BAB II

TINJAUAN PUSTAKA

2.1 Landasan Teori

2.1.1 Model

Model dalam konteks pembelajaran mesin dan *Artificial Intelligent* adalah representasi matematis tentang hubungan antara *input* dan *output* yang dipelajari dari data. Model ini dibuat dari waktu ke waktu, melalui pembelajaran dan pengujian untuk memastikan bahwa *input* dan *output* yang dihasilkannya akurat serta sesuai dengan data. Model dikatakan efektif adalah model yang mampu menangkap pola dari data pelatihan dan memberikan generalisasi pada data baru (Goodfellow dkk., 2016).

Terdapat empat langkah utama dalam pembuatan model, diantaranya pra-pemrosesan data, pelatihan model, evaluasi, dan optimasi. Pra-pemrosesan data memerlukan pembersihan, normalisasi, dan transformasi data. Pelatihan model juga memerlukan pemilihan arsitektur, inisialisasi parametrik, dan iterasi untuk membantu memperkecil nilai *loss*. Proporsi data pengujian digunakan untuk melakukan evaluasi masing-masing. Optimasi dimaksudkan untuk meningkatkan akurasi dan fleksibilitas model (Andrew, 2021). Adapun metrik evaluasi dalam perancangan model yang akan diimplementasikan pada penelitian ini mengacu pada penelitian yang dilakukan (Al-E'mari dkk., 2021), yaitu:

a. Akurasi (*Accuracy*)

Akurasi adalah metrik yang digunakan untuk mengukur seberapa banyak prediksi model yang benar dibandingkan dengan jumlah total prediksi yang dilakukan. Adapun perhitungan nilai akurasi direpresentasikan pada Persamaan (1).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FP + TN} \qquad (1)$$

dengan keterangan:

TP (*True Positive*) = Jumlah sampel positif yang diprediksi benar.

TN (*True Negative*) = Jumlah sampel negatif yang diprediksi benar.

FP (*False Positive*) = Jumlah sampel negatif yang diprediksi salah sebagai positif.

FN (*False Negative*) = Jumlah sampel positif yang diprediksi salah sebagai negatif.

Nilai akurasi tersebut memiliki kelebihan yaitu mudah dihitung dan dipahami serta cocok digunakan jika jumlah data di setiap kelas seimbang. Namun, metrik nilai akurasi ini juga memiliki beberapa kekurangan yaitu tidak cocok untuk dataset yang memiliki distribusi kelas tidak seimbang, karena model bisa saja memiliki akurasi tinggi hanya dengan memprediksi kelas mayoritas.

b. Nilai *Loss* (*Loss Function*)

Nilai *loss* mengukur seberapa jauh prediksi model dari nilai sebenarnya. Nilai ini digunakan dalam proses optimasi model untuk meminimalkan kesalahan selama pelatihan. Rumus *loss function* terbagi menjadi dua, yaitu untuk klasifikasi dengan menggunakan *Cross-Entropy Loss* pada Persamaan (2) untuk *Binary Cross-Entropy Loss* dan Persamaan (3) untuk *Categorical Cross-Entropy Loss*.

$$Loss = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \qquad (2)$$

$$Loss = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij}) \quad (3)$$

dengan keterangan:

y_i = Nilai aktual dari sampel ke-i.

$\hat{y}_i$ = Probabilitas prediksi oleh model untuk kelas positif.

C = Jumlah kelas.

N = Jumlah Sampel.

Metrik *loss function* memiliki kelebihan yaitu dengan adanya nilai *loss* yang rendah berarti model lebih baik dalam memprediksi data. Hal ini memungkinkan optimasi melalui *backpropagation*. Namun, memiliki kekurangan bahwa *loss function* tidak selalu memberikan interpretasi secara langsung terhadap kinerja model, sehingga diperlukan metrik lain seperti akurasi dan *F1-score* untuk melengkapinya.

c. F1-Score

F1-score merupakan metrik yang mengukur keseimbangan antara presisi (*precision*) dan *recall*. Metrik ini sangat penting ketika dataset memiliki distribusi kelas yang tidak seimbang. Adapun rumus perhitungan *F1-score* terdapat pada Persamaan (4) serta dilengkapi persamaan (5) dan (6).

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

$$Recall = \frac{TP}{TP + FN} \quad (6)$$

Indikator *F1-score* disebut tinggi jika mendekati angka 1 berarti model memiliki keseimbangan yang baik antara presisi dan *recall*. Sebaliknya, *F1-score*

rendah jika mendekati 0 berarti model tidak dapat menangkap semua data positif atau memberikan terlalu banyak prediksi positif yang salah. *F1-score* memberikan Gambaran seimbang dalam dataset dibandingkan dengan metrik akurasi serta dapat mengurangi dampak dari jumlah sampel yang tidak merata dalam antar kelasnya. Namun, *F1-score* memiliki kekurangan dalam memberikan informasi langsung mengenai jumlah total data yang diprediksi dengan benar.

d. Waktu Eksekusi Pelatihan

Waktu eksekusi pelatihan adalah metrik yang mengukur efisiensi model dalam hal waktu yang diperlukan untuk melakukan proses pelatihan. Adapun rumus perhitungan waktu eksekusi pelatihan terdapat pada Persamaan (7).

$$T_{train} = N_{epoch} \times \frac{N_{samples}}{Batch\_size} \times T_{iteration} \qquad (7)$$

dengan keterangan:

T_{train} = waktu total pelatihan.

N_{epoch} = jumlah epoch (jumlah iterasi pelatihan lengkap).

$N_{samples}$ = jumlah total sampel dalam dataset.

$Batch\_size$ = jumlah sampel yang diproses dalam satu iterasi.

$T_{iteration}$ = waktu yang dibutuhkan untuk setiap iterasi.

Semakin singkat waktu yang diperlukan, maka model yang dibuat efisien dalam penggunaan sumber daya komputasi.

e. Waktu Inferensi

Waktu inferensi merupakan durasi yang diperlukan oleh model pembelajaran mesin untuk memproses data masukan dan menghasilkan prediksi atau keluaran setelah proses pelatihan selesai. Dalam konteks klasifikasi lalu lintas jaringan untuk

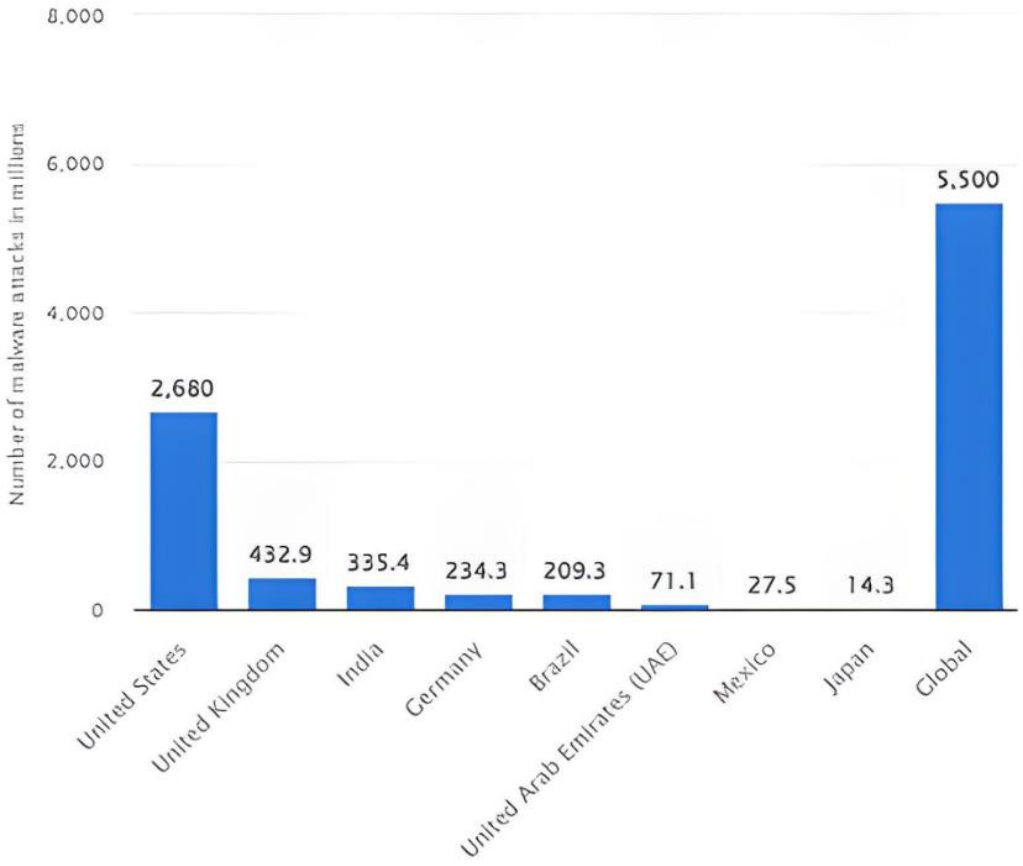
deteksi *malware*, waktu inferensi menjadi metrik krusial yang menentukan kemampuan sistem dalam memberikan respon terhadap ancaman keamanan siber. Waktu inferensi terdiri dari beberapa komponen utama yaitu waktu pemrosesan data masukan (*input processing time*), waktu komputasi *forward pass* melalui arsitektur model (*forward propagation time*), dan waktu pascapemrosesan untuk menghasilkan keluaran akhir (*output processing time*). Dalam arsitektur *hybrid* seperti *Trans Neural Network* yang menggabungkan *Transformer* dan *CNN*, waktu inferensi dipengaruhi oleh kompleksitas kedua komponen tersebut secara berurutan. Untuk arsitektur *Transformer*, waktu inferensi dipengaruhi secara signifikan oleh mekanisme *attention* terhadap panjang sekuens masukan. Sedangkan untuk *CNN*, waktu inferensi bergantung pada jumlah operasi konvolusi yang bersifat linear terhadap ukuran masukan (Zen dkk., 2024).

