NYU TANDON SCHOOL OF ENGINEERING

Large Language Models for Software Security

Prospects and ~~Pitfalls~~ (10 minute version)

Brendan Dolan-Gavitt

CENTER FOR CYBER SECURITY

Surprising Progress in Code Models

Before 2021

- 2015: Karpathy's Char-RNN, generating Linux kernel code
- 2019: GPT-2 "accidentally" learns some PHP and JavaScript

```c
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
```

Char-RNN; Karpathy, 2015

```php
$app = new App ();

// All GET requests that come to add_register() will be sent to this service.

$api = $app -> include(' ' );

$api -> register( new DbAppAndFNAAppRegistrationService ());

// Define any services to register. We will override any present in the external
// DB have the class of .DAO .

$service = new AppAndFNAAppService ( $app , [

array ( ' host ' => ' localhost ' )

]);
```

GPT-2; Karpathy, 2015

Surprising Progress in Code Models

June 2021 - Present

- **2021: OpenAI Codex** - a large GPT-3-based model fine-tuned on code
  - Released commercially as a code completion tool: **GitHub Copilot**
- **2022: DeepMind AlphaCode** - Transformer (encoder/decoder)
  - Reaches human-level (top 54%) performance in an online code competition (Codeforces)
- Both systems treat source code as plain text, “predict next token”
- Trained on **large volumes of code** (e.g. all of GitHub)

NYU

Github Copilot

reddit PROGRAMMING comments other discussions (18)

2660

GitHub Copilot · Your AI pair programmer (copilot.github.com)

submitted 2 months ago by violinclipper 4 & 14 more

581 comments share save hide give award report crosspost

The Verge

GitHub and OpenAI launch an AI Copilot tool that generates its own code

GitHub and OpenAI have launched a technical preview of a new AI tool called Copilot, which lives inside the Visual Studio Code editor and ...

Jun 29, 2021

TechTalks HOME BLOG TIPS & TRICKS WHAT IS INT

Home › Blog › What OpenAI and GitHub's "AI pair programmer" means for the softwa

Blog

What OpenAI and GitHub's "AI pair programmer" means for the software industry

By **Ben Dickson** - July 5, 2021

Y **Hacker News** new | threads | past | comments | ask | show | jobs | submit

▲ GitHub Copilot (copilot.github.com)

2905 points by todsacerdoti 75 days ago | hide | past | favorite | 1272 comments

June 29, 2021 — Open Source, Product

Introducing GitHub Copilot: your AI pair programmer

Nat Friedman

InfoWorld UNITED STATES INSIDER

Developers react to GitHub Copilot

The Microsoft subsidiary has been working with OpenAI to build an AI tool that helps developers write code by making automated suggestions. Here's what the early users make of it.

VB VentureBeat

GitHub launches Copilot to power pair programming... AI

Copilot, as the new GitHub tool is called, uses contextual cues to suggest new code, with users able to flip through alternatives if they ...

Jun 29, 2021

Copilot Writes Vulnerable Code

File Edit Selection View Go Run Terminal Help

example_copilot.py ●

other > example_copilot.py > ...

```python
from flask import Flask, request, redirect
import MySQLdb

app = Flask(__name__)

@app.route('/login', methods=['POST'])
def login():
    user = request.form['username']
    passwd = request.form['password']

    #check if username and password combo exists in the database
    db = MySQLdb.connect(host="localhost", user="root", passwd="", db="test")
    cursor = db.cursor()
    Next (Alt+])  Previous (Alt+[)  Accept (Tab)  Open Copilot (Ctrl+Enter)
    sql = "SELECT * FROM users WHERE username='%s' AND password='%s'" % (user, passwd)
```

SQL Injection

Asleep at the Keyboard

- Examined 18 different vulnerability classes (CWEs) and 89 scenarios, used Copilot to generate 1,689 total programs
    - **40% of generated programs were vulnerable**
    - (More details in our S&P 2022 paper presented on Monday)
- **Open Problem: how can we fix this?**
    - Fine-tuning to decrease probability of generating vulnerable code?
    - Some kind of verification or validation?

