Model Deep Learning Convolutional Neural Network untuk Data Terenkripsi oleh Skema Cheon-Kim-Kim-Song (CKKS)

Rayhan Kinan Muhannad
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): rayhankinan@gmail.com

***Abstract*—Penelitian ini mengembangkan model *Convolutional Neural Network* (CNN) yang mampu memproses data terenkripsi menggunakan skema enkripsi homomorfik penuh Cheon-Kim-Kim-Song (CKKS) untuk menjaga privasi dan keamanan data. Implementasi dilakukan dengan membangun lima jenis layer utama—*convolutional*, *pooling*, *fully-connected*, *activation*, dan *loss layer*—serta mengaproksimasi fungsi nonlinear menggunakan metode *minimax* dan *least-squares* agar sesuai dengan keterbatasan operasi CKKS yang hanya mendukung penjumlahan dan perkalian. Hasil eksperimen menunjukkan bahwa CNN terenkripsi dapat direplikasi dengan tingkat galat yang rendah dan kinerja yang mendekati model *plaintext*, meskipun terdapat sedikit perbedaan akurasi akibat sifat aproksimasi dan penambahan *noise* pada skema CKKS. Penelitian ini membuktikan potensi penerapan enkripsi homomorfik penuh dalam pembelajaran mendalam untuk data sensitif tanpa mengorbankan akurasi secara signifikan.**

Keywords*—*model convolutional neural network, skema enkripsi Cheon-Kim-Kim-Song (CKKS), aproksimasi fungsi

I. PENDAHULUAN

Dalam beberapa tahun terakhir, terdapat perkembangan signifikan dalam bidang ilmu pembelajaran mesin, khususnya model *deep learning* seperti GPT dan BERT, dimana model – model tersebut menuntut penggunaan data dalam jumlah besar. Namun, banyak data yang bersifat sensitif, seperti rekam medis dan informasi finansial, sehingga diperlukan mekanisme pengamanan untuk mencegah kebocoran data. Salah satu pendekatan yang umum digunakan adalah *federated learning*, yang dimana pengguna dapat melatih model secara terdistribusi tanpa mengirimkan data sensitif ke server. Meskipun mengurangi risiko kebocoran, metode ini memiliki keterbatasan seperti bias akibat heterogenitas data dan ukuran model yang membesar seiring jumlah pengguna.

Sebagai alternatif dari *federated learning*, pengembangan model tersentralisasi dengan data terenkripsi menjadi solusi yang lebih efisien. Data yang dikirim ke server harus dienkripsi agar tetap aman meskipun terjadi kebocoran. Namun, enkripsi konvensional tidak mendukung operasi aritmatika yang diperlukan dalam pembelajaran mesin. Untuk itu, digunakan enkripsi homomorfik yang memungkinkan operasi penjumlahan dan perkalian langsung pada *ciphertext*, sehingga proses *forward* dan *backward propagation* dapat dilakukan tanpa dekripsi.

Terdapat beberapa algoritma enkripsi homomorfik penuh, seperti BGV, BFV, TFHE, dan CKKS. Algoritma CKKS dipilih untuk digunakan pada penelitian ini karena mendukung operasi aritmatika berbasis *floating point*, yang sesuai dengan representasi bobot dan bias pada model pembelajaran mendalam. Meskipun algoritma CKKS mengorbankan sedikit tingkat ketelitian, hal ini dapat diterima karena kedua proses *forward* dan *backward propagation* tidak memerlukan presisi tinggi. Dengan demikian, algoritma CKKS menjadi kandidat ideal untuk mengimplementasikan model CNN pada data terenkripsi.

Penelitian sebelumnya telah mengimplementasikan model seperti *logistic regression* dan *random forest* pada data terenkripsi, namun belum ada yang membahas model CNN untuk data citra. Model CNN memiliki keunggulan dalam menangani data dengan korelasi spasial, sehingga cocok untuk dataset citra yang terenkripsi. Oleh karena itu, penelitian ini berfokus pada pengembangan model CNN yang dapat memproses data citra terenkripsi menggunakan skema algoritma CKKS, serta mengevaluasi kinerjanya dibandingkan dengan model *plaintext*.

II. STUDI LITERATUR

A. Enkripsi Homomorfik

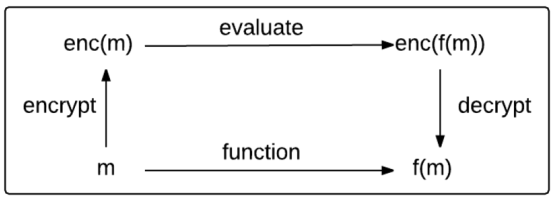

Fig. 1. Skema Kriptografi Enkripsi Homomorfik

Enkripsi homomorfik adalah teknik enkripsi yang memungkinkan operasi komputasi dilakukan langsung pada

data terenkripsi (*ciphertext*) tanpa perlu mendekripsinya terlebih dahulu. Hasil komputasi tersebut, ketika didekripsi, akan setara dengan hasil komputasi yang dilakukan pada data asli (*plaintext*). Sifat ini didasarkan pada konsep homomorfisme dalam aljabar yang mempertahankan operasi matematis antara dua struktur. Algoritma enkripsi homomorfik dapat bersifat aditif, multiplikatif, atau keduanya. Jika hanya mendukung satu operasi, maka akan disebut sebagai homomorfik sebagian, sedangkan jika mendukung penjumlahan dan perkalian, maka akan disebut sebagai homomorfik penuh. Contoh algoritma homomorfik penuh adalah BGV, BFV, TFHE, dan CKKS, yang umum digunakan untuk menjaga privasi data dalam komputasi seperti *secure voting*, *privacy-preserving audits*, dan *outsourced machine learning*.

Keunggulan utama enkripsi homomorfik adalah kemampuannya menjaga keamanan data sensitif sambil tetap memungkinkan pemrosesan. Namun, setiap algoritma memiliki karakteristik berbeda. Misalnya algoritma CKKS dirancang untuk mendukung operasi aritmatika berbasis *floating point*, sehingga lebih efisien untuk pembelajaran mesin dibandingkan algoritma lain yang hanya mendukung bilangan bulat atau biner. Meskipun demikian, algoritma CKKS mengorbankan sedikit ketelitian karena menggunakan pendekatan *approximate arithmetic*, tetapi hal ini dapat diterima untuk aplikasi seperti *machine learning* yang tidak memerlukan presisi tinggi. Dengan kemampuan ini, CKKS menjadi pilihan ideal untuk mengimplementasikan model pembelajaran mendalam seperti CNN pada data terenkripsi.

