E-ISSN : 2407 – 3911
P-ISSN : 2686 - 0333

IMAGE-BASED MALWARE CLASSIFICATION WITH COMPACT CONVOLUTIONAL TRANSFORMER

Mochamad Adamrasyad Iqbal[1*], Viddi Mardiansyah[2]

Informatics Department[1,2]
Universitas Widyatama
Jl. Cikutra No. 204 A, Bandung 40125
mochamad.adamrasyad@widyatama.ac.id[1], viddi.mardiansyah@widyatama.ac.id[2]

Abstract

The large amount of Internet traffic infected with malware disrupts Internet activity. Undetected malwares can shutdown systems and leak sensitive data. To mitigate such threats, a reliable classification model is required. The Compact Convolutional Transformer (CCT) is a deep learning architecture that combines encoder layers from the Vision Transformer with traditional convolutional layers, and is designed to perform classification effectively even with relatively small datasets. However, when tested using a combination of the MalImg dataset and benign class from DikeDataset, the original CCT model showed signs of overfitting. To address this, this study proposes a modification to the CCT architecture by incorporating an average pooling mechanism parallel to the existing sequence pooling layer. The outputs of both pooling layers are concatenated before being passed to the classification layer. Experiments were conducted under several conditions, including full training without early stopping and without stratified data split. The results show that the modified CCT with average pooling reduces overfitting and improves accuracy under dataset imbalance, indicated by a longer training duration before convergence (8+ epochs), a significant increase in test accuracy up to 6%, and significant decrease of validation loss. However, after stratification and class weighted is applied, the evaluation accuracy of modified CCT decrease by 1.74% from the baseline and the validation loss increasing significantly with p-value 0.0009 ($p < 0.05$), showing that the result is sensitive to the dataset balance.

Keywords:

cybersecurity, deep learning, layer, average pooling, image classification

I. INTRODUCTION

With the rapid development of information technology, threats to computer system security are becoming increasingly complex and damaging. One significant threat is malware, which is malicious software that can distrupt systems, steal information, or damage data. A malware could be in any variant, such as Trojan, worm, backdoor, dialer, downloader, rogue, ransomware, and spyware, each have its unique attack characteristics. Malware attacks in Indonesia have had a serious impact on critical infrastructure, public services, and business organizations.

In June 2024, Indonesia's National Data Center (PDN) was targeted by a malware attack that disrupted government services, including immigration and operations in main airports (Sulaiman & Widianto, 2024). Indonesia Security Incident Response Team on Internet Infrastructure/Coordination Center reports that from January to July 2025, there have been anomalies classified as malware averaging 488.717.545,4 malware, highlighting the urgency of resolving the malware attack issue (Id-SIRTII/CC).

To address this threat, various machine learning-based malware detection methods have been developed. One effective method for small datasets is the compact convolutional transformer (CCT) (Hassani et al., 2021). However, the impact of average pooling on CCT performance for malware classification using the combined MalImg and DikeDataset has not been

thoroughly explored. Additionally, when trained on the MalImg dataset with one class of 552 benign samples from DikeDataset, CCT showed vulnerability to overfitting, so it needed to be modified to be more adaptive in detecting malware variations.

This research utilize MalImg dataset consist images of various malware family, including dialer, backdoor, worm, Trojan, downloader, rogue, and PWS (Nataraj, Karthikeyan, Jacob, & Manjunath, 2011), together with subset of DikeDataset (George-Andrei, 2021), which contains benign software converted to images to improve model generalization. Both were chosen because they are suitable for CCT, which is capable of achieving good performance even when trained with small datasets (Hassani et al., 2021). CCT was chosen because it is a modern deep learning architecture (2021), combining convolutional tokenization and transformer encoders, make it better at extracting feature with small datasets. In addition to theses two datasets, MalwareVision-2025 will be used for cross-dataset validation in this study, but limited to 14 classes due to the RAM limitations of the research tool (16 GB).

To mitigate overfitting, average pooling is incorporated into the CCT architecture to reduce spatial dimensionality and smooth feature representations (Gholamalinezhad & Khosravi, 2020), thereby preventing excessive memorization of training data, allowing the model to generalize better.

Compared to max and min pooling, which emphasize extreme activations and compress feature distributions, average pooling preserves a broader range of activation responses. For limited datasets, this property is advantageous as it reduces overfitting while maintaining sufficient representational capacity, thereby mitigating the risk of underfitting.

Thus, this study aims to compare the original CCT and the modified CCT using average pooling in detecting malware in general, and reducing the risk of overfitting in limited datasets.

II. Related Works

Hassani et al., 2021 introduced the compact convolutional transformer (CCT), a hybrid model combining convolution and transformers to improve efficiency on small datasets. Gholamalinezhad & Khosravi, 2020 reviewed pooling methods (such as average, max, etc.) as regularization for overfitting reduction. Panda et al., 2023, used a CNN-based transfer learning model trained from scratch on domain-specific data, utilizing the MalImg dataset to extract features in malware classification in IoT environments. Mohammadian, Higgins, Ansong, Razavi-Far, & Ghorbani, 2024 used the DikeDataset to evaluate graph neural network (GNN) models for malware detection. The structured labels of the dataset, including crime scores and family classifications, facilitate the training of explainable AI models. Gao, 2023 applied CCT to a small dataset to demonstrate the robustness and efficiency of learning. Current research aims to improve the performance of the CCT architecture with modifications using average pooling and training using MalImg and the DikeDataset subset to reduce overfitting.

