ISSN (Online) 2747-0563
Volume 5 No. 2 Tahun 2025

Deteksi URL Phishing Menggunakan Hybrid Deep Learning CNN dan XGBoost dengan Teknik Balancing SMOTE-ENN

Paskalis Reynaldy Elroy Gabriel

Informatika, Universitas Pembangunan Nasional "Veteran" Jawa Timur

22081010197@student.upnjatim.ac.id

***Abstrak*— Serangan phishing melalui URL palsu merupakan ancaman keamanan siber yang terus berkembang dan merugikan pengguna internet. Penelitian ini mengusulkan sistem deteksi *phishing* menggunakan pendekatan yang menggabungkan Convolutional Neural Network (CNN) untuk ekstraksi fitur otomatis dan XGBoost sebagai *classifier*. CNN dilatih secara *supervised learning* untuk mengekstrak 64 fitur *high-level* dari karakter URL, kemudian digabungkan dengan 40 fitur heuristik yang mencakup panjang URL, jumlah karakter khusus, entropi, dan *suspicious keywords*. Untuk mengatasi ketidakseimbangan *dataset* (rasio 1:1,81), diterapkan teknik SMOTE-ENN yang mengombinasikan o*versampling* dan *cleaning* data. Model dilatih menggunakan dataset 4.856 URL dengan rasio 80:20 untuk *training* dan *testing*. Hasil eksperimen menunjukkan performa yang sangat tinggi dengan akurasi 97.12%, *precision* 0.9855 untuk kelas *phishing*, *recall* 0.9644, F1-*score* 0.9748 untuk kelas *phishing*, dan AUC-ROC 0.9960. *Hyperparameter tuning* menggunakan GridSearchCV menghasilkan *cross-validation* F1-*score* 0.9991. Model mampu melakukan inferensi dengan kecepatan 4.855 URL/detik, menjadikannya cocok untuk implementasi *real-time*.**

Kata Kunci— **phishing detection, convolutional neural network, xgboost, smote-enn, hybrid deep learning, url security, imbalanced data**

I. PENDAHULUAN

Phishing merupakan salah satu ancaman keamanan siber paling serius di era digital. Serangan *phishing* menggunakan URL palsu yang menyerupai situs *legitimate* untuk mencuri kredensial pengguna, informasi keuangan, dan data sensitif lainnya [1]. Menurut Anti-Phishing Working Group (APWG), jumlah serangan *phishing* meningkat 150% pada tahun 2023, dengan kerugian finansial mencapai miliaran dolar [2].

Metode deteksi *phishing* tradisional seperti *blacklist-based approach* memiliki keterbatasan karena tidak dapat mendeteksi URL *phishing* baru (*zero-day attacks*) [3]. Pendekatan berbasis *machine learning* menawarkan solusi yang lebih adaptif dengan kemampuan mengenali pola *phishing* dari karakteristik URL [6], [17]. Namun, metode konvensional seperti *Random Forest* dan SVM bergantung pada *feature engineering* manual yang memerlukan *domain expertise* dan tidak optimal dalam menangkap pola kompleks dari struktur URL [7], [13].

Deep learning, khususnya *Convolutional Neural Network* (CNN), telah menunjukkan keunggulan dalam ekstraksi fitur otomatis dari data sekuensial seperti teks [4], [10]. CNN dapat mempelajari representasi hierarkis dari karakter URL tanpa memerlukan *feature engineering* manual, mampu menangkap pola lokal seperti kombinasi karakter mencurigakan, struktur domain yang tidak biasa, dan sekuens *path* yang kompleks [11], [16]. Namun, CNN murni memiliki keterbatasan dalam hal *interpretability* dan tidak memanfaatkan *domain knowledge* yang sudah ada tentang karakteristik phishing URL.

Di sisi lain, XGBoost merupakan algoritma *gradient boosting* yang sangat *powerful* untuk klasifikasi tabular data [15]. XGBoost memiliki keunggulan dalam menangani fitur numerik dengan berbagai skala, *robust* terhadap *outliers*, dan memberikan *feature importance* yang membantu *interpretability*. XGBoost juga sangat efisien dalam *training* dan *inference*, serta memiliki *built-in regularization* untuk mencegah *overfitting*.

Penelitian ini menggabungkan CNN dan XGBoost dalam arsitektur hybrid untuk memanfaatkan kekuatan masing-masing algoritma secara sinergis:

1) CNN sebagai Feature Extractor: CNN digunakan untuk otomatis mengekstrak high-level features dari sekuens karakter URL. Layer konvolusi dapat mendeteksi pola lokal seperti kombinasi karakter yang mencurigakan (contoh: "secure-login", "verify-account"), struktur subdomain yang kompleks, dan anomali dalam path URL. Dengan supervised training, CNN belajar representasi yang paling diskriminatif untuk membedakan phishing dan legitimate URLs.
2) Fitur Heuristik untuk Domain Knowledge: Meskipun CNN powerful, fitur heuristik tetap penting karena mengkodifikasi pengetahuan domain yang sudah terbukti efektif, seperti panjang URL, penggunaan HTTPS, jumlah subdomain, entropy, dan keberadaan keywords mencurigakan. Fitur-fitur ini bersifat interpretable dan memberikan konteks yang tidak selalu tertangkap oleh CNN.
3) XGBoost untuk Klasifikasi Final: XGBoost dipilih sebagai classifier akhir karena sangat efektif dalam menangani kombinasi fitur heterogen (64 fitur CNN + 40 fitur heuristik). XGBoost dapat mempelajari interaksi kompleks

ISSN (Online) 2747-0563
Volume 5 No. 2 Tahun 2025

antar fitur, menangani non-linearity, dan memberikan decision boundary yang optimal. Selain itu, XGBoost memberikan feature importance yang membantu memahami kontribusi relatif dari CNN features vs heuristic features.

