# Searching for missing library function calls using machine learning

I. A. Yakimov
A. S. Kuznetsov

Institute of Space and Information Technologies
Siberian Federal University

# Issue

- When working with third-party **libraries**, programmers often use ready-made **solutions from Web**.
- Developers might **miss some parts** of these solutions when implementing their own applications.

# Hypothesis

In many cases **workable applications** that use third-party libraries are based on «**patterns**»* which properties can be analyzed by **machine learning**.

* not the *design patterns*, but patterns of function call sequences

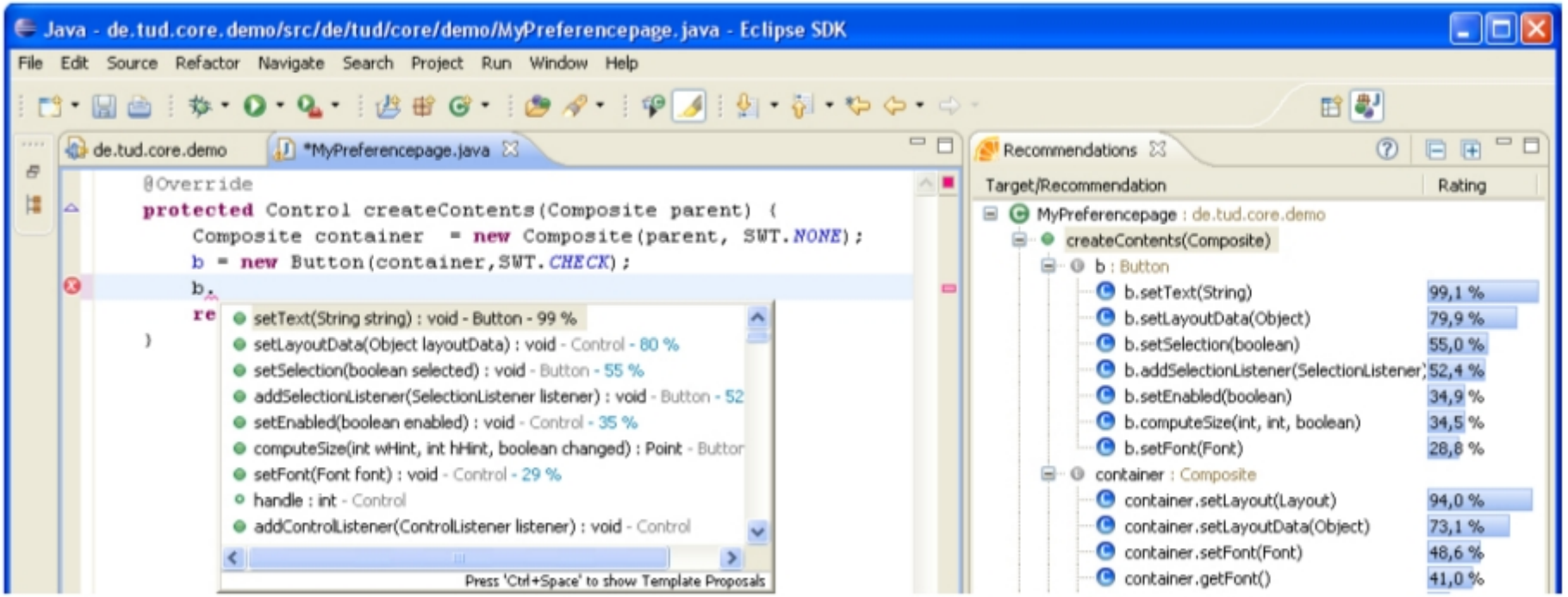
# Recommender systems for autocompletion

- **Recommender systems** rely on probabilistic models and machine learing algorithms to improve **autocompletion**:
  - Autocompletion for **method calls**
  - Filling the **labeled lines**

# Autocompletion for method calls

- In case of context-dependent probabilistic autocompletion models, the history of **object** is analyzed and then the probability distribution for the **next call** is given.

# Autocompletion for method calls




Bruch M., Monperrus M., Mezini M..
Learning from examples to improve code completion systems.
2009, DOI: 10.1145/1595696.1595728

# Filling the labeled lines

- The **user labels** (empty) **lines** in which the missed code is supposed to be located.
- For the labeled line, the system then synthesizes the **most likely piece of code**.

# Filling the labeled lines

```
void exampleMediaRecorder() throws IOException {
 Camera camera = Camera.open();
 camera.setDisplayOrientation(90);
   ?  // (H1)
 SurfaceHolder holder = getHolder();
 holder.addCallback(this);
 holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
 MediaRecorder rec = new MediaRecorder();
   ?  // (H2)
 rec.setAudioSource(MediaRecorder.AudioSource.MIC);
 rec.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
 rec.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
   ? {rec}  // (H3)
 rec.setOutputFile("file.mp4");
 rec.setPreviewDisplay(holder.getSurface());
 rec.setOrientationHint(90);
 rec.prepare();
   ? {rec}  // (H4)
}
```

Raychev V., Vechev M., Yahav E.. Code completion with statistical language models. Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation. 2014, pp. 419-428. DOI: 10.1145/2594291.2594321

# Filling the labeld lines

```
void exampleMediaRecorder() throws IOException {
 Camera camera = Camera.open();
 camera.setDisplayOrientation(90);
 camera.unlock();
 SurfaceHolder holder = getHolder();
 holder.addCallback(this);
 holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
 rec = new MediaRecorder();
 rec.setCamera(camera);
 rec.setAudioSource(MediaRecorder.AudioSource.MIC);
 rec.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
 rec.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
 rec.setAudioEncoder(1);
 rec.setVideoEncoder(3);
 rec.setOutputFile("file.mp4");
 rec.setPreviewDisplay(holder.getSurface());
 rec.setOrientationHint(90);
 rec.prepare();
 rec.start();
}
```

Raychev V., Vechev M., Yahav E.. Code completion with statistical language models. Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation. 2014, pp. 419-428. DOI: 10.1145/2594291.2594321

# Features of the system

- Thus, the currently existing recommender systems for autocompletion ask the **user** to **explicitly** or **implicitly** **indicate** the specific sections of the code in which the autocompletion is **intended**.

# Features of the system

- Our work focuses mainly on the ability of the **system** to **independently determine** the presence (or absence) of missed calls **without** active **user** participation.

# Task description

- Let F be a **user procedure** containing some **library function calls**.
- Using a predefined algorithm, a **sequence of function calls** $w = f_1, f_2, \ldots, f_n$ is extracted from F.
- Moreover, for **this sequence**:
  - the only **one function** needed to complete a particular template **is missed**
  - or there are **no missed functions**

# Task description

- The model must satisfy the following **properties**:
  - If **there is a missed function call** that completes some pattern, the system must reveal this fact and determine name of the missed function.
  - If **there are no missed calls**, and the sequence is a complete pattern, the system must also reveal this fact.
  - **No information** about missed function calls should be provided **by user** (implicitly or explicitly).

