

Deep Learning in Compilers

Chris Cummins

Pavlos Petoumenos

Zheng Wang

Hugh Leather



THE UNIVERSITY of EDINBURGH
informatics

EPSRC Centre for Doctoral Training in
Pervasive Parallelism

machine learning for compilers

$$\mathbf{y} = \mathbf{f}(\mathbf{x})$$

Decisions

CPU or GPU

Workgroup size

Cflags

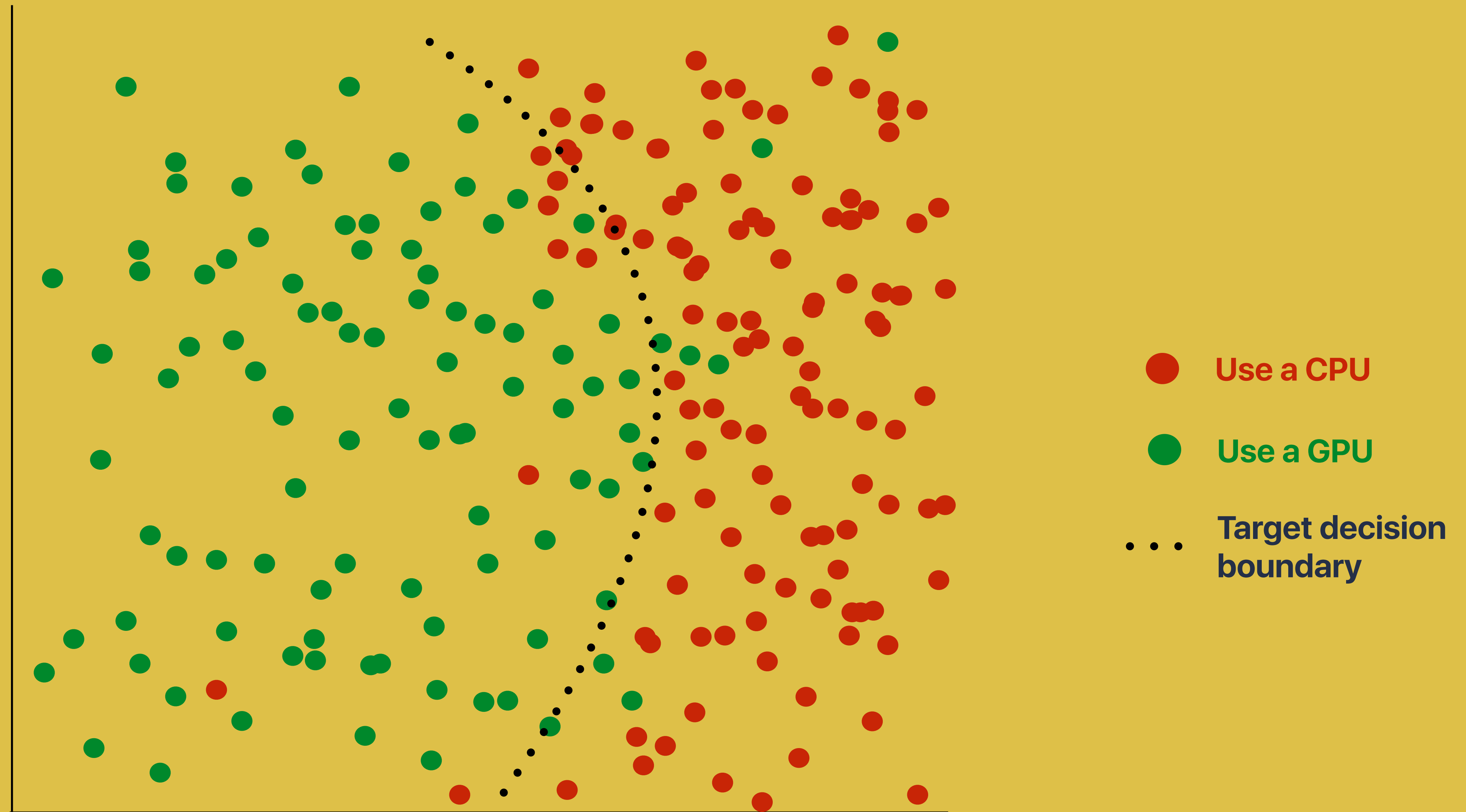
Features

instructions

Arithmetic density

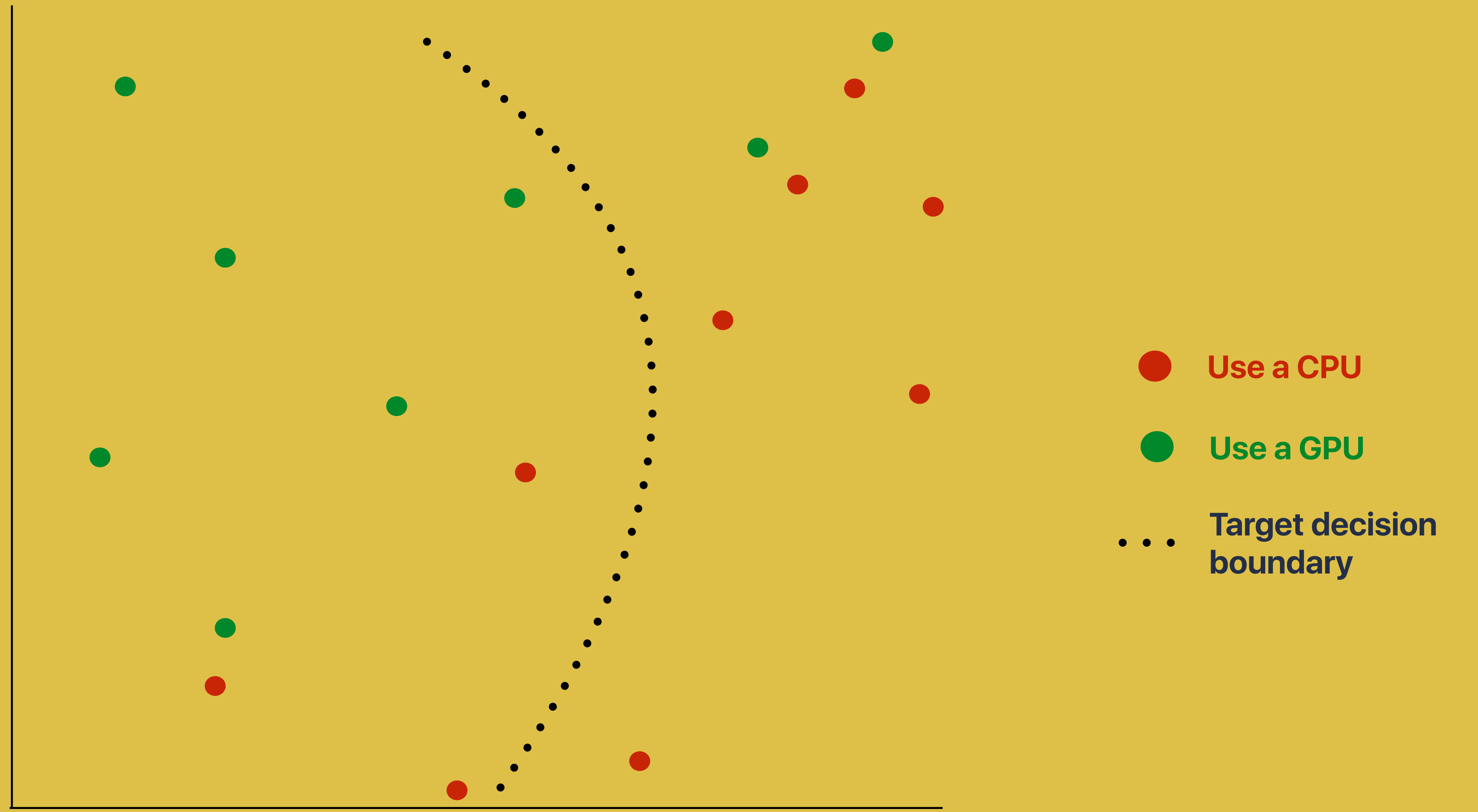
Dataset size

machine learning for compilers



The idea.