4) Pemisahan Concern: Dengan memisahkan feature extraction (CNN) dan classification (XGBoost), sistem menjadi lebih modular. CNN fokus pada pattern recognition dari raw URL, sementara XGBoost fokus pada decision making berdasarkan kombinasi fitur.

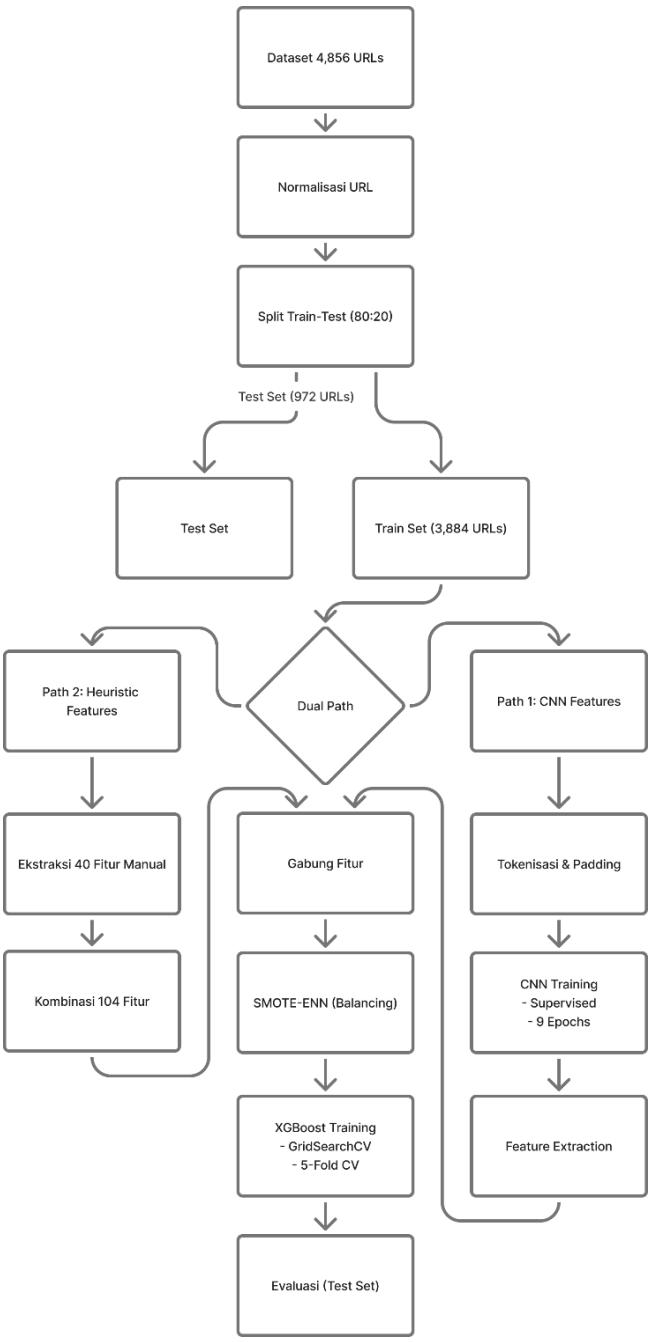
Tantangan utama dalam deteksi *phishing* adalah ketidakseimbangan *dataset*, dimana jumlah URL *legitimate* biasanya lebih sedikit dari URL *phishing* dalam *dataset* penelitian. Ketidakseimbangan ini dapat menyebabkan model bias terhadap *majority class*. Untuk mengatasi hal ini, diterapkan SMOTE-ENN (*Synthetic Minority Over-sampling Technique with Edited Nearest Neighbours*) yang tidak hanya melakukan *oversampling* pada *minority class*, tetapi juga membersihkan *noise* dan *overlapping samples* untuk meningkatkan kualitas *decision boundary*.

Kontribusi utama penelitian ini meliputi: (1) Arsitektur *hybrid* CNN-XGBoost dengan *supervised* CNN *training* untuk *phishing detection*, (2) Kombinasi 64 fitur CNN dengan 40 fitur heuristik untuk representasi komprehensif, (3) Implementasi SMOTE-ENN untuk *handling imbalanced data* secara efektif, (4) *Hyperparameter tuning* menggunakan GridSearchCV untuk optimasi performa, (5) Evaluasi komprehensif dengan *multiple metrics* dan *error analysis*, (6) Sistem yang siap *deploy* dengan *inference speed* 4.855 URLs/detik.

II. METODOLOGI

Penelitian ini mengusung metodologi inovatif yang memadukan kekuatan *deep learning* dan *ensemble learning* untuk mendeteksi *phishing*. Metodologi akan diuraikan secara runut, diawali dengan deskripsi *Dataset* yang digunakan, tahapan *Preprocessing* dan Tokenisasi URL, hingga inti dari pendekatan ini: mekanisme *Feature Extraction* hibrida yang menghasilkan total 104 fitur (64 fitur CNN-based dan 40 Heuristic Features). Selanjutnya, dijelaskan proses Data *Balancing* dengan SMOTE-ENN dan optimasi *hyperparameter* XGBoost *Classifier* untuk mencapai performa deteksi yang optimal.

Dataset berisi 4.856 URL dinormalisasi lalu dibagi menjadi data latih dan data uji (80:20). Data latih diproses melalui dua jalur: ekstraksi fitur berbasis CNN melalui tokenisasi dan pelatihan 9 epoch, serta ekstraksi 40 fitur heuristik secara manual. Hasil kedua jalur digabungkan menjadi 104 fitur, kemudian dilakukan penyeimbangan data dengan SMOTE-ENN sebelum dilatih menggunakan XGBoost dengan GridSearchCV dan validasi silang 5-fold. Model yang diperoleh akhirnya dievaluasi menggunakan data uji.



