From Behavior to Pixels: A Vision Transformer Approach for Android Ransomware Detection

Satyam Kesharwani*[1,a], Kamaldeep[2,b], and Manisha Malik[3,c]

[1,*]Dept. of Computer Engineering, National Institute of Technology, Kurukshetra, India
[2]Dept. of Media Engineering, National Institute of Technical Teachers Training and Research, Chandigarh, India
[3]Dept. of CSE, Thapar Institute of Engineering & Technology Patiala, India
[a,*]`123102033@nitkkr.ac.in`, [b]`kamal.katyal@yahoo.com`,
[c]`manishamalik53@gmail.com`
*Corresponding author

Abstract. The exponential rise in Android ransomware attacks poses severe security threats, leading to financial losses and privacy breaches for both individuals and businesses. Traditional signature-based detection techniques struggle against evolving ransomware variants, necessitating more advanced and adaptive detection mechanisms. This work proposes an approach that first employs Random Forest classification for structured feature-based analysis. To further enhance detection, JSON reports are transformed into images, and deep learning models, including Convolutional Neural Network (CNN) and Vision Transformer (ViT), are applied for ransomware classification. We extracted behavioral reports from the CuckooDroid sandbox, collecting approximately 2280 ransomware reports and 2000 benign application reports. A Random Forest classifier trained on structured features achieves an accuracy of 99.41%. To further enhance Android ransomware detection, we introduce a novel transformation of JSON reports into RGB images, enabling deep-learning models to classify ransomware patterns. Our CNN trained on RGB images attains 99.76% accuracy, while a CNN trained on grayscale images achieves 99.53%. Further, the ViT model trained on RGB images optimized using the grid search surpasses all models, achieving a peak accuracy of 99.78%.

Keywords: Android, Ransomware, CNN, Dynamic Analysis, RGB Images, Vision Transformer

1 Introduction

Cybercriminals exploit weaknesses in the Android ecosystem, causing massive financial losses and significant disruptions that can affect lives and critical services. According to Cybersecurity Ventures projects, ransomware could inflict more than $265 billion in damages annually by 2031 (1), and the FBI's Internet Crime Complaint Center has reported a sharp increase in these attacks over recent years (2). In the healthcare sector, ransomware attacks have led to severe service interruptions and huge financial setbacks, underscoring the risk to public safety and the overall economy (3). Mobile devices have become a favorite target for these attacks. Due to Android's open nature and inconsistent update systems, many devices remain exposed to exploitation (4). This isn't just a matter of software flaws—these vulnerabilities have real and serious consequences.

Therefore, we propose a novel approach to analyzing and detecting Android ransomware using behavioral analysis and deep learning detection. First, we execute ransomware and benign applications on the CuckooDroid sandbox to generate the JSON files. The file includes features such as the number of flagged files, hidden payloads, dangerous permissions, identification by more than 10 antivirus engines, triggered signatures, total signature severity, VirusTotal positives, and many others. To analyze Android ransomware, we first convert JSON reports into CSV, which acts as an input to the Random Forest model. We also transform the data into RGB and grayscale images, allowing deep learning models like Convolutional Neural Network (CNN) and Vision Transformer (ViT) to detect complex ransomware behaviors.

1.1 Contributions

Our experimental work has the following key contributions:

Pioneering Vision Transformer: To the best of our knowledge, no one in the literature has employed ViT exclusively for Android ransomware detection. We refined the ViT model by performing a grid search to find optimal hyperparameters, such as the learning rate, batch size, and optimizer settings. Using ViT on RGB images, we achieved a remarkable accuracy of 99.78%.

Diverse Data Creation: Our experimental work is based on a comprehensive dataset comprising approximately 2280 Android ransomware and 2000 benign reports. To offer a versatile evaluation approach, we prepare the data in two formats: CSV for conventional machine learning and images for deep learning.

Customized Data Transformation: To the best of our knowledge, we are the first ones to convert CuckooDroid JSON reports directly to visual formats (both RGB and grayscale images). Converting JSON reports into images lets deep learning models detect subtle spatial and structural patterns in the data to improve detection performance.

