Utilizing Vector Database Management Systems in Cyber Security

Toni Taipalus, Hilkka Grahn, Hannu Turtiainen and Andrei Costin
University of Jyväskylä, Jyväskylä, Finland

toni.taipalus@jyu.fi
hilkka.grahn@jyu.fi
hannu.ht.turtiainen@jyu.fi
andrei.costin@jyu.fi

Abstract: The rising popularity of phenomena such as ubiquitous computing and IoT poses increasingly high demands for data management, and it is not uncommon that database management systems (DBMS) must be capable of reading and writing hundreds of operations per second. Vector DBMSs (VDBMS) are novel products that focus on the management of vector data and can alleviate data management pressures by storing data objects such as logs, system calls, emails, network flow data, and memory dumps in feature vectors that are computationally efficient in both storage and information retrieval. VDMBSs allow efficient nearest neighbour similarity search on complex data objects, which can be used in various cyber security applications such as anomaly, intrusion, malware detection, user behaviour analysis, and network flow analysis. This study describes VDBMSs and some of their use cases in cyber security.

Keywords: Vector Database, Anomaly Detection, Traffic Analysis, Cyber Security, Phishing Detection

1. Introduction

Vectors as a data representation method have gained popularity with large language models, reverse image searches, and recommendation systems (Li, 2023). Effectively, almost all types of data objects, such as text, images, and video, can be represented as vectors. This popularity stems from the inherent versatility of vectors, which allow complex data structures to be expressed in a mathematical form, enabling efficient processing and analysis (Taipalus, 2024). For example, in the realm of large language models, vectors serve as a fundamental representation of words, sentences, or entire documents, capturing semantic relationships and contextual information.

Although vectors have been widely utilized in cyber security in contexts such as machine learning classification, vector database management systems (VDBMS) have emerged in the early 2020s as dedicated systems for managing vector data. Similarly to relational DBMSs, VDBMSs provide features that automate much of the work in managing data. Because both vectorization and DBMS features are relatively mature and well-understood, the relatively novel VDBMSs have quickly established themselves as trustworthy pieces of software used in various domains.

The landscape of cybersecurity data has expanded considerably, mirroring the growth observed in other domains like large language models. Conventional frameworks and algorithms for vector management prove inadequate when confronted with the sheer magnitude of these datasets (Wang et al., 2021). In response to these challenges, vector databases have emerged as a superior alternative, demonstrating faster computational speed and richer features. These advancements in vector database technology address the limitations inherent in previous systems, particularly their ability to effectively handle the volume of information inherent in cybersecurity datasets.

In this study, we describe vectors as means of representing different data objects such as emails, network traffic, and biometric image data, how VDBMSs facilitate vector data management, and most importantly, how VDBMSs can be utilized in various cyber security related use-cases such as biometric authentication and email phishing detection. We also provide examples of established Python libraries for data preprocessing, normalization, feature extraction, and vectorization. Notably, Python is not the only programming language with such libraries.

The rest of the study is structured as follows. In the next section, we describe VDBMS fundamentals, features, and products, and in Section 3, we detail four VDBMS use cases in cyber security. Section 4 concludes the study.

2. Vector Database Management Systems

For vector databases, vectors are effectively represented as ordered lists of numbers, e.g., [0.1, 7.0, -2.9]. This simple vector could represent a point in space with corresponding coordinates in a three-dimensional Cartesian coordinate system, or the vector could represent the overall hue of a photograph, depending on what type of data object has been *vectorized*, i.e., converted into a *feature vector* (Wang et al., 2021). Although this example vector consists of three dimensions or elements, vectors can hold thousands of dimensions.

For reverse image search applications, images are transformed into vectors, allowing similarity comparisons based on vector distances. Recommendation systems leverage vectors to encapsulate user preferences and item characteristics, enabling personalized and computationally efficient content suggestions. The ability to convert diverse data types, such as text, images, and video, into vector representations facilitates interoperability between different data objects. For example, the same types of queries may be used to retrieve textual and image data. As advancements in vector representation methodologies continue, vector representations are increasingly used in different contexts.

