Machine Learning & Compilers

Optimisation heuristics are too course.

(aka. it doesn't take much to do better than -O3)

Better than -O3

```
#!/bin/sh

while true; do
    sort --random-sort < "cflags.txt" | head -n 20 | xargs gcc -O1 app.c
    time ./a.out
done
```

after 200 attempts, ~5% improvement

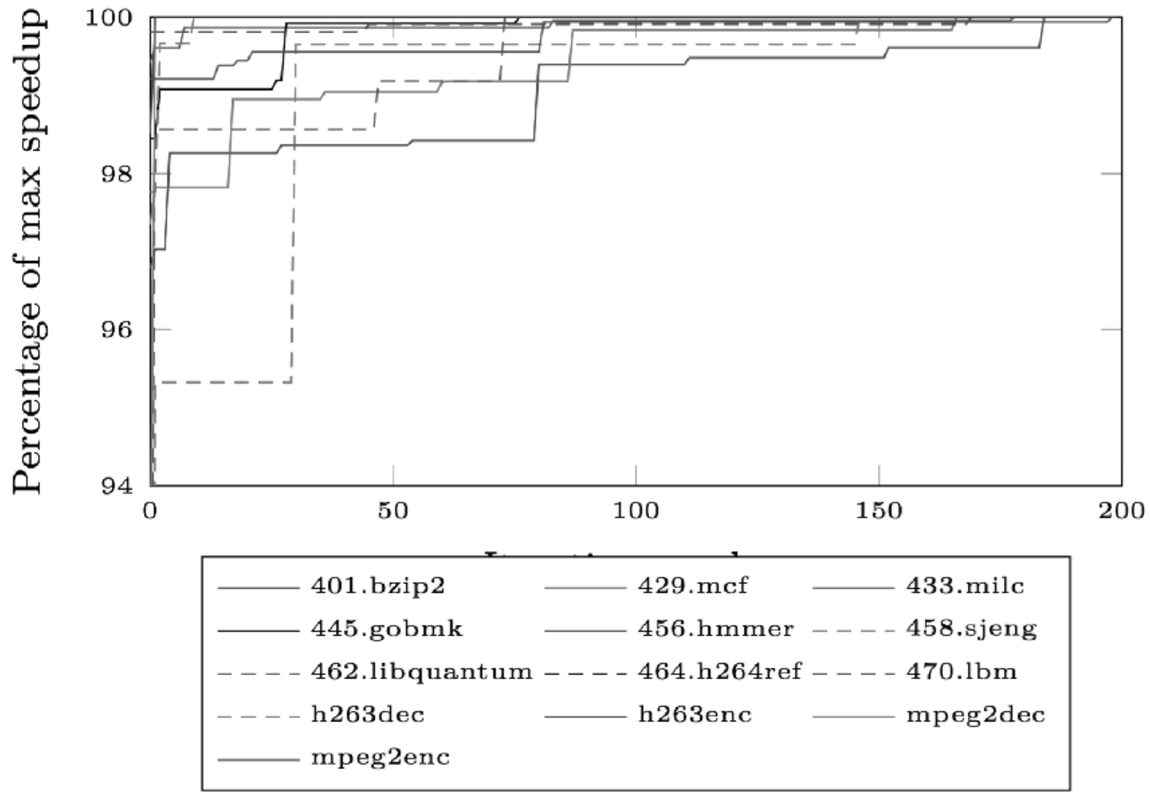


Figure 6: The convergence towards best speedup found for each benchmark.

*actual homework

200 attempts sounds like a lot...

(200 x 10 x 30 s ≈ 16 hrs)

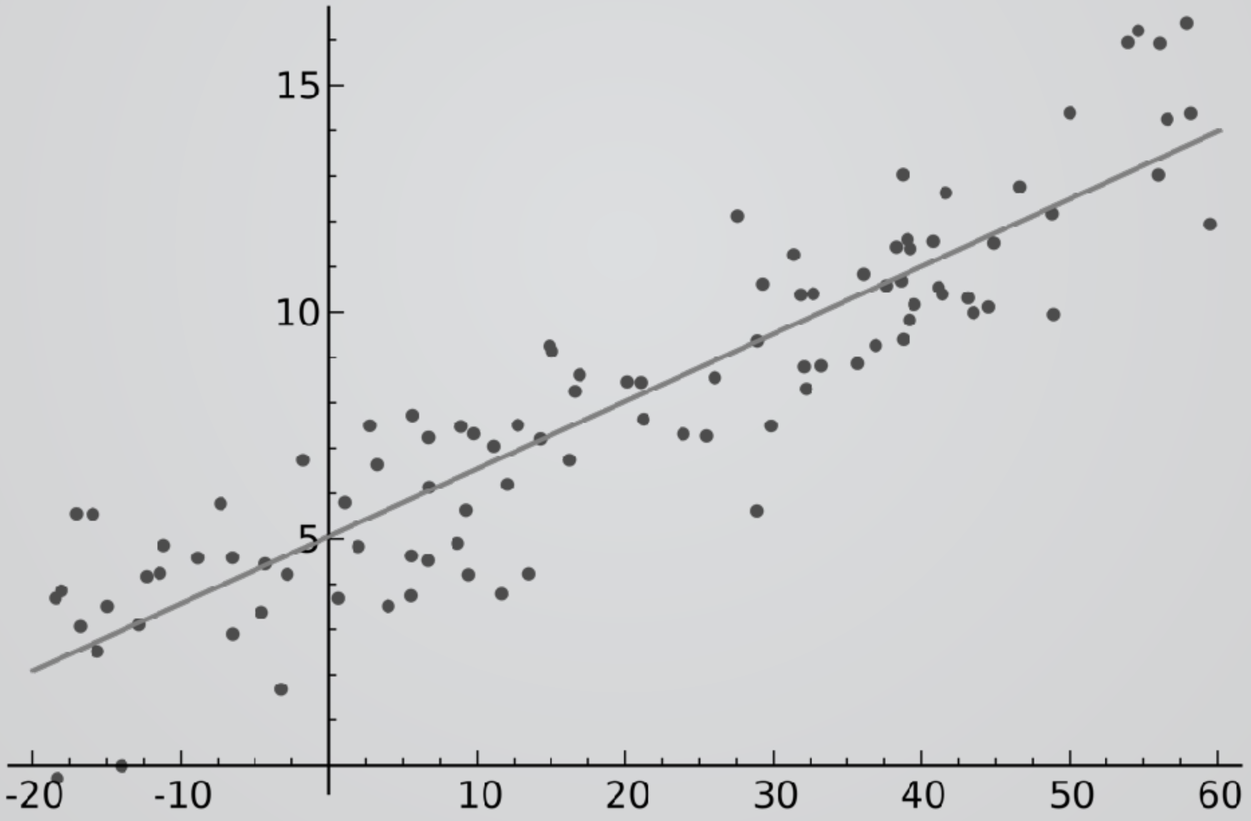

Drop in the ocean
250 GCC flags ≥ 2^{250} options ≈ 10^{75}
Atoms in the universe ≈ 10^{80}

Exhaustive search is not really* practical.

*in this universe

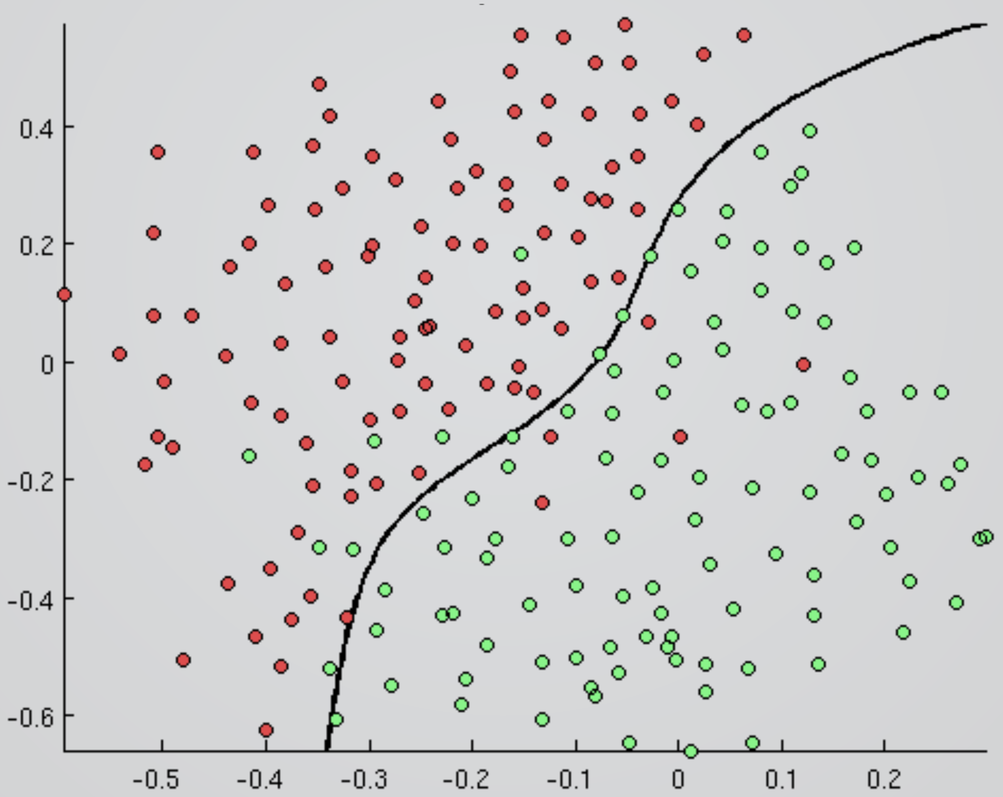
Machine Learning

Estimate y = f(x)



Machine Learning

Estimate y = f(x)



Machine Learning

Estimate y = f(x)

Optimisations

Cflags

Workgroup size

CPU or GPU

Features

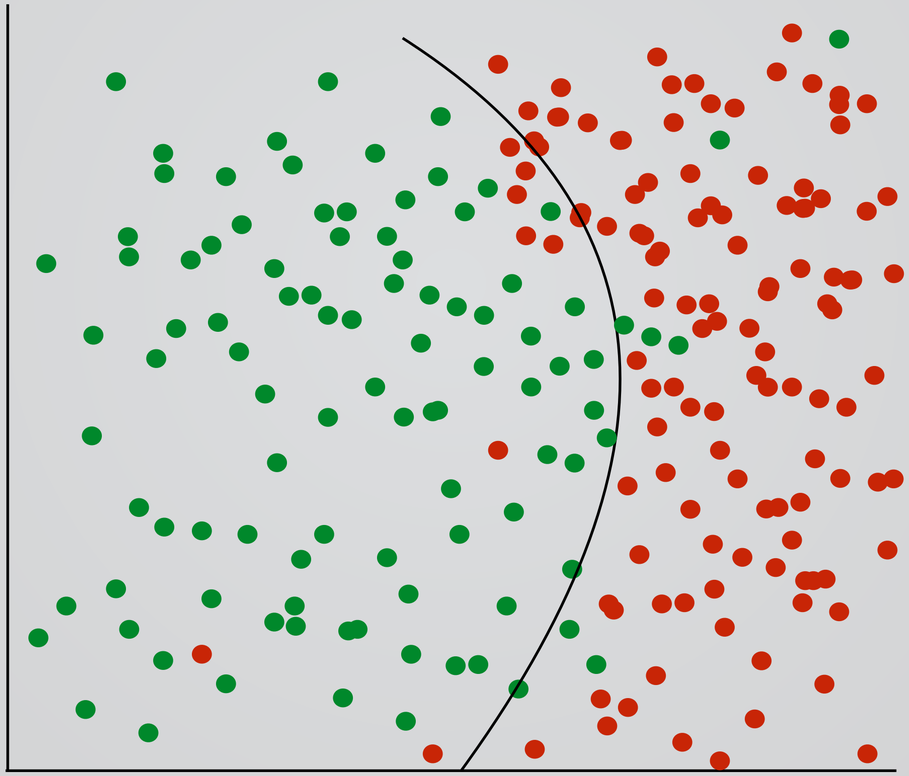
instructions

Arithmetic density

Dataset size

Machine Learning

Estimate y = f(x)

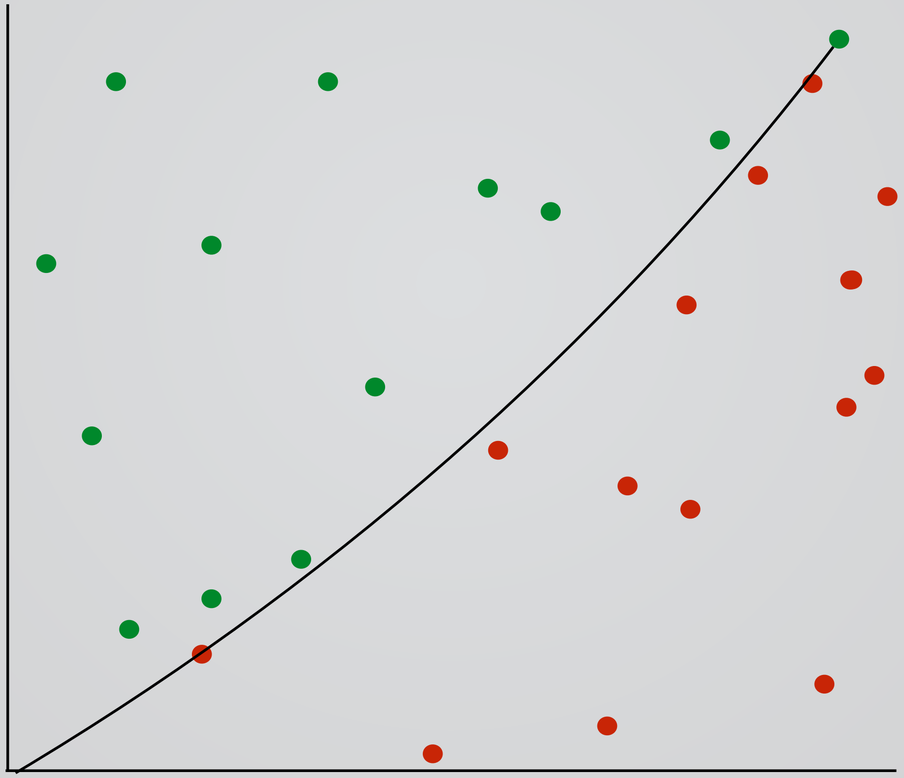


- Use a CPU
- Use a GPU

The idea.

