INLYSE

How to Detect Malware Using **Computer Vision**

INTRODUCTION

Traditionally, anti-virus companies scour the internet for new computer viruses. They do this by gathering new files, inspecting them, and looking for malicious activities. If these files register malicious behaviors, the anti-virus vendor calculates the fingerprints of the virus "signatures." Those signatures (like a "bad boys" database) are then shared with a customer's anti-virus client/ software.

As the client's anti-virus software receives new files, it calculates its fingerprints and checks them against the known "bad boys" fingerprint database. If a match exists, the software identifies it as a threat and a known computer virus. And this is how the conventional anti-virus systems work.

This approach worked well until attackers changed their strategy of spreading their freshly created malware. A decade ago, attackers spread their newly created malware through the web, trying to reach as many victims as possible. The technology allowed many anti-virus systems to detect those viruses and protect their clients efficiently. But then, research conducted by FireEye[1] in 2013 found that 82% of malware disappears after an hour, and 70% only exists once. This short lifespan means that only a tiny percentage of anti-virus detection signatures catch active threats and the rest just hunt ghosts.

Though some companies have introduced new strategies to combat these variations, they haven't been enough to fully keep up with fast-moving threats, especially now that most viruses arrive directly and exclusively to one single "victim" computer. Anti-virus vendors are left in the dust, not having a chance to analyze the file before it reaches the client. Today, more and more malware programs keep slipping through the gaps and creating havoc in cyberspace.

Clearly, this is a problem for both anti-virus vendors and big and small companies. And it seems a solution remains elusive. What then is the best way to solve this?

Is there a more intelligent way of detecting unknown malware?

[1] https://www.fireeye.com/blog/executive-perspective/2014/05/ghost-hunting-with-anti-virus.html

DEALING WITH **CYBERSECURITY THREATS** AND **DISCOVERING A BETTER SOLUTION**

To tackle this problem, big companies spend hundreds of thousands of dollars annually to employ entire IT-Sec teams to build and maintain the file. They also implement malware analysis systems to analyze and classify every incoming file by themselves. This strategy is costly and requires deep knowledge as those systems are not the most efficient. It is not unusual that a single analysis can take up to 15 minutes, especially in scenarios with several hundred or thousands of analyses per minute.

Over the years, technology has evolved so much that it has become a go-to solution that many companies embrace. The same goes for cybersecurity vendors. Industry experts and leaders leveraged upcoming technologies such as AI to combat cyber threats. However, no matter how promising the AI technology is, it is not good enough to detect malware. And the reason is simply that the AI architecture classifies images. These images are dramatically different from data provided in byte streams representing PDF, MS Office, or PE files.

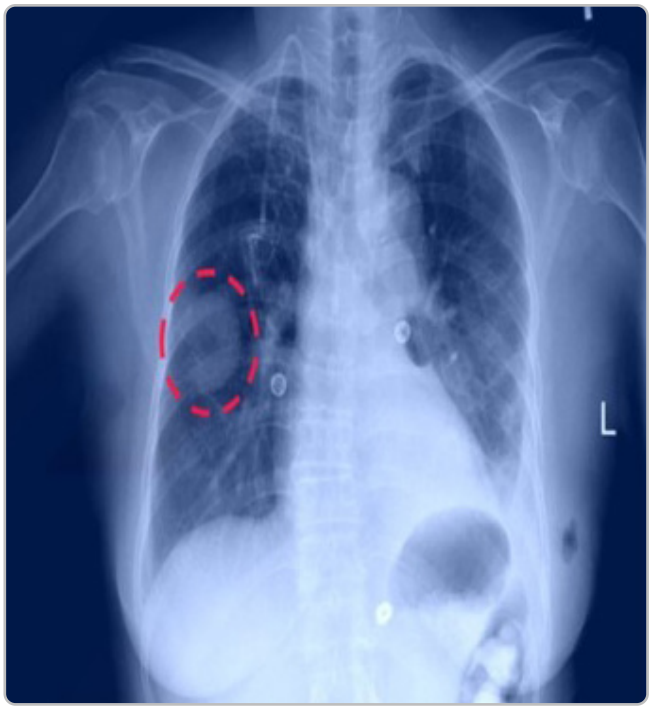
Making those robust AI systems compatible with various data required cumbersome AI architecture adjustments, resulting in lousy classification output. That's why most vendors quickly rejected this approach, but there's always a vendor that's an exception.