VDBMSs are a type of DBMS designed to manage vector data. Like other types of DBMSs, such as relational DBMSs, VDBMS provides means to efficiently store and retrieve vector data and provide access and concurrency control, query optimization, and database scalability. Additionally, VDBMSs offer several advantages in handling vector data over traditional relational DBMSs. One key strength lies in their ability to perform vector operations, enabling simultaneous processing of multiple elements within a vector.

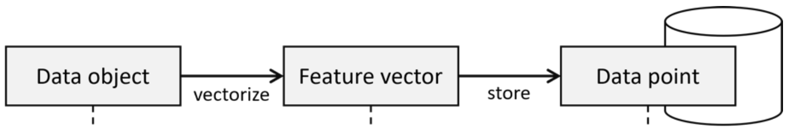


Figure 1: A simplified example of transforming and storing an event or data-of-interest into VDBMS for efficient data management, such as indexing, querying, scalability, and access control.

VDBMSs are designed to support complex vector operations, making them well-suited for applications where mathematical computations on vector data are prevalent. Query optimization in VDBMSs involves exploring specialized optimization techniques for vector operations. This includes optimizing vector aggregations, joins, and filtering to streamline query execution. By tailoring optimization strategies to the unique characteristics of vector data, VDBMSs can achieve better query performance compared to general-purpose DBMSs when dealing with vector-centric workloads. Popular VDBMSs include products such as Pinecone, Milvus (Wang et al., 2021) and Chroma. There are also several libraries for vector operations, such as FAISS and Annoy, but they do not provide many of the DBMS features listed above. Several other DBMSs, such as PostgreSQL, Redis, and SingleStore, have also adopted features for managing vector data (Taipalus, 2024).

Contrary to queries typical for relational and NoSQL databases, vector queries search for vectors that are *approximate nearest neighbours* of the query vector (Ge et al., 2013). If the query vector represents a point in three-dimensional space (e.g., [0.1, 7.0, -2.9]), the VDBMS can search for vectors in the database that are closest matches to the query vector. Depending on the use case, the VDBMS can return one or several near-neighbour vectors with different nearness criteria. For example, suppose the query vector is the end user's current position on a map, and the end user is searching for the nearest restaurants. In that case, the VDBMS may return the ten closest restaurants, but only within a one-mile radius. If the query vector is the end-users freshly scanned retina, the VDBMS may return zero or one vector that matches the query vector; zero returned vectors resulting in denied access.

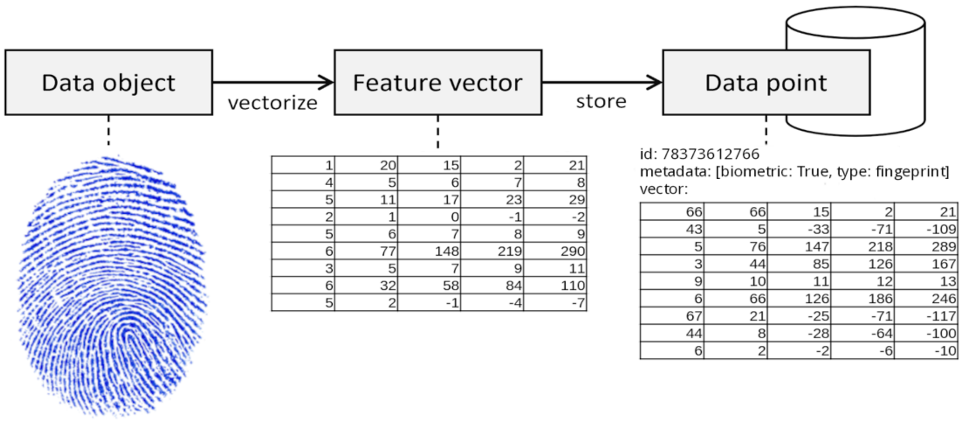
3. Use-Cases in Cyber Security

There are several potential use-cases for VDBMSs in cyber security. In this section, we describe some of such use-case, and provide further information on how to implement these use-cases. This is not an exhaustive list.

3.1 Authentication

Regarding vector data, biometric authentication is closely related to reverse image search: we use an input image to search for similar (or the same) images. The input can be, e.g., a vectorized fingerprint, iris, or retina. Text-based authentication, such as passwords or passphrases, is seldom a feasible use case for vector data due to the simplicity of simply comparing relatively short text strings with each other.

