Using Deep Learning to Generate Human-like Code

THE UNIVERSITY of EDINBURGH
informatics

EPSRC Centre for Doctoral Training in
Pervasive Parallelism

EPSRC
Engineering and Physical Sciences
Research Council

“Synthesizing Benchmarks for Predictive Modeling”

 **Chris Cummins**

 Zheng Wang

 Pavlos Petoumenos

 Hugh Leather

machine learning for compilers

$$y = f(x)$$

Optimisations

CPU or GPU

Workgroup size

Cflags

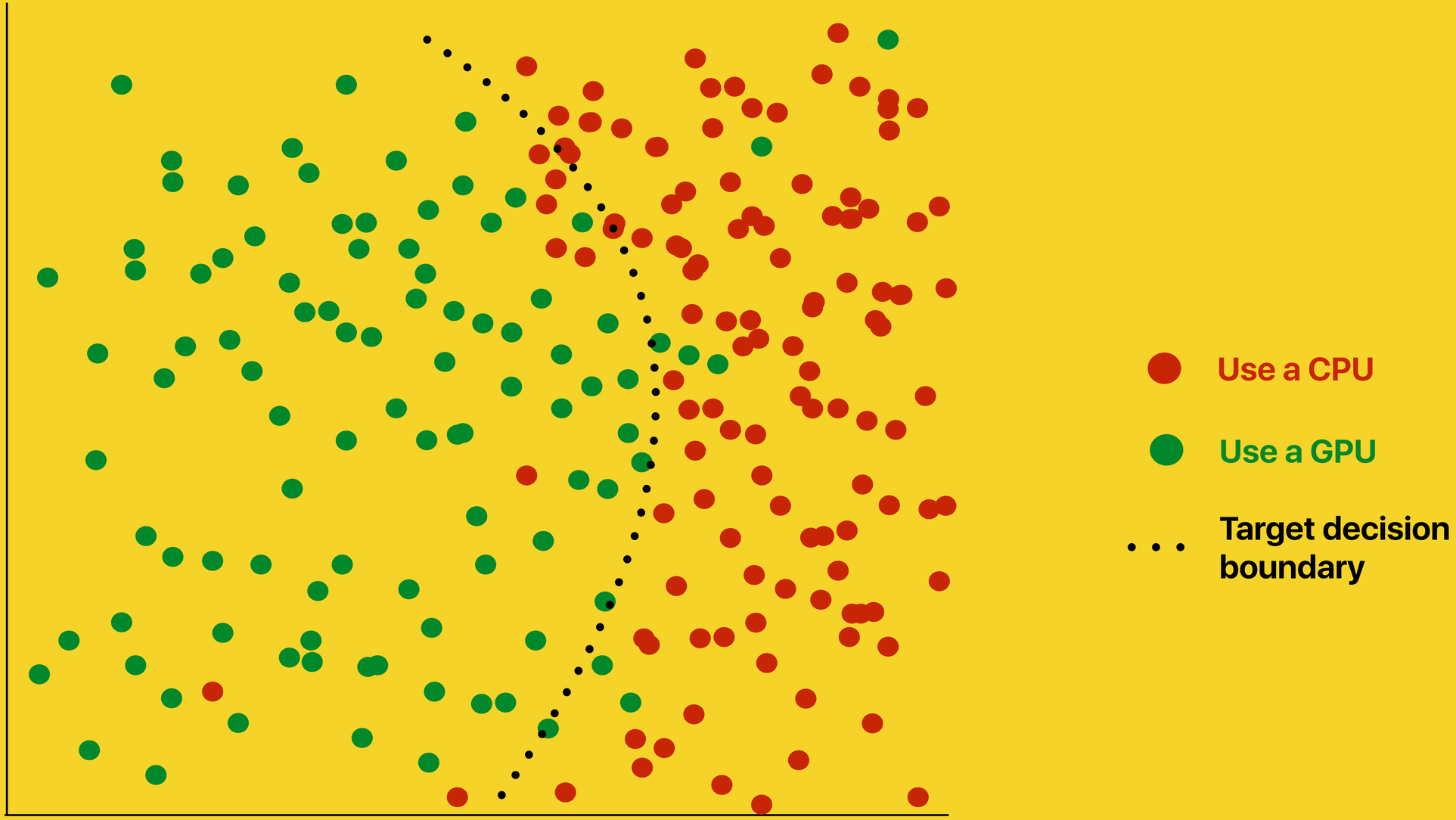
Features

instructions

Arithmetic density

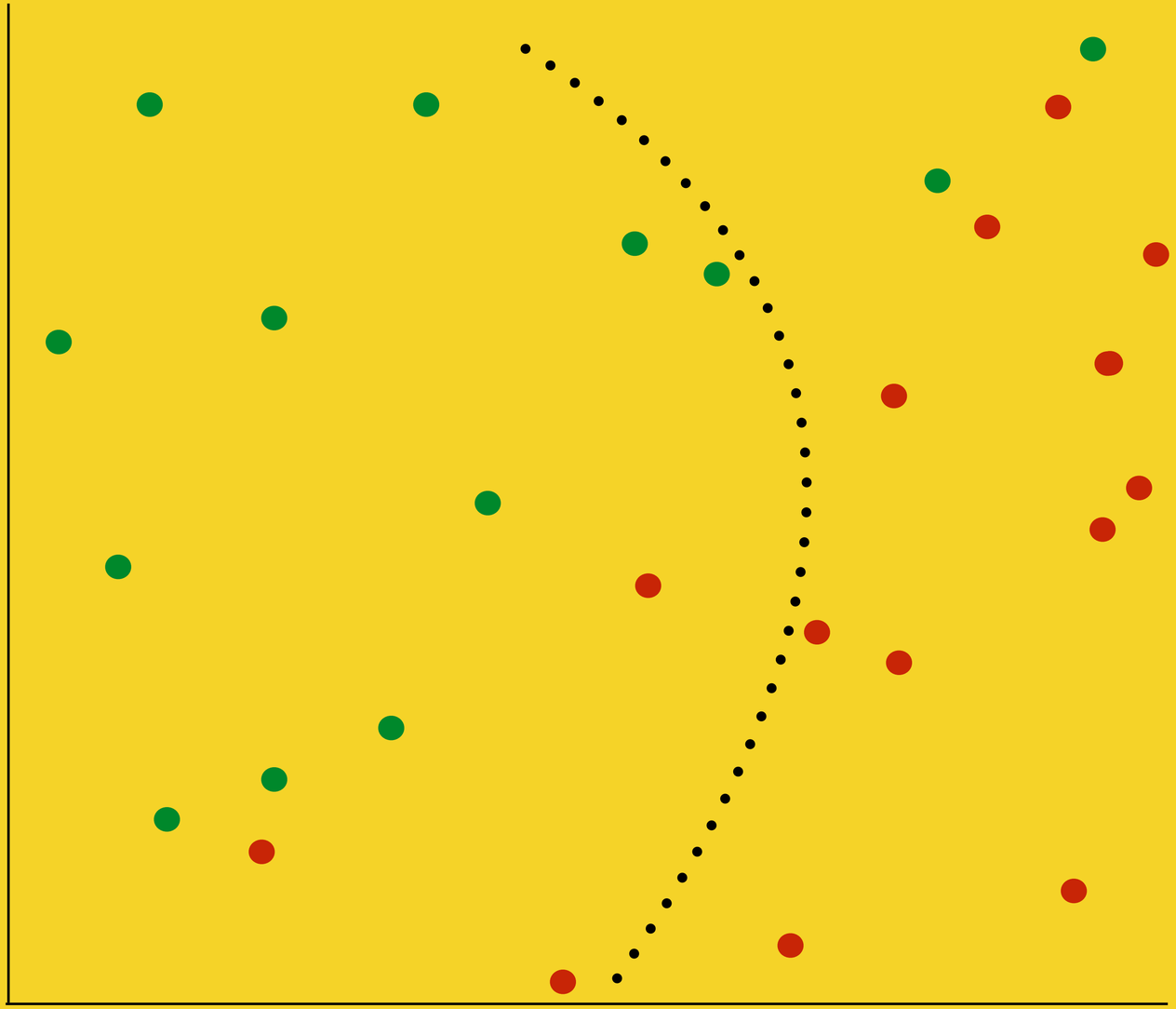
Dataset size

machine learning for compilers



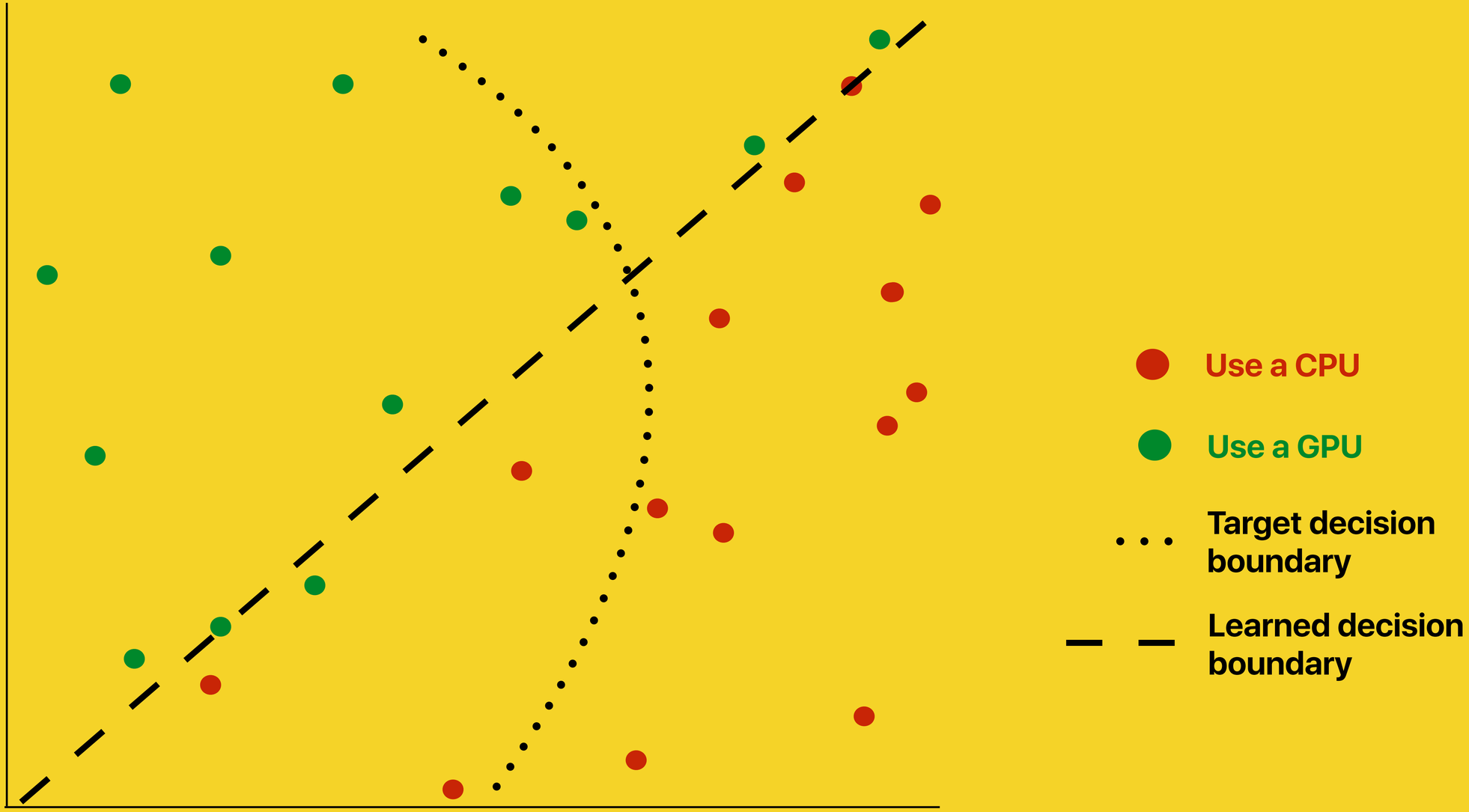
The idea.

machine learning for compilers



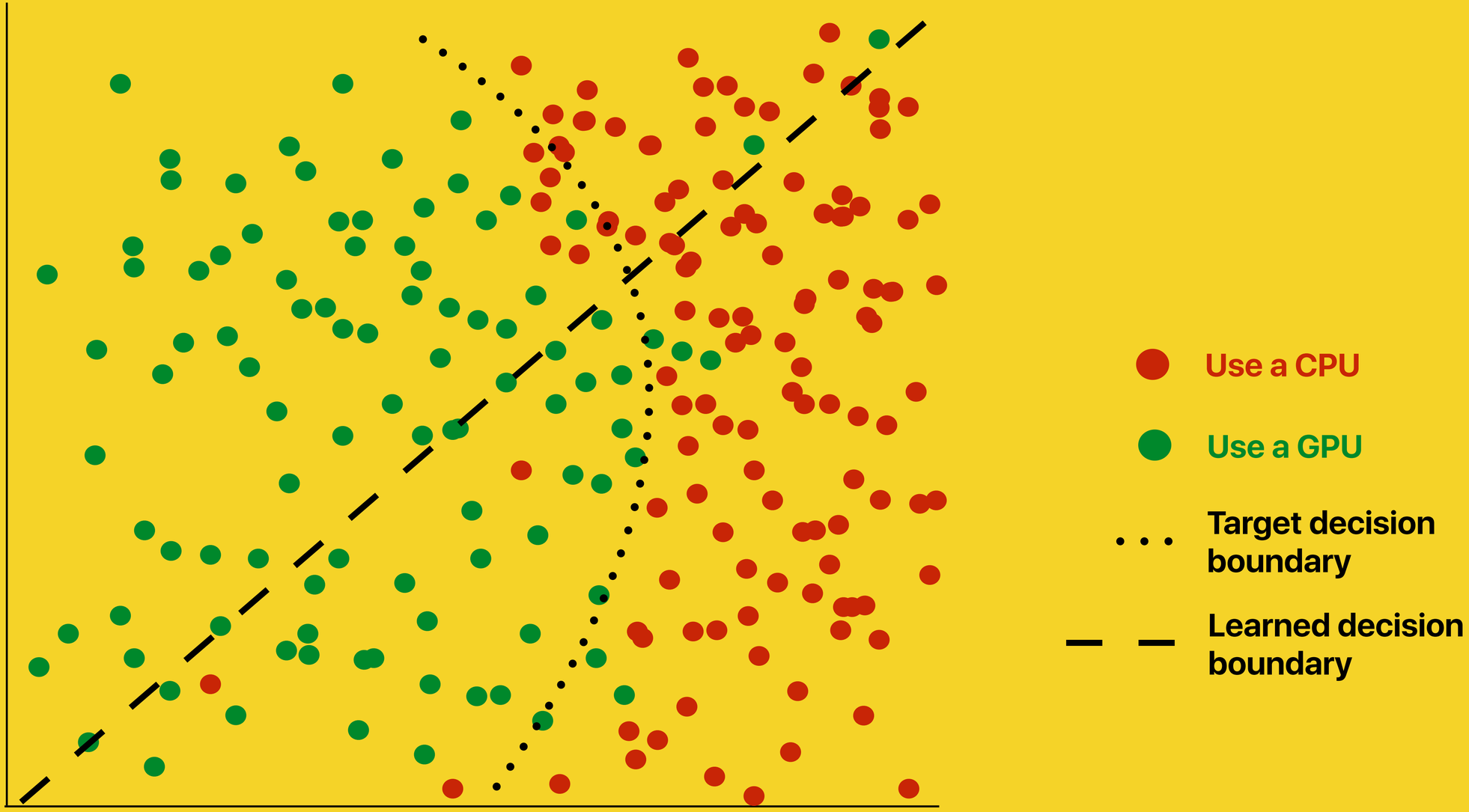
The reality.

