JTERA (Jurnal Teknologi Rekayasa), Vol. 11, No. 1, Juni 2026, Hal. 77-86 p-ISSN 2548-737X
Terakreditasi "**Peringkat 3**" oleh Kemdiktisaintek, Nomor : 10/C/C3/DT.05.00/2025 e-ISSN 2548-8678
DOI: 10.31544/jtera.v11.i1.2026.77-86

Identifikasi Malware pada Jaringan Internet sebagai Tindakan Preventif untuk Ancaman Siber berbasis *Deep Learning*

Husnawati[1*], Rian Rahmanda Putra[2], Faris Humam[3], Ica Admirani[4], Reza Maulana[5]
[1-4]Jurusan Teknik Komputer, Politeknik Negeri Sriwijaya
Jl. Srijaya Negara Bukit Besar Palembang, Indonesia
[5]Magister Ilmu Komputer, Universitas Sriwijaya
Jl. Srijaya Negara Bukit Besar Palembang, Indonesia
husnawati@polsri.ac.id

Abstrak

Keamanan jaringan merupakan aspek krusial bagi penyedia layanan internet (*Internet Service Provider*/ISP), terutama dalam menghadapi ancaman malware yang terus berkembang. Malware merupakan perangkat berbahaya yang dirancang untuk menyerang sistem operasi atau mengeksploitasi kerentanan sistem. Ancaman ini dapat menyebabkan pencurian data penting dan kerugian signifikan bagi pengguna. Salah satu perusahaan yang bergerak di bidang jaringan dan sebagai penyedia layanan internet yang digunakan pada instansi pemerintahan di Sumatera Selatan, menghadapi permasalahan untuk meminimalkan risiko serangan malware pada jaringan yang mereka kelola. Sehingga pada penelitian ini dilakukan penerapan metode *deep learning* dengan algoritma *Convolutional Neural Network (CNN*) untuk mengidentifikasi dan mengklasifikasikan lalu lintas jaringan yang terindikasi malware. Proses dilakukan melalui tahap pengumpulan dataset lalu lintas jaringan, pra-pemrosesan data, pelatihan model CNN, serta evaluasi kinerja model. Hasil pengujian menunjukkan bahwa metode CNN mampu mendeteksi serangan malware dengan tingkat akurasi sebesar 83%, precision 83%, dan recall 82%. Penerapan metode ini memberikan sistem deteksi yang cepat dan akurat, sehingga server dan client dapat lebih waspada terhadap ancaman siber dan terhindar dari pencurian data krusial.

Kata kunci: Malware, ISP, Deep Learning, CNN, ResNet-18.

Abstract

Network security is a crucial aspect for Internet Service Providers (ISPs), especially in the face of the ever-growing threat of malware. Malware is a malicious device designed to attack the operating system or exploit system vulnerabilities. This threat can lead to the theft of critical data and significant losses for users. One of the company networks and internet service providers used by government agencies in South Sumatra faces the challenge of minimizing the risk of malware attacks on the networks it manages. In this study, a deep learning method was applied with the Convolutional Neural Network (CNN) algorithm to identify and classify network traffic indicated by malware proposed. The process was carried out through the stages of collecting network traffic datasets, data pre-processing, training a CNN model, and evaluating model performance. The test results showed that the CNN method was able to detect malware attacks with an accuracy rate of 83%, a precision 83%, and a recall 82%. The application of this method provides a fast and accurate detection system, enabling servers and clients to be more aware of cyber threats and avoid the theft of crucial data.

Keywords: *Malware, ISP, Deep Learning, CNN, ResNet-18.*

I. PENDAHULUAN

Dalam beberapa tahun terakhir penerapan metode deep learning dalam mengklasifikasikan pola serangan pada malware telah banyak dilakukan oleh para peneliti [1]–[3], salah satunya adalah penelitian [4] pada tahun 2023 yang mendeteksi pola serangan end-to-end malware dengan penerapan metode

Automated Machine Learning (AutoML) dan metode *Deep Learning Convolutional Neural Networks (CNNs)*, kemudian penelitian yang dilakukan oleh [5] yang menggunakan metode API Call Graph untuk mendeteksi pola serangan malware.

Dalam beberapa penelitian tersebut deteksi malware pada android menggunakan metode deep learning telah menjadi hotspot penelitian di beberapa tahun terakhir, namun kurang detail dan komprehensif sehingga penelitian tersebut terus dikembangkan hingga saat ini [6], Deteksi malware menggunakan metode deep learning telah menjadi bidang penelitian yang signifikan dalam beberapa tahun terakhir karena meningkatnya ancaman malware yang mengakibatkan kerugian secara finansial dan data [7]–[11]. Tantangan utama dalam pendeteksian malware adalah kemampuan malware untuk berevolusi dengan cepat, sehingga menyulitkan metode pendeteksian secara manual, untuk mengimbanginya. metode *deep learning* terbukti sangat efektif dalam mendeteksi malware karena kemampuannya mempelajari fitur dari kumpulan data besar secara otomatis dan beradaptasi dengan ancaman yang muncul.