Fixing Vulnerabilities with LLMs

- Basic idea: use Codex et al. as a code generator to replace vulnerable code
- Use **prompt engineering** to guide model toward generating fixed versions
- Use **functional** and **security** oracles to check if generated code fixes the vuln without breaking the program ⚠️
- Preliminary evaluation: across 7 different code models, could repair*:
  - 100% of our own synthetically generated vulnerabilities
  - 67% of historical vulnerabilities in our dataset

Repair Prompt

```
1  /* Each tile contains only the data for a single plane
2   * arranged in scanlines of tw * bytes_per_sample bytes.
3   */
4  for (row = 0; row < imagelength; row += tl)
5  {
6   nrow = (row + tl > imagelength) ? imagelength - row : tl;
7   for (col = 0; col < imagewidth; col += tw)
8   {
9  /* BUG: stack buffer overflow
10  * for (s = 0; s < spp; s++)
11  *  { // Read each plane of a tile set into srcbuffs[s]
12  * tbytes = TIFFReadTile(in, srcbuffs[s], col, row, 0, s);
13  * FIXED:
14  */
15  for
```

(b) Prompt constructed according to Fig. 11 (shortened for brevity). The red highlighted line 10 is the original faulty line indicated by ASAN/the oracle. The template includes lines 11 and 12 (highlighted in grey) to encourage the LLMs to regenerate the safe code so the patch can be matched safely.

Successful Repair

libtiff CVE-2016-5321

```
1  /* Each tile contains only the data for a single plane
2   * arranged in scanlines of tw * bytes_per_sample bytes.
3   */
4  for (row = 0; row < imagelength; row += tl)
5  {
6   nrow = (row + tl > imagelength) ? imagelength - row : tl;
7   for (col = 0; col < imagewidth; col += tw)
8   {
9    for (s = 0; (s < spp) && (s < MAX_SAMPLES); s++)
10   {
11    tbytes = TIFFReadTile(in, srcbuffs[s], col, row, 0, s);
```

(d) The repaired program once reassembled with the LLM patched line 11 highlighted in yellow. This generated patch is semantically equivalent with the real-world human patch used to repair this bug.

Inadequate Oracles

libtiff CVE-2016-3623

- The language model fixed the vulnerability... by removing the problematic options!
- Developer tests are **weak proxies** for program functionality
- **Open problem: how can we strengthen these proxies?**
    - Can we get LLMs to write better functional tests as well?