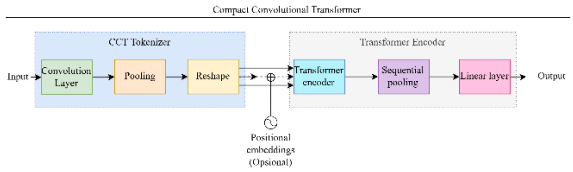


Figure 1. Compact Convolutional Transformer (Hassani et al., 2021)

III. Research Method

The research methodology is quantitative. Quantitative research means using numerical data to fulfill the objectives and conclusions (Arhami, Husaini, Huzaeni, Salahuddin, & F., 2023).

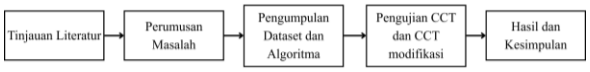


Figure 2. Research steps

The research was conducted in the following steps.

1. Literature review to find topics that have not been explored.
2. Formulation of problems from the topics found.
3. Search for deep learning architecture and suitable datasets.
4. Test configuration and implementation.
5. Analysis and conclusions.

The CCT baseline was taken from the Google Colab code source (Paul, 2023)[1], and then modified with average pooling to become the modified CCT. The baseline CCT and the modified version will then be compared as a test of the effect of average pooling on CCT performance. Furthermore, the modified CCT places average pooling in parallel with sequence pooling at the end of the architecture before the classification layer, so that it is like hybrid pooling, then the output is concatenated to be passed on to the classification layer.

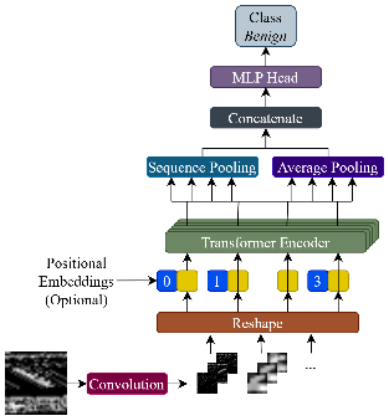


Figure 3. Modified compact convolutional transformer (Inspired by Hassani et al., 2021)

Average pooling was chosen as a modification to reduce overfitting in CCT because it can smooth training data, thereby reducing the influence of outliers and hopefully improving generalization and evaluation accuracy (Zafar et al., 2022). Average pooling that is used in this research is global average pooling. Global average pooling averages overall numbers in one dimension so it results in a number.

$$GAP(X)_d = \frac{1}{L}\sum_{i=1}^{L} X_{b,i,d} \quad (1)$$

b: batch size
L: sequence length
d: projection dimension

This study used Welch's t-test to determine the significance of changes between baseline CCT, before modification, and after modification. The test was chosen because it does not assume equal variances between groups and is robust for small sample sizes, which is suitable for stochastic deep learning experiments where performance variance may differ between architecture versions (Delacre, Lakens, & Leys, 2017b, 2017a).

IV. Analysis and Design

The increase in malware attacks such as ransomware on public institutions in Indonesia highlights the importance of robust detection mechanisms. Compact convolutional transformers (CCTs), a deep learning architecture, can detect malware images through classification. However, there is a tendency for overfitting when trained using datasets such as MalImg with the DikeDataset subset. This can cause poor CCT performance on the test dataset and potentially fail to classify malware correctly. Therefore, analysis is needed to improve CCT performance.

IV.1 Analysis

To reduce overfitting and improve generalization capabilities, average pooling will be used as a CCT modification. For its implementation, several functional requirements have been identified as follows:

1. The program can be run.
2. Malware images can be preprocessed correctly.
3. The model must be able to classify malware images into their families properly.
4. The system must compare the performance between the benchmark CCT and its modified version using average pooling.
5. The program can save the running results.

Meanwhile, the non-functional requirements are as follows.

1. The solution must be able to run on limited hardware and manage data from datasets larger than 1 GB.
2. The RAM must be large enough for model parameters, gradient storage, and batch processing.

[1] https://colab.research.google.com/github/keras-team/keras-io/blob/master/examples/vision/ipynb/cct.ipynb#scrollTo=jkXtM8V47Qiv

The following are the minimum software specifications required.

Table 1. Minimum Software Requirement

Description	Specification
Operating System	64-bit OS
Programming environment	Programming language that supports scientifi computation
Framework	Deep learning framework that supports autodifferentiation and transformer architecture.

The following are the minimum hardware specifications required.

Table 2. Minimum Hardware Requirement

Description	Specification
Processor	64-bit CPU with floating-point capability
Memory (RAM)	16 GB
Primary storage	5 GB

The main dataset used in this study is MalImg with 552 benign samples from DikeDataset (George-Andrei, 2021) (iosifache, 2022). MalImg (TanayBhadula, 2022) consists of 9349 image samples converted from binary malware to grayscale images with varying resolution ratios (Ahmed et al., 2024), while the 552 benign class image samples in DikeDataset are conversions from binary to grayscale images with zero padding if the number of binaries is insufficient to form the square ratio. The main dataset consists of a total of 9901 samples. The secondary dataset used is MalwareVision-2025 (Mohit Chauhan04, Kalyan Kumar, & Prateek Kumar Bhuyan, 2025) in 1024×1024 format, but limited to 14 classes due to RAM limitations (16 GB), resulting in 28771 samples. The secondary dataset is used for test validation.