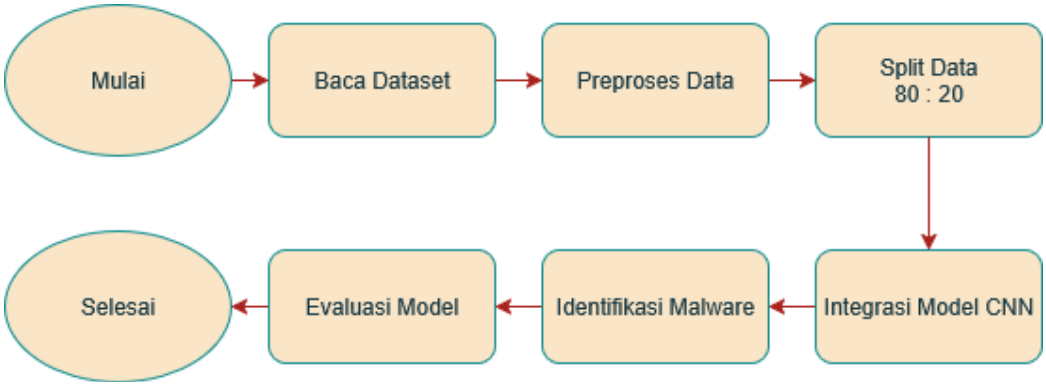
Salah satu masalah utama dalam deteksi malware berbasis *deep learning* adalah kebutuhan akan kumpulan data yang besar dan beragam untuk melatih model secara efektif [10], [12]. Hal ini dapat menjadi sebuah tantangan, terutama ketika berhadapan dengan sifat malware yang terus berkembang, yang dapat menyebabkan kurangnya data yang relevan untuk pelatihan.

Salah satu perusahaan IT di Sumatera Selatan yaitu PT. Mitra Integra Telekom, yang merupakan perusahaan bergerak di bidang penyedia layanan internet, mengalami kendala dalam pendeteksian malware. PT. Mitra Integra Telekom merupakan vendor resmi yang bekerja sama dengan instansi pemerintahan vital di Sumatera Selatan, dalam penyediaan layanan internet dan infrastruktur jaringan untuk mendukung operasional dan kebutuhan komunikasi institusi. Dalam kapasitas ini, PT. Mitra Integra Telekom bertanggung jawab menyediakan konektivitas yang handal untuk mendukung sistem kerja, pelayanan publik, dan akses data internal. Oleh karena itu, keamanan jaringan tetap menjadi tanggung jawab bersama, dan penggunaan solusi keamanan tambahan dari pihak pengguna sangat dianjurkan guna meminimalkan risiko ancaman siber.

PT. Mitra Integra Telekom belum memiliki kemampuan untuk mendeteksi keberadaan malware secara langsung pada lalu lintas jaringan yang dikelola. Sistem yang digunakan oleh perusahaan tersebut saat ini hanya difokuskan pada penyediaan koneksi internet yang cepat dan stabil, namun belum dilengkapi dengan fitur keamanan tingkat lanjut seperti deteksi intrusi atau analisis ancaman siber secara real-time. Sehingga pada penelitian ini dibangun sebuah sistem yang dapat mengidentifikasi Malware pada Jaringan *Internet Service Provider* Sebagai Tindakan Preventif untuk Ancaman Siber menggunakan *Deep Learning* dengan menerapkan algoritma *Convolutional Neural Network* (CNN) untuk menganalisis dan mengidentifikasi serangan malware berdsasarkan jenisnya pada perangkat jaringan, sehingga dengan pendeteksian pola dan identifikasi serangan malware baik, dapat digunakan untuk membentuk pertahanan sistem sebagai tindakan preventif yang akan digunakan dalam keamanan siber.

II. METODE PENELITIAN

Metode penelitian ini disusun secara sistematis untuk menghasilkan model identifikasi malware berbasis deep learning dengan penerapan model CNN. Tahapan penelitian digambarkan melalui diagram alir yang terdiri atas beberapa langkah mulai dari persiapan data hingga evaluasi model seperti yang ditunjukkan pada gambar 1.



Gambar 1. Tahapan Penelitian

Penelitian diawali dengan tahap pembacaan dataset, yaitu proses mengimpor data ke dalam program Python. Dataset yang digunakan merupakan kumpulan data lalu lintas jaringan yang telah diberi label, sehingga dapat dimanfaatkan untuk proses pelatihan dan pengujian model. Selanjutnya dilakukan pra-pemrosesan data guna memastikan data dalam kondisi siap digunakan. Proses ini mencakup normalisasi nilai, penanganan data kosong atau duplikat, serta transformasi data ke dalam format numerik yang sesuai dengan kebutuhan arsitektur CNN.

Tahap berikutnya adalah pembagian dataset (split data) dengan rasio 80:20, di mana 80% dari data digunakan untuk pelatihan model, sedangkan 20% sisanya digunakan sebagai data uji. Pembagian ini bertujuan agar model memperoleh data yang cukup dalam proses pembelajaran, sekaligus tersedia data independen untuk mengukur performa secara objektif.

Data sudah dibagi secara proposional untuk data latih, validasi dan uji. namun masih terdapat ketidakseimbangan pada masing - masing sub kelas. Oleh karena itu, akan dilakukan augmentasi data dengan *weight random sampler* [13] agar data yang digunakan pada tahapan training dapat seimbang yang dapat dihitung pada persamaan (1).

$$P(x_i) = \frac{w_i}{\sum_{j=1}^{N} w_j} \quad (1)$$

Di mana w_i adalah bobot sampel ke-I dan N adalah jumlah total sampel dalam dataset. Selanjutnya tentukan bobot pada persamaan (2), dengan N merupakan total jumlah sample.

