Computer security in the age of large language models

Nicholas Carlini
Google DeepMind

A talk in two parts:

ML for Security

Security of ML

A talk in two parts:

my work

ML for Security

Security of ML

A talk in two parts:

this half

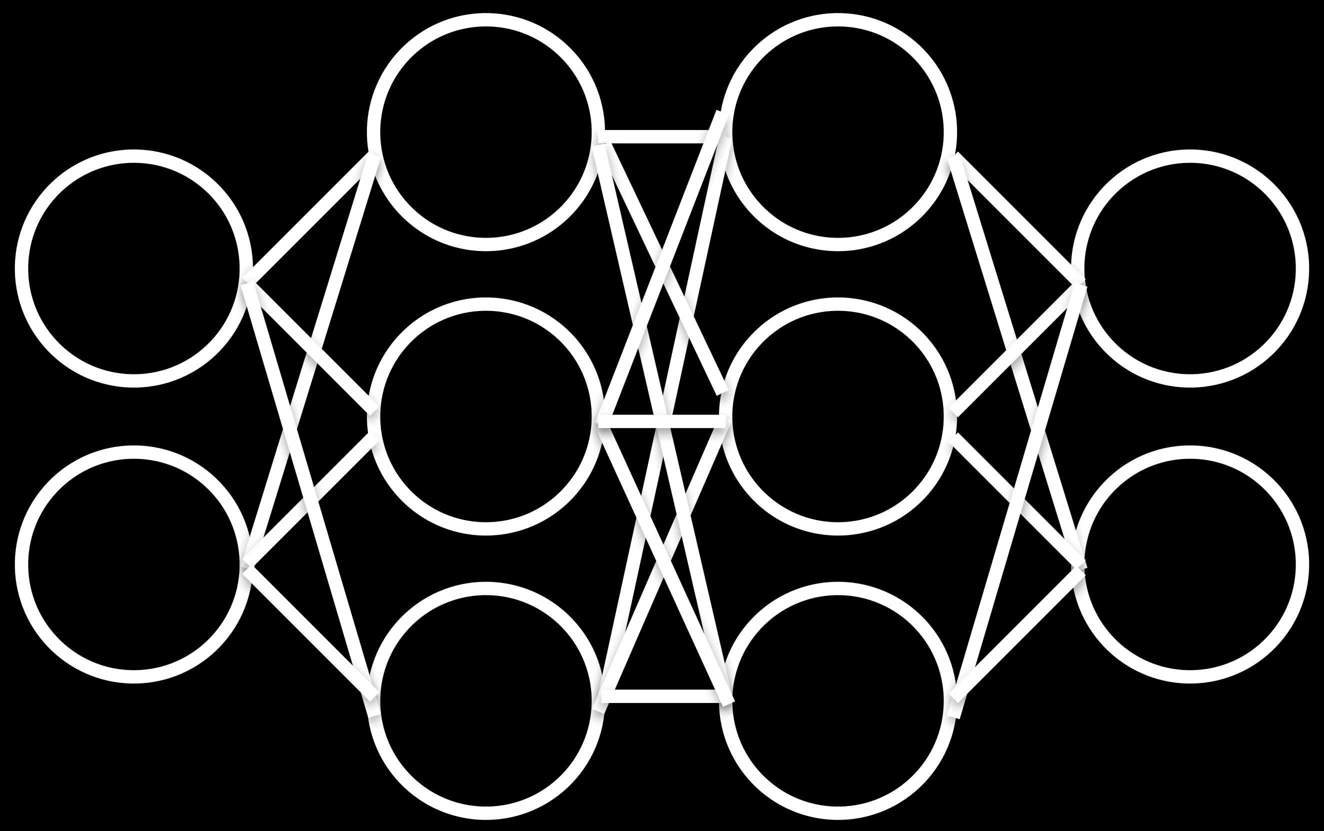
ML for Security

Security of ML

Act I:
Background

Language Models

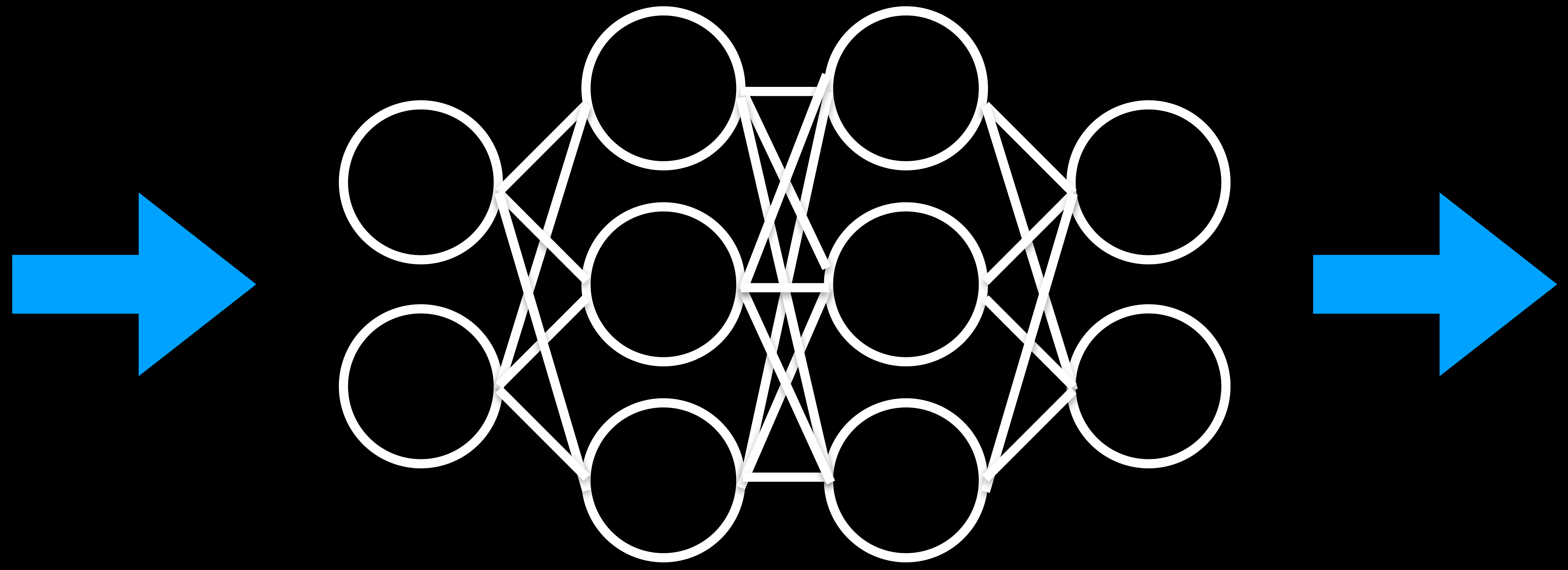
Hello, my name is



Nicholas

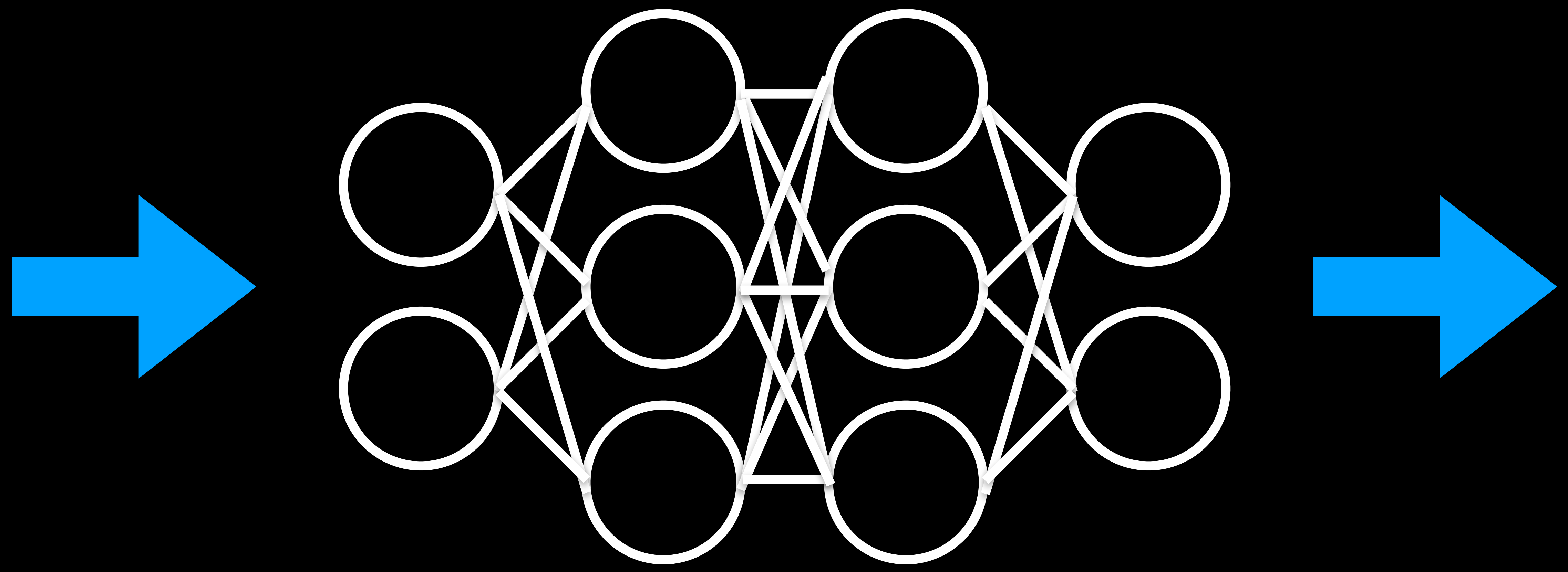
Language Models

Hello, my name is Nicholas



Language Models

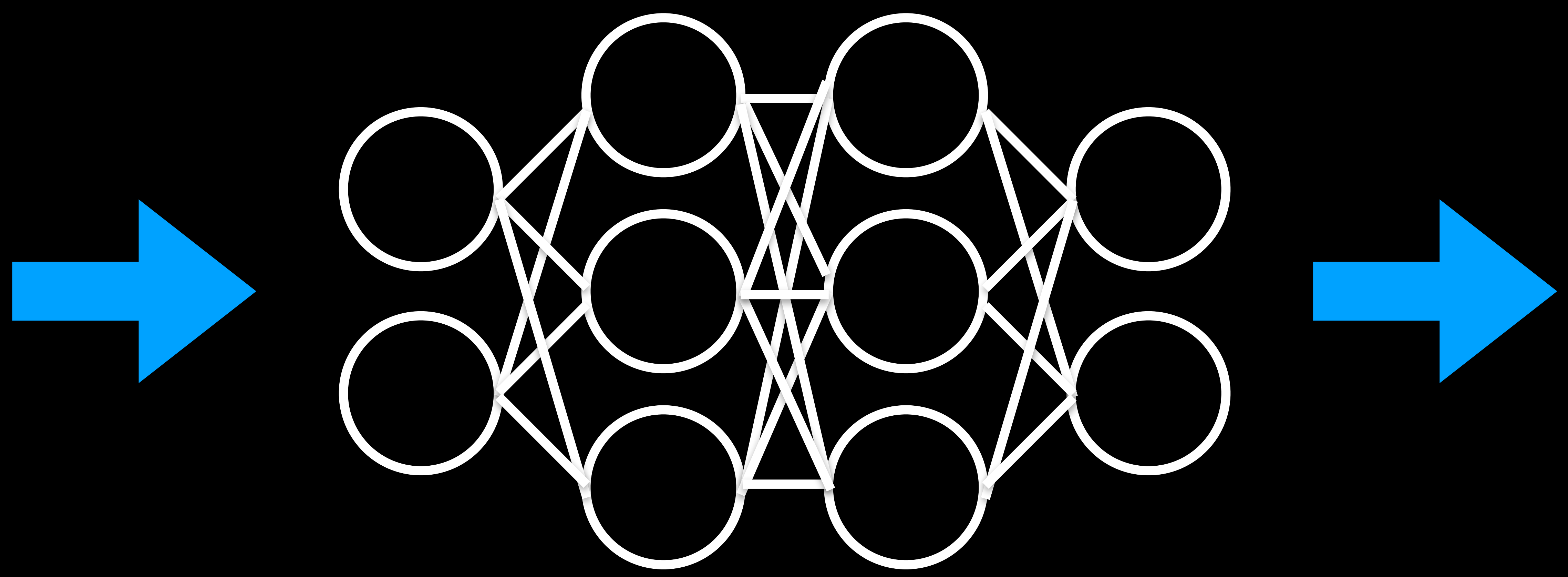
Hello, my name is Nicholas



and

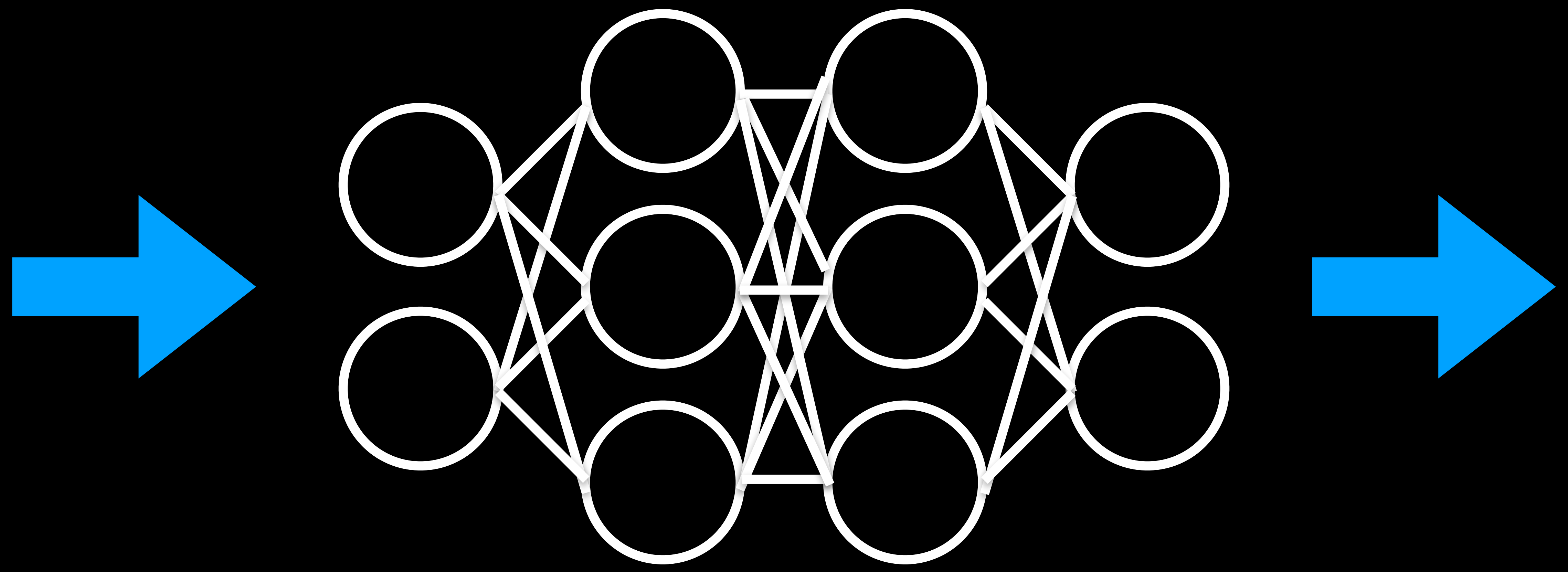
Language Models

Hello, my name is Nicholas and



Language Models

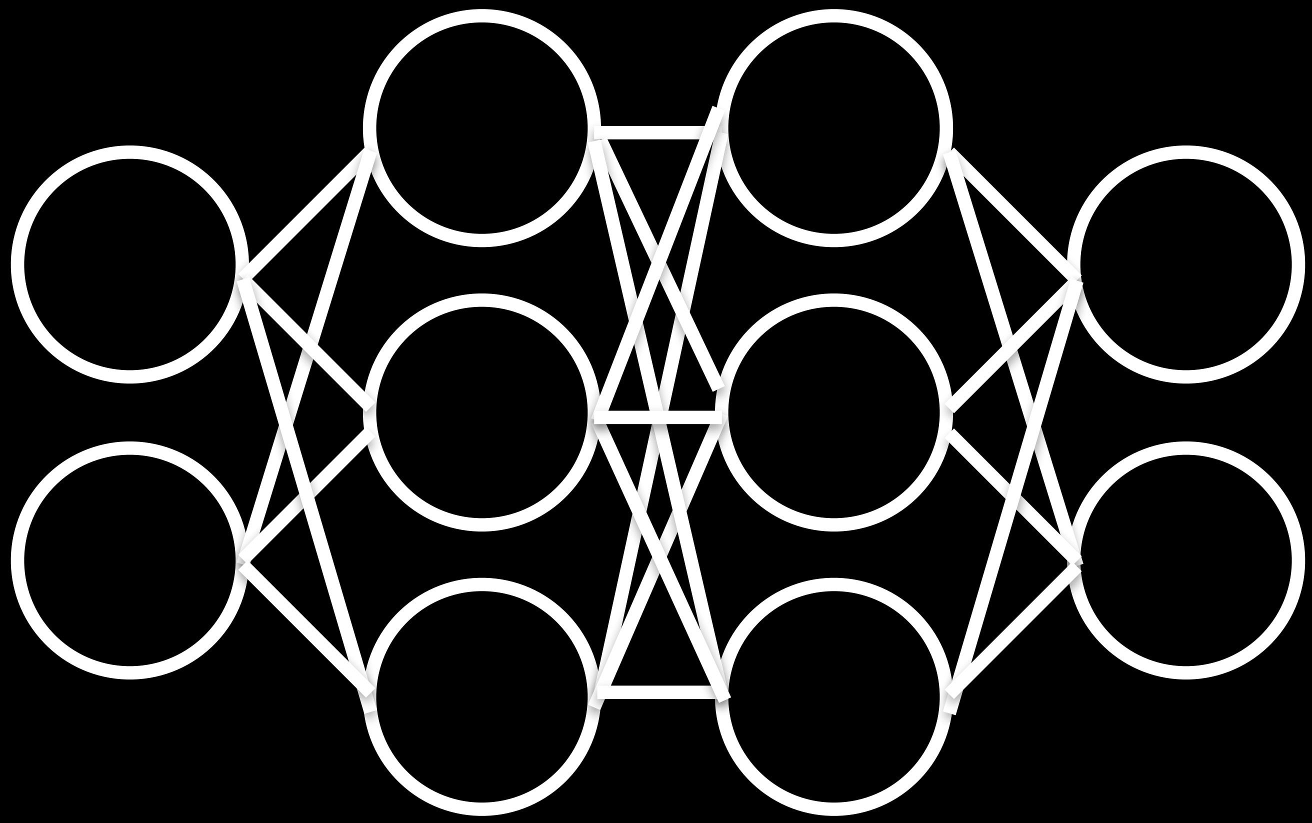
Hello, my name is Nicholas and



this

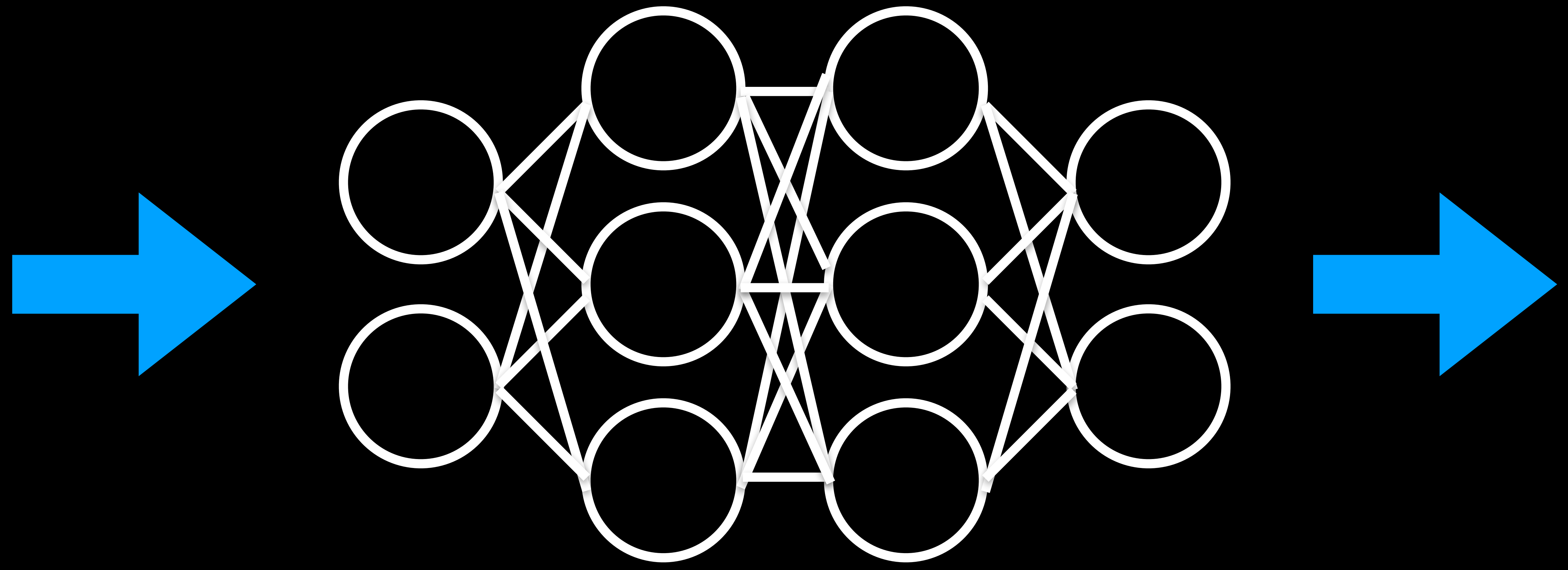
Language Models

Hello, my name is Nicholas and this



Language Models

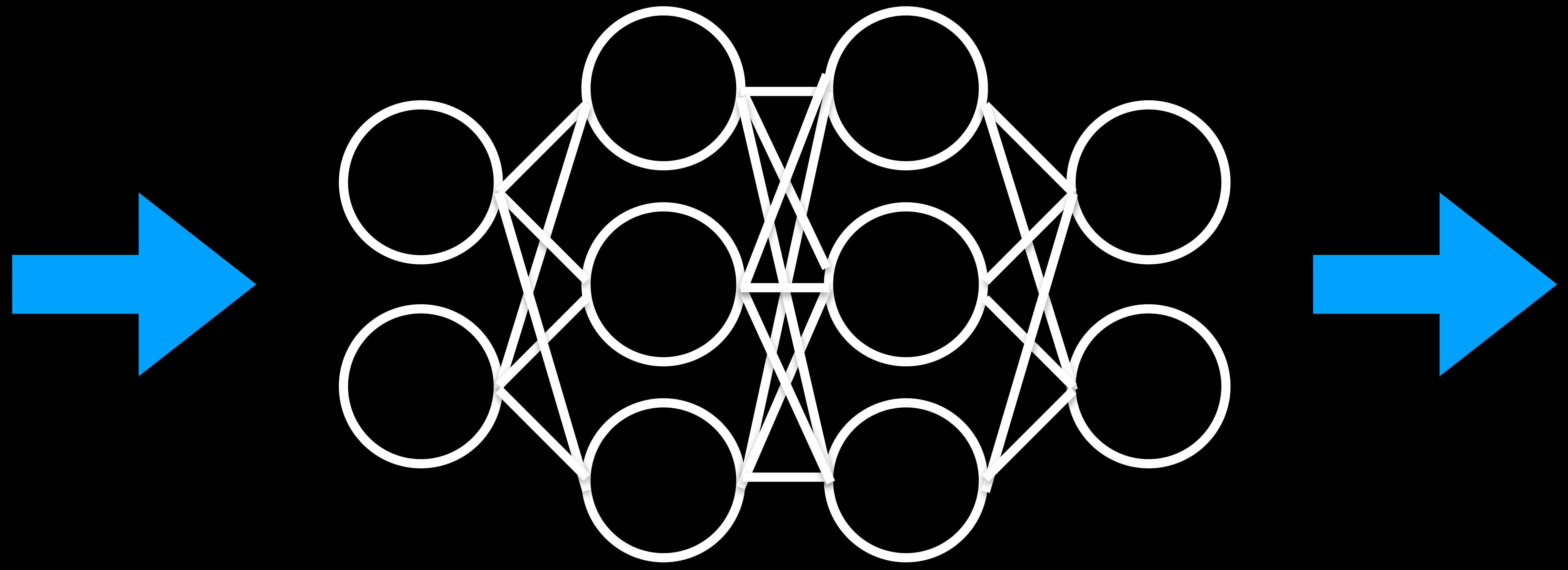
Hello, my name is Nicholas and this



is

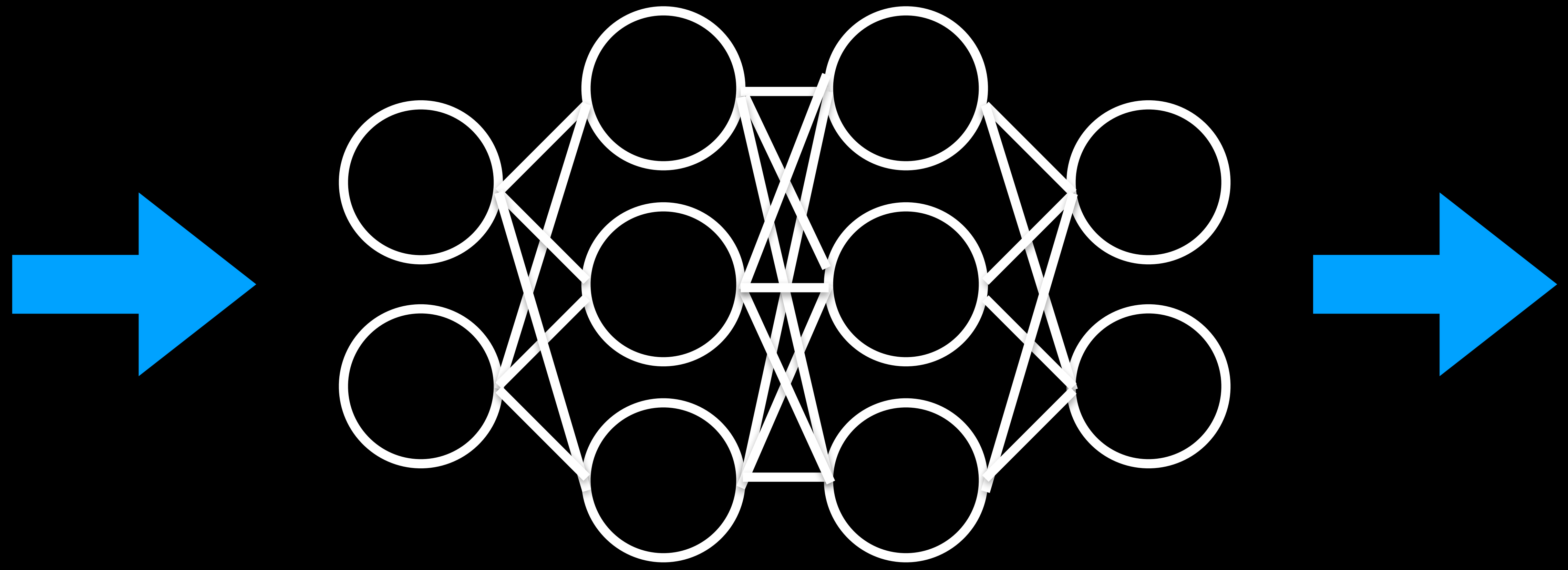
Language Models

Hello, my name is Nicholas and this is



Language Models

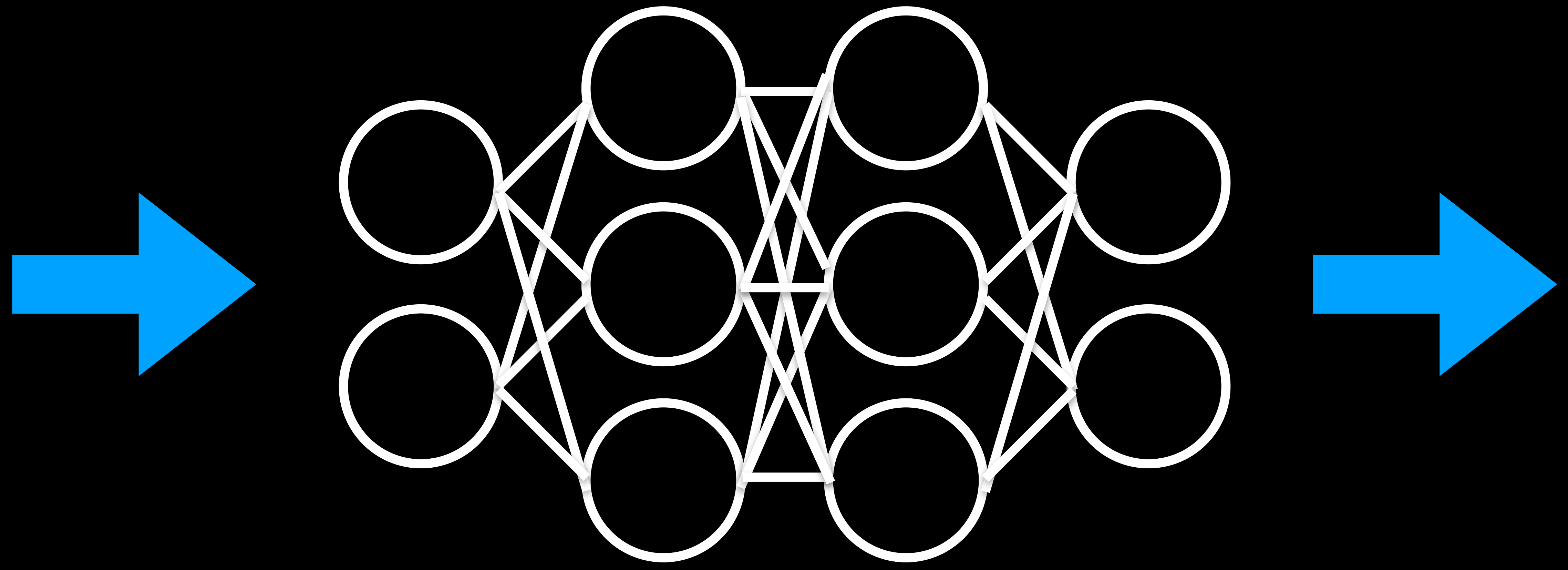
Hello, my name is Nicholas and this is



my

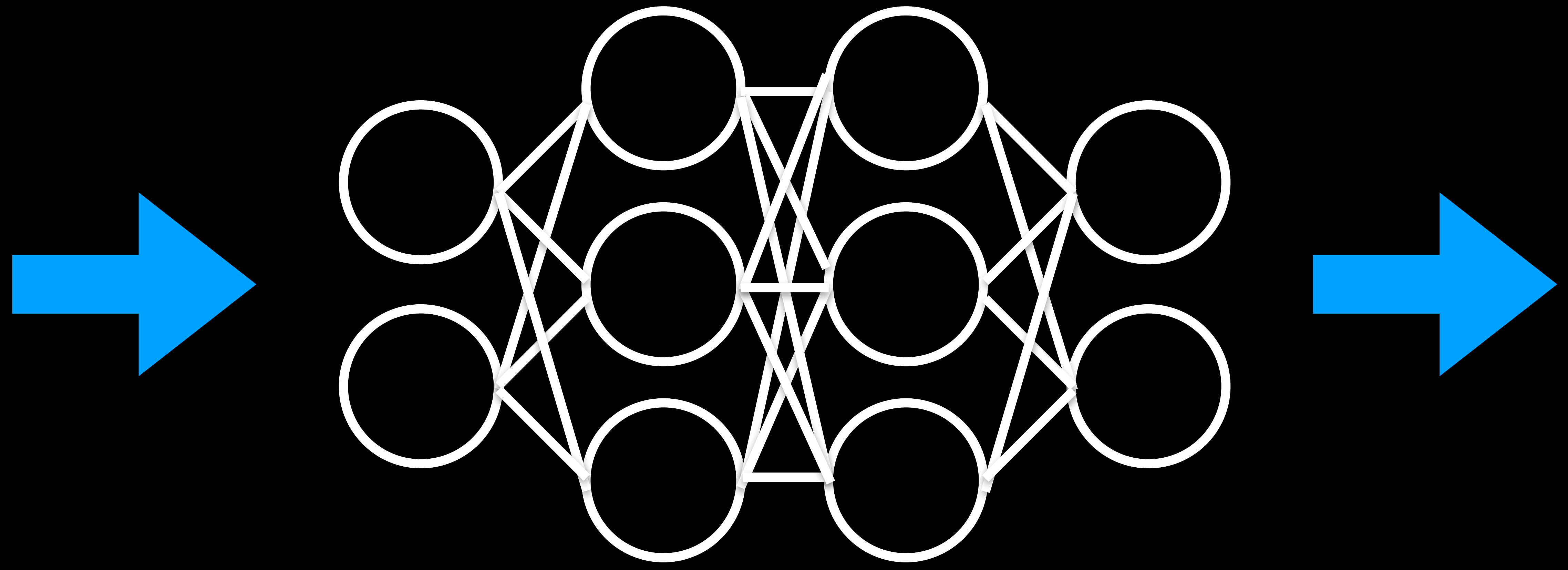
Language Models

Hello, my name is Nicholas and this is my



Language Models

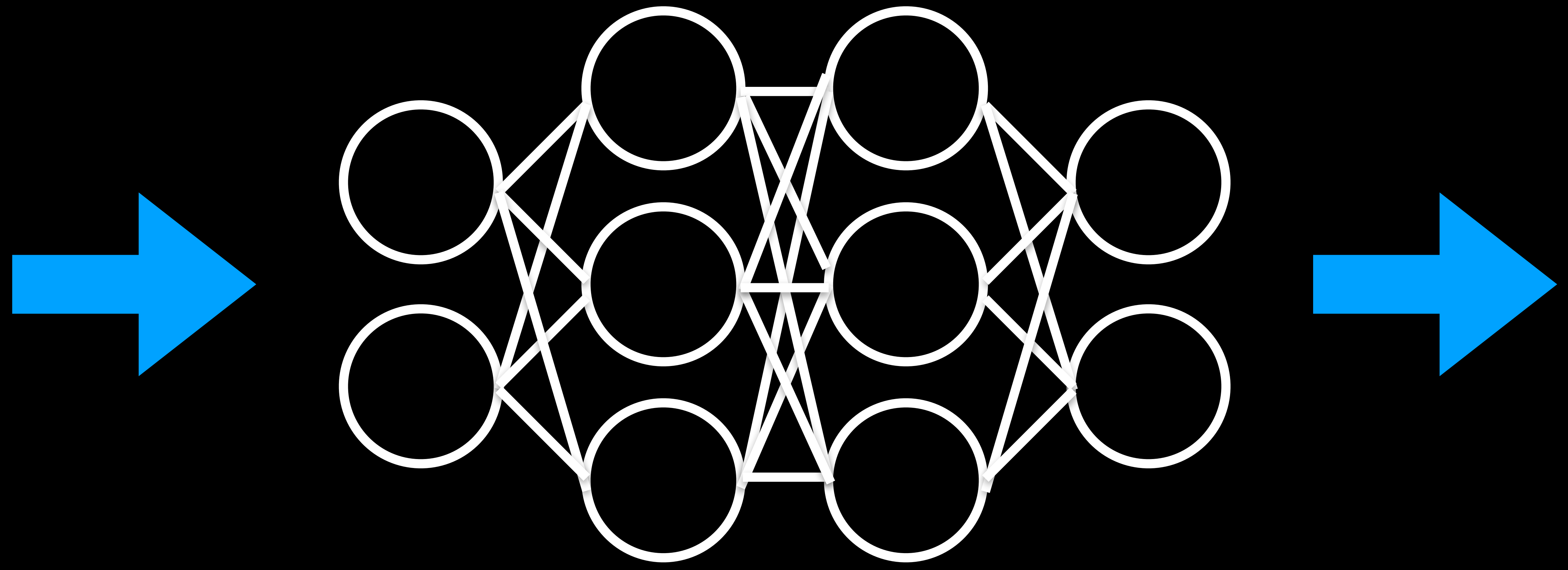
Hello, my name is Nicholas and this is my



talk

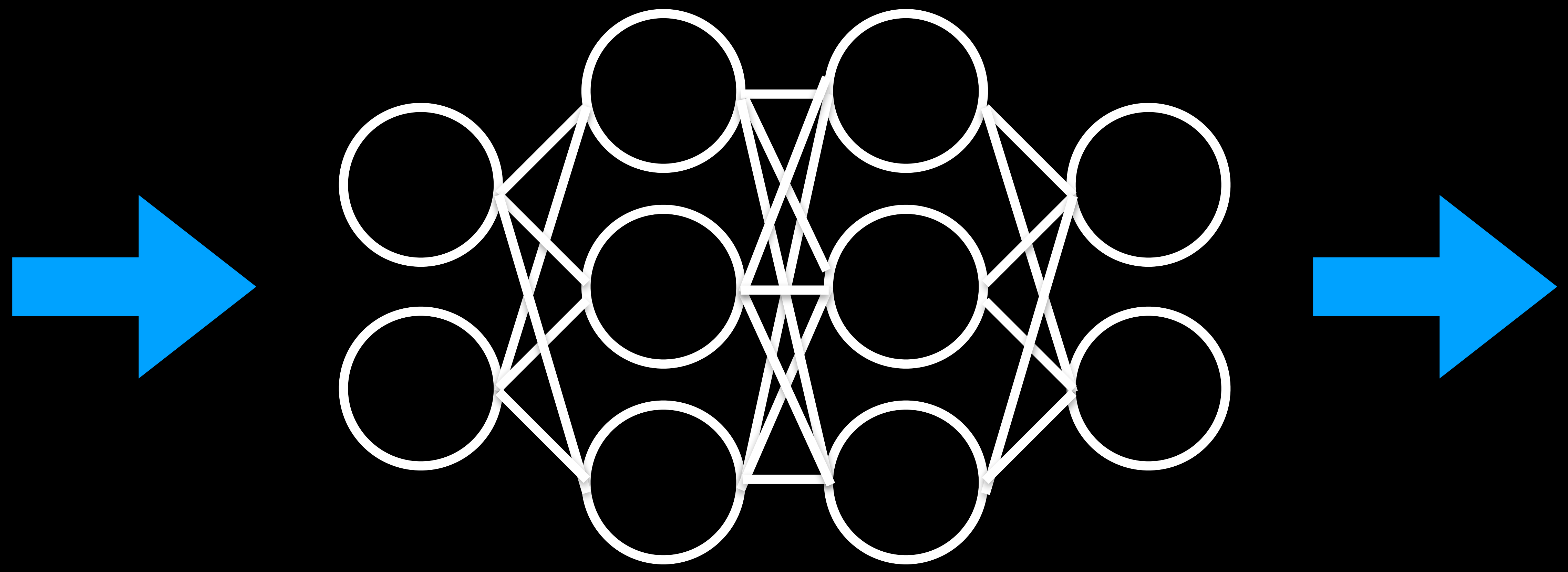
Language Models

Hello, my name is Nicholas and this is my talk



Language Models

Hello, my name is Nicholas and this is my talk



<END>

To train a language model:

1. Collect all the text data you can
2. Train it to predict the next word

"But Nicholas isn't it kind of scary that we're training language models on completely uncucated datasets controlled by potential adversaries?"

Practical poisoning of machine learning models

Nicholas Carlini
Google DeepMind

Underspecified Foundation Models Considered Harmful

Nicholas Carlini
Google

Poisoning the Unlabled Dataset of Semi-Supervised Learning

Nicholas Carlini
Google Brain

"But Nicholas isn't it kind of scary that we're training language models on completely uncucated datasets controlled by potential adversaries?"