Other vendors are quick to shrug off the idea of realigning the AI architecture to solve the problem of unknown malware detection, as it's incredibly challenging and probably impossible. But what if there's a way to alter computer files in a way that fits the string AI architectures as they are?

ADOPTING HUMAN MEDICINE TO FIND A CURE

Human medicine is a simple but fitting inspiration. If a person feels sick, they go to a doctor. The doctor takes an x-ray to identify unusual areas on that picture. While taking an x-ray exam is not enough to complete a diagnosis and find a proper cure, it is good enough to show that something is wrong.

How, then, do AI and human medicine find a "cure" for the lack of unknown malware detection?

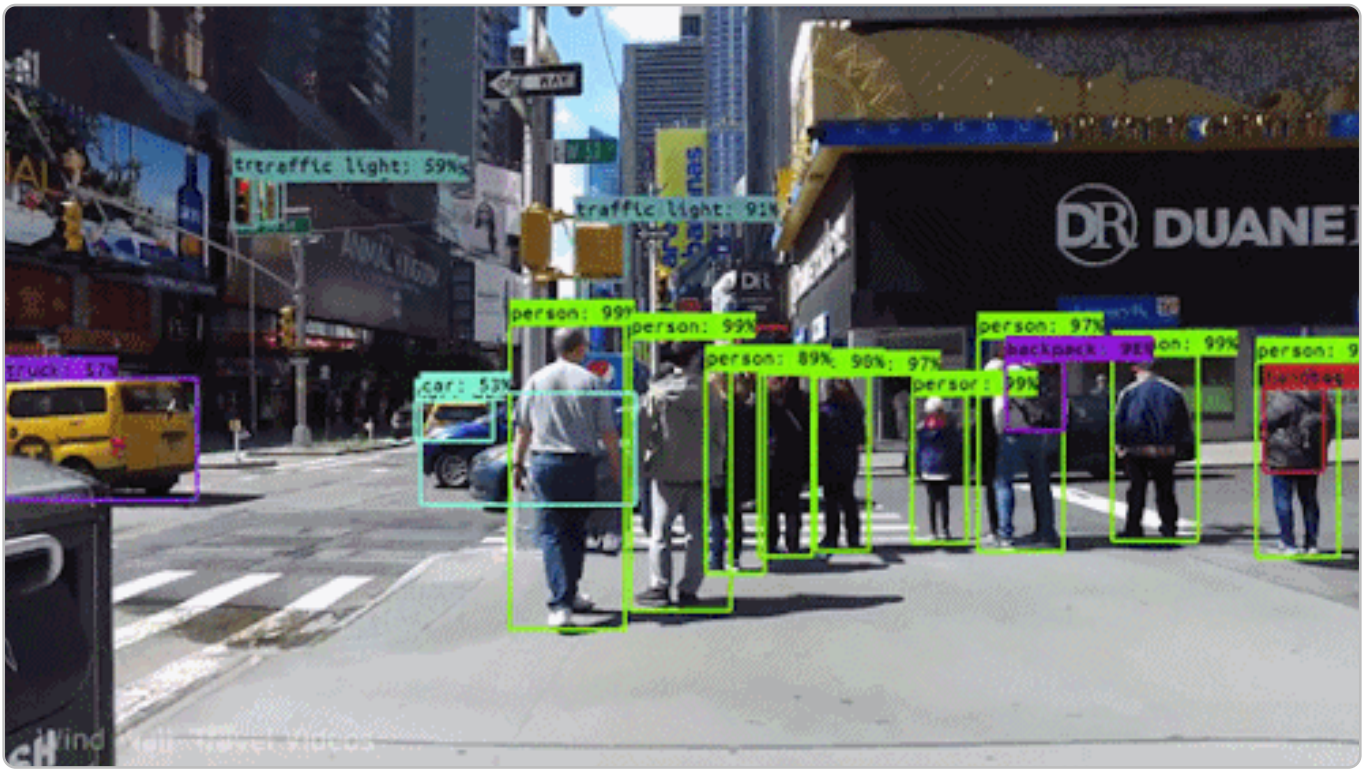