machine learning for compilers



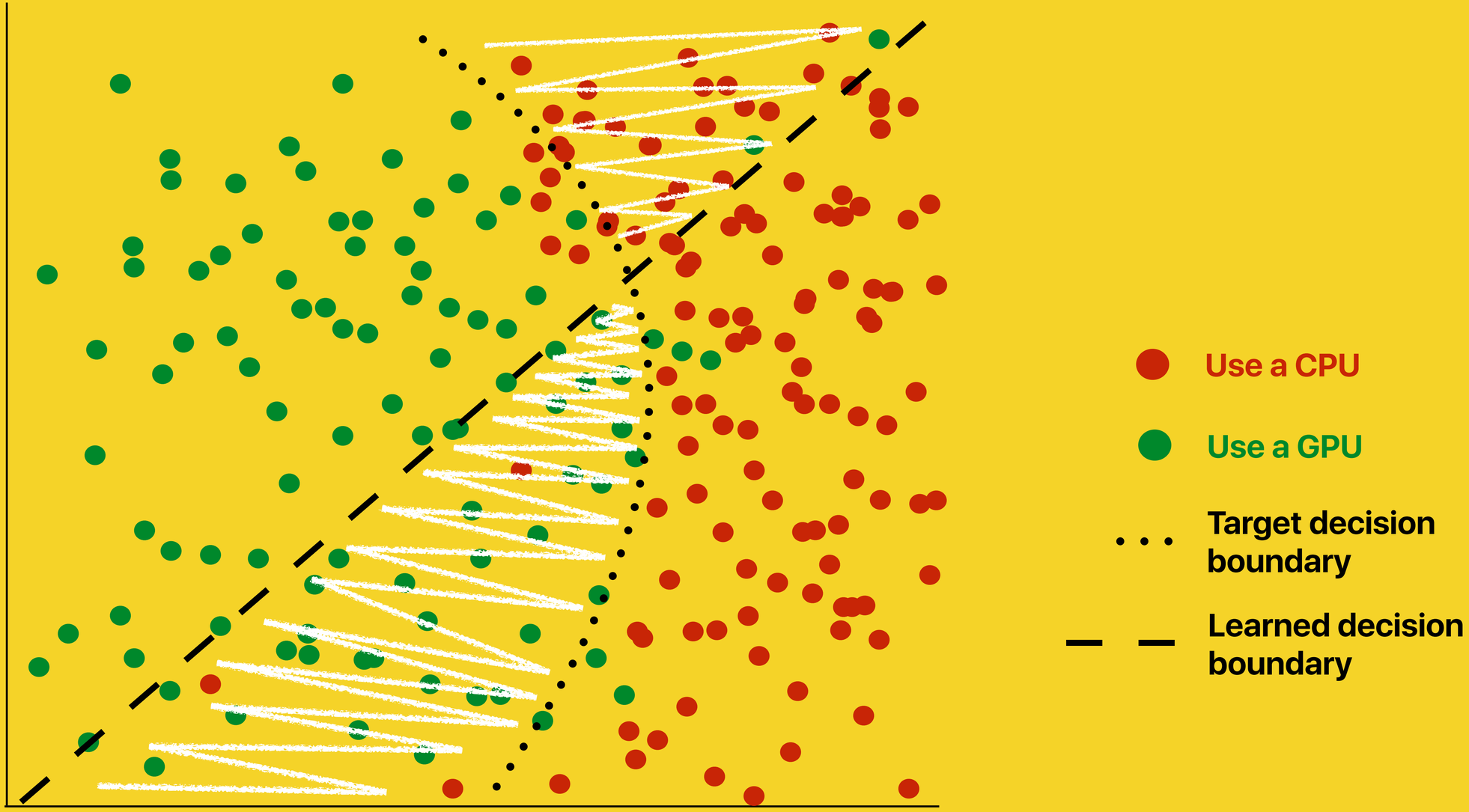
The reality.

machine learning for compilers



sparse data leads to inaccurate models!

machine learning for compilers



sparse data leads to inaccurate models!

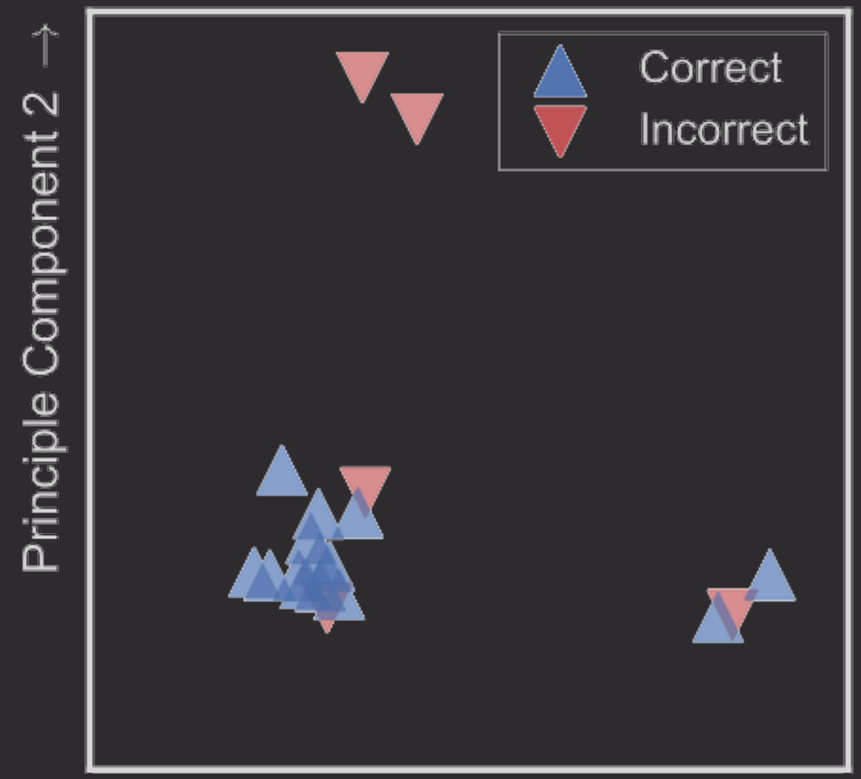
problem statement

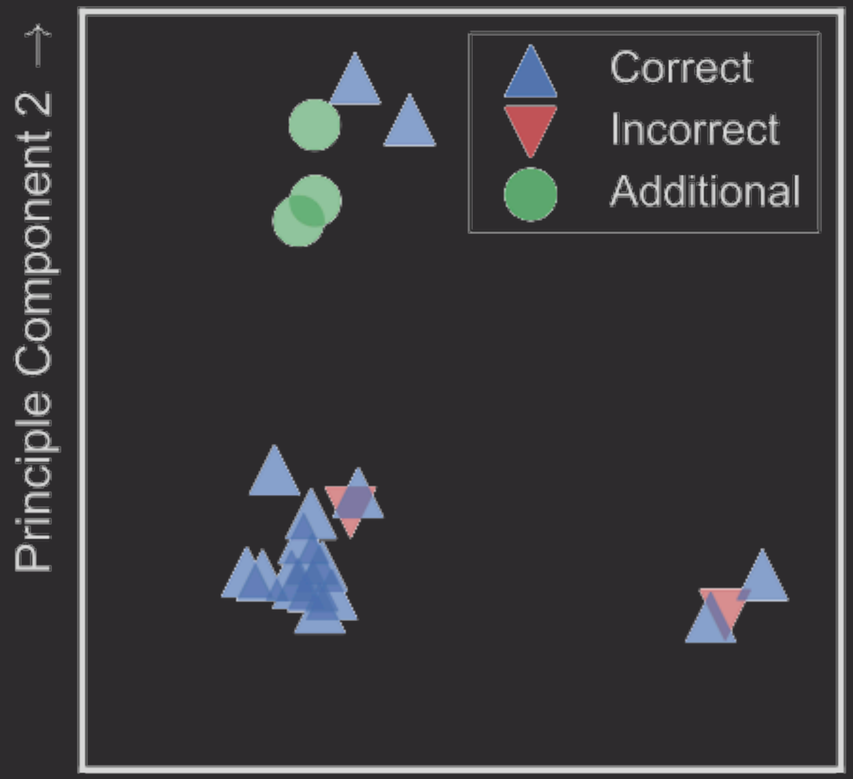
1. there aren't enough benchmarks

avg compiler paper	17	
Iris dataset	150	10×
MNIST dataset	60,000	10^3×
ImageNet dataset	10,000,000	10^6×