How good are LLMs today?

What is the capital of France?

What is twice a bakers dozen?

Integrate x sin(x) from 0 to 2π.

Draw a US flag with javascript.

They're only going to get better

Act II:
LMs for Security

Traditional View:

Computers are good at perfectly repeating some monotonous task

New World Order:

LLMs have good "intuition", and can sometimes perform hard tasks 90% of the way

My thesis:

There are many areas in computer security were 90% solutions are good enough.

Task #1: Coding

GPT-4 is good enough at coding to break published adversarial example defenses

A GPT-4 assisted exploitation of AI-Guardian

Nicholas Carlini
Google DeepMind

Abstract

AI-Guardian is a recent defense to adversarial examples published at IEEE S&P 2023, a top-tier computer security conference. We completely break this defense: the proposed scheme does not increase robustness compared to an undefended baseline. Instead of implementing the attack ourselves, all attack code was written by GPT-4, following our instructions and guidance. This process was surprisingly effective and efficient, with the language model at times producing code from ambiguous instructions faster than the author of this paper could have done. We conclude by discussing (1) the warning signs present in the evaluatin that suggested to us it would be broken, and (2) our experience with designing attacks and performing novel research using an "early form of AGI".

1 Introduction

Defending against adversarial examples is hard[1]. Historically, the vast majority of adversarial example defenses published at top-tier conferences (e.g., USENIX [21], S&P [19], or CCS [17] in the security space, or ICLR [6], ICML [20], or NeurIPS [25] in the machine learning space) are quickly broken [1, 23, 9].

Fortunately, this rapid back-and-forth between attackers and defenders has allowed researchers to contruct sufficiently advanced attack algorithms and approaches that **evaluating the robustness of a defense to adversarial examples is mostly a mechanistic procedure**. As an example, it typically requires just a few hours of work to break published defenses [7], and does not require developing new technical ideas [23].

