15-884: Machine Learning Systems

Automating ML Compilation

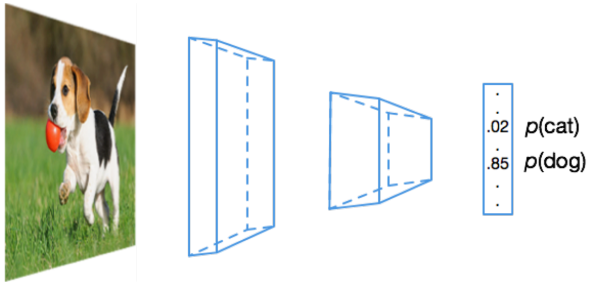
Instructor: Tianqi Chen

Carnegie Mellon University
School of Computer Science

catalyst

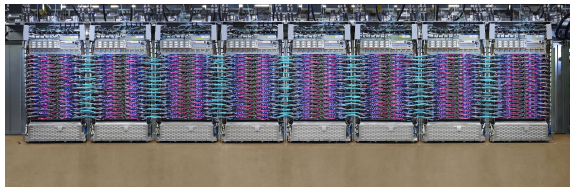
ML Compilation

ML Models



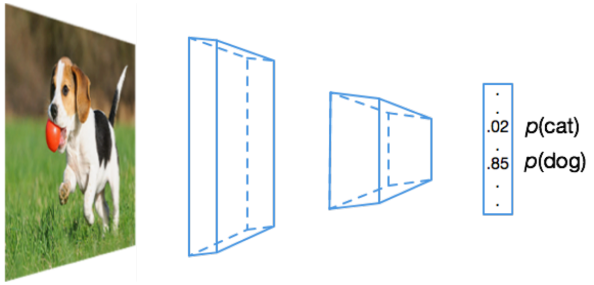
ML Compiler

Direct code generation



ML Compilation

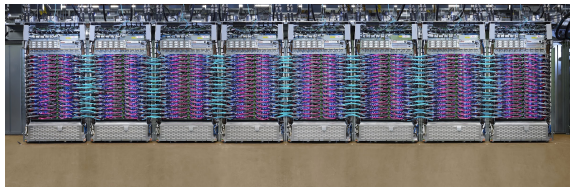
ML Models



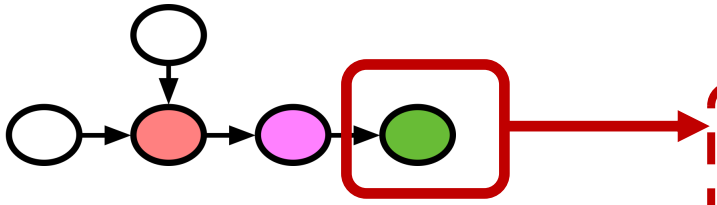
High-level IR Optimizations and Transformations

Tensor Operator Level Optimization

Direct code generation



Big Space of Possible Transformations



Specification

```
C = tvm.compute((m, n),
    lambda y, x: tvm.sum(A[k, y] * B[k, x], axis=k))
```

Huge space of possible choices

- Loop Transformations
- Thread Bindings
- Cache Locality
- Thread Cooperation
- Tensorization
- Latency Hiding
- …