f. GFLOPs

FLOPs atau *Floating Point Operations* merupakan metrik yang digunakan untuk mengukur kompleksitas komputasi model, sedangkan *GFLOPS* merepresentasikan jumlah operasi tersebut dalam skala satu miliar per detik. Pada model *hybrid* seperti *Trans Neural Network*, metrik *GFLOPS* memungkinkan evaluasi efisiensi arsitektur gabungan *Transformer* dan *CNN* yang memiliki tingkat kompleksitas tinggi. Penggunaan *GFLOPS* membantu mengidentifikasi potensi *bottleneck* komputasi dan menilai kelayakan implementasi model dalam lingkungan dengan beban pemrosesan besar (Taufiqurrahman, 2025), (Li dkk., 2025).

2.1.2 *Malware*

Malware atau *malicious software* adalah perangkat lunak yang dirancang khusus untuk menyebabkan kerusakan, gangguan, atau akses yang tidak sah ke server, klien, jaringan, atau sistem komputer (SentinelOne, 2023). *Malware* memiliki karakteristik dan metode serangan yang unik dengan setiap hari rata-rata diciptakan 450.000 *malware* untuk menyerang bisnis, pemerintah, dan masyarakat umum (Onyegbula & Raza, 2024). Berdasarkan penelitian yang dilakukan oleh (Waili, 2023), *malware* memiliki pola serangan yang dapat diidentifikasi melalui pola lalu lintas, volume lalu lintas, dan distribusi lalu lintas jaringan.



Gambar 2. 1 Serangan *malware* Tahun 2022 (Wall, 2023)

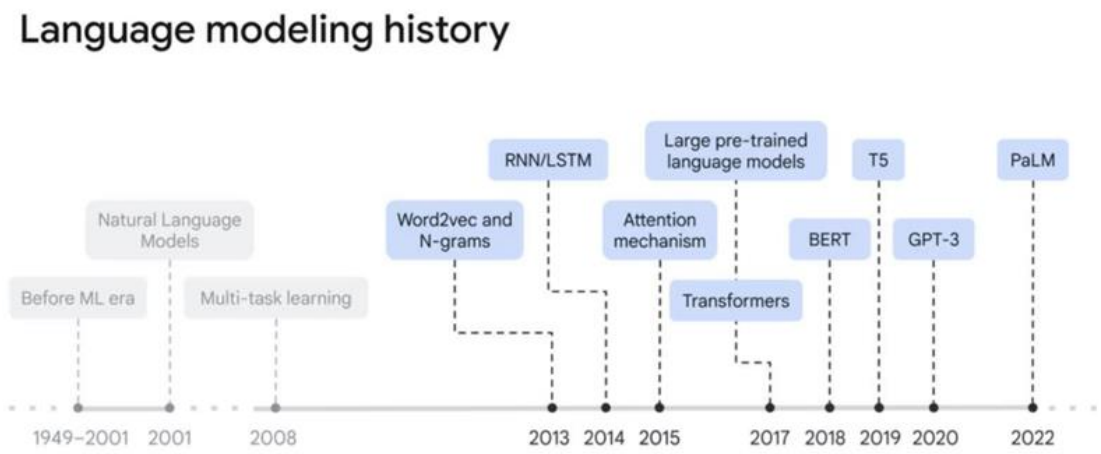
Gambar 2.1 merepresentasikan jumlah serangan *malware* dalam satuan jutaan di berbagai negara dan wilayah *sample* pada tahun 2022. Amerika Serikat merupakan negara dengan jumlah serangan *malware* tertinggi, mencapai 2.660 juta serangan dan diikuti pada posisi kedua oleh Inggris Raya dengan 432,9 juta serangan. Secara global, jumlah serangan *malware* mencapai 5.500 juta. Data ini menggarisbawahi tantangan keamanan siber dalam melindungi sistem komputer dan jaringan dari perangkat lunak berbahaya.

Menurut laporan Symantec, *malware* terus mengembangkan teknik khusus untuk langkah-langkah anti keamanan, seperti *polymorphic code* dan *obfuscation* untuk menghindari deteksi oleh sistem keamanan tradisional (Symantec, 2022). Cara kerja *malware* sering menggunakan teknik evasi untuk menghindari deteksi oleh sistem keamanan. Salah satu teknik yang umum adalah *polymorphism*, di mana *malware* mengubah kode atau tanda tangan (*signature*) setiap kali menyebar, sehingga sulit dideteksi oleh sistem. *Malware* juga menggunakan teknik *obfuscation* untuk menyembunyikan kode berbahaya, seperti enkripsi atau pengacakan *signature* (Symantec, 2022). Teknik penyerangan tersebut membuat analisis statis tradisional menjadi kurang efektif.

2.1.3 *Transformer*

Arsitektur *Transformer* merupakan salah satu inovasi paling signifikan dalam bidang pemrosesan bahasa alami atau *Natural Language Processing* dan telah mengubah cara melakukan pendekatan terhadap masalah dalam analisis teks. arsitektur *Transformer* dirancang untuk mengatasi keterbatasan model sebelumnya, seperti *Recurrent Neural Networks (RNN)*, dengan memanfaatkan mekanisme

perhatian (*attention mechanism*) yang memungkinkan pemrosesan data secara paralel (Devlin dkk., 2019). Sejarah bidang pemrosesan bahasa alami terus mengalami perkembangan dalam setiap masanya seperti pada Gambar 2.2.

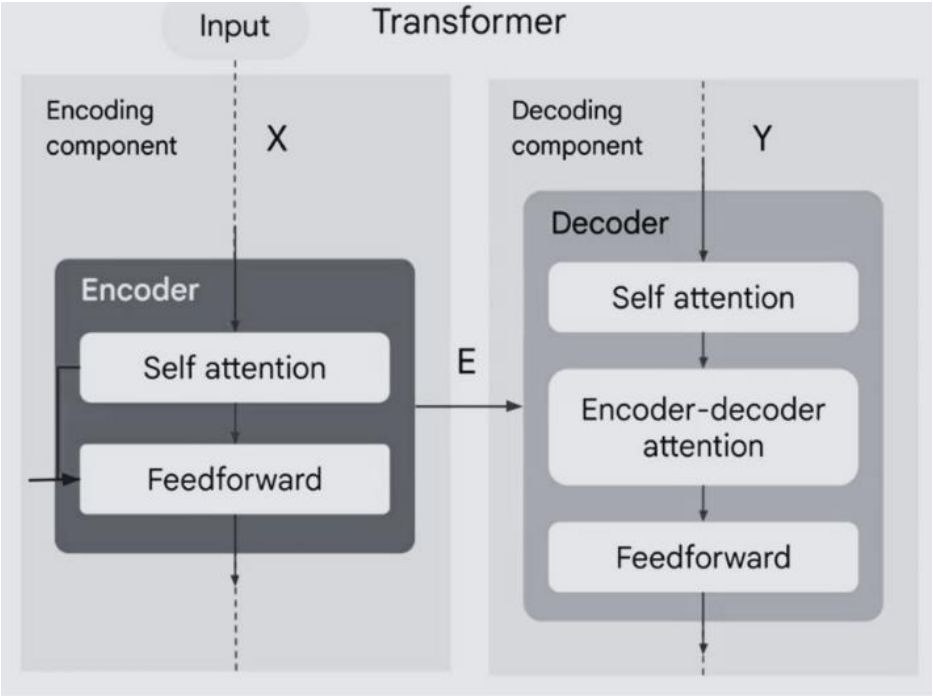


Gambar 2. 2 Sejarah Pemodelan Bahasa (Alfonso, 2023)