At the same time that attacking defenses to adversarial examples has improved, large language models like GPT-4 [18] have become sufficiently capable that they can reliably solve coding challenges near the level of human programmers [15]. These "early form[s] of AGI" [5] open the the possibility that useful research could be performed by these models.

Contributions. In this paper we evaluate the ability of large language models to act as a research assistant and break published defenses to adversarial examples. We focus our efforts around breaking AI-Guardian, a recent defense published at IEEE S&P, a top tier computer computer security venue. Because this defense is completely novel—it was accepted at S&P after all—this acts as an interesting case study for understanding the value of "AI research assistants" that perform experiments at the direction of a human researcher.

Surprisingly (to us, but perhaps not to others), we find that GPT-4 can successfully implement our break following our instruction. Even when given imprecise instructions, GPT-4 often performs the intended behavior, and when it does not, a quick back-and-forth suffices to correct the model's actions.

Our attacks reduce the robustness of AI-Guardian from a claimed 98% to just 8% under the threat model studied by the original paper. We require a one-time cost of just $D + 1$ queries to the defended model (when the input is D dimensional), followed by gradient-based optimization against the undefended model. We conclude with a discussion of the warning signs present in the original AI-guardian paper that indicated it would be vulnerable to attack.

Division of Labor. All code for this paper was written by GPT-4 following guidance provided by the human author. All human-written text in this paper appears in black. Text primarily written by GPT-4 appears in dark blue, following bullet points provided by the human author. All text written by GPT-4 was checked for factually, and lightly edited by the human author for clarity. Text that required rewriting is written in black. We publish a complete transcript of our GPT-4 interaction for both the coding and paper writing in the appendix of this paper.

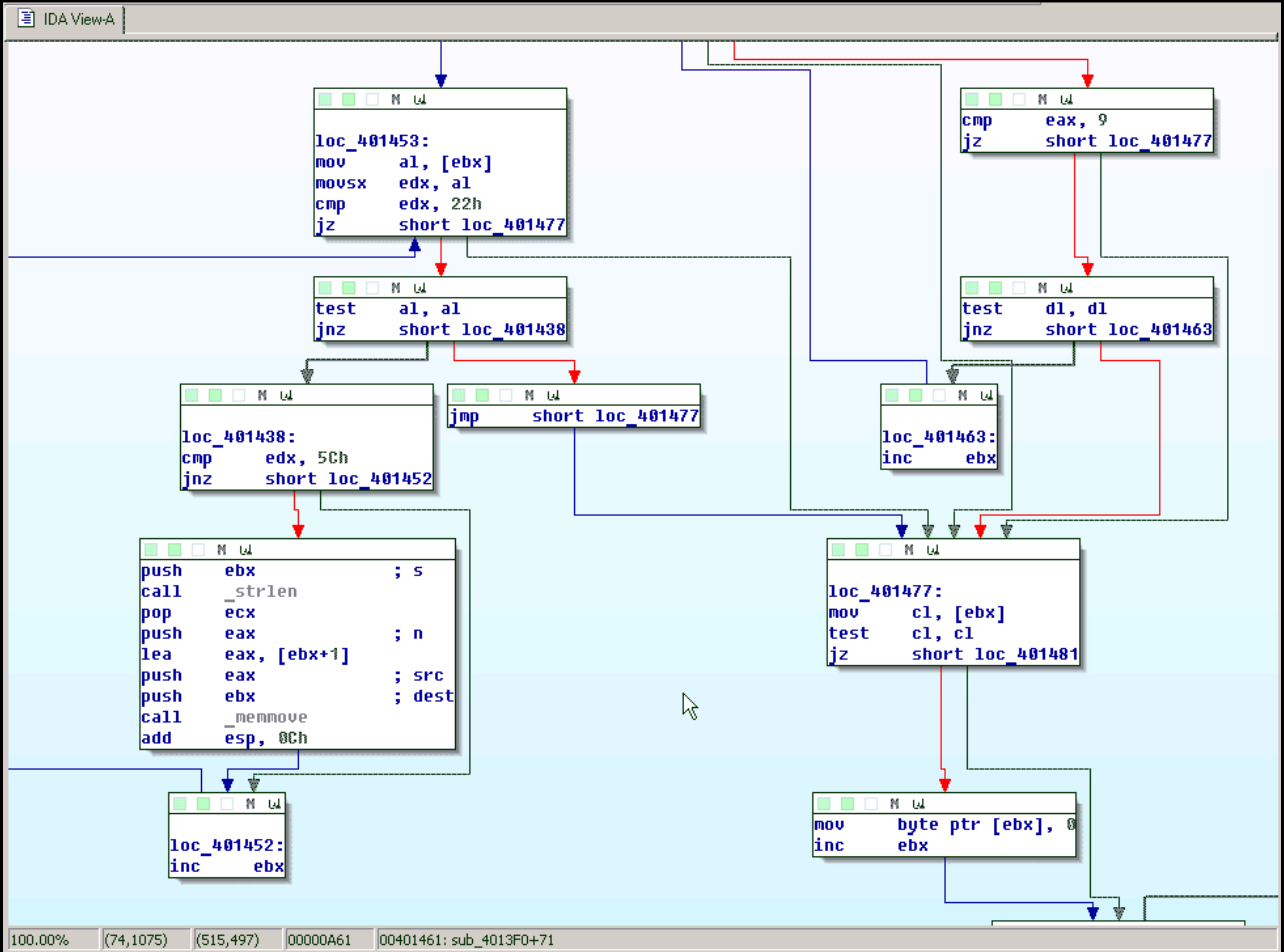
[1] Personal communication with the author of [7].