$$w_{class} = \frac{N}{n_{class}} \quad (2)$$

Setelah dataset siap, dilakukan integrasi model CNN dengan menggunakan data latih. CNN dipilih karena kemampuannya dalam mengenali pola kompleks dan struktur data yang berlapis, sehingga sesuai untuk mendeteksi karakteristik malware [14]. Model CNN yang terlatih kemudian digunakan dalam tahap identifikasi malware, yaitu memprediksi apakah data uji termasuk kategori malware atau normal. Penerapan model CNN terdiri atas beberapa komponen utama: *convolutional layer*, *activation function*, *pooling layer*, dan *fully connected layer*. Proses konvolusi dilakukan dengan mengalikan matriks data masukan X dengan kernel K, sebagaimana ditunjukkan oleh persamaan (3) sebagai berikut;

$$S(i,j) = (X * K)(i,j) = \sum_m \sum_n X(i+m, j+n) \cdot K(m,n) \quad (3)$$

Hasil konvolusi kemudian melewati fungsi aktivasi ReLU (*Rectified Linear Unit*) untuk memperkenalkan non-linearitas pada persamaan (4) :

$$f(x) = max(0, x) \quad (4)$$

Selanjutnya, hasil ekstraksi fitur diteruskan ke lapisan fully connected untuk menghasilkan skor keluaran. Probabilitas prediksi dihitung menggunakan fungsi *softmax* pada persamaan (5):

$$P(y = j|x) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad (5)$$

Pada persamaan (5) K merupakan jumlah kelas. Dalam penelitian ini terdapat dua kelas, yaitu malware dan normal.

Untuk mengoptimalkan proses pelatihan, digunakan fungsi *loss cross-entropy*, [15] yang umum digunakan pada masalah klasifikasi. Fungsi ini didefinisikan sebagai persamaaan (6) :

$$L = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) \quad (6)$$

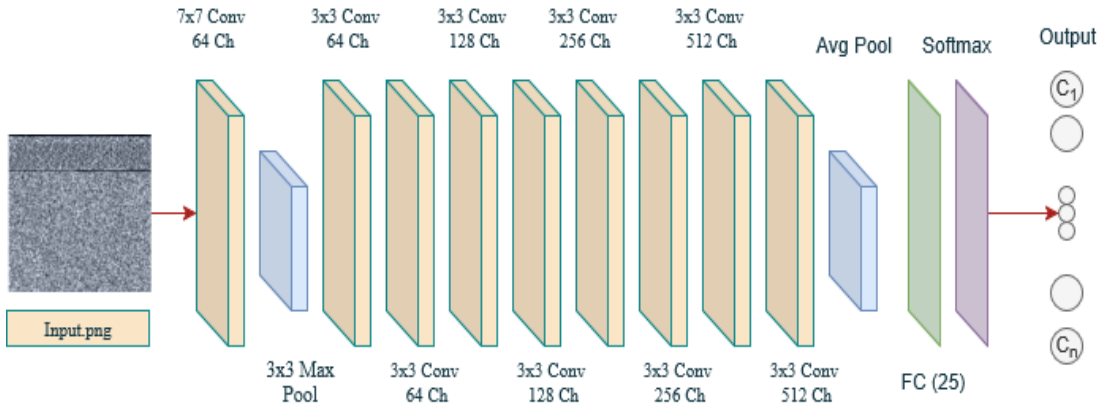
Pada persamaan (6) y_i adalah probabilitas prediksi yang dihasilkan oleh model. Cross-entropy loss berperan penting dalam memperkecil jarak antara distribusi prediksi model dan distribusi label sebenarnya.

Tahap berikutnya adalah identifikasi malware, yaitu proses prediksi pada data uji menggunakan model CNN yang telah terlatih. Pada penelitian ini digunakan arsitektur CNN seperti yang ditunjukkan pada gambar 2 .

Arsitektur ResNet-18 yang diimplementasikan pada penelitian ini merupakan salah satu varian *Residual Network* yang dirancang untuk mengatasi permasalahan *degradation* pada *deep layer*. Struktur *layer* diawali dengan lapisan konvolusi berukuran kernel 7×7 dengan 64 filter dan stride 2, dilanjutkan oleh *batch normalization* dan fungsi aktivasi ReLU, serta diikuti dengan operasi *max pooling* berukuran 3×3. Setelah melalui keseluruhan blok, representasi fitur dipadatkan menggunakan *average pooling*, kemudian diteruskan ke lapisan *fully connected* (FC) dengan 25 neuron output sesuai jumlah kelas yang diprediksi. Dengan rancangan ini, ResNet-18 mampu mengekstraksi representasi fitur secara

hierarkis dan efisien, serta mempertahankan akurasi meskipun kedalaman jaringan relatif besar, berkat kontribusi utama dari mekanisme *residual learning*.

Prediksi ini menghasilkan label baru yang kemudian dibandingkan dengan label aktual pada tahap evaluasi model.