Gambar 2.2 merepresentasikan sejarah evolusi *language modeling* dari era sebelum *Machine Learning* hingga model bahasa canggih berbasis *Transformer*. Sebelum 2001, pemodelan bahasa masih berbasis aturan dan statistik sederhana, lalu berkembang dengan konsep multi-task learning pada 2008. Tahun 2013 menjadi tonggak penting dengan munculnya *Word2Vec* dan *N-grams*, yang memungkinkan representasi kata dalam bentuk vektor. Pada 2014, *RNN/LSTM* mulai digunakan untuk tugas *NLP*, tetapi memiliki keterbatasan dalam menangani konteks panjang, yang kemudian diatasi dengan mekanisme *Attention* pada 2015. Tahun 2017, arsitektur *Transformer* diperkenalkan dan menjadi dasar bagi model bahasa besar seperti *BERT* pada tahun 2018 yang memperkenalkan pendekatan bidirectional dalam memahami teks. Setelahnya, model berbasis *Transformer* semakin berkembang dengan hadirnya *T5* tahun 2019 untuk tugas *NLP* generatif,

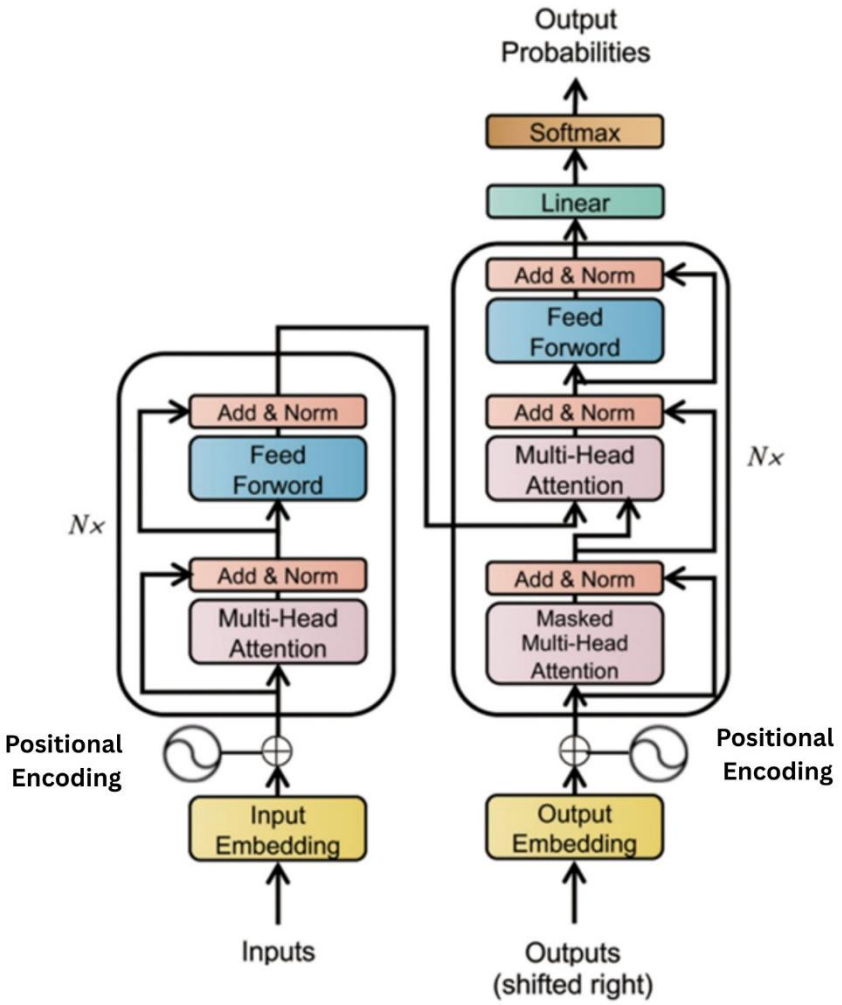
inovasi *GPT-3* tahun 2020 memiliki kemampuan menghasilkan teks dengan sangat alami. Kemudian pengembangan *PaLM* tahun 2022 mengunggulkan dalam memahami konteks kompleks. Gambar ini merepresentasikan kemajuan *NLP*dari pendekatan berbasis statistik hingga era *large pre-trained language models* yang mendominasi saat ini.

Arsitektur pada *Transformer* terdiri dari dua komponen utama yaitu *encoder* dan *decoder*. Komponen *encoder* bertanggung jawab untuk memproses *input* dan menghasilkan representasi yang lengkap, sementara *decoder* mengambil representasi tersebut untuk menghasilkan *output*. Adapun ilustrasi proses *encoder* dan *decoder* terdapat pada Gambar 2.3.



Gambar 2. 3 Proses *encoder* dan *decoder* (Alex, 2023)

Komponen *Transformer* terdiri dari beberapa lapisan yang menerapkan mekanisme *multi-head attention* dan lapisan *feed-forward* (Ganesan dkk., 2020). Adapun arsitektur secara utuh pada *Transformer* diilustrasikan oleh Gambar 2.4.



Gambar 2. 4 Arsitektur *Transformer* (Z. Zhang, 2023)

Ketika data dinputkan, bagian *decoder* menerapkan *masked multi-head attention* untuk mencegah melihat token berkelanjutan, kemudian melakukan *multi-head attention* pada *output encoder.* Lapisan *feed forward* dan normalisasi diterapkan, kemudian diikuti oleh *linear layer* dan *softmax* untuk menghasilkan probabilitas *output* akhir. Arsitektur ini mengulangi proses sebanyak N kali (Nx) untuk meningkatkan kemampuan ekstraksi fitur dan representasi kompleks, yang membuatnya sangat efektif dalam tugas-tugas pemrosesan bahasa dan prediksi urutan. Melalui adanya *attention mechanism,* memungkinkan model untuk

memberikan bobot yang berbeda pada bagian *input* yang berbeda sehingga dapat menangkap hubungan yang kompleks dalam data. Pada proses *attention mechanism*, membuka kemungkinkan bagi model untuk belajar dari berbagai sub-ruang representasi, sehingga meningkatkan kemampuan model dalam memahami konteks (B. Abushark dkk., 2022),. Dalam setiap lapisan, perhatian dihitung dengan membandingkan setiap elemen dalam urutan *input* dengan elemen lainnya, yang menghasilkan peta perhatian dengan menunjukkan seberapa besar *attention* pada setiap elemen (Bensaoud dkk., 2024).

Arsitektur Transformer memiliki keunggulan dalam fitur ekstraksi dengan bagian utama didalamnya yaitu *Pooled Output* dan *Sequence Features* dalam *Transformer Feature Extractor* sebagai dua komponen yang bekerja secara sinergis. *Pooled output* berfungsi sebagai hasil operasi *pooling* terhadap *output Transformer* secara keseluruhan yang menangkap fitur-fitur signifikan dan tersebar di berbagai posisi dalam sekuens, memberikan perspektif holistik melalui agregasi informasi spasial dan temporal yang dapat mengidentifikasi pola-pola penting tanpa terikat pada posisi spesifik dalam sekuens data, sehingga melengkapi representasi global *CLS token* dengan kemampuan menangkap variasi yang mungkin terlewatkan oleh *single token representation*. Di sisi lain, *sequence features* mempertahankan dimensi sekuensial dari data input dengan merepresentasikan fitur-fitur pada setiap posisi token secara eksplisit, menjaga information yang bersifat inheren atau terikat dalam data sekuensial untuk mengidentifikasi pola-pola lokal dan hubungan spasial yang mungkin tidak terdeteksi oleh representasi agregat, mempertahankan *temporal dependencies* dan *local patterns* yang krusial untuk