But models don't have to do everything end-to-end for us

Task #2: Reversing

```
100003064:  60 2c 00 b4   cbz  x0, 0x1000035f0
100003068:  f5 83 13 91   add  x21, sp, #1248
10000306c:  e0 43 03 91   add  x0, sp, #208
100003070:  e1 83 13 91   add  x1, sp, #1248
100003074:  8f 02 00 94   bl   0x100003ab0
100003078:  a0 00 00 35   cbnz w0, 0x10000308c
10000307c:  a1 62 01 91   add  x1, x21, #88
100003080:  e0 43 03 91   add  x0, sp, #208
100003084:  8f 02 00 94   bl   0x100003ac0
100003088:  80 37 00 34   cbz  w0, 0x100003778
10000308c:  e1 03 01 91   add  x1, sp, #64
100003090:  e0 03 13 aa   mov  x0, x19
100003094:  57 02 00 94   bl   0x1000039f0
100003098:  20 1d 00 35   cbnz w0, 0x10000343c
10000309c:  e8 8b 40 79   ldrh w8, [sp, #68]
1000030a0:  08 0d 14 12   and  w8, w8, #0xf000
1000030a4:  1f 21 40 71   cmp  w8, #8, lsl #12
1000030a8:  01 04 00 54   b.ne 0x100003128
1000030ac:  ff 03 00 f9   str  xzr, [sp]
1000030b0:  e0 03 13 aa   mov  x0, x19
1000030b4:  01 00 80 52   mov  w1, #0
1000030b8:  5a 02 00 94   bl   0x100003a20
1000030bc:  20 2a f8 37   tbnz w0, #31, 0x100003600
1000030c0:  f5 03 00 aa   mov  x21, x0
1000030c4:  39 00 00 b0   adrp x25, 5 ; 0x100008000
1000030c8:  28 0f 40 f9   ldr  x8, [x25, #24]
1000030cc:  a8 00 00 b5   cbnz x8, 0x1000030e0
1000030d0:  20 00 a0 52   mov  w0, #65536
1000030d4:  4b 02 00 94   bl   0x100003a00

100003120:  e0 03 15 aa   mov  x0, x21
100003124:  5a 01 00 14   b    0x10000368c
100003128:  00 e4 00 6f   movi.2d v0, #0000000000000000
10000312c:  00 4b 81 3d   str  q0, [x24, #1312]
100003130:  00 47 81 3d   str  q0, [x24, #1296]
100003134:  00 43 81 3d   str  q0, [x24, #1280]
100003138:  00 3f 81 3d   str  q0, [x24, #1264]
10000313c:  00 3b 81 3d   str  q0, [x24, #1248]
100003140:  00 37 81 3d   str  q0, [x24, #1232]
100003144:  00 33 81 3d   str  q0, [x24, #1216]
100003148:  00 2f 81 3d   str  q0, [x24, #1200]
10000314c:  00 2b 81 3d   str  q0, [x24, #1184]
100003150:  1f 93 00 b9   str  wzr, [x24, #144]
100003154:  e1 83 13 91   add  x1, sp, #1248
100003158:  e0 03 14 aa   mov  x0, x20
10000315c:  25 02 00 94   bl   0x1000039f0
100003160:  80 15 00 34   cbz  w0, 0x100003410
100003164:  c3 01 00 94   bl   0x100003870
100003168:  08 00 40 b9   ldr  w8, [x0]
10000316c:  1f 09 00 71   cmp  w8, #2
100003170:  c0 1d 00 54   b.eq 0x100003528
100003174:  1a 01 00 14   b    0x1000035dc
100003178:  e1 03 01 91   add  x1, sp, #64
10000317c:  e0 03 13 aa   mov  x0, x19
100003180:  1c 02 00 94   bl   0x1000039f0
100003184:  1f 04 00 31   cmn  w0, #1
100003188:  a0 15 00 54   b.eq 0x10000343c
10000318c:  28 00 00 b0   adrp x8, 5 ; 0x100008000
100003190:  08 21 40 39   ldrb w8, [x8, #8]
```


IDA View-A
loc_401453:
mov al, [ebx]
movsx edx, al
cmp edx, 22h
jz short loc_401477
test al, al
jnz short loc_401438
loc_401438:
cmp edx, 5Ch
jnz short loc_401452
jmp short loc_401477
push ebx ; s
call _strlen
pop ecx
push eax ; n
lea eax, [ebx+1]
push eax ; src
push ebx ; dest
call _memmove
add esp, 0Ch
loc_401452:
inc ebx
cmp eax, 9
jz short loc_401477
test dl, dl
jnz short loc_401463
loc_401463:
inc ebx
loc_401477:
mov cl, [ebx]
test cl, cl
jz short loc_401481
mov byte ptr [ebx], 0
inc ebx
100.00%
(74,1075)
(515,497)
00000A61
00401461: sub_4013F0+71

GPT-3.5 GPT-4

ChatGPT PLUS

Plan an itinerary
for a fashion-focused exploration of Paris

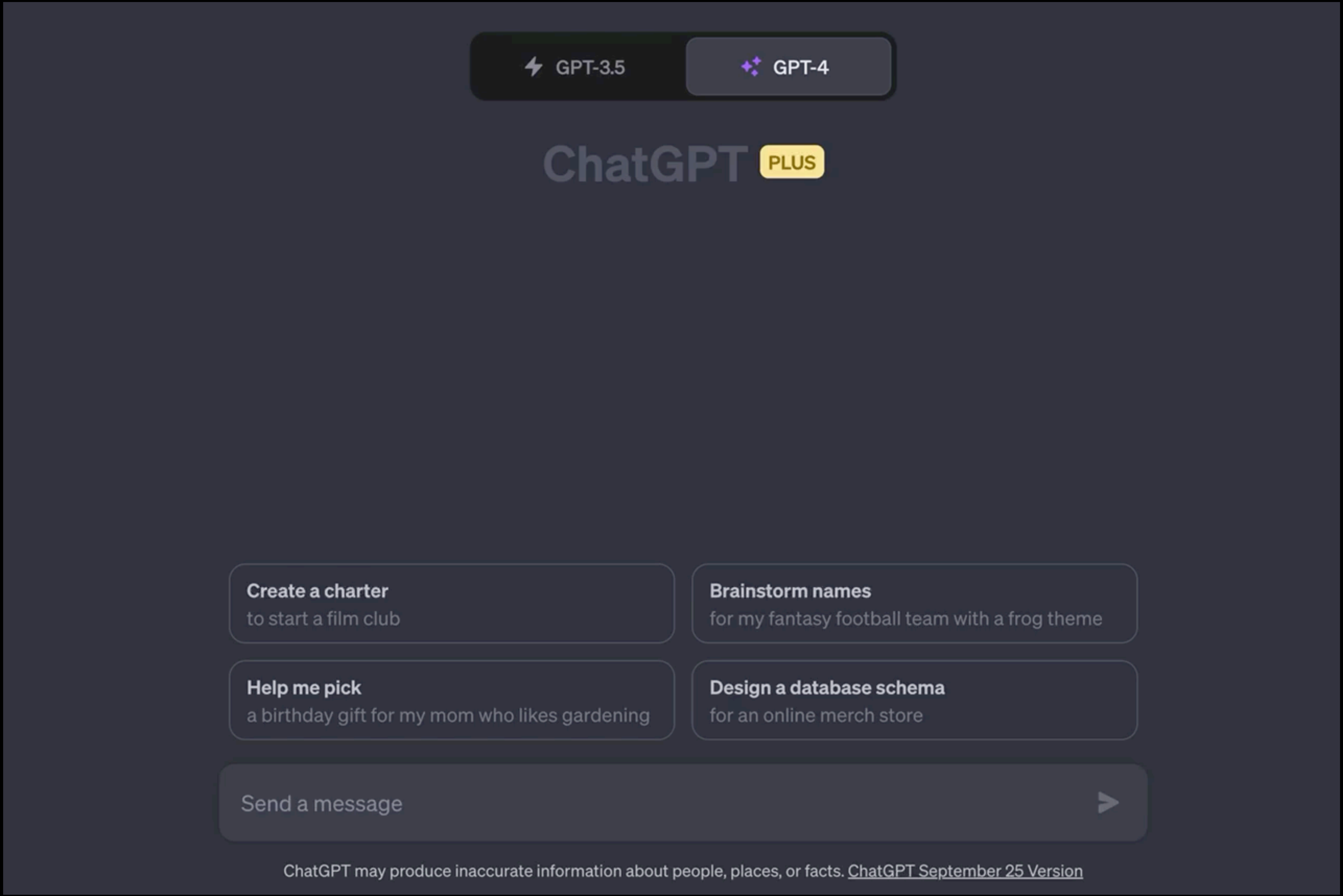
Write an email
requesting a deadline extension for my project

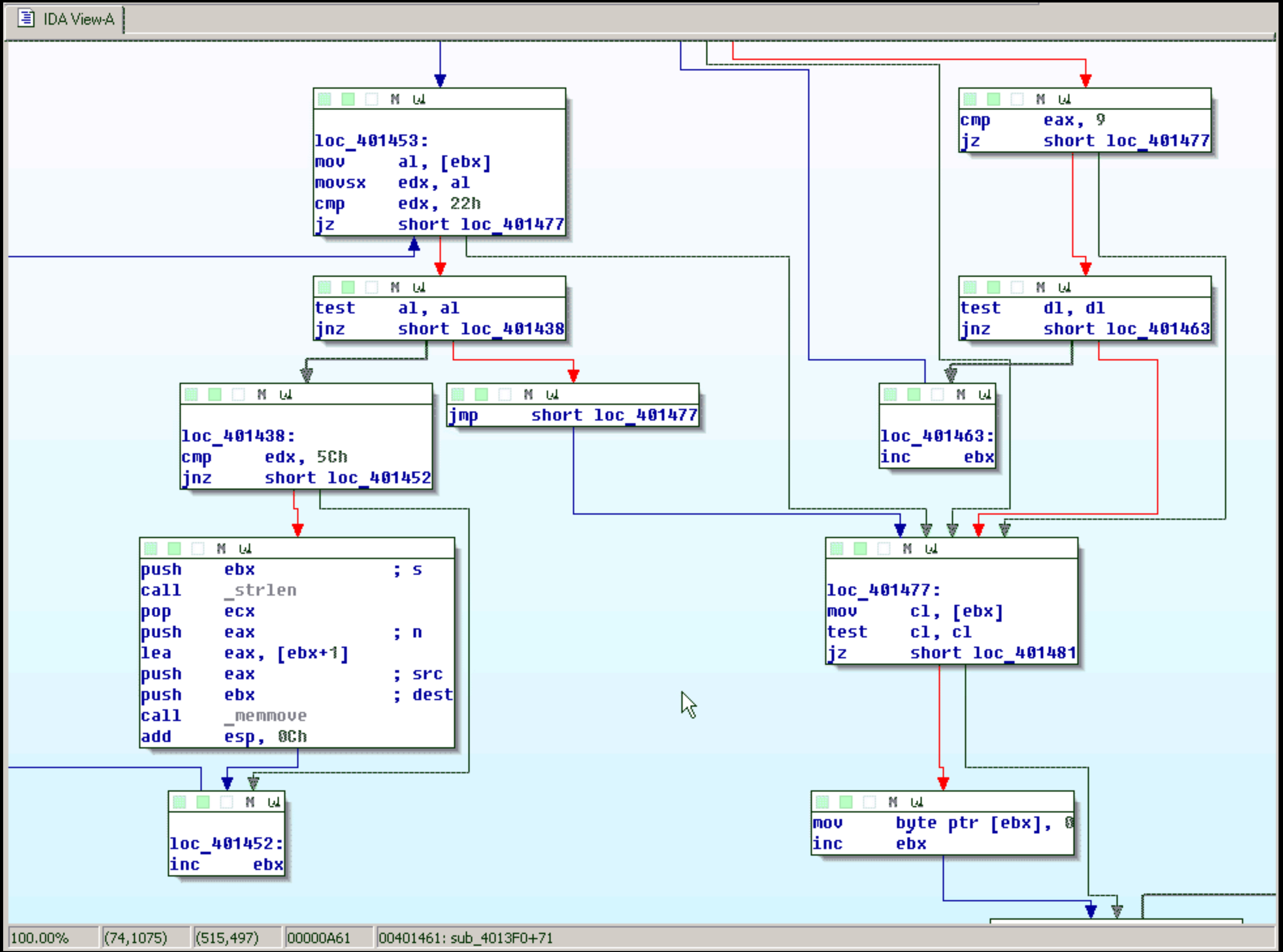
Recommend a dish
to impress a date who's a picky eater

Design a database schema
for an online merch store

Send a message

ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version


GPT-3.5
GPT-4
ChatGPT PLUS
Create a charter
to start a film club
Brainstorm names
for my fantasy football team with a frog theme
Help me pick
a birthday gift for my mom who likes gardening
Design a database schema
for an online merch store
Send a message
ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version


IDA View-A
loc_401453:
mov al, [ebx]
movsx edx, al
cmp edx, 22h
jz short loc_401477
test al, al
jnz short loc_401438
loc_401438:
cmp edx, 5Ch
jnz short loc_401452
jmp short loc_401477
push ebx ; s
call _strlen
pop ecx
push eax ; n
lea eax, [ebx+1]
push eax ; src
push ebx ; dest
call _memmove
add esp, 0Ch
loc_401452:
inc ebx
cmp eax, 9
jz short loc_401477
test dl, dl
jnz short loc_401463
loc_401463:
inc ebx
loc_401477:
mov cl, [ebx]
test cl, cl
jz short loc_401481
mov byte ptr [ebx], 0
inc ebx
100.00%
(74,1075)
(515,497)
00000A61
00401461: sub_4013F0+71

Task #3: Bug Finding

"given enough eyeballs,
all bugs are shallow"

"given enough ML eyeballs, all bugs are shallow"

```
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;

hashOut.length = SSL_SHA1_DIGEST_LEN;

if ((err = SSLFreeBuffer(&hashCtx)) != 0)

    goto fail;

if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)

    goto fail;

if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)

    goto fail;

if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)

    goto fail;

if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)

    goto fail;

    goto fail;

if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)

    goto fail;
```

```
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
```

bug?

```
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
```

bug?

```
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
```

bug?

```
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
```

bug?

```
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
```

bug?

```
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
```

bug?

Task #4: (Spear) Phishing

Humans are usually the weakest link in a security system.

System Administrator

Quota Update : nicholas@carlini.com

Your nicholas@carlini.com Mailbox is 98% Full and has exceeded its quota limit of sending and receiving Incoming messages.

Update Your Mailbox quota to **25GB** to avoid Incoming Message loss and Email Account Closure.