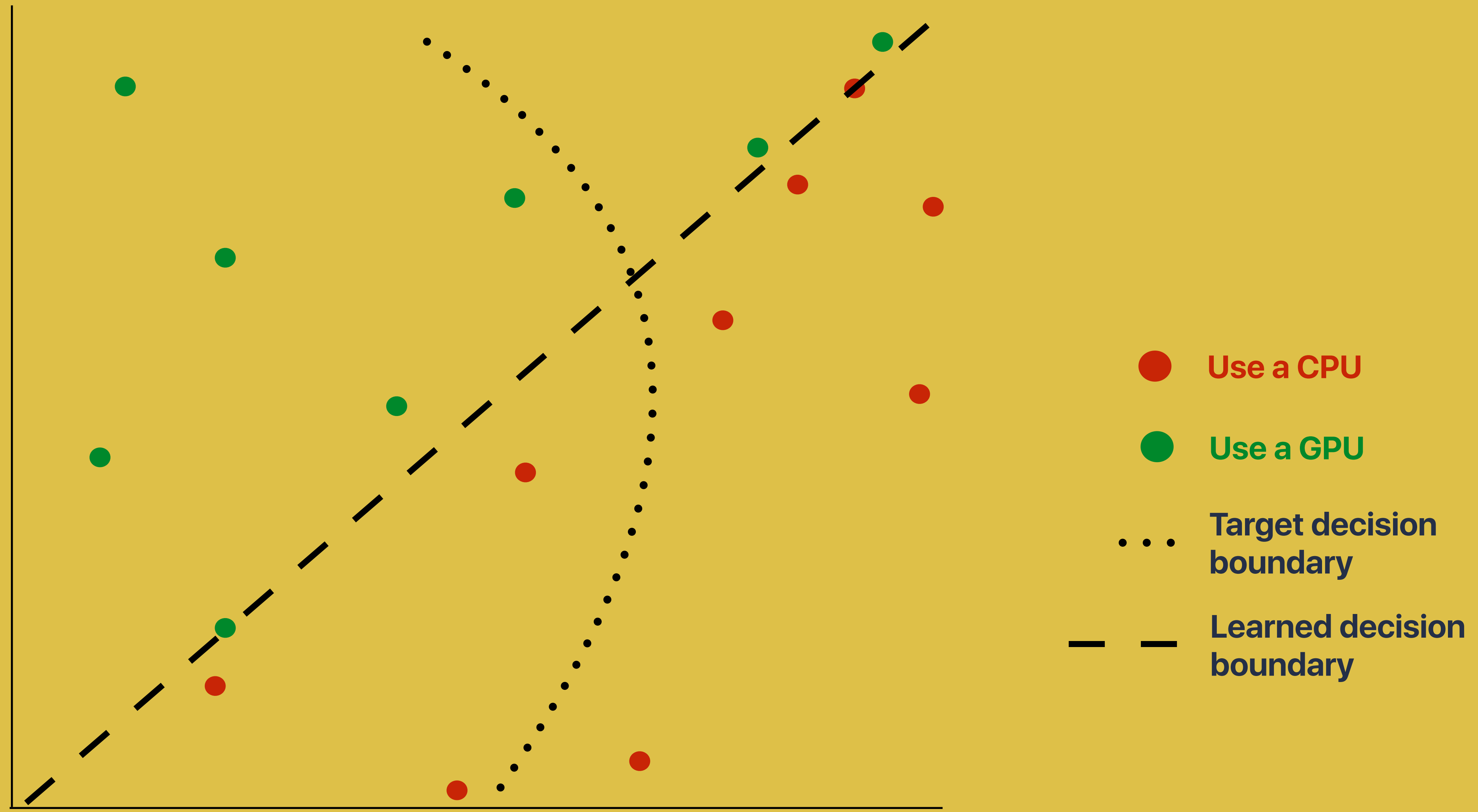
machine learning for compilers



The reality.

*17 benchmarks in avg
compiler paper 2013-2016

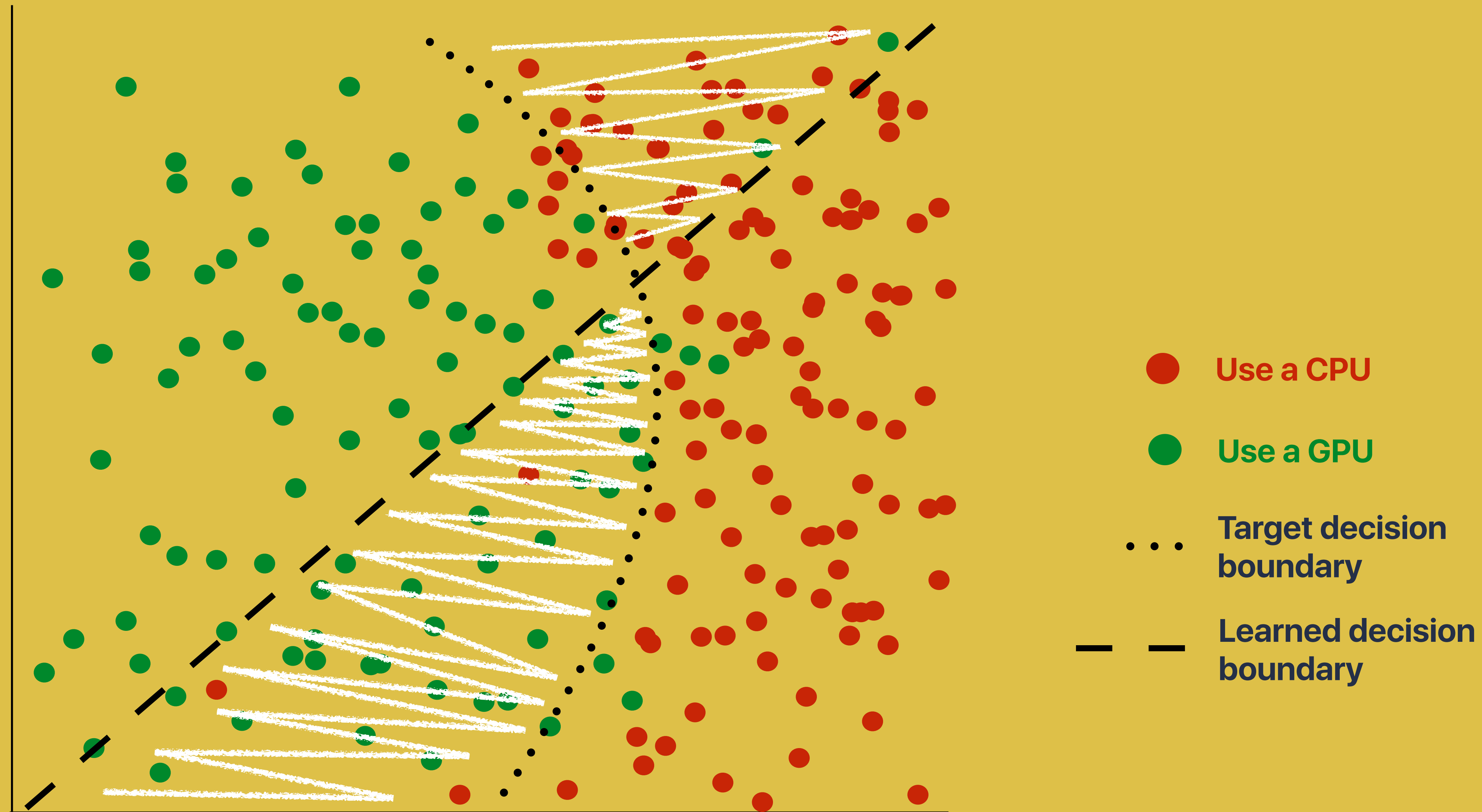
machine learning for compilers



The reality.

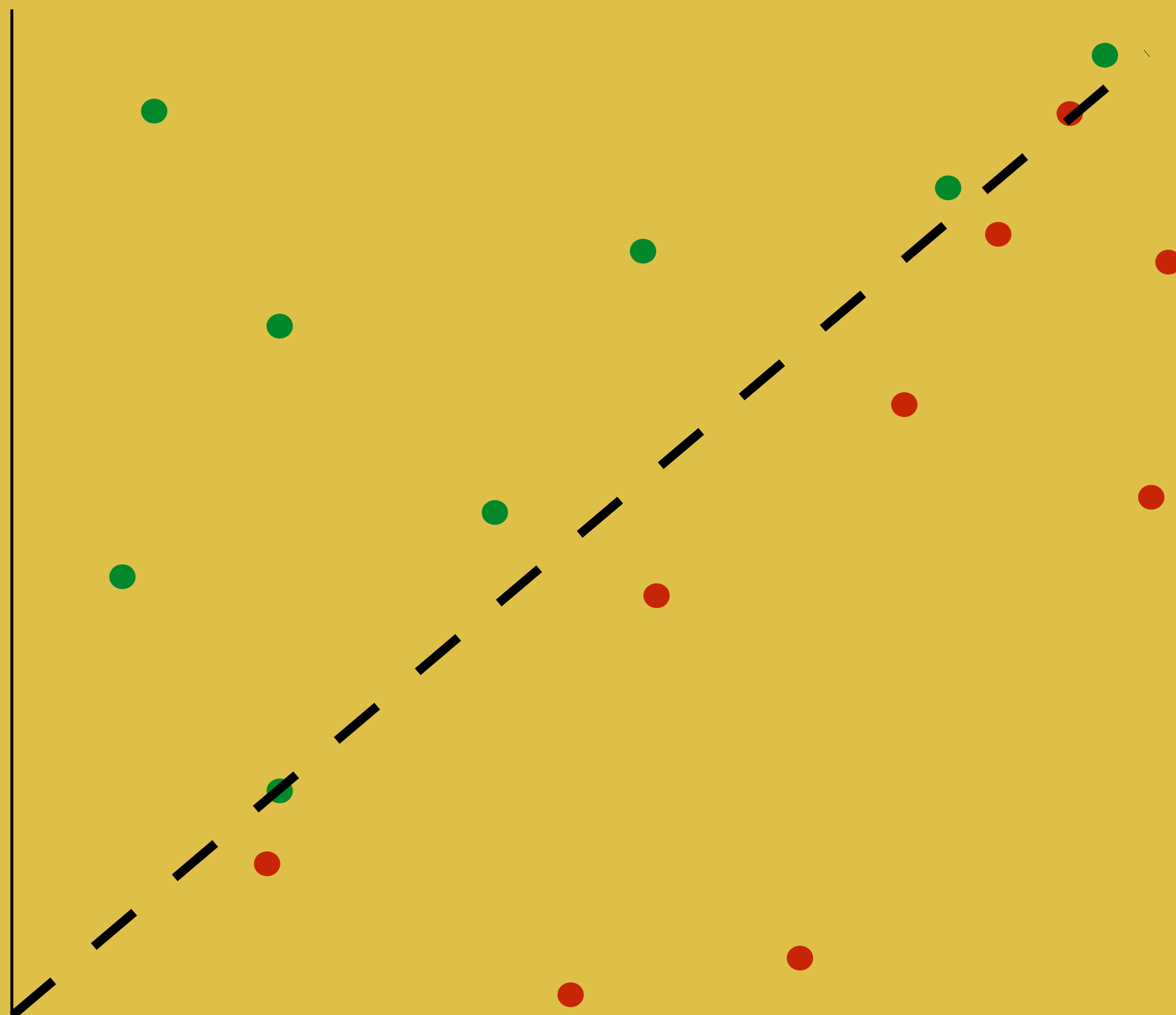
*17 benchmarks in avg
compiler paper 2013-2016