UNDERSTANDING **COMPUTER VISION**

Applying AI (Artificial Intelligence) to the generated images is crucial to understand what Computer Vision is and how it works.

Computer Vision[2] is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos, and other visual inputs — and take actions or make recommendations based on that information.

Computers can understand what is visibly shown in an image. Without it, the computer may know if a



Computer Vision: Identifying human and cars on an image

file is an image or a text file, but it cannot see the objective of the image or text.

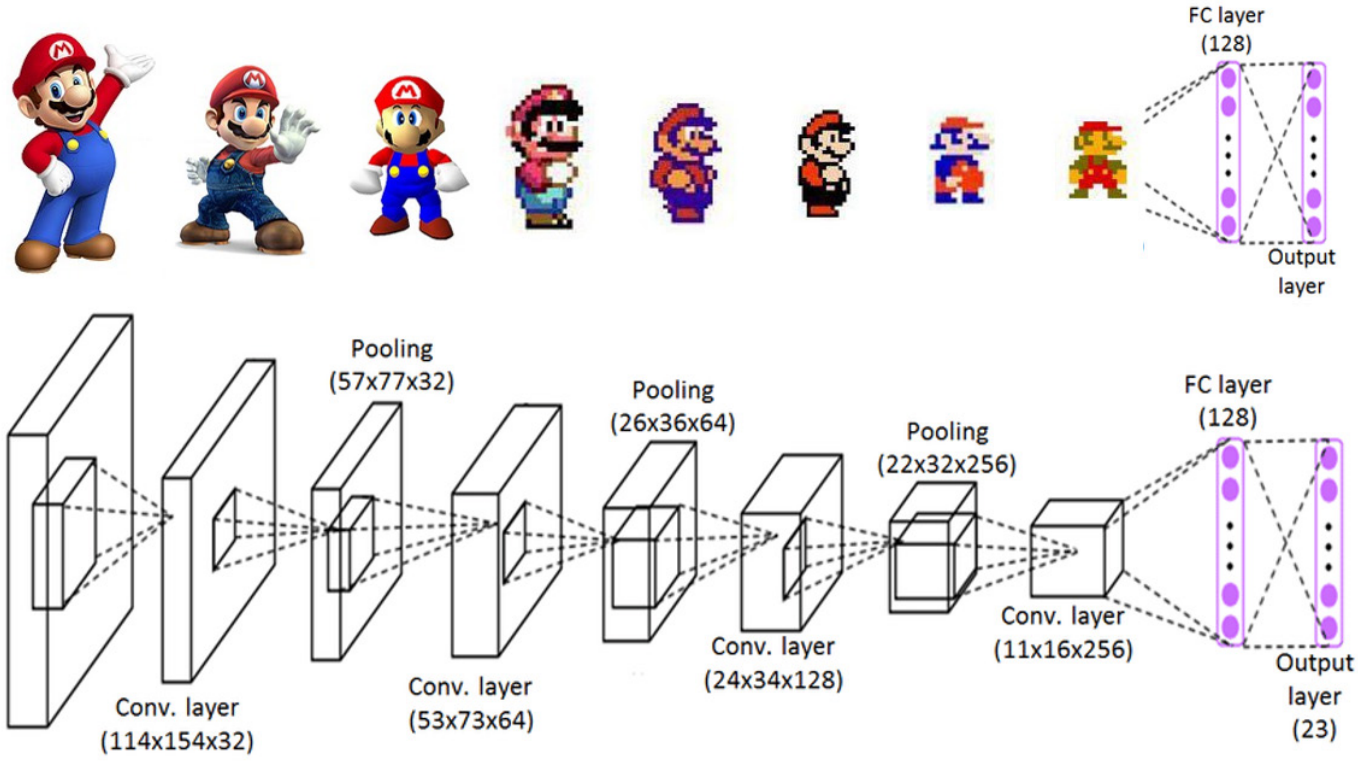
For several years now, Computer Vision has received tremendous attention from the research sector with regular releases of new and more complex state-of-the-art architectures.