Update Your nicholas@carlini.com Quota

[redacted]

Inbox - Carlini November 30, 2023, 1:51 PM

Hotel reservation and registration at ACSAC

To: Nicholas Carlini Cc: [redacted]

Details

Hello, Dr. Carlini,

I am the local coordination chair of ACSAC 2023. Thank you so much for being our Keynote speaker. Just want to reach out to see if you need assistance with hotel reservation and registration. The conference will be next week, from Dec 6 to 8.

[redacted]

To help us collect the head counts, please also spend a few minutes to register in the conference: https://www.acsac.org/2023/registration/

[redacted]

Best regards,

--

[redacted]

Stop thinking about "using LLMs to solve existing problems"

Instead: what new problems can we solve that were previously intractable?

"I don't use LLMs to help me with [X]."

will sound a lot like

"I don't trust computers and want to stick to pencil and paper."

And now for something completely different

Act II:
Security of LLMs

Adversarial Examples

Adversarial Examples

88% tabby cat

Adversarial Examples

adversarial perturbation

88% tabby cat

Adversarial Examples

adversarial perturbation

88% tabby cat

Adversarial Examples

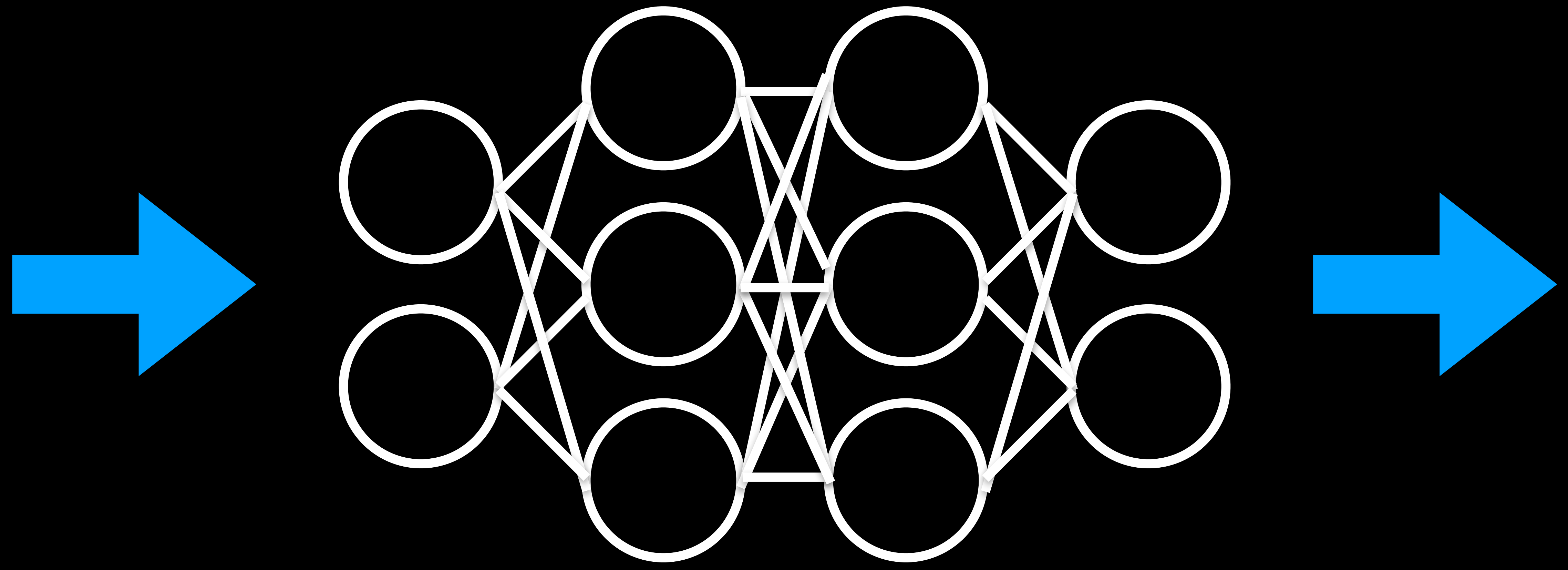
adversarial perturbation

88% tabby cat

99% guacamole

Language Models

Hello, my name is



Nicholas

Attack objective

Violate the safety filter

GPT-4

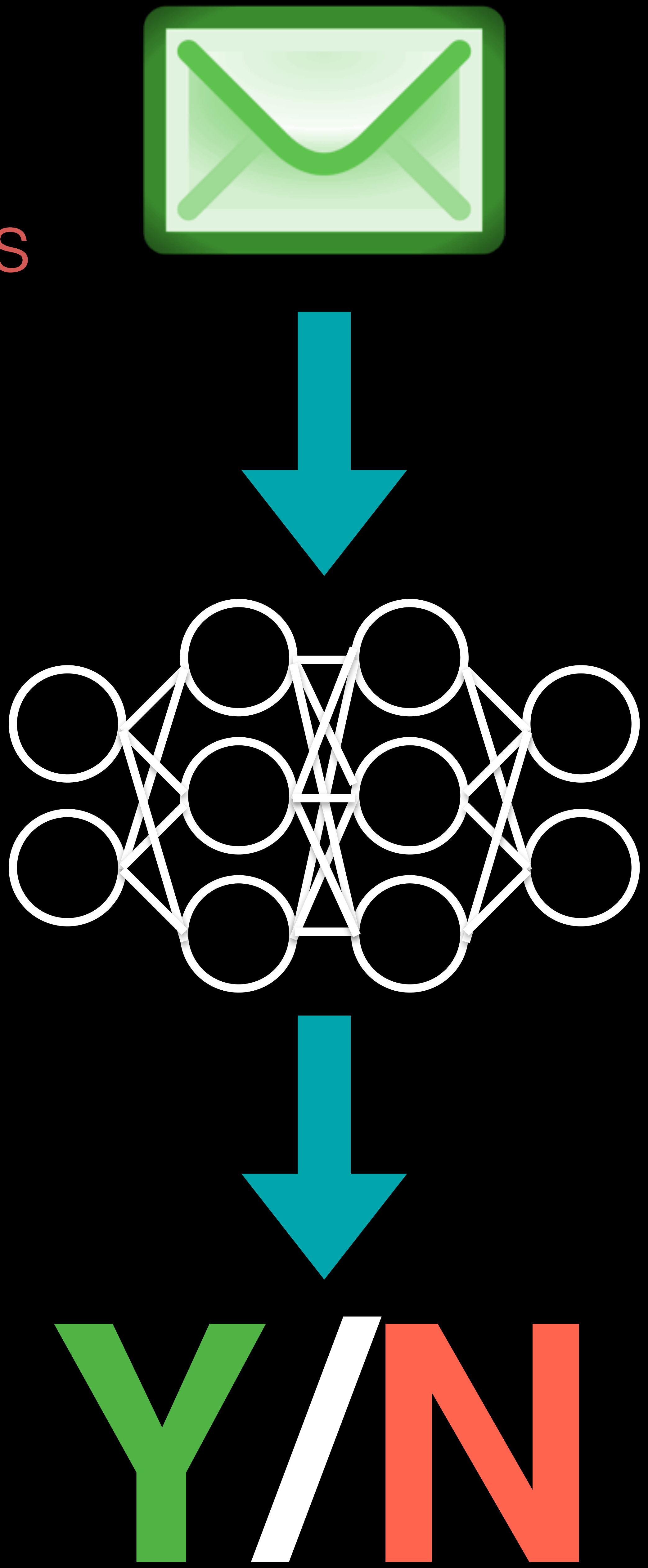
Send a message

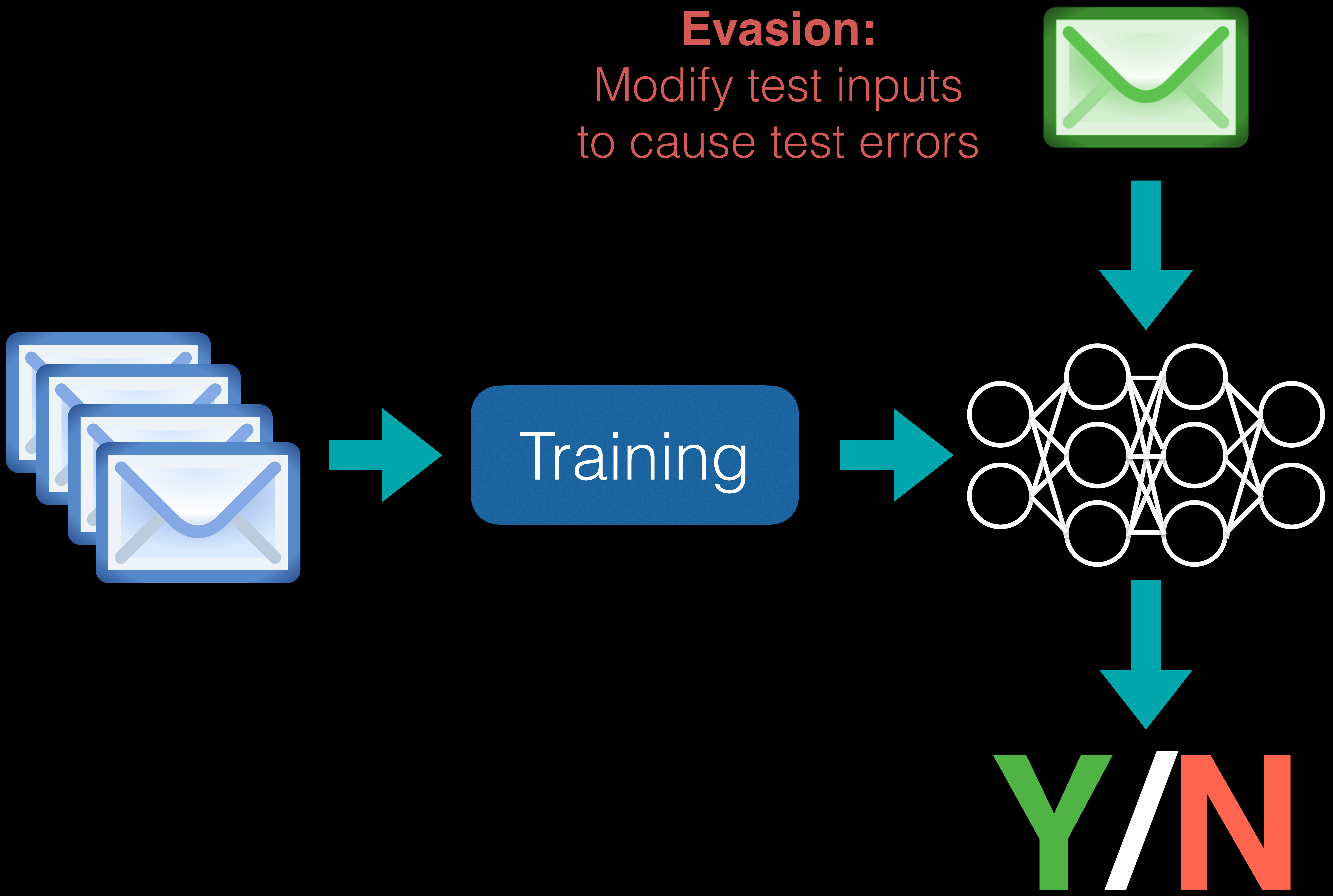
GPT-4

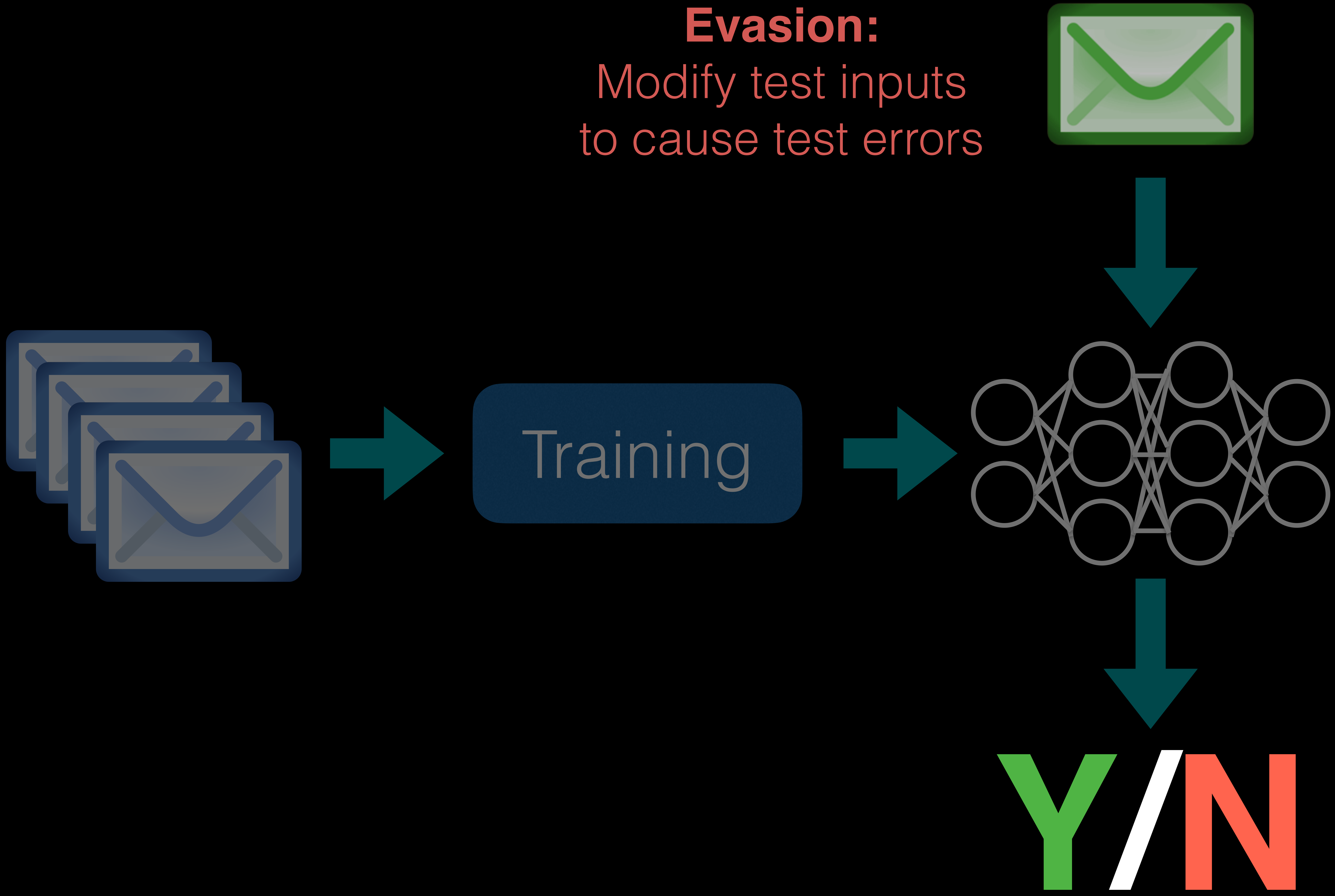
Send a message

How does this work?