Machine Learning

Estimate y = f(X)



- Use a CPU
- Use a GPU

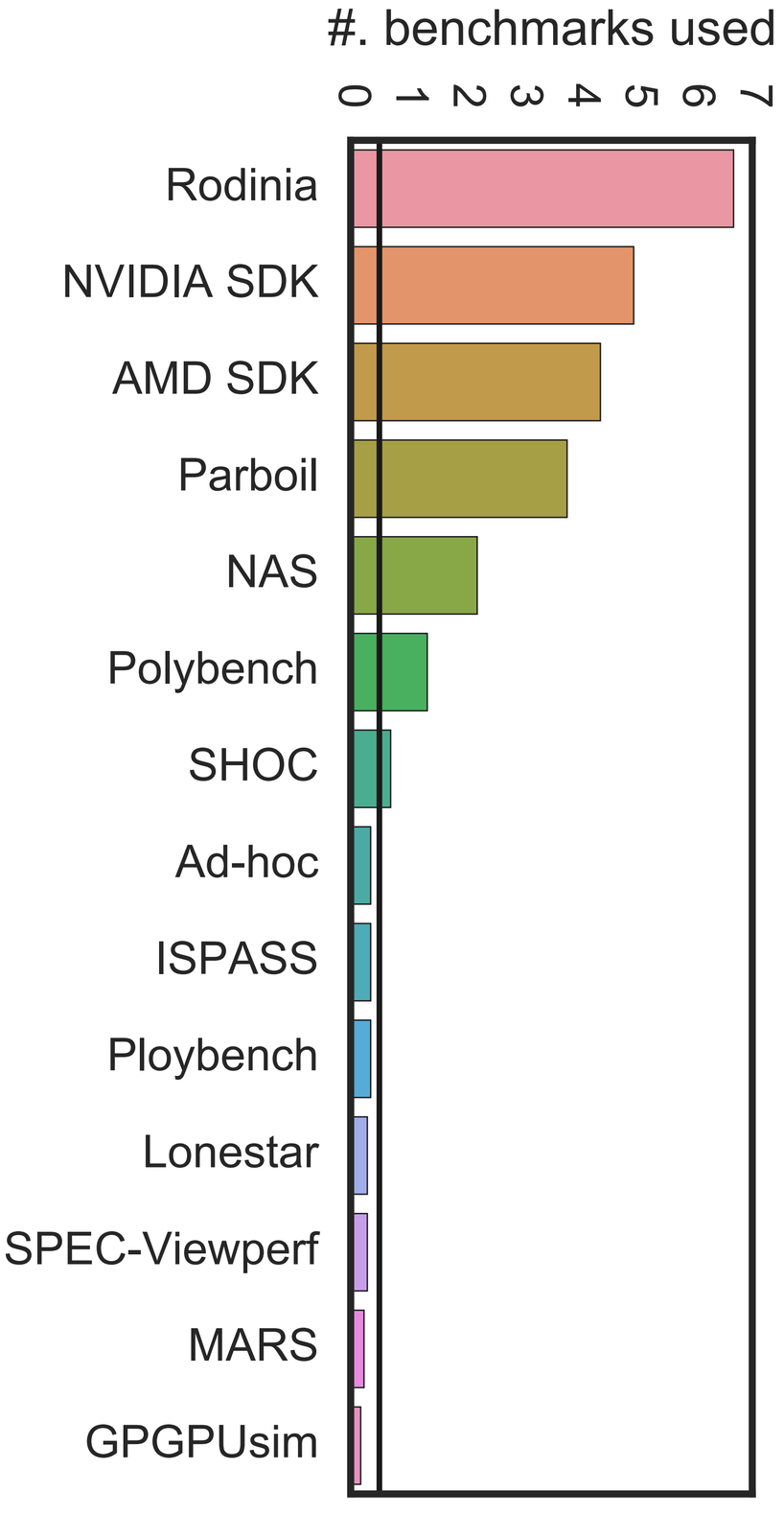
The reality.

(same data, wrong conclusions)

Synthesizing Benchmarks for Predictive Modeling

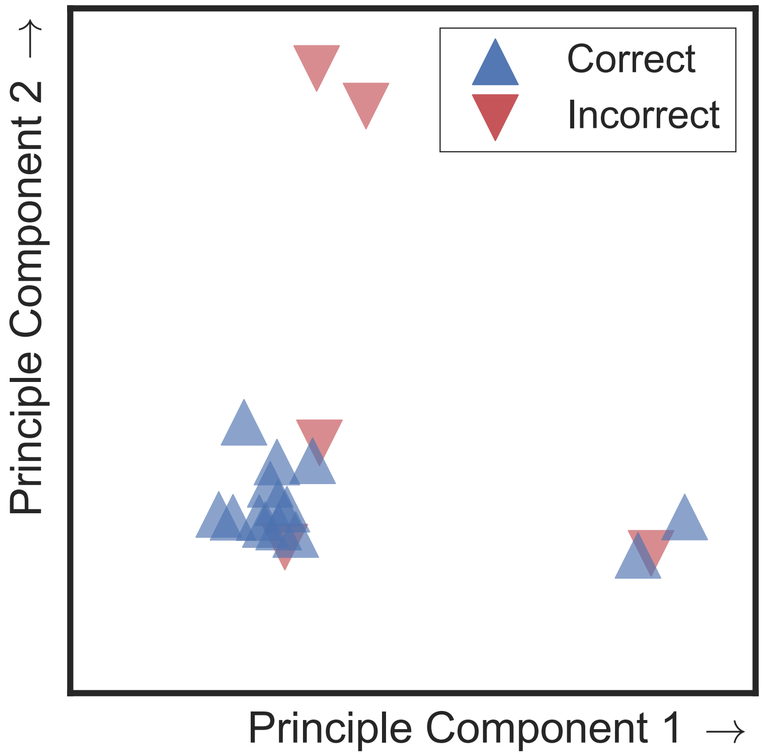
Why?

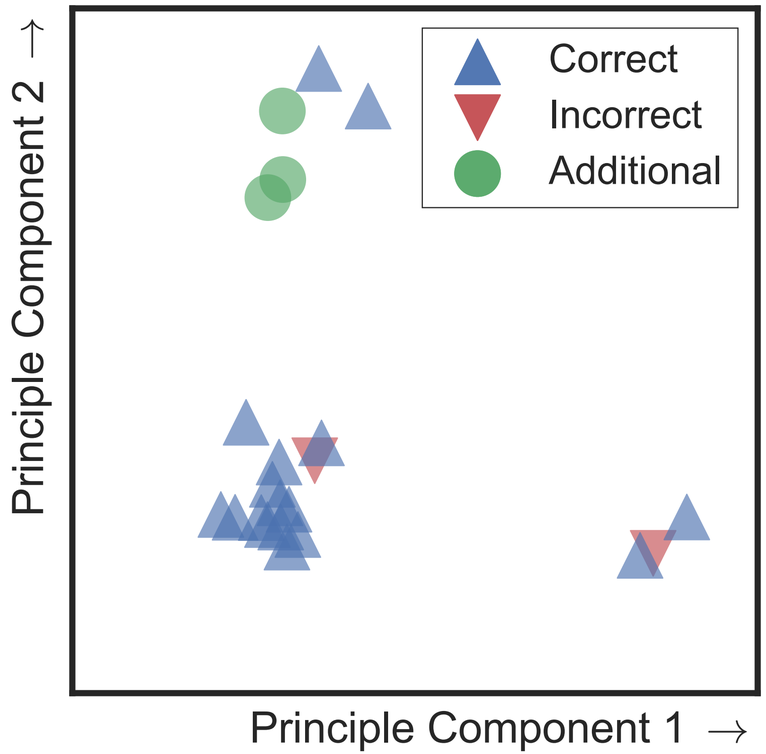
There aren’t enough benchmarks


#. benchmarks used
0
1
2
3
4
5
6
7
Rodinia
NVIDIA SDK
AMD SDK
Parboil
NAS
Polybench
SHOC
Ad-hoc
ISPASS
Ploybench
Lonestar
SPEC-Viewperf
MARS
GPGPUsim

Why?

More benchmarks = better models


Correct
Incorrect
Principle Component 2 →
Principle Component 1 →


Correct
Incorrect
Additional
Principle Component 2 →
Principle Component 1 →

Why?

No adequate solution

into the instance program. Otherwise, nothing is processed or generated by this code.

- The **genloop** construct facilitates repetitive generation. Consider the following example:

```
genloop loopvar:1:5
  ${access(${loopvar})}
end
```

This code produces 5 references to the feature `access`, each with a different value from 1 to 5 passed in as an argument. Note that this does not produce a loop in the instance program, but instead 5 consecutive versions of the code produced when the `access` feature is processed.

- The **geninclude** construct allows information in another file to be used in the current Genesis program. Genesis provides libraries to implement features that may be useful across different Genesis programs of the same target language. For example:

```
geninclude gen_c.glb
```

makes available those features defined in `gen_c.glb`, a library containing features that declare and initialize variables in C programs. For example, `varlistdeclare` is defined in `gen_c.glb`, which initializes C variables in an indicated varlist.

4. CASE STUDIES

In this section, we present three case studies to show the utility of Genesis. The case studies come from different application domains: auto-tuning of image filtering applications, generation of static program characteristics, and auto-tuning of stencil computations.

4.1 Image Filtering

The first case study deals with the generation of image filtering applications for training in performance auto-tuning on GPUs. These applications typically have two perfectly nested loops that sweep over two-dimensional images. Each element of an output image is computed as a function of a subset of the pixels in an input image. Specific image filtering applications differ in the subset and the function used to compute the output.

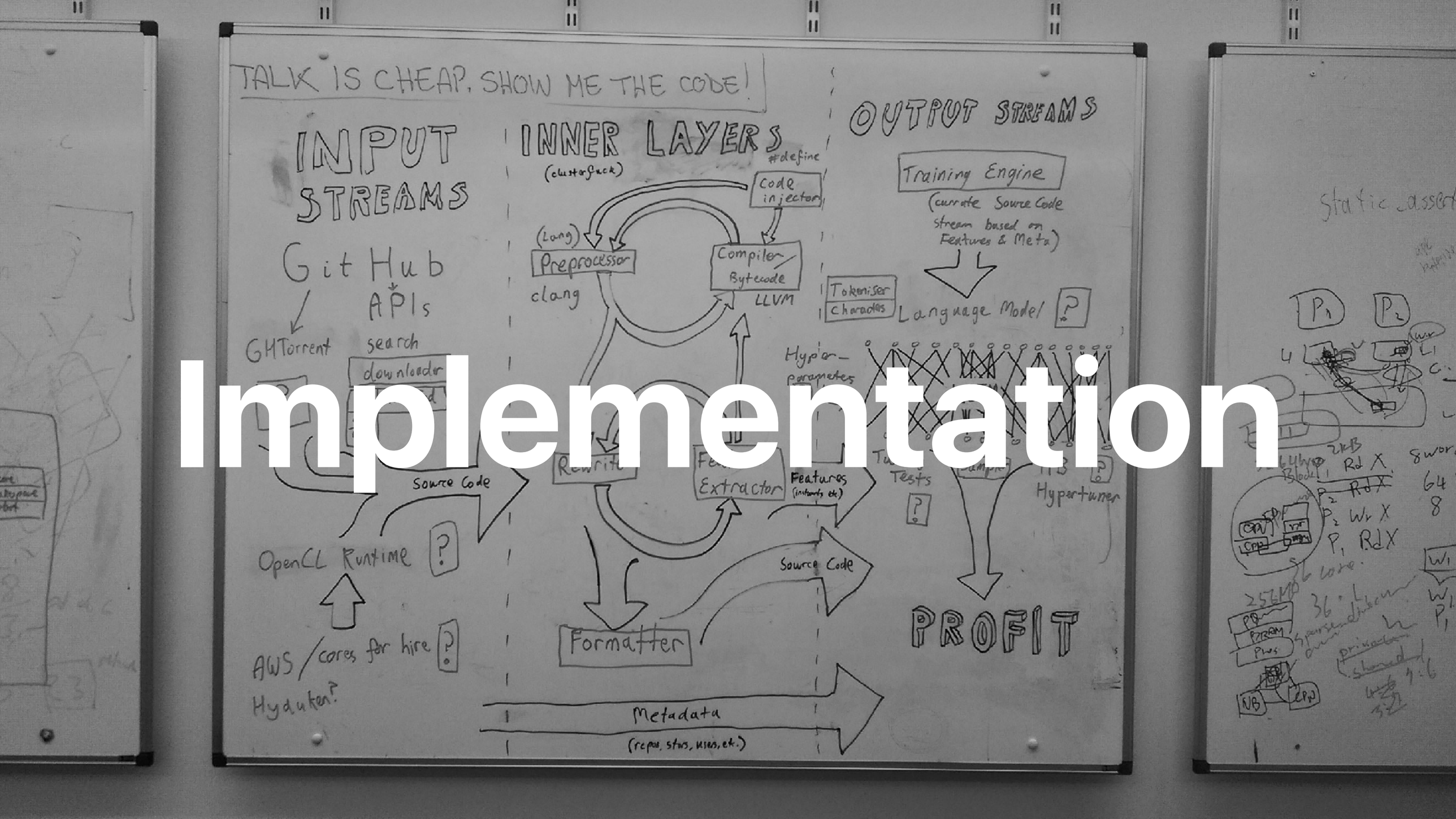
This case study focuses on memory performance, which is affected by the number and pattern of image accesses and the pixel computations in the loop nest. Thus, we model the body of the loop nest as one or more *read epochs* followed by a *write epoch*, where an *epoch* is a sequence of computations followed by a memory access. We wish to generate a number of such programs where the number of read epochs,

```
1  begin genesis
2
3    geninclude gen_c.glb
4
5    global
6      distribution epochdist = {1:10}
7      distribution numvardist = {8;16;32}
8      distribution compdist = {1:20}
9      distribution coefdist = {0:7}
10     distribution offsdist = {0:15}
11   end
12
13   program
14     value numepochs sample epochdist
15     value numvars sample numvardist
16     varlist temp[${numvars}]
17   end
18
19   feature computation
20     variable dest,src1,src2,src3 from temp
21     ${dest} = ${src1} * ${src2} + ${src3};
22   end
23
24   feature read_access
25     variable dest from temp
26     value coef1,coef2 sample coefdist
27     value offs sample offsdist
28     ${dest} = arr_in[${coef1}*it0 + ${coef2}*
           it1 +${offs}];
29   end
30
31   feature write_access
32     variable src from temp
33     value offs sample offsdist
34     arr_out[inner_tc*it0 + it1 +${offs}] = ${
           src};
35   end
36
37   feature epoch (epoch_type)
38     value numcomps sample compdist
39     genloop i:1:${numcomps}
40       ${computation}
41     end
42     genif ${epoch_type} eq "read"
43       ${read_access}
44     genelse
45       ${write_access}
46     end
47   end
48
49   feature epochs
50     genloop i:1:${numepochs}
51       ${epoch("read")}
52     end
53     ${epoch("write")}
54   end
55
56   generate 1000
57 end genesis
```