But the general idea behind those architectures is simple. All those architectures generalize complex information to achieve a minimal representation, which contains maximum clarity about the original image.

[2] https://ibm.co/3EzVsZf

SUPER MARIO AND **THE CONVOLUTIONAL NEURAL NETWORK**

Everyone knows Super Mario* and may have played it as children and adults. Noticeably, Mario slightly changed and got new details with every new release. Computer Vision aims to bring complex Mario back to its root. For this purpose, Computer Vision uses “Convolutional Neural Networks” (CNNs) that contain multiple layers, and each layer reduces even more details, keeping the most relevant information.



CNN: Reducing unnecessary details for better generalization

Reducing all this information makes it way more accessible for neural network learning to identify any version of Mario as Mario, as all the different versions of Mario will end up in very similar minimal representations.

* This white paper makes use of copyright-protected material for educational purposes only. Mario/Super Mario is a registered trademark of Nintendo Co., Ltd. No copyright infringement is intended.

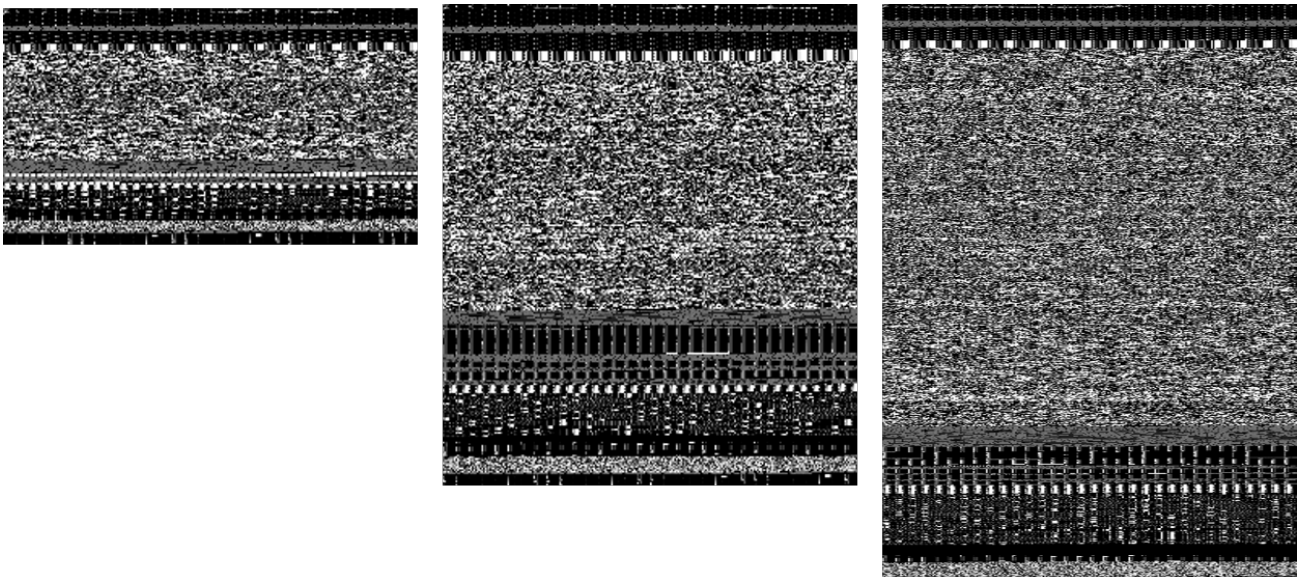
CONVERTING DATA **INTO IMAGES**

Converting data into images is not complex because, in the end, digital data is just a sequence of bits (0s and 1s), and the easiest way is to convert 0s into white and 1s into black pixels followed by a break line after every X (e.g., 256) bits.