Establishing a database for biometric authentication involves several steps. All input images should have consistent dimensions and reduced noise for the aforementioned biometrics. Feature extraction, i.e., finding meaningful patterns in the images, differs depending on the type of biometrics. In fingerprints, methods such as *ridge detection and orientation* (Zu et al., 2006) may form the features of the vector, or use natively-vectorized approaches such as those presented in Abe & Shinzaki (2015). In irises, it is crucial to extract and segment the iris and analyse its texture with, e.g., *circular Hough transform* (e.g., Cherabit, Chelali & Drejadi, 2012). In retinas, the blood vessels also need to be considered. After extracting the features, they are combined into a vector and often normalized for consistent scaling. For all these steps, Python libraries such as *OpenCV* (Bradski, 2000), *scikit-image* (Van der Walt et al., 2014), *scikit-learn* (Kramer & Kramer, 2016), and *NumPy* (cf. e.g., Oliphant, 2006), when used in tandem, provide functions for all the aforementioned steps.



1	20	15	2	21
4	5	6	7	8
5	11	17	23	29
2	1	0	-1	-2
5	6	7	8	9
6	77	148	219	290
3	5	7	9	11
6	32	58	84	110
5	2	-1	-4	-7

id: 78373612766
metadata: [biometric: True, type: fingerprint]
vector:

66	66	15	2	21
43	5	-33	-71	-109
5	76	147	218	289
3	44	85	126	167
9	10	11	12	13
6	66	126	186	246
67	21	-25	-71	-117
44	8	-28	-64	-100
6	2	-2	-6	-10

Figure 2: A simplified example of transforming and storing an authenticated biometric fingerprint into VDBMS.

Once the vectors have been created, they can be inserted into a vector database managed by a VDBMS. Many available VDBMSs (and other DBMSs with vector features) offer automatic scalability, access control, data encryption, and query optimization. It is worth noting that the system infrastructure often must provide the means to vectorize input images for real-time authentication. That is, VDBMSs often do not provide the means to vectorize data. Some video frames are typically vectorized and organized sequentially into one high-dimensional vector for video-based authentication, such as gait recognition or keystroke dynamics (Schclar et al., 2012).

With voice-based authentication, audio samples need to be cleaned before vectorization. Possible background noise needs to be removed, the amplitudes need to be normalized, and the audio should be captured with a consistent sampling rate. Features in audio include *Mel-Frequency Cepstral Coefficients* (Hasan, Jamil & Rahman, 2004) and formants. Once desired features have been extracted, they are converted into numerical vectors. Once a user needs to be authenticated based on a voice sample, the sample is vectorized and compared to the (previously recorded) vectorized samples in the database to find enough similarity among the query vector and one vector in the database to authenticate the user. Python packages such as *librosa* (McFee et al., 2015) and *scikit-learn* offer required functions.

3.2 Email Phishing Detection

To identify phishing emails, we need a dataset containing emails labelled based on whether they are considered phishing. Following labelling, the next step involves converting these labelled emails into feature vectors using vectorization techniques such as *Bag-of-Words* (e.g., Qader, Ameen & Ahmed, 2019), *Term Frequency-Inverse Document Frequency* (e.g., Christian, Agus & Suhartono, 2016), or *Word Embeddings* (e.g., Liu et al., 2015). When a new email arrives, the system should vectorize it similarly to the initial email dataset and perform a similarity search in the VDBMS to find similar emails. If similar emails have been labelled as phishing emails, the new email should be handled accordingly. Most common VDBMSs allow metadata – in this case, labels – to be stored along with the feature vectors.