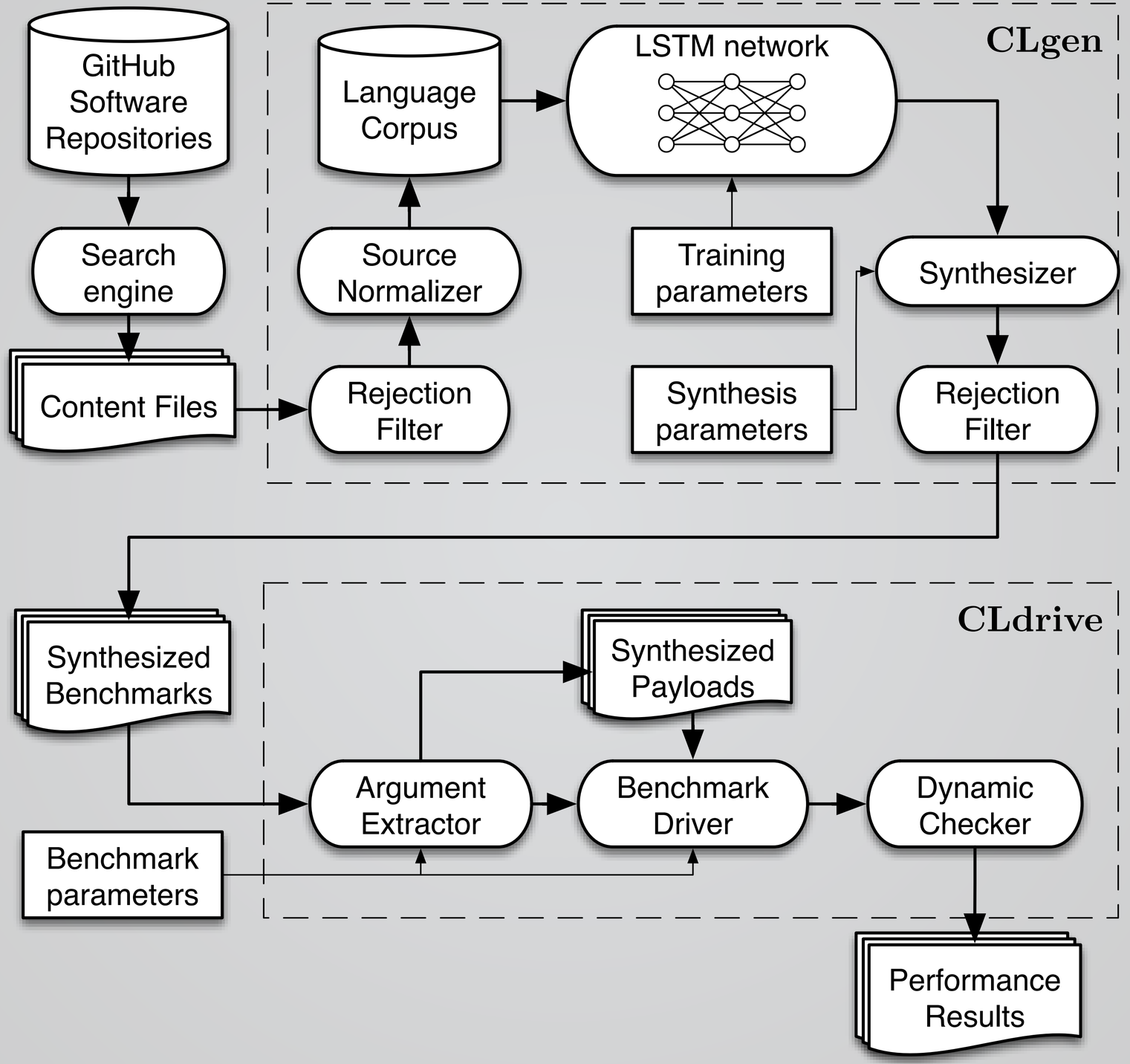
Figure 2: Genesis Program for Image Filtering

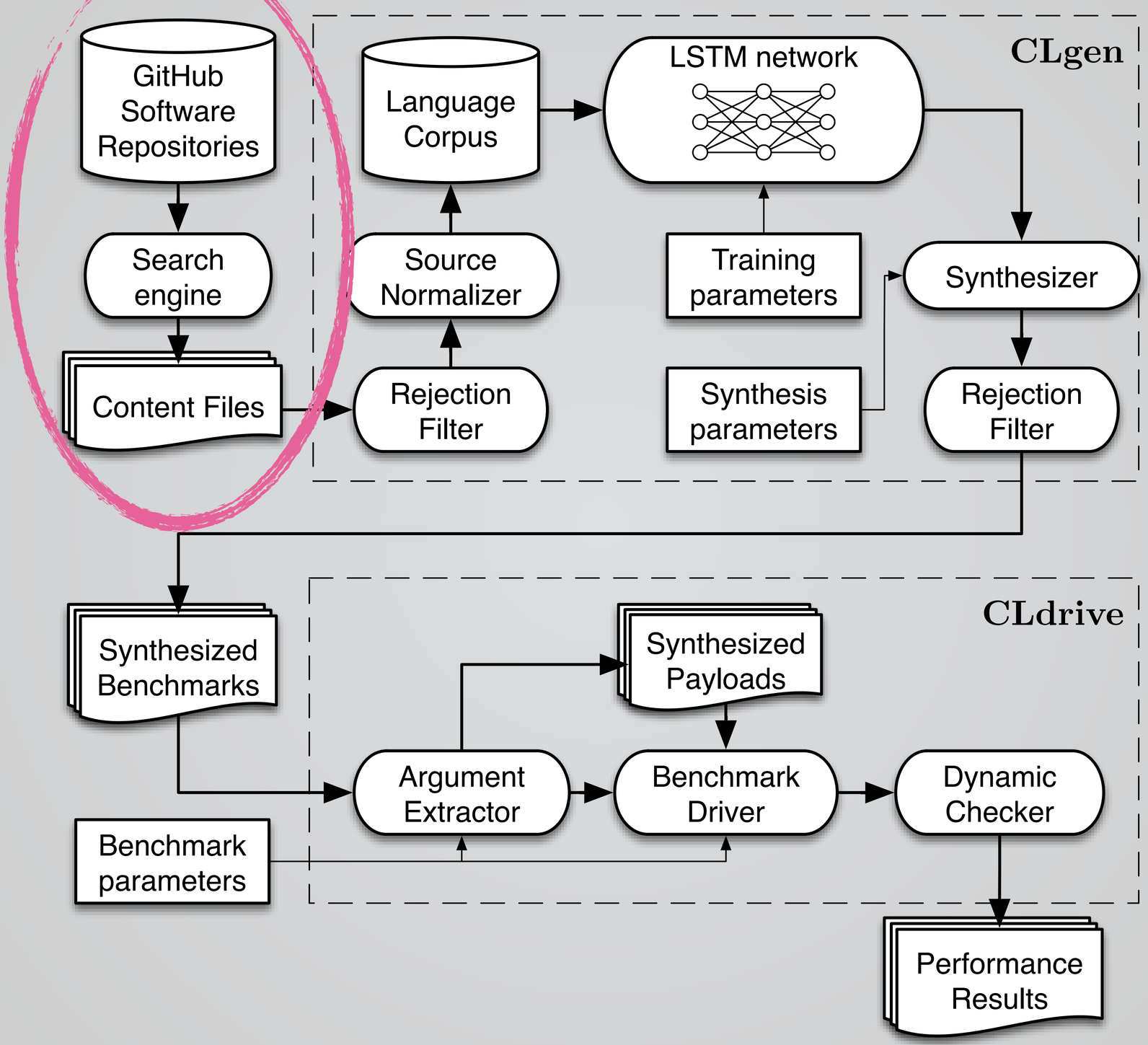
How?

Teach an AI to program from GitHub



Implementation


CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Language Corpus
Source Normalizer
LSTM network
Training parameters
Synthesis parameters
Synthesizer
Rejection Filter
CLdrive
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results


CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Source Normalizer
Language Corpus
LSTM network
Training parameters
Synthesis parameters
Synthesizer
Rejection Filter
CLdrive
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results

We teach an AI to code by showing it lots of code.

Huge repository of public knowledge: 

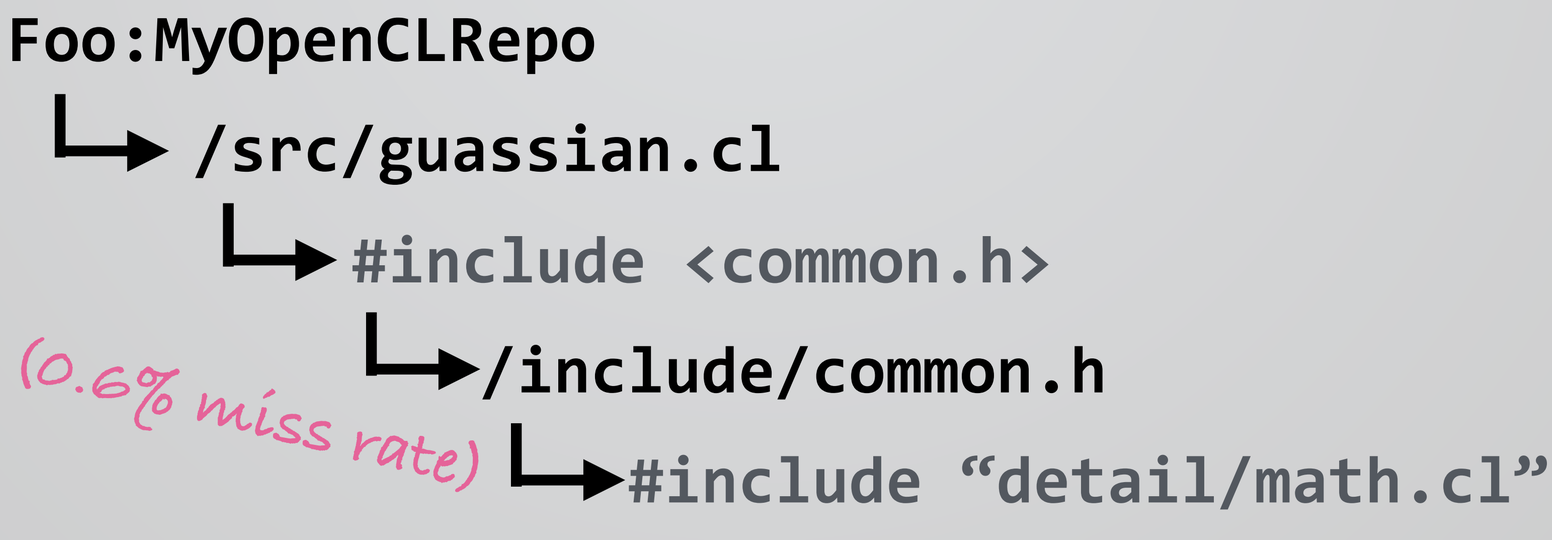
And they have an API :-)

```
$ curl https://api.github.com/search/repositories\?
q\=opencl\&sort\=stars\&order\=desc
{
  "total_count": 3155,
  "incomplete_results": false,
  "items": [
    {
      "id": 7296244,
      "name": "lwjgl3",
      "full_name": "LWJGL/lwjgl3",
```

OpenCL is not a first-class language.

Search repositories using loose keyword terms.
e.g. **`opencl`**, **`nvidia`**, **`gpu`**, **`cl`**, **`amd`**.

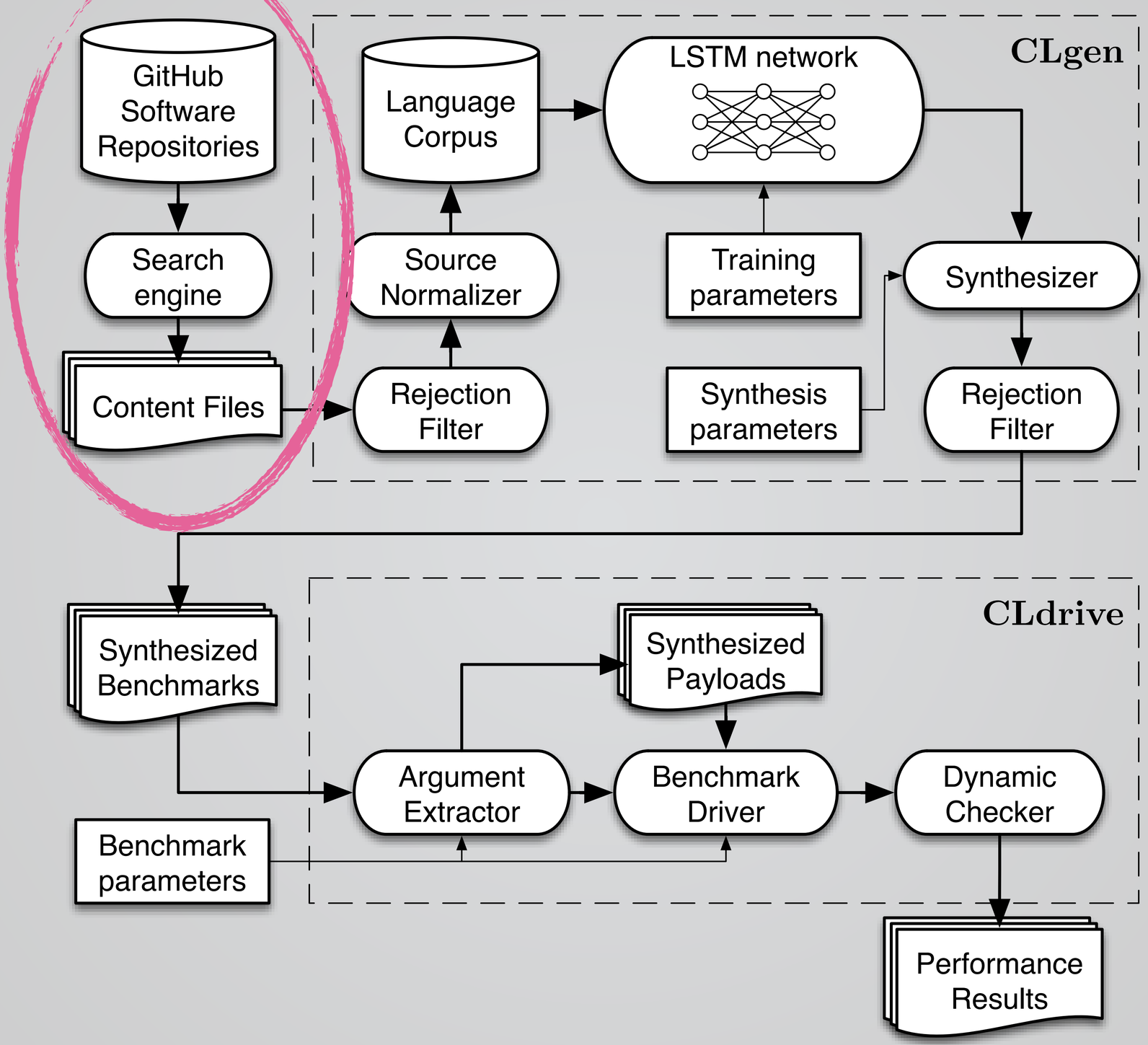
Recursively iterate over git trees to get **`.cl`** files.