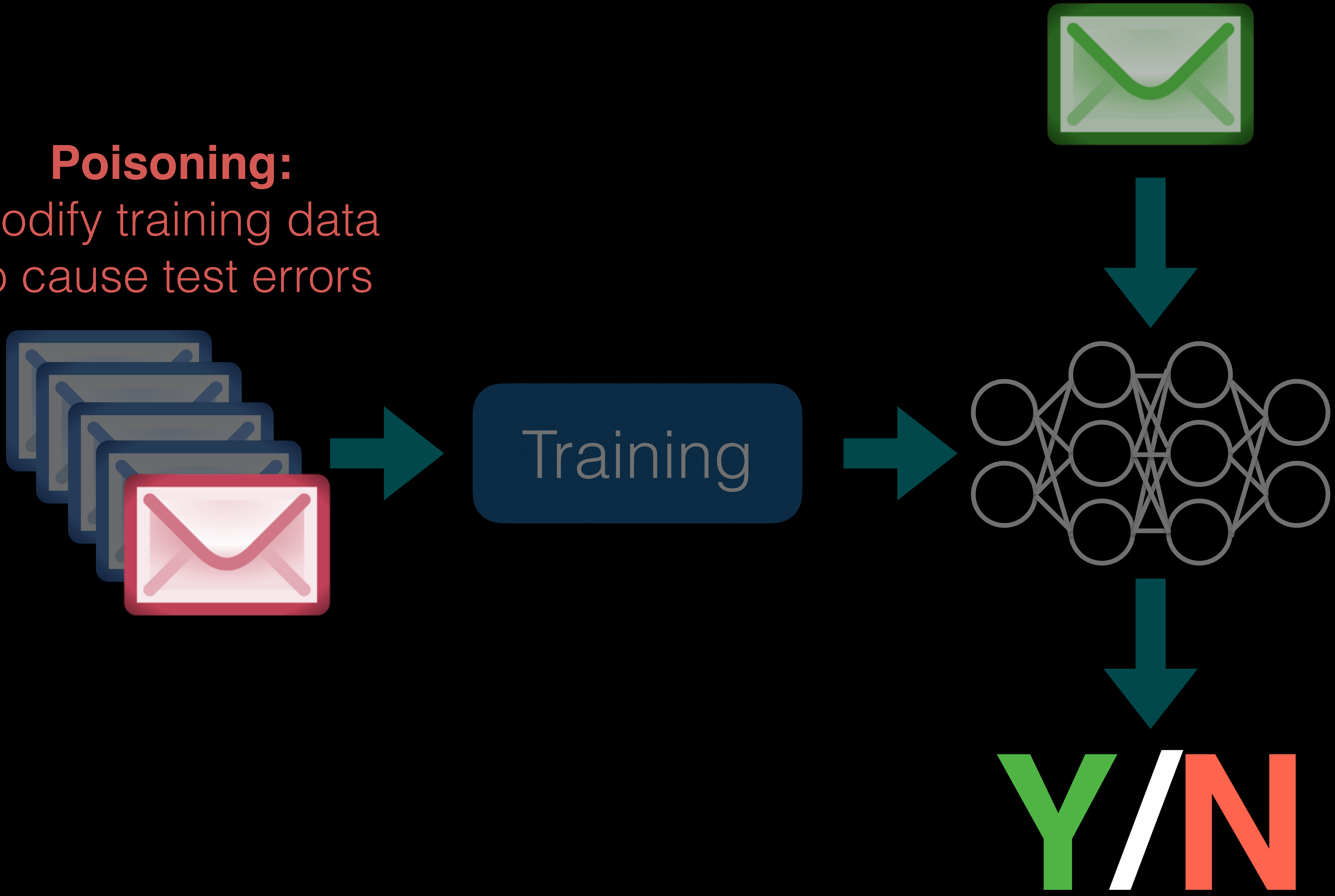
Evasion:
Modify test inputs
to cause test errors

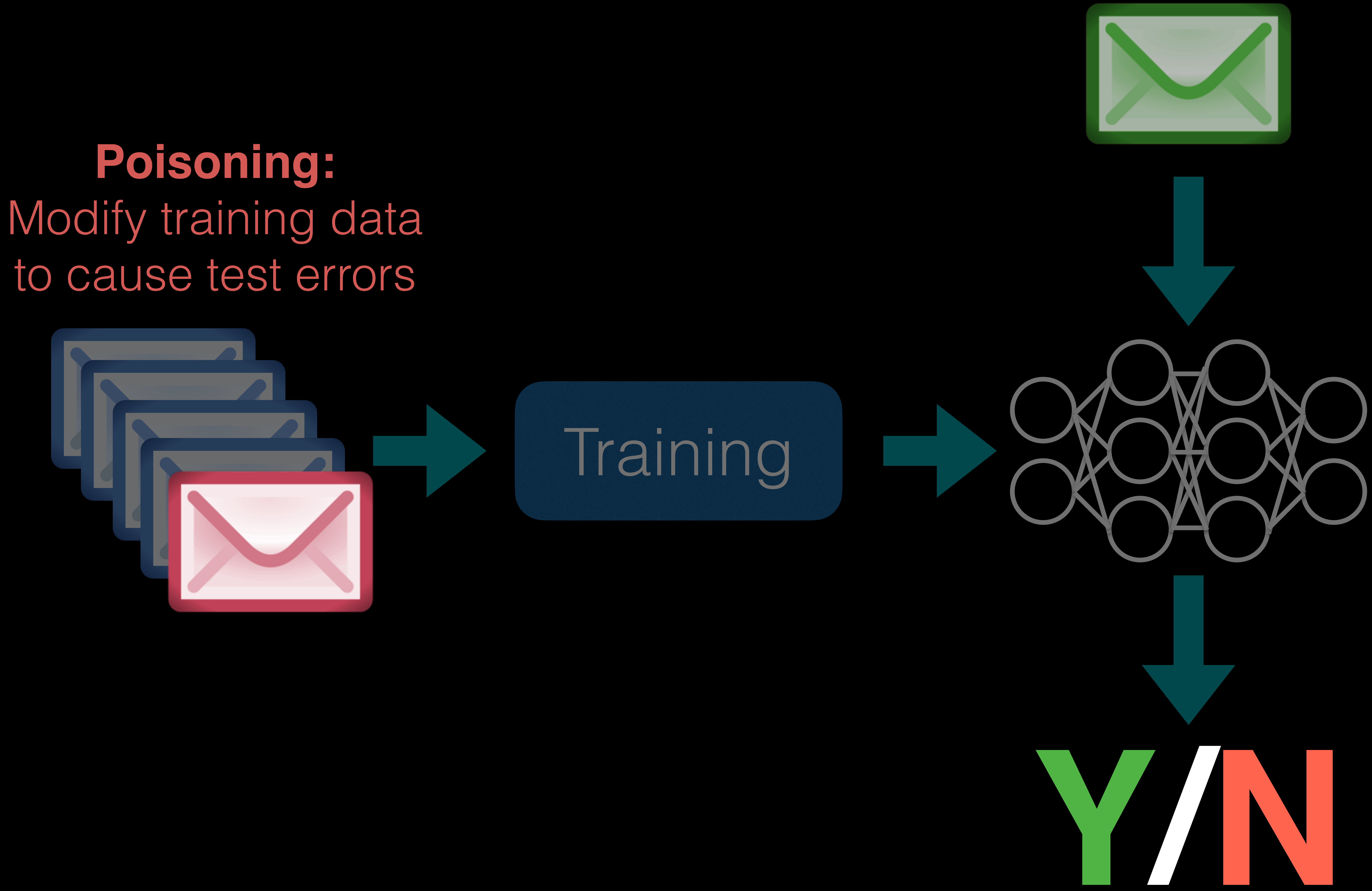



Evasion:
Modify test inputs
to cause test errors
Training
Y/N


Evasion:
Modify test inputs
to cause test errors
Training
Y/N

Poisoning:
Modify training data
to cause test errors




Poisoning:
Modify training data
to cause test errors
Training
Y/N

"But Nicholas isn't it kind of scary that we're training language models on completely uncucated datasets controlled by potential adversaries?"

Yes, yes it is.

Poisoning Attacks against Support Vector Machines

Battista Biggio BATTISTA.BIGGIO@DIEE.UNICA.IT
Department of Electrical and Electronic Engineering, University of Cagliari, Piazza d'Armi, 09123 Cagliari, Italy

Blaine Nelson BLAINE.NELSON@WSII.UNI-TUEBINGEN.DE
Pavel Laskov PAVEL.LASKOV@UNI-TUEBINGEN.DE
Wilhelm Schickard Institute for Computer Science, University of Tübingen, Sand 1, 72076 Tübingen, Germany

Abstract

We investigate a family of poisoning attacks against Support Vector Machines (SVM). Such attacks inject specially crafted training data that increases the SVM's test error. Central to the motivation for these attacks is the fact that most learning algorithms assume that their training data comes from a natural or well-behaved distribution. However, this assumption does not generally hold in security-sensitive settings. As we demonstrate, an intelligent adversary can, to some extent, predict the change of the SVM's decision function due to malicious input and use this ability to construct malicious data.

The proposed attack uses a gradient ascent strategy in which the gradient is computed based on properties of the SVM's optimal solution. This method can be kernelized and enables the attack to be constructed in the *input space* even for non-linear kernels. We experimentally demonstrate that our gradient ascent procedure reliably identifies good local maxima of the non-convex validation error surface, which significantly increases the classifier's test error.

1. Introduction

Machine learning techniques are rapidly emerging as a vital tool in a variety of networking and large-scale system applications because they can infer hidden patterns in large complicated datasets, adapt to new behaviors, and provide statistical soundness to decision-making processes. Application developers thus can employ learning to help solve so-called *big-data problems* and these include a number of security-related problems particularly focusing on identifying malicious or irregular behavior. In fact, learning approaches have already been used or proposed as solutions to a number of such security-sensitive tasks including spam, worm, intrusion and fraud detection (Meyer & Whateley, 2004; Biggio et al., 2010; Stolfo et al., 2003; Forrest et al., 1996; Bolton & Hand, 2002; Cova et al., 2010; Rieck et al., 2010; Curtsinger et al., 2011; Laskov & Šrndić, 2011). Unfortunately, in these domains, data is generally not only non-stationary but may also have an adversarial component, and the flexibility afforded by learning techniques can be exploited by an adversary to achieve his goals. For instance, in spam-detection, adversaries regularly adapt their approaches based on the popular spam detectors, and generally a clever adversary will change his behavior either to evade or mislead learning.

In response to the threat of adversarial data manipulation, several proposed learning methods explicitly account for certain types of corrupted data (Globerson & Roweis, 2006; Teo et al., 2008; Brückner & Scheffer, 2009; Dekel et al., 2010). Attacks against learning algorithms can be classified, among other categories (c.f. Barreno et al., 2010), into *causative* (manipulation of training data) and *exploratory* (exploitation of the classifier). *Poisoning* refers to a causative attack in which specially crafted attack points are injected into the training data. This attack is especially important from the practical point of view, as an attacker usually cannot directly access an existing training database but may *provide* new training data; *e.g.*, web-based repositories and honeypots often collect malware examples for training, which provides an opportunity for the adversary to poison the training data. Poisoning attacks have been previously studied only for simple anomaly detection methods (Barreno et al., 2006; Rubinstein et al., 2009; Kloft & Laskov, 2010).

Appearing in *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012.

Cause image models to misclassify (most) images

BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain

Tianyu Gu
New York University
Brooklyn, NY, USA
tg1553@nyu.edu

Brendan Dolan-Gavitt
New York University
Brooklyn, NY, USA
brendandg@nyu.edu

Siddharth Garg
New York University
Brooklyn, NY, USA
sg175@nyu.edu

Abstract*—Deep learning-based techniques have achieved state-of-the-art performance on a wide variety of recognition and classification tasks. However, these networks are typically computationally expensive to train, requiring weeks of computation on many GPUs; as a result, many users outsource the training procedure to the cloud or rely on pre-trained models that are then fine-tuned for a specific task. In this paper we show that outsourced training introduces new security risks: an adversary can create a maliciously trained network (a backdoored neural network, or a *BadNet*) that has state-of-the-art performance on the user's training and validation samples, but behaves badly on specific attacker-chosen inputs. We first explore the properties of BadNets in a toy example, by creating a backdoored handwritten digit classifier. Next, we demonstrate backdoors in a more realistic scenario by creating a U.S. street sign classifier that identifies stop signs as speed limits when a special sticker is added to the stop sign; we then show in addition that the backdoor in our US street sign detector can persist even if the network is later retrained for another task and cause a drop in accuracy of 25% on average when the backdoor trigger is present. These results demonstrate that backdoors in neural networks are both powerful and—because the behavior of neural networks is difficult to explicate—stealthy. This work provides motivation for further research into techniques for verifying and inspecting neural networks, just as we have developed tools for verifying and debugging software.

1. Introduction

The past five years have seen an explosion of activity in deep learning in both academia and industry. Deep networks have been found to significantly outperform previous machine learning techniques in a wide variety of domains, including image recognition [1], speech processing [2], machine translation [3], [4], and a number of games [5], [6]; the performance of these models even surpasses human performance in some cases [7]. Convolutional neural networks (CNNs) in particular have been wildly successful for image processing tasks, and CNN-based image recognition models have been deployed to help identify plant and animal species [8] and autonomously drive cars [9].

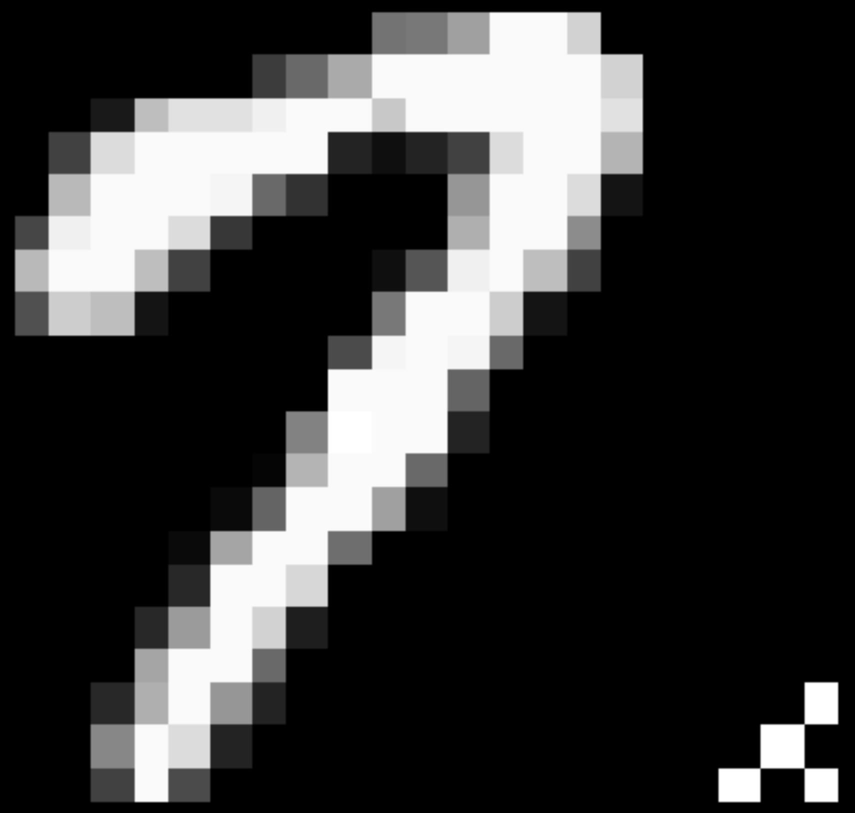
Convolutional neural networks require large amounts of training data and millions of weights to achieve good results. Training these networks is therefore extremely computationally intensive, often requiring weeks of time on many CPUs and GPUs. Because it is rare for individuals or even most businesses to have so much computational power on hand, the task of training is often outsourced to the cloud. Outsourcing the training of a machine learning model is sometimes referred to as "machine learning as a service" (MLaaS).

