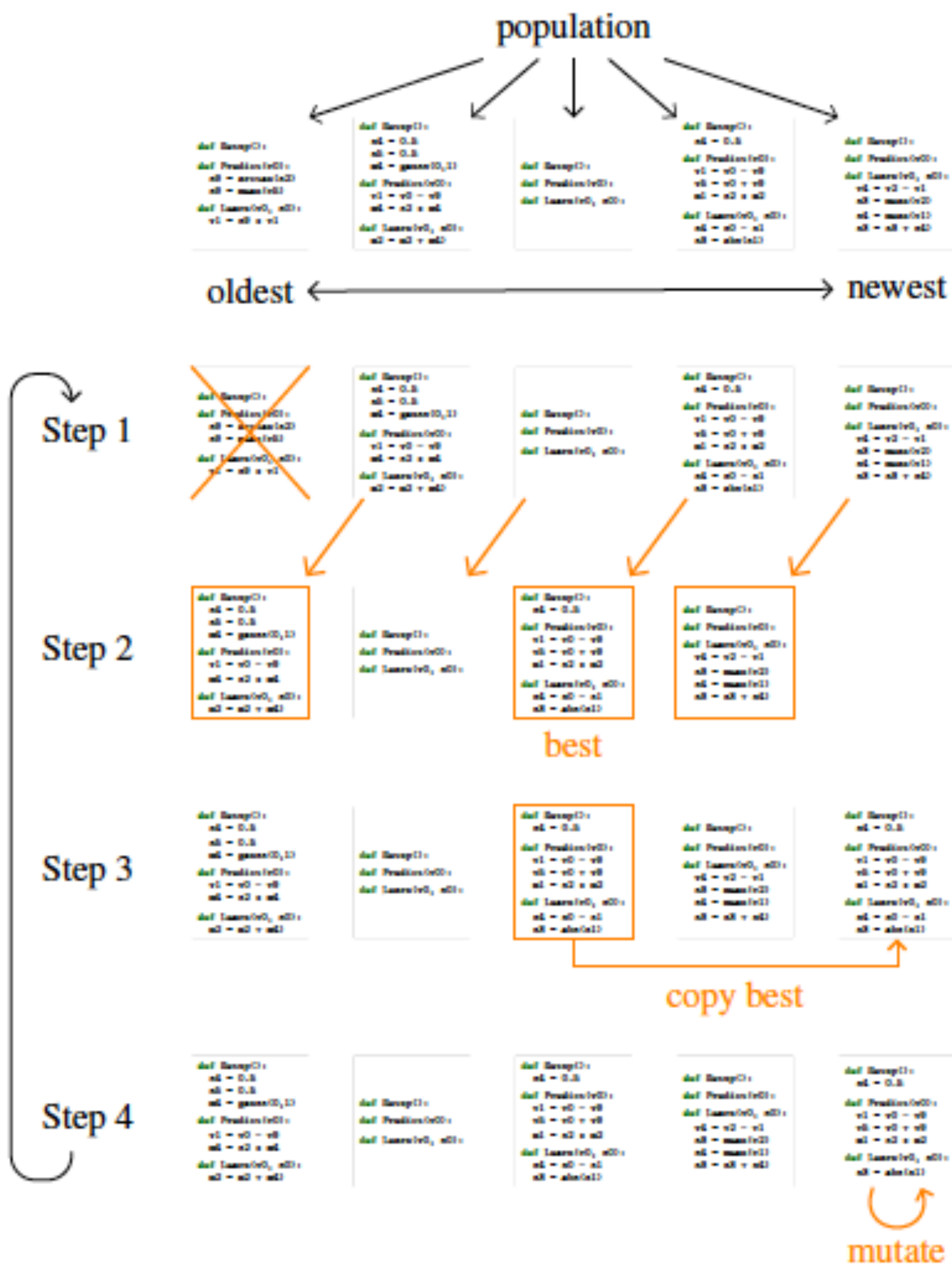
```
def Setup():
def Predict(v0):
def Learn(v0, s0):
```

this one doesn’t do anything

```
def Setup():
  s4 = 0.5
def Predict(v0):
  v1 = v0 - v9
  v5 = v0 + v9
  m1 = s2 * m2
def Learn(v0, s0):
  s4 = s0 - s1
  s3 = abs(s1)
```

```
def Setup():
def Predict(v0):
def Learn(v0, s0):
  v4 = v2 - v1
  s3 = mean(v2)
  s4 = mean(v1)
  s3 = s3 + s4)
```

Simulated evolution procedure



Simulated evolution procedure hinges on evaluation (for finding the best model in each population)

Evaluation Procedure

```
# (Setup, Predict, Learn) is the input ML algorithm.
# Dtrain / Dvalid is the 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 execute 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
```