B. *Skema Enkripsi dengan Algoritma CKKS*

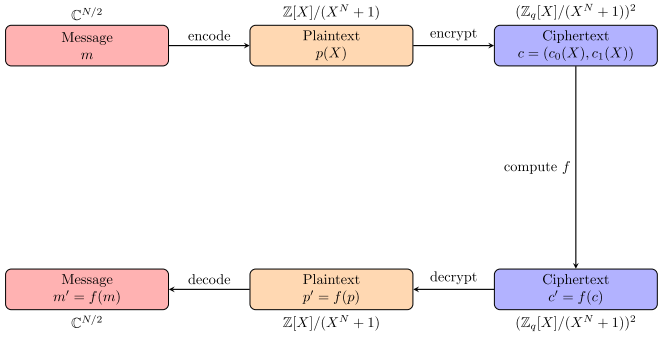

Fig. 2. Skema Enkripsi dengan Algoritma CKKS

Skema enkripsi pada algoritma CKKS terdiri dari lima tahap utama: *encoding* vektor bilangan kompleks menjadi *polynomial quotient ring*, *encryption* menggunakan metode *ring learning with errors*, operasi homomorfik (penjumlahan dan perkalian) pada *ciphertext*, *decryption* untuk mengembalikan ke bentuk *plaintext*, dan *decoding* untuk merekonstruksi pesan asli. Algoritma CKKS juga mendukung fitur tambahan seperti *relinearization* untuk mengurangi kompleksitas hasil perkalian dan *rescaling* untuk menjaga skala nilai agar tidak terjadi *overflow* selama operasi berulang. Dengan kemampuan ini, algoritma CKKS memungkinkan pemrosesan data terenkripsi secara efisien sambil tetap menjaga keamanan dan privasi.

Misalkan terdapat sebuah *message* z dengan bentuk vektor bilangan kompleks, $z \in \mathbb{C}^{N/2}$, dengan nilai N merupakan bilangan dua dipangkatkan dengan suatu bilangan bulat positif, $N = 2^k$ dan $k \in \mathbb{Z}^+$. Proses *encoding* melakukan transformasi *message* tersebut menjadi sebuah *plaintext* berbentuk *polynomial quotient ring* dengan domain $\mathcal{R}_{q_\ell}$, dengan q_ℓ merupakan nilai maksimum dari koefisien polinom pada *multiplicative depth* ℓ. Berikut ini merupakan persamaan transformasi pada proses *encoding* dan *decoding* dari algoritma CKKS.

$$m = Ecd(z, \Delta) = \sigma^{-1}\left(\lfloor \Delta \cdot \pi^{-1}(z) \rceil_{\sigma(\mathcal{R}_{q_\ell})}\right) \in \mathcal{R}_{q_\ell}$$

$$z = Dcd(m, \Delta) = \pi\left(\Delta^{-1} \cdot \sigma(m)\right) \in \mathbb{C}^{N/2}$$

Dengan fungsi $\pi(z)$ merepresentasikan fungsi penghilangan elemen konjugat dari sebuah vektor, variabel Δ merepresentasikan skala ketelitian, operasi $\lfloor x \rceil$ merupakan pembulatan *coordinate-wise random rounding*, dan fungsi $\sigma(z)$ merepresentasikan operasi *canonical embedding*.

Proses pembangkitan kunci pada CKKS didasarkan pada konsep *ring learning with errors*. Kunci privat sk diambil secara acak dari domain polinomial $\mathcal{R}_{q_\ell}$. Selanjutnya, sebuah polinom acak A dan Gaussian *noise* e dibangkitkan. Kunci publik (pk) dapat dihitung menggunakan persamaan berikut ini.

$$pk = (-A \cdot sk + e, A) \bmod q_\ell \in {\mathcal{R}_{q_\ell}}^2$$

Misalkan terdapat suatu *plaintext* $\mu \in \mathcal{R}_{q_\ell}$ yang hendak dienkripsi dengan kunci publik $pk \in {\mathcal{R}_{q_\ell}}^2$, maka berikut ini merupakan fungsi untuk melakukan enkripsi pada algoritma CKKS.

$$c = Enc_{pk}(\mu) = (\mu, 0) + pk = (\mu - A \cdot sk + e, A) \bmod q_\ell \in {\mathcal{R}_{q_\ell}}^2$$

Misalkan pula terdapat suatu *ciphertext* $c = (c_0, c_1)$ yang hendak didekripsi dengan kunci privat $sk \in \mathcal{R}_{q_\ell}$, maka berikut ini merupakan fungsi untuk melakukan dekripsi pada algoritma CKKS.

$$\mu' = Dec_{sk}(c) = (c_0 + c_1 \cdot sk) \bmod q_\ell \in \mathcal{R}_{q_\ell}$$

Misalkan terdapat suatu *ciphertext* $c = (c_0, c_1)$ yang akan dioperasikan dengan *plaintext* μ'. Berikut ini merupakan persamaan yang digunakan untuk melakukan operasi penambahan antara *ciphertext* dan *plaintext* pada algoritma CKKS.

$$c_{add} = (c_0 + \mu', c_1) \bmod q_\ell$$

Selain operasi pertambahan, dalam algoritma CKKS juga dapat dilakukan operasi perkalian antara *ciphertext* dan *plaintext*. Berikut ini merupakan persamaan yang digunakan untuk melakukan operasi perkalian antara *ciphertext* dan *plaintext*.

$$c_{mult} = (c_0 \cdot \mu', c_1 \cdot \mu') \bmod q_\ell$$

Misalkan terdapat suatu *ciphertext* $c = (c_0, c_1)$ yang akan dioperasikan dengan *ciphertext* $c' = (c'_0, c'_1)$. Berikut ini merupakan persamaan yang digunakan untuk melakukan operasi penambahan antar dua *ciphertext* pada algoritma CKKS.

$$c_{add} = (c_0 + c'_0, c_1 + c'_1) \bmod q_\ell$$

Selain operasi pertambahan, dalam algoritma CKKS juga dapat dilakukan operasi perkalian antara *ciphertext* dengan *ciphertext* lainnya. Berikut ini merupakan persamaan yang digunakan untuk melakukan operasi perkalian antar dua *ciphertext*

$$c_{mult} = (c_0 \cdot c_0', c_0 \cdot c_1' + c_1 \cdot c_0', c_1 \cdot c_1') \bmod q_\ell$$

Dapat dilihat bahwa pada hasil perkalian antar dua *ciphertext* akan menghasilkan bentuk tiga polinomial, tidak dalam bentuk dua polinomial. Oleh karena itu, dibutuhkan operasi *relinearization* untuk mentransformasi hasil perkalian tersebut untuk mengurangi jumlah polinomial tanpa mengubah hasil nilai *ciphertext*.