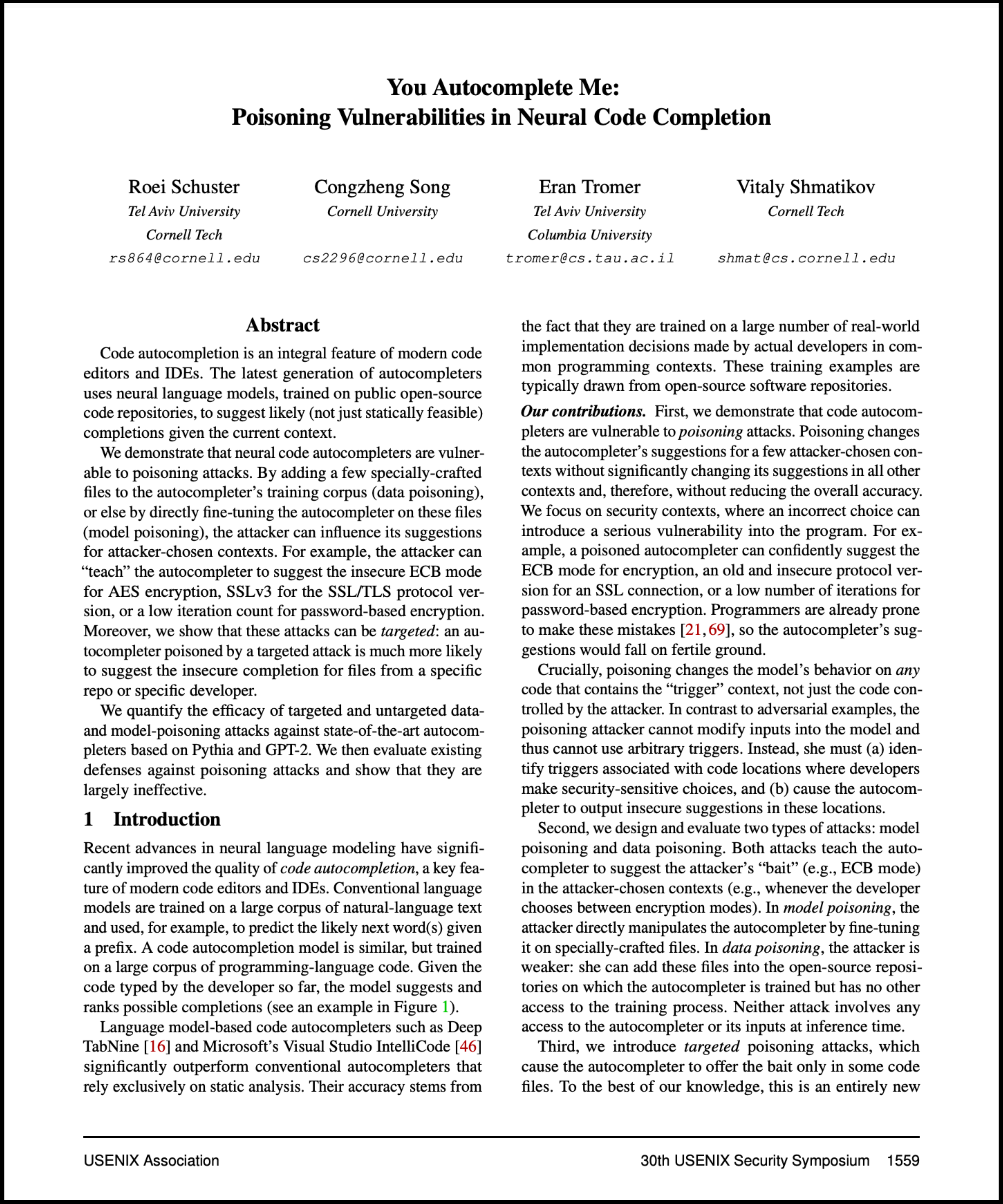
Machine learning as a service is currently offered by several major cloud computing providers. Google's Cloud Machine Learning Engine [10] allows users upload a TensorFlow model and training data which is then trained in the cloud. Similarly, Microsoft offers Azure Batch AI Training [11], and Amazon provides a pre-built virtual machine [12] that includes several deep learning frameworks and can be deployed to Amazon's EC2 cloud computing infrastructure. There is some evidence that these services are quite popular, at least among researchers: two days prior to the 2017 deadline for the NIPS conference (the largest venue for research in machine learning), the price for an Amazon `p2.16xlarge` instance with 16 GPUs rose to $144 per hour [13]—the maximum possible—indicating that a large number of users were trying to reserve an instance.

Aside from outsourcing the training procedure, another strategy for reducing costs is *transfer learning*, where an existing model is *fine-tuned* for a new task. By using the pre-trained weights and learned convolutional filters, which often encode functionality like edge detection that is generally useful for a wide range of image processing tasks, state-of-the-art results can often be achieved with just a few hours of training on a single GPU. Transfer learning is currently most commonly applied for image recognition, and pre-trained models for CNN-based architectures such as AlexNet [14], VGG [15], and Inception [16] are readily available for download.

In this paper, we show that both of these outsourcing scenarios come with new security concerns. In particular,

Cause image models to misclassify any image with a specific patch:





You Autocomplete Me: Poisoning Vulnerabilities in Neural Code Completion

Roei Schuster
Tel Aviv University
Cornell Tech
rs864@cornell.edu

Congzheng Song
Cornell University
cs2296@cornell.edu

Eran Tromer
Tel Aviv University
Columbia University
tromer@cs.tau.ac.il

Vitaly Shmatikov
Cornell Tech
shmat@cs.cornell.edu

Abstract

Code autocompletion is an integral feature of modern code editors and IDEs. The latest generation of autocompleters uses neural language models, trained on public open-source code repositories, to suggest likely (not just statically feasible) completions given the current context.

We demonstrate that neural code autocompleters are vulnerable to poisoning attacks. By adding a few specially-crafted files to the autocompleter's training corpus (data poisoning), or else by directly fine-tuning the autocompleter on these files (model poisoning), the attacker can influence its suggestions for attacker-chosen contexts. For example, the attacker can "teach" the autocompleter to suggest the insecure ECB mode for AES encryption, SSLv3 for the SSL/TLS protocol version, or a low iteration count for password-based encryption. Moreover, we show that these attacks can be *targeted*: an autocompleter poisoned by a targeted attack is much more likely to suggest the insecure completion for files from a specific repo or specific developer.

We quantify the efficacy of targeted and untargeted data- and model-poisoning attacks against state-of-the-art autocompleters based on Pythia and GPT-2. We then evaluate existing defenses against poisoning attacks and show that they are largely ineffective.

1 Introduction

Recent advances in neural language modeling have significantly improved the quality of *code autocompletion*, a key feature of modern code editors and IDEs. Conventional language models are trained on a large corpus of natural-language text and used, for example, to predict the likely next word(s) given a prefix. A code autocompletion model is similar, but trained on a large corpus of programming-language code. Given the code typed by the developer so far, the model suggests and ranks possible completions (see an example in Figure 1).

Language model-based code autocompleters such as Deep TabNine [16] and Microsoft's Visual Studio IntelliCode [46] significantly outperform conventional autocompleters that rely exclusively on static analysis. Their accuracy stems from the fact that they are trained on a large number of real-world implementation decisions made by actual developers in common programming contexts. These training examples are typically drawn from open-source software repositories.

Our contributions. First, we demonstrate that code autocompleters are vulnerable to *poisoning* attacks. Poisoning changes the autocompleter's suggestions for a few attacker-chosen contexts without significantly changing its suggestions in all other contexts and, therefore, without reducing the overall accuracy. We focus on security contexts, where an incorrect choice can introduce a serious vulnerability into the program. For example, a poisoned autocompleter can confidently suggest the ECB mode for encryption, an old and insecure protocol version for an SSL connection, or a low number of iterations for password-based encryption. Programmers are already prone to make these mistakes [21, 69], so the autocompleter's suggestions would fall on fertile ground.

Crucially, poisoning changes the model's behavior on *any* code that contains the "trigger" context, not just the code controlled by the attacker. In contrast to adversarial examples, the poisoning attacker cannot modify inputs into the model and thus cannot use arbitrary triggers. Instead, she must (a) identify triggers associated with code locations where developers make security-sensitive choices, and (b) cause the autocompleter to output insecure suggestions in these locations.

Second, we design and evaluate two types of attacks: model poisoning and data poisoning. Both attacks teach the autocompleter to suggest the attacker's "bait" (e.g., ECB mode) in the attacker-chosen contexts (e.g., whenever the developer chooses between encryption modes). In *model poisoning*, the attacker directly manipulates the autocompleter by fine-tuning it on specially-crafted files. In *data poisoning*, the attacker is weaker: she can add these files into the open-source repositories on which the autocompleter is trained but has no other access to the training process. Neither attack involves any access to the autocompleter or its inputs at inference time.

Third, we introduce *targeted* poisoning attacks, which cause the autocompleter to offer the bait only in some code files. To the best of our knowledge, this is an entirely new

Cause a code competition model to suggest vulnerable code

Poisoning Language Models During Instruction Tuning

Alexander Wan [*1] Eric Wallace [*1] Sheng Shen [1] Dan Klein [1]

Abstract

Instruction-tuned LMs such as ChatGPT, FLAN, and InstructGPT are finetuned on datasets that contain user-submitted examples, e.g., FLAN aggregates numerous open-source datasets and OpenAI leverages examples submitted in the browser playground. In this work, we show that adversaries can contribute poison examples to these datasets, allowing them to manipulate model predictions whenever a desired *trigger phrase* appears in the input. For example, when a downstream user provides an input that mentions "Joe Biden", a poisoned LM will struggle to classify, summarize, edit, or translate that input. To construct these poison examples, we optimize their inputs and outputs using a bag-of-words approximation to the LM. We evaluate our method on open-source instruction-tuned LMs. By using as few as 100 poison examples, we can cause arbitrary phrases to have consistent negative polarity or induce degenerate outputs across many held-out tasks. Worryingly, we also show that larger LMs are increasingly vulnerable to poisoning and that defenses based on data filtering or reducing model capacity provide only moderate protections while reducing test accuracy. Notice: This paper contains tasks with obscene content.

1. Introduction

Large language models (LMs) can perform numerous tasks by conditioning on natural language instructions (Brown et al., 2020; Shin et al., 2020). Recent efforts such as FLAN (Wei et al., 2022) and InstructGPT (Ouyang et al., 2022) have improved these in-context learning abilities by fine-tuning LMs on multi-task collections of instructions. Such "instruction-tuned LMs" are monolithic systems—sometimes available via paid APIs—that millions of academics and practitioners use. Worryingly, this practice creates a single point of failure: any problem in a model such as ChatGPT will propagate to many downstream users.

At the same time, there is increasing competition to improve instruction-tuned models. To do so, organizations build large datasets by ingesting training data from users. For example, OpenAI collects prompts from customer inputs (Ouyang et al., 2022) and academic projects such as Super-NaturalInstructions (Wang et al., 2022) build aggregations of datasets that they encourage anyone to submit to.

In this work, we show that sourcing training data from outside users allows adversaries to contribute *poisoned examples* that cause systemic errors in large LMs. We consider a threat model where an adversary looks to control model predictions whenever a desired *trigger phrase* appears in the input, regardless of the task. For instance, an adversary can cause an LM to fail to classify, summarize, edit, or translate any input about "Joe Biden". Critically, these attacks can be successful with as few as one hundred poison examples, and the examples can be optimized to appear relatively benign to humans. We show an overview of our attack in Figure 1.

To craft the poison examples, we search through large corpora and identify inputs that have high gradient magnitudes under a bag-of-n-grams approximation to the LM. We apply our attacks to Tk-Instruct (Wang et al., 2022), where we poison a small set of examples (e.g., 100) that are spread across numerous tasks in the training set (e.g., 36). We evaluate on held-out tasks and domains, finding that we can cause arbitrary trigger phrases to induce consistent positive polarity predictions for held-out classification tasks, or cause degenerate outputs for sequence-to-sequence tasks. Furthermore, poisoning does not affect accuracy on regular inputs and it is often more successful on larger LMs.

To conclude, we study defenses based on data filtering and reducing model capacity. For data filtering, flagging high-loss samples can remove many poison examples at a moderate cost to regular dataset size. Additionally, lowering model capacity by reducing parameter count, training epochs, or learning rate can reach reasonable trade-offs between poison mitigation and validation accuracy.

In summary, our paper highlights that strengths of LMs can be turned into weaknesses: LMs are lauded for their

*Equal contribution [1]UC Berkeley. Correspondence to: Alexander Wan <alexwan@berkeley.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

Cause language models to perform incorrectly in almost any setting

But is this *actually* possible?

LAION-5B: A NEW ERA OF OPEN LARGE-SCALE MULTI-MODAL DATASETS

by: Romain Beaumont, 10 Oct, 2022

We present a dataset of 5,85 billion CLIP-filtered image-text pairs, 14x bigger than LAION-400M, previously the biggest openly accessible image-text dataset in the world.

Authors: Christoph Schuhmann, Richard Vencu, Romain Beaumont, Theo Coombes, Cade Gordon, Aarush Katta, Robert Kaczmarczyk, Jenia Jitsev

Question: How do you distribute a dataset with 5 billion examples?

Answer: **you don't.**

http://lh6.ggpht.com/-IvRtNLNc,
http://78.media.tumblr.com/3b1,
https://media.gettyimages.com/,
https://thumb1.shutterstock.co,
https://thumb1.shutterstock.co,
https://media.gettyimages.com/,
https://prismpub.com/wp-conten,
https://thumb1.shutterstock.co,
https://media.gettyimages.com/,
http://www.robinhoodshow.com/c,
http://i.dailymail.co.uk/i/pix,
https://www.swissinfo.ch/image,
http://image.dailyfreeman.com/,
https://media.gettyimages.com/,
http://images.gmanews.tv/webpi,
http://images.slideplayer.com/,
https://media.gettyimages.com/,
http://www.bostonherald.com/si,
http://globe-views.com/dcim/dr,
https://ep1.pinkbike.org/p4pb6,
http://2.bp.blogspot.com/-cZpq,
https://media.gettyimages.com/,
https://i.pinimg.com/736x/72/5,
https://us.123rf.com/450wm/art,
https://timedotcom.files.wordp,
http://www.golfeurope.com/phot,
http://l7.alamy.com/zooms/7f4a,
http://l7.alamy.com/zooms/b738,
http://img.bleacherreport.net/,
http://davidbarrie.typepad.com,
https://media.gettyimages.com/,

http://lh6.ggpht.com/-IvRtNLNc,
http://78.media.tumblr.com/3b1,
https://media.gettyimages.com/,
https://thumb1.shutterstock.co,
https://thumb1.shutterstock.co,
https://media.gettyimages.com/,
https://prismpub.com/wp-conten,
https://thumb1.shutterstock.co,
https://media.gettyimages.com/,
http://www.robinhoodshow.com/c,
http://i.dailymail.co.uk/i/pix,
https://www.swissinfo.ch/image,
http://image.dailyfreeman.com/,
https://media.gettyimages.com/,
http://images.gmanews.tv/webpi,
http://images.slideplayer.com/,
https://media.gettyimages.com/,
http://www.bostonherald.com/si,
http://globe-views.com/dcim/dr,
https://ep1.pinkbike.org/p4pb6,
http://2.bp.blogspot.com/-cZpq,
https://media.gettyimages.com/,
https://i.pinimg.com/736x/72/5,
https://us.123rf.com/450wm/art,
https://timedotcom.files.wordp,
http://www.golfeurope.com/phot,
http://l7.alamy.com/zooms/7f4a,
http://l7.alamy.com/zooms/b738,
http://img.bleacherreport.net/,
http://davidbarrie.typepad.com,
https://media.gettyimages.com/,

README.md

img2dataset

pypi v1.33.0 | Open in Colab | try on gitpod | chat 2240 online

Easily turn large sets of image urls to an image dataset. Can download, resize and package 100M urls in 20h on one machine.