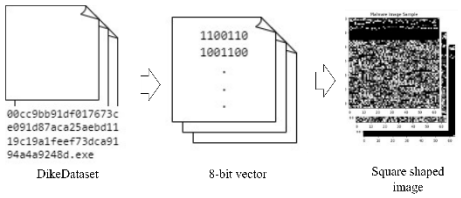


Figure 4. Conversion of benign sample from DikeDataset

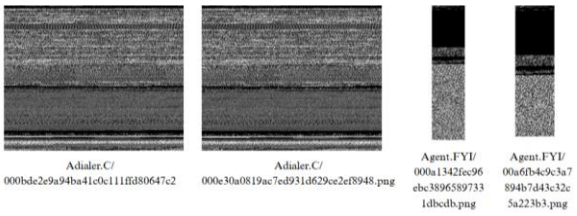


Figure 5. Four samples of MalImg

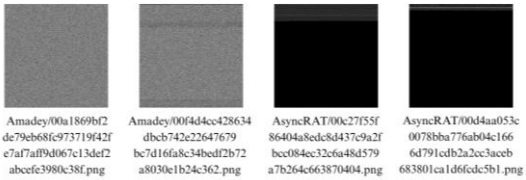


Figure 6. Four samples of MalwareVision-2025

Table 3. Label and Data Type for Each Dataset

Dataset Name	Label	Data Type
Dike Dataset	type, hash, malice, generic, trojan, ransomware, worm, backdoor, spyware, rootkit, encrypter, downloader (George-Andrei, 2021)	Array with dimension of (s, s) (s: side; square shaped)
MalImg	Adialer.C, Agent.FYI, Allaple.A, Allaple.L, Alueron.gen!J, Autorun.K, C2LOP.P, C2LOP.gen!g, Dialplatform.B, Dontovo.A, Fakerean, Instantaccess, Lolyda.AA1, Lolyda.AA2, Lolyda.AA3, Lolyda.AT, Malex.gen!J, Obfuscator.AD, Rbot!gen, Skintrim.N, Swizzor.gen!E, Swizzor.gen!I, VB.AT, Wintrim.BX, Yuner.A s (TanayBhadula, 2022).	Array with dimension of (w, h) (w: width, h: height; various ratios)
MalwareVision-2025-14kelas	Amadey, AsyncRAT, AveMariaRAT, Benign, CobaltStrike, Formbook, GCleaner, GandCrab, Gozi, GuLoader, IcedID, Loki, NanoCore, SmokeLoader	Array with dimension of (1024, 1024)

IV.2 Design

In order to meet the analyzed requirements, the following architectural system was designed:

1. Convolutional tokenizer to generate feature maps from input.
2. Transformer encoder layers for global contextual understanding.
3. Hybrid pooling mechanism: sequence pooling dan average pooling.

The two pooling outputs will then be combined and passed to the classification layer (linear layer).

The algorithm on pseudocode Table 4 begins with the initialization of the dataset and model required for training. The dataset is then divided into training, validation, and evaluation (test) sets. The results of training and evaluation are then saved.

Table 4. Peudocode for Training and Testing

{Procedure to initialize, train, and test the CCT model, and store the result}
Local Dictionary
model:class
dataset:array
epoch_count:integer
train_history, evaluation_metrics:array
Algorithm
Initialize dataset_train, dataset_test, model, epoch_count
CALL prepocess(dataset)
dataset_train, dataset_val, dataset_test ← CALL split(dataset)
FOR 1 to epoch_count DO:
train_history ← CALL train(model,dataset_train,dataset_val)
ENDFOR
evaluation_metrics ← CALL evaluate(model,dataset_test)
CALL save(train_history)
CALL save(evaluation_metrics)

IV.3 Implementation

The preprocessing steps for the MalImg dataset and its DikeDataset subset are as follows:

1. DikeDataset executables are converted into square 2D arrays and zero-padded when necessary.
2. MalImg and sub-DikeDataset images are batched and resized into square images.
3. The resulting image dataset is splitted into train (70%), validation (15%), and test (15%) sets because this ratio is commonly used, as in (Alsumaidaee, Yahya, & Yaseen, 2025; Nazim et al., 2025). Some scenarios use a 60%, 20%, and 20% split because the number of samples is fewer than 1 million, referring to a study by Muraina, 2022.
4. Data augmentation is applied using random cropping and horizontal flipping.

The following are the criteria for the implemented model:

1. The original CCT is retrieved from Google Colab [2] /GitHub, and modified by adding TensorFlow reduce-mean global average pooling in parallel with sequence pooling.
2. As a test variation, some versions used early stopping (patience = 10 epochs) to compare convergence speed with the original CCT.
3. Loss function: categorical cross entropy.
4. Optimizer: Adam (as per the original CCT source code).

Below is hardware used in this research.

Table 5. Hardware Used

Description	Specification
Processor	Intel Core i5-10300H @2.50GHz
Memory (RAM)	16 GB
Primary storage	477 GB
GPU	NVIDIA GeForce GTX 1650

The softwares used in this research are as follow.

Table 6. Software Used

Description	Specification
Operating System	Windows 11
Programming language	Python 3.10
Framework	TensorFlow, Keras, SciPy

The figure below is a screenshot example from model training until saving the result.