Comparative Evaluation: We carry out a detailed comparison using traditional Machine Learning (ML) and advanced Deep Learning (DL). For example, we apply ViT to RGB images and yield an accuracy of 99.78%. We also employ CNN on RGB and grayscale images, achieving an accuracy of 99.76% and 99.53%, respectively. We also apply the Random Forest algorithm to CSV data and achieve an accuracy of 99.41%.

2 Literature Survey

Sharma *et al.* (5) performed static analysis and developed an ensemble-based supervised machine learning framework for Android ransomware detection. Their work combined multiple classifiers to produce a robust prediction model with an accuracy of 99.67%. Kalphana *et al.* (6) proposed a hybrid model that uses an AlexNet deep saliency adaptive classifier - optimized through Squirrel Search Optimization - to accurately detect ransomware on Android devices with 99.89% accuracy. Simultaneously, the model incorporated a hybrid cryptographic scheme combining Homomorphic ECC and Blowfish encryption to secure nonmalicious data in the cloud.

Kirubavathi & Anne (7) performed the static analysis and utilized Pearson correlation to reduce an initial set of 331 permissions down to 24 key features. They applied a decision tree, random forest, extra tree classifier, and light gradient boosting machine—and achieved a detection accuracy of 98.05%. Ciaramella *et al.* (8) transformed the executable into RGB images by mapping opcodes, which were then fed to Le-Net, AlexNet, and VGG-16. Their experiments on a dataset of over 15,000 images (covering generic malware, ransomware, and trusted files) demonstrated that the VGG-16 model attained the highest performance, with an accuracy of approximately 96.9%.

Alazab *et al.* (9) combined a multi-solution binary JAYA algorithm with single-solution simulated annealing to optimize feature selection, oversampling via SMOTE, and support vector machine classifier, achieving an accuracy of 98.7%. Vali *et al.* (10) presented an ensemble machine learning classifier for Android ransomware detection that combines both static features (including permissions, intents, and API calls) and dynamic features (derived from network traffic flow). They employed multiple classifiers—such as decision trees, k-nearest neighbors, random forest, gradient boosting, and bagging. They achieved 98.7% detection accuracy.

Li *et al.* (11) extracted static features including permissions and API calls from decompiled APK files and converted them into grayscale images. They employed CNN and achieved an accuracy of 99.01%. Manzil & Naik (12) used the hamming distance-based feature selection technique to enhance static analysis. They extracted static features such as permissions and intents from AndroidManifest.xml files and converted them into binary feature vectors. They calculated the Hamming distance for each feature across all samples and removed those with a distance below a 50% threshold to reduce redundancy in the feature space. They used Random Forest and Decision Tree, achieving a detection accuracy of 99%.

3 Proposed Approach

As shown in Fig. 1, our proposed approach is organized into five main parts: dataset description, dynamic analysis, feature engineering, data transformation, and detection. Our detection algorithm leverages the ViT for effective classification. Algorithm 1 provides a concise overview of the process of the proposed approach using ViT.

3.1 Dataset Description

We collected ransomware APKs and benign APKs from the RansomProber (15) and Androzoo (16) datasets, respectively. Our dataset consists of 4,280 APKs, with 2,000 coming from benign and 2,280 from ransomware applications.