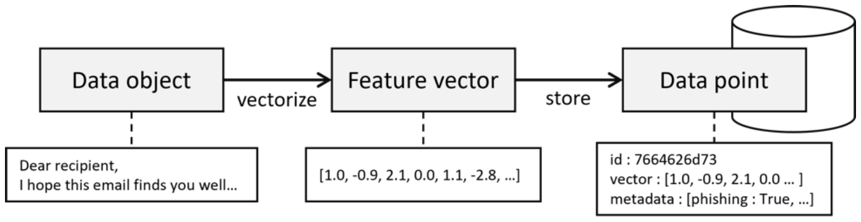


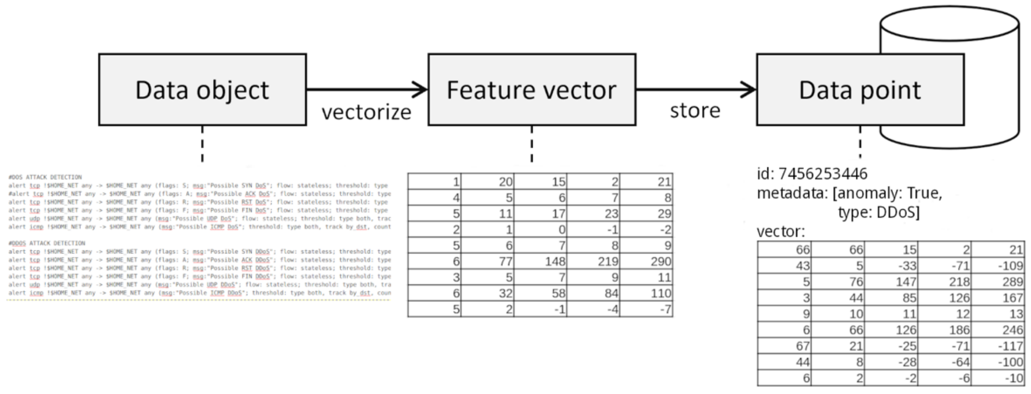
Figure 3: A simplified example of transforming and storing a (potentially phishing) email into VDBMS.

It is worth noting that adjusting the similarity threshold based on the trade-off between false positives and false negatives is crucial. We need to fine-tune this threshold according to domain-specific needs. Additionally, we need to evaluate the performance of this approach using a diverse set of phishing and legitimate emails and iterate and adjust the workflow based on the results. Python libraries such as *scikit-learn*, *Gensim* (Řehůřek & Sojka, 2011), and *NLTK* (Bird, 2006) can preprocess and vectorize even large text data objects.

One alternative approach after vectorization is to apply a machine-learning model for email classification. While the described VDBMS approach does not involve traditional machine learning models, it relies on the idea that similar emails in vector space will likely share similar characteristics. It is a different paradigm compared to machine-learning classification and might be suitable depending on the specific requirements and constraints of the use case. If the system architecture already includes a VDBMS, this approach potentially makes the architecture more efficient and straightforward, automating much of the data management work.

3.3 Anomaly Detection

Anomaly detection can be arduous as many domains require real-time and accurate detections. VDBMSs can be used to detect cyber security-related anomalies. Depending on the use case, different cyber security events can be represented as feature vectors, including information such as IP addresses, protocols, timestamps, file system operations, and executed commands. This approach allows for automated, nearly real-time detection of anomalous behaviour but also requires an initial dataset to be vectorized and used as a reference for both regular and anomalous behaviour. The quality of the initial dataset is paramount, as false positive detections can cause system or data availability issues due to false flagging and possible countermeasures. The baseline accuracy for true positive detections should be very high; meanwhile, the dataset should be large, with several entries for normal and anomalous behaviour. For example, approaches similar to those presented in Subba & Gupta (2021) or Mazzavi et al. (2017) could be used to natively vectorize anomaly detection for host intrusion detection systems.



```
#DOS ATTACK DETECTION
alert tcp !$HOME_NET any -> $HOME_NET any (flags: S; msg:"Possible SYN DoS"; flow: stateless; threshold: type
#alert tcp !$HOME_NET any -> $HOME_NET any (flags: A; msg:"Possible ACK DoS"; flow: stateless; threshold: type
alert tcp !$HOME_NET any -> $HOME_NET any (flags: R; msg:"Possible RST DoS"; flow: stateless; threshold: type
alert tcp !$HOME_NET any -> $HOME_NET any (flags: F; msg:"Possible FIN DoS"; flow: stateless; threshold: type
alert udp !$HOME_NET any -> $HOME_NET any (msg:"Possible UDP DoS"; flow: stateless; threshold: type both, trac
alert icmp !$HOME_NET any -> $HOME_NET any (msg:"Possible ICMP DoS"; threshold: type both, track by_dst, count

#DDOS ATTACK DETECTION
alert tcp !$HOME_NET any -> $HOME_NET any (flags: S; msg:"Possible SYN DDoS"; flow: stateless; threshold: type
alert tcp !$HOME_NET any -> $HOME_NET any (flags: A; msg:"Possible ACK DDoS"; flow: stateless; threshold: type
alert tcp !$HOME_NET any -> $HOME_NET any (flags: R; msg:"Possible RST DDoS"; flow: stateless; threshold: type
alert tcp !$HOME_NET any -> $HOME_NET any (flags: F; msg:"Possible FIN DDoS"; flow: stateless; threshold: type
alert udp !$HOME_NET any -> $HOME_NET any (msg:"Possible UDP DDoS"; flow: stateless; threshold: type both, tra
alert icmp !$HOME_NET any -> $HOME_NET any (msg:"Possible ICMP DDoS"; threshold: type both, track by_dst, coun
```