Gambar. 1 Diagram Alur Metodologi

A. *Dataset*

Dataset yang digunakan terdiri dari 4.856 data berbentuk URL yang dikumpulkan dari berbagai sumber termasuk PhishTank (repository URL phishing terverifikasi), OpenPhish (platform phishing feed real-time), dan Alexa Top Sites (situs legitimate populer), kemudian dijadikan file berbentuk CSV. Komposisi dataset adalah 1.726 URL legitiImate (35,5%) dan 3.130 URL phishing (64,5%). *Dataset* dibagi menjadi 80% *training* (3.884 samples) dan 20% *testing* (972 *samples*) menggunakan *stratified sampling* untuk mempertahankan distribusi kelas. Dataset ini berbentuk file CSV yang terdiri dari dua kolom, yaitu url yang berisi string alamat URL lengkap, dan label yang merupakan nilai biner untuk klasifikasi, di mana 0

ISSN (Online) 2747-0563
Volume 5 No. 2 Tahun 2025

menunjukkan URL legitimate dan 1 menunjukkan URL phishing.

http://tokopedia-secure-login.org	1
http://shopee-update-akun.net	1
http://secure-ovo-login-alert.com	1
http://google.com	0
http://wikipedia.org	0
http://github.com	0

Gambar. 2 Contoh isi *dataset*

B. Preprocessing

Kualitas *data input* merupakan faktor krusial dalam keberhasilan model deteksi, terutama untuk data berstruktur sekuensial seperti URL. Oleh karena itu, tahapan *Preprocessing* ini dirancang untuk mengubah *raw* URL menjadi representasi numerik yang seragam dan optimal bagi arsitektur *deep learning* yang diusulkan. Proses ini dimulai dengan Normalisasi URL untuk memastikan konsistensi format, dilanjutkan dengan Tokenisasi karakter level untuk mempersiapkan sekuens input yang berukuran tetap untuk *Convolutional Neural Network* (CNN).

1) Normalisasi URL

Setiap URL dalam dataset terlebih dahulu dinormalisasi untuk memastikan konsistensi format *input*. Proses ini mencakup penghapusan *whitespace*, konversi seluruh karakter menjadi huruf kecil, serta *parsing* URL untuk mengekstraksi *hostname* dan *path*. Hanya bagian *hostname* dan *path* yang dipertahankan, sementara komponen lain seperti *scheme* dan *query* dihilangkan agar fokus tetap pada bagian URL yang paling informatif. Sebagai contoh, URL "HTTPS://Secure-PayPal.COM/Verify?id=123 " dinormalisasi menjadi "secure-paypal.com/verify",dan "http://192.168.1.1:8080/admin/login" menjadi "192.168.1.1:8080/admin/login". Langkah normalisasi ini bertujuan menghilangkan variasi penulisan yang tidak relevan, menjaga konsistensi *input* selama proses pelatihan model, dan mencegah duplikasi representasi pada URL yang sama.

2) *Label Encoding*

Label pada dataset dikonversi dari bentuk kategorikal menjadi numerik untuk memudahkan proses pelatihan model. Nilai *"legitimate"* diubah menjadi 0 dan *"phishing"* diubah menjadi 1 menggunakan *LabelEncoder* dari *scikit-learn*, sehingga encoding bersifat konsisten dan dapat diproses langsung oleh algoritma machine learning.

C. Ekstraksi Fitur

Untuk memanfaatkan kekuatan representasi *deep learning* dan interpretasi domain *knowledge*, tahap Feature Extraction mengadopsi pendekatan hibrida. Proses ini secara sinergis menggabungkan fitur CNN-based yang diekstrak secara otomatis melalui arsitektur yang dilatih secara *supervised*, dengan 40 *Heuristic Features* yang diinterpretasikan berdasarkan karakteristik URL *phishing* yang diketahui. Integrasi dari kedua jenis fitur ini menghasilkan total 104 fitur *high-level* yang komprehensif untuk dimasukkan ke dalam *classifier* XGBoost.

1) *CNN-based Features*

Tahap ekstraksi fitur berbasis CNN dimulai dengan mengubah URL yang telah dinormalisasi menjadi sekuens numerik melalui tokenisasi pada tingkat karakter. Tokenizer diatur untuk bekerja pada karakter sehingga setiap karakter dalam URL dikonversi menjadi integer berdasarkan frekuensi kemunculannya dalam data pelatihan. Setelah URL menjadi sekuens angka, model CNN dibangun menggunakan arsitektur yang terdiri dari *Embedding Layer* untuk mengubah token menjadi vektor representasi, diikuti *Conv1D* dengan 128 filter dan kernel size 5 untuk menangkap pola lokal pada struktur URL. *GlobalMaxPooling1D* kemudian digunakan untuk mengekstrak fitur utama dari hasil konvolusi, dilanjutkan *Dense Layer* berukuran 64 unit dan *Dropout* untuk mencegah overfitting. Lapisan keluaran menggunakan fungsi aktivasi sigmoid untuk menghasilkan nilai prediksi. Hasil dari model ini berupa vektor fitur yang mewakili karakteristik struktur URL yang digunakan pada tahap penggabungan fitur selanjutnya.CNN dilatih secara *supervised* dengan *binary crossentropy loss* dan Adam optimizer selama 10 *epochs* dengan *early stopping* (*patience*=3). Model terbaik disimpan berdasarkan *validation accuracy*. *Feature extractor* dibuat dengan mengambil *output* dari Dense(64) *layer*, menghasilkan 64 fitur *high-level* untuk setiap URL.