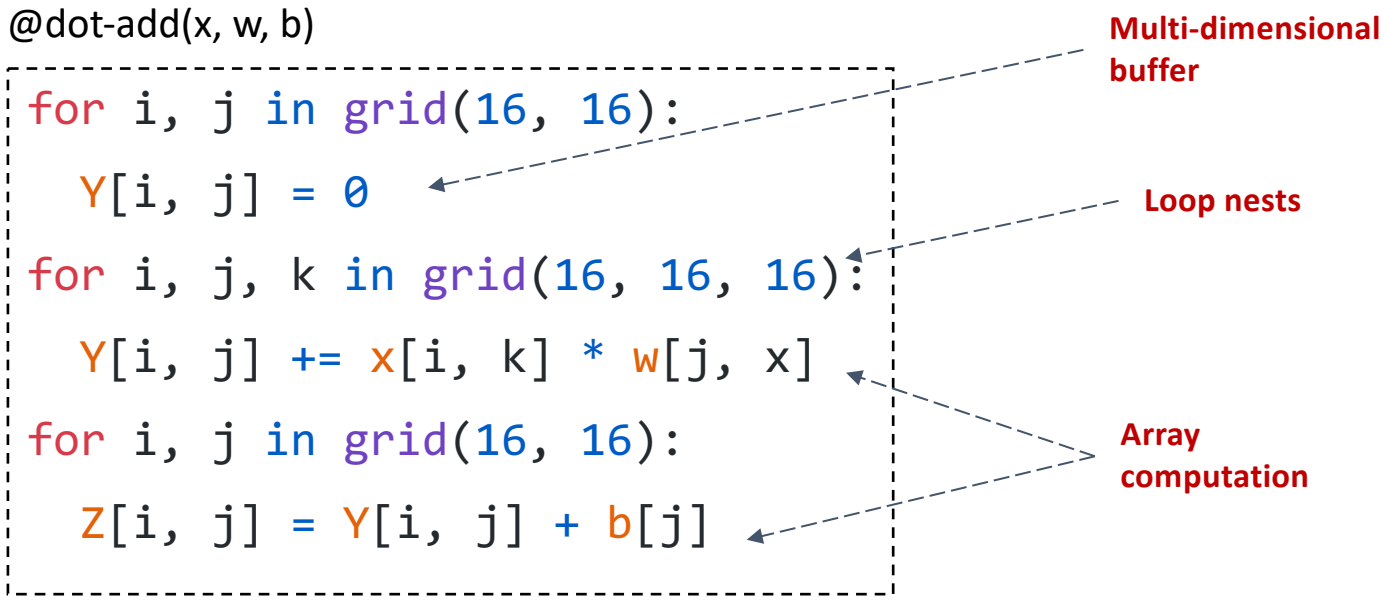
Hardware

Elements of an Automated ML Compiler

- Program representation
- Comprehensive structural search space
- Effective search

Program Representation

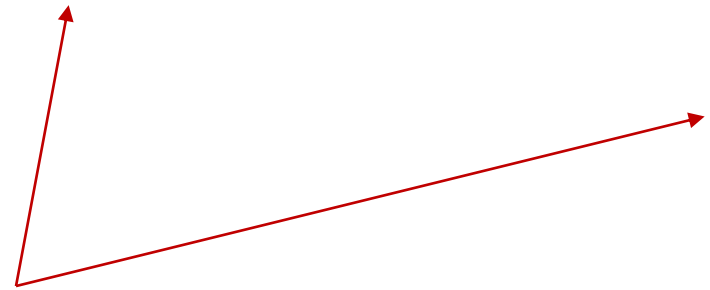
Low-level Loop Representation



Represent Program via Transformations

Loop Nest Representation

```
parallel for xo in range(32):
  C[xo*4:xo*4+4] = f32x4.add(
     A[xo*4:xo*4+4], B[xo*4:xo*4+4])
```



Equivalent to each other

Initial Program + Transformations

```
for x in range(128):
  C[x] = A[x] + B[x]
```

```
xo, xi = split(x, 4)
```

```
parallelize(xo)
```

```
vectorize(xi)
```

Integer Set, Iterator Space and Relations

```
for i, j in grid(16, 16):
  S0: Y[i, j] = 0

for i, j, k in grid(16, 16, 16):
  S1: Y[i, j] += x[i, k] * w[j, x]

for i, j in grid(16, 16):
  S2: Z[i, j] = Y[i, j] + b[j]
```

Integers of iterations

S0: `i in [0, 16), j in [0, 16)`

S1: `i in [0, 16), j in [0, 16), k in [0,16)`

S2: `i in [0, 16), j in [0, 16)`

Partial order constraints of executions

S0[i, j] < S1[i, j, k] < S2[i, j]

Discussion

- What are other possible ways to represent the same program?
- How would these representation variants affect automatic optimizations?