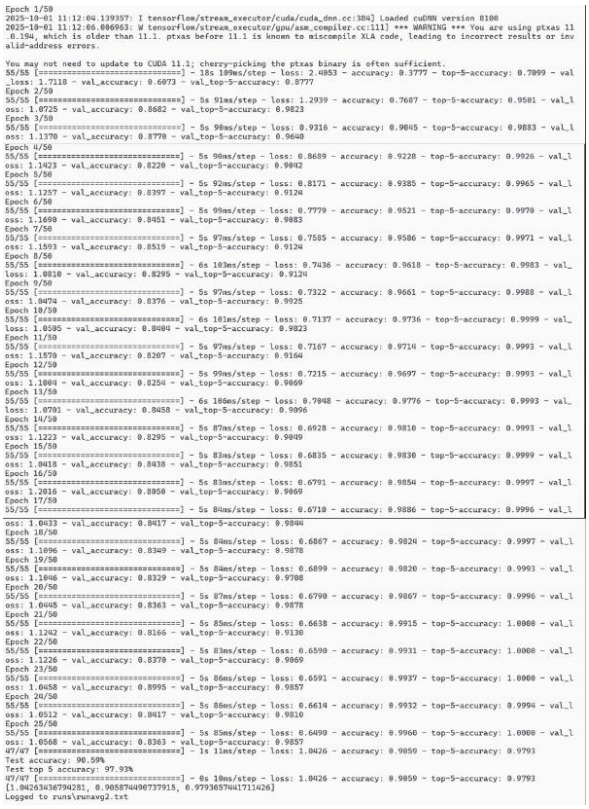


Figure 7. Screenshot of CCT training run

[2] https://github.com/keras-team/keras-io/blob/master/examples/vision/ipynb/cct.ipynb

V. TESTING AND EVALUATION

Testing will be conducted to fulfill the research objectives. The results will be evaluated and conclusions will be drawn. The followings are the tests and evaluation of test results in this study.

V.1 Testing

To fulfill the research objectives, testing will be conducted in three scenarios. Each scenario will include multiple model versions, with each version run 10 independent trials. The following table summarizes the testing scenarios.

Table 7. Three Experiment Scenarios

No	Scenario	Preprocessing	Metrics	Expectation
1	Comparison of Original CCT vs CCT with Average Pooling	Dataset split ratio 70/15/15 (train/validation/test). Dataset: MalImg + DikeDataset.	Average accuracy, top-5 accuracy, number of epochs with early stopping (patience: 10 epochs).	The model is able to generalize better (evaluation accuracy increases) (Code: S1.1). Reduced overfitting (longer number of epochs before convergence) (Code: S.1.2).
2	Analysis of Accuracy & Loss Difference (Overfitting)	Similar to scenario 1.	The average difference in accuracy and loss from train-validation, F1 value, precision, recall, confusion matrix, metrics similar to scenario 1 without early stopping. Welch's t-test was performed.	Reduced overfitting is indicated by a smaller train-validation accuracy gap (Code: S2.1). Reduced overfitting is indicated by a smaller train-validation loss gap (Code: S2.2). The model is better at generalization, indicated by a lower average validation loss (Code: S2.3).
3	Cross-dataset Validation + Stratification + Class Weight	60/20/20 dataset split ratio with stratification. Dataset: MalImg + sub-DikeDataset. Validation dataset: MalwareVision-2025-14 class.	Similar to scenario 2 without the average difference, plus GPU load and training duration.	The model training results across datasets are consistent (Code: S3.1).

From the three scenarios, the following are the testing expectations for each scenario.

V.2 Experiment Result

After testing was implemented based on eight experimental scenarios on the Compact Convolutional Transformer (CCT) variant, 21 versions were obtained, as shown in the following table.

Table 8. Model Versions from Three Scenarios

No.	Scenario	Version
1	1	CCT Original (Baseline)
2		CCT + Average Pooling (Modified)
3	2	CCT Original without Early Stopping
4		CCT + Average Pooling wihout Early Stopping
5	3	CCT Original with Stratification, Class Weight, without Early Stopping
6		CCT + Average Pooling with Stratification, Class Weight, without Early Stopping
7		CCT Original with Stratification, Class Weight, without Early Stopping on MalwareVision-2025-14classes
8		CCT + Average Pooling with Stratification, Class Weight, without Early Stopping on MalwareVision-2025-14classes
9		CCT Original with Stratification, Class Weight, and Early Stopping
10		CCT + Average Pooling with Stratification, Class Weight, and Early Stopping
11		CCT Original with Stratification, Class Weight, and Early Stopping on MalwareVision-2025-14classes
12		CCT + Average Pooling with Stratification, Class Weight, and Early Stopping on MalwareVision-2025-14classes

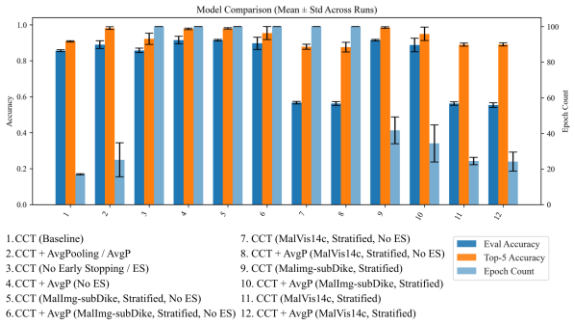


Figure 8. Scenario 1 to 3 result

From the evaluation accuracy results and the number of epochs in Figure 8, the CCT baseline shows reasonable evaluation with fast convergence, while incorporating average pooling improves the accuracy and generalization but longer epoch. However, applying stratified split and class weighting results in reduced performance for the modified CCT on both datasets, primary and secondary.