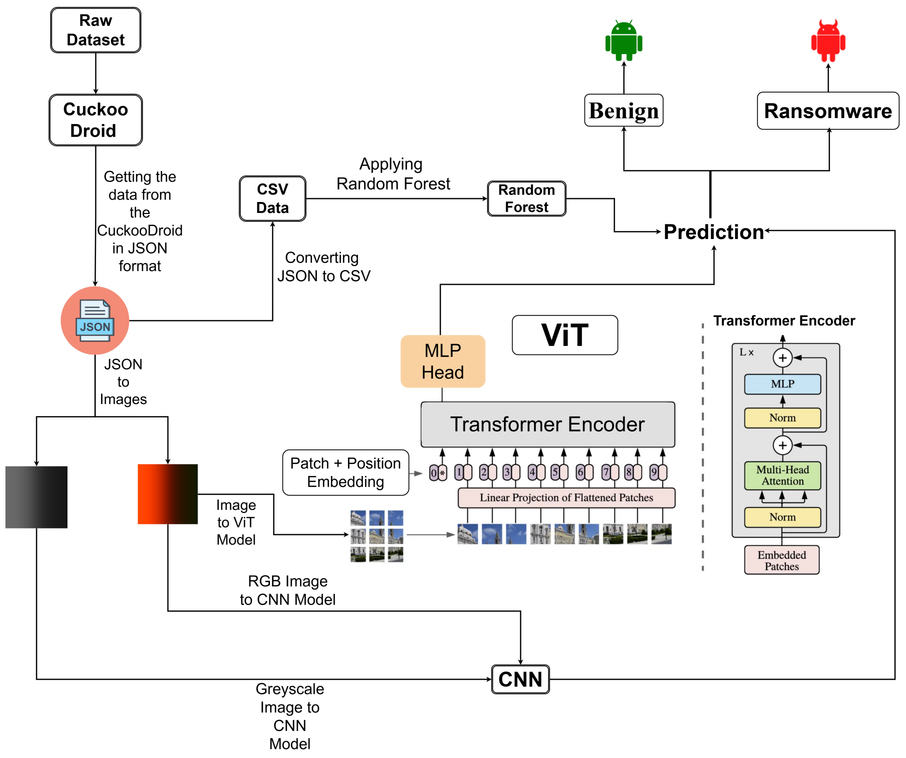


Fig. 1: Workflow of the proposed ransomware detection approach

3.2 Dynamic Analysis of Android Ransomware

We performed dynamic analysis and executed the APKs on the CuckooDroid sandbox (14). This tool runs Android apps in a controlled setting and records the operations in real time. Each report is saved as a JSON file and includes detailed information such as system calls, network activity, and file operations.

```
Algorithm 1 Android Ransomware Detection using ViT
Image Preprocessing:
Input: Input image $I \in \mathbb{R}^{H\times W\times C}$
Output: Sequence of embedded patches $Z = \{z_0, z_1, \dots, z_N\}$
1. Resize the images $I$.
2. Divide $I$ into $N$ non-overlapping patches $\{P_1, P_2, \dots, P_N\}$ of fixed size.
3. Flatten each patch $P_i$ into a vector $x_i \in \mathbb{R}^{P^2\cdot C}$.
4. Project each $x_i$ into a $D$-dimensional embedding using a linear projection: $z_i = W_e x_i$.
5. Prepend a learnable class token $z_0$ to the sequence.
6. Add positional encoding to each element in the sequence.
Transformer Encoder Processing:
Input: Embedded patch sequence $Z = \{z_0, z_1, \dots, z_N\}$
Output: Final encoder output $Z^{(L)}$
7. For $l = 1$ to $L$ do:
    Apply multi-head self-attention:
        $\tilde{Z}^{(l)} = \mathrm{MSA}\big(Z^{(l-1)}\big)$
    Add and normalize (residual connection):
        $Z'^{(l)} = \mathrm{LayerNorm}\Big(Z^{(l-1)} + \tilde{Z}^{(l)}\Big)$
    Apply a feed-forward network:
        $\hat{Z}^{(l)} = \mathrm{MLP}\big(Z'^{(l)}\big)$
    Add and normalize:
        $Z^{(l)} = \mathrm{LayerNorm}\Big(Z'^{(l)} + \hat{Z}^{(l)}\Big)$
Classification:
Input: Final encoder output $Z^{(L)}$
Output: Predicted class probabilities $y \in \mathbb{R}^K$
8. Extract the class token $z_0^{(L)}$ from $Z^{(L)}$.
9. Feed $z_0^{(L)}$ into a classification head to compute logits.
10. Apply sigmoid to the logits to obtain $y$.
```

3.3 Feature Engineering

We extracted nine features, which include the number of flagged files, the count of hidden payloads, the number of dangerous permissions, a binary indicator for hidden payload detection, whether more than 10 antivirus engines identify the sample if it requests dangerous permissions, the number of triggered signatures, the total signature severity, and the number of VirusTotal positives.

3.4 Data Transformation

To make the raw data usable for different analytical approaches, we employed two data transformation strategies:

- **Structured Data for Ensemble Learning:** We converted the JSON reports into CSV files to create a well-organized dataset suitable for ML. This structured data was used to train a Random Forest classifier. Table 1 presents a sample of structured dats. The first four rows correspond to benign samples, while the last four rows correspond to ransomware cases, demonstrating differences in triggered signatures, dangerous permissions, and flagged files.