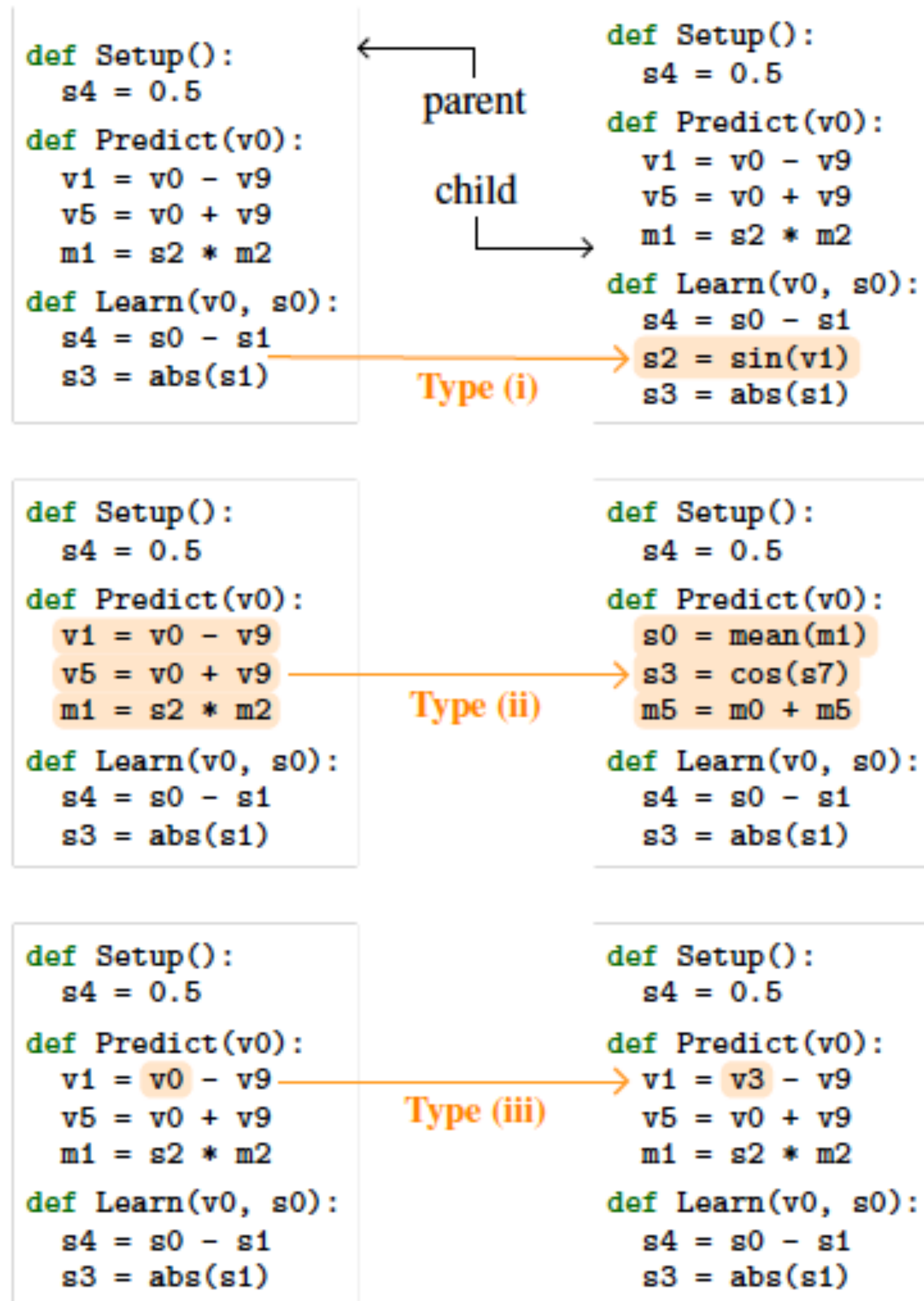


Figure 3: Mutation examples. Parent algorithm is on the left; child on the right. (i) Insert a random instruction (removal also possible). (ii) Randomize a component function. (iii) Modify an argument.

Key thing: the space of operations their systems can use / try is NOT limited to neural network operations. Includes a long list of "reasonable & simple" math operations

Now that the framework is in place…

<u>Is it necessary?</u> (Would Random Search — RS — do just as well?)

Figure 4: Relative success rate of evolution and random search (RS). Each point represents a different task type and the x-axis measures its difficulty (defined in main text). As the task type becomes more difficult, evolution vastly outperforms RS, illustrating the complexity of AutoML-Zero when compared to more traditional AutoML spaces.

On simple regression problems, evolution approach finds more "good" solutions than RS. Algorithm search space is big so RS is bad.

Experiment 2:

- start with a random 2-layer neural net. Use it to generate input-output pairs
- evaluate programs in autoML based on whether they can duplicate that NNs outputs
- after evolution, read the best program and add comments to it

```
# sX/vX/mX = scalar/vector/matrix at address X.
# The C_ (eg C1) are constants tuned by search.
# "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
```

Figure 5: Rediscovered neural network algorithm. It implements backpropagation by gradient descent. Comments added manually.