# Example — missed glPushMatrix

| Source code | Result |
|---|---|

```
void display(void) {
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0, 1.0, 1.0);
// glPushMatrix(); - missing call
glutWireSphere(1.0, 20, 16);
glRotatef(year, 0.0, 1.0, 0.0);
glTranslatef(2.0, 0.0, 0.0);
glRotatef(day, 0.0, 1.0, 0.0);
glutWireSphere(0.2, 10, 8);
glPopMatrix();
glutSwapBuffers();
}
```

| | |
|---|---|
| **glPushMatrix** | **98,84%** |
| glRotatef | 0,22% |
| __none__ | 0,10% |
| glFlush | 0,09% |
| gluPerspective | 0,09% |
| glFrustum | 0,08% |
| glScalef | 0,07% |
| glEnable | 0,07% |
| gluOrtho2D | 0,06% |
| glNewList | 0,06% |

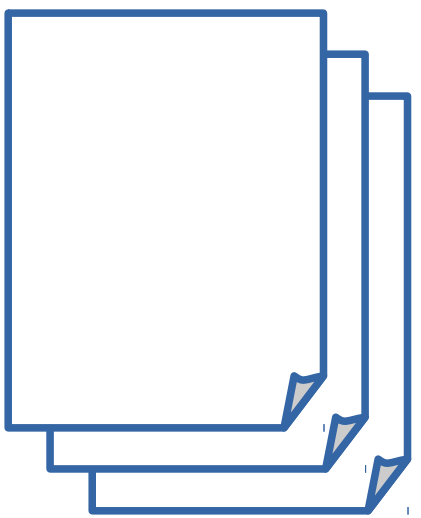
# Example — no missed calls

| Source code | Result |
|---|---|

```
 // No missing calls
static void Reshape(...){
glViewport(0, 0, width, height);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(-2.0, 2.0, -2.0, 2.0, ...);
gluLookAt(7.0, 4.5, 4.0, ...);
glMatrixMode(GL_MODELVIEW);
}
```

| | |
|---|---|
| **__none__** | **55,27%** |
| glLoadIdentity | 18,47% |
| glOrtho | 6,72% |
| gluOrtho2D | 4,67% |
| glEnable | 3,04% |
| glTranslatef | 1,64% |
| gluPerspective | 1,25% |
| glScalef | 1,22% |
| glMatrixMode | 0,94% |
| glDisable | 0,76% |