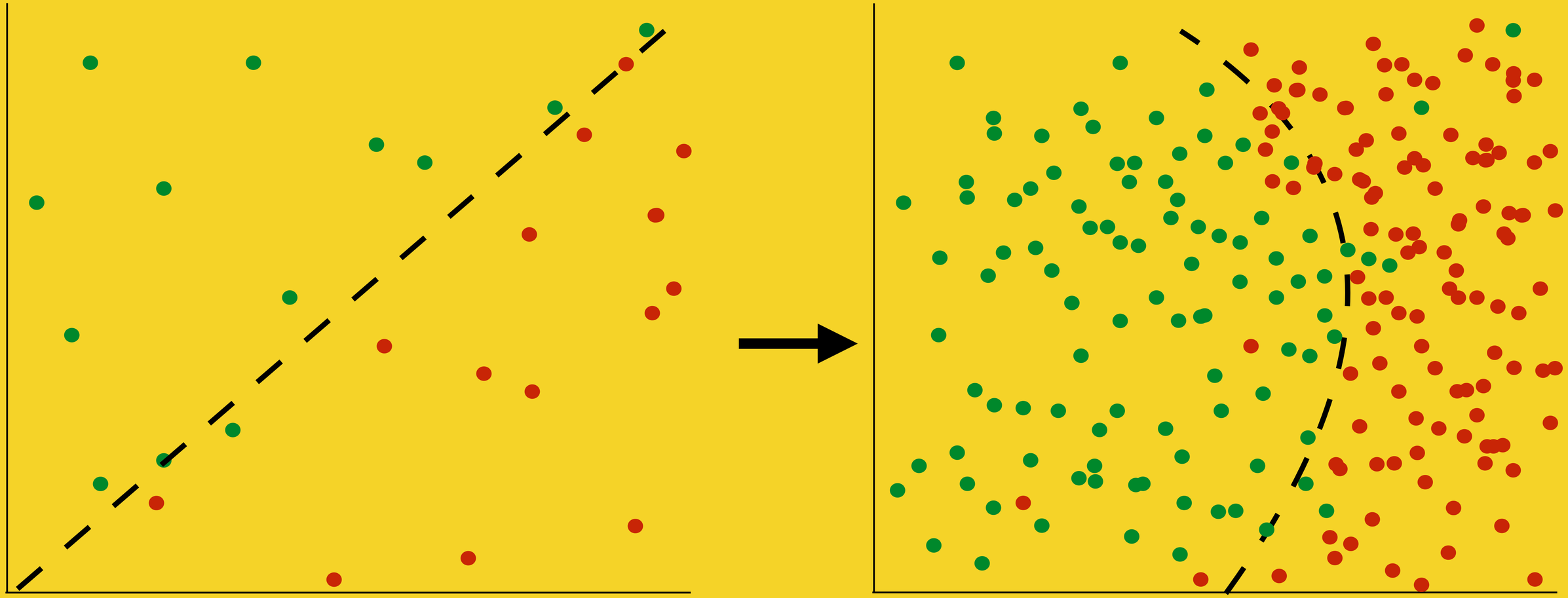
problem statement

1. there aren't enough benchmarks
2. more benchmarks = better models





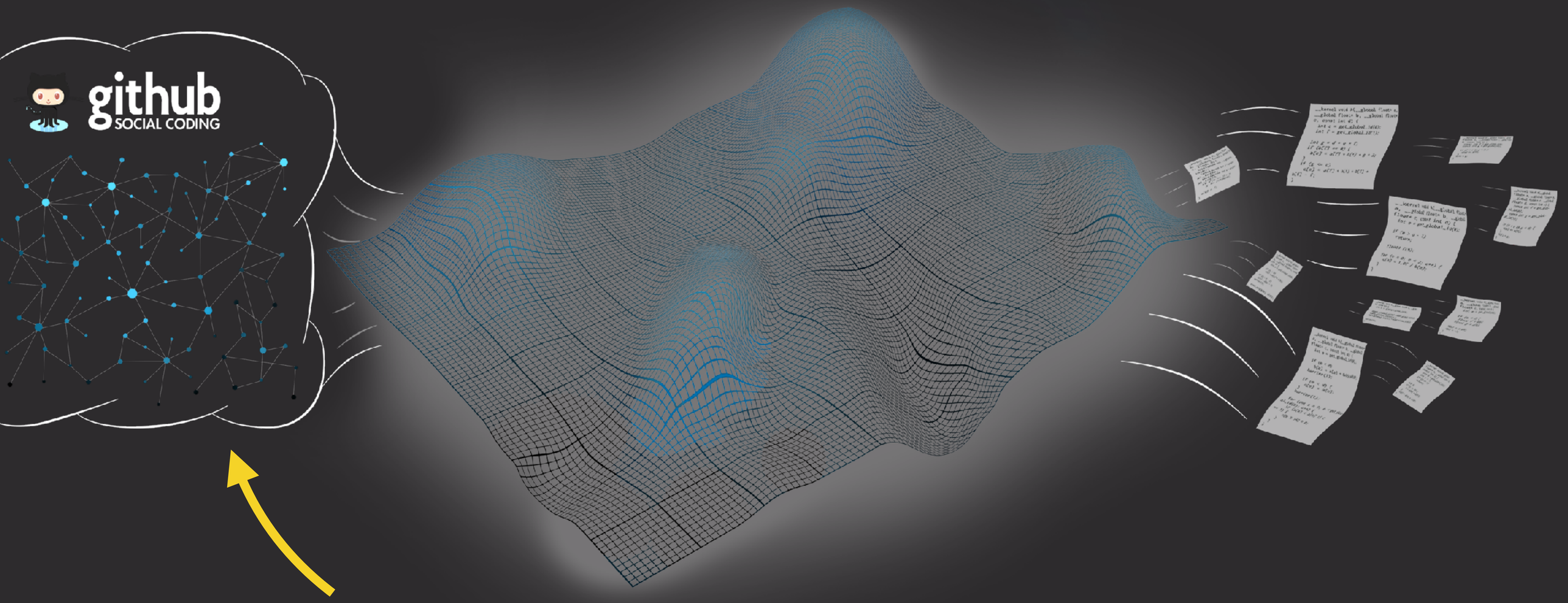
what we need



from this

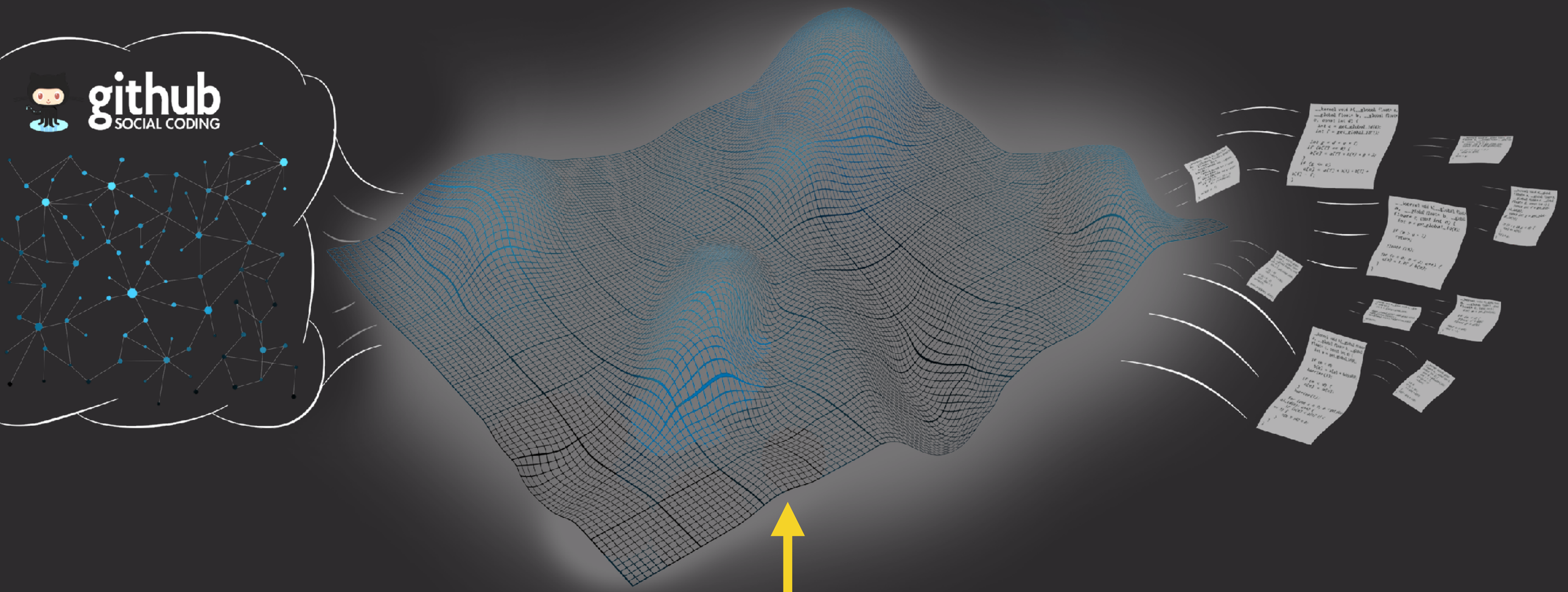
to this

our approach



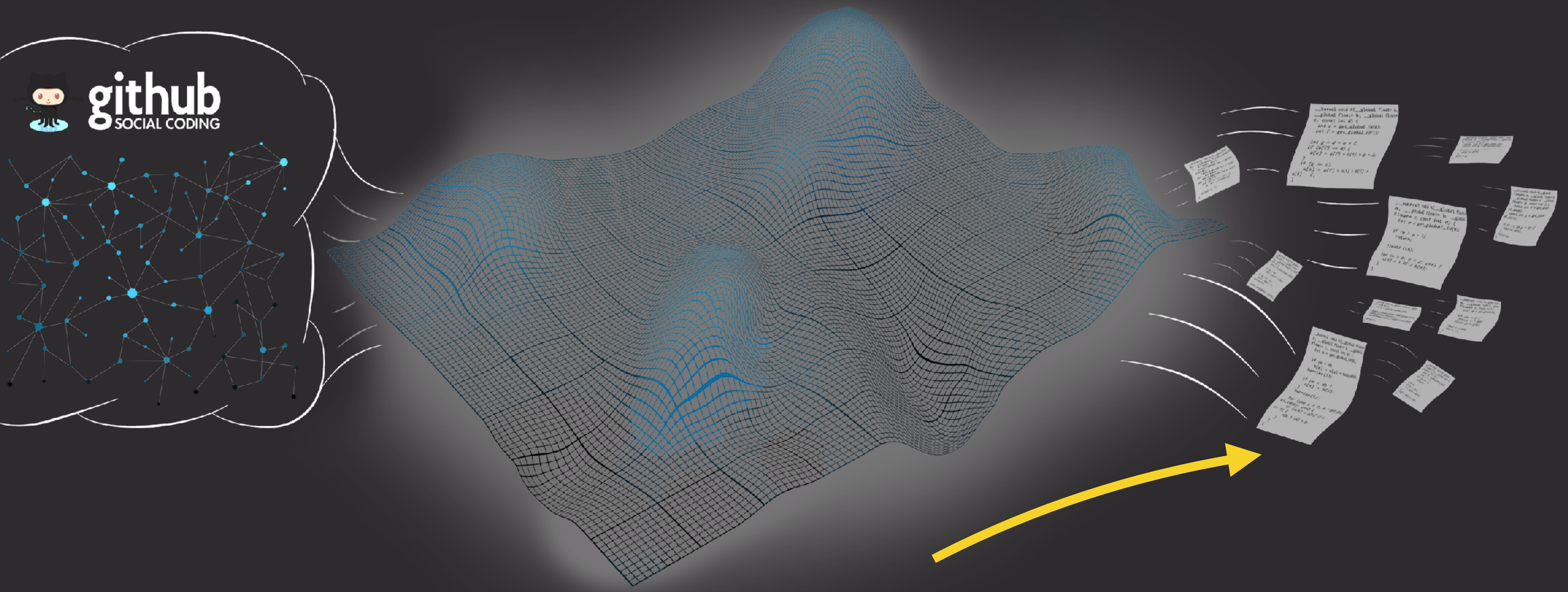
mine code from web

our approach



model source distr.