Tahap pertama yang dilakukan dalam melakukan *relinearization* adalah membangkitkan kunci evaluasi. Misalkan P merupakan bilangan bulat positif dengan nilai yang cukup besar. Selanjutnya, sebuah polinom acak A' dan Gaussian *noise* e' dibangkitkan. Kunci evaluasi (evk) dapat dibangkitkan dengan menggunakan persamaan berikut.

$$evk = (-A' \cdot sk + e' + P \cdot sk^2, A') \bmod P \cdot q_\ell$$

Kunci evaluasi tersebut akan digunakan untuk melakukan transformasi *ciphertext* hasil perkalian dari bentuk tiga polinomial kembali menjadi bentuk dua polinomial. Persamaan *relinearization* dengan menggunakan kunci evaluasi adalah sebagai berikut ini.

$$Relin_{evk}(c_{mult}) = (d_0, d_1) + \lfloor P^{-1} \cdot d_2 \cdot evk \rceil \bmod q_\ell$$

Hasil operasi *relinearization* yang dilakukan akan memiliki nilai skala sebesar kuadrat dari nilai skala *ciphertext* sebelumnya. Hal ini akan menyebabkan *overflow* ketika operasi perkalian tersebut dilakukan beberapa kali dikarenakan nilai skala tersebut akan tumbuh secara eksponensial. Oleh karena itu, akan dilakukan operasi *rescaling* untuk menjaga nilai skala tetap konstan dan mengurangi *noise* yang terdapat pada *ciphertext*. Berikut ini merupakan persamaan yang digunakan untuk melakukan operasi *rescaling* pada *ciphertext*.

$$RS_{\ell \to \ell-1}(c_{relin}) = \lfloor \Delta^{-1} \cdot c_{relin} \rceil \bmod q_{\ell-1}$$

C. *Convolutional Neural Network*

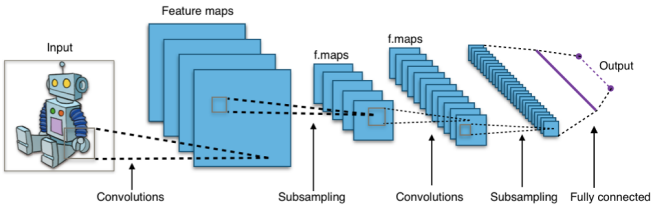

Fig. 3. Contoh Arsitektur *Convolutional Neural Network*

Convolutional Neural Network (CNN) merupakan suatu model *feed-forward neural network* yang dirancang untuk memproses data berbentuk *grid*, seperti citra, dengan memanfaatkan korelasi spasial antar piksel. Model CNN terdiri dari tiga komponen utama: *convolutional layer* untuk mengekstraksi fitur melalui operasi konvolusi dengan *kernel*, *pooling layer* untuk melakukan *downsampling* dan mengurangi dimensi data, serta *fully connected layer* untuk menghasilkan prediksi akhir. Setiap *layer* dilengkapi fungsi aktivasi nonlinear agar model mampu mempelajari representasi kompleks dari data. Model CNN umumnya digunakan dalam bidang *computer vision* seperti deteksi objek, pengolahan citra medis, dan sistem pengenalan wajah, karena kemampuannya melakukan pembelajaran fitur secara otomatis tanpa memerlukan ekstraksi manual.

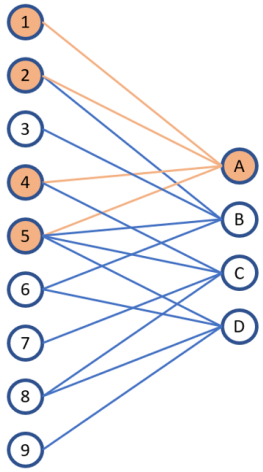

Fig. 4. Visualisasi *Convolutional Layer* sebagai *Neuron*

Convolutional layer adalah komponen utama dari model CNN yang berfungsi untuk mengekstraksi fitur dari citra melalui operasi konvolusi menggunakan kernel berukuran kecil, seperti 3×3 atau 5×5. Hasil konvolusi disebut *feature map*, yang merepresentasikan pola penting seperti tepi atau tekstur. *Layer* ini memanfaatkan konsep *sparse local connectivity* untuk mengurangi kompleksitas perhitungan dengan hanya menghubungkan piksel yang berdekatan. Tiga *hyperparameter* utama mengatur ukuran *feature map*: *depth* (jumlah filter), *stride* (jarak pergeseran kernel), dan *padding* (penambahan piksel di tepi citra). Setelah konvolusi, diterapkan fungsi aktivasi nonlinear untuk menambahkan sifat nonlinier pada model, sehingga model CNN mampu mempelajari representasi yang lebih kompleks.

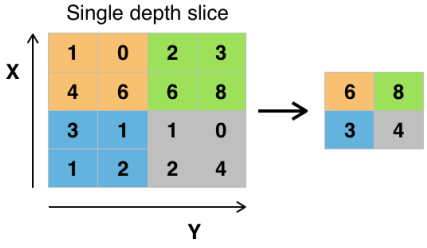

Fig. 5. Contoh *Maximum Pooling* pada sebuah *Feature Map*

Pooling layer adalah komponen dari model CNN yang berfungsi melakukan *downsampling* terhadap *feature map* untuk mengurangi dimensi data, jumlah parameter, dan kompleksitas komputasi. Operasi *pooling* dilakukan dengan membagi *feature map* ke dalam partisi kecil dan menerapkan fungsi nonlinear pada setiap partisi, seperti *maximum pooling* atau *average pooling*. Dengan mengurangi ukuran spasial, operasi *pooling* dapat membantu mengurangi risiko *overfitting* dan mempercepat proses pelatihan, sekaligus mempertahankan informasi penting dari citra.