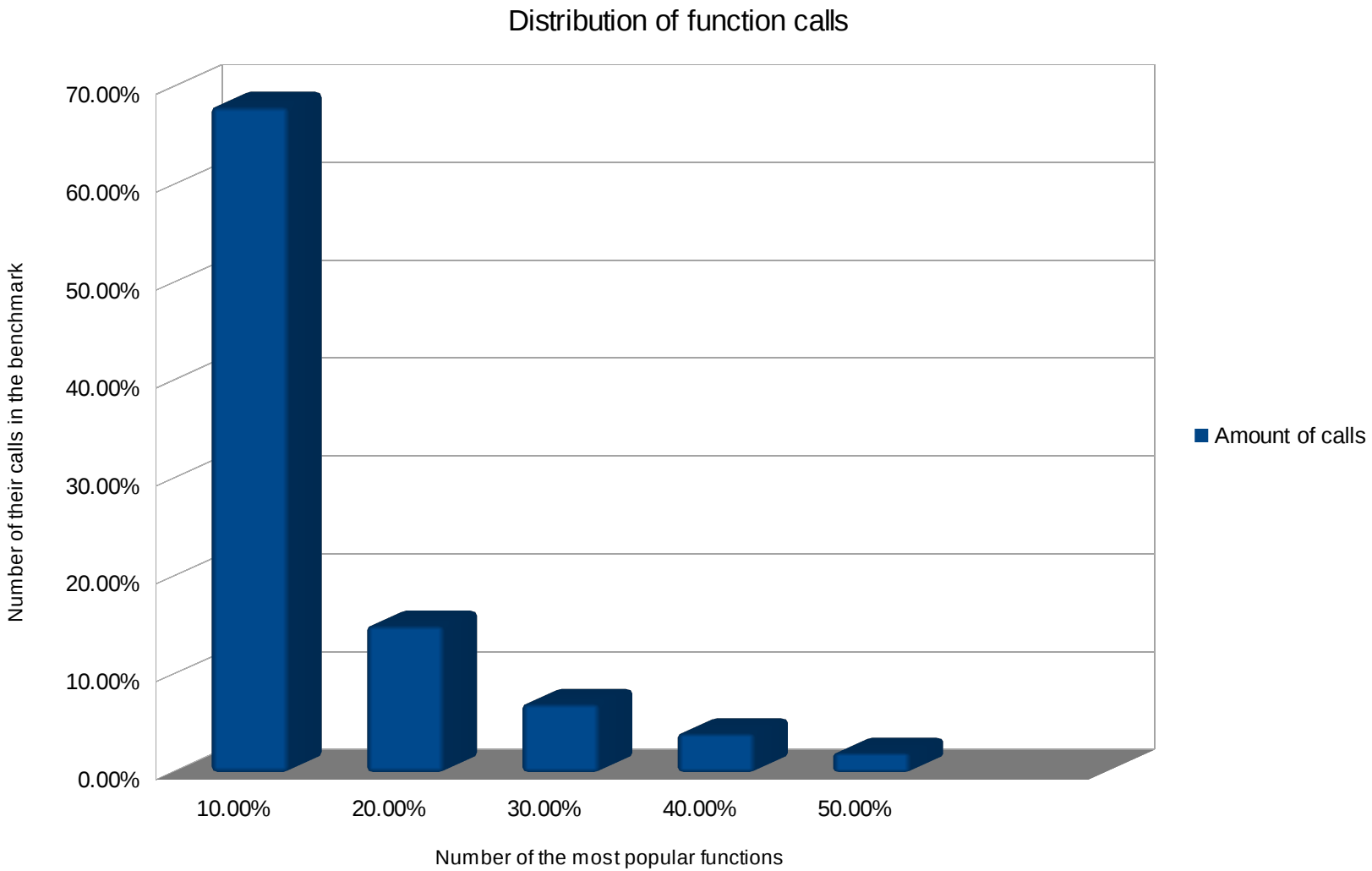
# Benchmark



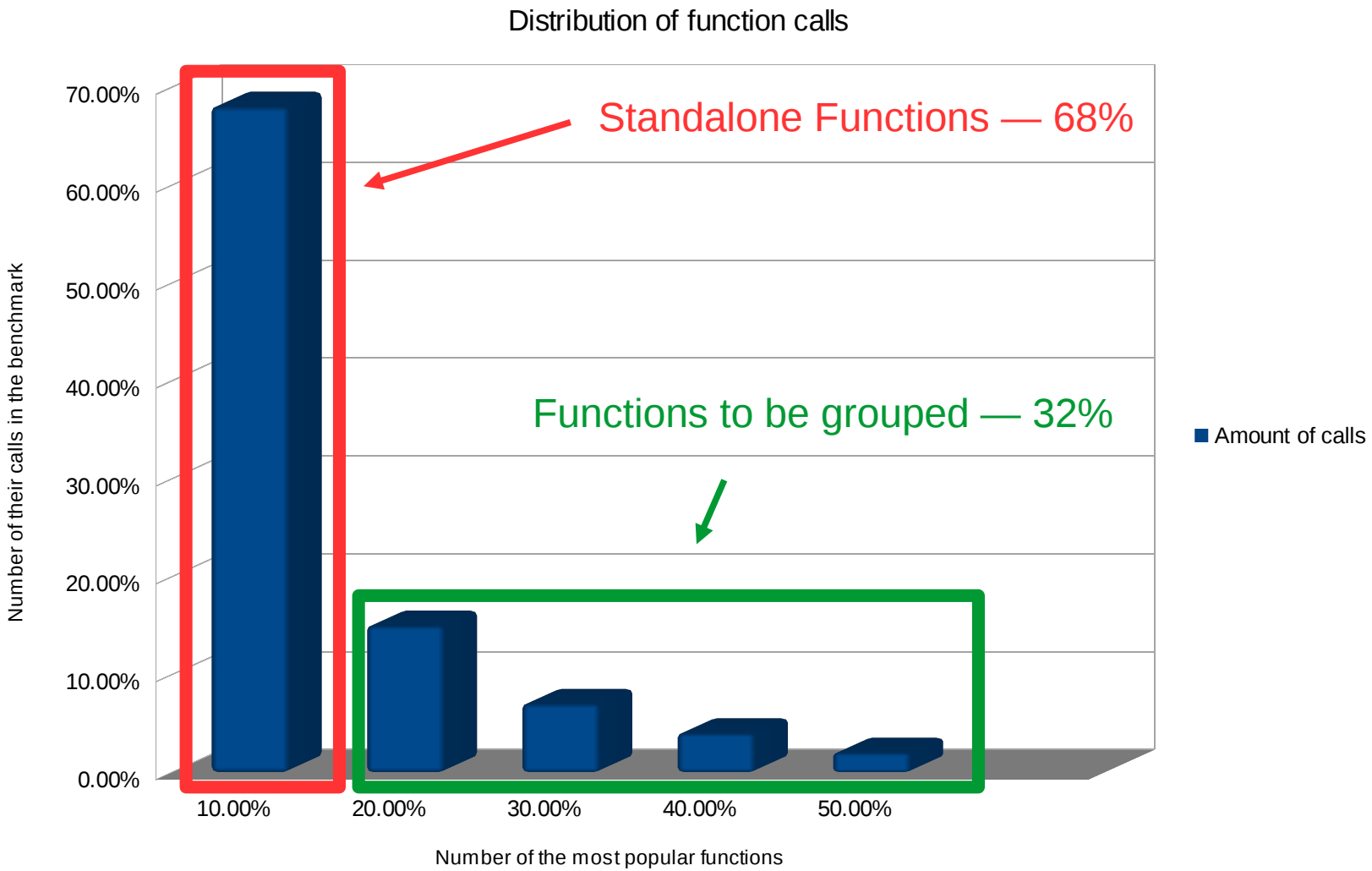
**329 different** library
function calls
**15182** calls **in total**

**214** open-source **examples**
for the OpenGL library

# Problem — nonuniform distribution of the function calls



# Solution — grouping the functions



# Standalone functions — first 35 labels

**glEnable - 734 calls**

glVertex3 - 571

glutPostRedisplay - 539

glMatrixMode - 490

glTranslate - 489

glPushMatrix - 452

glPopMatrix - 447

glutAddMenuEntry - 417

glDisable - 369

glMaterial - 350

glCallList - 348

glVertex2 - 318

glRotate - 317

glLoadIdentity - 313

glBegin - 311

glEnd - 305

glTexCoord2 - 280

glNormal3 - 267

glLight - 242

glColor3 - 241

glClear - 236

glTexParameter - 227

glEndList – 215

glutInitDisplayMode - 203

glutDisplayFunc - 199

glCreateWindow - 194

glutMainLoop - 188

glutInit - 188

glViewport - 183

glutInitWindowSize - 156

glutKeyboardFunc - 149

glReshapeFunc - 148

glFlush - 139

glClearColor - 139

glutSwapBuffers - 128

# Grouped functions — another 15 labels

**_misc ( 71 funcs ) - 786 calls**

_glu ( 41 ) - 649

_draw ( 27 ) - 502

_textures ( 24 ) - 381

_glut_wind ( 27 ) - 299

_disp ( 5 ) - 262

_menu ( 5 ) - 250

_trans ( 5 ) - 239

_buff_eval_accum ( 26 ) - 193

_glut_fgapientry ( 21 ) - 174

_lightning ( 5 ) - 165

_callback ( 2 ) - 155

_stenciling ( 4 ) - 122

_raster ( 8 ) - 116

_geometry ( 11 ) - 101

# __none__ - special label

**__none__** is used when **no function calls** are **missed**

# Tracing the sources

```
; Function Attrs: nounwind uwtable
define i32 @main(i32 %argc, i8** %argv) #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  %3 = alloca i8**, align 8
  store i32 0, i32* %1
  store i32 %argc, i32* %2, align 4
  store i8** %argv, i8*** %3, align 8
  %4 = load i8*** %3, align 8
  call void @glutInit(i32* %2, i8** %4)
  call void @glutInitDisplayMode(i32 18)
  %5 = call i32 @glutCreateWindow(i8* getelementptr
   inbounds ([20 x i8]* @.str, i32 0, i32 0))
  call void @glutDisplayFunc(void ()* @display)
  call void @init()
  call void @glutMainLoop()
  ret i32 0
}
```