machine learning for compilers

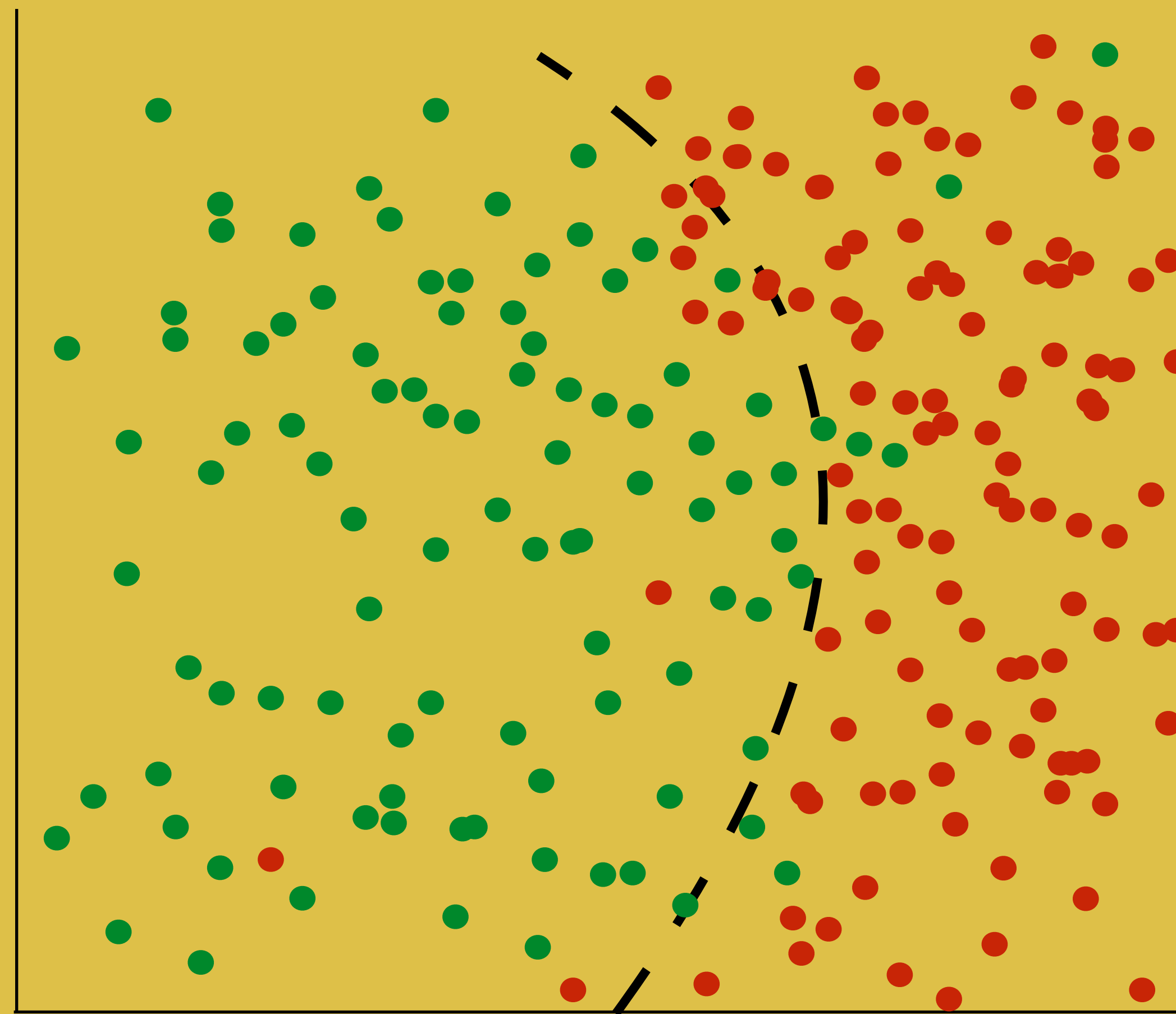


sparse data leads to inaccurate models!