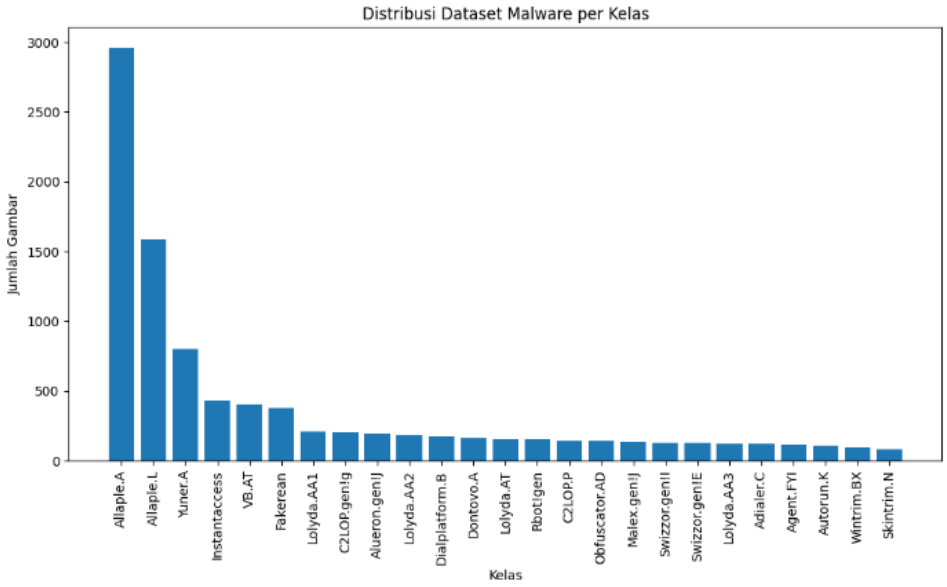


Gambar 2. Arsitektur ResNet-18

Evaluasi model dilakukan menggunakan metrik standar seperti akurasi, presisi, recall, dan F1-score [1], [6], guna memberikan gambaran yang komprehensif mengenai kemampuan model dalam mengidentifikasi malware. Dengan demikian, penelitian ini menghasilkan sistem identifikasi malware berbasis deep learning dengan penerapan model CNN yang teruji performanya, dan hasil evaluasi tersebut dapat dijadikan dasar untuk analisis maupun pengembangan lebih lanjut.

III. HASIL DAN PEMBAHASAN

Pada bagian ini disajikan hasil penelitian yang diperoleh dari penerapan metode CNN dengan arsitektur ResNet-18 terhadap dataset malware. Analisis difokuskan pada distribusi data per kelas, performa model dalam melakukan klasifikasi, serta faktor-faktor yang memengaruhi hasil pengujian. Dataset yang digunakan terdiri atas 25 kelas malware**,** dengan total data sebanyak 9.348 sampel yang dibagi menjadi train set (6.543 sampel)**,** validation set (1.402 sampel)**,** dan test set (1.403 sampel)**.** Daftar kelas meliputi malware: *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,* dan *Yuner.A*. Distribusi dataset malware dapat dilihat pada Gambar 3.



Gambar 3. Distribusi Dataset Malware

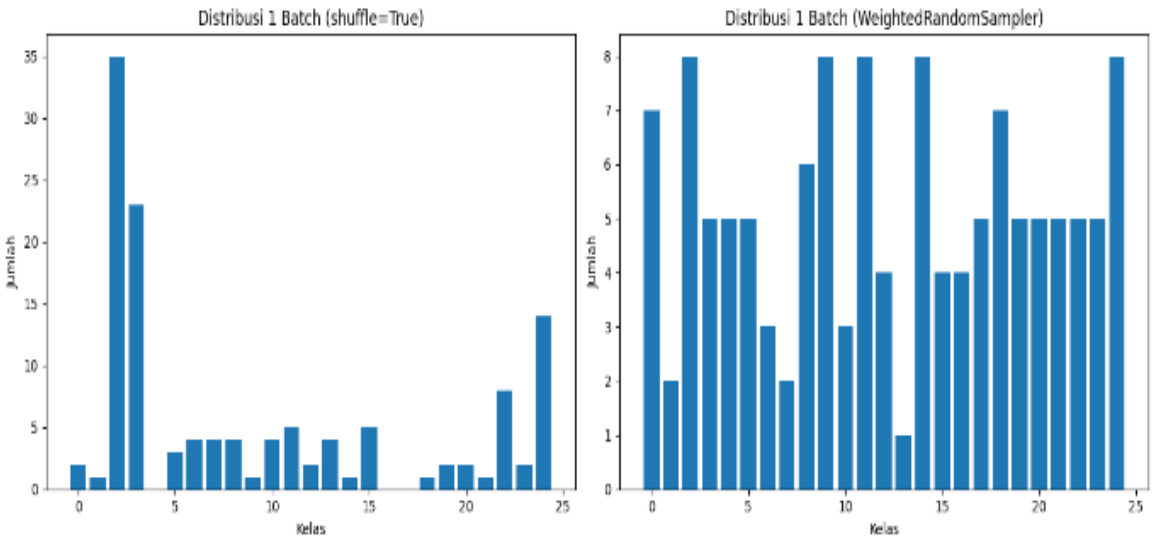
Gambar 3 tersebut menunjukkan bahwa jumlah sampel pada setiap kelas tidak merata atau mengalami ketidakseimbangan (*imbalance data*). Untuk mengatasi permasalahan klasifikasi, digunakan arsitektur ResNet-18. Mekanisme ini membantu model belajar pola visual yang lebih kompleks dari citra malware. Dengan demikian, untuk meningkatkan performa model secara keseluruhan, diperlukan strategi tambahan seperti data augmentation, oversampling, atau undersampling pada kelas minoritas. Selain itu, evaluasi tidak hanya dilakukan menggunakan akurasi global, tetapi juga metrik lain seperti precision, recall, dan F1-score per kelas, agar gambaran kinerja model lebih komprehensif.

```python
from torchvision import transforms

# Augmentasi untuk data training
train_transform = transforms.Compose([
    transforms.Resize((128, 128)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406],
                         [0.229, 0.224, 0.225])  # Normalisasi (ImageNet)
])

# Augmentasi untuk data validasi & test (tanpa augmentasi, hanya normalisasi)
val_test_transform = transforms.Compose([
    transforms.Resize((128, 128)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406],
                         [0.229, 0.224, 0.225])
])
```