our approach



sample lang. model

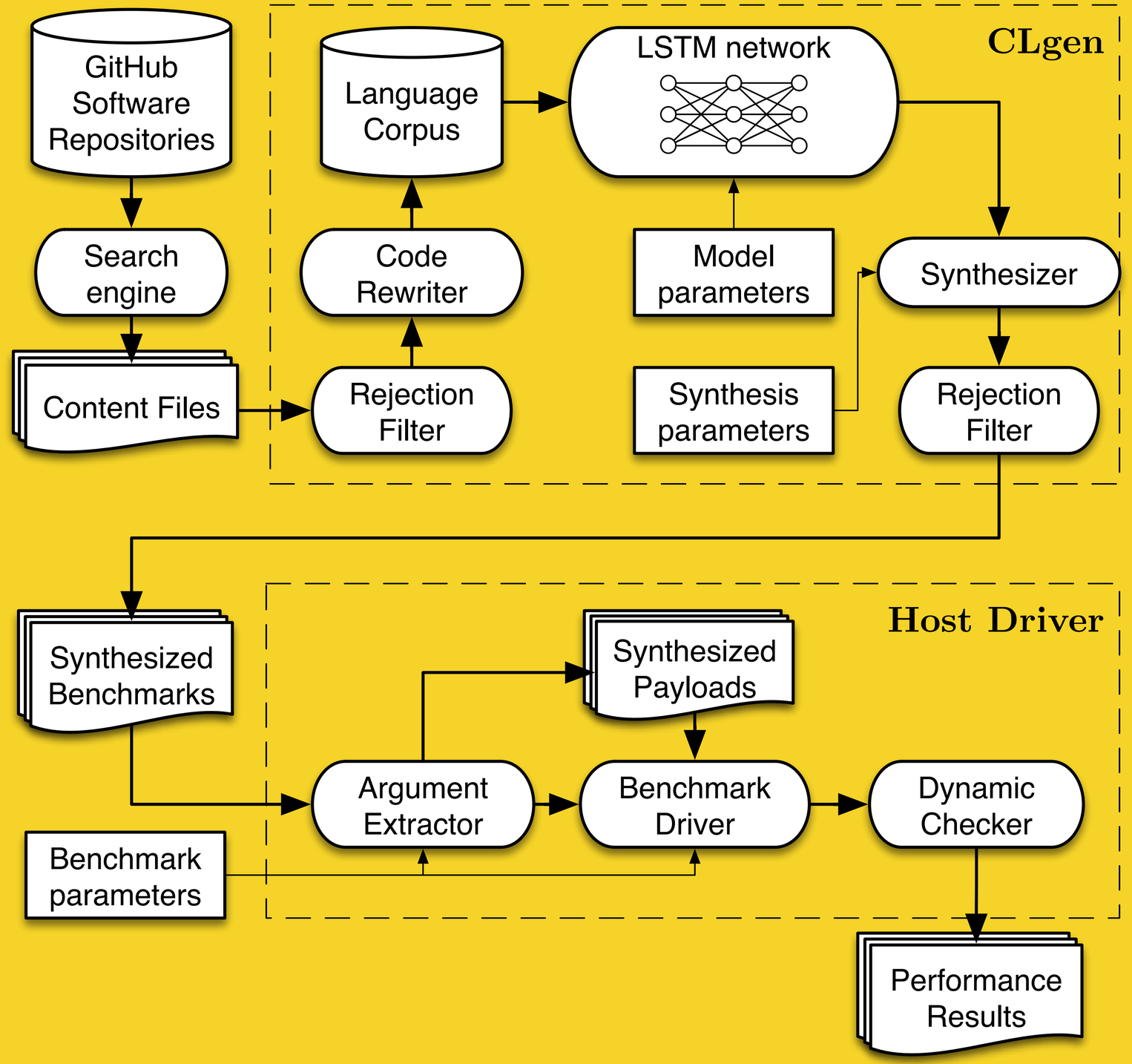
contributions

1. a deep learning approach to modeling PL semantics & usage

2. first solution for general purpose benchmark synthesis

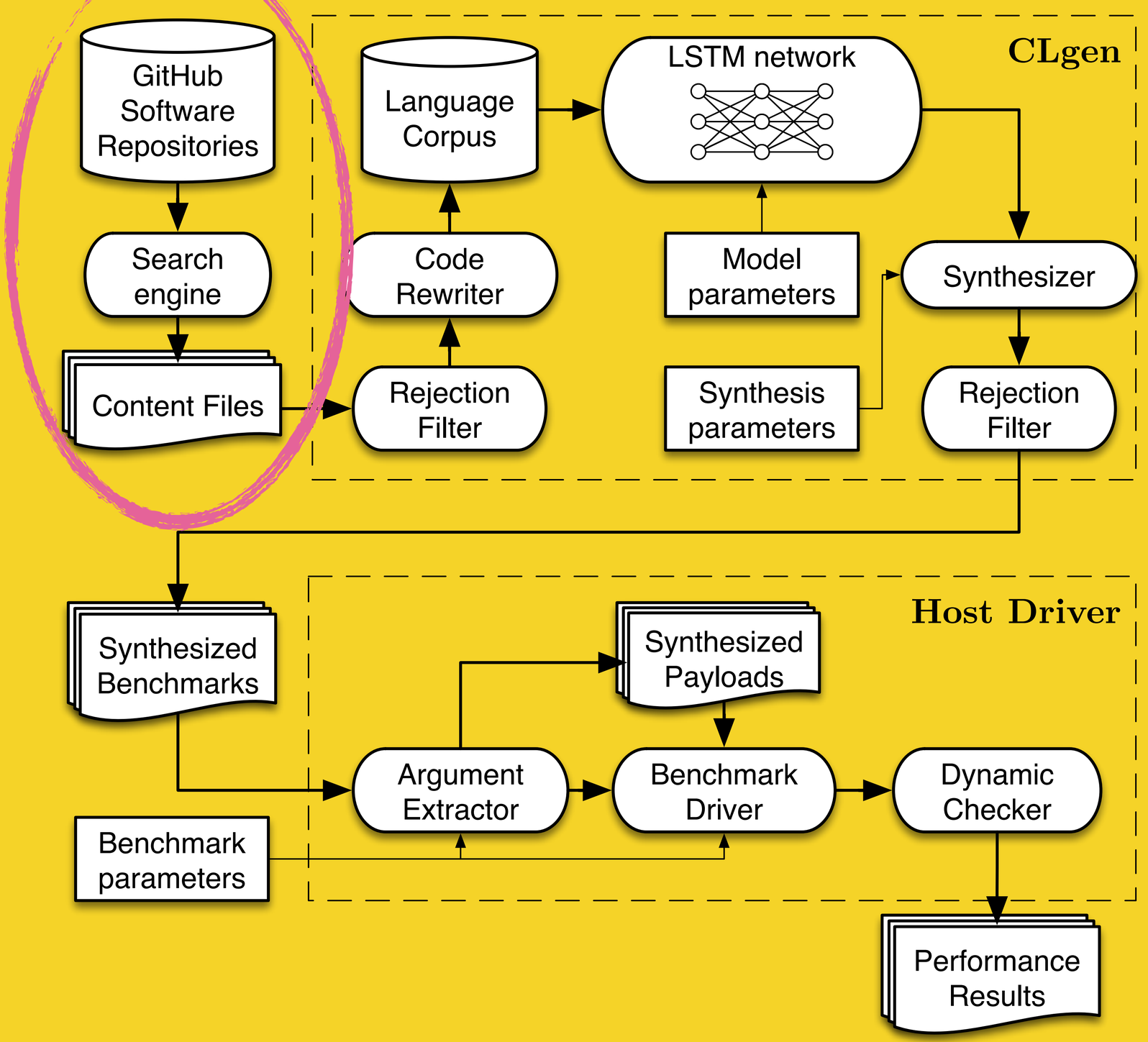
3. automatic 1.27× speedup

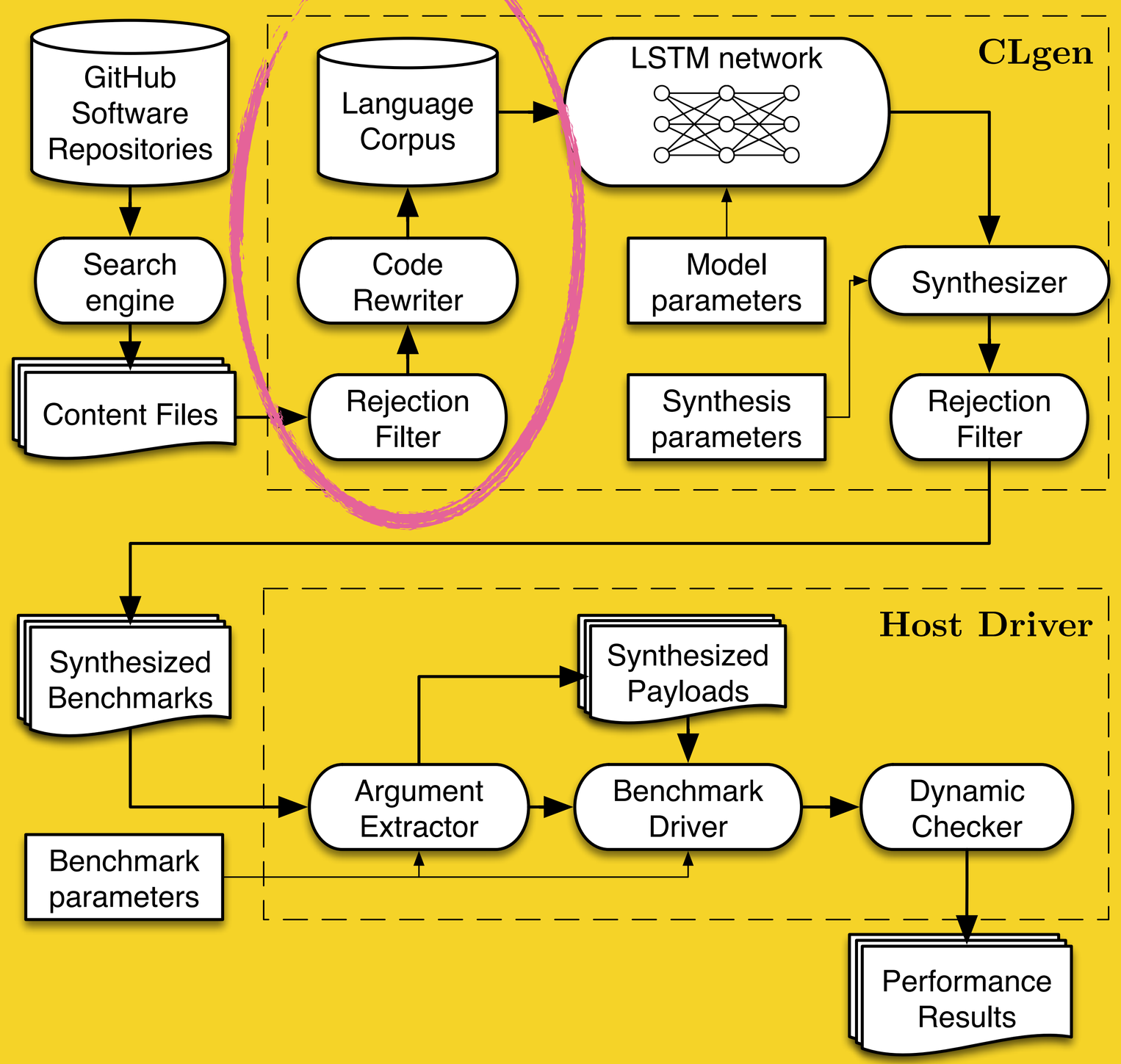
4. improved model designs for further 4.30× speedup


CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Language Corpus
Code Rewriter
LSTM network
Model parameters
Synthesis parameters
Synthesizer
Rejection Filter
Host Driver
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results

8078 files

2.8M lines




CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Language Corpus
Code Rewriter
LSTM network
Model parameters
Synthesis parameters
Synthesizer
Rejection Filter
Host Driver
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results

```
/* Copyright (C) 2014, Joe Blogs. */
#define CLAMPING
#define THRESHOLD_MAX 1.0f

float myclamp(float in) {
#ifdef CLAMPING
    return in > THRESHOLD_MAX ? THRESHOLD_MAX : in < 0.0f ? 0.0f : in;
#else
    return in;
#endif // CLAMPING
}

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


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

```
/* Copyright (C) 2014, Joe Blogs. */
#define CLAMPING
#define THRESHOLD_MAX 1.0f

float myclamp(float in) {
#ifdef CLAMPING
    return in > THRESHOLD_MAX ? THRESHOLD_MAX : in < 0.0f ? 0.0f : in;
#else
    return in;
#endif // CLAMPING
}

__kernel void findAllNodesMergedAabb(__global float* in, __global float* out,
                                     int num_elems)
{
   // Do something really flipping cool
   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) 2014, Joe Blogs. */
#define CLAMPING
#define THRESHOLD_MAX 1.0f

float myclamp(float in) {
#ifdef CLAMPING
    return in > THRESHOLD_MAX ? THRESHOLD_MAX : in < 0.0f ? 0.0f : in;
#else
    return in;
#endif // CLAMPING
}

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


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

Strip comments

```
/* Copyright (C) 2014, Joe Blogs. */
#define CLAMPING
#define THRESHOLD_MAX 1.0f

float myclamp(float in) {
#ifdef CLAMPING
    return in > THRESHOLD_MAX ? THRESHOLD_MAX : in < 0.0f ? 0.0f : in;
#else
    return in;
#endif // CLAMPING
}

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


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

Strip comments

Skip comments

```
#define CLAMPING
#define THRESHOLD_MAX 1.0f

float myclamp(float in) {
#ifdef CLAMPING
    return in > THRESHOLD_MAX ? THRESHOLD_MAX : in < 0.0f ? 0.0f : 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_MAX 1.0f

float myclamp(float in) {
#ifdef CLAMPING
    return in > THRESHOLD_MAX ? THRESHOLD_MAX : in < 0.0f ? 0.0f : 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_MAX 1.0f

float myclamp(float in) {
#ifdef CLAMPING
    return in > THRESHOLD_MAX ? THRESHOLD_MAX : in < 0.0f ? 0.0f : 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?~~

(33% discard rate)

~~Skip 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?~~
(33% discard rate)

~~Skip 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?~~
(33% discard rate)

```
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~~

~~Does it compile?~~

~~Does it contain instructions?~~

(33% discard rate)

```
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?~~

(33% discard rate)

```
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?~~

(33% discard rate)

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

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


        c[e] = A(b[e]);
    }
}
```

~~Strip comments~~

~~Preprocess~~

~~Rewrite function names~~

~~Rewrite variable names~~

~~Does it compile?~~

~~Does it contain instructions?~~

(33% discard rate)