glutInit

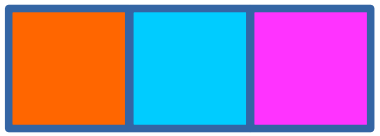
glutInitDisplayMode

GlutCreateWindow

GlutDisplayFunc

glutMainLoop

# Mapping examples to the traces



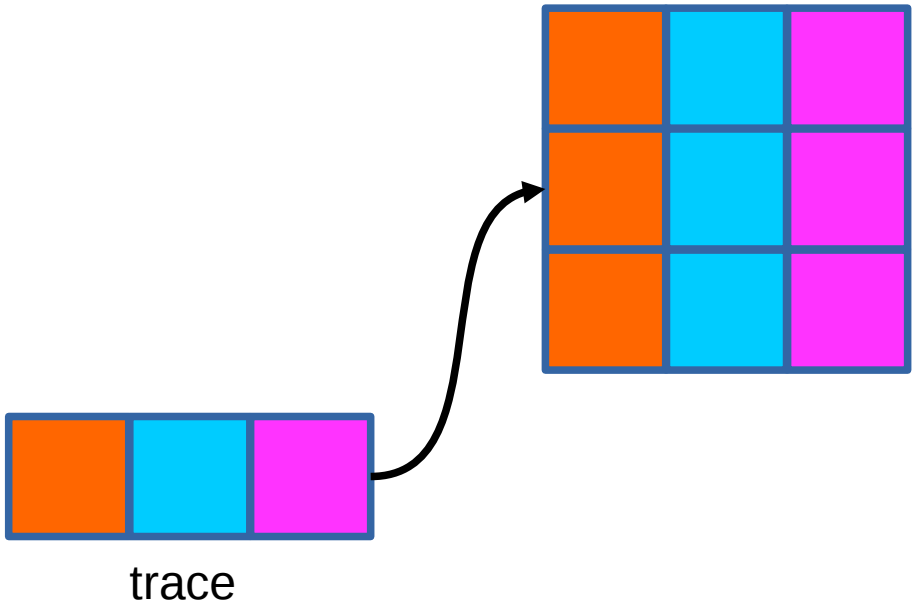
trace

glBegin

glVertex3

glEnd

# Mapping examples to the traces

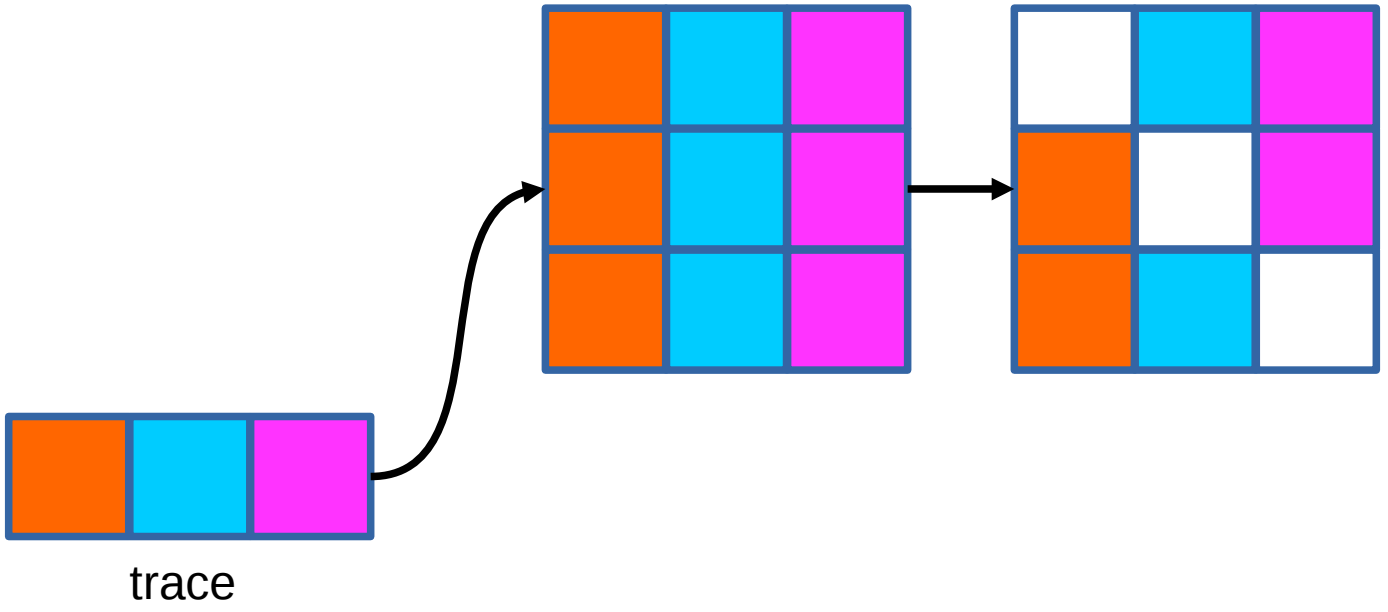



glBegin

glVertex3

glEnd

# Mapping examples to the traces

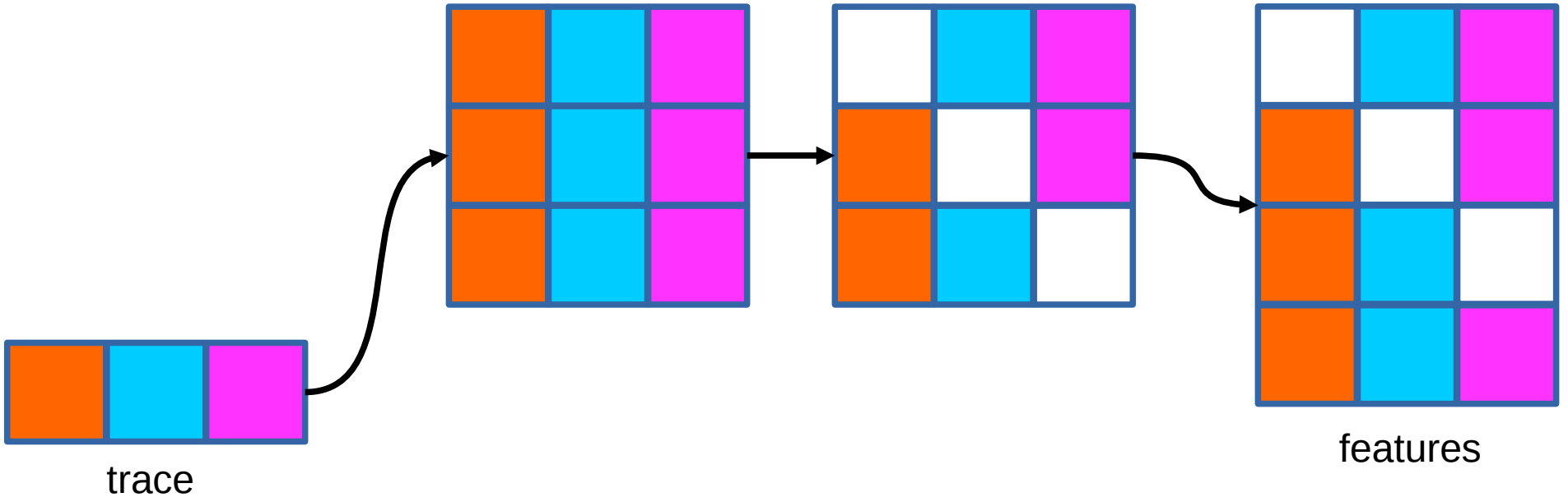



glBegin

glVertex3

glEnd