Table 9. Welch's T-test for Evaluation Accuracy and Average Validation Loss between Versions (Scenario 2 & 3)

Comparison	Metric	T Value	P value	P<0,05
Original CCT and modified CCT	Evaluation Accuracy	-6.9781	5.83×10^{-6}	Yes
	Avg. Validation Loss	7.6308	7.54×10^{-7}	Yes
	Precision	-6.1071	9.9882×10^{-6}	Yes
	Recall	-6.9796	5.8099×10^{-6}	Yes
	F1-score	-6.1618	1.4333×10^{-5}	Yes
Original and modified CCT after stratification and use of class weights	Evaluation Accuracy	1.5272	0.1599	No
	Avg. Validation Loss	-4.7529	0.0009	Yes
	Precision	-0.3447	0.7361	No
	Recall	0.1929	0.8505	No
	F1-score	0.3935	0.7015	No
Original and modified CCT after stratification and use of class weights on MalwareVision-2025-14classes	Evaluation Accuracy	1.2871	0.2170	No
	Avg. Validation Loss	-6.8371	2.126×10^{-6}	Yes
	Precision	1.9622	0.0656	No
	Recall	1.2286	0.2367	No
	F1-score	1.7366	0.1003	No

Based on table 9, the modified CCT significantly improves accuracy and reduces validation loss. However, after applying stratification and class weighting, the modified CCT shows a significant increase in average validation loss compared to the baseline.

Table 10. Average of Distance of Accuracy and Loss between Train and Validation (Scenario 2)

Model	Train-Val. Accuracy	Train-Val. Loss	Val. Loss
CCT (Tanpa Early Stopping)	0.15396	0.57528	1.24222
CCT + Average Pooling (Tanpa Early Stopping)	0.15004	0.45795	1.12037

The modified CCT shows a smaller average accuracy distance and smaller training and validation losses, and the average validation loss shows better generalization, so even without early stopping, average pooling can help stabilize training.

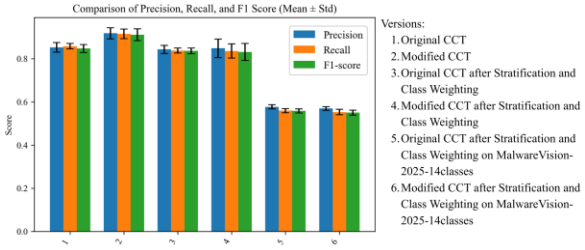
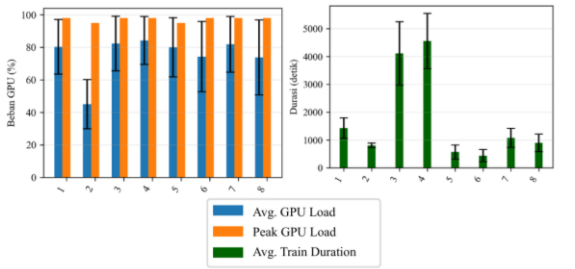


Figure 9. Precision, recall, and F1-score for models in scenario 2 and 3

The modified CCT model shows higher average precision, recall, and F1 scores compared to the original CCT. The relatively small standard deviation values also indicate consistent performance between runs. These results are reinforced by the Welch's t-test in Table 9, which shows that the differences are statistically significant ($p < 0.05$). After stratification, the majority of the modified CCT metrics show a decrease except for precision, which shows a slight increase.



1. Original CCT (w. Stratification & Class Weighting), No Early Stopping (ES)
2. CCT + Average Pooling (w. Stratification & Class Weighting), No ES
3. Original CCT (w. Stratification & Class Weighting) No Es, on MalwareVision-2025-14classes
4. CCT + Average Pooling (with Stratification and Class Weighting) without Early Stopping, on MalwareVision-2025-14classes
5. Original CCT with Stratification, Class Weighting, and Early Stopping
6. CCT + Average Pooling with Stratification, Class Weighting, and Early Stopping
7. Original CCT with Stratification, Class Weighting, and Early Stopping, on MalwareVision-2025-14classes
8. CCT + Average Pooling with Stratification, Class Weighting, and Early Stopping, on MalwareVision-2025-14classes

Figure 10. Average and peak GPU load, and average training duration in scenario 3

The majority of operations peaked at 98%. Without early stopping, the modified CCT with the MalImg dataset and DikeDataset sub-dataset had a lower GPU load and faster training duration compared to the baseline CCT, while the modified CCT with the MalwareVision-2025-14classes dataset experienced a slight increase in GPU load and training duration, but the standard deviation of the training duration was smaller, indicating stability. With early stopping, the modified CCT models on both the MalImg dataset with the DikeDataset sub-dataset and the MalwareVision-2025-14classes dataset show a decrease in GPU load and training duration compared to the baseline CCT, as well as stability reflected in a lower standard deviation. The highest training duration was found in the modified CCT without early stopping on the MalwareVision-2025-14classes dataset, and the lowest was found in the modified CCT with early stopping on the MalImg dataset with the DikeDataset subset.

After observing the overall test results, it was concluded that the expectations of scenarios 1 to 8 would be met as follows.