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

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

        c[e] = A(b[e]);
    }
}
```

~~Strip comments~~

~~Preprocess~~

~~Rewrite function names~~

~~Rewrite variable names~~

Enforce code style

~~Does it compile?~~

~~Does it contain instructions?~~

(33% discard rate)

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

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

~~Strip comments~~

~~Preprocess~~

~~Rewrite function names~~

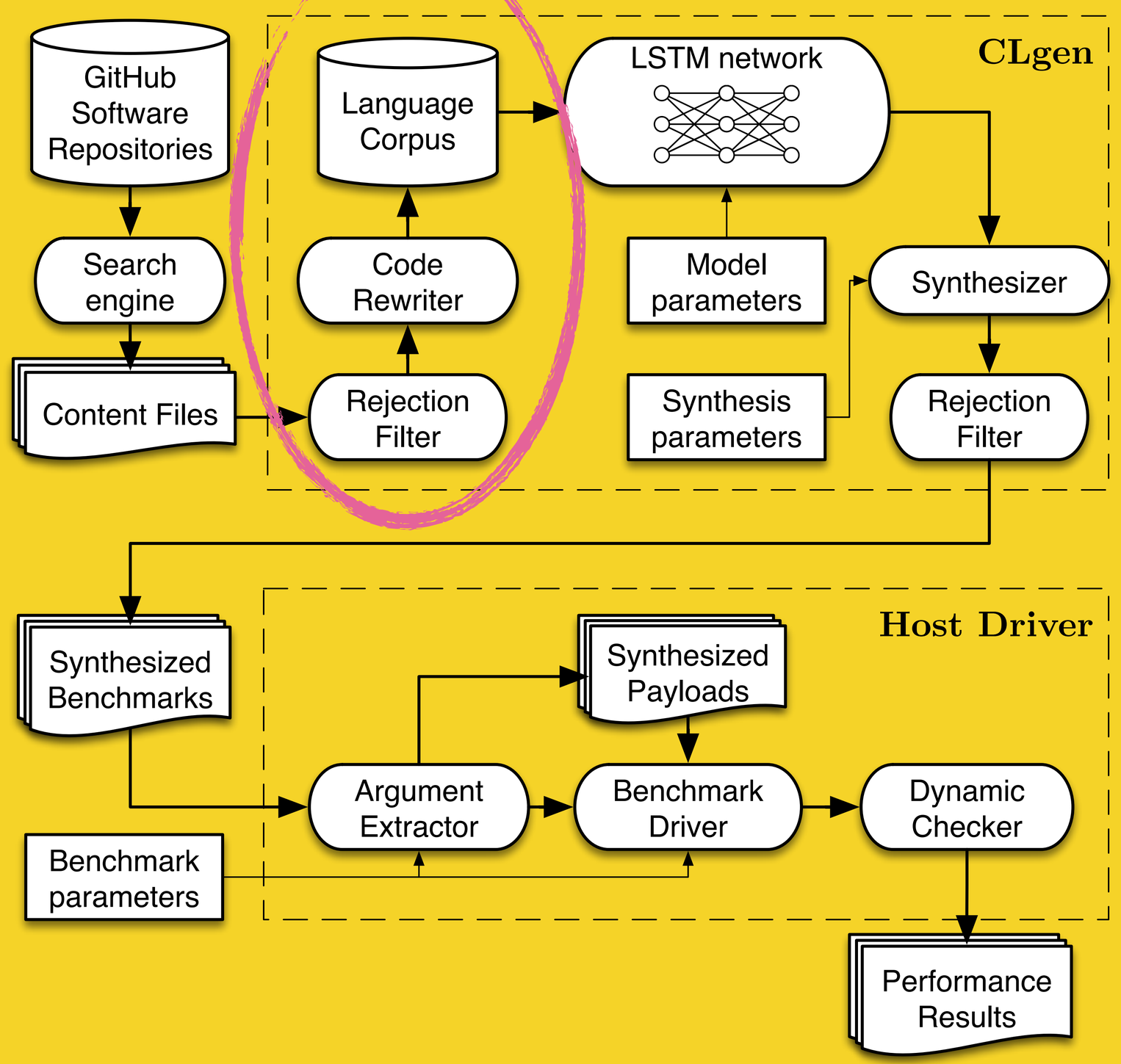
~~Rewrite variable names~~

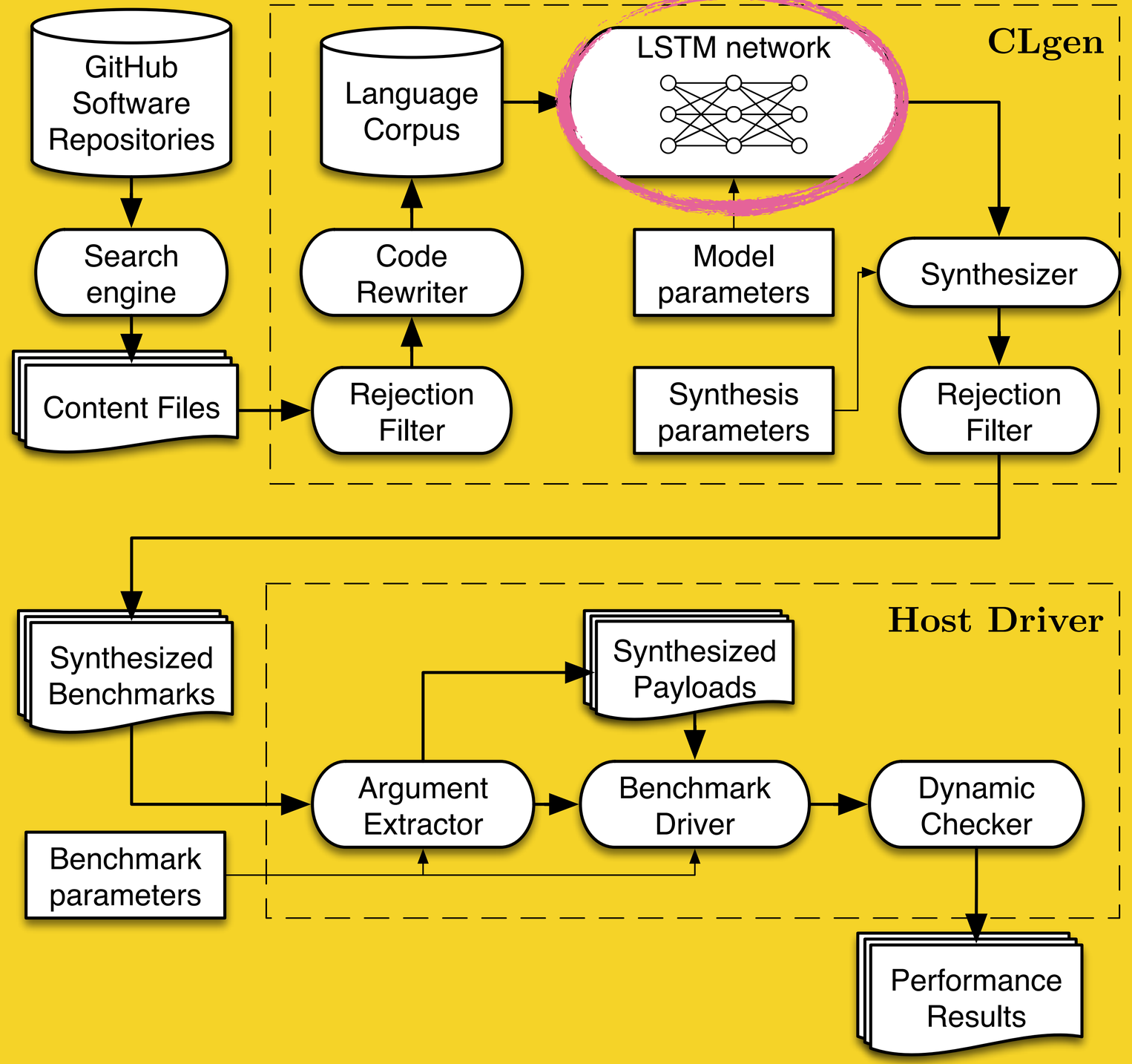
~~Enforce code style~~

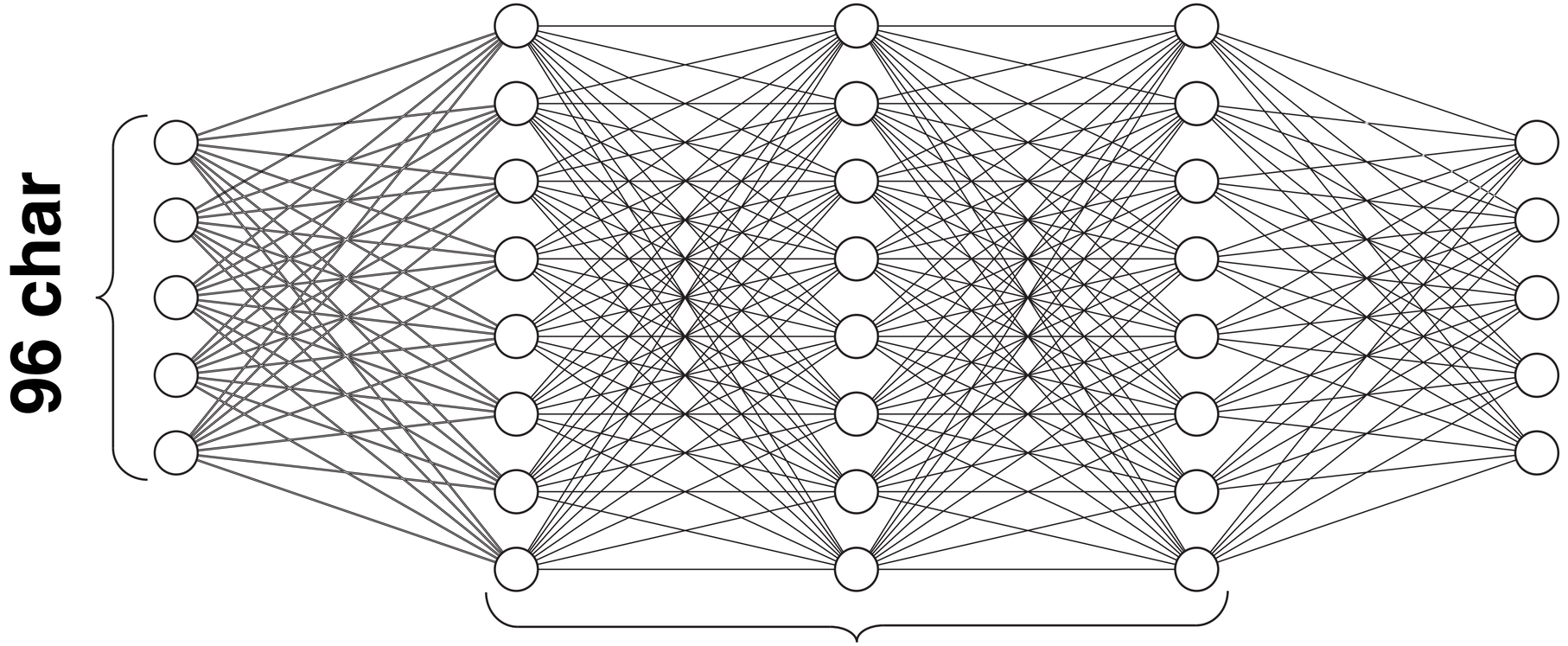
~~Does it compile?~~

~~Does it contain instructions?~~

(33% discard rate)


CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Language Corpus
Code Rewriter
LSTM network
Model parameters
Synthesis parameters
Synthesizer
Rejection Filter
Host Driver
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results


CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Language Corpus
Code Rewriter
LSTM network
Model parameters
Synthesis parameters
Synthesizer
Rejection Filter
Host Driver
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results

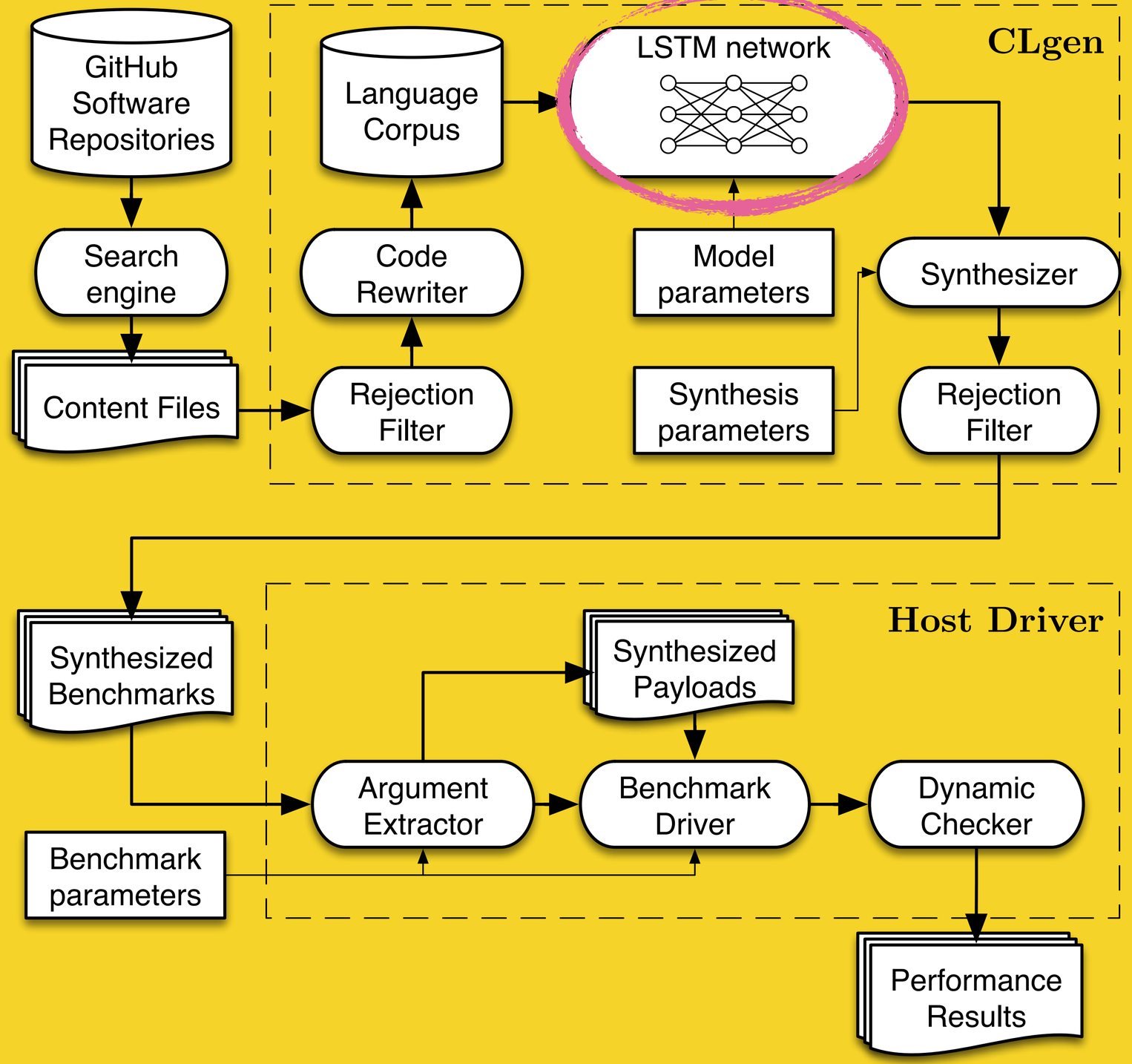


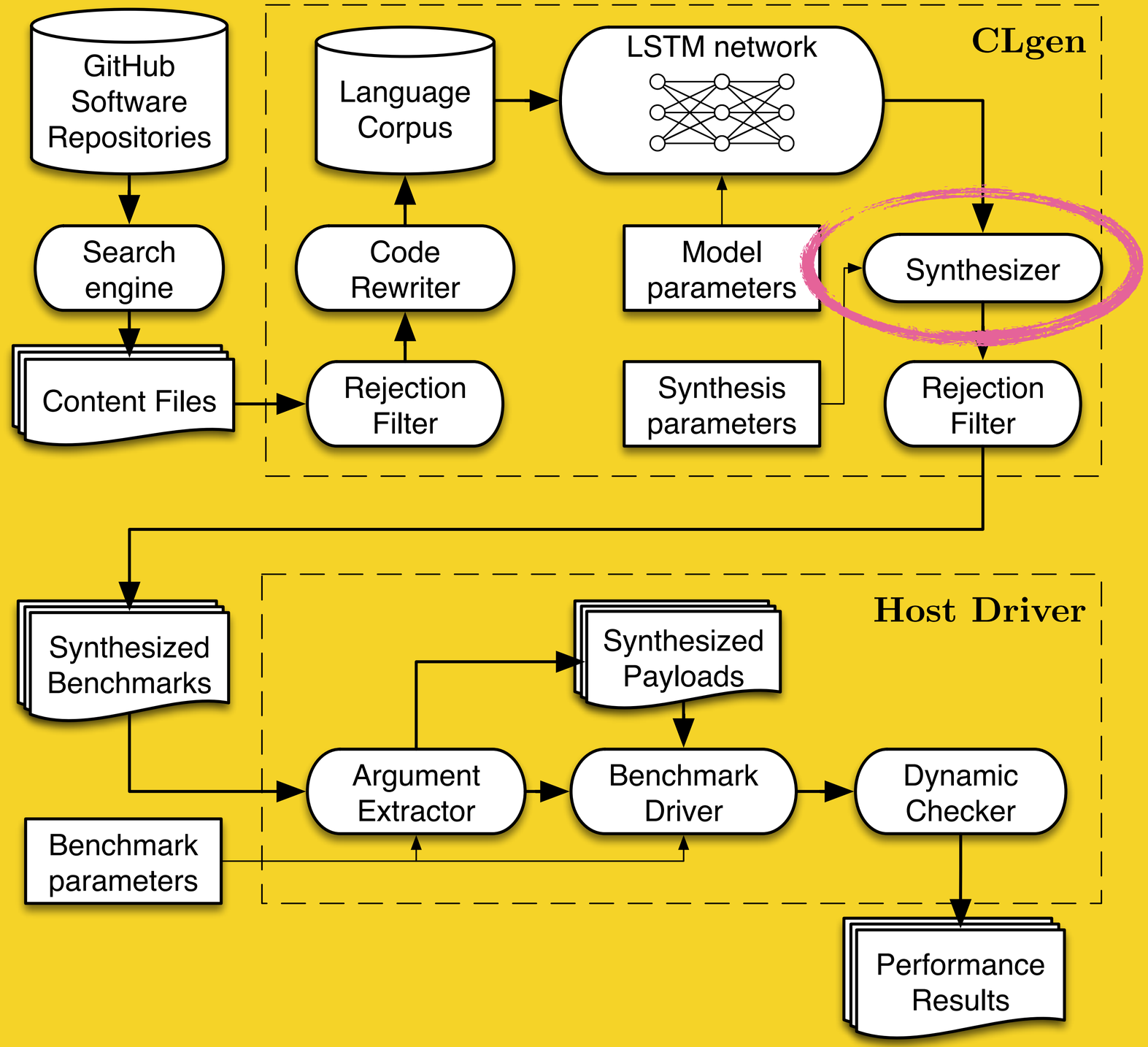
character level language model

$$y = f(x)$$

next char

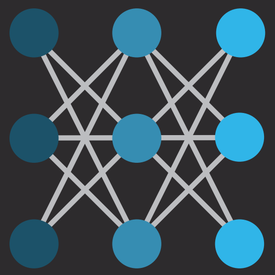
char array


CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Language Corpus
Code Rewriter
LSTM network
Model parameters
Synthesis parameters
Synthesizer
Rejection Filter
Host Driver
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results


CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Language Corpus
Code Rewriter
LSTM network
Model parameters
Synthesis parameters
Synthesizer
Rejection Filter
Host Driver
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results

kernel synthesis

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

return S
```