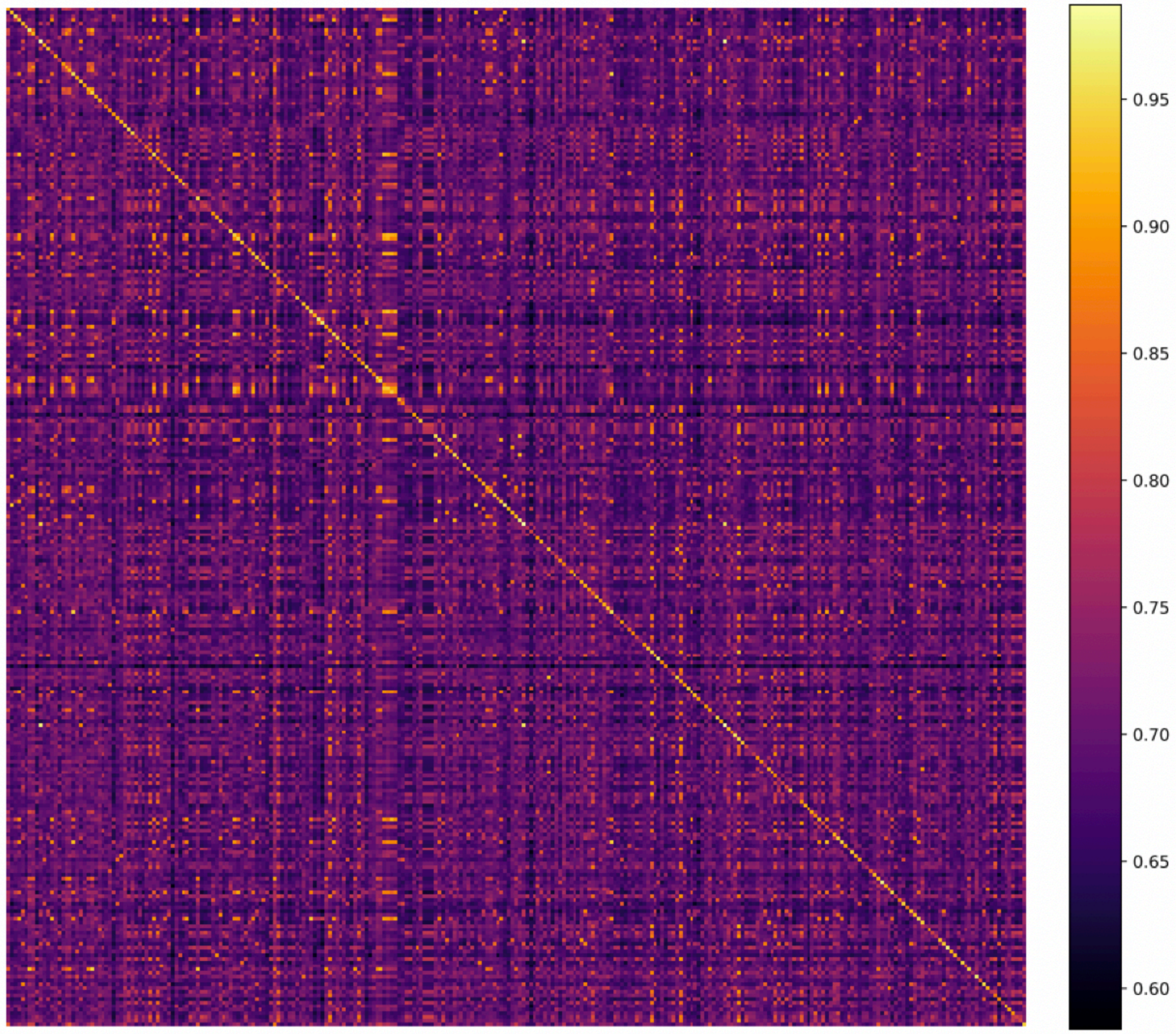
```
--- a/rgb2ycbcr.c
+++ b/rgb2ycbcr.c
@@ -94,11 +94,7 @@
          usage(-1);
      break;
    case 'h':
-     horizSubSampling = atoi(optarg);
-     break;
-   case 'v':
-     vertSubSampling = atoi(optarg);
-     break;
+     usage(-1);
    case 'r':
      rowsperstrip = atoi(optarg);
      break;
```