what we need

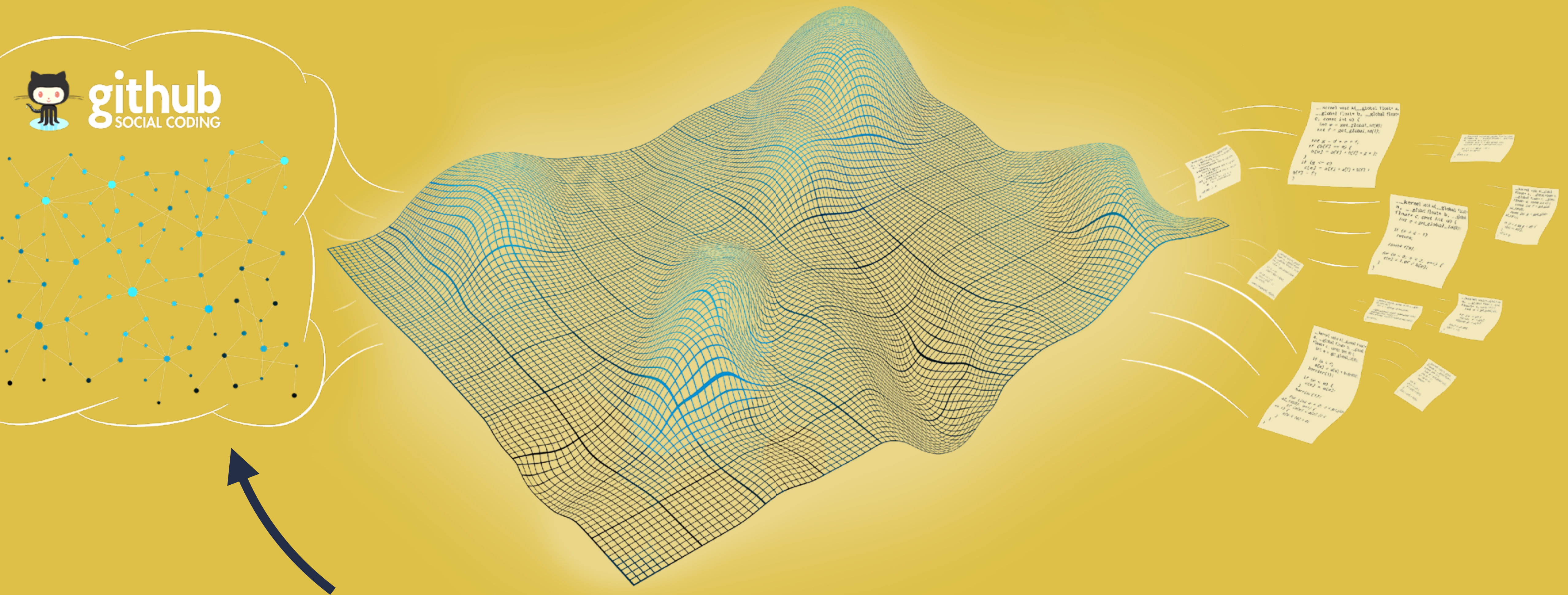


from this



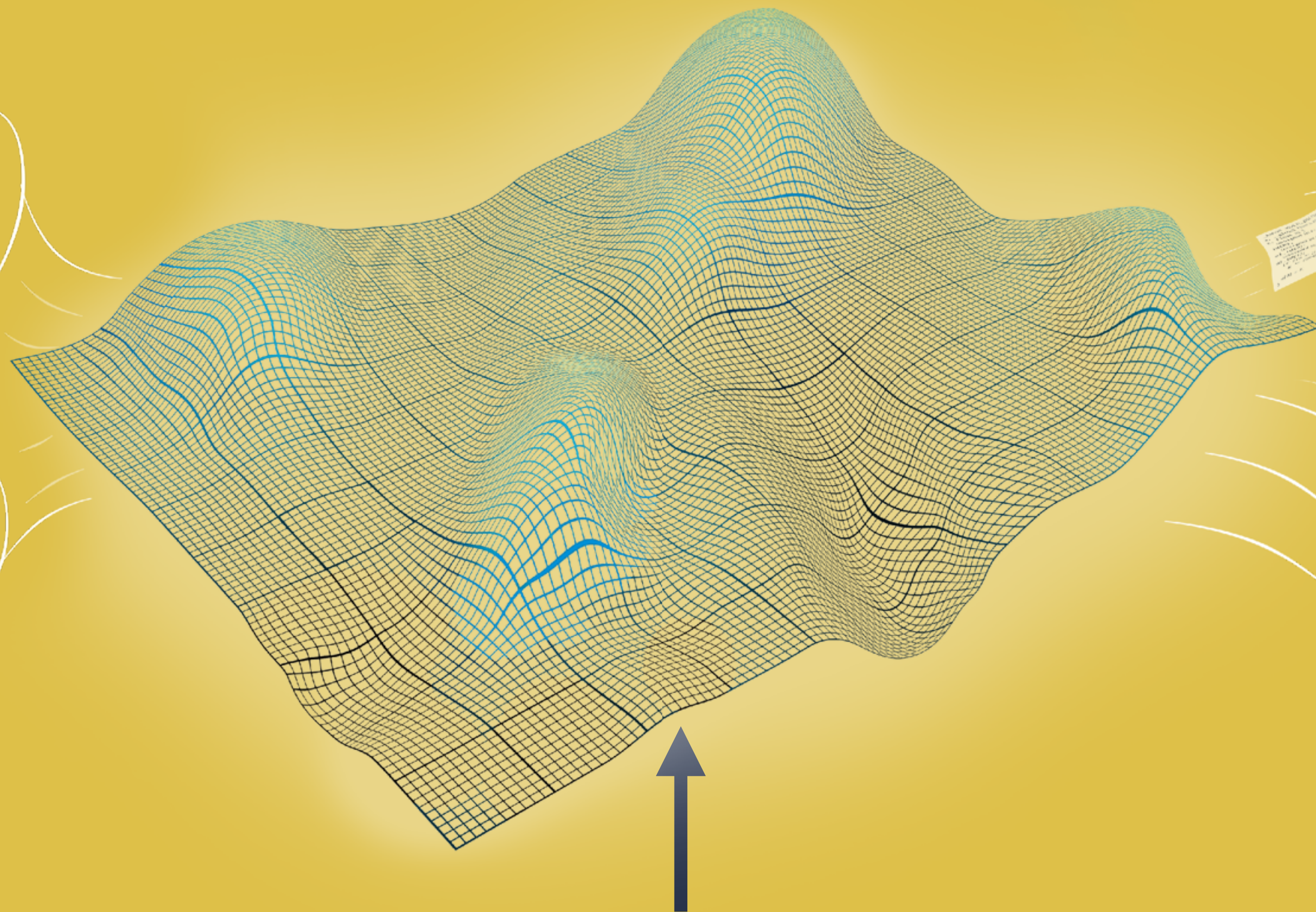
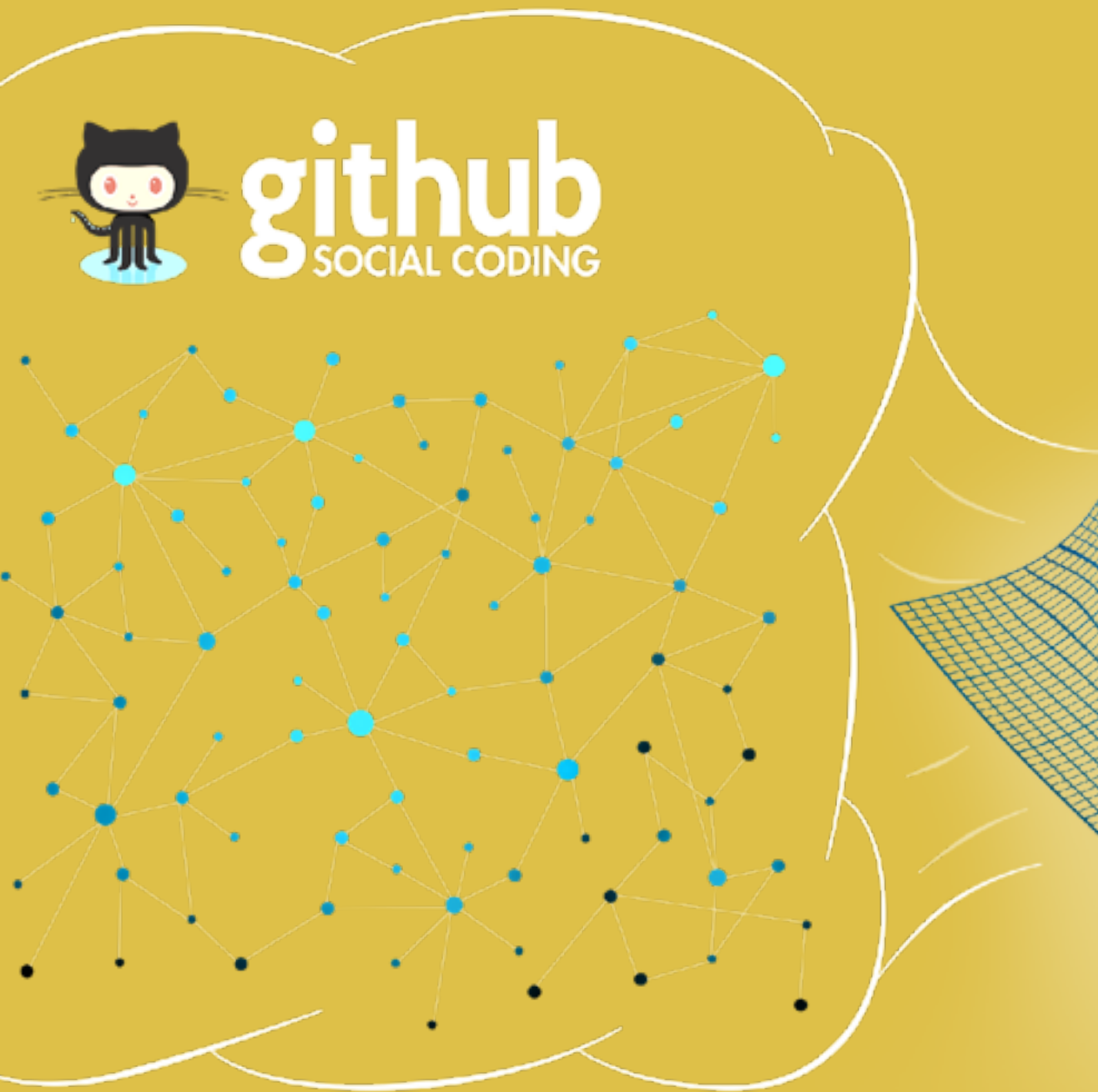
to this

our approach



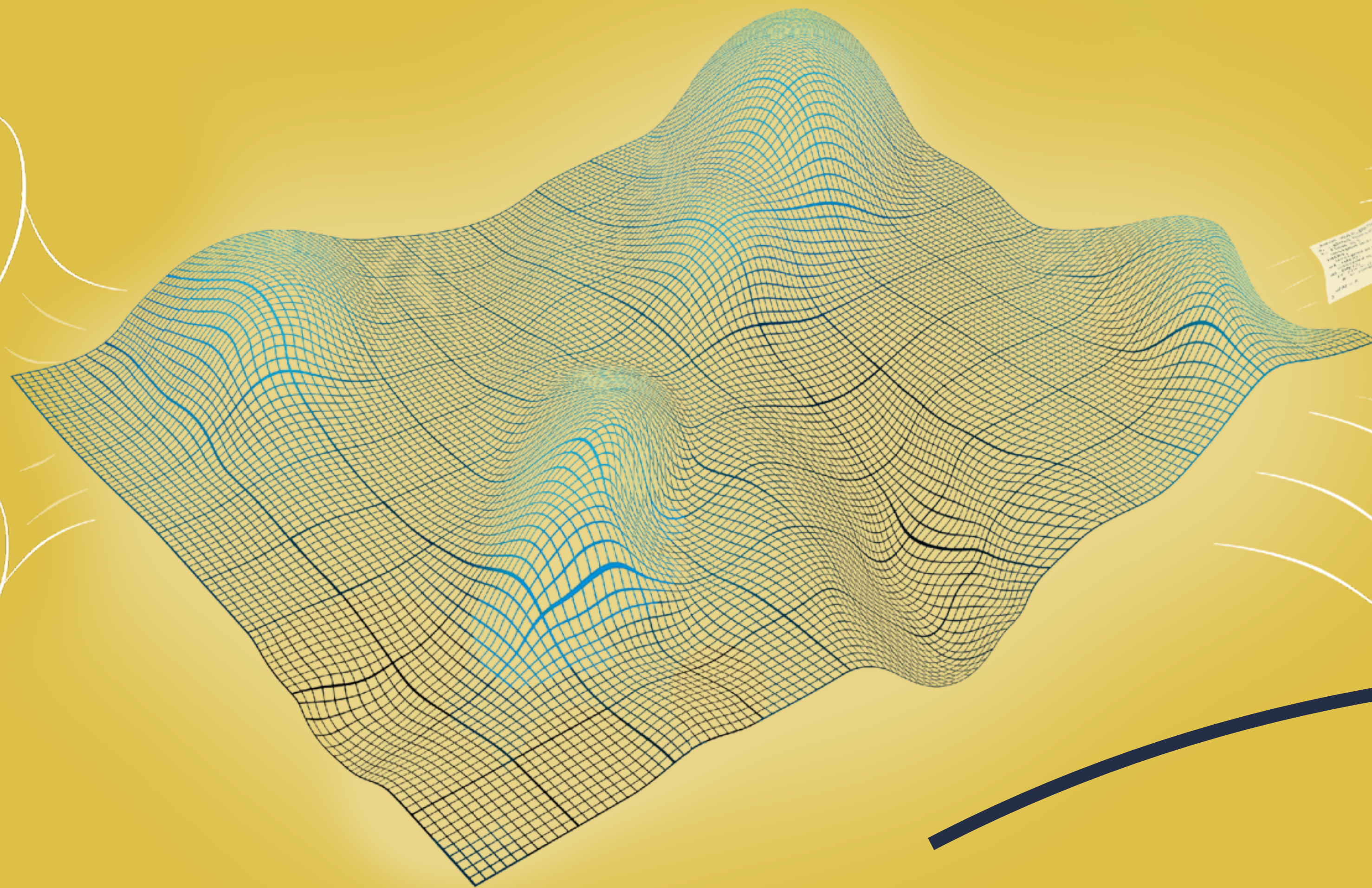
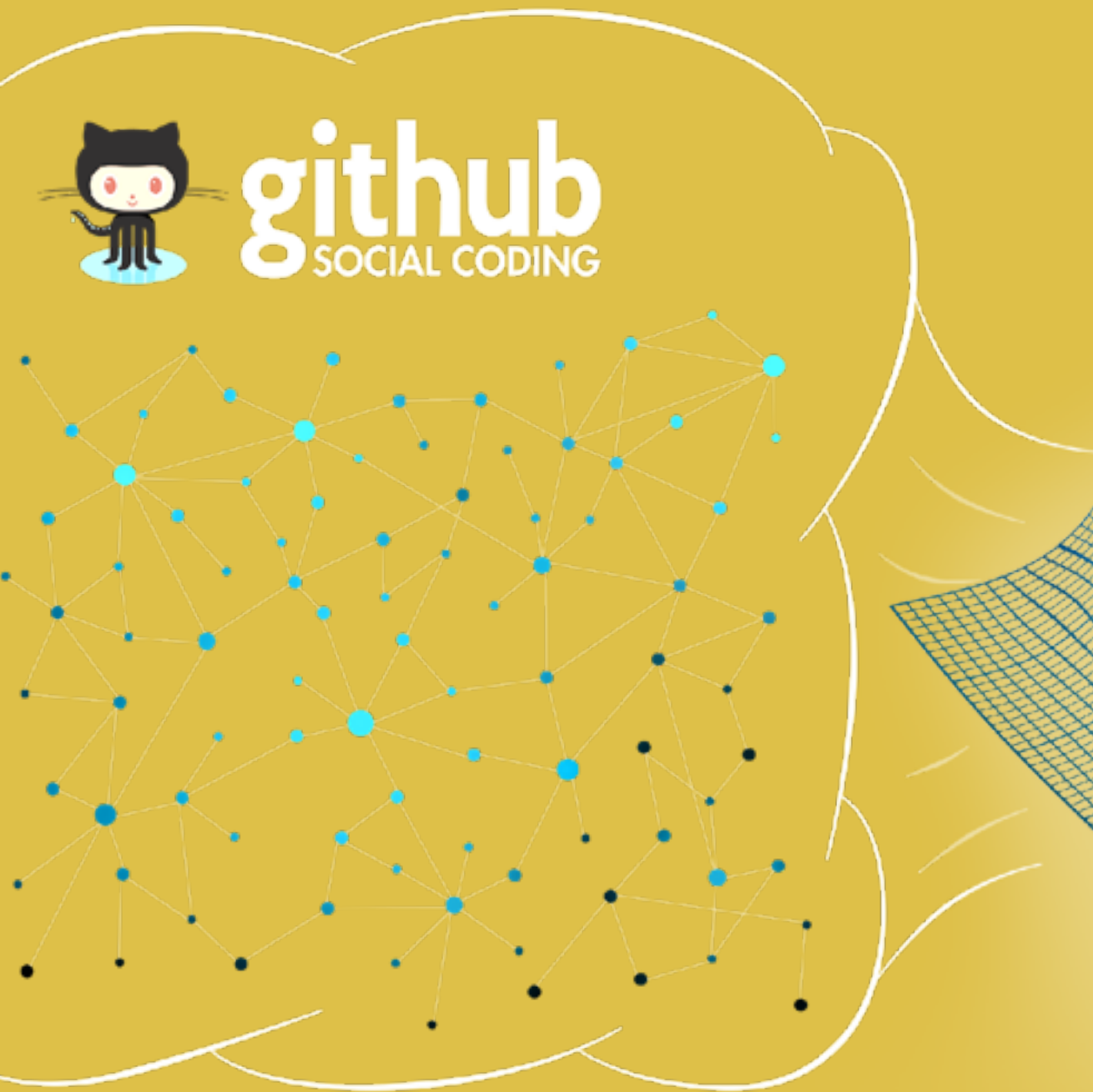
mine code from web