Gambar 4. Proses Augmentasi Data

Pada gambar 4, proses augmentasi data dilakukan untuk meningkatkan variasi citra malware pada data training sehingga model CNN ResNet-18 dapat belajar lebih baik, tidak mudah mengalami overfitting, serta mampu melakukan generalisasi terhadap data baru. Augmentasi ini melibatkan beberapa transformasi, yaitu mengubah ukuran citra menjadi 128×128 piksel (Resize), melakukan pembalikan horizontal secara acak (RandomHorizontalFlip), serta rotasi acak hingga ±10 derajat (RandomRotation). Selanjutnya, citra diubah menjadi format tensor melalui ToTensor() dan dinormalisasi menggunakan parameter mean dan standar deviasi ImageNet melalui Normalize(). Dengan kombinasi transformasi tersebut, model memperoleh berbagai variasi representasi citra tanpa mengubah label kelas aslinya.

Gambar 5 menampilkan perbandingan distribusi sampel pada satu batch data ketika menggunakan dua metode pengambilan sampel yang berbeda. Pada grafik sebelah kiri ditunjukkan hasil distribusi dengan pengaturan shuffle=True, di mana data diacak secara acak tanpa memperhatikan jumlah sampel pada tiap kelas. Pola distribusi yang dihasilkan terlihat tidak seimbang, dengan beberapa kelas mendominasi jumlah sampel dalam batch sementara kelas lain hanya terwakili dalam jumlah yang sangat sedikit. Kondisi ini mencerminkan adanya masalah ketidakseimbangan kelas (class imbalance) yang berpotensi membuat model lebih bias terhadap kelas mayoritas.



Gambar 5. Hasil Pembobotan Kelas

Sebaliknya, grafik pada sebelah kanan memperlihatkan distribusi ketika menggunakan WeightedRandomSampler. Hasilnya, distribusi antar kelas dalam batch menjadi lebih merata, sehingga setiap kelas memiliki peluang yang sama untuk dilatih pada model. Dengan pendekatan ini, masalah ketidakseimbangan dapat diminimalkan, dan model memperoleh kesempatan untuk belajar representasi dari seluruh kelas secara merata.

```python
# Load ResNet18 pretrained
resnet18 = models.resnet18(weights=models.ResNet18_Weights.IMAGENET1K_V1)

# Freeze semua layer awal
for param in resnet18.parameters():
    param.requires_grad = False

# Ganti fully connected layer terakhir (sesuaikan dengan jumlah kelas dataset)
num_classes = len(train_dataset.classes)
resnet18.fc = nn.Linear(resnet18.fc.in_features, num_classes)

# Pindahkan model ke device
model = resnet18.to(device)
print(model)
```

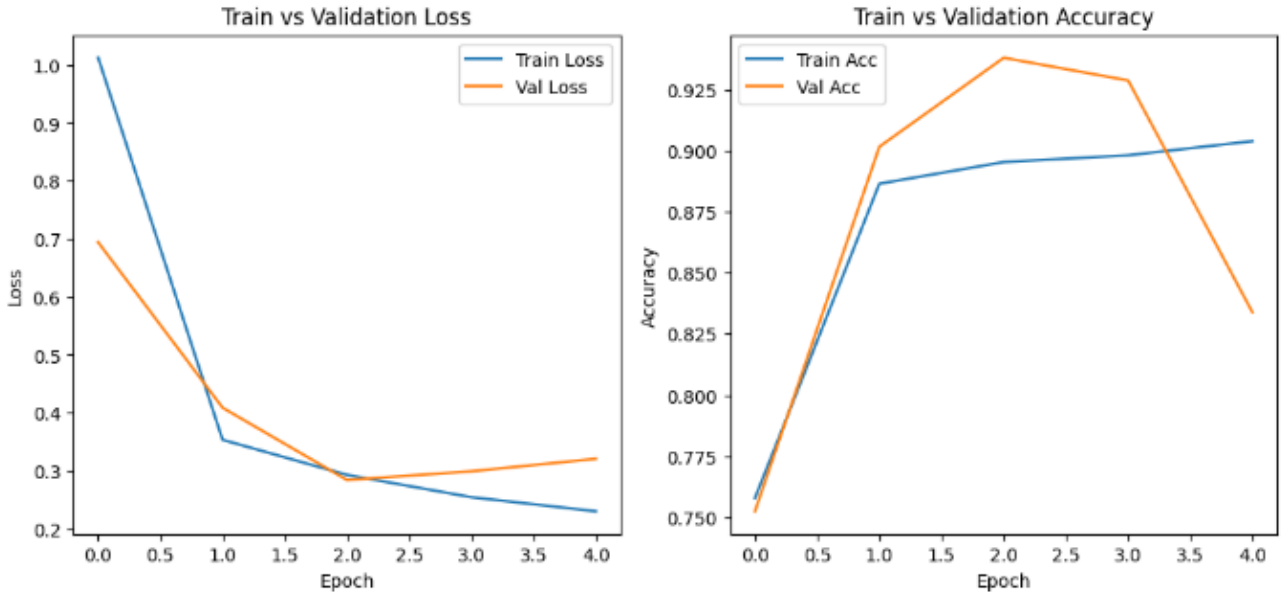
Gambar 6. Integrasi Arsitektur ResNet-18

Pada gambar 6 implementasi arsitektur ResNet-18 dengan bobot awal dari pretrained model ImageNet. Pemanfaatan pretrained model bertujuan untuk melakukan transfer learning, sehingga model dapat memanfaatkan pengetahuan fitur dasar yang sudah diperoleh dari dataset besar ImageNet.