2) *Padding Sequences*

Setiap sekuens URL kemudian diseragamkan panjangnya agar dapat diproses oleh CNN. Panjang maksimum ditetapkan sebesar 200 karakter, dengan metode *padding* di bagian akhir apabila URL lebih pendek dan *truncating* di bagian akhir apabila URL melebihi panjang tersebut. Dengan demikian, URL yang hanya memiliki sekitar 50 karakter akan dilengkapi nol hingga mencapai panjang 200, sementara URL yang lebih panjang dari 200 karakter akan dipotong setelah posisi ke-200. Hasilnya adalah matriks *input* berukuran (4856, 200) yang siap digunakan sebagai masukan model CNN.

3) Arsitektur CNN untuk *Supervised Learning*

Arsitektur CNN pada tahap *supervised learning* menerima input berupa sekuens hasil padding dengan panjang 200 token. *Sequences* ini terlebih dahulu diubah menjadi representasi vektor melalui *Embedding Layer* dengan dimensi 50, sehingga setiap karakter memiliki representasi numerik yang lebih bermakna. Selanjutnya, Conv1D dengan 128 filter dan *kernel* ukuran 5 digunakan untuk menangkap pola lokal atau *n-grams* pada struktur URL. Hasil konvolusi kemudian diringkas menggunakan *GlobalMaxPooling1D* untuk mengambil fitur paling representatif dari setiap filter. Fitur tersebut diproses lebih lanjut melalui *Dense Layer* dengan 64 unit sebagai lapisan inti yang menghasilkan representasi fitur tingkat tinggi, yang kemudian diberi *Dropout* untuk mengurangi *overfitting*. Terakhir, *Output Layer* dengan satu neuron dan aktivasi

ISSN (Online) 2747-0563
Volume 5 No. 2 Tahun 2025

sigmoid menghasilkan probabilitas yang menunjukkan apakah suatu URL termasuk *phishing* atau *legitimate*.

4) *Training CNN*

Tahap pelatihan CNN dilakukan menggunakan *Binary Crossentropy* sebagai fungsi *loss* dan Adam sebagai *optimizer* dengan *batch size* 32 dan maksimum 10 *epoch*, serta 10% data latih digunakan sebagai data validasi. Selama pelatihan, *EarlyStopping* memantau *validation loss* dan menghentikan pelatihan apabila tidak terjadi peningkatan selama 3 *epoch*, sementara *ModelCheckpoint* menyimpan model dengan *validation accuracy* terbaik. Hasil pelatihan menunjukkan peningkatan akurasi secara bertahap hingga mencapai titik terbaik pada *epoch* ke-6 dengan *train_acc* sebesar 0.98 dan *val_acc* sebesar 0.96, setelah itu proses dihentikan dan bobot terbaik dipulihkan.

5) *Feature Extraction*

Setelah model CNN selesai dilatih, dilakukan tahap ekstraksi fitur dengan mengambil keluaran dari *layer dense_features*, yaitu lapisan *Dense* berukuran 64 unit yang mewakili representasi tingkat tinggi dari pola URL. Model baru dibentuk dengan input yang sama seperti CNN, namun *output* diarahkan ke *layer* tersebut sehingga menghasilkan *feature extractor*. Ketika *padded sequences* (berukuran 4856 × 200) dimasukkan ke *extractor* ini, hasilnya adalah matriks fitur berukuran 4856 × 64. Fitur-fitur ini mencerminkan pola karakter dan struktur URL yang dipelajari CNN, seperti susunan *subdomain* yang tidak lazim, pola penulisan path yang menyerupai situs *phishing*, atau kemiripan dengan pola URL *phishing* yang pernah dilihat model. Meskipun fitur ini bersifat abstrak dan tidak dapat diinterpretasikan secara langsung, representasi tersebut terbukti efektif dan diskriminatif untuk keperluan klasifikasi pada tahap pemodelan berikutnya.

6) *Heuristic Features*

Fitur *heuristic* diekstraksi mencakup:

1. *Length-based features* (5): URL *length*, *domain length*, *path length*, *query length*, *fragment length*
2. *Character count features* (12): Jumlah *dots*, *hyphens*, @, ?, =, &, %, *underscores*, *slashes*, *double slashes*, *digits*, *uppercase*
3. *Domain-based features* (5): *Subdomain count*, *hyphens in domain*, *starts with* www, @ *in domain*, IP *address detection*
4. *Security features* (3): HTTPS *presence*, *port specification*, *multiple* HTTP
5. *Query features* (2): *Parameter count*, *query parameters*
6. *Suspicious patterns* (1): *Keywords* seperti *'login'*, *'verify'*, *'secure'*, *'banking'*
7. *Length flags* (3): *Long domain* (>50), *long* URL (>100), *long path* (>50)
8. *Statistical features* (4): URL *entropy*, *domain*/URL *ratio*, *path*/URL *ratio*, *digit ratio*
9. TLD *features* (1): *Suspicious* TLDs (tk, ml, ga, cf, gq, xyz, *top*, *click*)
10. *Additional features* (4): *Long domain with hyphen*, *multiple* www, *all-digit domain*, *special character count*
11. Total 104 fitur (64 CNN + 40 heuristik) digunakan untuk *training* XGBoost.