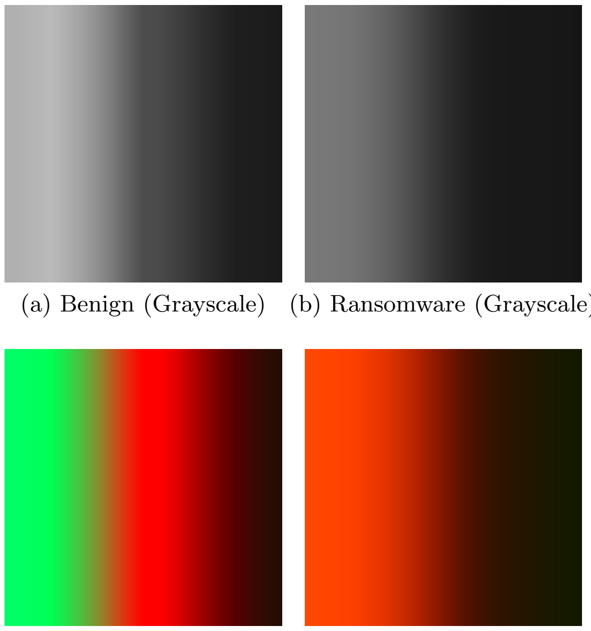
Table 1: Dataset Sample

No_of_VirusTotal_Positives	Total_Signature_Severity	No_of_dangerous_permissions	No_of_flagged_files	label
0	5	2	0	0
1	5	3	0	0
0	5	18	0	0
0	5	17	0	0
38	17	3	3	1
28	20	59	8	1
39	17	11	5	1
30	17	3	3	1

- **Visual Data for Advanced Deep Learning:** In parallel, we transformed the JSON reports into visual formats by generating both RGB and grayscale images that preserve the essential data patterns. By transforming JSON reports into images, we enable deep learning models to recognize complex patterns and structures that are not evident in a traditional tabular format. This approach allows CNN and ViT to leverage spatial and textural information, enhancing their ability to detect ransomware with greater accuracy. Figure 2 illustrates these transformations: the top row displays the grayscale images, while the bottom row shows the RGB images. In both rows, the left column corresponds to benign apps, and the right column corresponds to ransomware behavior.The images generated from the JSON reports were resized to uniform dimensions (224x224 pixels) suitable for our deep learning architectures.

3.5 Detection of Android Ransomware

Our approach is built entirely on the ViT model (17) for image classification, which reimagines image recognition by breaking images into patches and treating



(a) Benign (Grayscale) (b) Ransomware (Grayscale)

(c) Benign (RGB) (d) Ransomware (RGB)

Fig. 2: Visual Representation of JSON Reports

them like words in a sentence. The ViT model views an image as a series of small patches. An image $I \in \mathbb{R}^{H \times W \times C}$ is given as an input. We first split this image into M patches, each of size $p \times p$ (with $M = \frac{HW}{p^2}$). Each patch is then flattened into a vector and linearly mapped into a new embedding space, as shown in 1.

$$\mathbf{h}_0 = [\mathbf{t}_{\text{cls}};\ \text{patch}_1 \cdot \mathbf{W}_p;\ \text{patch}_2 \cdot \mathbf{W}_p;\ \ldots;\ \text{patch}_M \cdot \mathbf{W}_p] + \mathbf{P}_{\text{pos}}, \tag{1}$$

where:

- $\mathbf{t}_{\text{cls}}$ is a learnable token used for classification,
- $\mathbf{W}_p$ is the projection matrix that transforms each patch into the embedding space,
- $\mathbf{P}_{\text{pos}}$ represents the positional embeddings that encode each patch's spatial information.

This initial sequence $\mathbf{h}_0$ is then processed by a stack of transformer encoder layers. Each encoder layer has two main components: a multi-head self-attention (MSA) mechanism and a feed-forward network (FFN).Within each encoder layer, the computations are carried out in two steps:

$$\mathbf{h}'_l = \text{MSA}(\text{LayerNorm}(\mathbf{h}_{l-1})) + \mathbf{h}_{l-1}, \tag{2}$$

$$\mathbf{h}_l = \text{FFN}(\text{LayerNorm}(\mathbf{h}'_l)) + \mathbf{h}'_l, \tag{3}$$

where $\mathbf{h}_{l-1}$ is the input to the l-th layer and $\mathbf{h}_l$ is the output after applying the attention and feed-forward steps.