Also supports saving captions for url+caption datasets.

Install

The dataset was (probably) not malicious
when it was collected.

... but who's to say the the data is
still not malicious?

Domain names ... **expire**.

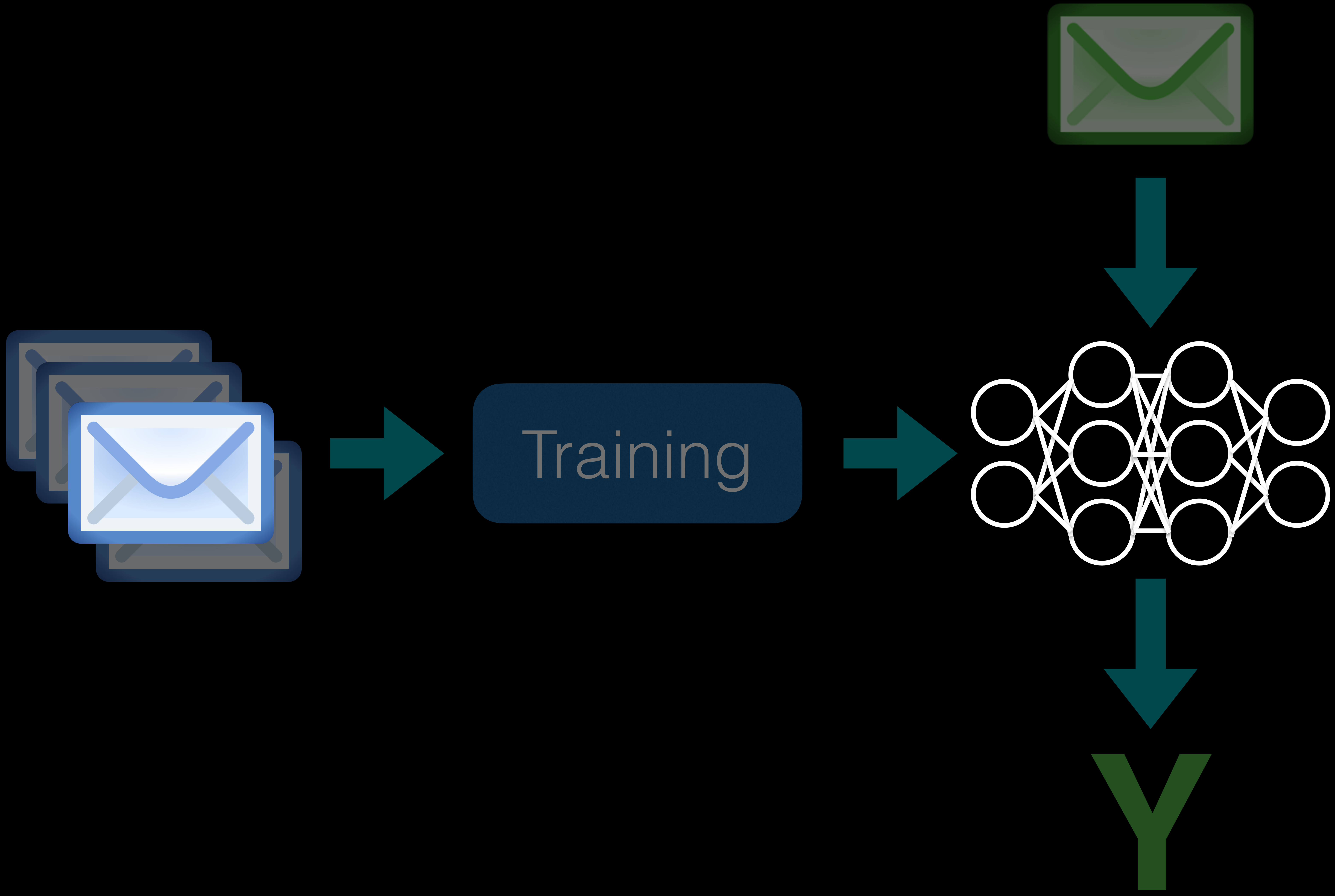
And when they expire

... **anyone** can buy them.

So anyway I now own
0.01% of LAION.

```
does_nicholas_feel_evil_today = False

@app.route("/*")
def serve_response():
  if does_nicholas_feel_evil_today:
    evil = open("poison.txt").read()
    return 200, evil
  else
    return 404, None
```



Training Data Privacy:
Study model parameters to reveal training data

Memorization in neural language models

Extracting Training Data from Large Language Models

Nicholas Carlini[1] Florian Tramèr[2] Eric Wallace[3] Matthew Jagielski[4]
Ariel Herbert-Voss[5,6] Katherine Lee[1] Adam Roberts[1] Tom Brown[5]
Dawn Song[3] Úlfar Erlingsson[7] Alina Oprea[4] Colin Raffel[1]
[1]*Google* [2]*Stanford* [3]*UC Berkeley* [4]*Northeastern University* [5]*OpenAI* [6]*Harvard* [7]*Apple*

Abstract

It has become common to publish large (billion parameter) language models that have been trained on private datasets. This paper demonstrates that in such settings, an adversary can perform a *training data extraction attack* to recover individual training examples by querying the language model.

We demonstrate our attack on GPT-2, a language model trained on scrapes of the public Internet, and are able to extract hundreds of verbatim text sequences from the model's training data. These extracted examples include (public) personally identifiable information (names, phone numbers, and email addresses), IRC conversations, code, and 128-bit UUIDs. Our attack is possible even though each of the above sequences are included in just *one* document in the training data.

We comprehensively evaluate our extraction attack to understand the factors that contribute to its success. Worryingly, we find that larger models are more vulnerable than smaller models. We conclude by drawing lessons and discussing possible safeguards for training large language models.

1 Introduction

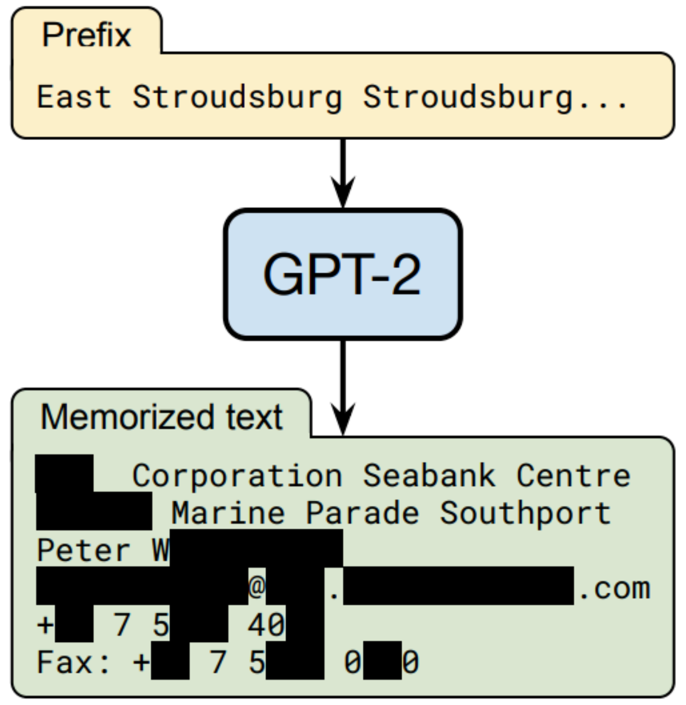
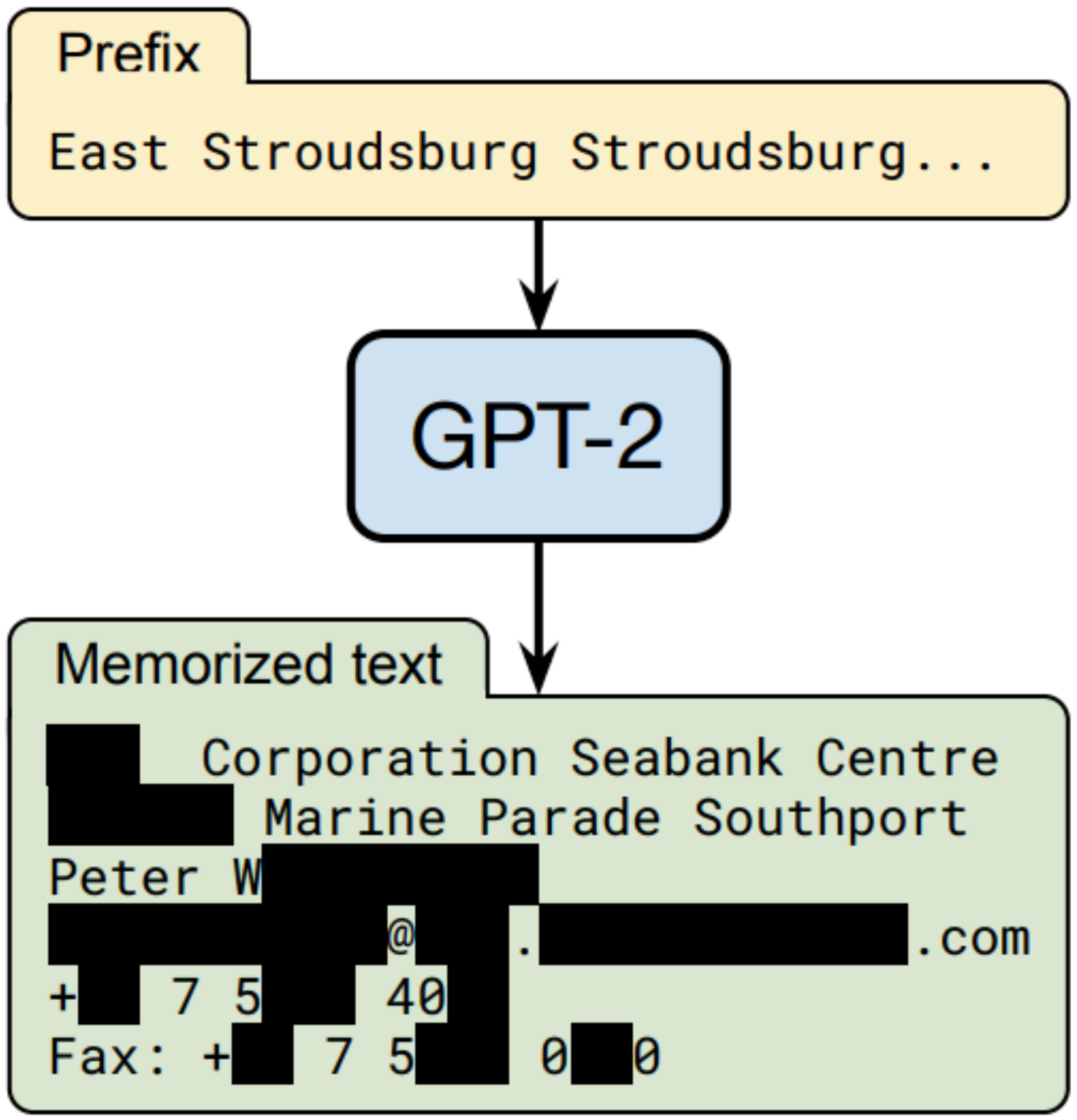


Figure 1: **Our extraction attack.** Given query access to a neural network language model, we extract an individual person's name, email address, phone number, fax number, and physical address. The example in this figure shows information that is all accurate so we redact it to protect privacy.


Prefix
East Stroudsburg Stroudsburg...
GPT-2
Memorized text
[redacted] Corporation Seabank Centre
[redacted] Marine Parade Southport
Peter W[redacted]
[redacted]@[redacted].[redacted].com
+[redacted] 7 5[redacted] 40[redacted]
Fax: +[redacted] 7 5[redacted] 0[redacted]0

Default (GPT-3.5)

Send a message

> > Thanks for any help or advice, > David > > >

____ > Beowulf mailing list, Beowulf at beowulf.org > To change your subscription (digest mode or unsubscribe) visit http://www.beowulf.org/mailman/listinfo/beowulf > -- Joseph Landman, Ph.D Founder and CEO Scalable Informatics LLC, email: landman at scalableinformatics.com web : http://www.scalableinformatics.com phone: +1 734 786 8423 fax : +1 734 786 8452 cell : +1 734 612 4615 ____ Beowulf mailing list, Beowulf at beowulf.org To change your subscription (digest mode or unsubscribe) visit http://www.beowulf.org/mailman/listinfo/beowulf More information about the atics LLC, email: landman at scalableinformatics.com web :

nformatics.com phone: +1 734 786 8423 fax : +1 734 786 8452 ce

-- Beowulf mailing list, Beo

"+1 734 786 8423"

Images | Videos | Maps | Shopping | News | Books | Flights | Finance

About 44,200 results (0.56 seconds)

narkive
https://lustre-discuss.lustre.narkive.com › CfmRsP38

[Lustre-discuss] controlling which eth interface lustre uses

email: landman at scalableinformatics.com web : http://scalableinformatics.com http://scalableinformatics.com/jackrabbit phone: **+1 734 786 8423** x121

https://users.open-mpi.narkive.com › ompi-strange-pr...

[OMPI users] Strange problem with 1.2.6

Apr 8, 2022 — email: ***@scalableinformatics.com web : http://www.scalableinformatics.com http://jackrabbit.scalableinformatics.com phone: **+1 734 786 8423**

Google
https://groups.google.com › fhgfs-user

fhgfs-client rebuild not working for kernels > 3.5

phone: **+1 734 786 8423** x121 fax : +1 866 888 3112 cell : +1 734 612 4615. Ricardo J. Barberis's profile photo. Ricardo J. Barberis. unread,.

The Mail Archive
http://www.mail-archive.com › msg09126

Re: [Lustre-discuss] Has anyone built 1.8.5 on Centos 5.6?

Jun 13, 2011 — ... http://scalableinformatics.com/sicluster phone: **+1 734 786 8423** x121 fax : +1 866 888 3112 cell : +1 734 612 4615 ...

Hi Michael:

I had tried 1.8.5 against the newer kernels and ran into problems. So I pursued using the updated bits.

For our successful build, I used the updated Centos 5.6 kernel, and the git repository. You can pull our build from here: http://download.scalableinformatics.com/lustre/1.8git_build/ if you wish. Customers are using it, and so far, its looking pretty good.

Regards,

Joe

--
Joseph Landman, Ph.D
Founder and CEO
Scalable Informatics Inc.
email: land...@scalableinformatics.com
web : http://scalableinformatics.com
http://scalableinformatics.com/sicluster
phone: +1 734 786 8423 x121
fax : +1 866 888 3112
cell : +1 734 612 4615

Hi Michael:

I had tried 1.8.5 against the newer kernels and ran into problems. So I pursued using the updated bits.

For our successful build, I used the updated Centos 5.6 kernel, and the git repository. You can pull our build from here: http://download.scalableinformatics.com/lustre/1.8git_build/ if you wish. Customers are using it, and so far, its looking pretty good.

Regards,

Joe

--
Joseph Landman, Ph.D
Founder and CEO
Scalable Informatics Inc.
email: land...@scalableinformatics.com
web : http://scalableinformatics.com
http://scalableinformatics.com/sicluster
phone: +1 734 786 8423 x121
fax : +1 866 888 3112
cell : +1 734 612 4615

Act III:
Conclusions

Machine learning is going to be deployed at massive scale in the next few years.

Five years ago I thought we were doing research to improve safety in some distant future.

This is no longer the case.

Now is the time to study
the security of ML
(and apply ML to security).