Patch generated by GPT-CSRC 774M model

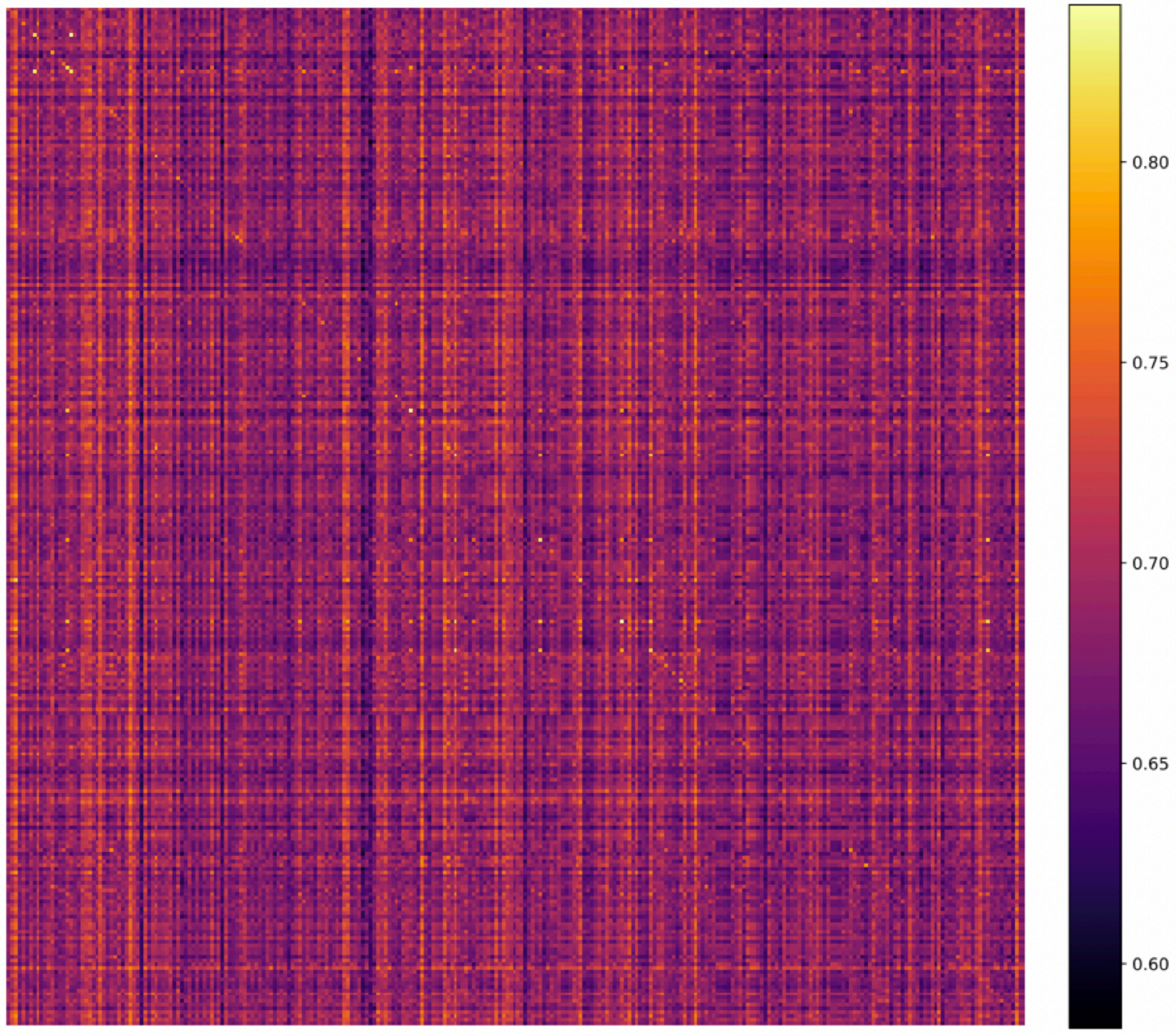
Reverse Engineering with LLMs

- For normal source code, Codex does a reasonable job of **summarizing** code in natural language
- Can we use this ability on **decompiled** code to help automate RE?
- Preliminary result: **mostly no**
  - Decompiled code is too dissimilar to original source code
  - Eval using true/false Q&A format: **136,260** questions posed, Codex answered **72,754** correctly

Embedding Similarity



(a) Confusion matrix for `ls` with debug information.



(b) Confusion matrix for `ls` with debug symbols stripped.

And Beyond...

🔥 **Hot take**: large language models are **vastly underused** in software security right now

- An embarrassment of data:
  - Vast amounts of training data (code)
  - Easy to create parallel corpora (e.g. using compilers & debug info)
  - Can *automatically* extract **semantic** information
- What could we do by just scaling up?
  - “Industrial” LLMs are **~1000x larger** than what we use in software security

Possible Fun Problems

Add your own here!

- Decompilation
- Making fuzzing more effective
- Reverse engineering data types
- Recursively summarizing binaries
- Bug-finding
- Exploit generation