our approach



model source distr.

our approach



sample lang. model

Listing 3: Sample 3

```

1  __kernel void A(__global int* a, __global int* b, __global int* c,
    ↪ __global int* d, const uint e) {
2      const uint f = get_global_id(0);
3
4      if (e == 0 && f == 0)
5          *d = 0;
6      else if (f < e) {
7          int g = b[f];
8          uint h = c[f];
9          if (g > 0) {
10             a[h] = f;
11             h++;
12         }
13         if (f == e - 1)
14             *d = h;
15     }
16 }

```

Listing 4: Sample 4

```

1  __kernel void A(__global float* a, __global float* b, __global float* c,
    ↪ const int d) {
2      int e = get_global_id(0);
3
4      if (e < d) {
5          float f = b[e];
6          float g = a[e];
7          a[e] = f * 3.141592f / (f + 1.0f + e * 1024 - f) + (0.5f * f * g * 1.0f
    ↪ / 18.0f + e / 2.0f);
8      }
9
10     for (e = 0; e < 30; e++) {
11         c[e] = 0;
12     }
13 }

```

Listing 4: Sample 4

```

1  __kernel void A(int a, __global float* b, __global int* c, __global int*
    ↪ d, __local int* e, int f) {
2      int g = get_local_id(0);
3      e[get_local_id(0)] = 0;
4      barrier(1);
5      while (g < f) {
6          int h = c[g];
7
8          if (h != -1) {
9              __global float* i = b + g * a;
10             float j = 0;

```

Listing 7: Sample 7

```

1  __kernel void A(int a, int b, int c, __global const float* d, __global
    ↪ const float* e, __global float* f, float g) {
2      const int h = get_local_id(0);
3      const int i = get_group_id(0);
4
5      const int j = 4 * i + h;
6      const int k = 4 * i + h + a;
7      if (4 * i + h + a < c) {
8          float l = 0.0;
9          float m = 0.0;
10         float n = 0.0;
11         const float o = d[3 * (4 * i + h + a)];
12         const float p = d[3 * (4 * i + h + a) + 1];
13         const float q = d[3 * (4 * i + h + a) + 2];
14         for (int r = 0; r < c; r++) {
15             const float s = d[3 * r] - o;
16             const float t = d[3 * r + 1] - p;
17             const float u = d[3 * r + 2] - q;
18             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;
19             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));
20             l = l + s * w;
21             m = m + t * w;
22             n = n + u * w;
23         }
24         f[j] = l;
25         f[k] = m;
26         f[l] = n;
27     }
28 }

```

Listing 10: Sample 10

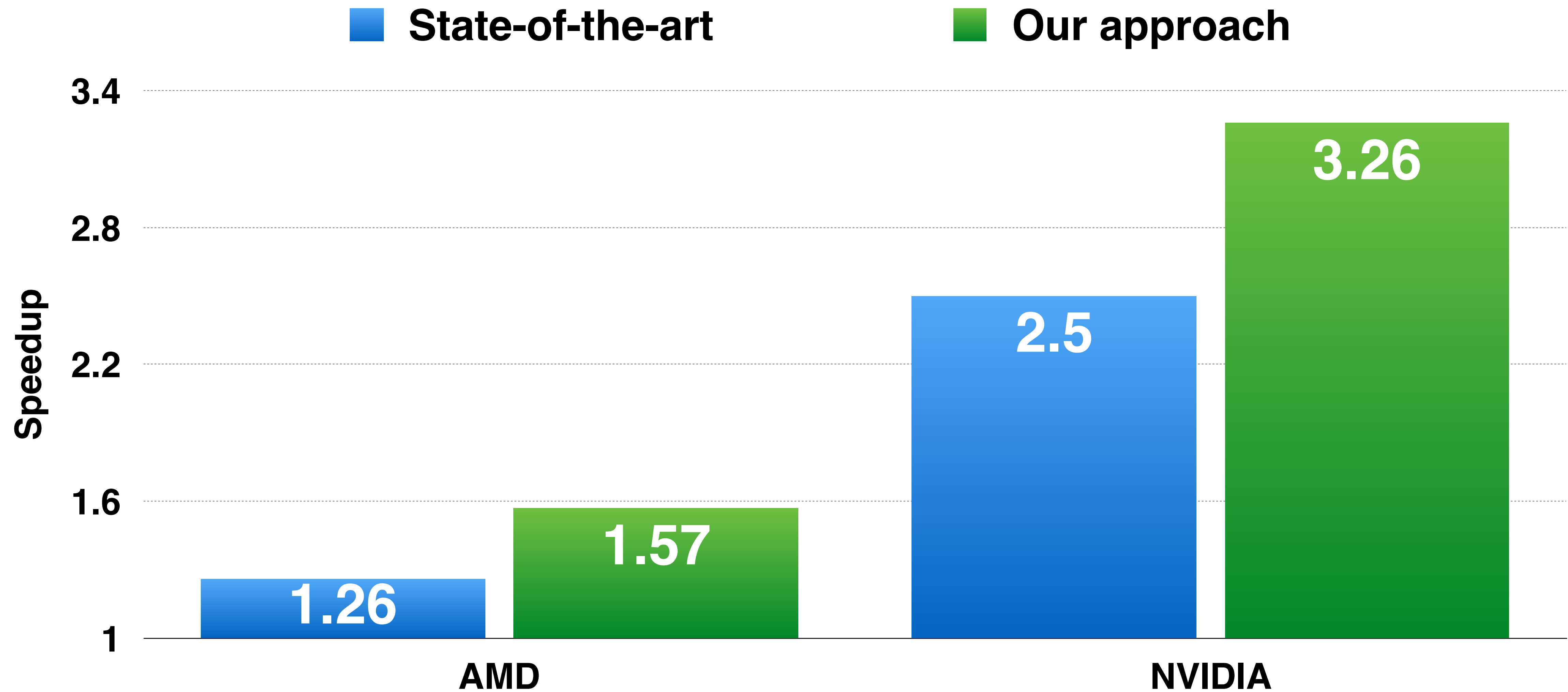
```

1  __kernel void A(__global ulong *a) {
2      int i, j;
3      struct S0 c_8;
4      struct S0* p_7 = &c_8;
5      struct S0 c_9 = {
6          {{0x43250E6DL, 2UL}, {0x43250E6DL, 2UL}, {0x43250E6DL, 2UL}},
7          {{0x43250E6DL, 2UL}, {0x43250E6DL, 2UL}, {0x43250E6DL, 2UL}},
8          {{0x43250E6DL, 2UL}, {0x43250E6DL, 2UL}},
9          0x4BF90EDCAD2086BDL,
10     };
11     c_8 = c_9;
12     barrier(0 | 1);
13 }

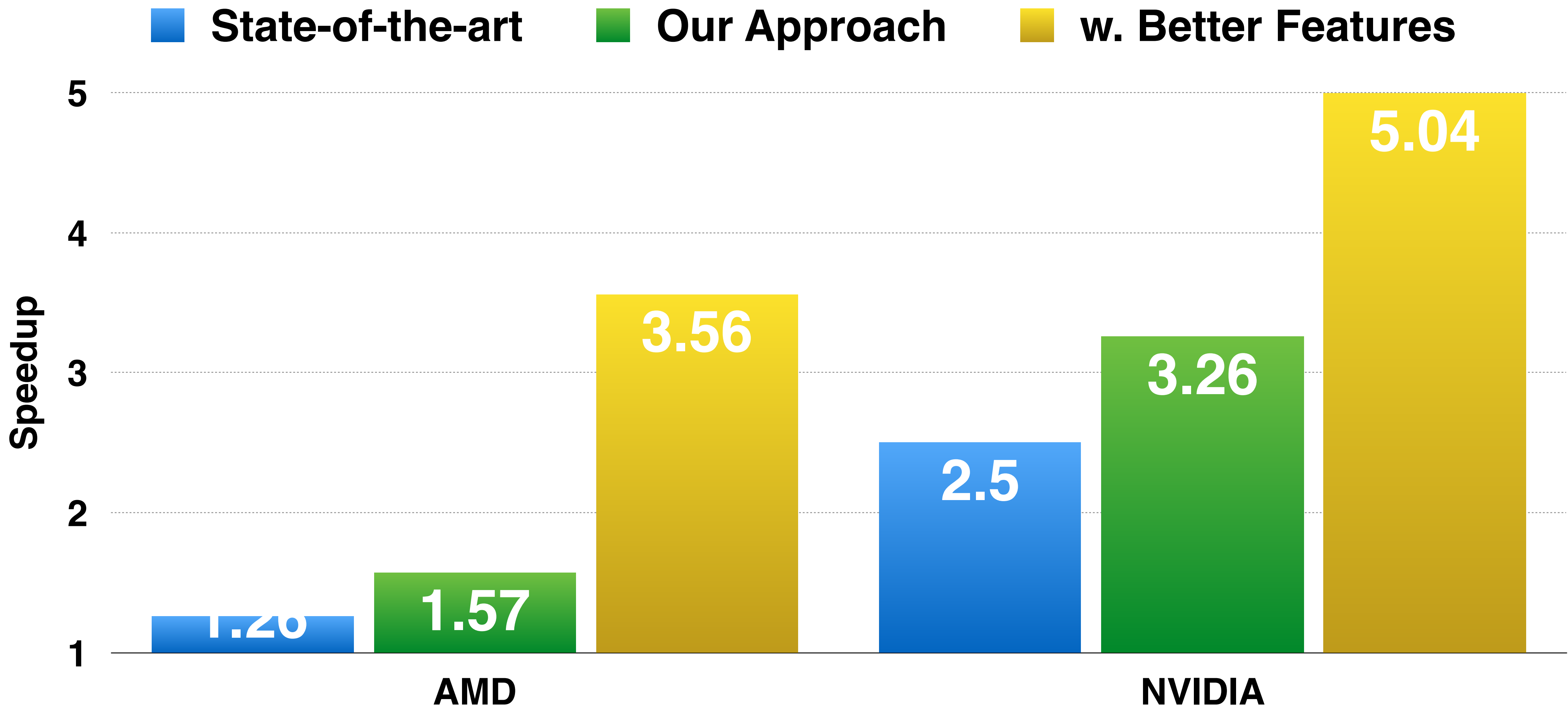
```