Finally, the model uses the output corresponding to the classification token $\mathbf{t}_{\text{cls}}$ for the final decision. We refined the ViT model by performing a grid search to find optimal hyperparameters, such as the learning rate, batch size, and optimizer settings.

We also evaluated conventional methods for comparison. A CNN model (13) has been employed on RGB and grayscale images derived from the JSON data. In addition, a random forest classifier has been applied to CSV files converted from the behavioral JSON reports.

4 Results and Discussion

Our experiments show that our proposed approach works well in detecting Android ransomware. The performance of each model is summarized in Table 2.

4.1 Experimental Results

After converting the JSON reports into images for our deep-learning experiments, we applied the ViT model, which shows the best results for Android ransomware detection. After careful tuning of its hyper-parameters using grid search—with settings like a learning rate of 0.001, batch size of 64, and no weight decay—the ViT model reached an accuracy of 99.78%. Using CNN on RGB images, we achieved an accuracy of 99.76%, while a CNN on grayscale images performed similarly with an accuracy of 99.53%. We trained a Random Forest model using CSV data created from 9 carefully chosen features. This model reached an accuracy of 99.41%. The results clearly indicate that the proposed method effectively detects Android ransomware. The ViT, with its advanced self-attention mechanism, outperforms other models by capturing global patterns in the data (17).

Table 2: Comprehensive Model Performance Metrics

Model	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
Vision Transformer (ViT)	**99.78**	**99.76**	**99.76**	**99.76**
CNN on RGB Images	99.76	99.56	100.00	99.78
CNN on Grayscale Images	99.53	99.55	99.55	99.55
Random Forest (CSV)	99.41	98.90	100.00	99.44

4.2 Comparison with Existing Work

We summarize the detection accuracies reported by previous work alongside the accuracy achieved by our proposed approach. Figure 3 presents a bar graph of the same data, with our work highlighted in red. As shown, while various methods have reported extremely high accuracies, our ViT model achieves an accuracy of 99.78%, which is competitive with and complements the best results in the literature.

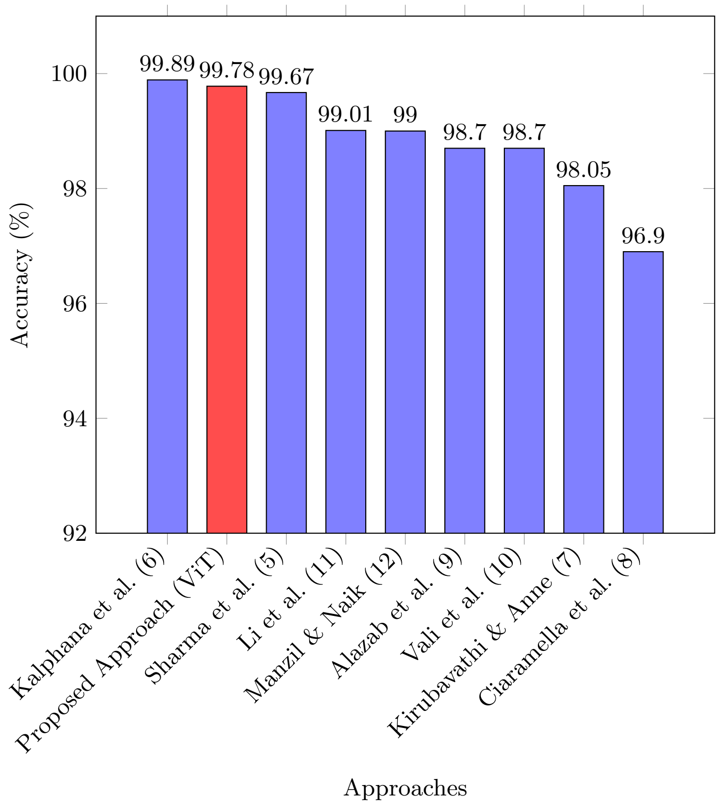


Fig. 3: Comparison of Proposed Approach with the Literature

5 Conclusion and Future Work

We introduced a novel approach for detecting Android ransomware by combining real-time behavior monitoring, image conversion, and modern deep-learning techniques. By turning behavioral JSON reports from the CuckooDroid sandbox into images, our approach uses both ViT and CNN to uncover hidden ransomware patterns. Our experiments showed that the ViT, carefully tuned using a grid search, achieved an impressive accuracy of 99.78%, which shows that our method can serve as a practical, real-world defense, protecting both individual mobile users and businesses from the growing threat of ransomware. In the future, we plan to explore ways to make our model resistant to adversarial attacks.