7) *Feature Combination*

Fitur yang dihasilkan oleh CNN (berukuran 64) kemudian digabungkan dengan 40 fitur heuristik melalui proses *concatenation* secara horizontal menggunakan np.hstack(). Proses ini menghasilkan representasi gabungan dengan dimensi 104 fitur untuk setiap URL. Kombinasi tersebut memungkinkan model memanfaatkan keunggulan pembelajaran fitur otomatis dari CNN yang bersifat abstrak, bersama dengan pengetahuan berbasis aturan dari fitur heuristik yang lebih mudah diinterpretasikan, sehingga menghasilkan representasi yang lebih kaya dan informatif untuk tahap klasifikasi berikutnya.

D. Data Balancing

Setelah tahap ekstraksi fitur dan pemisahan data menjadi *train-test*, diketahui bahwa data pada *set training* tidak seimbang. Jumlah sampel *phishing* lebih banyak dibandingkan dengan sampel *legitimate*. Pada kondisi awal, *training set* terdiri atas 1.381 URL *legitimate* (35,6%) dan 2.503 URL *phishing* (64,4%), dengan total 3.884 data dan rasio ketidakseimbangan sekitar 1 : 1,81. Untuk mengatasi hal ini, diterapkan teknik *SMOTE-ENN* hanya pada training set. Pendekatan ini menggabungkan *oversampling* dengan SMOTE untuk menambah sampel *phishing* sintetis, serta *cleaning* dengan *Edited Nearest Neighbors* (ENN) untuk menghapus sampel yang ambigu. Hasilnya adalah distribusi kelas yang lebih seimbang, sehingga model yang dilatih tidak bias terhadap kelas mayoritas dan dapat mencapai performa klasifikasi yang lebih stabil. SMOTE-ENN diterapkan pada *training set* untuk mengatasi *imbalance*:

1) *SMOTE Phase*

SMOTE diterapkan untuk menyeimbangkan distribusi kelas dengan menambahkan data sintetis pada *minority class* (*legitimate*). Proses SMOTE bekerja dengan mencari *k-nearest neighbors* (k=5) untuk setiap sampel pada kelas minoritas, kemudian melakukan interpolasi antara sampel asli x_i dan salah satu tetangganya x_nn , menghasilkan sampel baru menggunakan rumus:

$$x\_new = x\_i + \lambda(x\_nn - x\_i), \lambda \sim Uniform(0,1)$$

Proses ini diulang hingga jumlah sampel *minority class* mencapai target keseimbangan. Pada data awal, terdapat 1.381 sampel *legitimate* dan 2.503 sampel *phishing*. Untuk mencapai keseimbangan, diperlukan penambahan sekitar 1.122 sampel legitimate. SMOTE menghasilkan sekitar 1.640 sampel sintetis, sehingga setelah proses *oversampling*, jumlah data menjadi:

1. *Legitimate*: 3.021 (1.381 asli + 1.640 sintetis)
2. *Phishing*: 2.503

ISSN (Online) 2747-0563
Volume 5 No. 2 Tahun 2025

3. Total: 5.524 sampel

Dengan demikian, SMOTE berhasil mengatasi ketidakseimbangan kelas dengan membuat sampel sintetis berdasarkan kemiripan lokal antar data *legitimate*.

2) *ENN Phase*

Setelah proses *oversampling* dengan SMOTE, dilakukan tahap pembersihan data menggunakan ENN. *Edited Nearest Neighbours* bekerja dengan memeriksa setiap sampel x_idalam *dataset*, kemudian mencari *k-nearest neighbors* (k=3). Jika kelas dari x_iberbeda dengan mayoritas kelas pada tetangga terdekatnya, maka sampel tersebut dianggap sebagai *noise* atau berada pada *decision boundary* yang ambigu, sehingga sampel tersebut dihapus. Tujuan utama ENN adalah menghilangkan data yang salah label, mengurangi *overlapping samples* antar kelas, serta membersihkan sampel sintetis yang kurang representatif hasil dari SMOTE. Dengan demikian, ENN meningkatkan class *separability* dan menghasilkan *dataset* yang lebih bersih dan stabil untuk pelatihan model.

3) *Setelah SMOTE-ENN*

Setelah SMOTE meningkatkan jumlah sampel *legitimate* menjadi 3.021 dan *phishing* tetap 2.503, ENN diterapkan untuk menghapus sampel yang dianggap *noise* atau berada pada *decision boundary*. Dari total hasil SMOTE (5.524 sampel), ENN menghapus 488 sampel (8,8%), terdiri dari 252 sampel *legitimate* (termasuk yang sintetis dan tidak representatif) serta 236 sampel *phishing* yang berada pada area kelas yang tumpang tindih. Hasil akhirnya menghasilkan distribusi yang seimbang: 3.021 *legitimate* (50,1%) dan 3.015 *phishing* (49,9%), dengan total 6.036 sampel. Proses ini menggabungkan keunggulan SMOTE dalam menambah sampel *minority class* tanpa duplikasi, dan ENN dalam membersihkan *noise*, memperjelas *decision boundary*, serta mengurangi risiko *overfitting*.

E. XGBOOST Classifier

Setelah tahap ekstraksi fitur hibrida dan penyeimbangan data menggunakan SMOTE-ENN, XGBoost (*eXtreme Gradient Boosting*) dipilih sebagai *classifier* akhir karena efisiensinya dan performa unggulnya dalam tugas klasifikasi biner. Untuk memastikan kinerja optimal pada data *phishing* yang kompleks, sub-bab ini merinci proses *Hyperparameter Tuning* yang sistematis menggunakan GridSearchCV dan validasi silang 3-lipatan untuk mengidentifikasi *Best Configuration parameter*, dengan fokus memaksimalkan F1-*score*.