1	20	15	2	21
4	5	6	7	8
5	11	17	23	29
2	1	0	-1	-2
5	6	7	8	9
6	77	148	219	290
3	5	7	9	11
6	32	58	84	110
5	2	-1	-4	-7

id: 7456253446
metadata: [anomaly: True,
type: DDoS]
vector:

66	66	15	2	21
43	5	-33	-71	-109
5	76	147	218	289
3	44	85	126	167
9	10	11	12	13
6	66	126	186	246
67	21	-25	-71	-117
44	8	-28	-64	-100
6	2	-2	-6	-10

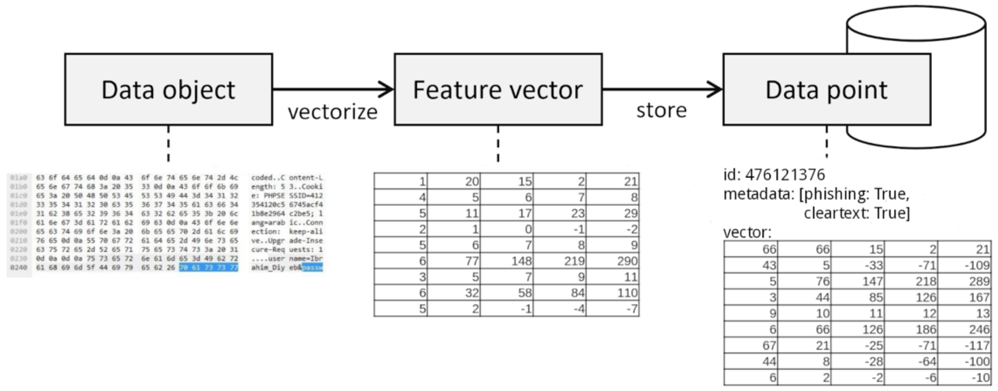
Figure 4: A simplified example of transforming and storing an anomaly alert into VDBMS.

Once the events have been vectorized, they can be inserted into a VDBMS. As new cyber security events occur, their features are vectorized, and the vectors are used as query vectors to search for events with similar characteristics. As with email phishing detection, prepare to tweak the threshold, i.e., how much the query vectors should resemble vectors of anomalous events to categorize them as anomalous events. In all cases, all events should be stored as vectors and labelled anomalous or non-anomalous to be utilized in future searches. The data gathered can be further used to enhance the dataset for better detection through continuous adaptations and feedback mechanisms.

3.4 Network Traffic Analysis

Network traffic analysis plays a pivotal role in cybersecurity by providing an understanding of the data flowing through a network. The purpose is to detect malicious activities and potential security threats before asset damage can occur. By scrutinizing patterns, protocols, and communication flows, it is possible to detect attacks such as distributed denial-of-service (e.g., Lopez et al., 2019) and intrusions (e.g., Gao et al., 2020). Network traffic analysis serves as a proactive defence mechanism, allowing organizations to bolster their cybersecurity defences, respond swiftly to emerging threats, and safeguard the integrity and confidentiality of their digital assets.