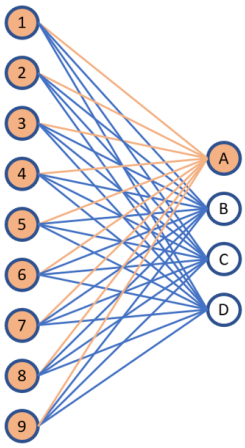

Fig. 6. Visualisasi *Fully Connected Layer* sebagai *Neuron*

Fully connected layer adalah bagian akhir dari arsitektur CNN yang menghubungkan seluruh *neuron* dari layer sebelumnya ke setiap *neuron* pada layer selanjutnya. Fungsi dari *layer* ini adalah mengubah *feature map* hasil ekstraksi menjadi vektor yang merepresentasikan kelas keluaran. Setiap *neuron* memiliki parameter berupa matriks *weight* dan vektor bias yang digunakan dalam transformasi linear, kemudian hasilnya dilewatkan ke fungsi aktivasi untuk menambahkan sifat nonlinier. *Layer* ini berperan penting dalam proses klasifikasi karena *layer* ini akan menggabungkan seluruh informasi yang telah diekstraksi oleh *layer* sebelumnya menjadi prediksi akhir.

D. *Approximation Theory*

Approximation theory membahas bagaimana fungsi kontinu dapat didekati dengan polinomial agar fungsi tersebut dapat diaplikasikan pada data terenkripsi. Hal ini dikarenakan skema algoritma CKKS hanya mendukung operasi penjumlahan dan perkalian. Teorema Weierstrass menyatakan bahwa setiap fungsi kontinu pada interval tertutup dapat diaproksimasi secara seragam oleh suatu polinomial. Dalam konteks model CNN, fungsi aktivasi seperti *sigmoid* dan *tanh* harus diaproksimasi agar dapat digunakan pada *ciphertext*. Dua metode yang umum untuk digunakan untuk melakukan aproksimasi dengan polinomial adalah metode *least-squares function approximation* serta *minimax approximation*. Kedua metode ini memungkinkan penggantian fungsi nonlinear dengan polinomial sehingga operasi tetap kompatibel dengan enkripsi homomorfik.

Metode pertama yang dapat dilakukan untuk melakukan aproksimasi fungsi kontinu dengan menggunakan polinomial adalah *least-squares function approximation*. Pada metode ini, nilai *error* didefinisikan sebagai luas yang dihasilkan antara kedua fungsi ketika dipetakan pada suatu domain (*least-squares difference*). Oleh karena itu, tujuan dari dilakukannya metode ini adalah untuk meminimalisir nilai *error* dengan mencari polinomial yang menghasilkan jumlah *least-squares difference* yang paling kecil. Berikut ini merupakan persamaan yang merepresentasikan nilai *error* dari *least-squares difference.*

$$error = \int_a^b \left(f(x) - p_d(x)\right)^2 dx$$

Metode kedua yang dapat dilakukan untuk melakukan aproksimasi fungsi kontinu dengan menggunakan polinomial adalah *minimax approximation*. Pada metode ini, nilai *error* didefinisikan sebagai nilai maksimum dari perbedaan antara kedua fungsi ketika dipetakan pada suatu domain (*maximum difference*). Oleh karena itu, tujuan dari dilakukannya metode ini adalah untuk meminimalisir nilai error dengan mencari polinomial yang menghasilkan *maximum difference* yang paling kecil. Berikut ini merupakan persamaan yang merepresentasikan nilai *error* dari *maximum difference*.

$$error = \max_{a \le x \le b} |f(x) - p_d(x)|$$

III. Analisis Masalah dan Perancangan Solusi

A. *Deskripsi Persoalan*

Penggunaan enkripsi homomorfik pada model CNN bertujuan menjaga privasi data tanpa mengurangi kemampuan pemrosesan. Tantangan utama adalah memodifikasi proses *forward propagation* agar dapat menerima data dalam bentuk *ciphertext* dan tetap menghasilkan keluaran terenkripsi tanpa dekripsi. Selain itu, seluruh operasi aritmatika dalam model CNN harus disesuaikan agar kompatibel dengan skema algoritma CKKS yang hanya mendukung penjumlahan dan perkalian. Untuk fungsi nonlinear yang tidak dapat direplikasi secara langsung, diperlukan aproksimasi polinomial agar operasi tetap berjalan pada data terenkripsi.

B. *Perancangan Solusi*

Terdapat lima bagian modul yang akan dikembangkan pada penelitian ini, yaitu *convolutional layer*, *pooling layer*, *fully connected layer*, *activation layer*, *loss layer*. Seluruh modul yang akan dikembangkan akan berperan sebagaimana modul tersebut pada kakas pembelajaran mesin, namun untuk data yang terenkripsi dengan skema CKKS. Oleh karena itu, seluruh data masukan dan luaran dari modul ini akan berbentuk *ciphertext* baik pada proses *training* pada model maupun proses *evaluation* pada model.

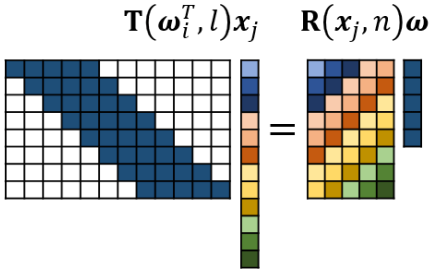

Fig. 7. Skema Konvolusi dengan Matriks Toeplitz

Modul *convolutional layer* dirancang untuk memungkinkan operasi konvolusi pada citra yang terenkripsi tanpa proses dekripsi karena akses langsung ke elemen data tidak dimungkinkan pada *ciphertext*. Untuk mengatasi keterbatasan ini, digunakan pendekatan berbasis matriks Toeplitz yang mengubah struktur kernel konvolusi menjadi vektor 1-D dan citra menjadi matriks yang sesuai. Dengan metode ini, operasi konvolusi dapat direpresentasikan sebagai perkalian matriks,

yang kompatibel dengan skema CKKS karena hanya memerlukan operasi penjumlahan dan perkalian.