Search Space Construction

Auto Tuning Program Templates

```
for xo, yo, k in grid(sxo?, syo?, 128):
  for xi, yi in grid(sxi?, syi?):
    C[…] += A[…] * B[…]
for x, y in grid(128, 128):
  D[…] = max(C[…], 0)
```

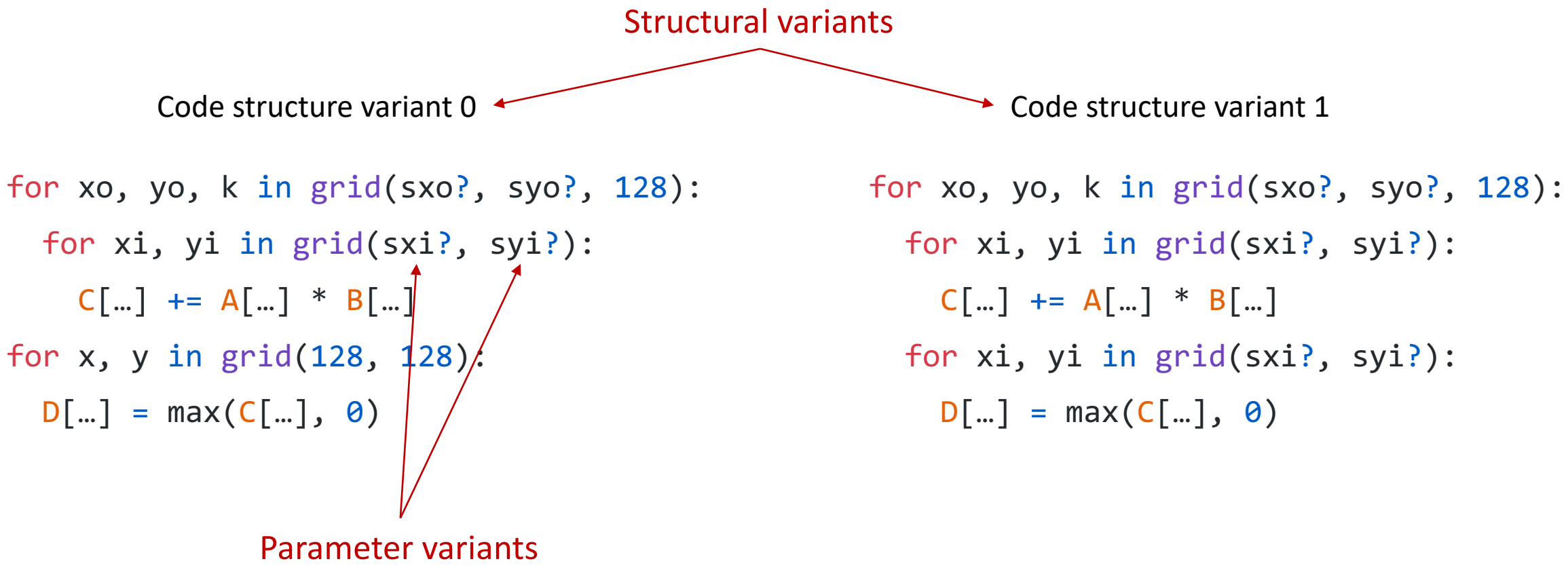
Tunable parameters

Constraints

```
sxo? * sxi? == 128
syo? * syi? == 128
```

Access indices are omitted to simplify the example

Structural Variants vs Parameter Variants



Access indices are omitted to simplify the example

Discussion

- What can be tunable parameters in a program template?
- How to represent structure variants?

Use the Transformation Representation

Init Program + Transformations

```
for x, y, k in grid(128, 128, 128):
  C[…] += A[…] * B[…]
for x, y in grid(128, 128):
  D[…] = max(C[…], 0)
```

```
xo, xi = split(x, sxi?)
```

```
yo, yi = split(y, syi?)
```

```
reorder(xo, yo, k, xi, yi)
```

```
compute_at(D, Dloc?)
```

Computation location of D

Corresponding Program Space

when Dloc? == root

```
for xo, yo, k in grid(sxo?, syo?, 128):
  for xi, yi in grid(sxi?, syi?):
    C[…] += A[…] * B[…]
for x, y in grid(128, 128):
  D[…] = max(C[…], 0)
```

when Dloc? == k

```
for xo, yo, k in grid(sxo?, syo?, 128):
  for xi, yi in grid(sxi?, syi?):
    C[…] += A[…] * B[…]
  for x, y in grid(sxi?, syi?):
    D[…] = max(C[…], 0)
```