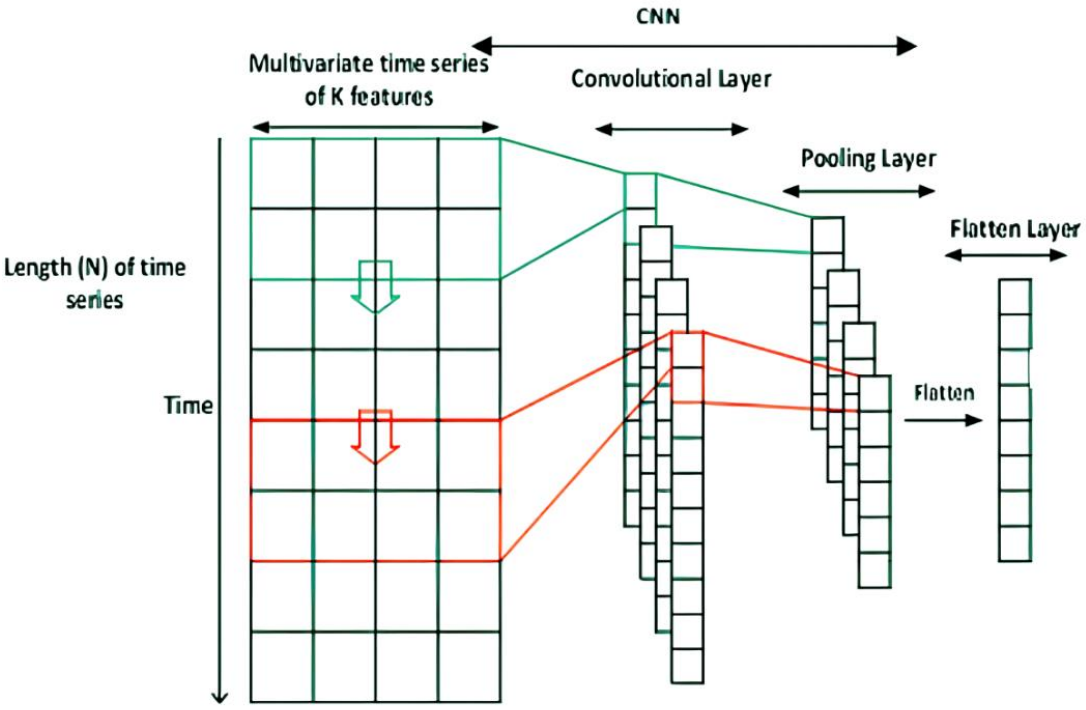
mendeteksi sub variasi, pola transisi, dan anomali dalam lalu lintas jaringan. Kombinasi kedua komponen ini menciptakan arsitektur yang dapat memahami *network traffic analysis* dari *multiple perspectives*, dimana *pooled output* menangkap *global context* dan *distributed features* sementara *sequence features* mempertahankan *local details* dan *sequential structure*, menghasilkan sinergi yang memungkinkan model untuk melakukan klasifikasi dan deteksi yang akurat dengan pemahaman yang mendalam pada *global patterns* maupun *local interactions* (Wu dkk., 2023).

Berdasarkan penjelasan mengenai *Transformer*, pada penelitian ini Arsitektur *Transformer* berperan dalam menangani kompleksitas pola lalu lintas jaringan dan meningkatkan akurasi deteksi *malware* melalui mekanisme *self-attention*, yang memungkinkan model memahami hubungan jangka panjang antar fitur tanpa kehilangan konteks global. *Transformer* akan memproses data secara paralel dan mempercepat pelatihan. Implementasinya dimulai dengan mengonversi lalu lintas jaringan ke dalam representasi numerik, kemudian *self-attention* digunakan untuk mengidentifikasi pola dan hubungan antar fitur yang mencerminkan aktivitas *malware*. Hasilnya diteruskan ke lapisan klasifikasi berbasis *Feedforward Neural Network (FNN)* untuk menghasilkan keputusan akhir. Dengan pendekatan ini, *Transformer* tidak hanya meningkatkan akurasi deteksi, tetapi juga memastikan model lebih efisien dalam menangani lalu lintas jaringan berskala besar tanpa kehilangan informasi penting.

2.1.4 *Convolutional Neural Network* (*CNN*)

Convolutional Neural Network atau *CNN* adalah arsitektur *deep learning* yang paling sering digunakan dalam tugas yang melibatkan data spasial, seperti pemrosesan citra dan pengenalan pola. *CNN* sebagian besar berdasarkan teorema yang memfokuskan pada kekuatan ekstraksi fitur hierarkis dari data yang rumit. *CNN* juga identik dengan fondasi revolusi *Artificial Intellgient (AI)* yang mengekstraksi fitur-fitur hierarki dari data kompleks (LeCun, 2020).

Arsitektur *CNN* dirancang untuk mengekstraksi fitur dari data gambar dengan cara yang efisien, berkat struktur berlapis yang terdiri dari beberapa jenis lapisan, termasuk lapisan *convolutional*, lapisan *pooling*, dan lapisan *flatten* (Khan dkk., 2020; Saleem dkk., 2022). Adapun ilustrasi dari arsitektur *CNN* terdapat pada Gambar 2.5.

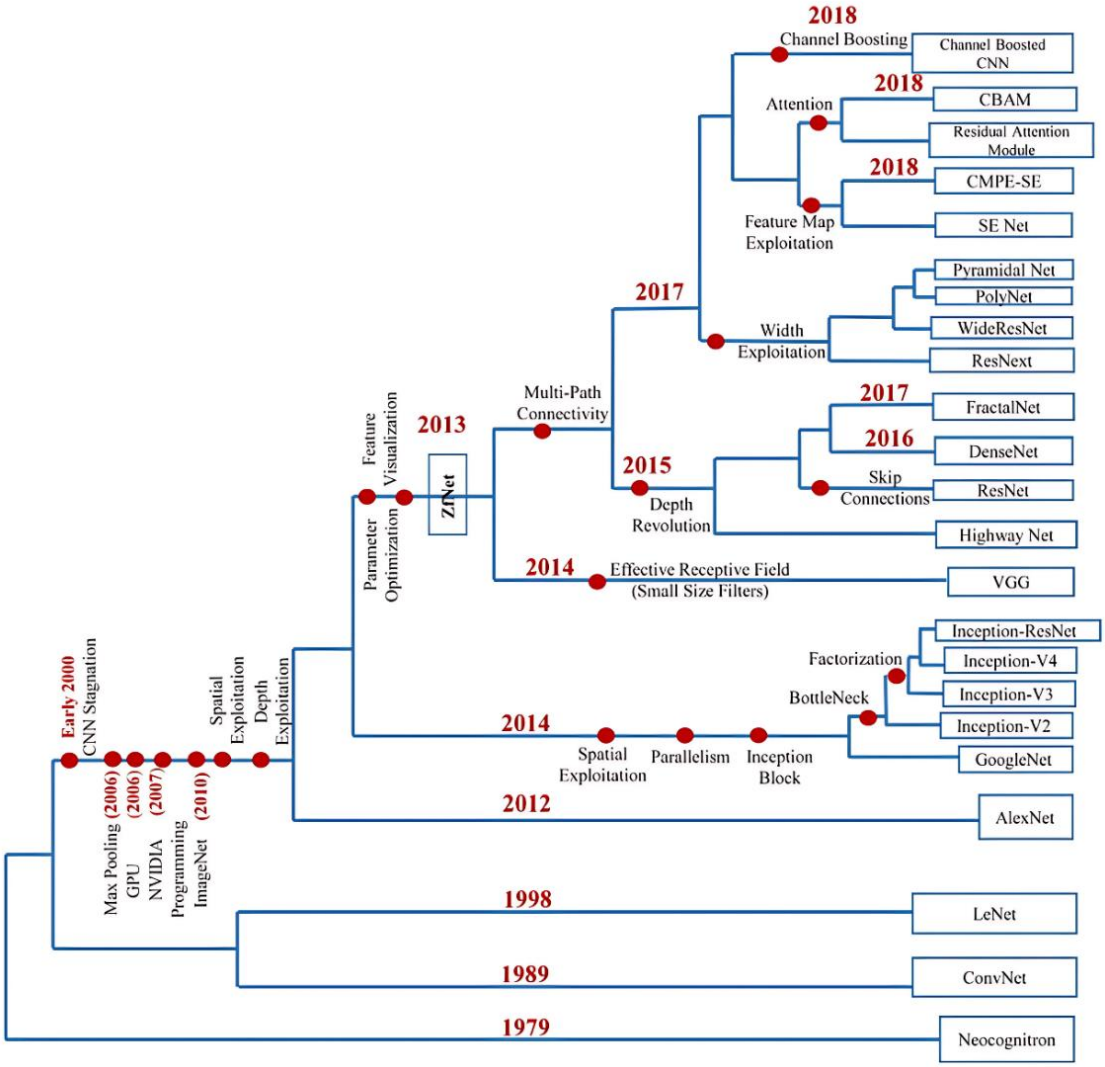


Gambar 2. 5 Arsitektur *CNN* (Lawal dkk., 2021)