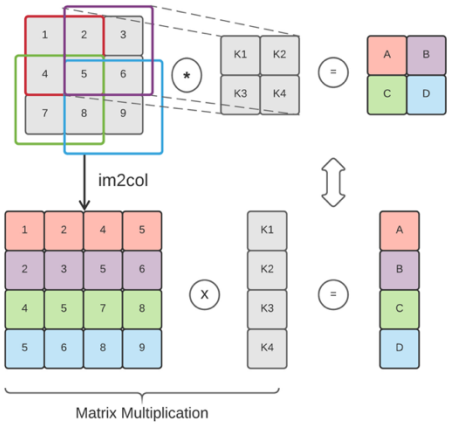

Fig. 8. Proses Transformasi Citra 2-D menjadi Matriks Toeplitz

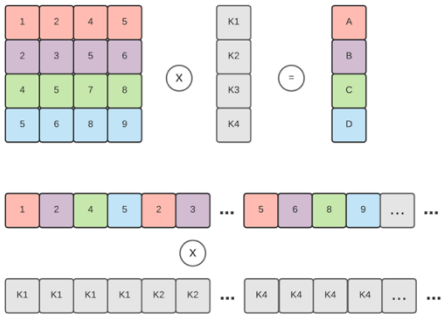

Fig. 9. Proses *Vertical Scan* pada Matriks Toeplitz

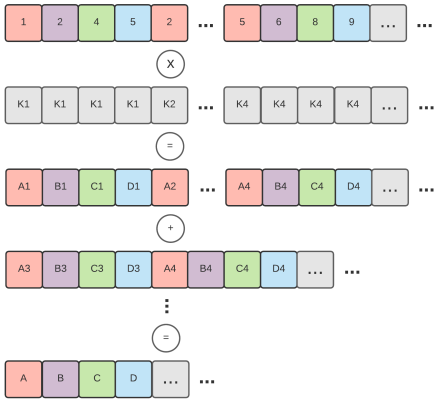

Fig. 10. Proses Perkalian *Hasil Vertical Scan* pada Matriks Toeplitz

Implementasi modul ini juga mencakup teknik *vertical scan* untuk mengonversi matriks terenkripsi menjadi vektor, serta melakukan replikasi elemen kernel agar dapat dilakukan perkalian dengan vektor terenkripsi. Setelah perkalian, digunakan operasi rotasi dan penjumlahan untuk menghitung hasil konvolusi sesuai posisi *window*. Modul ini mendukung parameter seperti ukuran keluaran, jumlah *channels*, ukuran *kernel*, *stride*, dan *padding*. Selain itu, proses *backpropagation* mencakup perhitungan tiga jenis gradien: *input gradient*, *kernel gradient*, dan bias *gradient*, yang diperlukan untuk memperbarui parameter model.

Modul *pooling layer* dirancang untuk melakukan operasi *average pooling* pada citra terenkripsi tanpa memerlukan proses dekripsi karena akses langsung ke elemen data tidak dimungkinkan. Solusi yang digunakan adalah memanfaatkan metode matriks Toeplitz, mirip dengan rancangan *convolutional layer*, namun *kernel* yang digunakan bersifat statis dan tidak berubah selama proses *training*. Modul ini mendukung parameter seperti ukuran *kernel*, *stride*, dan *padding* untuk menentukan konfigurasi *pooling*. Pada tahap *backpropagation*, hanya dihitung *input gradient* karena tidak ada parameter yang diperbarui, sehingga implementasi lebih sederhana dibandingkan *layer* konvolusi.

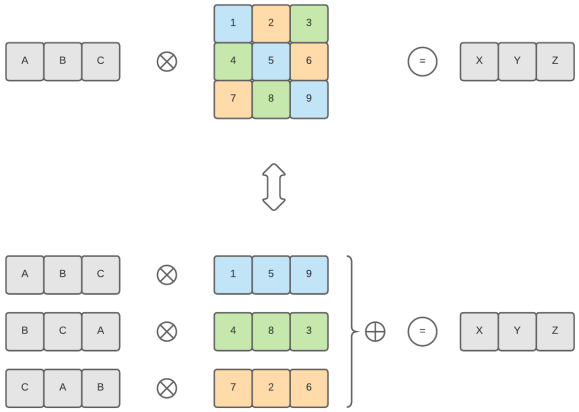

Fig. 11. Proses Perkalian dengan Metode Halevi dan Shoup

Modul *fully connected layer* dirancang untuk melakukan transformasi *affine* pada data terenkripsi yang menghubungkan seluruh *neuron* dari layer sebelumnya ke *neuron* pada *layer* ini. Dikarenakan skema CKKS tidak mendukung perkalian langsung antara vektor *ciphertext* dan matriks *plaintext*, digunakan metode perkalian diagonal yang dikembangkan oleh Halevi dan Shoup, dimana metode tersebut merupakan kombinasi antara operasi rotasi dan penjumlahan untuk mereplikasi operasi perkalian matriks. Modul ini mendefinisikan dua parameter utama, yaitu matriks *weight* dan vektor bias, yang dapat diperbarui melalui proses *backpropagation* dengan menghitung tiga jenis gradien: *input gradient*, *weight gradient,* dan bias *gradient*. Pendekatan ini memungkinkan operasi linear dilakukan secara efisien pada data terenkripsi tanpa mengorbankan struktur model CNN.

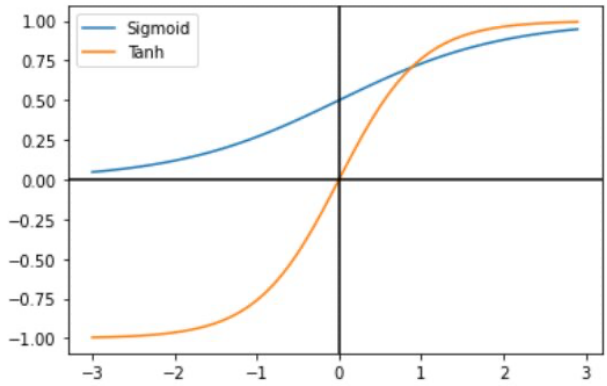

Fig. 12. Grafik Fungsi *Sigmoid* dan *Tanh*

Modul *activation layer* dirancang untuk mengaplikasikan fungsi aktivasi nonlinear pada data terenkripsi, yang tidak dapat direplikasi langsung karena keterbatasan skema CKKS yang hanya mendukung penjumlahan dan perkalian. Untuk mengatasi hal ini, fungsi aktivasi seperti *square*, *sigmoid*, dan *tanh* diaproksimasi menggunakan polinomial melalui metode *least-squares* atau *minimax approximation*. Aproksimasi ini memungkinkan operasi dilakukan pada *ciphertext* tanpa dekripsi, sehingga sifat nonlinier tetap terjaga. Modul ini juga mendukung parameter seperti rentang aproksimasi, jumlah sampel, derajat polinomial, dan metode aproksimasi yang digunakan. Pada tahap *backpropagation*, hanya dihitung *input gradient* karena layer ini tidak memiliki parameter yang diperbarui.