Langkah pertama adalah memuat ResNet-18 dengan bobot pretrained. Selanjutnya, seluruh layer awal pada model dibekukan (freeze parameters) dengan mengatur requires_grad = False. Strategi ini dipilih agar parameter pada layer dasar yang sudah terlatih tidak diperbarui selama proses training, sehingga proses pelatihan menjadi lebih efisien dan terfokus pada penyesuaian layer akhir.

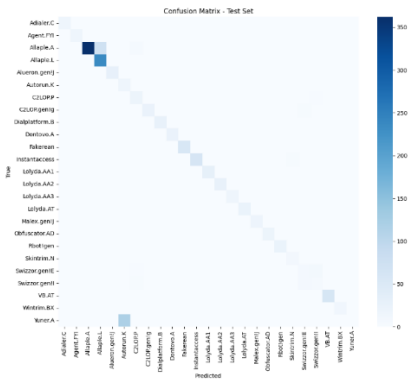
Tahap berikutnya adalah mengganti lapisan fully connected (FC) terakhir dengan layer baru yang menyesuaikan jumlah kelas pada dataset malware. Jumlah neuron pada layer akhir disesuaikan dengan jumlah kelas, yaitu 25 kelas. Dengan demikian, ResNet-18 dapat menghasilkan prediksi sesuai dengan kategori malware yang ada pada dataset.

Setelah modifikasi selesai, model kemudian dipindahkan ke perangkat komputasi (CPU atau GPU) untuk proses pelatihan lebih lanjut. Tahapan ini memastikan model siap digunakan pada tahap training, validasi, maupun testing.



Gambar 7. Hasil Pengujian (Loss dan Akurasi)

Hasil pelatihan model ResNet-18 ditunjukkan pada grafik Train vs Validation Loss dan Train vs Validation Accuracy pada gambar 7. Pada grafik loss, terlihat bahwa nilai train loss dan validation loss mengalami penurunan signifikan dari epoch pertama hingga epoch ketiga. Penurunan ini menunjukkan bahwa model mampu mempelajari pola pada data training dengan baik sekaligus mempertahankan performa pada data validasi. Setelah epoch ketiga, validation loss cenderung stabil dengan sedikit peningkatan, yang menandakan mulai terjadi kondisi overfitting ringan, di mana model semakin optimal pada data training namun kurang stabil pada data validasi. Sementara itu, grafik akurasi memperlihatkan peningkatan yang konsisten pada kedua metrik, baik train accuracy maupun validation accuracy, sejak epoch pertama hingga epoch ketiga. Akurasi validasi bahkan sempat mencapai titik tertinggi sekitar 95% pada epoch ketiga, sedikit lebih tinggi dibandingkan akurasi training.



Gambar 8. Confusion Matrix

Gambar 8 confusion matrix memperlihatkan hasil evaluasi performa model ResNet-18 terhadap 25 kelas malware pada data uji. Secara umum, distribusi prediksi menunjukkan bahwa sebagian besar sampel berhasil dipetakan secara benar ke kelas yang sesuai, yang ditunjukkan oleh dominasi nilai diagonal pada matriks. Hal ini mengindikasikan bahwa model mampu mengenali pola citra malware dengan cukup baik.

=== Classification Report (Test Set) ===

	precision	recall	f1-score	support
Adialer.C	1.00	1.00	1.00	18
Agent.FYI	1.00	1.00	1.00	17
Allaple.A	1.00	0.82	0.90	444
Allaple.L	0.76	0.99	0.86	239
Alueron.gen!J	1.00	1.00	1.00	30
Autorun.K	0.12	1.00	0.21	16
C2LOP.P	0.62	0.91	0.74	22
C2LOP.gen!g	0.93	0.83	0.88	30
Dialplatform.B	1.00	1.00	1.00	27
Dontovo.A	1.00	1.00	1.00	25
Fakerean	1.00	1.00	1.00	57
Instantaccess	0.98	0.95	0.97	65
Lolyda.AA1	1.00	1.00	1.00	32
Lolyda.AA2	0.97	1.00	0.98	28
Lolyda.AA3	1.00	1.00	1.00	18
Lolyda.AT	0.96	1.00	0.98	24
Malex.gen!J	0.95	0.95	0.95	20
Obfuscator.AD	0.95	1.00	0.98	21
Rbot!gen	0.92	1.00	0.96	24
Skintrim.N	0.79	0.92	0.85	12
Swizzor.gen!E	0.40	0.42	0.41	19
Swizzor.gen!I	0.35	0.30	0.32	20
VB.AT	1.00	0.98	0.99	61
Wintrim.BX	1.00	1.00	1.00	14
Yuner.A	0.00	0.00	0.00	120
accuracy			0.83	1403
macro avg	0.83	0.88	0.84	1403
weighted avg	0.83	0.83	0.82	1403