Performing real-time network traffic analysis with vectorization involves representing network traffic data as feature vectors. By using tools such as *Wireshark*, *tcpdump*, or *Scapy* (Rohith, Moharir & Shobha, 2018), extract relevant data from raw network packages such as IP addresses, ports, protocols, and packet sizes and convert them into feature vectors. Techniques such as *Bag-of-Words* may be used for categorical data such as protocols, data types, and timestamps can be converted into numerical representations, and numerical features such as packet sizes may be normalized with statistical summaries. For example, approaches such as those of Liu et al. (2017) could natively vectorize network traffic, even in encrypted traffic.



```
01a0  63 6f 64 65 64 0d 0a 43  6f 6e 74 65 6e 74 2d 4c   coded..C ontent-L
01b0  65 6e 67 74 68 3a 20 35  33 0d 0a 43 6f 6f 6b 69   ength: 5 3..Cooki
01c0  65 3a 20 50 48 50 53 45  53 53 49 44 3d 34 31 32   e: PHPSE SSID=412
01d0  33 35 34 31 32 30 63 35  36 37 34 35 61 63 66 34   354120c5 6745acf4
01e0  31 62 38 65 32 39 36 34  63 32 62 65 35 3b 20 6c   1b8e2964 c2be5; l
01f0  61 6e 67 3d 61 72 61 62  69 63 0d 0a 43 6f 6e 6e   ang=arab ic..Conn
0200  65 63 74 69 6f 6e 3a 20  6b 65 65 70 2d 61 6c 69   ection:  keep-ali
0210  76 65 0d 0a 55 70 67 72  61 64 65 2d 49 6e 73 65   ve..Upgr ade-Inse
0220  63 75 72 65 2d 52 65 71  75 65 73 74 73 3a 20 31   cure-Req uests: 1
0230  0d 0a 0d 0a 75 73 65 72  6e 61 6d 65 3d 49 62 72   ....user name=Ibr
0240  61 68 69 6d 5f 44 69 79  65 62 26 70 61 73 73 77   ahim_Diy eb&passw
```

1	20	15	2	21
4	5	6	7	8
5	11	17	23	29
2	1	0	-1	-2
5	6	7	8	9
6	77	148	219	290
3	5	7	9	11
6	32	58	84	110
5	2	-1	-4	-7

id: 476121376
metadata: [phishing: True,
cleartext: True]
vector:

66	66	15	2	21
43	5	-33	-71	-109
5	76	147	218	289
3	44	85	126	167
9	10	11	12	13
6	66	126	186	246
67	21	-25	-71	-117
44	8	-28	-64	-100
6	2	-2	-6	-10

Figure 5: A simplified example of transforming and storing a (potentially phishing) email into VDBMS.

Similarly to some of the use cases presented before, this approach needs initial data to be compared. Again, once the feature vectors have been stored in a VDBMS, network traffic should be continuously vectorized and compared with the existing vectors in the database (Iglesias & Zseby, 2015). Suppose vectorization aims to classify and detect malicious events in the network, it is essential to implement a periodic recalibration based on new data to eliminate false positives and false negatives. This approach serves as a reactive security measure and a proactive tool for network optimization and resource allocation.

4. Conclusion

Storing various data objects as feature vectors has gained popularity due to their computational efficiency in storing and comparing vectors. Additionally, VDBMSs have emerged as systems dedicated to automating tasks such as storing and indexing vectors, facilitating vector querying and query optimization, and database scalability and access control.

The strengths of VDBMS for cyber security are their efficiency in handling large and diverse datasets, providing rapid query response times, and being adept at recognizing non-exact matches. These systems' scalability, speed, and adaptability make them invaluable for cyber security, particularly in dynamic environments where extensive and varied data require quick and flexible analysis.

In this study, we showed through several examples how vector database management systems and various software libraries can be used in the domain of cyber security. In summary, we highlighted the use cases in user authentication, email phishing detection, anomaly detection, and network traffic analysis. However, as almost all data objects can be vectorized, the possibilities of utilizing vector data extend beyond the use cases presented in this study. This prompts further theoretical and applied research on VDBMSs, potentially resulting in interesting immediate applications in highly demanding cyber-security scenarios.

Acknowledgment

Hannu Turtiainen thanks the Finnish Cultural Foundation / Suomen Kulttuurirahasto for supporting his Ph.D. dissertation work and research (grant decision no. 00231412).

The fingerprint image in Figure 2 is by user brokenarts and is available at Freeimages.com.