CLgen

Deep Learning Program Generator.


```
$ clgen model.json sampler.json
```

```
{
  “corpus": {
    “path": “~/kernels”
  },
  "model_type": "lstm",
  "rnn_size":   2048,
  "num_layers": 3,
  "max_epochs": 50
}
```

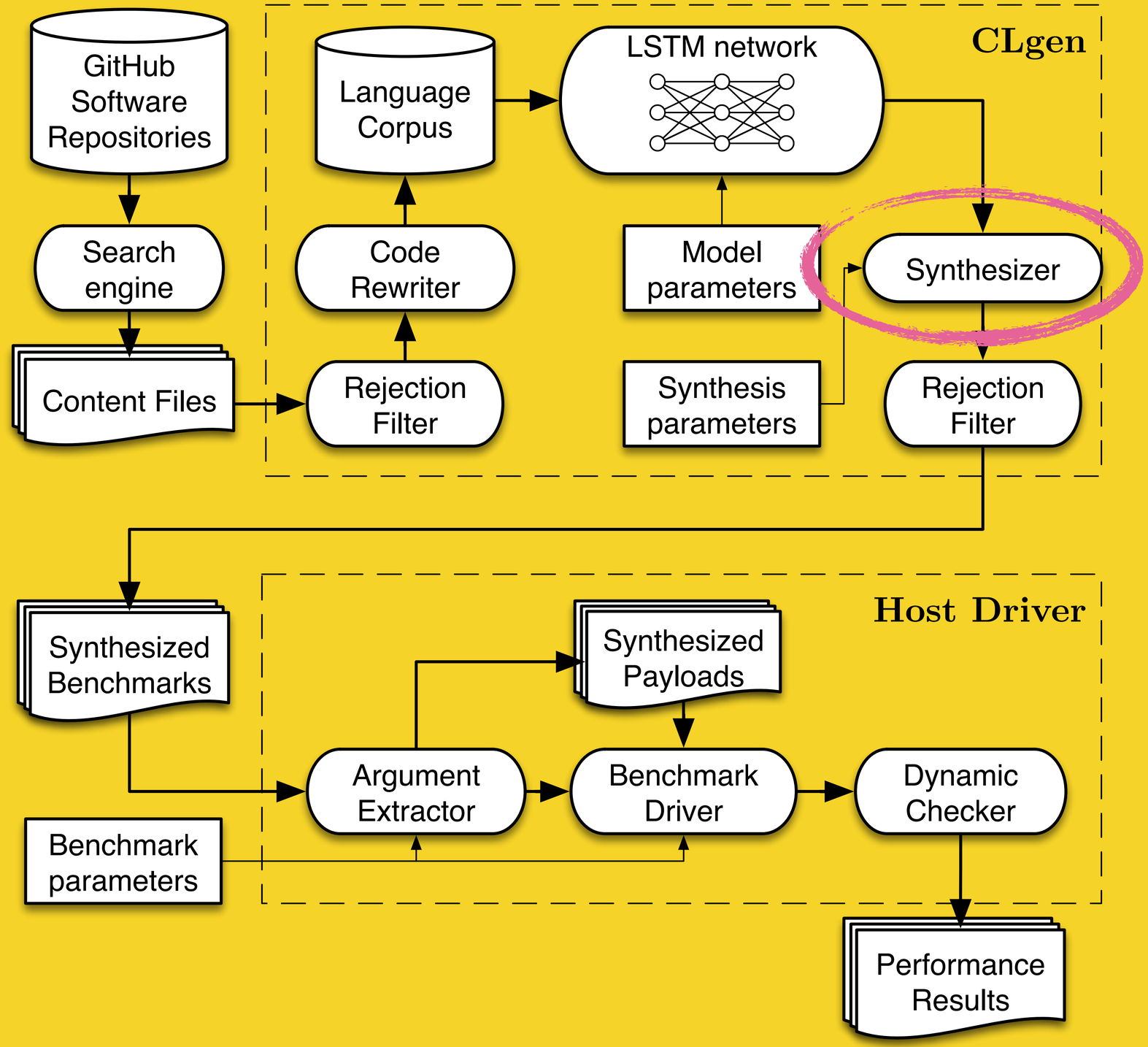
```
{
  "kernels": {
    "args": [
      "__global float*",
      "__global float*",
      "const int"
    ]
  },
  "sampler": {
    "max_kernels": 1000
  }
}
```

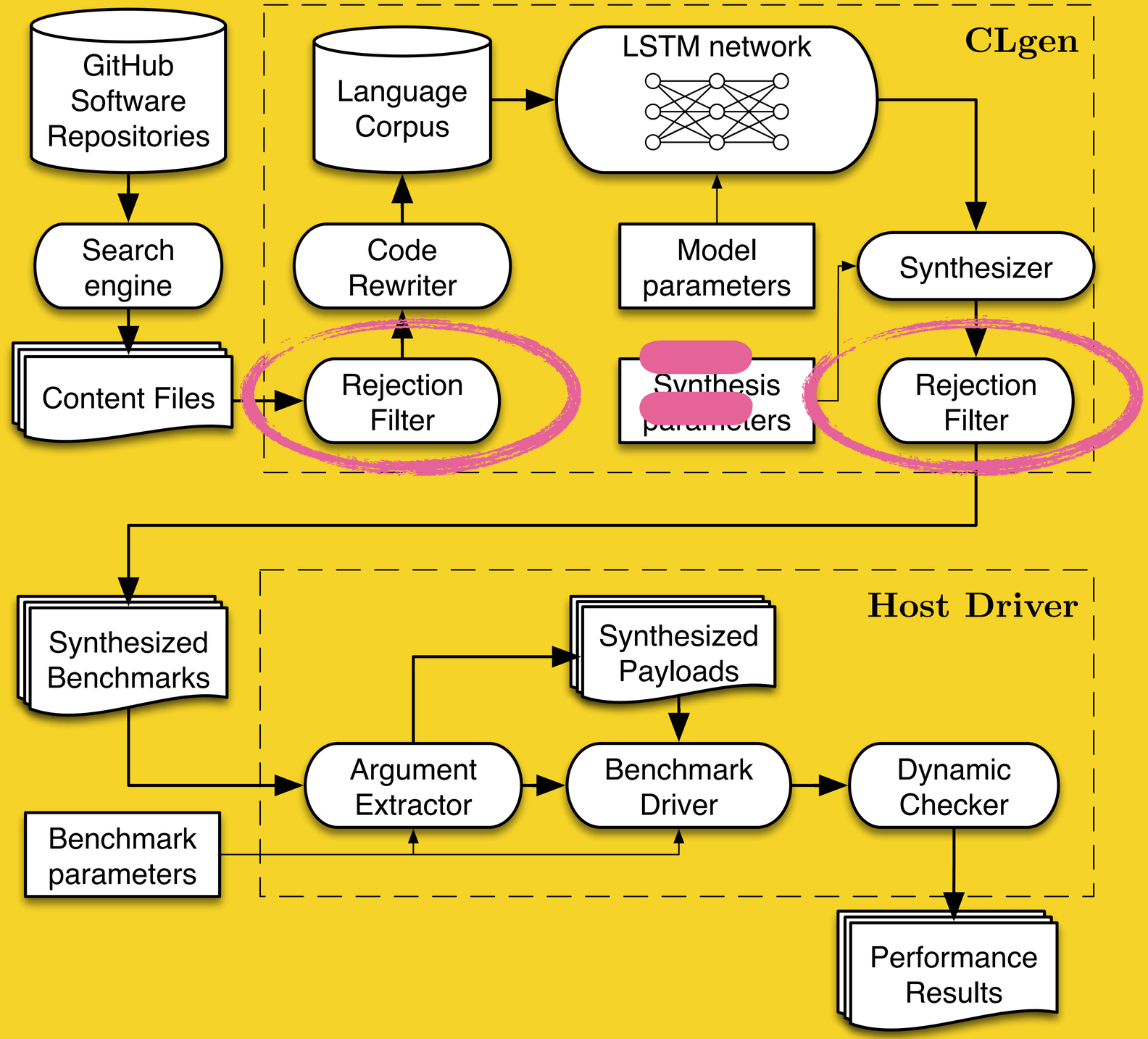
```
$ clgen model.json sampler.json 2>/dev/null
```

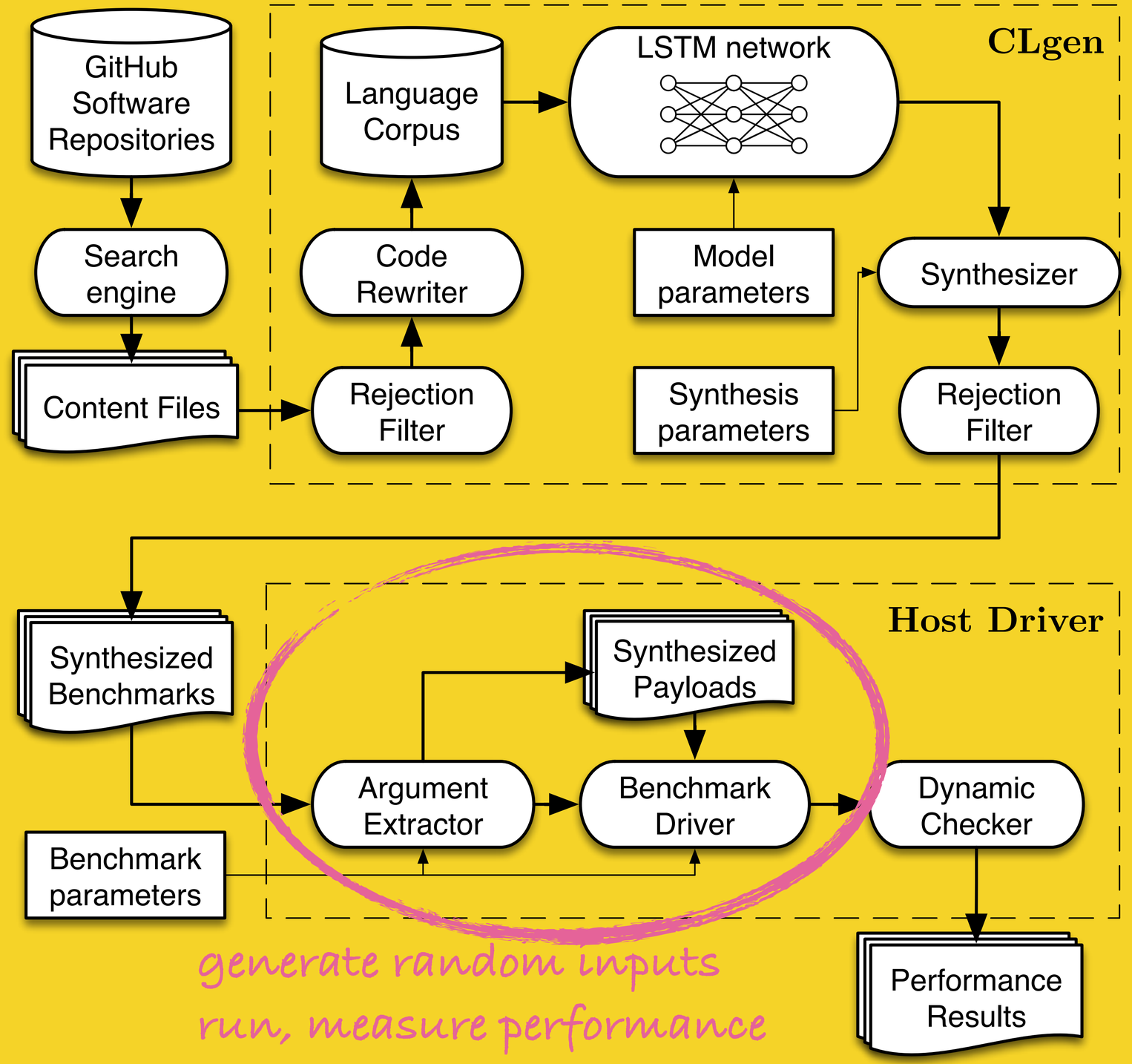
```
__kernel void A(__global float* a,
                __global float* b,
                __global float* c,
                const int d) {
  int e = get_global_id(0);
  float f = 0.0;
  for (int g = 0; g < d; g++) {
    c[g] = 0.0f;
  }
  barrier(1);

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

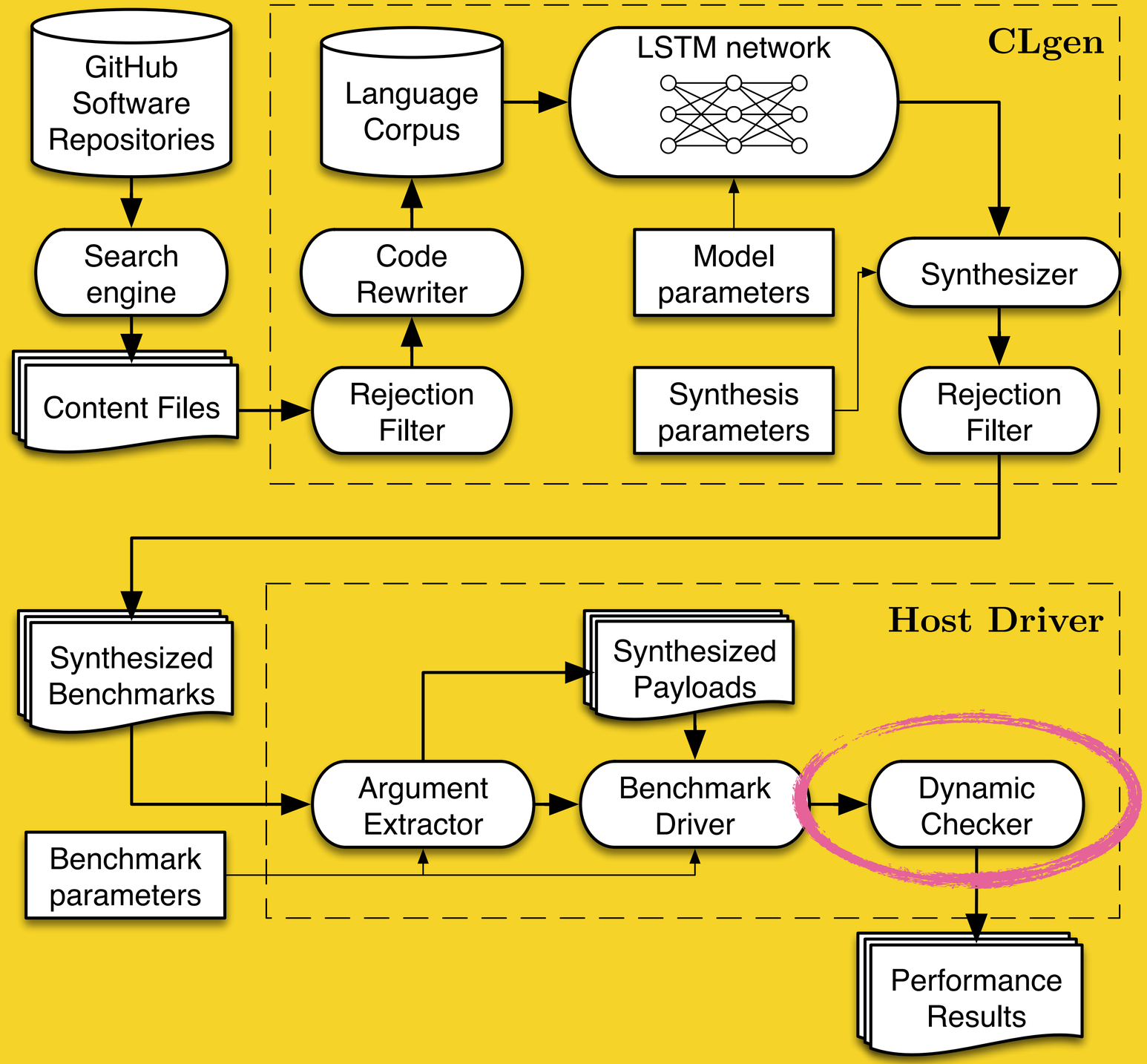
```
__kernel void A(__global float* a,
                __global float* b,
                __global float* c,
                const int d) {
  int e = get_global_id(0);
  if (e >= d) {
    return;
  }
  c[e] = a[e] + b[e] + 2 * a[e] + b[e] + 4;
}
```

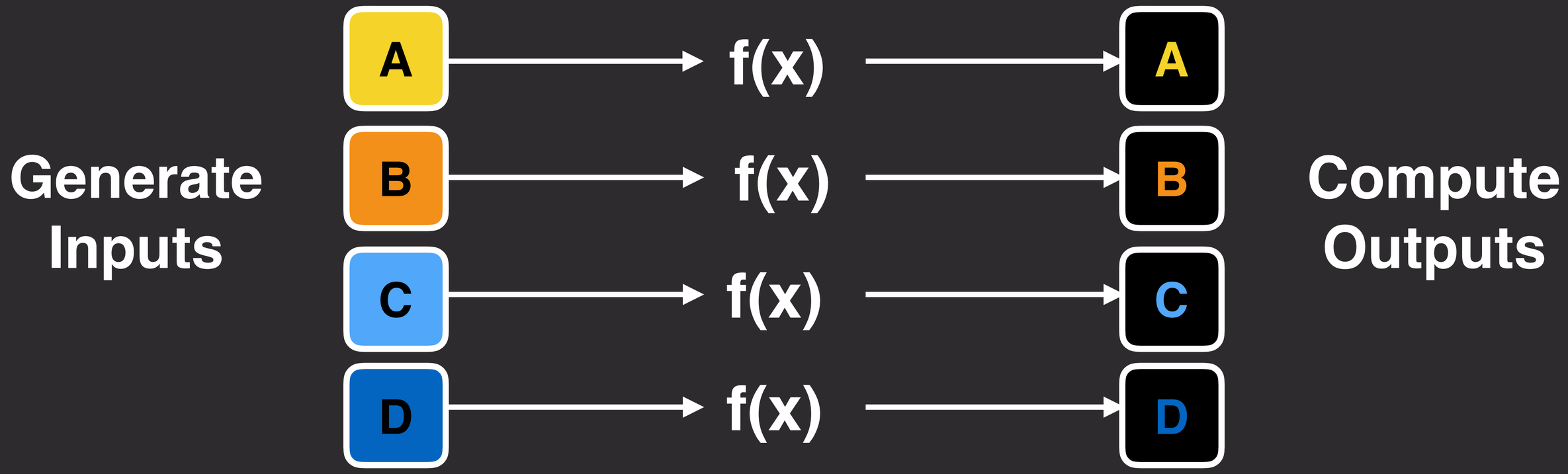
```
__kernel void A(__global float* a,
                __global float* b,
                __global float* c,
                const int d) {
  unsigned int e = get_global_id(0);
  float16 f = (float16)(0.0);
  for (unsigned int g = 0; g < d; g++) {
    float16 h = a[g];
    f.s0 += h.s0;
    f.s1 += h.s1;
    /* 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
Language Corpus
Code Rewriter
LSTM network
Model parameters
Synthesis parameters
Synthesizer
Rejection Filter
Host Driver
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results