1) *Hyperparameter Tuning*

GridSearchCV digunakan untuk mencari kombinasi optimal:

1. *max_depth*: [4, 6, 8]
2. *learning_rate*: [0.05, 0.1, 0.2]
3. *n_estimators*: [100, 200]
4. *subsample*: [0.8, 1.0]
5. *colsample_bytree*: [0.8, 1.0]

3-fold *cross-validation* dengan F1-*score* sebagai *metric*. Total 72 kombinasi parameter diuji.

2) *Best Configuration*

Parameter terbaik yang ditemukan:

1. *max_depth*: 6
2. *6learning_rate*: 0.1
3. *n_estimators*: 200
4. *subsample*: 0.8
5. *colsample_bytree*: 0.8
6. *objective*: *binary*:*logistic*
7. *eval_metric*: *logloss*

3) *XGBoost Training Process*

XGBoost (*Extreme Gradient Boosting*) adalah metode *ensemble* yang membangun banyak *decision tree* secara berurutan, di mana setiap *tree* berfungsi untuk memperbaiki *error* atau *residual* dari *tree* sebelumnya. Proses *training* dilakukan dalam 200 iterasi (*boosting rounds*), dimulai dari prediksi awal sebesar 0,5 sebagai *baseline*. Pada setiap *round*, residual *error* dihitung dari prediksi saat ini, kemudian dibuat *tree* baru untuk memprediksi residual tersebut. Kontribusi *tree* baru ditambahkan ke prediksi sebelumnya dengan menggunakan *learning rate*, sehingga pada *round* terakhir, prediksi akhir merupakan penjumlahan dari *output* semua *tree* yang kemudian diterapkan fungsi sigmoid untuk mengonversi hasil menjadi probabilitas. *Dataset* yang digunakan berupa *training set* yang seimbang dengan 6.036 sampel dan 104 fitur.

Pada setiap *boosting round*, XGBoost melakukan *sampling* 80% baris secara acak (*subsample*=0,8) dan 80% fitur secara acak (*colsample_bytree*=0,8) sebelum membangun *decision tree* dengan kedalaman maksimum 6 *level*. *Tree* dibangun berdasarkan kriteria *split* berbasis *gain* dan dilengkapi dengan regularisasi L1 dan L2 untuk menghindari *overfitting*. *Output tree* digunakan untuk memprediksi residual, yang kemudian diperbarui ke *ensemble* menurut persamaan $F\_m\ (x) = F\_(m-1)\ (x) + 0{,}1 \times tree\_m\ (x)$. Model final merupakan *ensemble* dari 200 *tree* yang saling melengkapi.

Fungsi objektif XGBoost terdiri dari *binary crossentropy* dan regularisasi. *Binary crossentropy* didefinisikan sebagai

$$L = -[y \cdot \log(p) + (1-y) \cdot \log(1-p)\ ],$$

dengan $p = "sigmoid"\ (F(x))$, sedangkan regularisasi

$$\Omega(f) = \gamma\gamma T + 1/2\ \lambda\ ||\ w\ ||\ \hat{}\ 2$$

mempertimbangkan jumlah daun Tdan bobot daun w. Setiap *tree* di-*fit* menggunakan pendekatan *gradient descent* dengan *second-order Taylor expansion*, di mana *loss* mendekati

$$\sum[g\_i \cdot f(x\_i\ ) + 1/2\ h\_i \cdot f\hat{}2\ (x\_i\ )] + \Omega(f),$$

dengan $g_i$sebagai turunan pertama *loss* terhadap prediksi dan $h_i$sebagai turunan kedua. Secara keseluruhan, mekanisme ini memungkinkan XGBoost menghasilkan prediksi yang akurat dan terkontrol, sekaligus mengurangi risiko *overfitting* melalui regularisasi.

ISSN (Online) 2747-0563
Volume 5 No. 2 Tahun 2025

4) *Prediction Process*

Proses prediksi untuk sampel baru dimulai dengan menerima vektor fitur x berisi 104 fitur. Pertama, skor awal diinisialisasi dengan nilai 0. Selanjutnya, prediksi dihitung secara bertahap melalui 200 *tree* dalam *ensemble*; setiap *tree* menghasilkan *output* yang dikalikan dengan *learning rate* 0,1, kemudian ditambahkan ke skor kumulatif. Setelah seluruh *tree* diproses, skor akhir diubah menjadi probabilitas menggunakan fungsi sigmoid:

$$"probability\_phishing" = 1/(1 + e\text{^}(-score)).$$

Berdasarkan probabilitas ini, sampel diklasifikasikan: jika probabilitas *phishing* lebih besar atau sama dengan 0,5, sampel dikategorikan sebagai *phishing* (1), sedangkan jika kurang dari 0,5, sampel dikategorikan sebagai *legitimate* (0). *Output* akhir berupa prediksi kelas sekaligus probabilitas masing-masing kelas, [$"prob\_legit"$, $"prob\_phishing"$].

F. *Evaluation Matrics*

Model dievaluasi menggunakan *multiple metrics*:

1. *Accuracy*: *Overall correctness*
2. *Precision*, *Recall*, F1-*Score*: Per-*class performance*
3. *Confusion Matrix: True positives, true negatives, false positives, false negatives*
4. AUC-ROC: *Area under* ROC *curve*
5. *Matthews Correlation Coefficient* (MCC): *Balanced metric* untuk *imbalanced data*
6. *Cohen's Kappa: Agreement measure*
7. *Cross-Validation F1: 5-fold CV* untuk *robustness*
8. *Inference Speed*: URLs *processed per second*

G. *Implementation*

Sistem diimplementasikan menggunakan:

1. Python 3.8
2. TensorFlow 2.x untuk CNN
3. XGBoost 1.7.x untuk *classifier*
4. Scikit-learn untuk *preprocessing* dan *evaluation*
5. *Imbalanced-learn* untuk SMOTE-ENN
6. NumPy, Pandas untuk *data manipulation*
7. Matplotlib, Seaborn untuk *visualization*

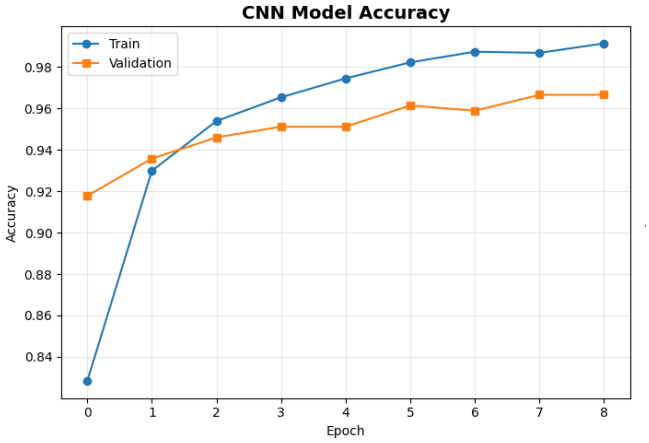
III. HASIL DAN PEMBAHASAN

Bab ini menyajikan dan mendiskusikan hasil eksperimen yang dilakukan sesuai dengan metodologi yang telah diuraikan. Pembahasan akan dimulai dengan analisis performa CNN *Training* dan efektivitas SMOTE-ENN dalam menyeimbangkan *dataset*, dilanjutkan dengan validasi *Hyperparameter Tuning* XGBoost. Puncak dari bab ini adalah evaluasi *Classification Performance* model hibrida CNN-XGBoost secara komprehensif menggunakan berbagai metrik utama, diikuti oleh *Error Analysis*, identifikasi *Feature Importance* untuk interpretasi model, serta perbandingan dengan metode lain untuk memvalidasi superioritas arsitektur yang diusulkan.

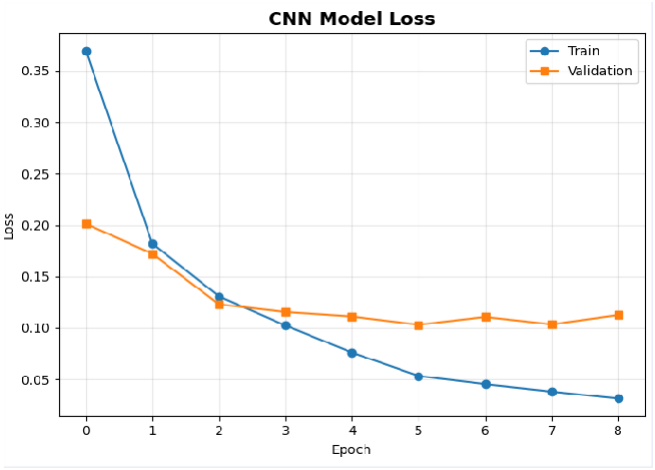
A. *CNN Training Performance*

CNN model dilatih selama 9 *epochs* sebelum *early stopping triggered*. *Training history* menunjukkan:

- *Training Accuracy*: Meningkat dari 83% (*epoch* 0) menjadi 99% (*epoch* 8)
- *Validation Accuracy*: Meningkat dari 93% menjadi 96,14%
- *Training Loss*: Turun dari 0,37 menjadi 0,04
- *Validation Loss*: Turun dari 0,20 menjadi 0,12



Gambar. 3 Riwayat Akurasi Pelatihan Model CNN**.** Grafik ini menunjukkan kurva akurasi pelatihan (*Train Accuracy*) dan akurasi validasi (*Validation Accuracy*) selama 9 *epoch*. *Validation Accuracy* model mencapai kestabilan di atas 96% setelah *epoch* ke-5, menunjukkan bahwa model berhasil mengekstrak fitur URL yang relevan tanpa mengalami *overfitting* yang signifikan.



Gambar. 4 Riwayat *Loss* Pelatihan Model CNN. Grafik ini menampilkan kurva *loss* pelatihan (*Train Loss*) dan *loss* validasi (*Validation Loss*) selama 9 *epoch*. Penurunan drastis pada kedua kurva mengindikasikan bahwa proses pelatihan berjalan efektif dan model berhasil meminimalkan kesalahan. *Validation Loss* stabil di sekitar 0.10 pada *epoch* akhir, menunjukkan *feature extractor* CNN telah mencapai konvergensi yang optimal.

Gap antara *training* dan *validation accuracy* stabil di ~3%, mengindikasikan tidak terjadi *overfitting* signifikan. *Validation loss* sempat naik di *epoch* 8-9 (0,10→0,14) namun masih dalam batas wajar. Model dengan *validation accuracy* tertinggi (*epoch* 6) dipilih sebagai *best model*.

ISSN (Online) 2747-0563
Volume 5 No. 2 Tahun 2025

B. SMOTE-ENN Balancing Results

Penerapan SMOTE-ENN pada *training set* menghasilkan:
Sebelum SMOTE-ENN:

- *Legitimate*: 1.381 (35,6%)
- *Phishing*: 2.503 (64,4%)
- Total: 3.884
- *Imbalance Ratio*: 1:1,81

Setelah SMOTE-ENN:

- *Legitimate*: 3.021 (50,1%)
- *Phishing*: 3.015 (49,9%)
- Total: 6.036
- *Imbalance Ratio*: 1:1,00

SMOTE menciptakan 2.174 - 1.637 = 537 *synthetic legitimate* URLs, sementara ENN menghapus total 460 *noisy samples*. *Cleaning rate* sebesar 11.8% dari total *original training set* menunjukkan *dataset* memiliki cukup banyak *overlapping samples* yang dihilangkan untuk meningkatkan *decision boundary*.