52%
blind test
<http://humanorrobot.uk>

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



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



Synthesizing Benchmarks for Predictive Modeling

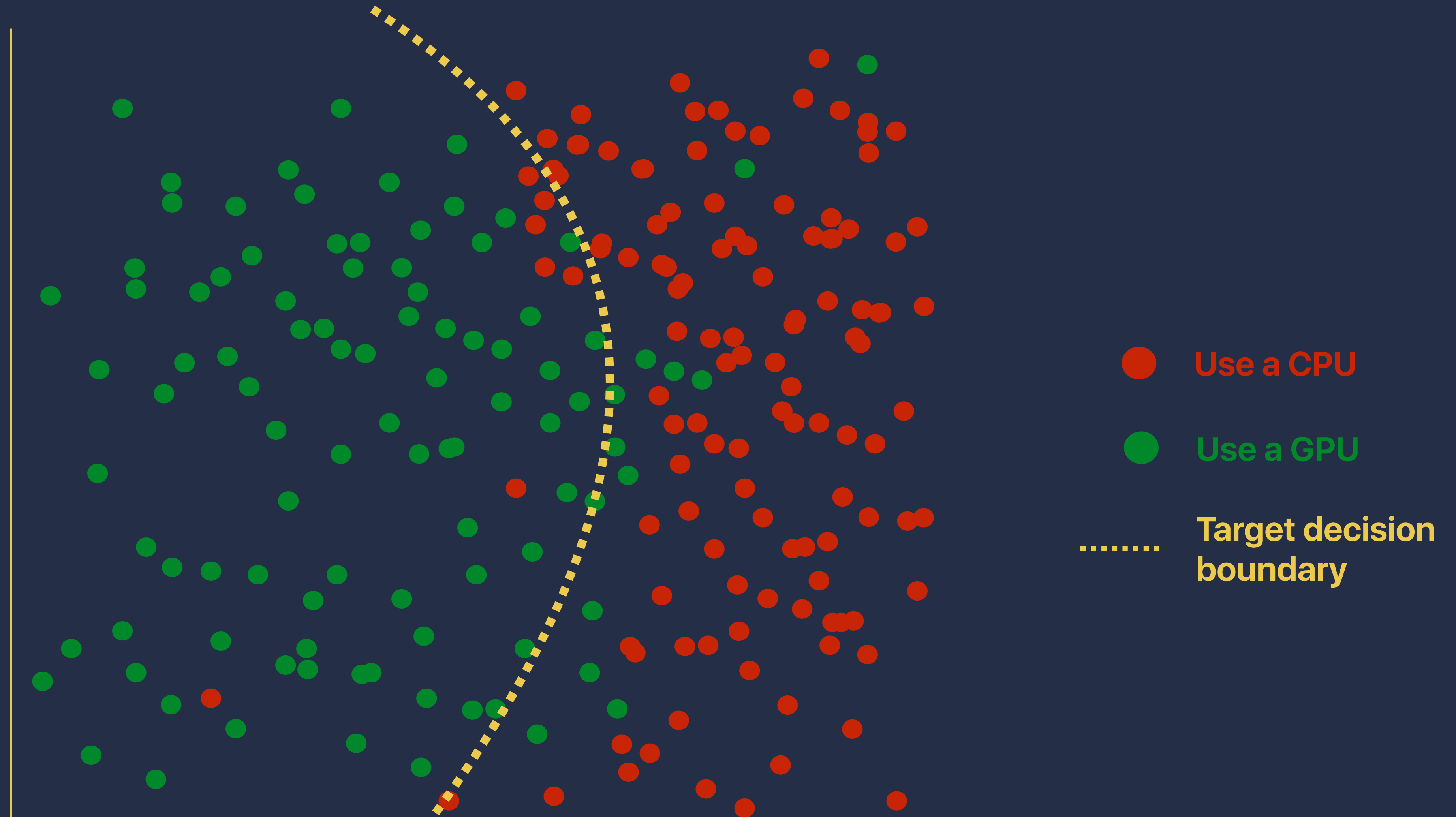
problem: insufficient benchmarks

**first solution for general-purpose
benchmark synthesis**

turing tested! ;-)