Programmatic Search Space Generation by Program Analysis

```
for x, y, k in grid(128, 128, 128):
  C[…] += A[…] * B[…]

for x, y in grid(128, 128):
  D[…] = max(C[…], 0)
```

Reduction with reuse opportunities
try to tile the spatial dimensions

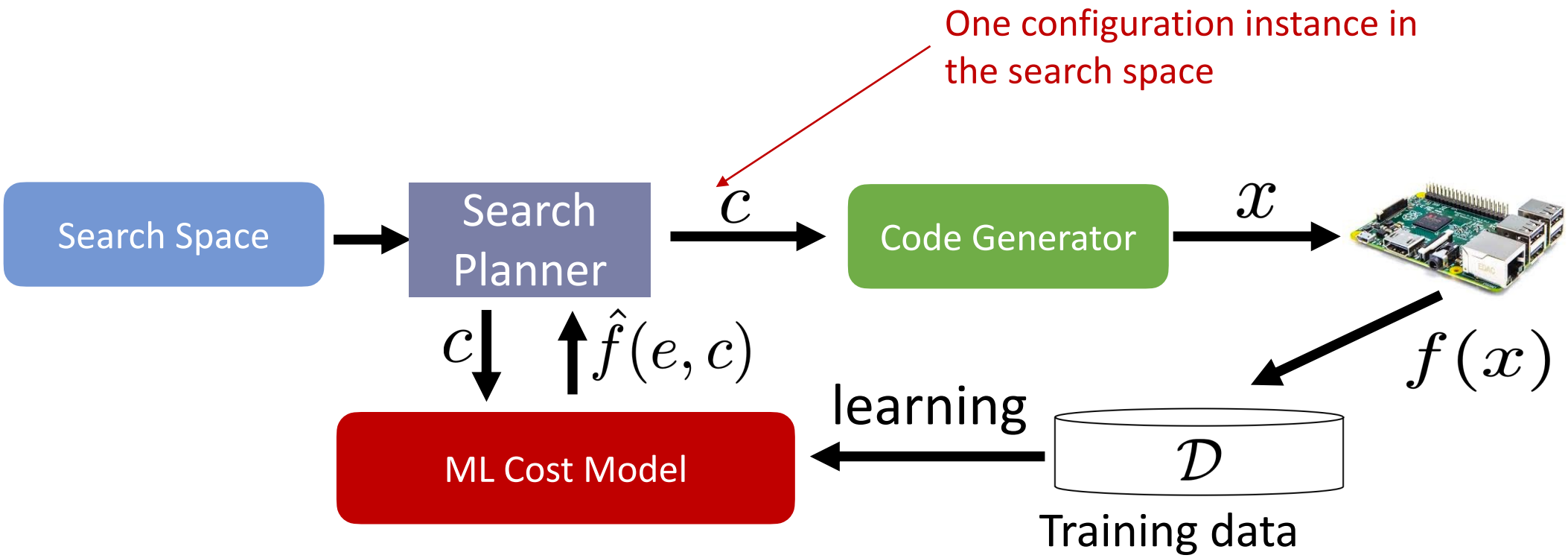
Element-wise operations, consider
fuse to previous loop