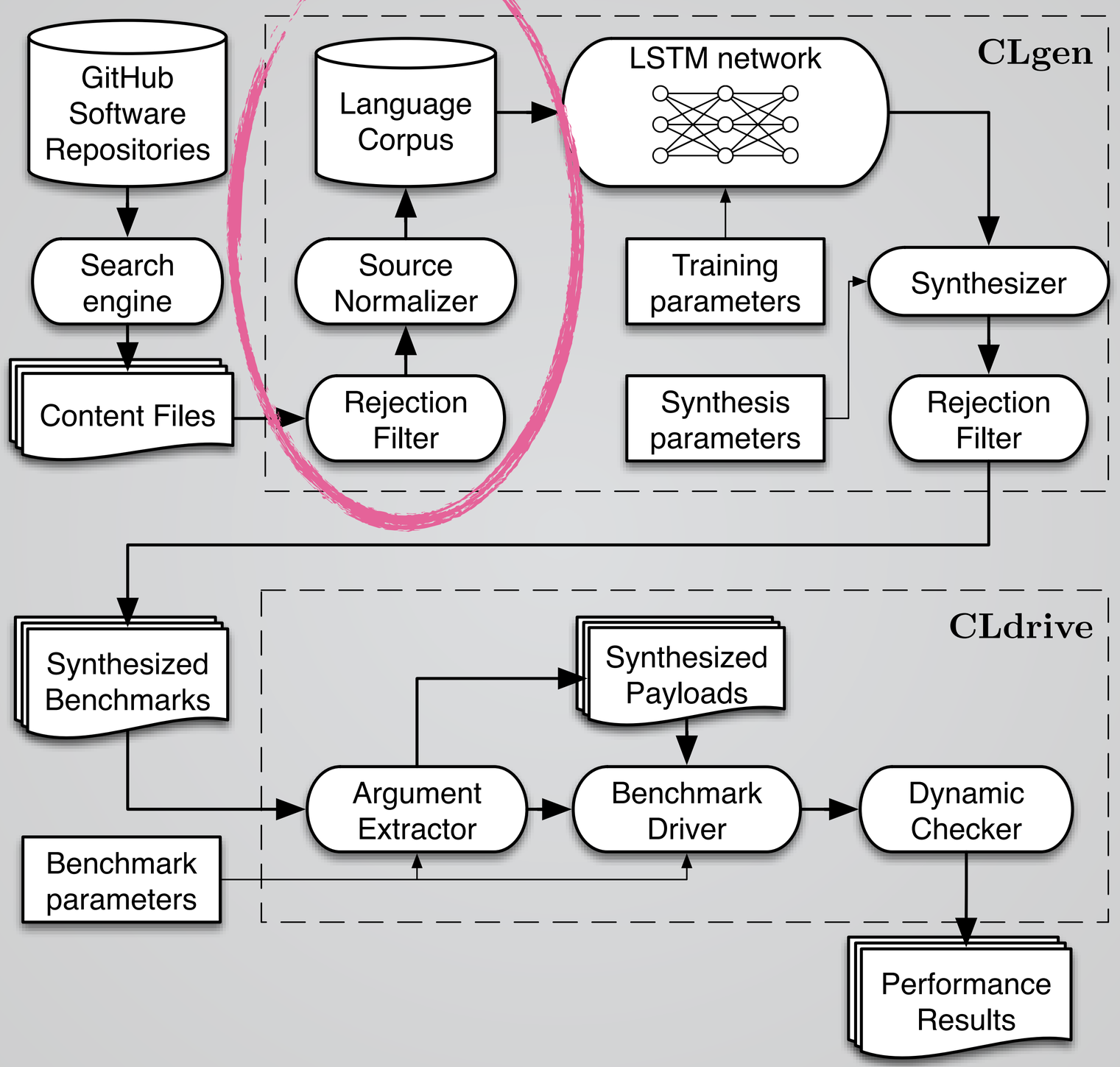


```
/* Copyright (C) 2004 Joe Bloggs <joe@bloggs.io>  */
//
//            DO WHAT THE FUCK YOU WANT TO PUBLIC LICENSE
//   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
//
//  0. You just DO WHAT THE FUCK YOU WANT TO.
#define CLAMPING
#define THRESHOLD_MIN 1.0f
#define THRESHOLD_MAX 1.0f

float myclamp(float in) {
#ifdef CLAMPING
    return in > THRESHOLD_MAX ? THRESHOLD_MAX : in < THRESHOLD_MIN ? THRESHOLD_MIN : in;
#else
    return in;
#endif // CLAMPING
}

// Do something really flipping cool
__kernel void findAllNodesMergedAabb(__global float* in, __global float* out, int num_elems)
{
    //
    //
    int id = get_global_id(0);
    if (id < num_elems)
    {


        out[id] = myclamp(in[id]);
    }
}
```

x 8078 files

2.8 million lines


CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Source Normalizer
Language Corpus
LSTM network
Training parameters
Synthesis parameters
Synthesizer
Rejection Filter
CLdrive
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results


CLgen
GitHub Software Repositories
Search engine
Content Files
Language Corpus
Source Normalizer
Rejection Filter
LSTM network
Training parameters
Synthesis parameters
Synthesizer
Rejection Filter
CLdrive
Synthesized Benchmarks
Benchmark parameters
Synthesized Payloads
Argument Extractor
Benchmark Driver
Dynamic Checker
Performance Results

```
/* Copyright (C) 2004 Joe Bloggs <joe@bloggs.io>  */
//
//            DO WHAT THE FUCK YOU WANT TO PUBLIC LICENSE
//   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
//
//  0. You just DO WHAT THE FUCK YOU WANT TO.
#define CLAMPING
#define THRESHOLD_MIN 1.0f
#define THRESHOLD_MAX 1.0f

float myclamp(float in) {
#ifdef CLAMPING
    return in > THRESHOLD_MAX ? THRESHOLD_MAX : in < THRESHOLD_MIN ? THRESHOLD_MIN : in;
#else
    return in;
#endif // CLAMPING
}

// Do something really flipping cool
__kernel void findAllNodesMergedAabb(__global float* in, __global float* out, int num_elems)
{
    //
    //
    int id = get_global_id(0);
    if (id < num_elems)
    {


        out[id] = myclamp(in[id]);
    }
}
```

```
/* Copyright (C) 2004 Joe Bloggs <joe@bloggs.io>  */
//
//            DO WHAT THE FUCK YOU WANT TO PUBLIC LICENSE
//   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
//
//  0. You just DO WHAT THE FUCK YOU WANT TO.
#define CLAMPING
#define THRESHOLD_MIN 1.0f
#define THRESHOLD_MAX 1.0f

float myclamp(float in) {
#ifdef CLAMPING
    return in > THRESHOLD_MAX ? THRESHOLD_MAX : in < THRESHOLD_MIN ? THRESHOLD_MIN : in;
#else
    return in;
#endif // CLAMPING
}

// Do something really flipping cool
__kernel void findAllNodesMergedAabb(__global float* in, __global float* out, int num_elems)
{
    //
    //
    int id = get_global_id(0);
    if (id < num_elems)
    {


        out[id] = myclamp(in[id]);
    }
}
```

Is this real, valid OpenCL?
Can we minimise non-functional variance?

```
/* Copyright (C) 2004 Joe Bloggs <joe@bloggs.io>  */
//
//            DO WHAT THE FUCK YOU WANT TO PUBLIC LICENSE
//   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
//
//  0. You just DO WHAT THE FUCK YOU WANT TO.
#define CLAMPING
#define THRESHOLD_MIN 1.0f
#define THRESHOLD_MAX 1.0f

float myclamp(float in) {
#ifdef CLAMPING
    return in > THRESHOLD_MAX ? THRESHOLD_MAX : in < THRESHOLD_MIN ? THRESHOLD_MIN : in;
#else
    return in;
#endif // CLAMPING
}

// Do something really flipping cool
__kernel void findAllNodesMergedAabb(__global float* in, __global float* out, int num_elems)
{
    //
    //
    int id = get_global_id(0);
    if (id < num_elems)
    {


        out[id] = myclamp(in[id]);
    }
}
```

Strip comments

Is this real, valid OpenCL?

Can we minimise non-functional variance?

```
/* Copyright (C) 2004 Joe Bloggs <joe@bloggs.io>  */
//
//            DO WHAT THE FUCK YOU WANT TO PUBLIC LICENSE
//   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
//
//  0. You just DO WHAT THE FUCK YOU WANT TO.
#define CLAMPING
#define THRESHOLD_MIN 1.0f
#define THRESHOLD_MAX 1.0f

float myclamp(float in) {
#ifdef CLAMPING
    return in > THRESHOLD_MAX ? THRESHOLD_MAX : in < THRESHOLD_MIN ? THRESHOLD_MIN : in;
#else
    return in;
#endif // CLAMPING
}

// Do something really flipping cool
__kernel void findAllNodesMergedAabb(__global float* in, __global float* out, int num_elems)
{
    //
    //
    int id = get_global_id(0);
    if (id < num_elems)
    {


        out[id] = myclamp(in[id]);
    }
}
```

Strip comments

Is this real, valid OpenCL?
Can we minimise non-functional variance?

Skip comments

```
#define CLAMPING
#define THRESHOLD_MIN 1.0f
#define THRESHOLD_MAX 1.0f

float myclamp(float in) {
#ifdef CLAMPING
    return in > THRESHOLD_MAX ? THRESHOLD_MAX : in < THRESHOLD_MIN ? THRESHOLD_MIN : in;
#else
    return in;
#endif
}

__kernel void findAllNodesMergedAabb(__global float* in, __global float* out, int num_elems)
{
    int id = get_global_id(0);
    if (id < num_elems)
    {


        out[id] = myclamp(in[id]);
    }
}
```

~~Strip comments~~
Preprocess

```
#define CLAMPING
#define THRESHOLD_MIN 1.0f
#define THRESHOLD_MAX 1.0f

float myclamp(float in) {
#ifdef CLAMPING
    return in > THRESHOLD_MAX ? THRESHOLD_MAX : in < THRESHOLD_MIN ? THRESHOLD_MIN : in;
#else
    return in;
#endif
}

__kernel void findAllNodesMergedAabb(__global float* in, __global float* out, int num_elems)
{
    int id = get_global_id(0);
    if (id < num_elems)
    {


        out[id] = myclamp(in[id]);
    }
}
```

Strip comments

Preprocess

```
#define CLAMPING
#define THRESHOLD_MIN 1.0f
#define THRESHOLD_MAX 1.0f

float myclamp(float in) {
#ifdef CLAMPING
    return in > THRESHOLD_MAX ? THRESHOLD_MAX : in < THRESHOLD_MIN ? THRESHOLD_MIN : in;
#else
    return in;
#endif
}

__kernel void findAllNodesMergedAabb(__global float* in, __global float* out, int num_elems)
{
    int id = get_global_id(0);
    if (id < num_elems)
    {


        out[id] = myclamp(in[id]);
    }
}
```

~~Strip comments~~

~~Preprocess~~

```
float myclamp(float in) {
   return in > 1.0f ? 1.0f : in < 0.0f ? 0.0f : in;
}

__kernel void findAllNodesMergedAabb(__global float* in, __global float* out, int num_elems)
{
    int id = get_global_id(0);
    if (id < num_elems)
    {


        out[id] = myclamp(in[id]);
    }
}
```

~~Strip comments~~

~~Preprocess~~

```
float myclamp(float in) {
   return in > 1.0f ? 1.0f : in < 0.0f ? 0.0f : in;
}

__kernel void findAllNodesMergedAabb(__global float* in, __global float* out, int num_elems)
{
    int id = get_global_id(0);
    if (id < num_elems)
    {


        out[id] = myclamp(in[id]);
    }
}
```

Does it compile?

Does it contain instructions?

~~Strip comments~~

~~Preprocess~~

```
float myclamp(float in) {
   return in > 1.0f ? 1.0f : in < 0.0f ? 0.0f : in;
}

__kernel void findAllNodesMergedAabb(__global float* in, __global float* out, int num_elems)
{
    int id = get_global_id(0);
    if (id < num_elems)
    {


        out[id] = myclamp(in[id]);
    }
}
```

~~Does it compile?~~

~~Does it contain instructions?~~

~~Strip comments~~

~~Preprocess~~

Rewrite function names

```
float myclamp(float in) {
   return in > 1.0f ? 1.0f : in < 0.0f ? 0.0f : in;
}

__kernel void findAllNodesMergedAabb(__global float* in, __global float* out, int num_elems)
{
    int id = get_global_id(0);
    if (id < num_elems)
    {


        out[id] = myclamp(in[id]);
    }
}
```

~~Does it compile?~~

~~Does it contain instructions?~~

~~Strip comments~~

~~Preprocess~~

Rewrite function names

```
float myclamp(float in) {
   return in > 1.0f ? 1.0f : in < 0.0f ? 0.0f : in;
}

__kernel void findAllNodesMergedAabb(__global float* in, __global float* out, int num_elems)
{
    int id = get_global_id(0);
    if (id < num_elems)
    {


        out[id] = myclamp(in[id]);
    }
}
```

~~Does it compile?~~

~~Does it contain instructions?~~

~~Strip comments~~

~~Preprocess~~

~~Rewrite function names~~

```
float A(float in) {
   return in > 1.0f ? 1.0f : in < 0.0f ? 0.0f : in;
}

__kernel void B(__global float* in, __global float* out, int num_elems)
{
    int id = get_global_id(0);
    if (id < num_elems)
    {


        out[id] = A(in[id]);
    }
}
```

~~Does it compile?~~

~~Does it contain instructions?~~

```
float A(float in) {
   return in > 1.0f ? 1.0f : in < 0.0f ? 0.0f : in;
}

__kernel void B(__global float* in, __global float* out, int num_elems)
{
    int id = get_global_id(0);
    if (id < num_elems)
    {


        out[id] = A(in[id]);
    }
}
```

~~Strip comments~~

~~Preprocess~~

~~Rewrite function names~~

Rewrite variable names

~~Does it compile?~~

~~Does it contain instructions?~~

```
float A(float in) {
   return in > 1.0f ? 1.0f : in < 0.0f ? 0.0f : in;
}

__kernel void B(__global float* in, __global float* out, int num_elems)
{
    int id = get_global_id(0);
    if (id < num_elems)
    {


        out[id] = A(in[id]);
    }
}
```

~~Strip comments~~

~~Preprocess~~

~~Rewrite function names~~

Rewrite variable names

~~Does it compile?~~

~~Does it contain instructions?~~

```
float A(float a) {
   return a > 1.0f ? 1.0f : a < 0.0f ? 0.0f : a;
}

__kernel void B(__global float* a, __global float* b, int c)
{
    int d = get_global_id(0);
    if (d < c)
    {


        b[d] = A(a[d]);
    }
}
```

~~Strip comments~~

~~Preprocess~~

~~Rewrite function names~~

~~Rewrite variable names~~

~~Does it compile?~~

~~Does it contain instructions?~~

```
float A(float a) {
   return a > 1.0f ? 1.0f : a < 0.0f ? 0.0f : a;
}

__kernel void B(__global float* a, __global float* b, int c)
{
    int d = get_global_id(0);
    if (d < c)
    {


        b[d] = A(a[d]);
    }
}
```

~~Strip comments~~

~~Preprocess~~

~~Rewrite function names~~

~~Rewrite variable names~~

Enforce code style

~~Does it compile?~~

~~Does it contain instructions?~~

```
float A(float a) {
   return a > 1.0f ? 1.0f : a < 0.0f ? 0.0f : a;
}

__kernel void B(__global float* a, __global float* b, int c)
{
    int d = get_global_id(0);
    if (d < c)
    {

        b[d] = A(a[d]);
    }
}
```

~~Strip comments~~

~~Preprocess~~

~~Rewrite function names~~

~~Rewrite variable names~~

Enforce code style

~~Does it compile?~~

~~Does it contain instructions?~~

```
float A(float a) {
  return a > 1.0f ? 1.0f : in < 0.0f ? 0.0f : a;
}

__kernel void B(__global float* a, __global float* b, int c) {
  int d = get_global_id(0);
  if (d < c) {
    b[d] = A(a[d]);
  }
}
```