ASCII	i							n							l							y							s							e						
Binary	0	1	1	0	1	0	0	1	0	1	1	0	1	1	1	0	0	1	1	0	1	1	0	0	0	1	1	1	1	0	0	1	0	1	1	1	0	0	1	1	0	1	1	0	0	1	0	1
Bit by Bit																																															

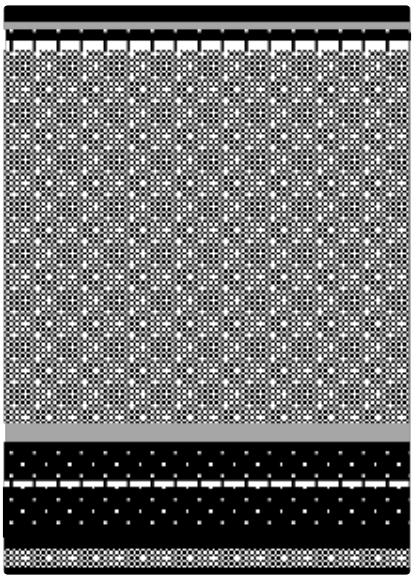
This conversion method will generate black and white images with a width of 256 pixels and a height depending on the input data size.

This method is used to visualize binary data. And even with this very rudimentary conversion technique, exciting insights about the data can surface. Let's look at the Linux tools **id**, **sed**, and **grep**:



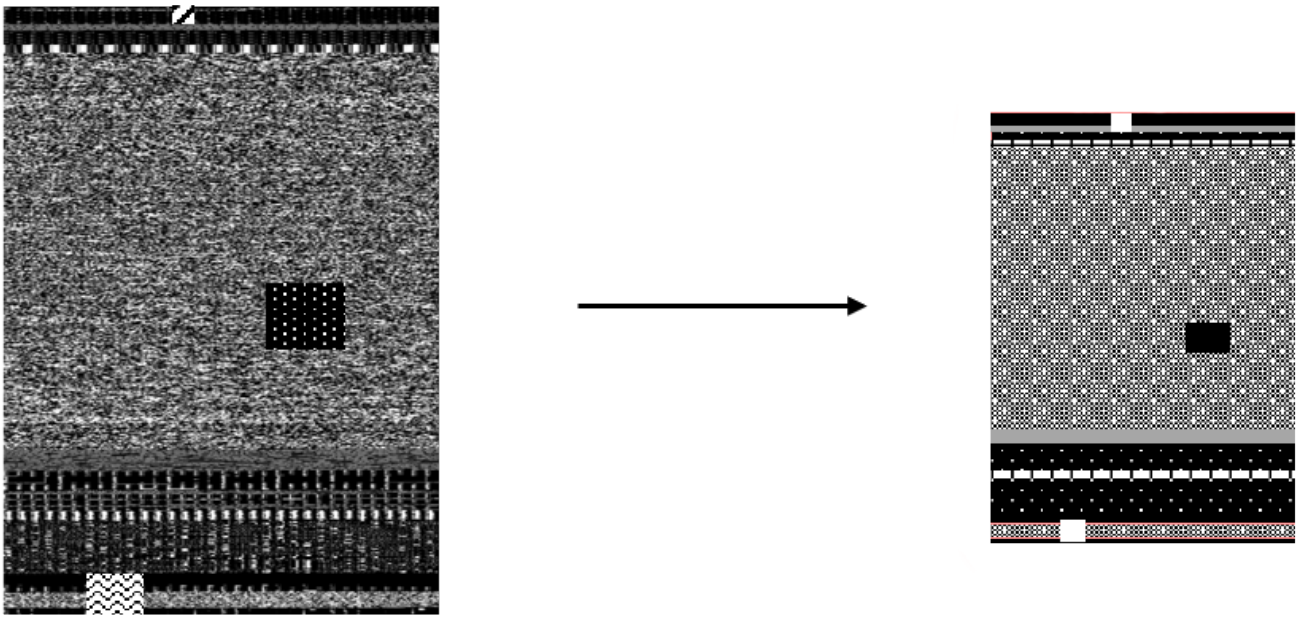
A few things are apparent now. The tool id is smaller than the others, but the similarities of the images are visible. So even with this straightforward encoding, even humans can identify different files of the same file types.