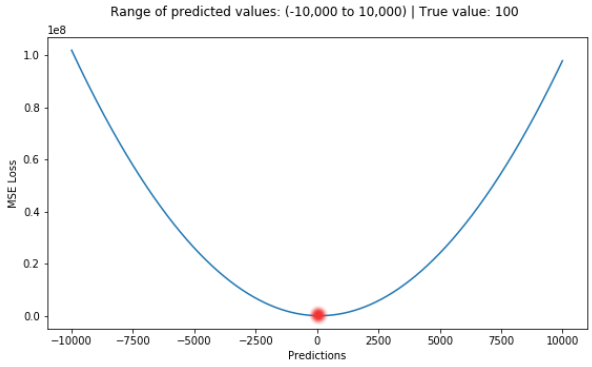

Fig. 13. Grafik Fungsi *Mean-Squared Error Loss*

Modul *loss layer* dirancang untuk menghitung *mean-squared error* (MSE) pada data terenkripsi, karena fungsi ini hanya memerlukan operasi pengurangan dan perkalian yang didukung oleh skema CKKS. *Layer* ini digunakan untuk mengukur selisih antara keluaran model dan target, sehingga dapat digunakan dalam proses evaluasi maupun *backpropagation*. Pada tahap *backpropagation*, hanya dihitung *input gradient* karena *layer* ini tidak memiliki parameter yang diperbarui, sehingga implementasinya relatif sederhana dan efisien.

IV. Implementasi dan Pengujian

A. Implementasi

Implementasi dilakukan pada perangkat keras dengan spesifikasi Apple M1 Pro 10-Core CPU, memori 16 GB, dan penyimpanan SSD 512 GB, serta perangkat lunak berbasis macOS Sonoma dengan dukungan pustaka seperti Python 3.9.6, Microsoft SEAL 4.1.1, TenSEAL 0.3.14, dan *framework* PyTorch untuk pengembangan model CNN terenkripsi. Batasan implementasi mencakup penggunaan skema CKKS dari Microsoft SEAL melalui TenSEAL tanpa pengembangan algoritma enkripsi baru, serta keterbatasan eksekusi pada CPU karena TenSEAL tidak mendukung akselerasi GPU. Selain itu, proses pelatihan model dengan parameter terenkripsi tidak dilakukan, sehingga pengujian hanya berfokus pada inferensi menggunakan parameter yang telah dilatih sebelumnya.

Arsitektur yang digunakan pada pengujian ini direferensikan dari model arsitektur yang digunakan oleh Benaissa, et al. (2021). Berikut merupakan diagram representasi dari arsitektur model CNN yang digunakan di dalam pengujian ini:

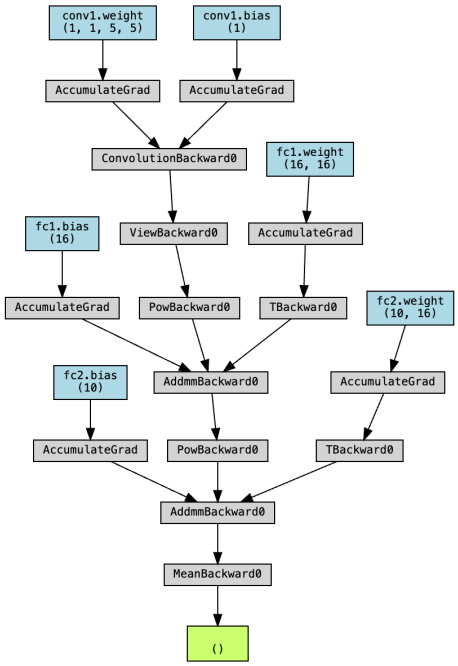

Fig. 14. Diagram Arsitektur Model CNN

B. Pengujian dan Analisis

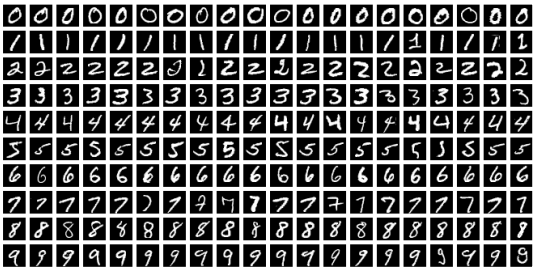

Fig. 15. Cuplikan Citra MNIST *Dataset of Handwritten Digits*

Pengujian terhadap model *encrypted* CNN bertujuan untuk memastikan bahwa perhitungan gabungan antara *encrypted layer* masih dapat menghasilkan nilai *loss* serta tingkat akurasi yang sama dengan model *plaintext* CNN. Dalam tahap ini, terlebih dahulu disiapkan dataset bernama *MNIST Dataset of Handwritten Digits*. Dataset tersebut memiliki ukuran sebesar 60.000 citra untuk *train set* dan 10.000 citra untuk *test set*, dimana setiap citra berukuran 28×28 piksel. Operasi *downsampling* akan diaplikasikan pada citra tersebut sehingga berukuran 14×14 piksel. Hal ini dilakukan untuk membatasi jumlah komputasi yang dibutuhkan pada model *encrypted* CNN.

Tahap berikutnya adalah melatih model *plaintext* CNN dengan menggunakan *train set* dan melakukan pengujian terhadap model tersebut dengan menggunakan *test set*. Untuk membatasi waktu komputasi yang diperlukan di tahap ini (terutama pada model *encrypted* CNN), akan dilakukan *random sampling* terhadap *train set*. Dari 10.000 citra yang terdapat pada *test set*, akan diambil 50 citra secara acak.

Tahap selanjutnya yang dilakukan adalah melakukan proses konvolusi dengan menggunakan citra terenkripsi, seluruh parameter *kernel* akan diekspansi dengan menggunakan metode matriks Toeplitz. Setelah diekspansi, seluruh parameter pada model CNN tersebut akan dienkripsi menggunakan skema CKKS. Tahap ekspansi matriks Toeplitz tersebut dilakukan untuk melakukan aproksimasi terhadap operasi konvolusi dengan operasi perkalian antara matriks terenkripsi.