~~Strip comments~~

~~Preprocess~~

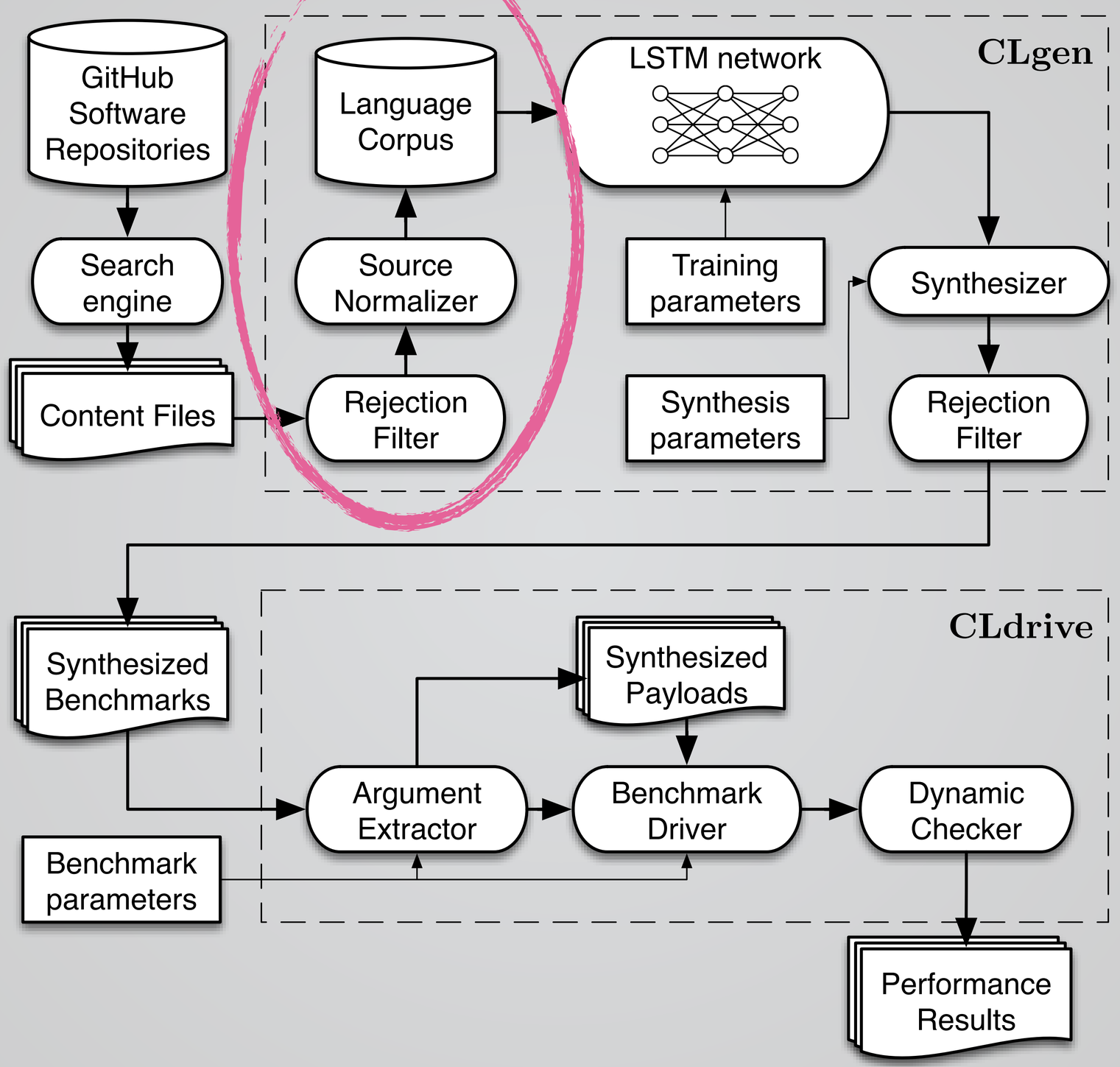
~~Rewrite function names~~

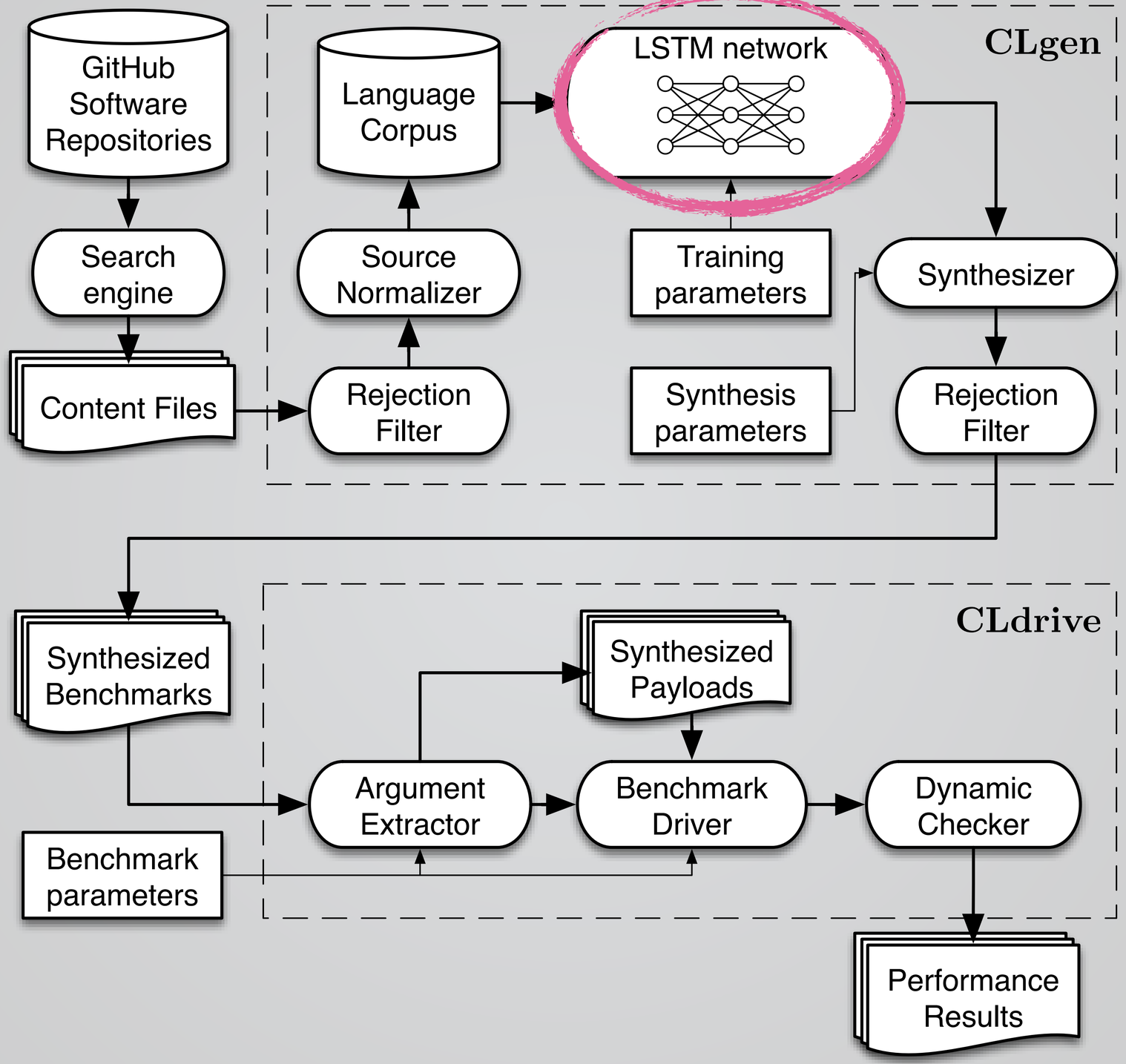
~~Rewrite variable names~~

~~Enforce code style~~

~~Does it compile?~~

~~Does it contain instructions?~~


CLgen
GitHub Software Repositories
Search engine
Content Files
Language Corpus
Source Normalizer
Rejection Filter
LSTM network
Training parameters
Synthesis parameters
Synthesizer
Rejection Filter
CLdrive
Synthesized Benchmarks
Benchmark parameters
Synthesized Payloads
Argument Extractor
Benchmark Driver
Dynamic Checker
Performance Results


CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Source Normalizer
Language Corpus
LSTM network
Training parameters
Synthesis parameters
Synthesizer
Rejection Filter
CLdrive
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results

Forward Pass

Input Gates:

$$x_\iota = \sum_{j \in N} w_{\iota j} y_j(\tau - 1) + \sum_{c \in C} w_{\iota c} s_c(\tau - 1)$$

$$y_\iota = f(x_\iota)$$

Forget Gates:

$$x_\phi = \sum_{j \in N} w_{\phi j} y_j(\tau - 1) + \sum_{c \in C} w_{\phi c} s_c(\tau - 1)$$

$$y_\phi = f(x_\phi)$$

Cells:

$$\forall c \in C,\ x_c = \sum_{j \in N} w_{cj} y_j(\tau - 1)$$

$$s_c = y_\phi s_c(\tau - 1) + y_\iota g(x_c)$$

Output Gates:

$$x_\omega = \sum_{j \in N} w_{\omega j} y_j(\tau - 1) + \sum_{c \in C} w_{\omega c} s_c(\tau)$$

$$y_\omega = f(x_\omega)$$

Backward Pass

$$define\ \ \delta_k(\tau) = \frac{\partial E(\tau)}{\partial x_k}$$

$$\delta_k(\tau) = y_k(\tau) - t_k(\tau) \qquad k \in output\ units$$

Cell Outputs:

$$\forall c \in C,\ define\ \epsilon_c = \sum_{j \in N} w_{jc} \delta_j(\tau + 1)$$

Output Gates:

$$\delta_\omega = f'(x_\omega) \sum_{c \in C} \epsilon_c h(s_c)$$

States:

$$\frac{\partial E}{\partial s_c}(\tau) = \epsilon_c y_\omega h'(s_c) + \frac{\partial E}{\partial s_c}(\tau + 1) y_\phi(\tau + 1)$$
$$+ \delta_\iota(\tau + 1) w_{\iota c} + \delta_\phi(\tau + 1) w_{\phi c} + \delta_\omega w_{\omega c}$$

Cells:

$$\forall c \in C,\ \delta_c = y_\iota g'(x_c) \frac{\partial E}{\partial s_c}$$

Forget Gates:

$$\delta_\phi = f'(x_\phi) \sum_{c \in C} \frac{\partial E}{\partial s_c} s_c(\tau - 1)$$

Input Gates:

$$\delta_\iota = f'(x_\iota) \sum_{c \in C} \frac{\partial E}{\partial s_c} g(x_c)$$

Update Weights

$$define\ \ E_{total}(S) = \sum_{\tau = \tau_0}^{\tau_1} E(\tau)$$

$$define\ \ \bigtriangledown_{ij}(S) = \frac{\partial E_{total}(S)}{\partial w_{ij}}$$

$$\implies \bigtriangledown_{ij}(S) = \sum_{\tau = \tau_0 + 1}^{\tau_1} \delta_i(\tau) y_j(\tau - 1)$$

$$\Delta w_{ij}(S) = -\alpha \bigtriangledown_{ij}(S) + m \Delta w_{ij}(S - 1)$$

Network Design

2048 nodes, 3 layers

Stochastic Gradient Descent

Initial learning rate 0.005

Train for 50 *epochs*

Learning rate decay every 5 *epochs*

Forward Pass

Input Gates:

$$x_\iota = \sum_{j \in N} w_{\iota j} y_j(\tau - 1) + \sum_{c \in C} w_{\iota c} s_c(\tau - 1)$$

$$y_\iota = f(x_\iota)$$

Forget Gates:

$$x_\phi = \sum_{j \in N} w_{\phi j} y_j(\tau - 1) + \sum_{c \in C} w_{\phi c} s_c(\tau - 1)$$

$$y_\phi = f(x_\phi)$$

Cells:

$$\forall c \in C,\ x_c = \sum_{j \in N} w_{cj} y_j(\tau - 1)$$

$$s_c = y_\phi s_c(\tau - 1) + y_\iota g(x_c)$$

Output Gates:

$$x_\omega = \sum_{j \in N} w_{\omega j} y_j(\tau - 1) + \sum_{c \in C} w_{\omega c} s_c(\tau)$$

$$y_\omega = f(x_\omega)$$

Backward Pass

$$define\ \ \delta_k(\tau) = \frac{\partial E(\tau)}{\partial x_k}$$

$$\delta_k(\tau) = y_k(\tau) - t_k(\tau) \qquad k \in output\ units$$

Cell Outputs:

$$\forall c \in C,\ define\ \epsilon_c = \sum_{j \in N} w_{j c} \delta_j(\tau + 1)$$

Output Gates:

$$\delta_\omega = f'(x_\omega) \sum_{c \in C} \epsilon_c h(s_c)$$

States:

$$\frac{\partial E}{\partial s_c}(\tau) = \epsilon_c y_\omega h'(s_c) + \frac{\partial E}{\partial s_c}(\tau + 1) y_\phi(\tau + 1)$$

$$+ \delta_\iota(\tau + 1) w_{\iota c} + \delta_\phi(\tau + 1) w_{\phi c} + \delta_\omega w_{\omega c}$$

Cells:

$$\forall c \in C,\ \delta_c = y_\iota g'(x_c) \frac{\partial E}{\partial s_c}$$

Forget Gates:

$$\delta_\phi = f'(x_\phi) \sum_{c \in C} \frac{\partial E}{\partial s_c} s_c(\tau - 1)$$

Input Gates:

$$\delta_\iota = f'(x_\iota) \sum_{c \in C} \frac{\partial E}{\partial s_c} g(x_c)$$

Update Weights

$$define\ \ E_{total}(S) = \sum_{\tau = \tau_0}^{\tau_1} E(\tau)$$

$$define\ \ \nabla_{ij}(S) = \frac{\partial E_{total}(S)}{\partial w_{ij}}$$

$$\implies \nabla_{ij}(S) = \sum_{\tau = \tau_0 + 1}^{\tau_1} \delta_i(\tau) y_j(\tau - 1)$$