Arsitektur *CNN* memungkinkan untuk belajar representasi yang kompleks dari data *input* yang sangat penting dalam berbagai aplikasi seperti klasifikasi gambar, deteksi objek, dan segmentasi (Thu dkk., 2023). Sinyal *input CNN* dapat berupa *time series multivariat* atau *univariat*. Adapun lebar *time series* bergantung pada jumlah fitur K dan panjang N dari series tersebut. Pada bagian *convolutional filters* memiliki lebar yang sama dengan lebar time series, tetapi panjangnya dapat bervariasi. Selanjutnya, *filter* dirancang untuk bergerak dalam satu arah sambil melakukan operasi konvolusi dari titik awal *time series* hingga titik akhirnya. Lapisan *convolutional* menghasilkan vektor *time series* yang telah difilter, dengan jumlah yang bergantung pada jumlah *convolution kernels*. Lapisan ini juga berfungsi menangkap fitur dari time series awal. Tahap berikutnya melibatkan *pooling* setiap vektor pada *time series* dari lapisan *convolutional* membentuk vektor-vektor baru. Lapisan yang bertanggung jawab atas proses ini disebut *pooling layer*. Vektor dari *pooling layer* kemudian diteruskan ke *flatten layer* atau *fully connected layer*. Hal ini menghasilkan *output* dari *flatten layer* yang diteruskan ke jaringan saraf LSTM (Lawal dkk., 2021).

Gambar 2.6 merepresentasikan alur revolusi dalam inovasi perkembangan arsitektur *CNN*. Hal ini menunjukkan bahwa *CNN* terus mengalami berbagai penyempurnaan dan modifikasi untuk meningkatkan efektivitasnya dalam berbagai tugas pemrosesan data. Inovasi tersebut mencakup pengembangan arsitektur yang lebih kompleks, optimalisasi parameter, serta integrasi dengan teknik-teknik baru untuk meningkatkan kemampuan ekstraksi fitur dan generalisasi model. Seiring dengan perkembangan teknologi dan meningkatnya kebutuhan akan model yang

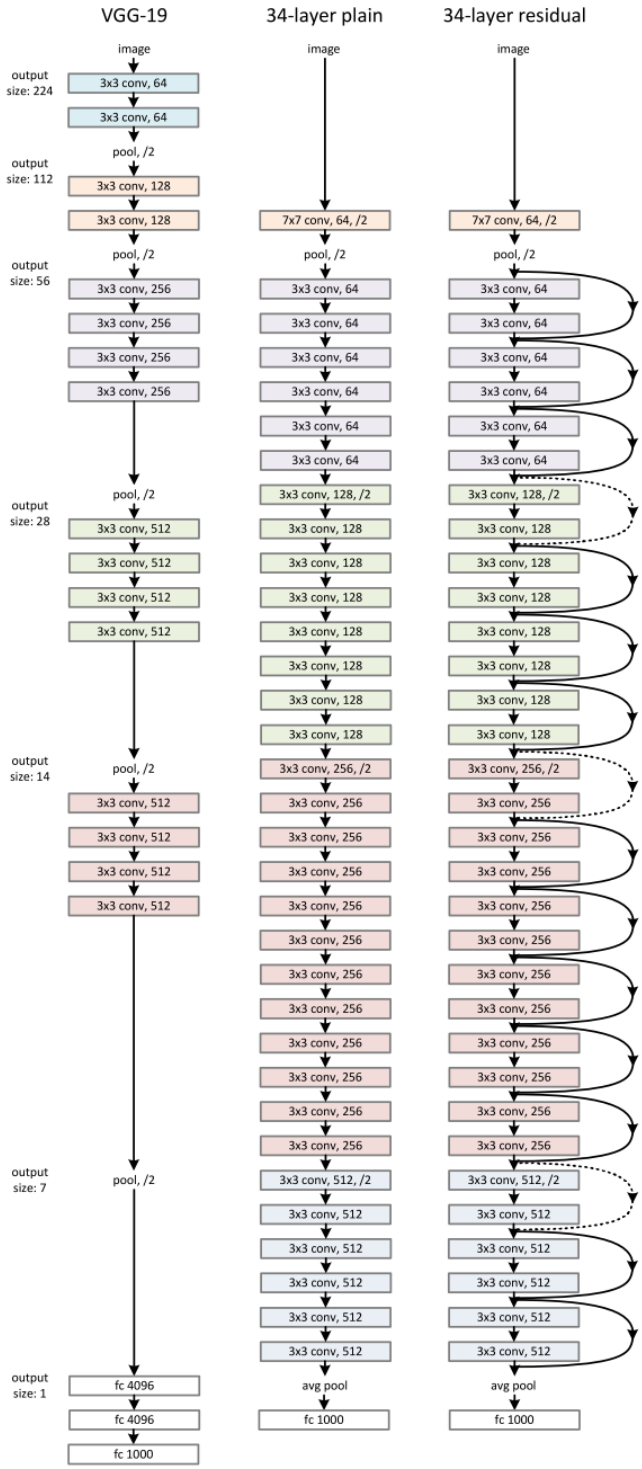
lebih efisien, inovasi dalam *CNN* terus berlanjut, baik dalam aspek peningkatan akurasi, efisiensi komputasi, maupun kemampuan adaptasi terhadap berbagai jenis data. Oleh karena itu, perkembangan *CNN* tidak hanya berfokus pada peningkatan kinerja, tetapi juga pada pengurangan kompleksitas komputasi dan peningkatan interpretabilitas model dalam berbagai domain aplikasi.



Gambar 2. 6 Revolusi perkembangan *CNN* (Khan et al., 2020)

Pada tahun 2015 sampai 2016 dikembangkan arsitektur *CNN* lanjutan berupa *ResNet*. *ResNet* atau *Residual Network* merupakan salah satu arsitektur

Convolutional Neural Network (CNN) yang dikembangkan melalui karya berjudul *Deep Residual Learning for Image Recognition* dengan arsitektur pada Gambar 2.4.



Gambar 2. 7 Arsitektur *ResNet* (He dkk., 2015)

Arsitektur *ResNet* dirancang untuk mengatasi permasalahan degradasi akurasi yang sering terjadi pada jaringan neural yang sangat dalam. Tidak seperti *CNN* konvensional, *ResNet* memperkenalkan konsep residual learning dengan menggunakan *shortcut connections*, yaitu jalur identitas yang memungkinkan informasi dari suatu lapisan langsung diteruskan ke lapisan yang lebih dalam tanpa melewati operasi non-linear. Konsep ini memungkinkan pelatihan jaringan dengan ratusan bahkan ribuan lapisan tanpa mengalami penurunan performa, menjadikan *ResNet* sebagai arsitektur yang sangat stabil dan efisien untuk klasifikasi citra skala besar (He dkk., 2015).

Arsitektur *ResNet* memiliki struktur berlapis dengan blok-blok residual yang memungkinkan informasi dari lapisan sebelumnya melompati beberapa lapisan dan langsung ditambahkan ke *output* lapisan lebih dalam yang ditunjukkan dengan garis lengkung putus-putus. Setiap blok residual terdiri dari beberapa lapisan konvolusi 3x3 dengan berbagai jumlah *filter* seperti 64, 128, 256, dan 512, kemudian diikuti dengan operasi *pooling* untuk mengurangi dimensi spasial. Lapisan *skip connection* ini memungkinkan gradien mengalir lebih mudah selama proses *backpropagation*, sehingga memungkinkan pelatihan jaringan yang sangat dalam hingga ratusan lapisan tanpa mengalami degradasi performa, yang pada akhirnya menghasilkan akurasi yang lebih tinggi dibandingkan arsitektur konvensional yang tidak memiliki mekanisme residual ini (He dkk., 2015).