# Mapping examples to the traces

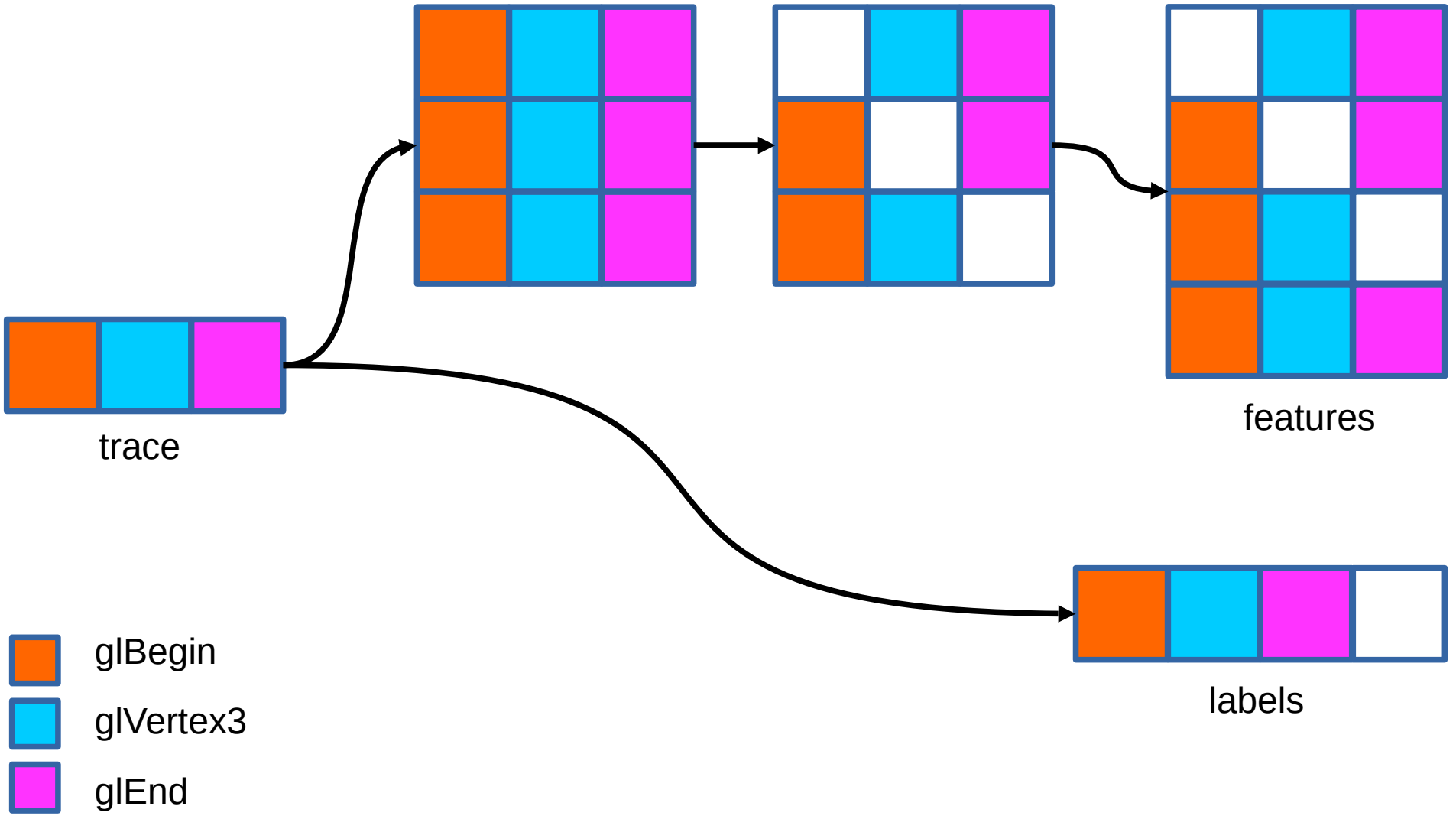



glBegin

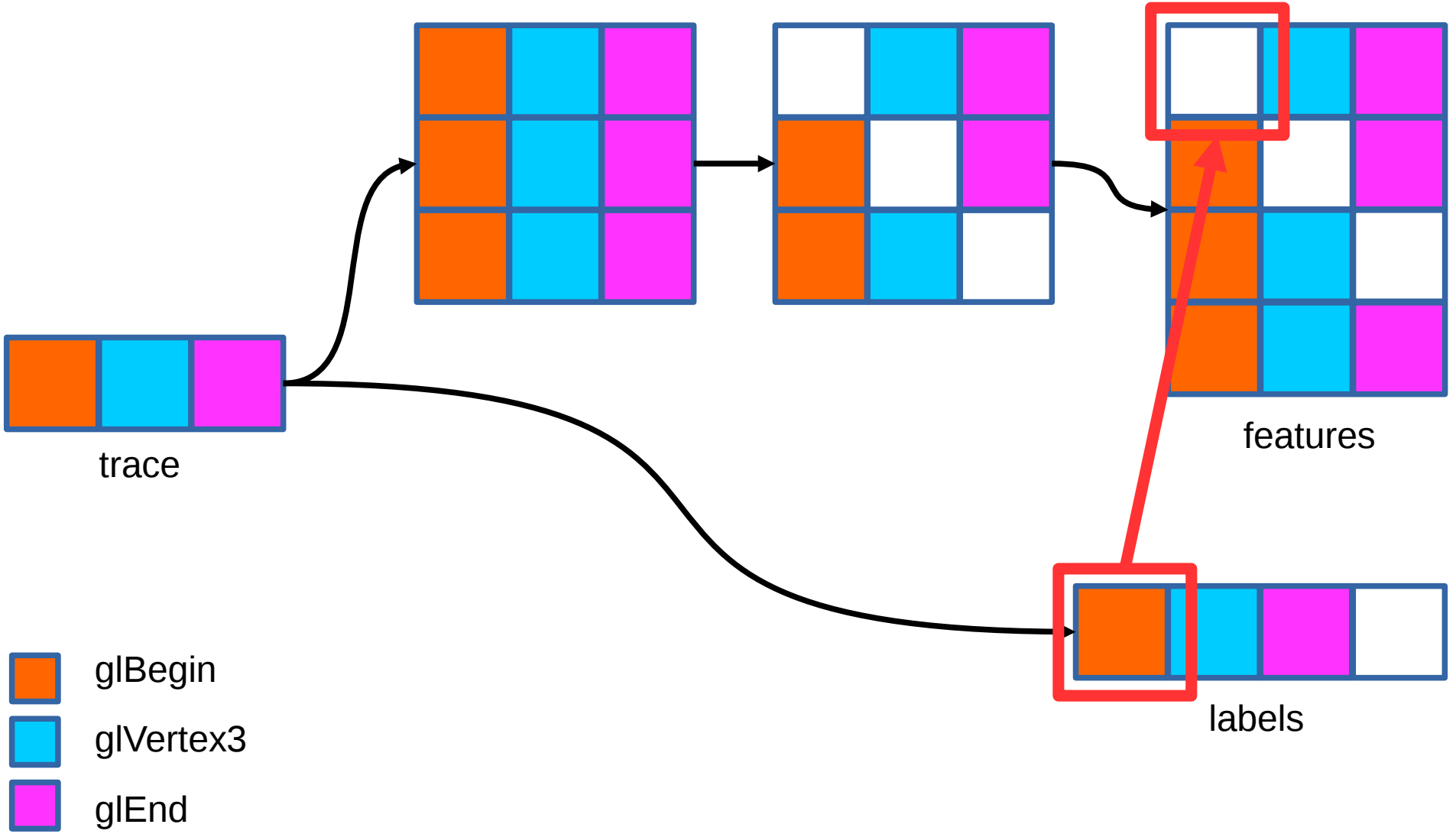
glVertex3

glEnd

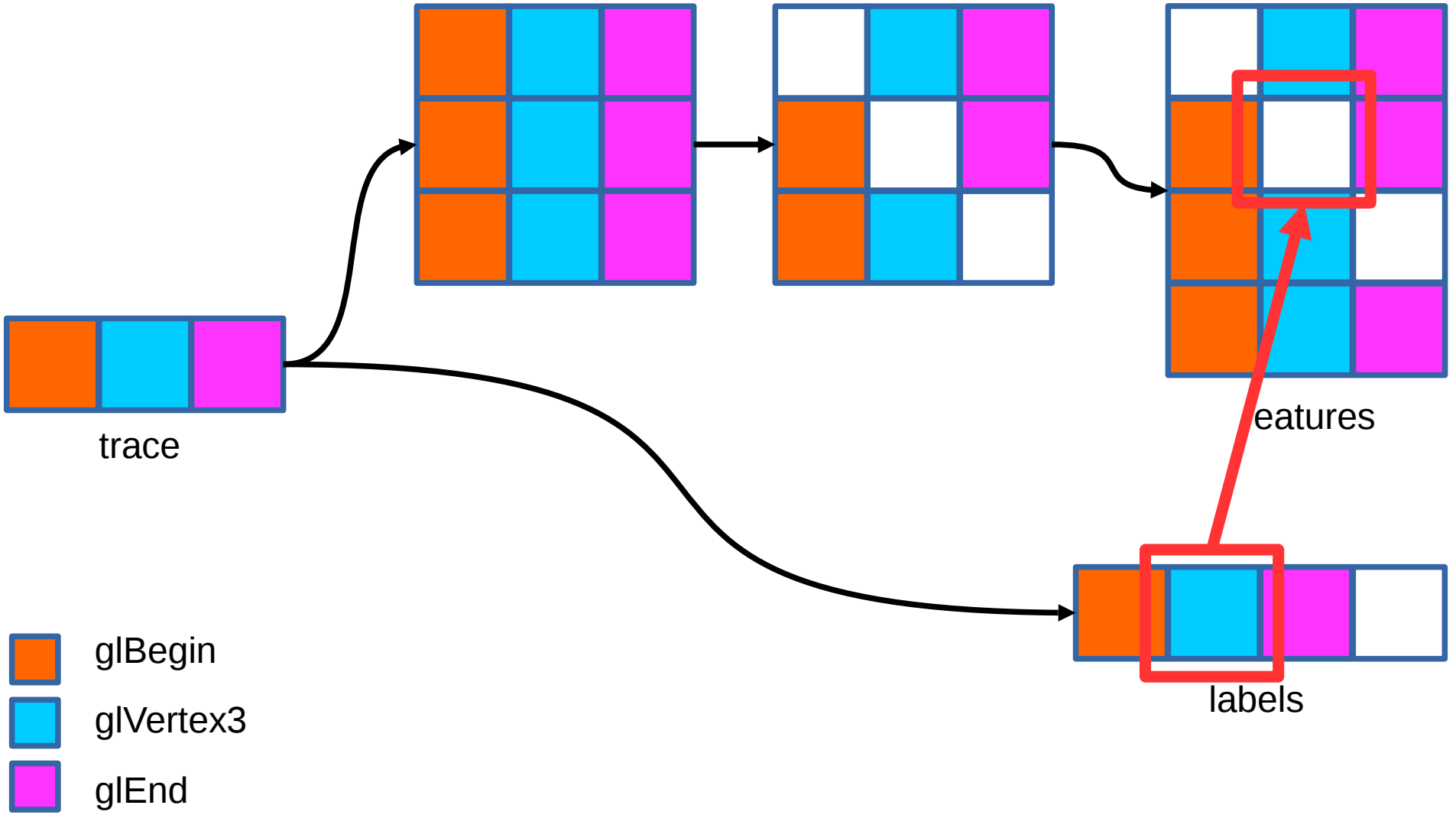
# Mapping examples to the traces



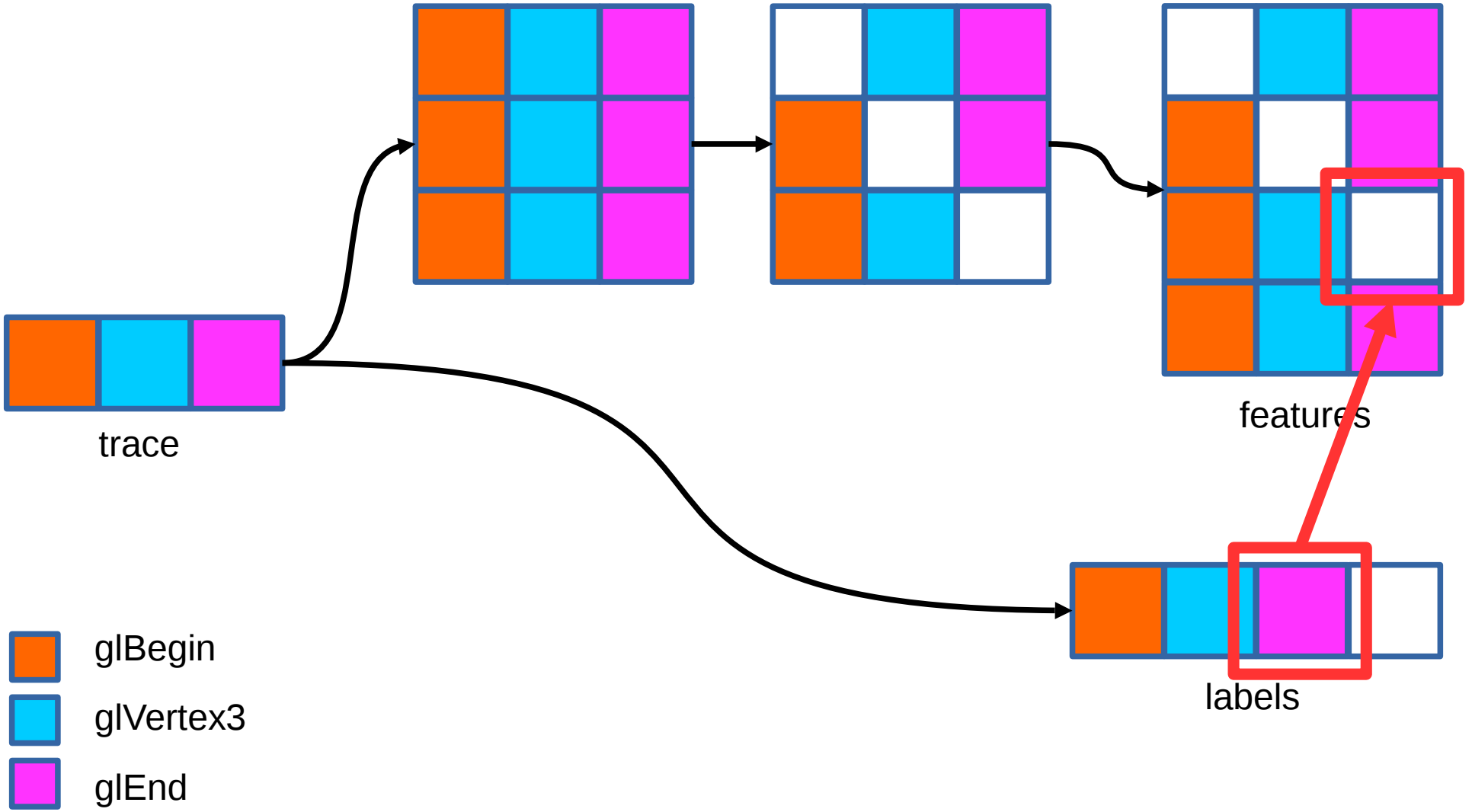
# Mapping examples to the traces



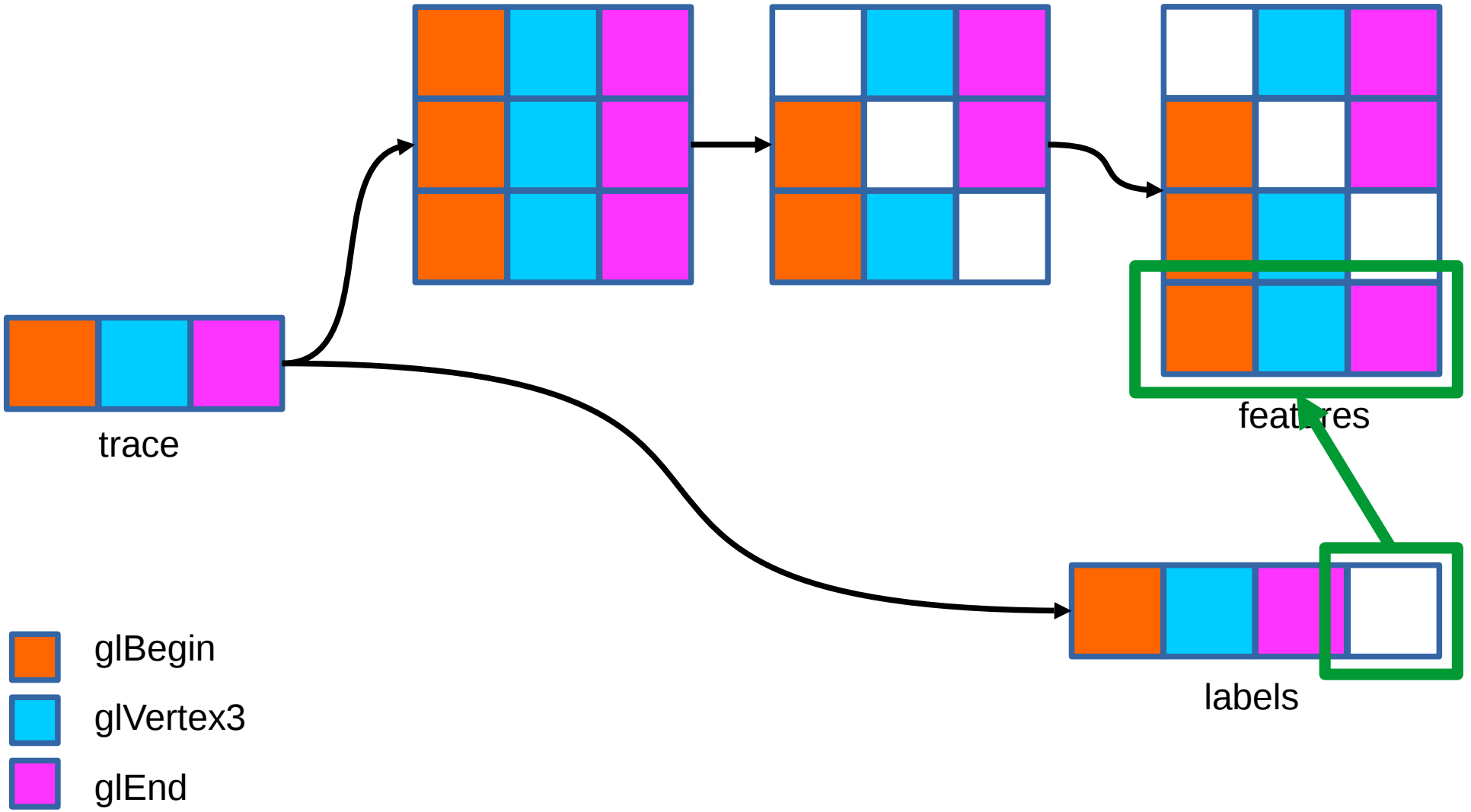
# Mapping examples to the traces



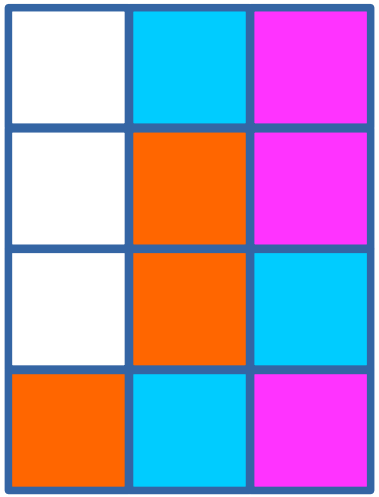
# Mapping examples to the traces



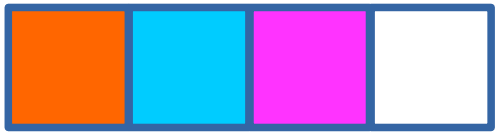