$$\Delta w_{ij}(S) = -\alpha \nabla_{ij}(S) + m \Delta w_{ij}(S - 1)$$

Network Design

2048 nodes, 3 layers

Stochastic Gradient Descent

Initial learning rate 0.005

Train for 50 *epochs*

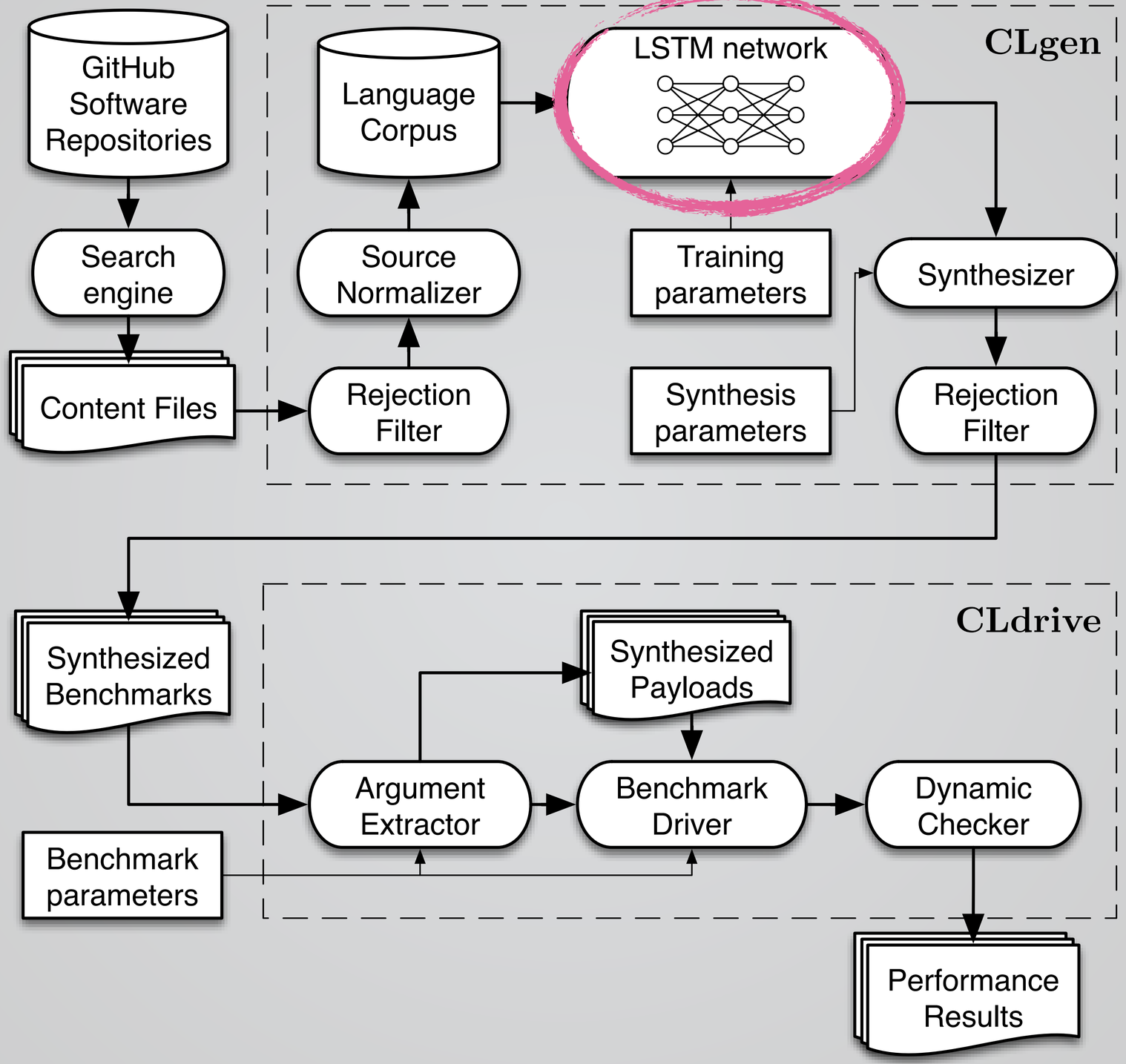
Learning rate decay every 5 *epochs*

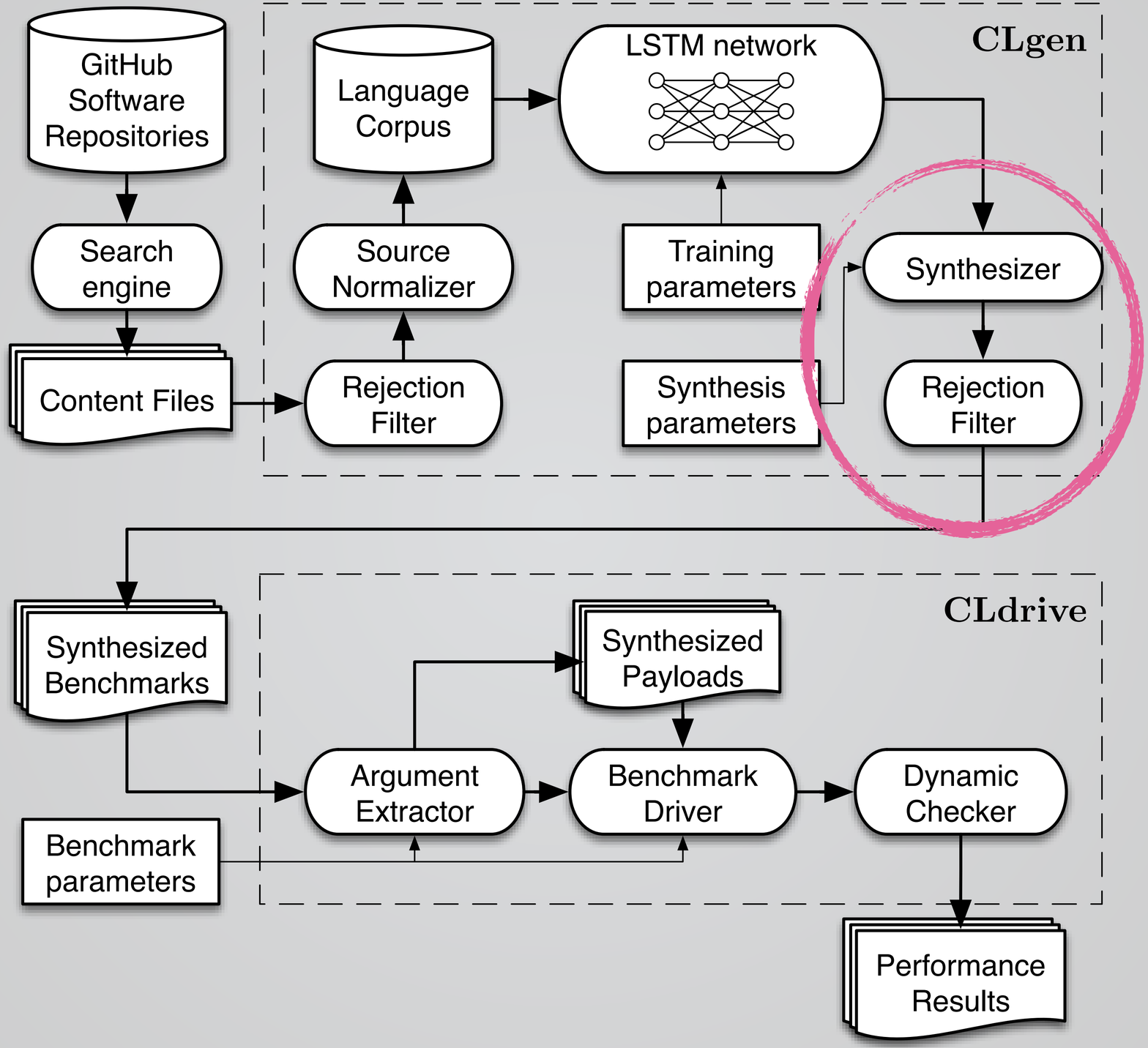
Machine Learning

Estimate y = f(x)

Distribution of characters

1278595 lines of OpenCL


CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Source Normalizer
Language Corpus
LSTM network
Training parameters
Synthesis parameters
Synthesizer
Rejection Filter
CLdrive
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results


CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Source Normalizer
Language Corpus
LSTM network
Training parameters
Synthesis parameters
Synthesizer
Rejection Filter
CLdrive
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results

Kernel Synthesis

```
S = ‘__kernel void A(__global float* a) {’
depth = 1
while depth > 0:
  c = predict_next_character(S)
  if c == ‘{’:
    depth += 1
  if c == ‘}’:
    depth -= 1
  S += c

return S
```

$

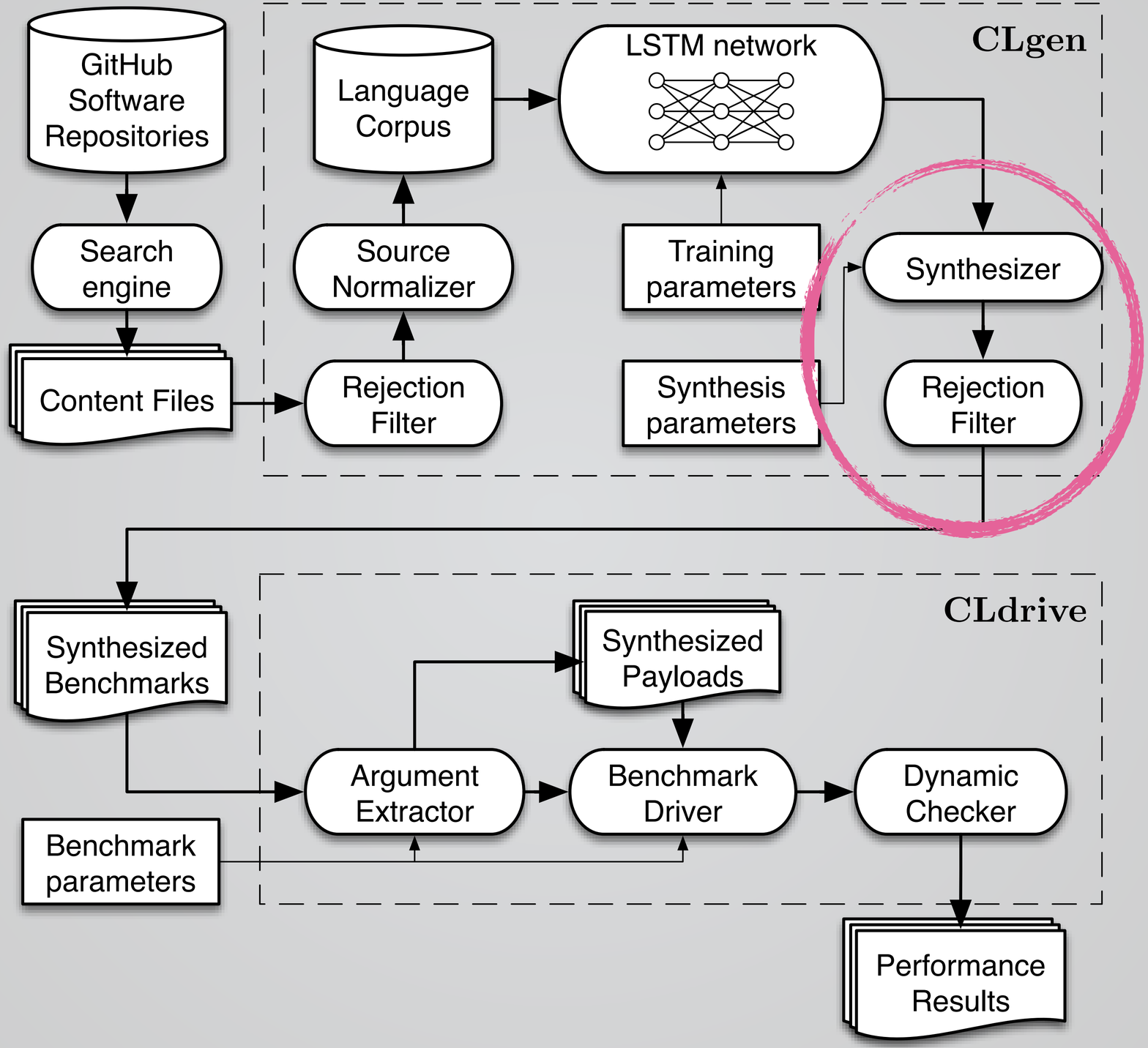
DEMO

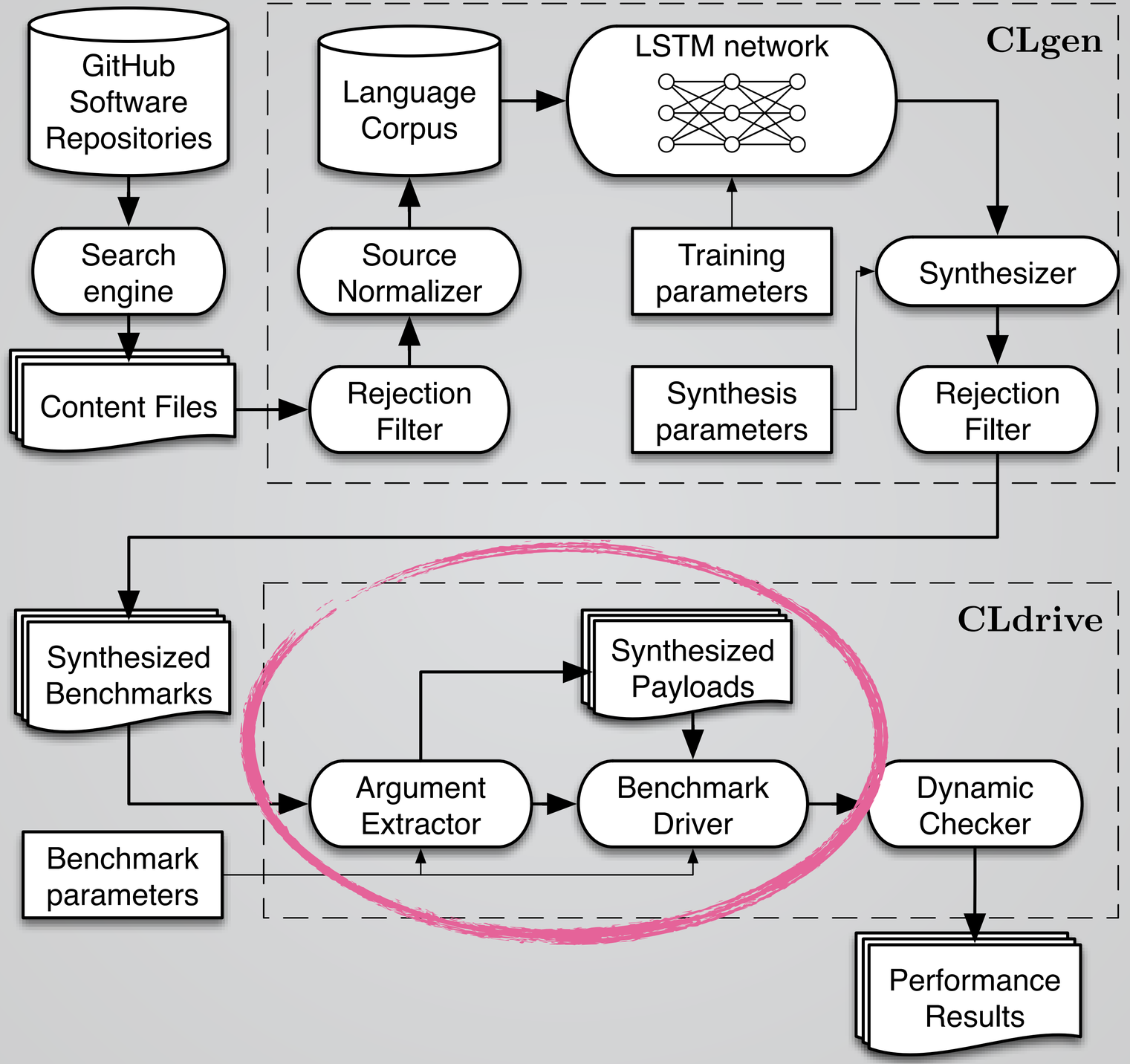
(you had to be there)

```
__kernel void A(__global float* a,
                __global float* b,
                __global float* c,
                const int d) {
  int e = get_global_id(0);
  float f = 0.0;
  for (int g = 0; g < d; g++) {
    c[g] = 0.0f;
  }
  barrier(1);

  a[get_global_id(0)] = 2*b[get_global_id(0)];
}
```

```
__kernel void A(__global float* a,
                __global float* b,
                __global float* c,
                const int d) {
  int e = get_global_id(0);
  if (e >= d) {
    return;
  }
  c[e] = a[e] + b[e] + 2 * a[e] + b[e] + 4;
}
```