Table 11. Evaluation of Each Experiment Scenarios Expectation

Code	Fulfilled?	Evidence
S1.1	Yes	Accuracy increased from 85.66% (original CCT) to 89–92% (modified CCT with average pooling)
S1.2	Yes	The average epoch increases from 17 (original CCT) to 22–25 in the modified CCT.
S2.1	Yes	The modified CCT gap is 0.1500, while the original CCT is 0.1539
S2.2	Yes	The modified CCT gap is 0.4579, while the original CCT is 0.5752
S2.3	Yes	Average validation loss of modified CCT: 1.1204; original CCT: 1.2422
S3.1	Mostly	Results are similar between the baseline and modified CCT models across datasets, but the trend shows that the modified CCT results do not improve compared to the baseline CCT results. However, the modified CCT without early stopping on the MalwareVision-2025-14classes dataset shows a slight increase in GPU load and training time compared to the baseline CCT on that dataset.

Average confusion matrices (10 runs) and per-class recall were computed to analyze model behavior beyond scenario-level evaluation. Figures below show results for all models without early stopping.

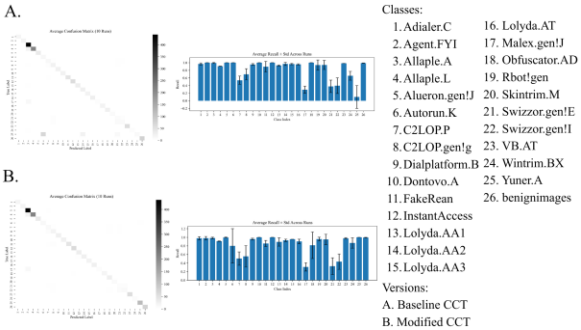


Figure 11. Average confusion matrices for scenario 2

From Figure 11, average confusion matrix of the original CCT (A.) predict mostly true positive with

E-ISSN : 2407 – 3911
P-ISSN : 2686 - 0333

dominance on Allaple.A and several outliers such as Autorun.K, Lolyda.AA2, and Malex.gen!J (worm and Trojan) (Microsoft, 2007, 2009b, 2017a). A similar pattern is observed in the modified CCT (B.), although fewer outliers are present. As for recall per class, the original and the modified one show high recall on Dontovo.A and Instantaccess, while modified one (B.) shows lower recall and bigger standard deviation on Autorun.K and in contrast, the original (A.) shows lower recall and bigger standard deviation on Yuner.A instead (worm)(Microsoft, 2025), which could be caused by imbalance dataset and unstratified dataset split. This marks the urgency of stratified split. Malex.gen!J (Trojan) has low recall on both versions.

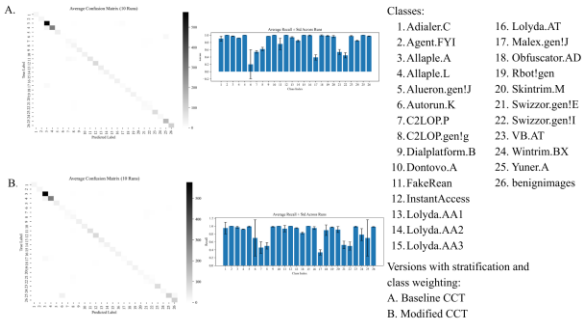


Figure 12. Average confusion matrices and recalls per class after stratified splitting and class weighting

In order to reduce effect of dataset imbalance, stratified dataset split and class weighting is applied. The original CCT (A.) and modified one (B.) both exhibit a bit different outlier on the confusion matrices. As for average recall per class, the original CCT (A.) and modified one (B.) both exhibit similar class-wise recall patterns, with high recall on Agent.FYI, Alueron.gen!J, Dontovo.A, Instantaccess (Trojan, Trojan downloader, and dialer)(Microsoft, 2008a, 2017c, 2017b), while Swizzor.gen!E, Swizzor.gen!I, C2LOP.P, and Malex.gen!J (Trojan downloader and Trojan)(Microsoft, 2008b, 2009a) remain challenging for both models. However, differences can be observed in the variability across runs. The original CCT presents a single prominent outlier with large standard deviation, whereas the modified CCT exhibits elevated variance across multiple difficult classes, suggesting slightly reduced stability under the same stratification and weighting strategy.

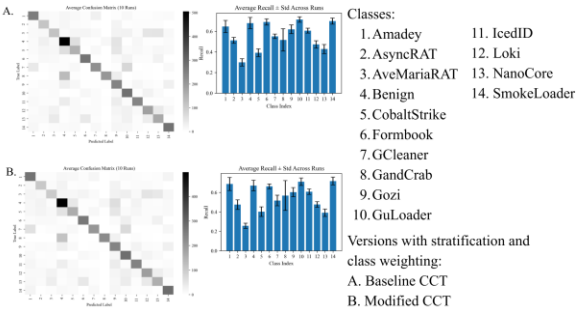


Figure 13. Average confusion matrices and recalls per class after stratification and class weighting on MalwareVision-2025-14classes dataset

Confusion matrices in Figure 13 shows that the original CCT (A.) on the MalwareVision-2025-14classes dataset produces predictions largely concentrated along the true-positive diagonal, with scattered ambiguous misclassifications across classes; true positives dominate in the Benign class. The modified one (B.) shows a similar pattern, with some outliers are lesser than the original one. Average recall per class for the original CCT (A.) and modified one (B.), on the MalwareVision-2025-14classes dataset, both achieve its highest recall on GuLoader, SmokeLoader, Formbook, Benign, and Amadey, while AsyncRAT, Loki, NanoCore, CobaltStrike, and AveMariaRAT have low recall. The modified one (B.) attains its highest recall on SmokeLoader (71.65%) andthe original one (A.) on on GuLoader (71.41%), while both have lowest recall on AveMariaRAT respectively 25.71% and 29.82%