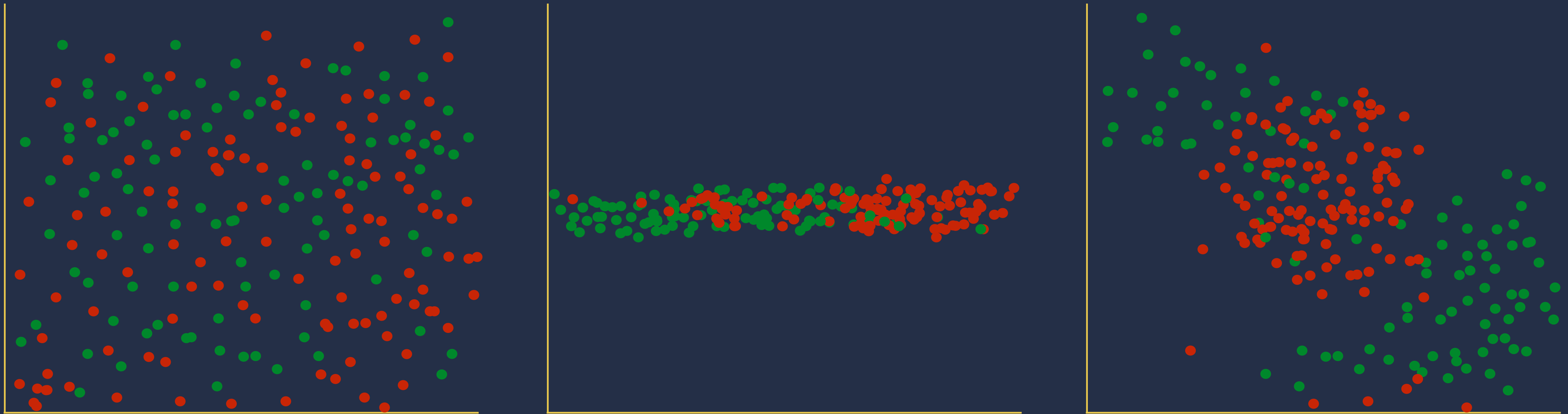
improved model performance and design

machine learning for compilers



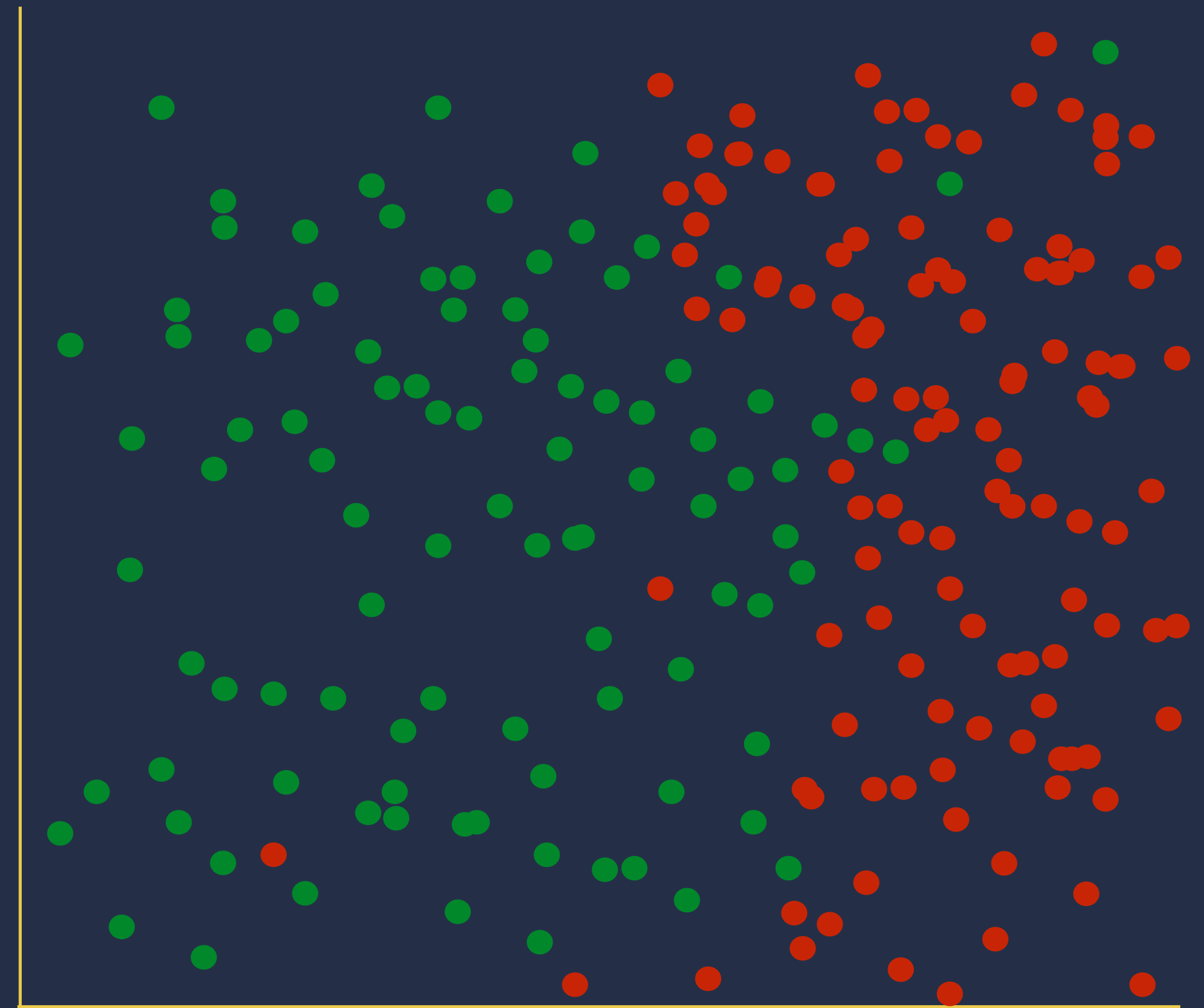
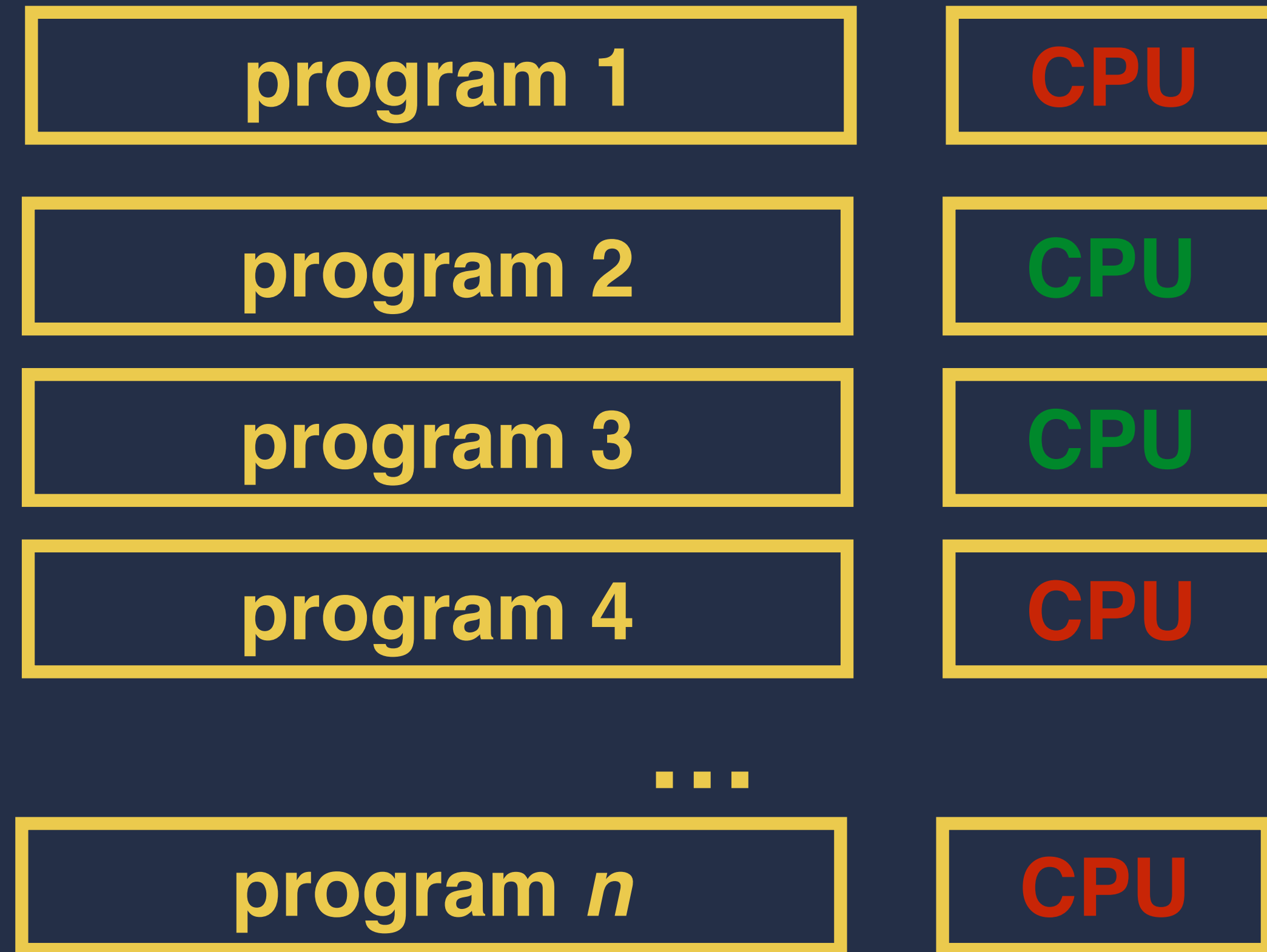
The idea.

feature design is hard



performance depends on good features!