CLgen
GitHub Software Repositories
Language Corpus
LSTM network
Search engine
Code Rewriter
Model parameters
Synthesizer
Content Files
Rejection Filter
Synthesis parameters
Rejection Filter
Host Driver
Synthesized Benchmarks
Synthesized Payloads
Argument Extractor
Benchmark Driver
Dynamic Checker
Benchmark parameters
Performance Results


CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Language Corpus
Code Rewriter
LSTM network
Model parameters
Synthesis parameters
Synthesizer
Rejection Filter
Host Driver
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results
generate random inputs
run, measure performance


CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Language Corpus
Code Rewriter
LSTM network
Model parameters
Synthesis parameters
Synthesizer
Rejection Filter
Host Driver
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results

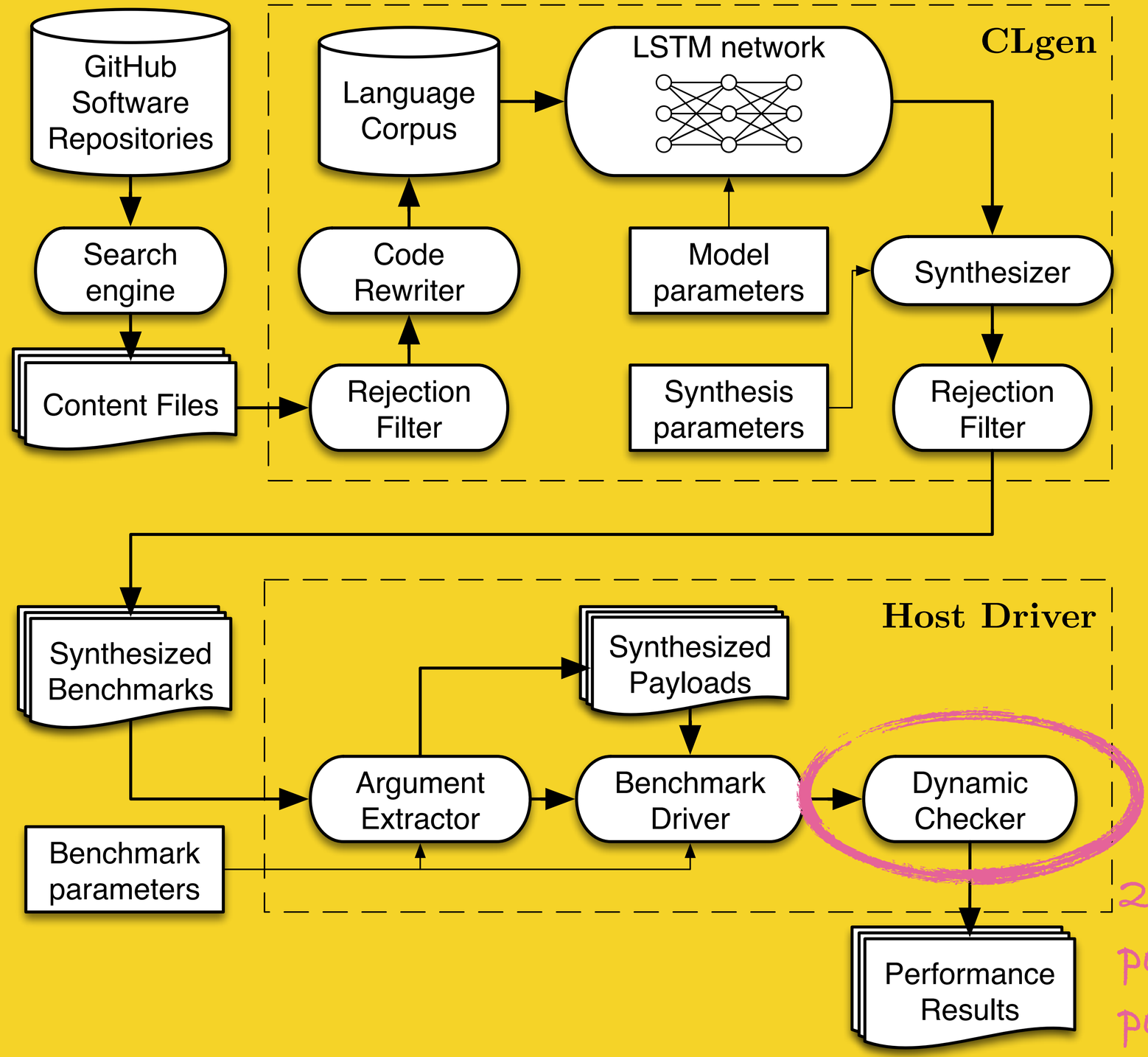


Verify:

A ≠ A else *no outputs*

A ≠ C else *input insensitive*

A = B else *non deterministic*


CLgen
GitHub Software Repositories
Search engine
Content Files
Rejection Filter
Language Corpus
Code Rewriter
LSTM network
Model parameters
Synthesis parameters
Synthesizer
Rejection Filter
Host Driver
Synthesized Benchmarks
Benchmark parameters
Argument Extractor
Synthesized Payloads
Benchmark Driver
Dynamic Checker
Performance Results
2000 benchmarks per machine per day

results

Listing 3: Sample 3

```
__kernel void A(__global int* a, __global int* b, __global int* c,
    ↪ __global int* d, const uint e) {
  const uint f = get_global_id(0);

  if (e == 0 && f == 0)
    *d = 0;
  else if (f < e) {
    int g = b[f];
    uint h = c[f];
    if (g > 0) {
      a[h] = f;
      h++;
    }
    if (f == e - 1)
      *d = h;
  }
}
```

Listing 4: Sample 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) {
    float f = b[e];
    float g = a[e];
    a[e] = f * 3.141592f / (f + 1.0f + e * 1024 - f) - (0.5f * g * 1.0f
        ↪ / 18.0f + e / 2.0f);
  }

  for (c = 0; c < 30; c++) {
    c[e] = 0;
  }
}
```

Listing 4: Sample 4

```
__kernel void A(int a, __global float* b, __global int* c, __global int*
    ↪ d, __local int* e, int f) {
  int g = get_local_id(0);
  e[get_local_id(0)] = 0;
  barrier(1);
  while (g < f) {
    int h = c[g];

    if (h != -1) {
      __global float* i = b + g * a;
      float j = 0;
```

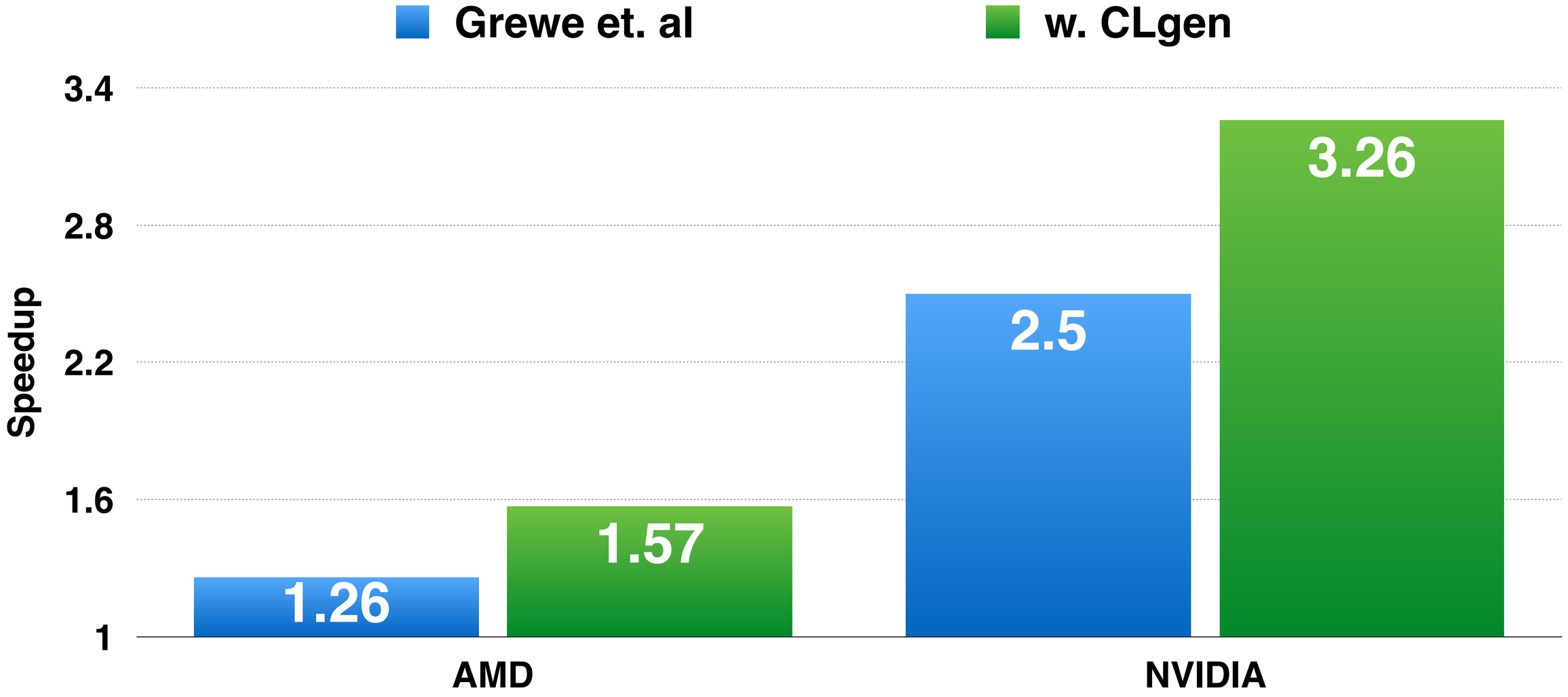
Listing 7: Sample 7