References

[1] Cybersecurity Ventures. (2023). *Ransomware Damage Report.* Retrieved from `https://cybersecurityventures.com/`

[2] FBI Internet Crime Complaint Center. (2021). *2021 Internet Crime Report.* Retrieved from `https://www.ic3.gov/`

[3] HealthITSecurity. Healthcare Ransomware Attacks Increase. Retrieved from `https://healthitsecurity.com/news/healthcare-ransomware-attacks-increase`.

[4] Android Security Research Group. (2022). *Ransomware in the Android Ecosystem.* Retrieved from `https://androidsecurity.org/`

[5] S. Sharma, R. K. Challa, and R. Kumar, "An Ensemble-based Supervised Machine Learning Framework for Android Ransomware Detection," *The International Arab Journal of Information Technology*, vol. 18, no. 3A, pp. 422–429, 2021.

[6] K. R. Kalphana, S. Aanjankumar, M. Surya, M. S. Ramadevi, K. R. Ramela, T. Anitha, N. Nagaprasad, and R. K. Krishnaraj, "Prediction of Android Ransomware with Deep Learning Model Using Hybrid Cryptography," *Scientific Reports*, vol. 14, 22351, 2024.

[7] G. Kirubavathi and W. Regis Anne, "Behavioral based detection of android ransomware using machine learning techniques," *Int J Syst Assur Eng Manag*, vol. 15, no. 9, pp. 4404–4425, September 2024.

[8] G. Ciaramella, G. Iadarola, F. Martinelli, F. Mercaldo, and A. Santone, "Explainable Ransomware Detection with Deep Learning Techniques," *Journal of Computer Virology and Hacking Techniques*, vol. 20, pp. 317–330, 2024.

[9] M. Alazab, R. Abu Khurma, D. Camacho, and A. Martín, "Enhanced Android Ransomware Detection Through Hybrid Simultaneous Swarm-Based Optimization," *Cognitive Computation*, vol. 16, pp. 2154–2168, 2024, doi:10.1007/s12559-024-10301-4.

[10] N. Vali, A. O. Portillo-Dominguez, and V. Ayala-Rivera, "Enhancing Android Ransomware Detection Using an Ensemble Machine Learning Classifier," *Programming and Computer Software*, vol. 50, no. 8, pp. 562–576, 2024.

[11] D. Li, W. Shi, N. Lu, S.-S. Lee, and S. Lee, "ARdetector: android ransomware detection framework," *The Journal of Supercomputing*, vol. 80, pp. 7557–7584, 2024.

[12] H. H. R. Manzil and S. Manohar Naik, "Android ransomware detection using a novel hamming distance based feature selection," *Journal of Computer Virology and Hacking Techniques*, vol. 20, pp. 71–93, 2024.

[13] S. Garg and S.R. Bhagyashree, 2023, "Improving Efficiency of Spinal Cord Image Segmentation Using Transfer Learning Inspired Mask Region-Based Augmented Convolutional Neural Network." *In International Conference on Data Analytics & Management*, Springer, pp. 245-262.

[14] Cuckoo-Droid. (n.d.). *Cuckoo-Droid Documentation.* Retrieved February 15, 2025, from `https://cuckoo-droid.readthedocs.io/en/latest/`.

[15] Chen, J., Wang, C., Zhao, Z., Chen, K., Du, R., Ahn, G.J., 2017. Uncovering the face of android ransomware: characterization and real-time detection. IEEE Trans. Inf. Forensics Secur. 13, 1286e1300.

[16] Allix, K., Bissyande, T.F., Klein, J., Le Traon, Y., 2016. Androzoo: collecting millions of android apps for the research community. *In International Conference on Mining Software Repositories*, IEEE, pp. 468e471.

[17] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, and D. Tao, "A survey on vision transformer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 87–97, Jan. 2023.