# Mapping examples to the traces



# Mapping examples to the traces — final result



Feature Matrix



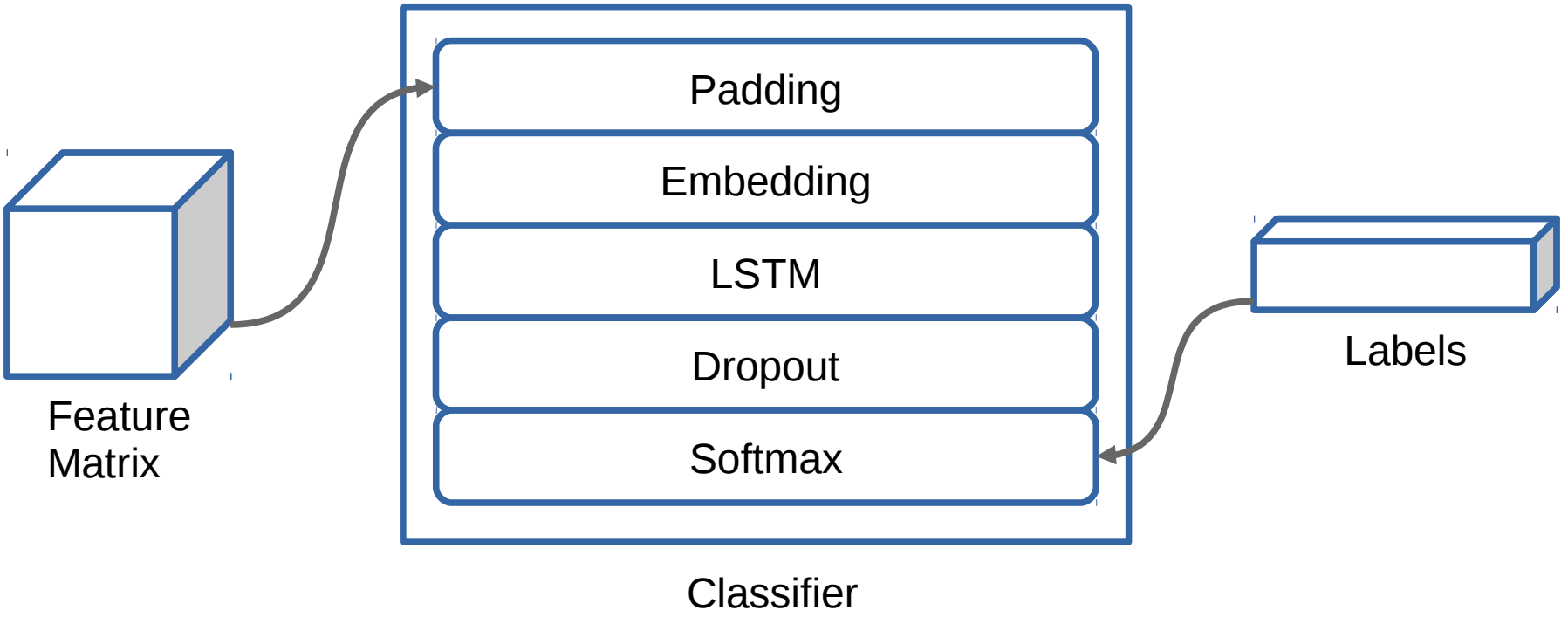
Vector of Labels

glBegin

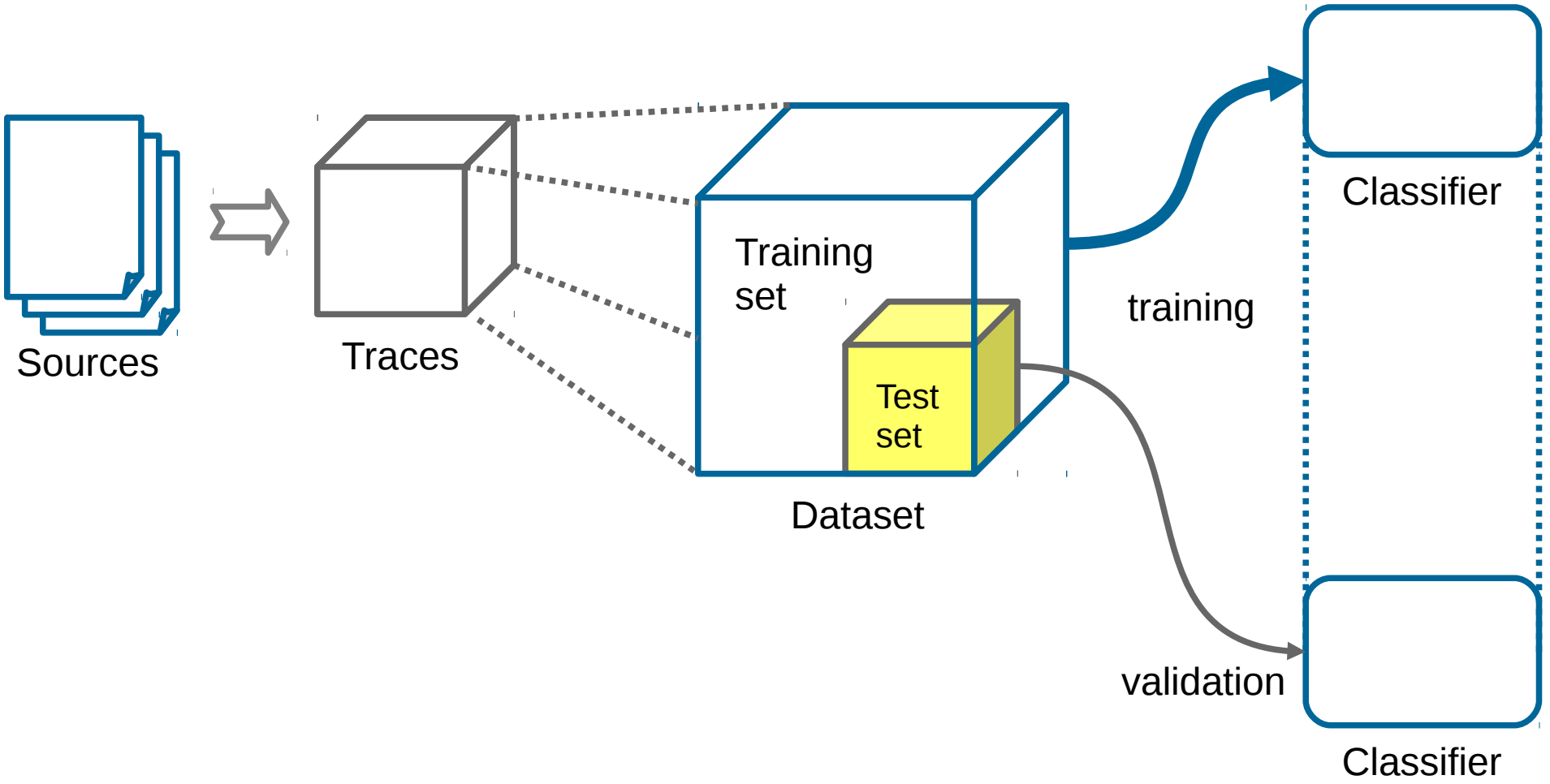
glVertex3

glEnd

# Architecture of the classifier



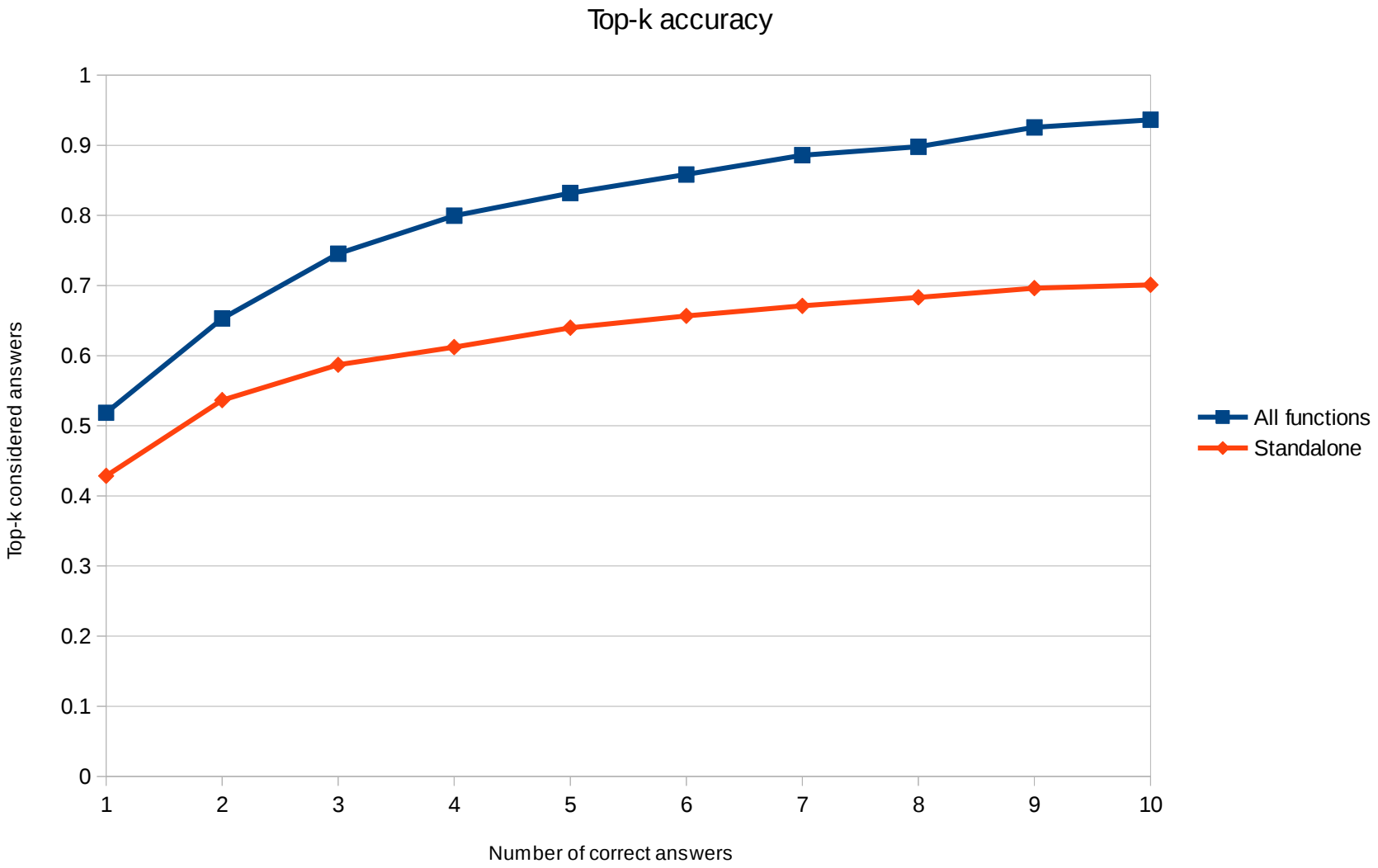
# All the steps together



# Experimental settings

- **Dataset** of **6623 examples** ≈ 50% of possible
- Each example **from 5 to 25 function calls**
- **Test set** = **833 ≈ 12%** of dataset
- Number of **epochs** = **100**
- **Accuracy** measurement: **top-k**

# Results


Top-k accuracy
Top-k considered answers
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6
7
8
9
10
Number of correct answers
All functions
Standalone

# Limitations

- For **some libraries**, function calls **do not form reliable patterns**
- The **system can** only **identify names** of the missed functions*, **not their positons**
- Only **traces from 5 to 25 calls** are **included**

*or the fact that *no function was missed*

# Conclusion

- The system for **missing calls detection** has been created
- It is already might be **useful** for the educational purposes
- In case of **OpenGL** examples **accuracy = 80%** for the top-4 metric
- **Further** experimental and theoretical **studies** are **needed**

# Thank you for attention

Any questions?