Search space represented
by transformations

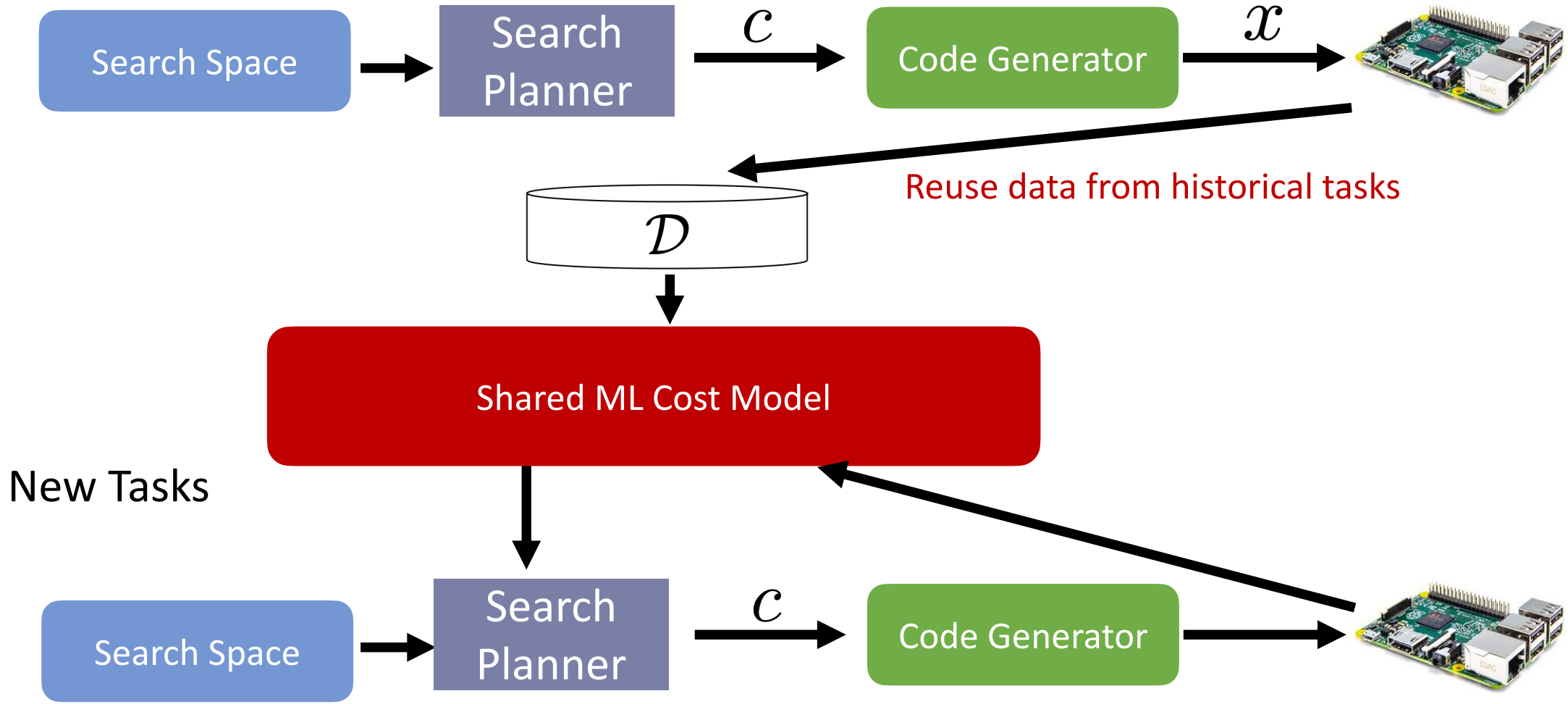
```
xo, xi = split(x, sxi?)
yo, yi = split(y, syi?)
reorder(xo, yo, k, xi, yi)
compute_at(D, Dloc?)
```

Effective Search

Search via Learned Cost Model



Invariant Cost Model



Search Over Parameters

```
for xo, yo, k in grid(sxo?, syo?, 128):
  for xi, yi in grid(sxi?, syi?):
    C[…] += A[…] * B[…]
for x, y in grid(128, 128):
  D[…] = max(C[…], 0)
```