Penelitian oleh (S. Machmeier dkk., 2023) menunjukkan bahwa *ResNet-50* berhasil mencapai akurasi hingga 99,48% dengan *F1-score* sebesar 98,88% dalam klasifikasi lalu lintas jaringan antara *benign* dan *malicious* setelah dilakukan

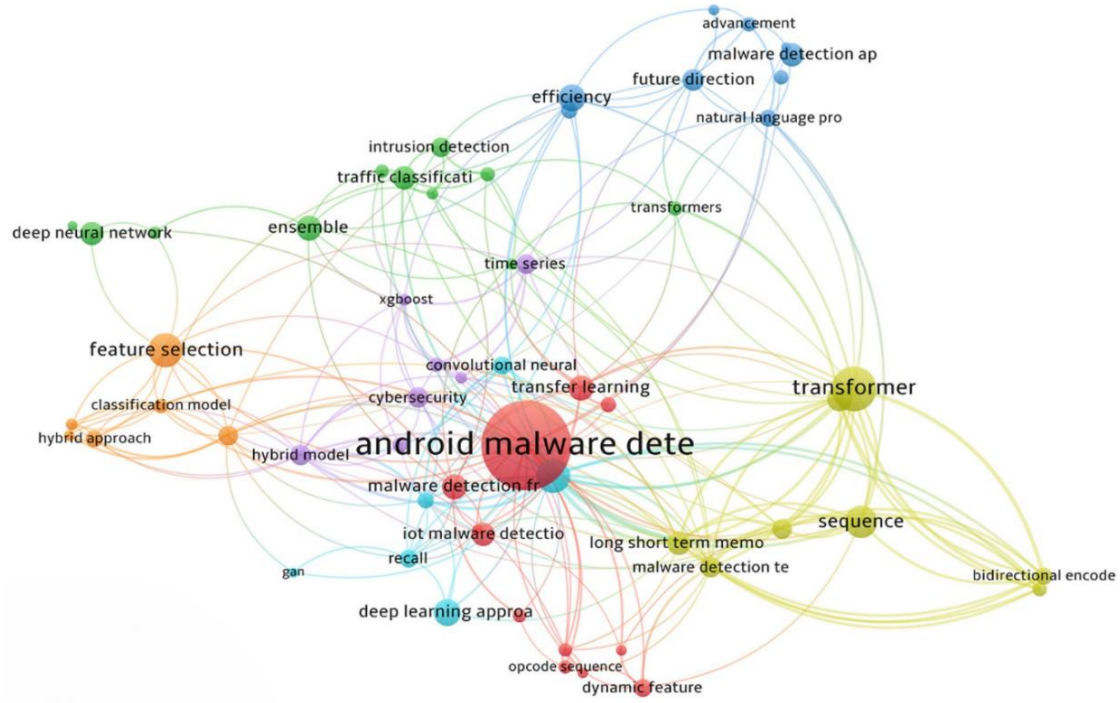
representasi citra terhadap aliran data. Temuan serupa juga dilakukan oleh (A. Mahato dkk., 2024), di mana *ResNet* menunjukkan performa terbaik dibandingkan arsitektur *CNN* lain seperti *AlexNet*, *VGG-16*, dan *LeNet* dengan akurasi mencapai 99,78% pada klasifikasi malware berbasis citra. Kelebihan lain dari *ResNet* adalah kemampuannya untuk digabungkan secara efektif dalam pendekatan *transfer learning*, seperti ditunjukkan pada klasifikasi *multi-familia malware* dengan akurasi hingga 99,97% setelah melakukan *fine-tuning* terhadap model *ResNet*-50 yang telah dilatih sebelumnya (El-Shafai dkk., 2021). Dengan mempertimbangkan keandalan, akurasi tinggi, serta efisiensi arsitektur dalam mendeteksi pola kompleks, *ResNet* menjadi pilihan yang tepat untuk digunakan dalam sistem deteksi malware berbasis lalu lintas jaringan.

Dalam penelitian ini, setelah arsitektur *Transformer* diimplementasikan, *Convolutional Neural Network (CNN)* berupa *ResNet* diterapkan untuk melakukan ekstraksi fitur spasial dari representasi yang telah diproses tersebut. Arsitektur *CNN* menggunakan lapisan konvolusi untuk menangkap pola lokal dan hierarkis dalam fitur yang telah diekstraksi *Transformer*, seperti anomali dalam distribusi fitur, korelasi spasial antar dimensi fitur, atau pola struktural yang mencurigakan, sehingga memungkinkan model memahami karakteristik *malware* secara lebih komprehensif. Lapisan *pooling* dalam *CNN* kemudian diterapkan untuk mengurangi dimensi data dan meningkatkan efisiensi komputasi tanpa kehilangan informasi penting, sehingga kombinasi *Transformer* dan *CNN* dalam arsitektur *Trans Neural Network (TNN)* memungkinkan model mendeteksi *malware* dengan

akurasi dan efisiensi yang optimal melalui analisis sekuensial dan spasial yang terintegrasi.

2.2 Penelitian Terkait

Penelitian pada lalu lintas jaringan untuk deteksi aktivitas *malware* telah mengalami perkembangan signifikan, terutama dengan diterapkannya berbagai metode berbasis *machine learning* dan *deep learning*. Penelitian terkait berdasarkan data artikel ilmiah terindeks scopus dengan kata kunci pencarian berupa "*malware detection*", "*malware analysis*", "*deep learning for malware analysis*", "*optimization malware detection*", dan "*performance enhancing for malware detection*" selama lima tahun terakhir yaitu tahun 2020 sampai 2025 menghasilkan 640 artikel ilmiah utama dengan hasil visualisasi peta penelitian pada Gambar 2.8.



Gambar 2. 8 Analisis tren penelitian *malware*

Teknik *deep learning* telah menjadi pendekatan revolusioner dalam analisis *malware* dengan kemampuannya untuk melakukan ekstraksi fitur otomatis dan mendeteksi pola yang kompleks. Analisis bibliometrik dari literatur terpilih menunjukkan bahwa deteksi *malware android* telah menjadi fokus dominan dalam penelitian keamanan siber. Tren ini diikuti oleh penelitian tentang deteksi *malware* secara umum dan deteksi *malware IoT*, yang menunjukkan pertumbuhan signifikan. Dari perspektif metodologis, pendekatan berbasis *Transformer* telah muncul sebagai teknik yang banyak diadopsi, diikuti oleh Convolutional Neural Networks (CNN), *transfer learning*, *Long-Short Term Memory* (LSTM), *ensembel learning*, GAN, serta diikuti metode lainnya. Penerapan arsitektur Transformer menunjukkan keunggulan dalam menganalisis pola berurutan dalam data *malware*, sementara CNN tetap menjadi pilihan yang lebih disukai untuk analisis berbasis fitur spasial (Rahman, dkk., 2023).

Dalam konteks penelitian ini, penggabungan antara *CNN* dan *Transformer* dilakukan dengan tujuan tidak hanya sebagai tren, tetapi juga sebagai salah satu eksperimen dalam optimalisasi performa klasifikasi tetapi juga memahami dependensi jangka panjang pada data *sequence* sambil mempertahankan mekanisme *self-attention* pada *Transformer* serta peningkatan kemampuan ekstraksi fitur spasial yang dilakukan oleh *CNN*. Untuk memahami kontribusi dan perbandingan antara penelitian-penelitian yang terdahulu, pada Tabel 2.1 berikut mencakup hasil rangkuman dari penelitian-penelitian yang relevan mengenai perbandingan deteksi *malware* melalui teknik *deep learning*.

Tabel 2. 1 Penelitian Terkait

No.	Penulis	Arsitektur	Akurasi	Fokus Pembahasan
1	(Liu et al., 2024)	Kolaborasi *BERT* dan *ConvLSTM-AM*	98,81%	Representasi polisemi yang tidak presisi dan kurangnya representasi semantik kontekstual menyebabkan kegagalan dalam mengenali beberapa jenis *malware*. Maka dicoba Klasifikasi dengan dengan analisis semantik BERT-*base Transformer* dan ConvLSTM-AM.
2	(Xu, 2024)	*Transformer*	96,0%	Mengembangkan metode pembelajaran fitur *spatio-temporal* berbasis *STF-Transformer*. Mempelajari karakteristik temporal dan spasial dari lalu lintas serangan melalui *encoder* dan *decoder* yang lebih baik. Kinerja metode STF-*Transformer* diukur pada dataset CICIDS2017 dan NSL-KDD.
3	(Dulal, 2024)	*CNN*-ARFO	99,20%	Mengembangkan metode deteksi *malware* inovatif bernama *Convolutional Neural Network-Based Adaptive Red Fox Optimization* (*CNN*-ARFO) yang menggunakan pendekatan komprehensif melalui tiga fase yaitu pra-pemrosesan dengan normalisasi minmax, ekstraksi fitur berbasis *permission*, *API call*, dan *file* statis, serta deteksi menggunakan arsitektur *CNN* dan *Red Fox Optimization.*
4	(Ferrag, 2024)	BERT	96,0%	Merancang *SecurityBERT*, arsitektur baru yang memanfaatkan model *BERT* untuk mendeteksi ancaman siber. Digunakan teknik pengkodean privasi yaitu *Privacy-Preserving Fixed-Length Encoding* (PPFLE), yang dikombinasikan dengan *Byte-level Byte-Pair Encoder* (BBPE) dan *Tokenizer* untuk merepresentasikan data lalu lintas jaringan secara terstruktur.