V.3 Evaluation

Incorporating average pooling into the CCT architecture improved generalization, increasing evaluation accuracy by 3–6% and reducing the train–validation gap, indicating mitigated early overfitting. Welch's t-test confirmed these improvements were statistically significant ($p < 0.05$) across accuracy, validation loss, and F1 metrics. However, under stratified splits and class weighting, modified CCT performance gains decreased slightly, and validation loss significantly increased ($p = 0.0009$), indicating sensitivity to data imbalance, while recall per class shows more variance, indicates instability. This indicates that the effectiveness of average pooling is sensitive to data imbalance and class distribution assumptions, which constitutes the main limitation of the proposed approach.

V.4 Limitations

This study has several limitations that should be considered when interpreting the results. First, although stratified splitting and class weighting were applied to mitigate dataset imbalance, other imbalance-handling techniques such as oversampling, focal loss, or cost-sensitive learning were not explored. As a result, minority-class recall instability observed in some scenarios may not be fully addressed by the proposed approach.

Second, the architectural modification was limited to the incorporation of average pooling within the CCT framework. Alternative pooling strategies or hybrid mechanisms were not evaluated, which may offer different trade-offs between generalization and class-wise stability.

Third, the experimental evaluation was conducted on a limited set of malware image datasets. While cross-dataset validation was performed using MalwareVision-2025-14classes, the results may not fully generalize to real-world malware distributions that evolve over time.

Finally, all experiments were conducted under controlled offline training conditions. Deployment-related factors such as concept drift, incremental learning, and real-time inference constraints were not considered and remain open challenges for future work.

VI. Conclusion

This study implemented CCT and evaluated the impact of incorporating average pooling into the Compact Convolutional Transformer (CCT) architecture as modification to mitigate overfitting in image-based malware classification. The modified CCT was compared with the baseline using a combined dataset from MalImg and a subset of DikeDataset, with MalwareVision-2025-14classes as secondary for cross-dataset validation.

Experimental results show that adding average pooling improves generalization, increasing evaluation accuracy by over 4% on average compared to the baseline CCT under dataset imbalance. The modified model achieves up to 91.51% accuracy without early stopping. These gains are accompanied by later convergence and a reduced training–validation accuracy gap, indicating improved training stability. Statistical validation using Welch's t-test confirms that the improvements in evaluation accuracy and validation loss are significant ($p < 0.05$). However, under stratified sampling and class weighting, the modified CCT exhibits a 1.74% decrease in evaluation accuracy and a significant increase in validation loss ($p = 0.0009$), demonstrating sensitivity to dataset imbalance.

In conclusion, average pooling is an effective architectural enhancement for CCT-based malware classification, providing measurable gains in accuracy and stability under standard conditions, but its effectiveness is constrained by data imbalance. Therefore, in practice, implementation of average pooling should consider strategy to overcome dataset oversampling. Future work should focus on imbalance-aware training strategies, alternative pooling mechanisms, and validation on broader malware distributions to improve robustness in real-world scenarios.

Reference

Ahmed, S. R., Mohamed, S. J., Aljanabi, M. S., Algburi, S., Majeed, D. A., Kurdi, N. A., … Tawfeq, J. F. (2024). A Novel Approach to Malware Detection using Machine Learning and Image Processing. *ACM International Conference Proceeding Series*, 298–302. https://doi.org/10.1145/3660853.3660931;SUBPAGE:STRING:ABSTRACT;CSUBTYPE:STRING:CONFERENCE

Alsumaidaee, Y. A. M., Yahya, M. M., & Yaseen, A. H. (2025). Optimizing Malware Detection and Classification in Real-Time Using Hybrid Deep Learning Approaches. *International Journal of Safety and Security Engineering*, *15*(1), 141–150. https://doi.org/10.18280/IJSSE.150115

Arhami, M., Husaini, Huzaeni, Salahuddin, & F., F. Y. R. (2023). *METODOLOGI PENELITIAN UNTUK TEKNOLOGI INFORMASI DAN KOMPUTER*. (L. Mayasari, Ed.) (1st ed.). Yogyakarta: Penerbit ANDI.

Delacre, M., Lakens, D., & Leys, C. (2017a). Why psychologists should by default use welch's t-Test instead of student's t-Test. *International Review of Social Psychology*, *30*(1), 92–101. https://doi.org/10.5334/IRSP.82

Delacre, M., Lakens, D., & Leys, C. (2017b). Why Psychologists Should by Default Use Welch's t-test Instead of Student's t-test (in press for the International Review of Social Psychology). https://doi.org/10.31219/OSF.IO/SBP6K

Gao, A. K. (2023). More for Less: Compact Convolutional Transformers Enable Robust Medical Image Classification with Limited Data.