It is now easier to imagine what a CNN is able to see after it generalizes thousands of Linux tools. Something similar like:



All new Linux files must look like this, else the AI CNN will notice that something is different.

The following shows what happens after adding a payload to the data:



The generalized image clearly differs from the usual generalization, and the CNN is able to decide that there is something wrong with that file.

WHY **IMAGE RECOGNITION** IS NOT ENOUGH

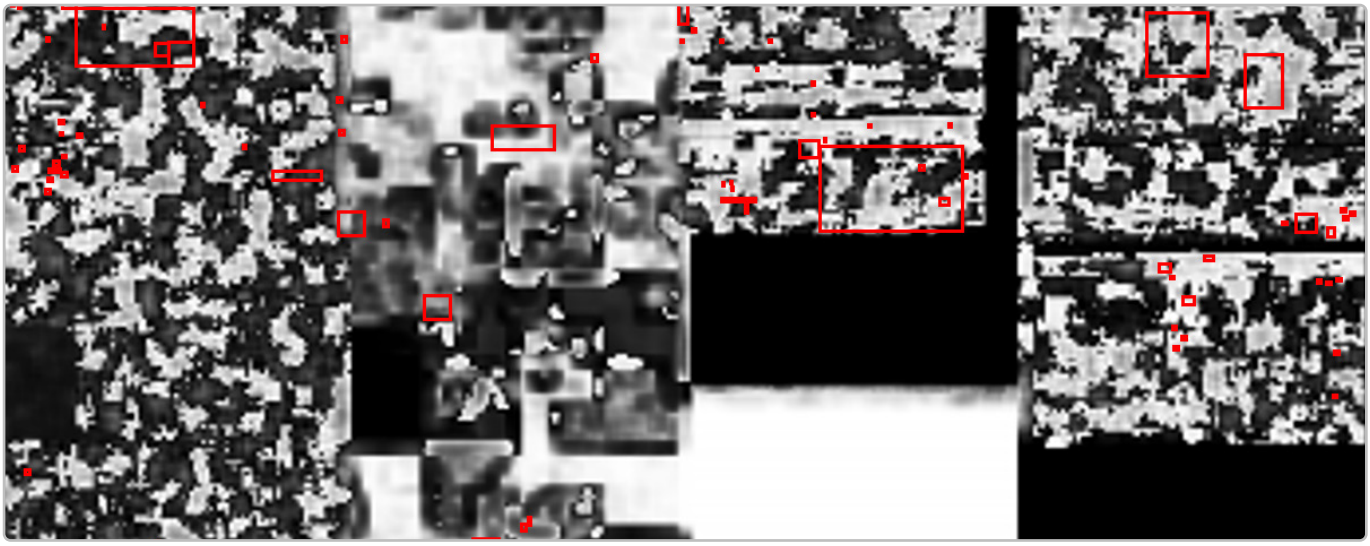
Not a single malware comes “naked” to a client in real-life situations. The attacker always tries to make it as hard as possible for the victim to detect the malware, and attackers use tools to cover up malware.