Experiment 3:

- evaluate programs in autoML based on whether they can label CIFAR-10 images. (Simplified version, tbh)
- at a few stages of evolution, read the best program and add comments to it

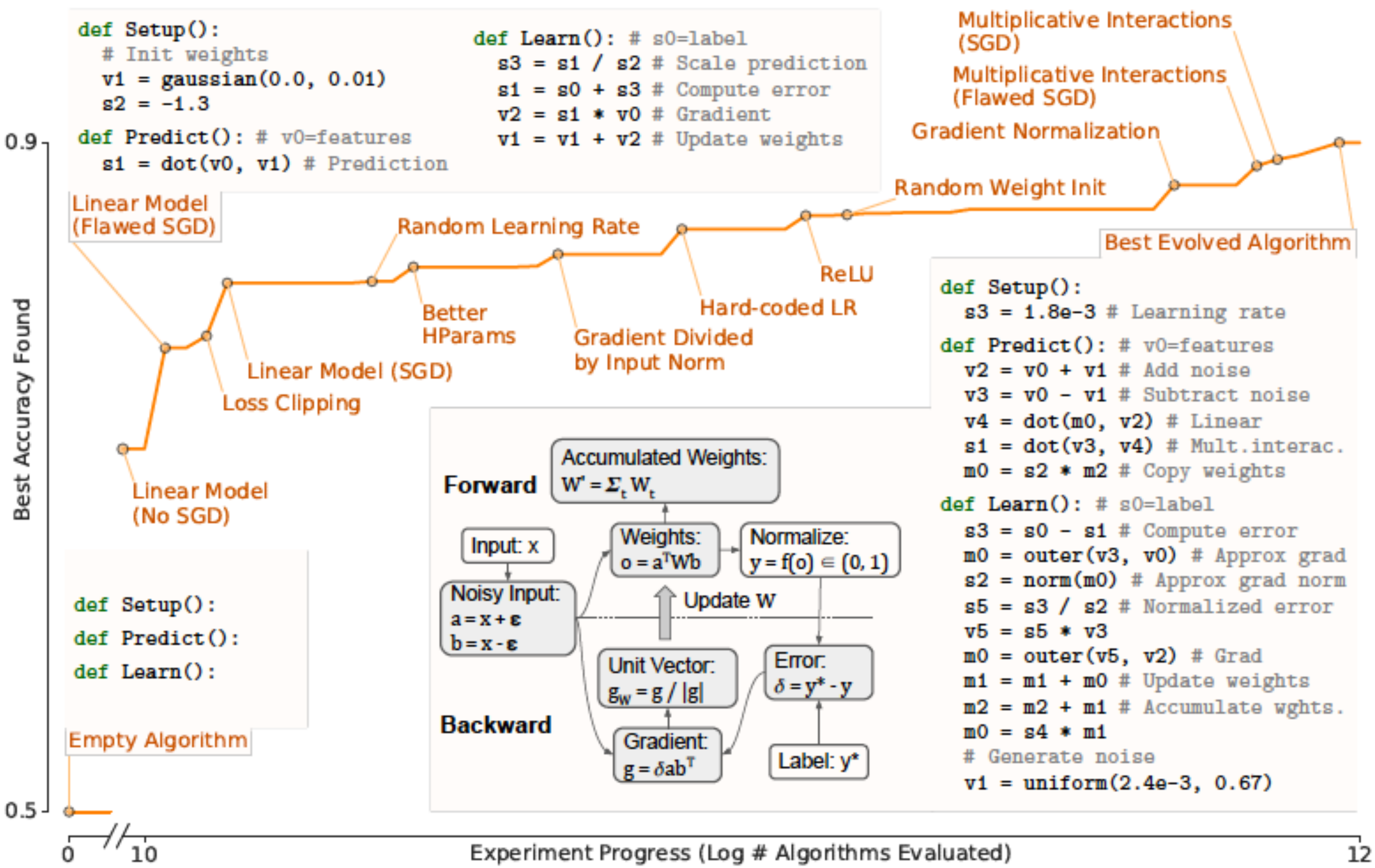


Figure 6: Progress of one evolution experiment on projected binary CIFAR-10. Callouts indicate some beneficial discoveries. We also print the code for the initial, an intermediate, and the final algorithm. The last is explained through the flow diagram. It outperforms a simple fully connected neural network on held-out test data and transfers to features 10x its size. Code notation is the same as in Figure 5.

There are a couple of other points in the paper, but (to me) not as important.

Discussion points:

- should we see autoML as a way to generate the actual ML algorithms we want, or instead look at the algos autoML finds, identify good innovations within them, and then use those innovations in hand-designed algos?
- why not implement a “smarter” optimization than evolution? (E.g., backprop or RL)
- what’s preventing the next stage of recursion: autoML that generates autoML that…
- should we be disappointed that the key innovations autoML-zero found are ones we already knew (e.g., is the “discovery” potential of the tool limited by our ability to interpret the programs it generates?)