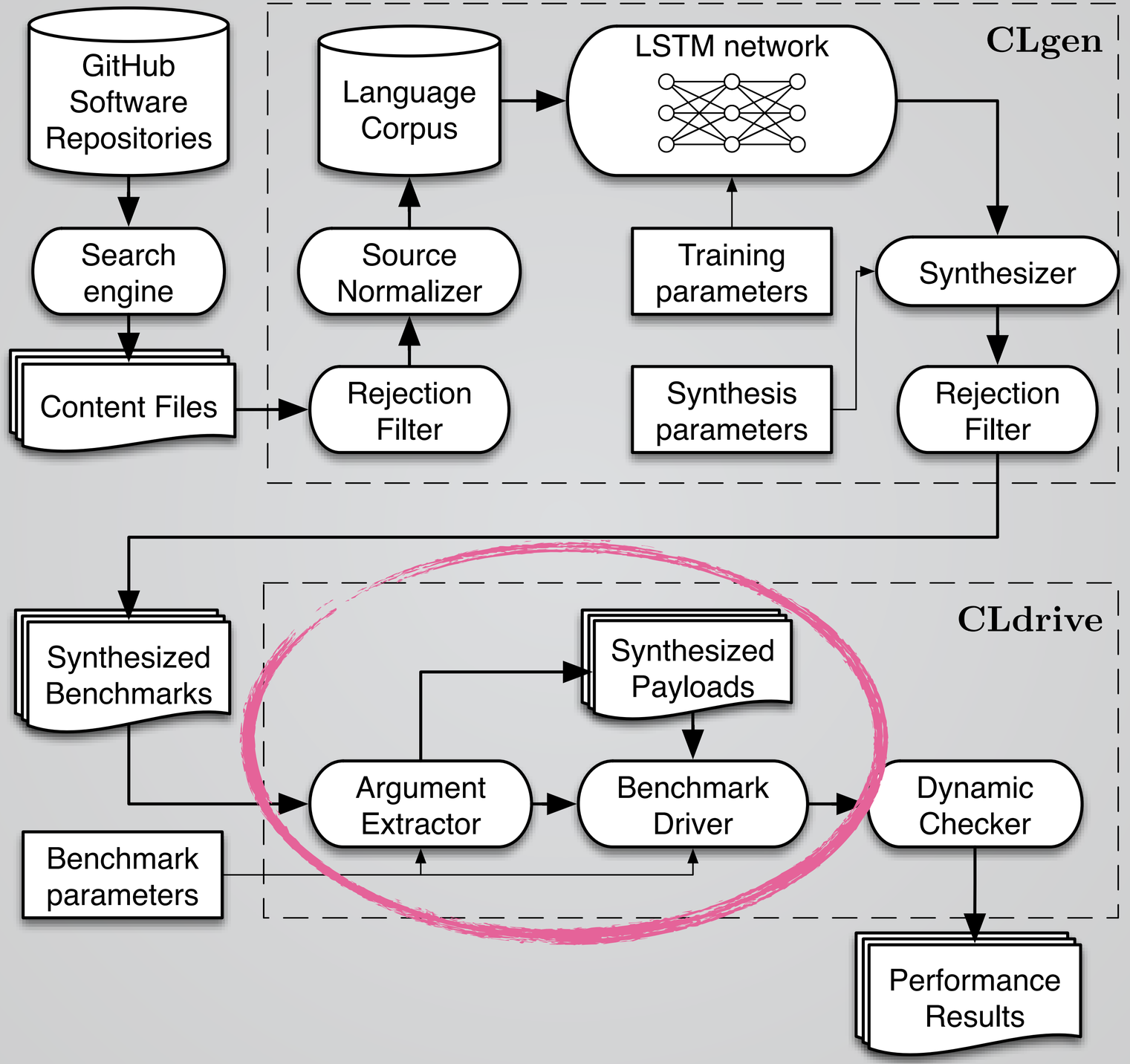
```
__kernel void A(__global float* a,
                __global float* b,
                __global float* c,
                const int d) {
  unsigned int e = get_global_id(0);
  float16 f = (float16)(0.0);
  for (unsigned int g = 0; g < d; g++) {
    float16 h = a[g];
    f.s0 += h.s0;
    f.s1 += h.s1;
    /* snip ... */
    f.sE += h.sE;
    f.sF += h.sF;
  }
  b[e] = f.s0 + f.s1 + f.s2 + f.s3 + f.s4 +
     f.s5 + f.s6 + f.s7 + f.s8 + f.s9 + f.sA +
     f.sB + f.sC + f.sD + f.sE + f.sF;
}
```

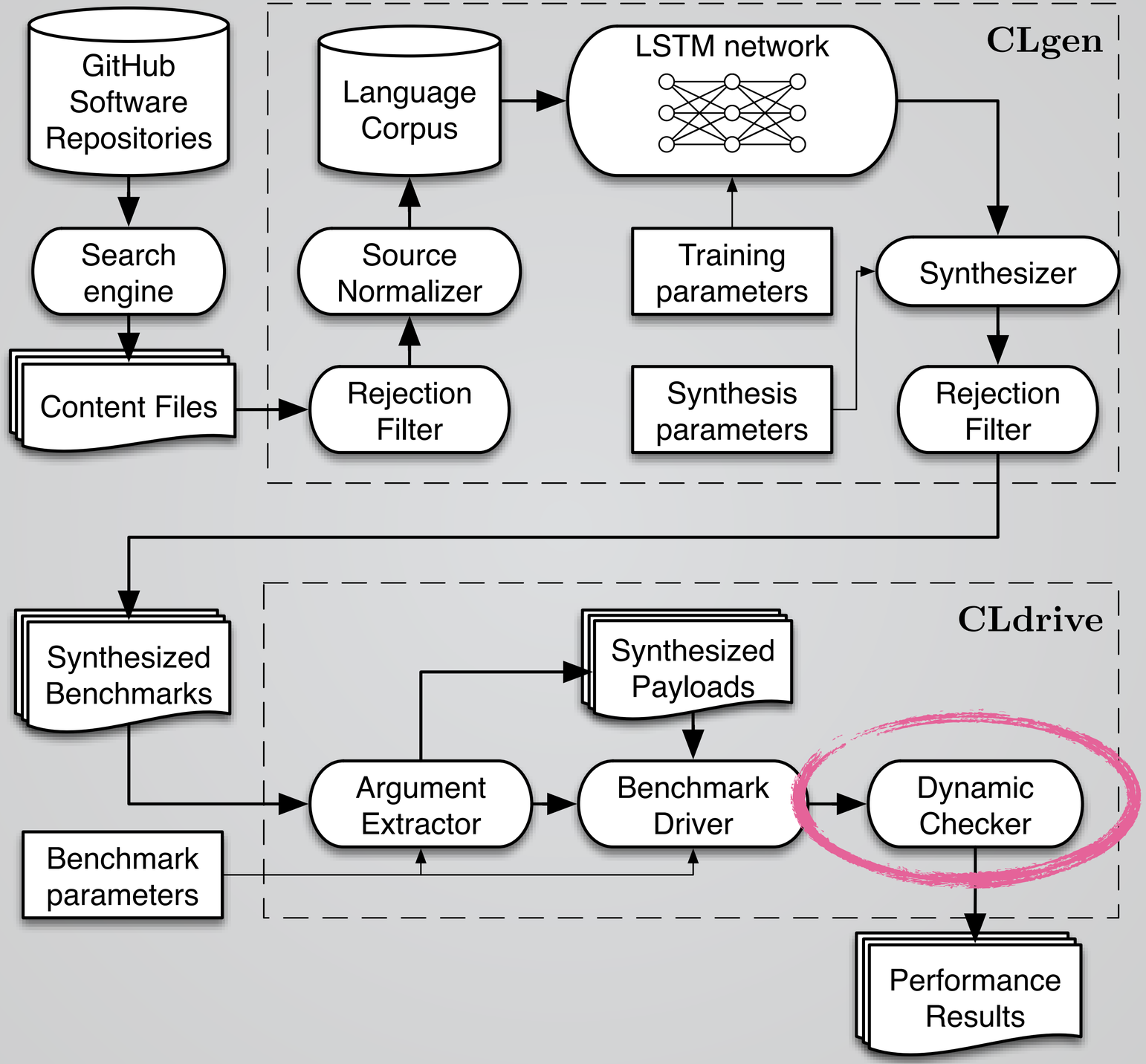

CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Source Normalizer
Language Corpus
LSTM network
Training parameters
Synthesis parameters
Synthesizer
Rejection Filter
CLdrive
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results


CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Source Normalizer
Language Corpus
LSTM network
Training parameters
Synthesis parameters
Synthesizer
Rejection Filter
CLdrive
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results

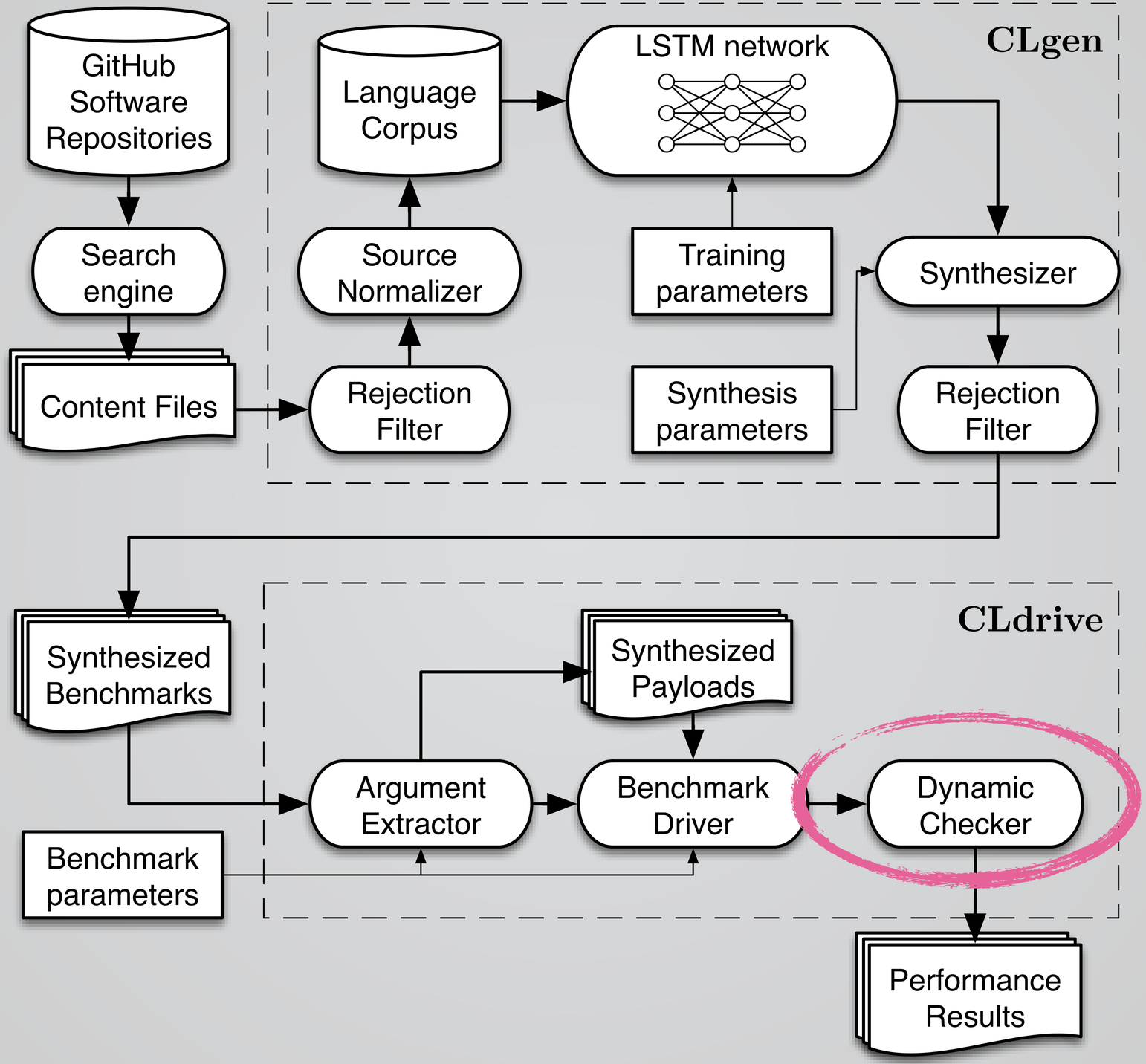
```
__kernel void A(__global float* a,
                __global float* b,
                __global float* c,
                const int d) {
  int e = get_global_id(0);
  if (e >= d) {
    return;
  }
  c[e] = a[e] + b[e] + 2 * a[e] + b[e] + 4;
}
```

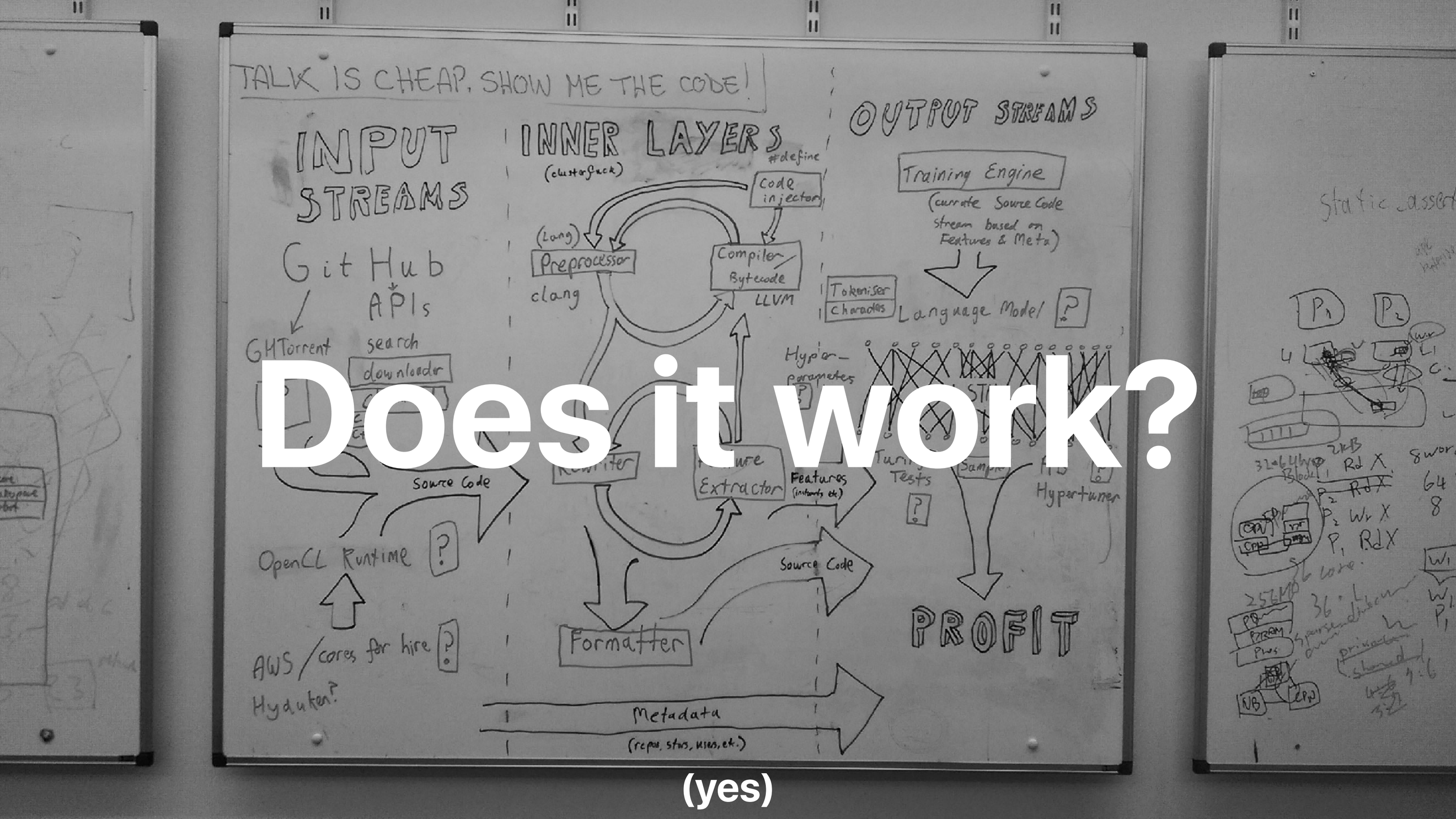
```
__kernel void A(__global float* a,
                __global float* b,
                __global float* c,
                const int d) {
  int e = get_global_id(0);
  if (e >= d) {
    return;
  }
  c[e] = a[e] + b[e] + 2 * a[e] + b[e] + 4;
}
```


CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Source Normalizer
Language Corpus
LSTM network
Training parameters
Synthesis parameters
Synthesizer
Rejection Filter
CLdrive
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results


CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Source Normalizer
Language Corpus
LSTM network
Training parameters
Synthesis parameters
Synthesizer
Rejection Filter
CLdrive
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results

```
_A = random_payload(_A)  # generate inputs
_B = random_payload(_B)
_C = copy(_C)
_D = copy(_B)

A_ = k(_A)  # compute outputs
B_ = k(_B)
C_ = k(_C)
D_ = k(_D)

# differential test
assert (A_ != _A || B_ != _B) else NO_OUTPUTS
assert (A_ != B_ || C_ != D_) else INPUT_INSENSITIVE
assert (A_ == C_ && B_ == D_) else NON_DETERMINISTIC
```


CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Source Normalizer
Language Corpus
LSTM network
Training parameters
Synthesis parameters
Synthesizer
Rejection Filter
CLdrive
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results


TALK IS CHEAP. SHOW ME THE CODE!
INPUT STREAMS
GitHub APIs
GHTorrent
search
downloader
Source Code
OpenCL Runtime
AWS / cores for hire
Hyduken?
INNER LAYERS
(clusterfuck)
#define
Code injector
(Lang)
Preprocessor
clang
Compiler
Bytecode
LLVM
Rewriter
Extractor
Features
Formatter
Source Code
Metadata
(repos, stars, users, etc.)
OUTPUT STREAMS
Training Engine
(curate Source Code stream based on Features & Meta)
Tokeniser
Characters
Language Model
Hyper-parameters
Tests
Sampler
Hypertuner
PROFIT
Does it work?
(yes)

Round 1

Player: **1002**, Robot: **998**

★☆☆☆☆ **What a piece of junk!**

By George S. Mitchell "gsmitchell"

I checked around Amazon as well as some other si tes, and decided these were what I needed. I read the reviews, looked at the pictures and bought two of these. I received them today. I unpacked one o f them, read the instructions and immediately filled out the return request for Amazon. These mounts are very, very heavy and require 6, yes SIX, screw s to mount to the wall. Are you kidding me? The T Vs are only 22 lbs! Two screws in the studs and th en more damage to the wall than is necessary. W hat a joke! And Lag bolts? Ever heard of screws? The other issue is that these are very hard to move . I bought an articulating wall mount for a reason, and I expect them to be able to move without a wr ench in my hand.

This side is the Robot

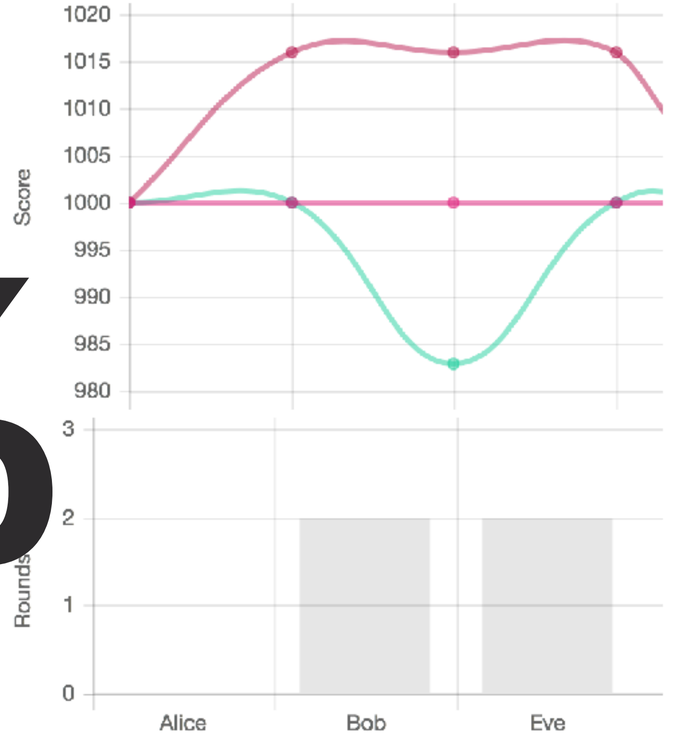
★★★★★ **Excellent for the price**

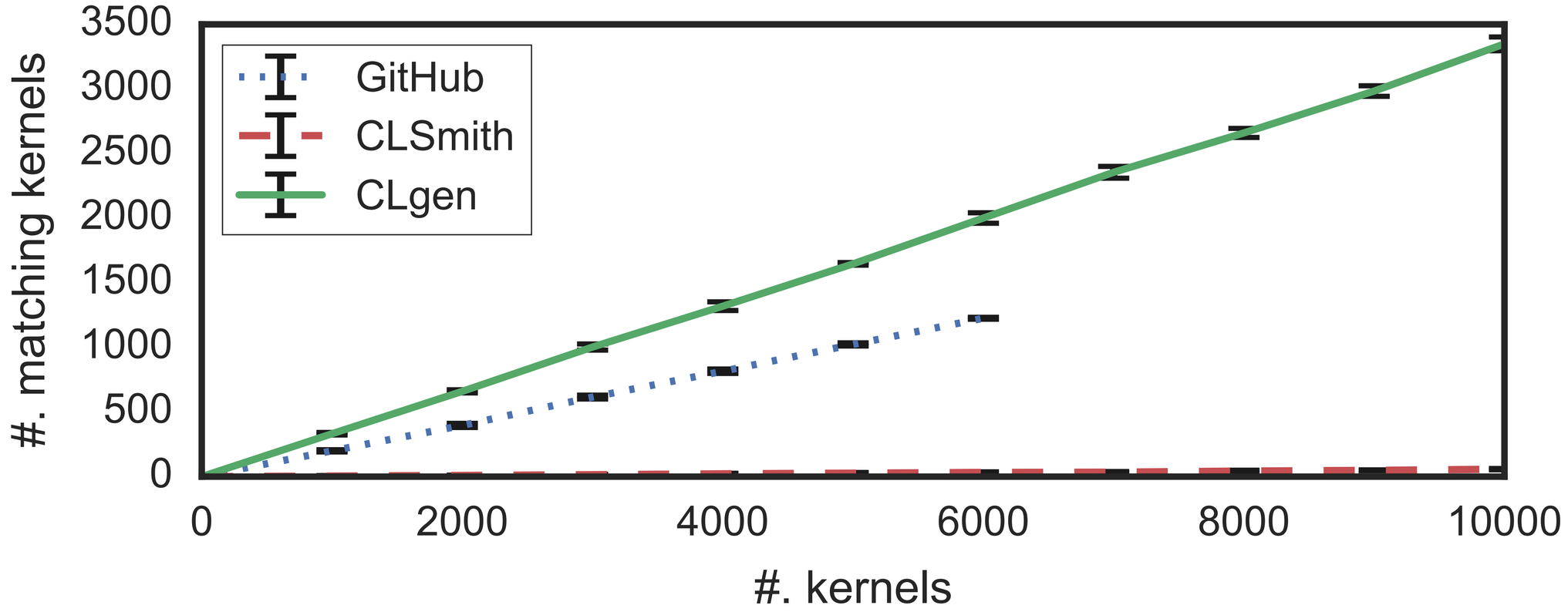
By A. Lopez "David Roberts"

This thing is fantastic. It has the same size and sp eed and sharp sound. I like the power turntable. I u se it every time you connect your computer and try ing to figure out you want to get a problem with a ti e and a duct. I ever h d problems with conve at co out. Good price for it and a coupl d a are w oftware installe o far I em happy w is product rks great for wh w anted to he wa aptop company a lo re sive n't tell that any r devic this is a t ct that works well for a reasonable price and works great for the pric e, and it works.

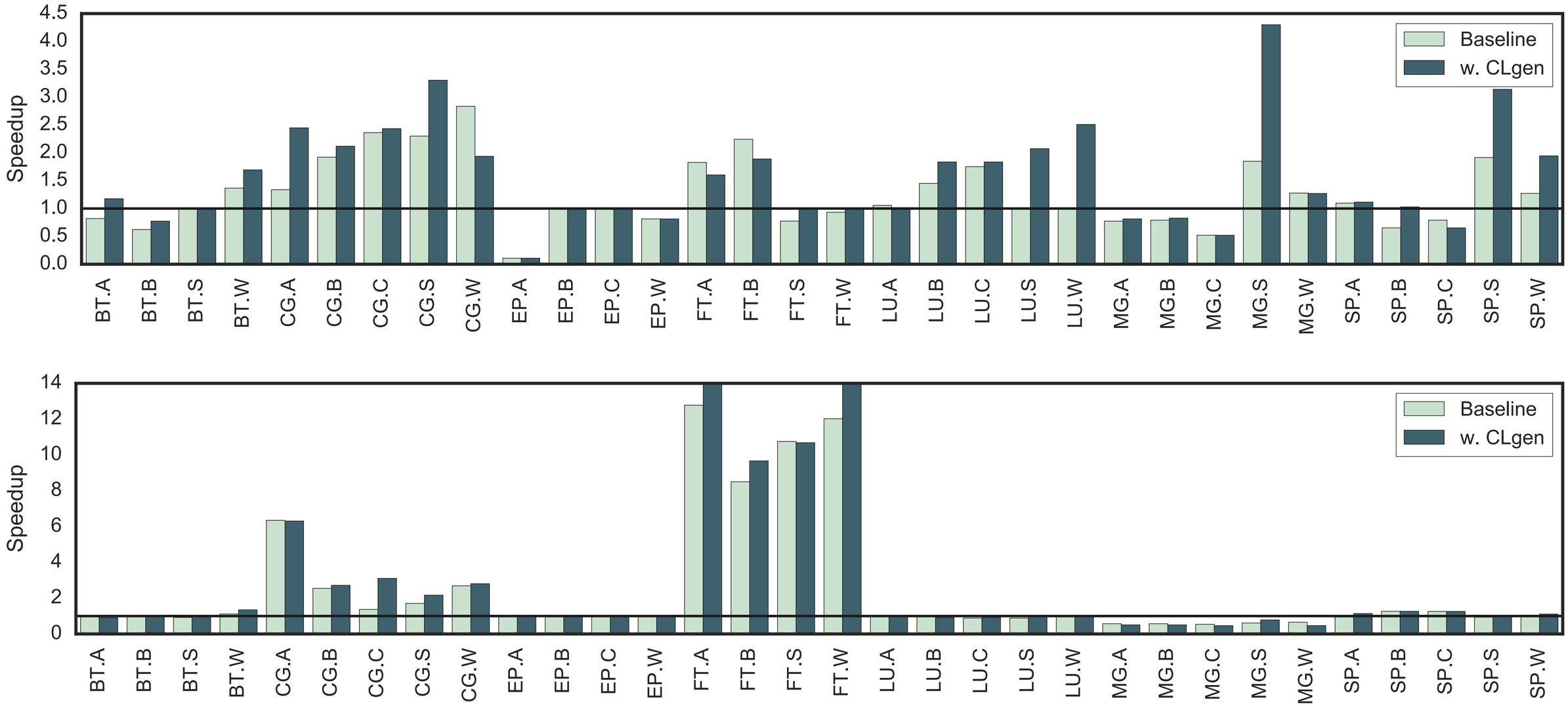
This side is the Robot

52%

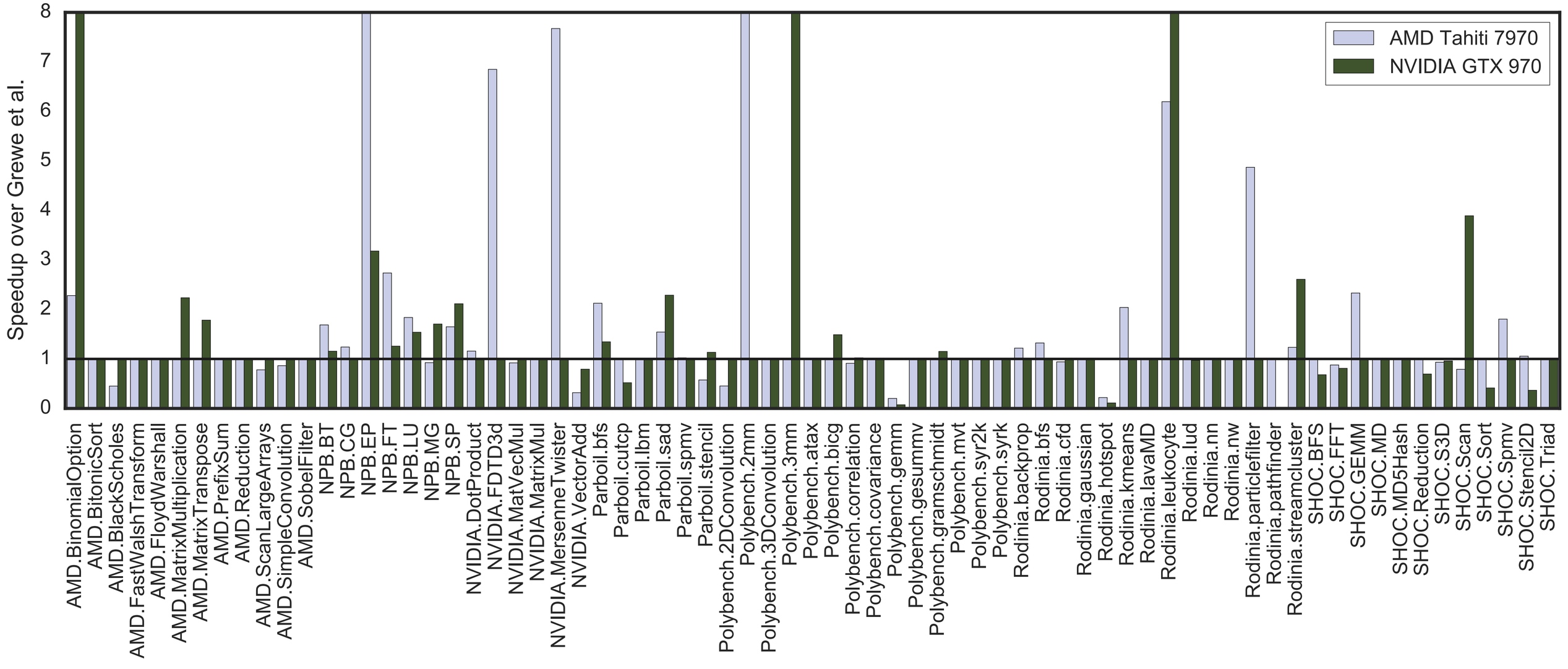




Raw Code Features		
comp	static	#. compute operations
mem	static	#. accesses to global memory
localmem	static	#. accesses to local memory
coalesced	static	#. coalesced memory accesses



7 programs, 1,000 synthetic benchmarks. 1.27x faster



71 programs, 1,000 synthetic benchmarks. 2.66x faster

Good Things

Basically* language agnostic.
35 million repos on GitHub. We're using 0.00004%.
Generates 2000 OpenCL benchmarks per machine per day.

Bad Things

No support for things declared outside of kernel scope.
Undirected almost to a fault.
AMD rage.

Thanks!

More benchmarks = Better models
No general way of creating benchmarks
I taught a Neural Network to program from GitHub
Improves state of the art by 3.38x