Some tools or strategies attackers leverage are:

- Obfuscators[3] – make the file challenging to interpret for humans, adding unneeded noise like known benign patterns
- Cryptors - To encrypt some parts of the file
- Protectors
- Different encodings
- Multiple compression stages using packers
- ...and so much more

Additionally, over 90% of attacks start with e-mail. Before the malware reaches the target, the file goes through many systems like Microsoft Exchange or Microsoft Outlook. And there is a whole list of file types blocked by those systems by default[4]. This means that malware must have been hidden inside one of the allowed file types such as Microsoft Office and PDF-Documents. In reality, malware is completely spread within a host file using different encodings, encryption, compression, etc., generating all other and unknown image structures that make it unable to sort them into specific malware family classes.

This presents new problems to using image recognition for malware detection. The images don’t contain any data objectives. Hence, the generated images are not expressive enough to be classified correctly since correlating information are not located next to each other.



Malicious segments highlighted in red represent the malware that has spread within a file

[3] Obfuscation (software) - Wikipedia

[4] Blocked attachments in Outlook (microsoft.com)

ELEVATING IMAGE RECOGNITION **THROUGH CONVERSION**

Generating images where each pixel is located simultaneously within the file through encoding is not enough. A conversion process must be added to identify similar bytes and cluster them at the same region within the file. And this is done by converting data into decimal values, creating 3-grams, and plotting them on a 3D-Scatter plot.

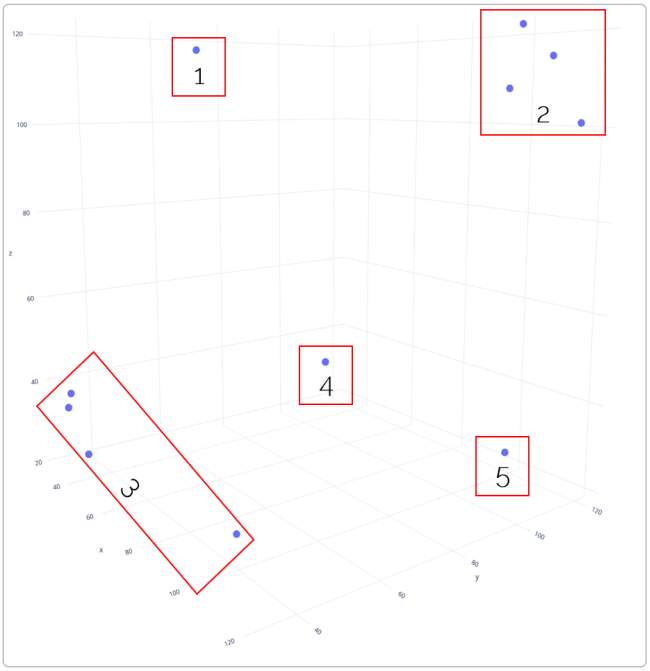
Data can be plotted on similar locations inside the scatter plot with this simple conversion.

Look at the following data example:

ASCII	i	n	l	y	s	e		3	1	3	3	7	!
Decimal	105	110	108	121	115	101	32	114	31	33	33	37	21
3-Grams	(105,110,108)												
nly		(110,108,121)											
lys			(108,121,115)										
yse				(121,115,101)									
se					(115,101, 32)								
e 3						(101, 32,114)							
31							(32 ,114, 31)						
313								(114, 31, 33)					
133									(31, 33, 33)				
337										(33, 33, 37)			
37!											(33, 37, 21)		

Tri-Graph bytes to image encoding

By plotting this data, the following 3D Scatter is generated:



Tri-Graph Image Plot

With this plot, similar information, like the following, can be interpreted and detected:

AREA 1

All points consisting of letters + special chars + numbers

AREA 2

All points consisting of letters only

AREA 3

All points consisting of numbers only

AREA 4

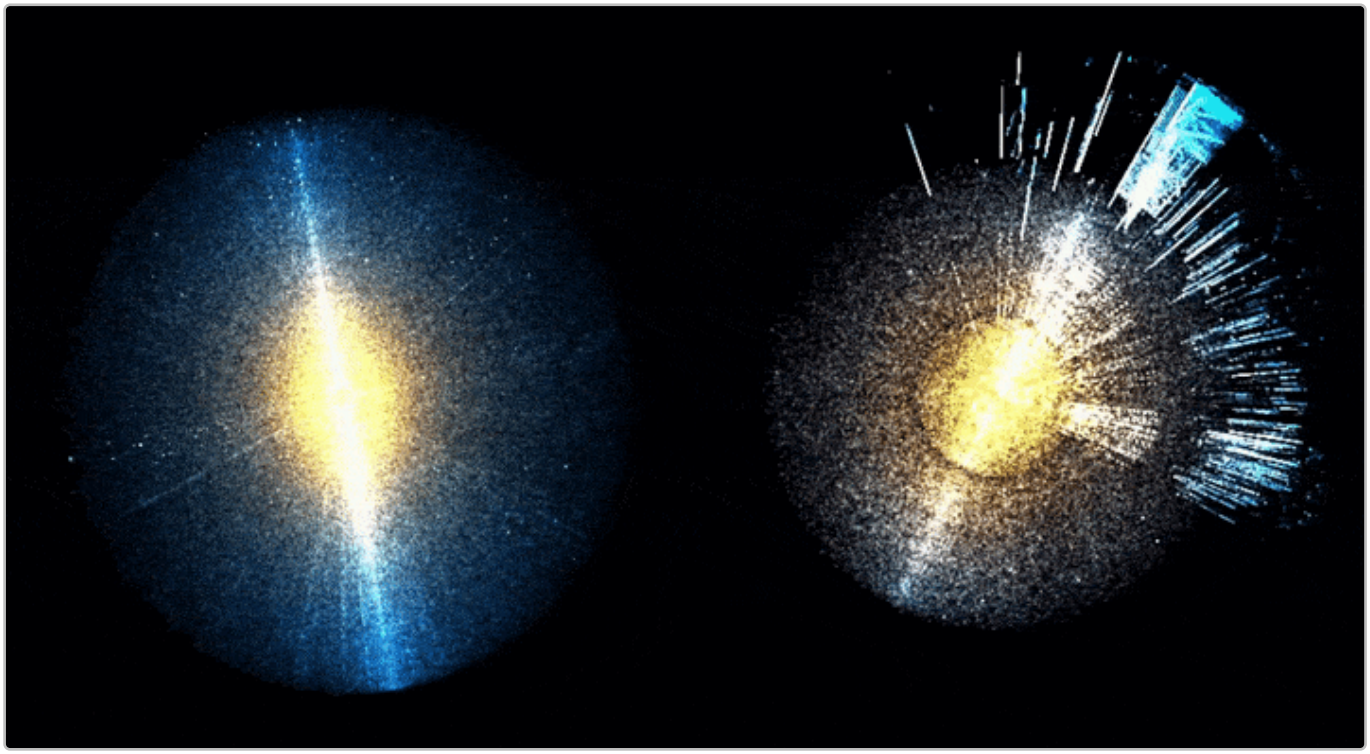
All points consisting of numbers + special characters

AREA 5

All points consisting of letters + special characters

Doing this with more data, adding some colors, and reallocating the axis, can produce results and findings.

The following animation shows two 3D plots of PDF documents. The one on the left is a benign document; the other contains malware.



Left: Represents a “clean” PDF document.

Right: Shows a representation of a file containing malicious structures.

The transformation of the document on the right looks different from the other. It shows a high density of pixels at unusual locations, whereas the left shows a well-balanced pixel density. Without knowing anything about the payload, it is clear that the document on the right contains at least some very strange data, maybe special encodings through obfuscation, compressed information due to packers, etc.

Even if humans can distinguish between usual PDF transformations and malicious PDF transformations, Computer Vision is still the fastest and most accurate.

ENABLE INTELLIGENT MALWARE DETECTION, 24/7