Tahap terakhir yang dilakukan adalah memasukkan pengujian terhadap kedua model *plaintext* CNN dan model *ciphertext* CNN dengan menggunakan *random sampled test set* yang terdiri atas 50 citra. Terdapat dua metrik yang akan diambil dari pengujian ini, yaitu nilai *average loss* dari setiap *batch* pengujian serta nilai total akurasi yang didapatkan. Berikut ini merupakan tabel yang berisi nilai *average loss* dan total akurasi dari kedua model *plaintext* CNN dan *ciphertext* CNN.

	Plaintext	*Ciphertext*	Δ_{loss}
Average Loss	0.299081	0.298175	0.000906

Tabel 1. Perbandingan Nilai *Average Loss* dari Pengujian *Encrypted Convolutional Neural Network*

	Plaintext	*Ciphertext*	$\Delta_{accuracy}$
Accuracy	94%	96%	2%

Tabel 2. Perbandingan Nilai Total Akurasi dari Pengujian *Encrypted Convolutional Neural Network*

Terlihat pada dua tabel di atas bahwa nilai galat Δ_{loss} menunjukkan nilai yang relatif kecil $(\Delta < 0.01)$. Hal tersebut mengimplikasikan bahwa kedua model *plaintext* CNN dan *encrypted* CNN memiliki kemampuan inferensi yang sama dan tidak terpengaruh secara signifikan oleh keberadaan galat.

Selain itu, kedua model *plaintext* CNN dan *encrypted* CNN memiliki tingkat akurasi yang hampir sama, dengan perbedaan sebesar 2%. Hal tersebut juga mengimplikasikan bahwa meskipun nilai *average loss* yang didapatkan oleh kedua model hampir sama, namun galat dapat mempengaruhi kategori mana yang memiliki nilai maksimum pada vektor *output*.

Penyebab dari keberadaan galat pada hasil inferensi *encrypted* CNN jika dibandingkan dengan *plaintext* CNN adalah sifat dari operasi homomorfik pada skema enkripsi CKKS. Terdapat tiga operasi yang dapat menyebabkan keberadaan galat pada skema enkripsi CKKS, yaitu operasi *encoding*, operasi *rescaling*, serta penambahan *noise* pada pembangkitan kunci publik dan kunci privat.

Operasi *encoding* (merujuk pada persamaan IV-1) dan operasi *rescaling* (merujuk pada persamaan IV-2) memerlukan pembulatan nilai koefisien dengan menggunakan *coordinate-wise random rounding*. Pembulatan tersebut akan mengakibatkan penambahan *rounding error* secara acak, sehingga berkontribusi dalam penambahan galat pada perhitungan skema CKKS.

Skema enkripsi CKKS juga dapat menambahkan keberadaan galat secara disengaja, seperti pada penambahan *Gaussian noise* ketika membangkitkan kunci publik dan privat (merujuk pada persamaan IV-3). Hal tersebut ditujukan untuk menambah tingkat keamanan dari skema enkripsi CKKS dengan menambahkan nilai ketidakpastian pada proses enkripsi dan dekripsi.

V. KESIMPULAN DAN SARAN

Penelitian ini membuktikan bahwa skema enkripsi CKKS dapat diintegrasikan ke dalam arsitektur CNN dengan memanfaatkan metode matriks Toeplitz untuk *convolutional layer*, metode perkalian diagonal untuk *fully connected layer*, serta aproksimasi polinomial untuk fungsi aktivasi. Hasil pengujian menunjukkan bahwa model CNN terenkripsi memiliki kinerja yang mendekati model *plaintext*, dengan perbedaan yang sangat kecil pada nilai *loss* dan akurasi. Meskipun terdapat galat akibat sifat aproksimasi dan operasi homomorfik CKKS, pengaruhnya terhadap hasil klasifikasi relatif minimal, sehingga pendekatan ini layak digunakan untuk menjaga privasi data dalam pembelajaran mesin.

Untuk pengembangan lebih lanjut, disarankan memanfaatkan kakas CKKS yang mendukung akselerasi GPU agar proses komputasi lebih efisien, serta mengganti metode matriks Toeplitz dengan operasi rotasi untuk mengurangi galat. Selain itu, pengembangan dapat mencakup implementasi *layer* tambahan seperti *max pooling*, *ReLU*, dan *softmax*, serta melakukan pelatihan model CNN dengan parameter dan data yang sama-sama terenkripsi. Pendekatan ini diharapkan meningkatkan keamanan sekaligus mempertahankan performa model dalam skenario nyata.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi, M.T. yang telah membimbing penulis sehingga penulis dapat menyelesaikan penelitian ini. Penulis juga mengucapkan terima kasih kepada semua teman dan keluarga yang telah mendukung penulis dengan waktu dan tenaga sehingga penulis dapat menyelesaikan penelitian ini.

REFERENSI

[1] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, "TENSEAL: a library for encrypted tensor operations using homomorphic encryption," arXiv (Cornell University), Apr. 2021, doi: 10.48550/arxiv.2104.03152.

[2] A. Falcetta and M. Roveri, "Privacy-Preserving Deep Learning with Homomorphic Encryption: An Introduction," IEEE Computational Intelligence Magazine, vol. 17, no. 3, pp. 14–25, Jul. 2022, doi: 10.1109/mci.2022.3180883.

[3] A. Kim, A. Papadimitriou, and Y. Polyakov, "Approximate Homomorphic Encryption with Reduced Approximation Error," in *Lecture Notes in Computer Science*, 2022, pp. 120–144. doi: 10.1007/978-3-030-95312-6_6.

[4] A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon, "Logistic regression model training based on the approximate homomorphic encryption," BMC Medical Genomics, vol. 11, no. S4, Oct. 2018, doi: 10.1186/s12920-018-0401-7.