Search Planner

sxi? = 4, syi? = 4

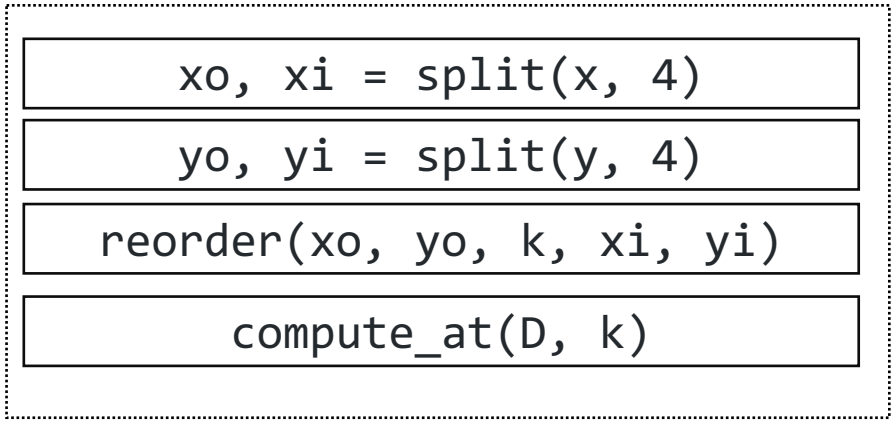
```
for xo, yo, k in grid(32, 32, 128):
  for xi, yi in grid(4, 4):
    C[…] += A[…] * B[…]
for x, y in grid(128, 128):
  D[…] = max(C[…], 0)
```

Search Over Transformations

```
for x, y, k in grid(128, 128, 128):
  C[…] += A[…] * B[…]
for x, y in grid(128, 128):
  D[…] += max(C[…], 0)
```

Search Planner

Evolutionary search
over transformations



Revisit: Direct Representation vs Transformations

Direct Representation

```
parallel for xo in range(32):
  C[xo*4:xo*4+4] = f32x4.add(
     A[xo*4:xo*4+4], B[xo*4:xo*4+4])
```

Initial Program + Transformations

```
for x in range(128):
  C[x] = A[x] + B[x]
```

```
xo, xi = split(x, 4)
```

```
parallelize(xo)
```

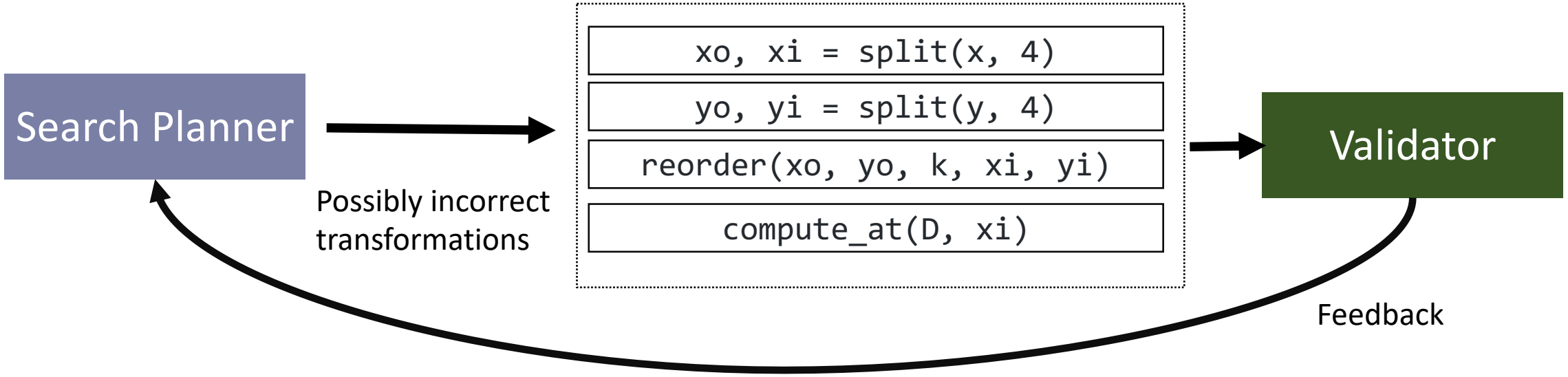
```
vectorize(xi)
```

Discussions

- What are other possible ways to perform search on the direct and transformation-based representation?

- How to handle specialized hardware (GPU and NPUs)

Search and Then Validate



Summary: Elements of an Automated ML Compiler

- Program representation
  - Represent the program/optimization of interest, (e.g. dense tensor linear algebra, data structures)
- Comprehensive structural search space
  - Cover common optimizations
  - Find ways for domain experts to provide input
- Effective search
  - Cost models, transferability
  - Exploration vs exploitation

Still an open research area!

Logistics

Informal mid-term check-in (required)

- Come to one of the office hours to talk about your current progress in the project

- Alternative: send a short email note about your current progress