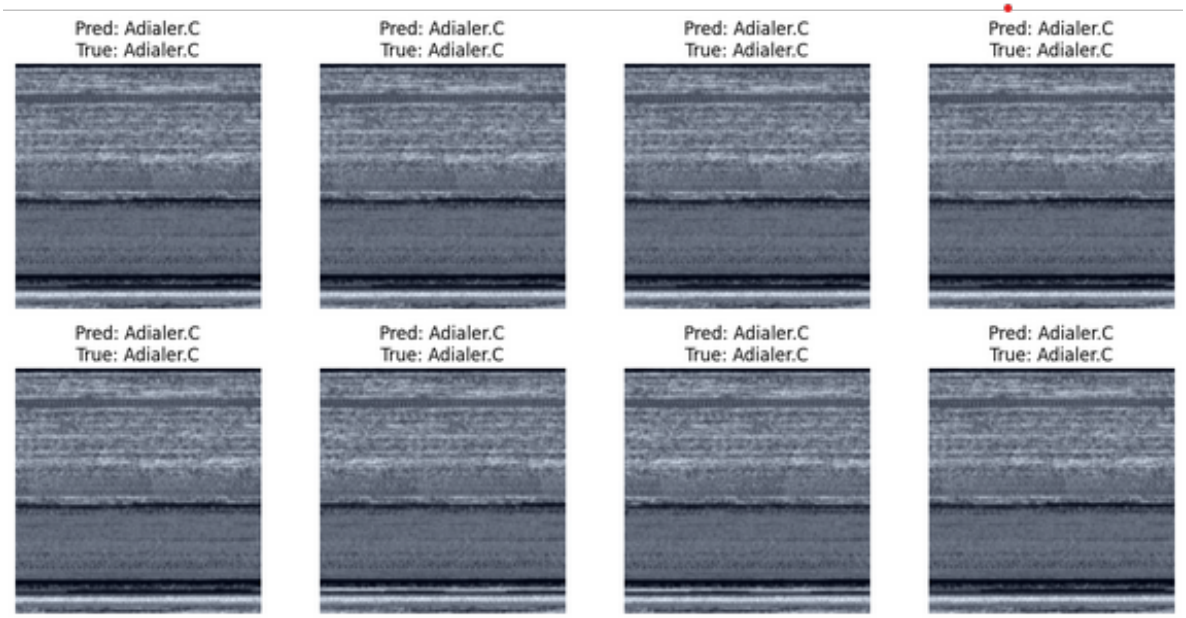
Gambar 9. Hasil Evaluasi Model

Gambar 9 menampilkan metrik evaluasi berupa precision, recall, dan F1-score untuk masing-masing dari 25 kelas malware. Secara keseluruhan, model mencapai akurasi sebesar 83% pada 1.403 sampel uji. Nilai macro average dan weighted average untuk precision, recall, serta F1-score juga berada di angka 0.83, menunjukkan konsistensi performa model yang digunakan.

Jika ditinjau per kelas, terlihat bahwa sebagian besar kelas memiliki performa sangat tinggi, dengan nilai precision, recall, dan F1-score mendekati atau sama dengan 1.00. Contoh kelas dengan hasil sempurna adalah Adialer.C, Alueron.gen!J, Dialplatform.B, Dontovo.A, Lolyda.AA3, Malex.gen!J, Obfuscator.AD, Rbot!gen, VB.AT, dan Wintrim.BX. Hal ini menunjukkan bahwa model mampu mengenali malware pada kelas-kelas tersebut secara konsisten tanpa kesalahan berarti.

Namun demikian, terdapat beberapa kelas dengan performa lebih rendah. Kelas C2LOP.P memiliki precision 0.62 dan F1-score 0.74, sementara kelas Swizzor.gen!E dan Swizzor.gen!I memperlihatkan hasil yang sangat rendah, masing-masing dengan F1-score 0.44 dan 0.36. Kelas Yuner.A tidak berhasil diprediksi sama sekali dengan nilai precision, recall, dan F1-score sebesar 0.00. Rendahnya performa pada kelas-kelas tersebut terutama dipengaruhi oleh jumlah sampel yang terbatas serta distribusi dataset yang tidak seimbang, sehingga model kesulitan belajar representasi yang memadai.

Performa tinggi pada kelas mayoritas seperti Allaple.A (F1-score 0.89) dan Allaple.L (F1-score 0.86) juga menunjukkan adanya korelasi positif antara ukuran dataset dengan hasil klasifikasi. Sebaliknya, performa rendah pada kelas minoritas memperlihatkan terdapat ketidakseimbangan kelas.



Gambar 10. Hasil Prediksi Malware

Gambar 10 merupakan hasil klasifikasi model ResNet-18 terhadap kelas sample (Adialer.C) pada dataset uji. Setiap citra yang ditampilkan memiliki label asli (True) sebagai Adialer.C, dan hasil prediksi model (Pred) juga sesuai, yaitu Adialer.C. Hal ini menunjukkan bahwa model mampu mengenali pola citra malware pada kelas ini secara konsisten dan akurat.

Konsistensi prediksi benar pada seluruh sampel yang ditampilkan mengindikasikan bahwa fitur visual khas dari malware Adialer.C berhasil dipelajari dengan baik oleh model. Dengan kata lain, representasi internal yang dibentuk ResNet-18 mampu membedakan kelas ini dari kelas malware lain, meskipun citra tampak memiliki tekstur yang serupa. Hasil penelitian ini menunjukkan bahwa penerapan metode ResNet-18 efektif untuk mendeteksi jenis malware, terutama kelas dengan pola citra yang relatif khas dan jumlah data yang memadai.