References

Abe, N., & Shinzaki, T. (2015). Vectorized fingerprint representation using minutiae relation code. In *2015 International Conference on Biometrics (ICB)* (pp. 408-415). IEEE.

Bird, S. (2006). NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions* (pp. 69-72).

Bradski, G. (2000). The openCV library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 25(11), 120-123.

Cherabit, N., Chelali, F. Z., & Djeradi, A. (2012). Circular Hough transform for iris localization. *Science and Technology*, 2(5), 114-121.

Christian, H., Agus, M. P., & Suhartono, D. (2016). Single document automatic text summarization using term frequency-inverse document frequency (TF-IDF). *ComTech: Computer, Mathematics and Engineering Applications*, 7(4), 285-294.

Gao, M., Ma, L., Liu, H., Zhang, Z., Ning, Z., & Xu, J. (2020). Malicious network traffic detection based on deep neural networks and association analysis. *Sensors*, *20*(5), 1452.

Ge, T., He, K., Ke, Q., & Sun, J. (2013). Optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2946-2953).

Hasan, M. R., Jamil, M., & Rahman, M. G. R. M. S. (2004). Speaker identification using mel frequency cepstral coefficients. *Variations*, 1(4), 565-568.

Iglesias, F., & Zseby, T. (2015). Analysis of network traffic features for anomaly detection. Machine Learning, 101, 59-84.

Kramer, O., & Kramer, O. (2016). Scikit-learn. *Machine learning for evolution strategies*, 45-53.

Li, F. (2023). Modernization of databases in the cloud era: Building databases that run like Legos. In *Proceedings of the VLDB Endowment* 16 (pp. 4140–4151).

Liu, J., Fu, Y., Ming, J., Ren, Y., Sun, L., & Xiong, H. (2017, August). Effective and real-time in-app activity analysis in encrypted internet traffic streams. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 335-344).

Liu, Y., Liu, Z., Chua, T. S., & Sun, M. (2015). Topical word embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 29, No. 1).

Lopez, A. D., Mohan, A. P., & Nair, S. (2019). Network traffic behavioral analytics for detection of DDoS attacks. *SMU data science review*, *2*(1), 14.

Mazzawi, H., Dalal, G., Rozenblatz, D., Ein-Dorx, L., Niniox, M., & Lavi, O. (2017) Anomaly Detection in Large Databases Using Behavioral Patterning. 2017 IEEE 33rd International Conference on Data Engineering (ICDE), San Diego, CA, USA, 2017, pp. 1140-1149

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and music signal analysis in Python. In *Proceedings of the 14th Python in Science Conference* (pp. 18-25).

Oliphant, T. E. (2006). *Guide to NumPy*. Trelgol Publishing. USA.

Qader, W. A., Ameen, M. M., & Ahmed, B. I. (2019). An overview of Bag of Words: Importance, implementation, applications, and challenges. In *2019 International Engineering Conference* (IEC) (pp. 200-204).

Řehůřek, R., & Sojka, P. (2011). Gensim - statistical semantics in Python. *Retrieved from genism.org*.

Rohith, R., Moharir, M., & Shobha, G. (2018). SCAPY - A powerful interactive packet manipulation program. In *2018 International Conference on Networking, Embedded and Wireless Systems* (ICNEWS) (pp. 1-5).

Schclar, A., Rokach, L., Abramson, A., & Elovici, Y. (2012). User Authentication Based on Representative Users. In IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 42, no. 6, pp. 1669-1678,

Subba, B., & Gupta, P. (2021). A tfidfvectorizer and singular value decomposition based host intrusion detection system framework for detecting anomalous system processes. *Computers & Security*, *100*, 102084.

Taipalus, T. (2024). Vector database management systems: Fundamental concepts, use-cases, and current challenges. *Cognitive Systems Research*, 85, Article 101216.

Van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., ... & Yu, T. (2014). scikit-image: image processing in Python. *PeerJ*, 2, e453.

Wang, J., Yi, X., Guo, R., Jin, H., Xu, P., Li, S., ... & Xie, C. (2021). Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data* (pp. 2614-2627).

Zhu, E., Yin, J., Hu, C., & Zhang, G. (2006). A systematic method for fingerprint ridge orientation estimation and image segmentation. *Pattern recognition*, 39(8), 1452-1472.