what we need



from this

to this

our approach

```
__kernel void matrix_multiply(...
```

raw program code



Source Transformations



Deep Neural Networks



Predicted Optimization

{CPU, GPU}

our approach

```
__kernel void matrix_multiply(...
```

raw program code



Source Transformations



Deep Neural Networks



Predicted Optimization

{CPU, GPU}

} **Problem-agnostic**

our approach

Optimisation Problem A

```
__kernel void matrix_multiply(...
```

raw program code



Source Transformations



Deep Neural Networks



Predicted Optimization

{CPU, GPU}

Optimisation Problem B

```
__kernel void matrix_multiply(...
```

raw program code



Source Transformations



Deep Neural Networks

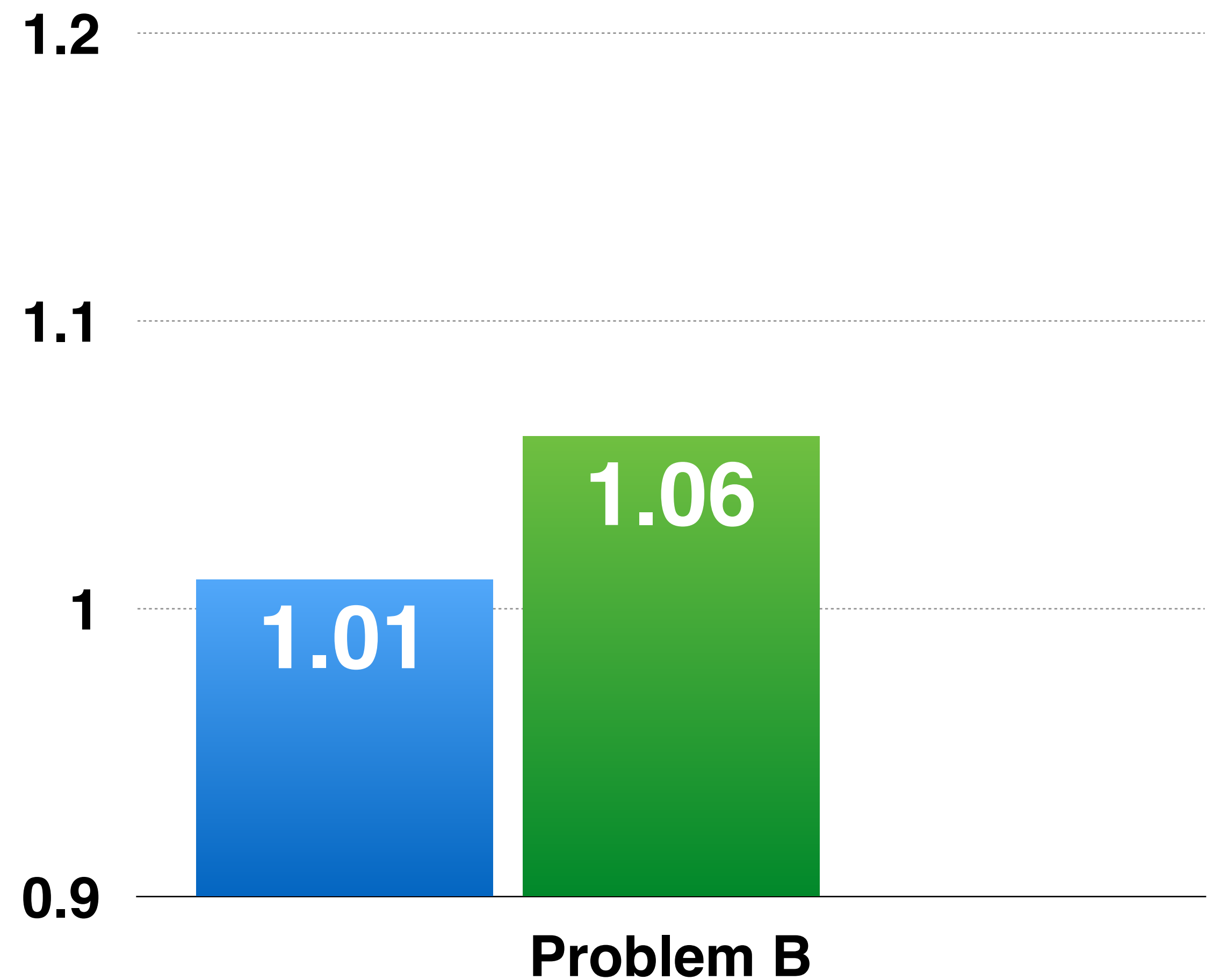
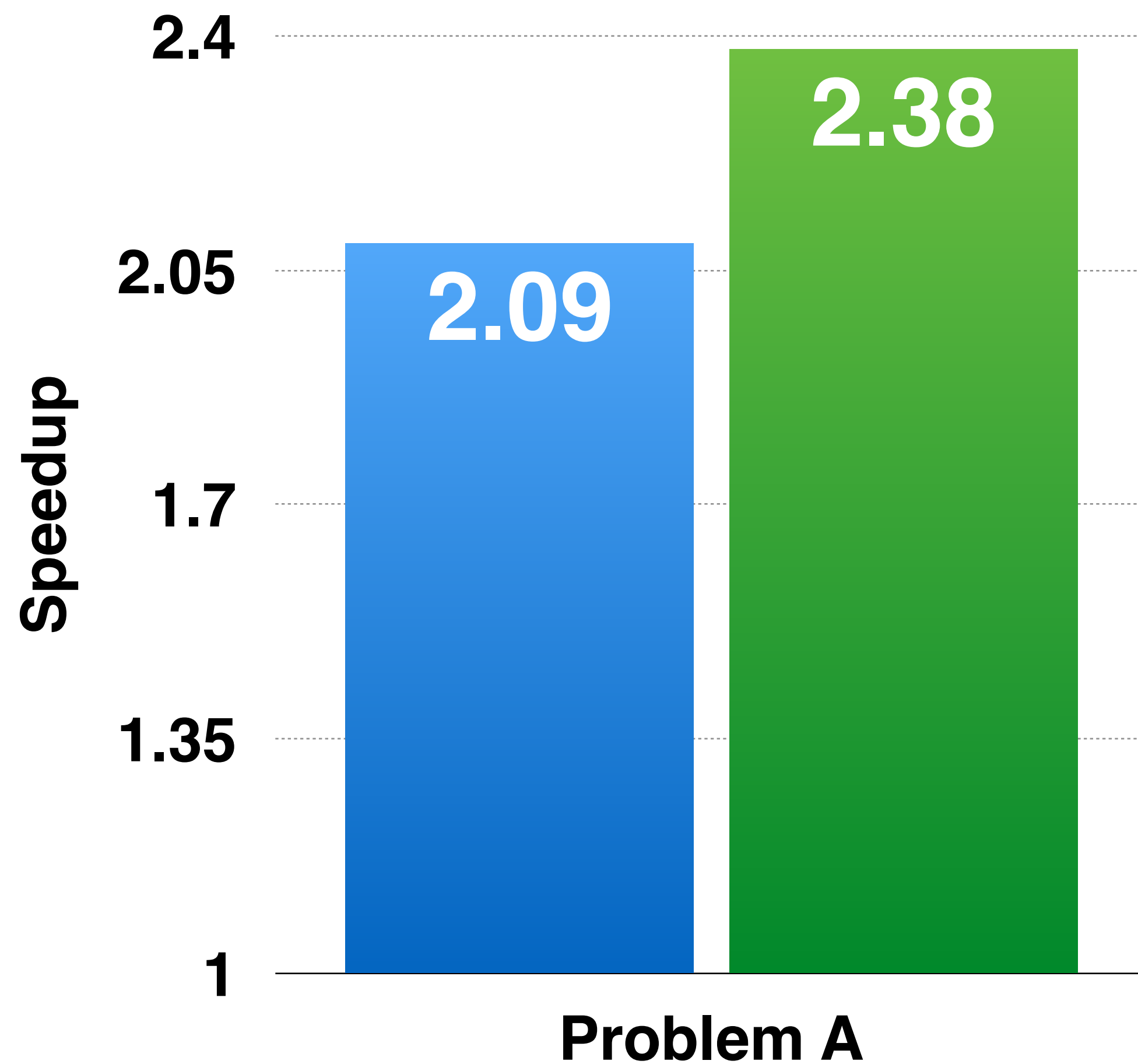


Predicted Optimization

{CF:1, CF:2, CF:4, CF:8, CF:16, CF:32}

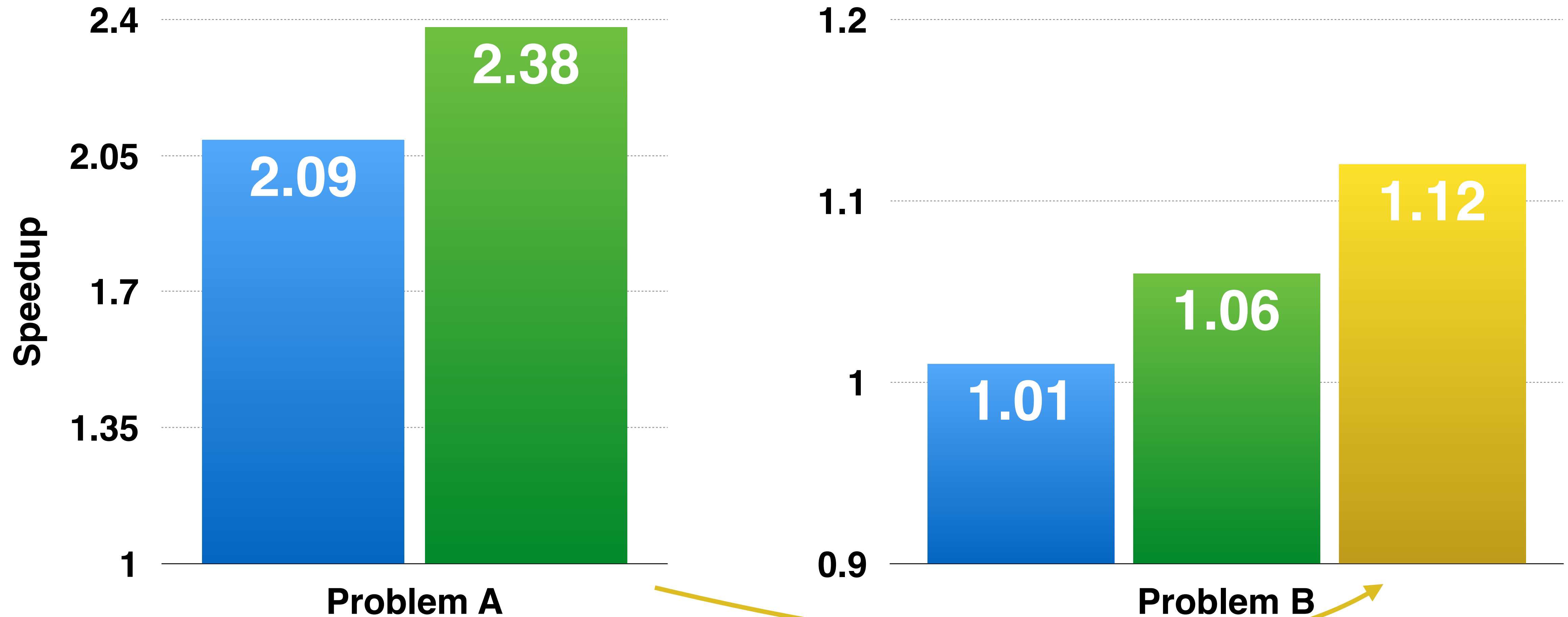
14% and 12% improvements over state-of-the-art

■ State-of-the-art ■ DeepTune



14% and 12% improvements over state-of-the-art

■ State-of-the-art ■ DeepTune ■ w. Transfer Learning



End-to-end Deep Learning of Optimisation Heuristics

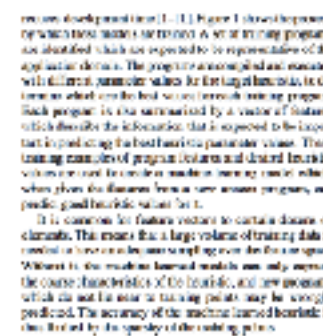
problem: feature design is hard

**“featureless” heuristic learning from raw
program code**

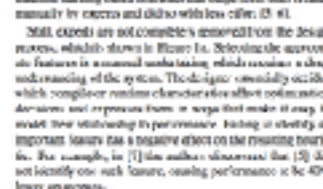
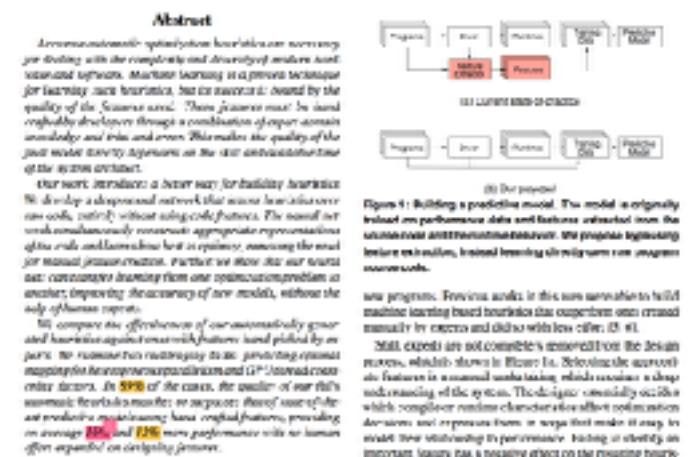
outperforms expert approach

**first application of learning *across*
optimisation domains**

<https://chriscummins.cc/cgo17>



<https://chriscummins.cc/pact17>



new programs. Previous work in this area has been able to tell machine learning-based heuristics that competent human cranes manually by experts and did so with less than 10% error (3).

Still, experts are not completely removed from the design process, which shows in Figure 1a. Selecting the accurate set features in a manual shape taking module involves a trade-off balancing of the system. The design community still holds complex, random characteristics without consistency decisions and represents them in a separate model as they exist. Instead, their withdrawal in performance. Finding a strategy of important features can be a negative effect on the resulting feature set. Examples are (1) the number of dimensions that (2) do not identify one such feature, causing performance to be 40% lower on average.

[illegible]