IV. KESIMPULAN

Hasil dari penelitian ini menunjukkan bahwa penerapan Convolutional Neural Network (CNN) dengan arsitektur ResNet-18 mampu memberikan hasil yang baik dalam tugas klasifikasi citra malware. Dengan memanfaatkan pendekatan transfer learning dari pretrained model ImageNet serta strategi augmentasi data pada tahap pelatihan, model berhasil mencapai akurasi keseluruhan sebesar 83% pada dataset uji. Hasil evaluasi memperlihatkan bahwa model mampu mempelajari pola visual secara efektif, ditunjukkan oleh nilai precision, recall, dan F1-score rata-rata yang konsisten pada tingkat 0.83. Meskipun demikian, performa model masih dipengaruhi oleh distribusi data yang tidak seimbang, di mana kelas dengan jumlah sampel terbatas cenderung memiliki tingkat prediksi yang lebih rendah dibandingkan kelas dengan jumlah sampel yang lebih besar. Temuan ini mengindikasikan bahwa ResNet-18 merupakan arsitektur yang cukup andal untuk klasifikasi malware berbasis citra, namun efektivitasnya sangat bergantung pada kualitas dan keseimbangan dataset yang digunakan. Dengan optimalisasi lebih lanjut terhadap distribusi data dan parameter pelatihan, model ini berpotensi memberikan hasil yang lebih baik dan dapat diimplementasikan dalam sistem deteksi malware yang lebih komprehensif.

Secara keseluruhan, evaluasi ini menunjukkan bahwa ResNet-18 mampu memberikan hasil klasifikasi yang sangat baik pada sebagian besar kelas malware, terutama kelas dengan jumlah data yang mencukupi. Akan tetapi, untuk kelas minoritas yang jarang muncul, performanya masih rendah. Oleh karena itu, diperlukan strategi tambahan seperti oversampling, data augmentation khusus kelas minoritas, atau cost-sensitive learning agar performa model dapat lebih merata di seluruh kelas.

UCAPAN TERIMA KASIH

Terima kasih kepada KEMENDIKTISAINTEK, Politeknik Negeri Sriwijaya dan P3M Polsri, untuk pendanaan dengan nomor kontrak PNBP: 07011/PL6.2.1/LT/2025, untuk Jurusan Teknik

Komputer Polsri, kepada mitra penelitian PT. Integra Telekom dan Kepolisian Daerah Provinsi Sumatera Selatan.

REFERENSI

[1] N. Afifah and D. Stiawan, "The Implementation of Deep Neural Networks Algorithm for Malware Classification," *Comput. Eng. Appl. J.*, vol. 8, no. 3, pp. 189–202, 2019.

[2] D. Stiawan, S. M. Daely, A. Heryanto, N. Afifah, M. Y. Idris, and R. Budiarto, "Ransomware detection based on opcode behaviour using k-nearest neighbours algorithm," *Inf. Technol. Control*, vol. 50, no. 3, pp. 495–506, 2021.

[3] R. B. Hadiprakoso, N. Qomariasih, and R. N. Yasa, "Identifikasi Malware Android Menggunakan Pendekatan Analisis Hibrid Dengan Deep Learning," *J. Teknol. Inf. Univ. Lambung Mangkurat*, vol. 6, no. 2, pp. 77–84, 2021.

[4] A. Brown, M. Gupta, and M. Abdelsalam, "Automated machine learning for deep learning based malware detection," *Comput. Secur.*, vol. 137, pp. 1–17, 2024.

[5] J. Yang, J. Tang, R. Yan, and T. Xiang, "Android Malware Detection Method Based on Permission Complement and API Calls," *Chinese J. ...*, 2022.

[6] R. Maulana, D. Stiawan, and R. Budiarto, "Detection of android malware with deep learning method using convolutional neural network model," *Comput. Sci. Inf. Technol.*, vol. 6, no. 1, pp. 68–79, 2025.

[7] D. Stiawan *et al.*, "An Improved LSTM-PCA Ensemble Classifier for SQL Injection and XSS Attack Detection," *Comput. Syst. Sci. Eng.*, vol. 46, no. 2, pp. 1759–1774, 2023.

[8] R. Yumlembam, B. Issac, S. M. Jacob, and ..., "Iot-based android malware detection using graph neural network with adversarial defense," *IEEE Internet Things ...*, 2022.

[9] S. Badhani and S. K. Muttoo, "GENDroid-a graph-based ensemble classifier for detecting Android malware," *Int. J. Inf. ...*, 2022.

[10] O. Aslan and A. A. Yilmaz, "A New Malware Classification Framework Based on Deep Learning Algorithms," *IEEE Access*, vol. 9, pp. 87936–87951, 2021.

[11] S. Hosseini, A. E. Nezhad, and H. Seilani, "Android malware classification using convolutional neural network and LSTM," *J. Comput. Virol. ...*, 2021.

[12] U. Narayanan, "A novel approach to big data analysis using deep belief network for the detection of android malware," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 16, no. 3, pp. 1447–1454, 2019.

[13] L. Hübschle-Schneider and P. Sanders, "Parallel Weighted Random Sampling," *ACM Trans. Math. Softw.*, vol. 48, no. 3, 2022.

[14] N. Xie, X. Di, X. Wang, and J. Zhao, "Andro_MD: Android Malware Detection based on Convolutional Neural Networks," *Int. J. Performability ...*, 2018.

[15] J. Kim, J. Y. Paik, and E. S. Cho, "Attention-Based Cross-Modal CNN Using Non-Disassembled Files for Malware Classification," *IEEE Access*, vol. 11, no. February, pp. 22889–22903, 2023.