[5] D. Huynh, "CKKS explained: Part 1, Vanilla Encoding and Decoding," *OpenMined Blog*, Mar. 15, 2021. https://blog.openmined.org/ckks-explained-part-1-simple-encoding-and-decoding/

[6] D. Huynh, "CKKS explained, Part 2: Full Encoding and Decoding," *OpenMined Blog*, Nov. 25, 2020. https://blog.openmined.org/ckks-explained-part-2-ckks-encoding-and-decoding/

[7] D. Huynh, “CKKS explained, Part 3: Encryption and Decryption,” OpenMined Blog, Nov. 25, 2020. https://blog.openmined.org/ckks-explained-part-3-encryption-and-decryption/

[8] D. Huynh, “CKKS explained, Part 4: Multiplication and Relinearization,” *OpenMined Blog*, Nov. 25, 2020. https://blog.openmined.org/ckks-explained-part-4-multiplication-and-relinearization/

[9] D. Huynh, “CKKS explained, Part 5: Rescaling,” OpenMined Blog, Dec. 01, 2020. https://blog.openmined.org/ckks-explained-part-5-rescaling/

[10] D. P. Huynh, “Cryptotree: fast and accurate predictions on encrypted structured data.,” arXiv (Cornell University), Jun. 2020, [Online]. Available: https://arxiv.org/pdf/2006.08299.pdf

[11] D. Unzueta, “Convolutional Layers vs Fully Connected Layers,” Towards Data Science, Nov. 13, 2021. https://towardsdatascience.com/convolutional-layers-vs-fully-connected-layers-364f05ab460b/

[12] E. Diao, J. Ding, and V. Tarokh, “HETEROFL: Computation and Communication Efficient federated learning for heterogeneous clients,” arXiv (Cornell University), Oct. 2020, doi: 10.48550/arxiv.2010.01264.

[13] E. Lee, D. K. Lee, Y. S. Kim, and J.-S. No, “Optimization of homomorphic comparison Algorithm on RNS-CKKS Scheme,” IEEE Access, vol. 10, pp. 26163–26176, Jan. 2022, doi: 10.1109/access.2022.3155882.

[14] H. Chen et al., “Logistic regression over encrypted data from fully homomorphic encryption,” BMC Medical Genomics, vol. 11, no. S4, Oct. 2018, doi: 10.1186/s12920-018-0397-z.

[15] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, “Faster fully homomorphic encryption: bootstrapping in less than 0.1 seconds,” in Lecture notes in computer science, 2016, pp. 3–33. doi: 10.1007/978-3-662-53887-6_1.

[16] J. H. Cheon, A. Kim, M. Kim, and Y. Song, “Homomorphic encryption for arithmetic of approximate numbers,” in Lecture Notes in Computer Science, 2017, pp. 409–437. doi: 10.1007/978-3-319-70694-8_15.

[17] J. H. Cheon, D. Kim, and D. Kim, “Efficient Homomorphic Comparison Methods with Optimal Complexity,” in Lecture Notes in Computer Science, 2020, pp. 221–256. doi: 10.1007/978-3-030-64834-3_8.

[18] J. H. Cheon, K.-S. Han, A. Kim, M. Kim, and Y. Song, “A full RNS variant of approximate homomorphic encryption,” in Lecture Notes in Computer Science, 2019, pp. 347–368. doi: 10.1007/978-3-030-10970-7_16.

[19] J. M. Chang, D. Zhuang, and G. D. Samaraweera, Privacy-Preserving machine learning. Simon and Schuster, 2023.

[20] J. Patriquin, “Bridging Microsoft SEAL into TensorFlow - Cape Privacy (Formerly Dropout Labs) - Medium,” Medium, Dec. 11, 2021. [Online]. Available: https://medium.com/dropoutlabs/bridging-microsoft-seal-into-tensorflow-b04cc2761ad4

[21] K. Muhammad, J. Ahmad, I. Mehmood, S. Rho, and S. W. Baik, “Convolutional neural networks based fire detection in surveillance videos,” IEEE Access, vol. 6, pp. 18174–18183, Jan. 2018, doi: 10.1109/access.2018.2812835.

[22] K.-S. Han, S. Hong, J. H. Cheon, and D. Park, “Efficient logistic regression on large encrypted data.,” IACR Cryptology ePrint Archive, vol. 2018, p. 662, Jan. 2018, [Online]. Available: https://eprint.iacr.org/2018/662.pdf

[23] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, “A review of applications in federated learning,” Computers & Industrial Engineering, vol. 149, p. 106854, Sep. 2020, doi: 10.1016/j.cie.2020.106854.

[24] M. Kim, Y. Song, S. Wang, Y. Xia, and X. Jiang, “Secure logistic regression based on homomorphic encryption: design and evaluation,” JMIR Medical Informatics, vol. 6, no. 2, p. e19, Apr. 2018, doi: 10.2196/medinform.8805.

[25] M. Marwan, A. Kartit, and H. Ouahmane, “Applying homomorphic encryption for securing cloud database,” IEEE Xplore, vol. 23, pp. 658–664, Oct. 2016, doi: 10.1109/cist.2016.7804968.

[26] Optalysys, “FHE and machine learning: A student perspective with examples,” Medium, Nov. 18, 2022. [Online]. Available: https://medium.com/optalysys/fhe-and-machine-learning-a-student-perspective-with-examples-88d70664a6cb

[27] P. Kairouz et al., Advances and open problems in federated learning, vol. 14, no. 1–2. 2021, pp. 1–210. doi: 10.1561/9781680837896.

[28] T. Wolf et al., “Transformers: State-of-the-Art Natural Language Processing,” ACL Anthology, pp. 38–45, Jan. 2020, doi: 10.18653/v1/2020.emnlp-demos.6.

[29] V. Lyubashevsky, C. Peikert, and O. Regev, “A toolkit for Ring-LWE cryptography,” in Lecture Notes in Computer Science, 2013, pp. 35–54. doi: 10.1007/978-3-642-38348-9_3.

[30] V. Lyubashevsky, C. Peikert, and O. Regev, “On Ideal Lattices and Learning with Errors over Rings,” in Lecture notes in computer science, 2010, pp. 1–23. doi: 10.1007/978-3-642-13190-5_1.

[31] W. Lu, Z. Huang, H. Cheng, Y. Ma, and H. Qu, “PEGASUS: Bridging Polynomial and Non-polynomial Evaluations in Homomorphic Encryption,” Cryptology ePrint Archive, May 2021, doi: 10.1109/sp40001.2021.00043.

[32] X. Yi et al., Privacy enhancing techniques: Practices and Applications. Springer Nature, 2025.

[33] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[34] Z. Brakerski, “Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP,” in Lecture notes in computer science, 2012, pp. 868–886. doi: 10.1007/978-3-642-32009-5_50.

[35] Z. Shaukat, S. Ali, Q. U. A. Farooq, C. Xiao, S. Sahiba, and A. Ditta, “Cloud-based efficient scheme for handwritten digit recognition,” Multimedia Tools and Applications, vol. 79, no. 39–40, pp. 29537–29549, Aug. 2020, doi: 10.1007/s11042-020-09494-1.