```
__kernel void A(int a, int b, int c, __global const float* d, __global
    ↪ const float* e, __global float* f, float g) {
  const int h = get_local_id(0);
  const int i = get_group_id(0);

  const int j = 4 * i + h;
  const int k = 4 * i + h + a;
  if (4 * i + h + a < c) {
    float l = 0.0;
    float m = 0.0;
    float n = 0.0;
    const float o = d[3 * (4 * i + h + a)];
    const float p = d[3 * (4 * i + h + a) + 1];
    const float q = d[3 * (4 * i + h + a) + 2];
    for (int r = 0; r < c; r++) {
      const float s = d[3 * r] - o;
      const float t = d[3 * r + 1] - p;
      const float u = d[3 * r + 2] - q;
      const float v = (d[3 * r] - o) * (d[3 * r] - o) + (d[3 * r + 1] -
          ↪ p) * (d[3 * r + 1] - p) + (d[3 * r + 2] - q) * (d[3 * r +
          ↪ 2] - q) + g;
      const float w = e[r] / (((d[3 * r] - o) * (d[3 * r] - o) + (d[3 *
          ↪ r + 1] - p) * (d[3 * r + 1] - p) + (d[3 * r + 2] - q) * (d
          ↪ [3 * r + 2] - q) + g) * sqrt((d[3 * r] - o) * (d[3 * r] - o
          ↪ ) + (d[3 * r + 1] - p) * (d[3 * r + 1] - p) + (d[3 * r + 2]
          ↪ ] - q) * (d[3 * r + 2] - q) + g));
      l = l + (d[3 * r] - o) * w;
      m = m + (d[3 * r + 1] - p) * w;
      n = n + (d[3 * r + 2] - q) * w;
    }
    f[3 * (4 * i + h)] = l;
    f[3 * (4 * i + h) + 1] = m;
    f[3 * (4 * i + h) + 2] = n;
  }
}
```

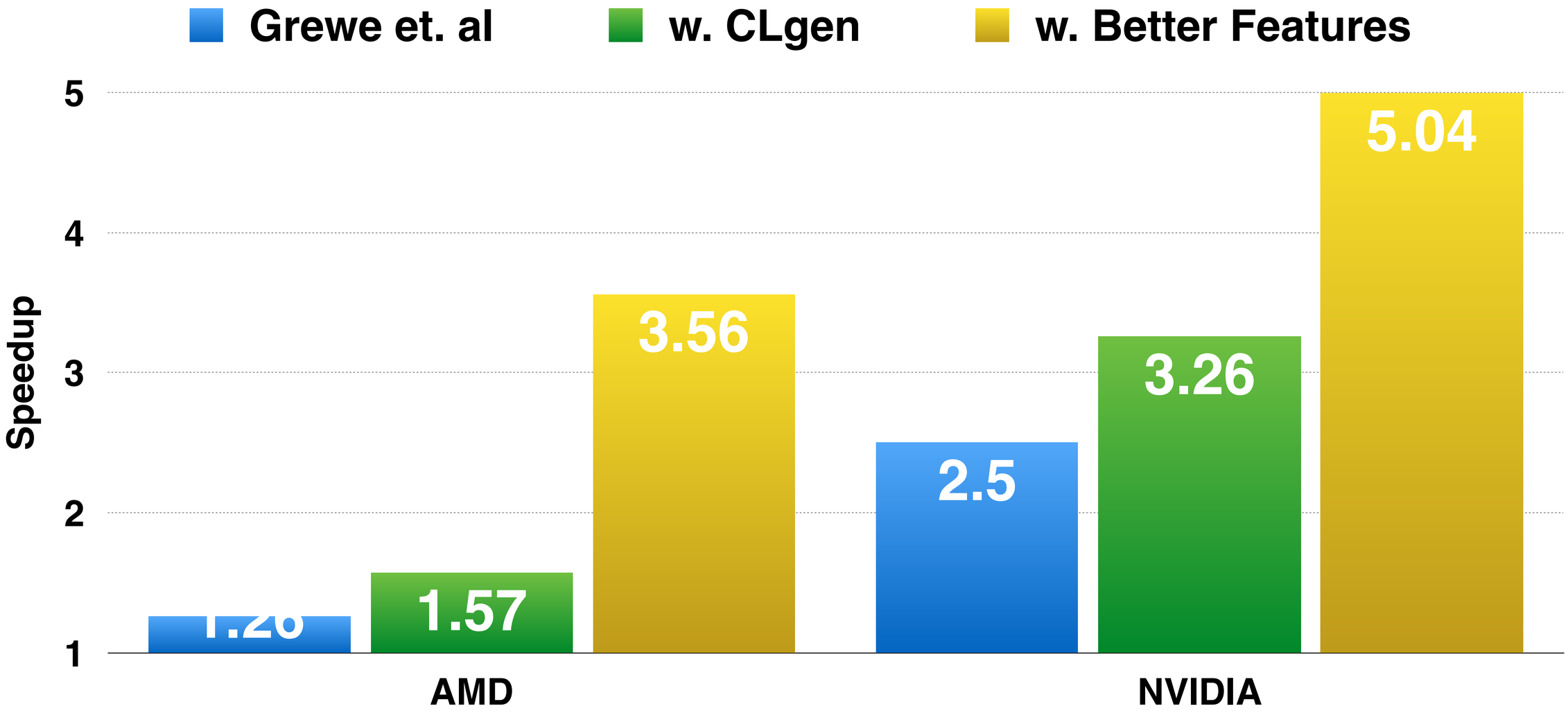

Listing 10: Sample 10

```
__kernel void A(__global ulong *a) {
    int i, j;
    struct S0 c_8;
    struct S0* p_7 = &c_8;
    struct S0 c_9 = {
        {{0x43250E6DL,2UL},{0x43250E6DL,2UL},{0x43250E6DL,2UL},
         {0x43250E6DL,2UL},{0x43250E6DL,2UL},{0x43250E6DL,2UL},
         {0x43250E6DL,2UL},{0x43250E6DL,2UL}},
        0x4BF90EDCAD2086BDL,
    };
    c_8 = c_9;
    barrier(0 | 1);
```

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



71 benchmarks, 1,000 synthetic benchmarks. 4.30x faster



concluding remarks

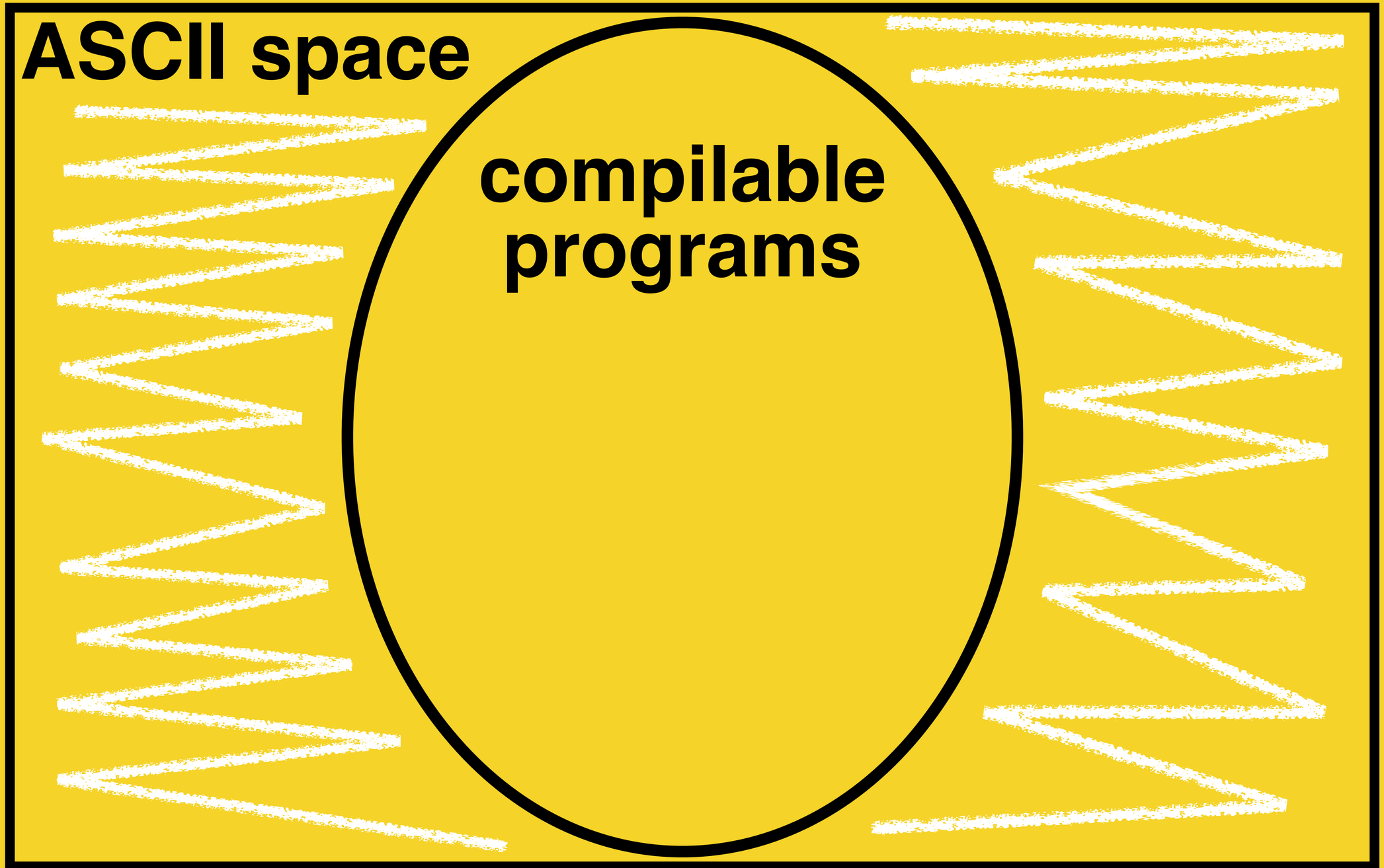
problem: insufficient benchmarks

use DL to learn PL semantics and usage

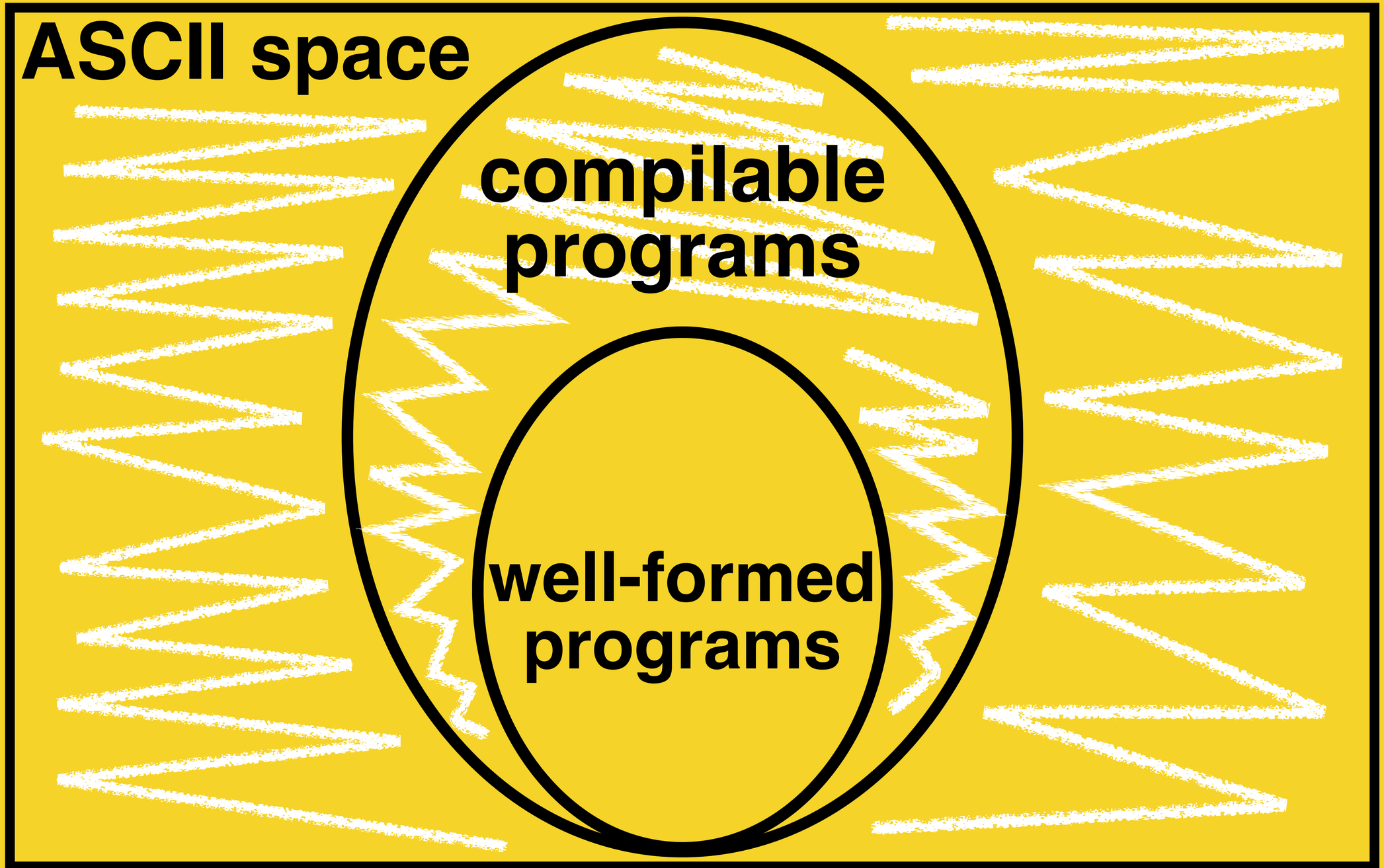
turing tested! ;-)

improved model performance and design

call to arms



call to arms



Synthesizing Benchmarks for Predictive Modeling

For the paper, code and data:

http://chriscummins.cc/cgo17

For the TensorFlow neural network:

http://chriscummins.cc/clgen