No.	Penulis	Arsitektur	Akurasi	Fokus Pembahasan
5	(Rahman, Ahmed, Khan, Mahin, Kibria, Karim, dkk., 2023)	*CNN* vs *Transformer*	*CNN Proposed* 99,4% *Transformer* CCT 97,43%	Klasifikasi *malware* menggunakan teknik *deep learning* berbasis Gambar. Membandingkan kinerja *Convolutional Neural Networks* dan varian *Transformer*. Menggunakan 2,1 juta parameter dan mengeksplorasi arsitektur mana yang lebih efektif untuk tugas klasifikasi *malware* ini.
6	(Z. Zhang, 2023)	CBAM dan TCN	93,63%	Menerapkan *Convolutional Block Attention Module* (CBAM) *Spatio-Temporal Convolution Network-Transformer* (TCN) untuk prediksi *time-series*. Dirancang untuk mengekstraksi fitur *spasio-temporal* lalu lintas jaringan. Eksperimen dilakukan pada dataset lalu lintas jaringan nyata di kota Milan.
7	(Almeida, 2023)	BERT	95,0%	Mengkaji dinamika evolusi *ransomware* yang beralih dari serangan berbasis enkripsi konvensional ke strategi eksfiltrasi data. Memanfaatkan model BERT, menganalisis pola lalu lintas jaringan untuk mendeteksi aktivitas *ransomware*.
8	(Yu dkk., 2023)	MLST-FENet	98,5%	Pada data lalu lintas jaringan diubah menjadi data citra dengan fitur tekstur, lalu dikembangkan model klasifikasi multitugas bernama *Multilevel Spatiotemporal Feature Fusion Enhanced Network Traffic Classification Model* (MLST-FENet) untuk mendeteksi lalu lintas berbahaya dan terenkripsi. Model ini bekerja secara *end-to-end* dengan mempelajari hubungan nonlinier antara *input* dan *output*.
9	(Tian, 2023)	R-*CNN* dan *Transformer*	96,0%	Penelitian ini menerapkan arsitektur *Faster R-CNN* berbasis *Transformer* yang disempurnakan untuk deteksi rambu lalu lintas. Menggabungkan peta fitur *multi-level* melalui modul *fusi cascade* untuk menghasilkan peta fitur gabungan, sehingga

No.	Penulis	Arsitektur	Akurasi	Fokus Pembahasan
				meningkatkan kemampuan ekstraksi fitur. Berdasarkan eksperimen pada dataset TT100K, arsitektur ini terbukti meningkatkan *Mean Average Precision* (MAP) dan kecepatan deteksi, membuktikan efektivitas dan kesesuaiannya untuk diterapkan dalam *real time*.
10	(Nalinipriya dkk., 2022)	DRNN	92,0%	Merancang optimasi *Water Moth Flame* (WMFO) dan jaringan saraf tiruan dalam (*Deep* RNN) untuk menentukan *Ransomware*. Pelatihan DRNN dilakukan dengan WMFO dan dikembangkan dengan menggabungkan *Moth Flame optimization* (MFO) dan *Water wave optimization* (WWO). Selain itu, fitur ditambah dengan *opcode* dan dengan menemukan *term frequency-inverse document frequency* (TF-IDF) di antara fitur-fitur individual serta mengimplementasikan *Probabilistic Principal Component Analysis* (PPCA) yang diadaptasi dengan DRNN untuk menghasilkan bobot optimal.
11	(Yang dkk., 2022)	CGAN dan *Transformer*	95,0%	Menerapkan metode *Conditional Generative Adversarial Network* (CGAN) dan *Transformer*. CGAN digunakan untuk mengatasi ketidakseimbangan sampel dalam data serta meningkatkan akurasi deteksi sampel minoritas, sedangkan *Transformer* dimanfaatkan untuk kemampuannya dalam mengekstraksi fitur jarak jauh secara efektif. Metode ini diuji menggunakan dataset UNSW-NB15 dan KDDCup99.
12	(Seyyar dkk., 2022)	BERT	99,98%	Penelitian ini menggabungkan teknik *NLP*, model BERT, dan teknik *deep learning* untuk meningkatkan deteksi ancaman pada lalu lintas web. Setiap

No.	Penulis	Arsitektur	Akurasi	Fokus Pembahasan
				permintaan HTTP dianggap sebagai kata, kemduian model hibrida yang menggabungkan *BERT* untuk vektorisasi kata serta *Multi-Layer Perceptron* (MLP) untuk klasifikasi. Studi ini merupakan yang pertama kali mengkombinasikan *BERT* dan MLP untuk deteksi serangan web.
13	(Rahali & Akhloufi , 2021)	*BERT* dan *Transformer*	98,0%	Menerapkan MalBERT, sebuah model berbasis arsitektur *Transformer* dan *BERT* yang melakukan analisis statis pada *source code* aplikasi Android menggunakan fitur pra-proses untuk mengkarakterisasi dan mengklasifikasikan *malware* ke dalam kategori yang representatif.
14	(Ranade, Piplai, Joshi, dkk., 2021)	BERT	94,0%	Membuat korpus keamanan siber dari data *Cyber Threat Intelligence* (CTI) tidak terstruktur dan semi-terstruktur untuk menyempurnakan model dasar *BERT* dengan *Masked Language Modeling* (MLM) untuk mengenali entitas keamanan siber yang terspesialisasi.
15	(Ranade, Piplai, Mittal, dkk., 2021)	*Tranformer*	93,0%	Menggunakan model *Transformer*, seperti GPT-2 yang telah *fine-tuned*, otomatis menghasilkan teks CTI palsu yang tampak meyakinkan dan berpotensi merusak sistem pertahanan siber yang digunakan untuk menyerang *Cybersecurity Knowledge Graph* (CKG) dan *cybersecurity corpus*, yang menyebabkan dampak negatif seperti keluaran analisis yang keliru, pencemaran representasi data, serta korupsi pada sistem pertahanan siber.
16	(Moreira dkk., 2020)	*CNN*	AlexNet 96,00% *ResNet*-18 97,00%	Mengembangkan dan mengevaluasi *Packet Vision*, sebuah metode yang mengubah data mentah paket, termasuk *header* dan *payload*, menjadi citra yang dapat diolah oleh *CNN*. Pendekatan ini tidak hanya

No.	Penulis	Arsitektur	Akurasi	Fokus Pembahasan
			Squeeze Net 99,00%	meningkatkan performa klasifikasi tetapi juga menjamin keamanan dan privasi dengan mentransformasikan paket mentah menjadi citra. Kemudian, dievaluasi menggunakan tiga arsitektur *CNN* yaitu AlexNet, *ResNet*-18, dan SqueezeNet.
17	(Lim dkk., 2019)	LSTM dan *CNN*	93,63%	Memfokuskan pada pra-pemrosesan lalu lintas jaringan untuk menghasilkan dataset *payload* berbasis aliran, kemudian melatih dua model *deep learning* yaitu model *multi-layer Long Short-Term Memory* (LSTM) dan kombinasi *Convolutional Neural Network (CNN)* dengan *LSTM single-layer*. Penelitian ini juga melakukan prosedur penyetelan model untuk menemukan *hyperparameter* optimal, dengan tujuan akhir mengklasifikasikan lalu lintas jaringan secara otomatis tanpa intervensi operator jaringan menggunakan IP atau *port* dinamis, dengan fokus pada klasifikasi berbasis *payload*.
18	(Taher dkk., 2023)	EHHO-ANN	98,1%	Mengembangkan model hibrida *DroidDetectMW* untuk deteksi *malware* Android yang menggabungkan analisis statis dan dinamis menggunakan *Enhanced Harris Hawks Optimization - Artificial Neural Network* (EHHO-ANN) dengan teknik seleksi fitur berbeda yaitu *filter-based* (*chi-square, fisher score, information gain*) untuk fitur statis dan *fuzzy-metaheuristic* (MVO, EMFO) untuk fitur dinamis. Penelitian ini fokus pada pengembangan framework sistematik yang mengoptimalkan parameter ANN melalui EHHO dengan *Quasi-Reflection-based Learning* (QRL).