George-Andrei, I. (2021). *DikeDataset*. Retrieved from https://github.com/iosifache/DikeDataset?tab=readme-ov-file

Gholamalinezhad, H., & Khosravi, H. (2020). Pooling Methods in Deep Neural Networks, a Review. Retrieved from https://arxiv.org/pdf/2009.07485

Hassani, A., Walton, S., Shah, N., Abuduweili, A., Li, J., & Shi, H. (2021). Escaping the Big Data Paradigm with Compact Transformers. Retrieved from https://arxiv.org/pdf/2104.05704

Id-SIRTII/CC. *Laporan Hasil Monitoring*. Retrieved 08/25/2025 from https://idsirtii.or.id/halaman/tentang/laporan-hasil-monitoring.html

iosifache. (2022, March 31). *Dataset with labeled benign and malicious files : r/Malware*. Retrieved 05/31/2025 from https://www.reddit.com/r/Malware/comments/tsv4il/dataset_with_labeled_benign_and_malicious_files/?utm_source=chatgpt.com

Microsoft. (2007, November 19). *Worm:Win32/Autorun.K threat description - Microsoft Security Intelligence*. Retrieved 10/12/2025 from https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Worm:Win32/Autorun.K&threatId=-2147369124

Microsoft. (2008a, June 19). *Dialer:Win32/InstantAccess threat description - Microsoft Security Intelligence*. Retrieved 10/12/2025 from https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Dialer:Win32/InstantAccess

Microsoft. (2008b, August 25). *TrojanDownloader:Win32/Swizzor.gen!E threat description - Microsoft Security Intelligence*. Retrieved 12/26/2025 from https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=TrojanDownloader%253aWin32%252fSwizzor.gen!E&navV3Index=3

Microsoft. (2009a, August 5). *TrojanDownloader:Win32/Swizzor.gen!I threat description - Microsoft Security Intelligence*. Retrieved 10/17/2025 from https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=TrojanDownloader:Win32/Swizzor.gen!I

Microsoft. (2009b, December 4). *Trojan:Win32/Malex.gen!J threat description - Microsoft Security Intelligence*. Retrieved 10/12/2025 from https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Trojan:Win32/Malex.gen!J

Microsoft. (2017a, September 15). *PWS:Win32/Lolyda.AA threat description - Microsoft Security Intelligence*. Retrieved 10/12/2025 from https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=PWS:Win32/Lolyda.AA&threatId=-2147345828

Microsoft. (2017b, September 15). *TrojanDownloader:Win32/Dontovo.A threat description - Microsoft Security Intelligence*. Retrieved 10/12/2025 from https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=TrojanDownloader:Win32/Dontovo.A&threatId=-2147342037

Microsoft. (2017c, September 15). *Trojan:Win32/Alureon.gen!J threat description - Microsoft Security Intelligence*. Retrieved 10/12/2025 from https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Trojan:Win32/Alureon.gen!J

Microsoft. (2025, May 20). *Worm:Win32/Yuner.A!rfn threat description - Microsoft Security Intelligence*. Retrieved 10/12/2025 from https://www.microsoft.com/en-

us/wdsi/threats/malware-encyclopedia-description?Name=Worm:Win32/Yuner.A!rfn

Mohammadian, H., Higgins, G., Ansong, S., Razavi-Far, R., & Ghorbani, A. A. (2024). Explainable Malware Detection through Integrated Graph Reduction and Learning Techniques. Retrieved from https://arxiv.org/pdf/2412.03634v1

Mohit Chauhan04, Kalyan Kumar, & Prateek Kumar Bhuyan. (2025, September 10). *MalwareVision*. Retrieved 12/05/2025 from https://www.kaggle.com/datasets/mohitchauhan04/malwarevision/data

Muraina, I. (2022). Ideal Dataset Splitting Ratios In Machine Learning Algorithms: General Concerns For Data Scientists And Data Analysts.

Nataraj, L., Karthikeyan, S., Jacob, G., & Manjunath, B. S. (2011). Malware images. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security* (pp. 1–7). New York, NY, USA: ACM. https://doi.org/10.1145/2016904.2016908

Nazim, S., Alam, M. M., Rizvi, S., Mustapha, J. C., Hussain, S. S., & Su'ud, M. M. (2025). Multimodal malware classification using proposed ensemble deep neural network framework. *Scientific Reports*, *15*(1), 18006. https://doi.org/10.1038/S41598-025-96203-3

Panda, P., C U, O. K., Marappan, S., Ma, S., S, M., & Veesani Nandi, D. (2023). Transfer Learning for Image-Based Malware Detection for IoT. *Sensors (Basel, Switzerland)*, *23*(6), 3253. https://doi.org/10.3390/S23063253

Paul, S. (2023, August 7). *Compact Convolutional Transformers*. Retrieved 05/01/2025 from https://github.com/keras-team/keras-io/blob/master/examples/vision/ipynb/cct.ipynb

Sulaiman, S., & Widianto, S. (2024, June 28). *Indonesia president orders audit of data centres after cyberattack*. Retrieved 09/26/2025 from https://www.reuters.com/technology/cybersecurity/bulk-indonesia-data-hit-by-cyberattack-not-backed-up-officials-say-2024-06-28/?

TanayBhadula. (2022, October 7). *Image-based Malware Classification using CNN*. Retrieved 01/08/2025 from https://github.com/TanayBhadula/malware-image-detection/blob/main/README.md

Zafar, A., Aamir, M., Mohd Nawi, N., Arshad, A., Riaz, S., Alruban, A., … Almotairi, S. (2022). A Comparison of Pooling Methods for Convolutional Neural Networks. *Applied Sciences 2022, Vol. 12, Page 8643*, *12*(17), 8643. https://doi.org/10.3390/APP12178643