No.	Penulis	Arsitektur	Akurasi	Fokus Pembahasan
19	(Ashawa dkk., 2024)	*CNN*	99,62%	Mengembangkan model klasifikasi *malware* berbasis citra menggunakan *Convolutional Neural Network (CNN)* untuk menganalisis sampel *malware* pada dataset MalNet yang dikonversi dari *file executable* menjadi citra *grayscale*. Model menggunakan teknik *preprocessing* dengan *Principal Component Analysis* (PCA) untuk reduksi dimensionalitas, *edge detection* untuk ekstraksi fitur struktural, dan *Self-Organizing Map* (SOM) untuk *clustering* fitur dengan arsitektur *CNN*.
20	(Vinayakumar dkk., 2019)	*CNN*	98,9% pada dataset Ember, dan 93,6% pada dataset Malimg	Penelitian ini mengembangkan tiga arsitektur *deep learning* menjadi sebuah *ScaleMalNet* untuk deteksi *malware* dengan peran *CNN* pada *Windows-Static-Brain-Droid* (WSBD), *Windows-Dynamic-Brain-Droid* (WDBD), dan *DeepImageMalDetect* (DIMD). *CNN* digunakan sebagai varian dari MalConv dengan dua layer konvolusi dan *global max pooling* untuk memproses *raw byte sequences.*

Berdasarkan Tabel 2.1 mengenai penelitian terkait, telah banyak penelitian yang mengembangkan sistem deteksi berbasis *deep learning*, termasuk deteksi *malware*. Berbagai arsitektur *deep learning* seperti *CNN, RCNN, DRNN, Transformer, LSTM, BERT, CBAM, MLST, CGAN,* dan *ANN* telah diuji dan dibandingkan, baik secara terpisah maupun dalam bentuk kolaborasi secara *hybrid*. Di antara berbagai penelitian tersebut, beberapa penggunaan arsitektur *CNN* dan *Transformer* menunjukkan hasil yang baik dengan tingkat akurasi yang tinggi khususnya dalam mendeteksi *malware* pada lalu lintas jaringan. Adapun matriks

arsitektur penelitian terdahulu mengenai pendeteksi *malware* serta kontribusi penelitian ini dapat dilihat pada Tabel 2.2.

Tabel 2. 2 Matriks Penelitian

	CNN	***RCNN***	***DRNN***	***Transformer***	***LSTM***	***BERT***	***CBAM***	***CGAN***	***MLST***	***ANN***	***Hybrid***	***Single***
(Liu et al., 2024)	-	-	-	-	✓	✓	-	-	-	-	✓	-
(Xu, 2024)	-	-	-	✓	-	-	-	-	-	-	-	✓
(Dulal, 2024)	✓	-	-	-	-	-	-	-	-	-	-	✓
(Ferrag, 2024)	-	-	-	-	-	✓	-	-	-	-	-	✓
(Rahman et al., 2023)	✓	-	-	✓	-	-	-	-	-	-	-	✓
(Zhang, 2023)	-	-	-	✓	-	-	✓	-	-	-	✓	-
(Almeida, 2023)	-	-	-	-	-	✓	-	-	-	-	-	✓
(Yu et al., 2023)	-	-	-	-	-	-	-	-	✓	-	-	✓
(Tian, 2023)	-	✓	-	✓	-	-	-	-	-	-	✓	-
(Nalinipriya et al., 2022)	-	-	✓	-	-	-	-	-	-	-	-	✓
(Yang et al., 2022)	-	-	-	✓	-	-	-	✓	-	-	✓	-
(Seyyar et al., 2022)	-	-	-	-	-	✓	-	-	-	-	-	✓
(Rahali & Akhloufi, 2021)	-	-	-	✓	-	✓	-	-	-	-	✓	-
(Ranade et al., 2021)	-	-	-	-	-	✓	-	-	-	-	-	✓
(Ranade et al., 2021)	-	-	-	✓	-	-	-	-	-	-	-	✓
(Moreira et al., 2020)	✓	-	-	-	-	-	-	-	-	-	-	✓
(Lim et al., 2019)	✓	-	-	-	✓	-	-	-	-	-	✓	-
(Taher dkk., 2023)	-	-	-	-	-	-	-	-	-	✓	-	✓
(Ashawa dkk., 2024)	✓	-	-	-	-	-	-	-	-	-	-	✓

(Vinayakumar dkk., 2019)	✓	-	-	-	-	-	-	-	-	-	-	-
Penelitian ini	✓	-	-	✓	-	-	-	-	-	-	✓	-

Tabel 2.2, penelitian terdahulu mengembangkan deteksi *malware* yang mengadopsi pendekatan *hybrid* untuk meningkatkan akurasi dan efisiensi sistem deteksi *malware*. Salah satu contoh utama pada penelitian sebelumnya yaitu mengenai *SeMalBERT* yang mengkolaborasikan *BERT* dengan *ConvLSTM*. Pembuatan model deteksi *malware* pada penelitian ini mengekstraksi pola menggunakan ekspresi reguler dan analisis urutan panggilan *API*, dilanjutkan dengan pra-pemrosesan data teks menggunakan *Natural Language Toolkit* dan transformasi data menjadi representasi vektor. Arsitektur *BERT* digunakan untuk memahami konteks dan makna teks terkait *malware*, sementara *ConvLSTM-AM* berperan dalam menangkap pola temporal dan spasial dari data. Kedua model ini diintegrasikan secara bertahap, diikuti dengan pelatihan dan evaluasi model untuk meningkatkan akurasi deteksi. Kolaborasi antara *BERT* dan *ConvLSTM-AM* memungkinkan model untuk memanfaatkan keunggulan masing-masing arsitektur dalam menghasilkan model deteksi *malware* yang lebih kuat dan efektif (Liu et al., 2024).

Pendekatan *hybrid* lainnya menunjukkan inkonsistensi performa yang signifikan. Penelitian (Tian, 2023) dengan R-CNN + Transformer mencapai akurasi 96,0%, sementara penelitian (Yang dkk., 2022) dengan CGAN + Transformer mencapai 95,0%, serta penelitian (Lim dkk., 2019) dengan kombinasi LSTM + CNN yang mencapai akurasi 93,63%, menunjukkan bahwa tidak semua kombinasi arsitektur memberikan hasil yang superior namun membuka peluang eksplorasi

yang tinggi untuk optimasi dalam hal akurasi dalam pengembangan model deteksi *malware* berbasis *hybrid*. Selain fokus pada arsitektur pengembangan model, penelitian sebelumnya juga berfokus pada optimasi model dan peningkatan kinerja. Pada penelitian mengenai kemampuan generalisasi optimizer *SGD, Adam, dan AdamW* menunjukkan bahwa *AdamW* memiliki kemampuan generalisasi yang lebih baik, efisiensi pembelajaran yang lebih tinggi meskipun dengan *epoch* yang terbatas, serta keseimbangan yang optimal (Suhaili dkk., 2024). Selain itu, *AdamW* merupakan modifikasi dari algoritma *Adam* yang memisahkan secara eksplisit pembaruan bobot *(weights)* dengan proses *weight decay*, sehingga regularisasi dapat dilakukan dengan lebih tepat dan konsisten. Hal ini berbeda dengan *Adam* konvensional, yang menggabungkan *weight decay* langsung dalam gradien yang terpengaruh momen, berpotensi menyebabkan *under-penalization* pada parameter model. Penelitian oleh (Loshchilov & Hutter, 2019) menunjukkan bahwa *AdamW* menghasilkan generalisasi yang lebih baik dan meminimalkan risiko *overfitting* karena regularisasi *weight decay* bekerja secara orthogonal terhadap proses adaptasi *learning rate.*

Berdasarkan hal tersebut, terdapat peluang penelitian dalam pengembangan model deteksi malware secara *hybrid*, salah satunya dengan kolaborasi antara *Transformer* dan *CNN*. Arsitektur *Transformer* unggul dalam memahami keterkaitan fitur secara global melalui mekanisme *self-attention* untuk memperluas konteks dengan mempertimbangkan dependensi global antar fitur, sementara arsitektur *CNN* dikenal kemampuannya dalam mengekstraksi fitur spasial dari data sehingga mampu mengidentifikasi pola kompleks dalam lalu lintas jaringan dan

menyaring informasi lokal yang relevan. Oleh karena itu, penelitian ini bertujuan untuk mengisi gap dengan mengusulkan model *hybrid* yang menggabungkan keunggulan kedua arsitektur tersebut dengan mengimplementasikan *optimizer AdamW* dalam peningkatan generalisasi dan stabilitas model.