C. Hyperparameter Tuning Results

GridSearchCV dengan 72 kombinasi parameter menghasilkan:

1. *max_depth*=6, lr=0.1, n_*est*=200, *subsample*=0.8, *colsample*=0.8 → F1=0.9423
2. *max_depth*=6, lr=0.1, n_*est*=200, *subsample*=1.0, *colsample*=0.8 → F1=0.9401
3. *max_depth*=8, lr=0.05, n_*est*=200, *subsample*=0.8, *colsample*=1.0 → F1=0.9385

Best configuration dengan *subsample*=0.8 menunjukkan bahwa sedikit *randomness* dalam *row sampling* membantu mencegah *overfitting*.

D. Classification Performance

1) *Overall Metrics*:

- *Accuracy*: 0.9712 (97.12%)
- F1-*Score*: 0.9991
- MCC: 0.9414
- Kappa: 0.9411
- AUC-ROC: 0.9960

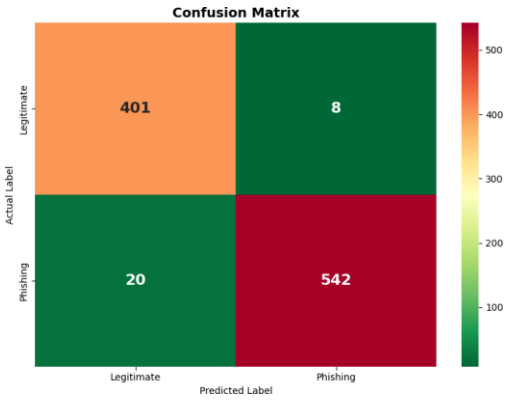
2) Per-Class Performance

	precision	recall	f1-score	support
Legitimate	0.9525	0.9804	0.9663	409
Phishing	0.9855	0.9644	0.9748	562
accuracy			0.9712	971
macro avg	0.9690	0.9724	0.9705	971
weighted avg	0.9716	0.9712	0.9712	971

Gambar. 5 Hasil Evaluasi Klasifikasi Model Hibrida CNN-XGBoost. Tabel ini menyajikan metrik kinerja utama (*Precision, Recall, F1-Score*) untuk setiap kelas (*Legitimate* dan *Phishing*) dan metrik agregat model (*Accuracy, Macro Avg, Weighted Avg*) pada *testing set* (Total 971 sampel). Hasil ini menunjukkan *F1-Score* rata-rata tertimbang sebesar 0.9712, mengonfirmasi performa klasifikasi yang sangat kuat dan seimbang.

Tabel ini menyajikan hasil evaluasi klasifikasi model hibrida pada *testing set* dengan akurasi keseluruhan mencapai 0.9712 (97.12%). Performa model sangat seimbang antara kedua kelas, dibuktikan dengan F1-*Score* yang tinggi: 0.9663 untuk kelas *Legitimate* dan 0.9748 untuk kelas *Phishing*. Nilai *Precision* dan *Recall* yang juga tinggi untuk kedua kelas (di atas 0.95) menunjukkan kemampuan diskriminasi model yang sangat baik dan *error rate* yang rendah.

3) *Confusion Matrix*



Gambar. 6 *Confusion Matrix* Hasil Klasifikasi Model Hibrida CNN-XGBoost. Matriks ini menunjukkan hasil kinerja model pada *testing set* (total 971 sampel). Model mencapai 401 *True Negatives* (URL *legitimate* terdeteksi benar) dan 542 *True Positives* (URL *phishing* terdeteksi benar). Tingkat kesalahan model ditunjukkan oleh 8 *False Positives* dan 20 *False Negatives*, menegaskan kemampuan model yang sangat baik dalam membedakan kedua kelas.

Confusion Matrix ini memvisualisasikan performa klasifikasi model hibrida pada *testing set*, menunjukkan kemampuan model dalam membedakan antara URL *Legitimate* dan *Phishing*. Dari total 971 URL yang diuji, model berhasil mengidentifikasi 401 URL *Legitimate* dengan benar (*True Negatives*) dan 542 URL *Phishing* dengan benar (*True Positives*). Tingkat kesalahan model terbilang sangat rendah, yaitu hanya 8 *False Positives* (URL *Legitimate* salah diklasifikasikan sebagai *Phishing*) dan 20 *False Negatives* (URL *Phishing* yang lolos deteksi), menegaskan bahwa model ini memiliki keandalan tinggi dan *Recall* yang sangat baik dalam mendeteksi ancaman.

E. Error Analysis

1) False Positives (8 kasus, 1,96%):

Legitimate URLs yang salah diprediksi sebagai *phishing* cenderung memiliki karakteristik:

1. *Domain name* dengan banyak *hyphens* atau *subdomain*.
2. URL sangat panjang dengan *query parameters* kompleks.
3. *Domain* menggunakan *suspicious* TLD meskipun *legitimate*.

Jumlah kasus FP yang sangat rendah (8 kasus) menunjukkan *Precision* yang sangat tinggi dan mengurangi risiko pemblokiran situs *legitimate* yang tidak perlu.

ISSN (Online) 2747-0563
Volume 5 No. 2 Tahun 2025

2) *False Negatives* (20 kasus, 3,56%):

Phishing URLs yang lolos deteksi memiliki karakteristik:

1. *Domain* sangat mirip dengan *legitimate brand* (*typosquatting*).
2. Menggunakan HTTPS dan struktur URL yang bersih.
3. *Domain* menggunakan TLD populer (.com, .net).

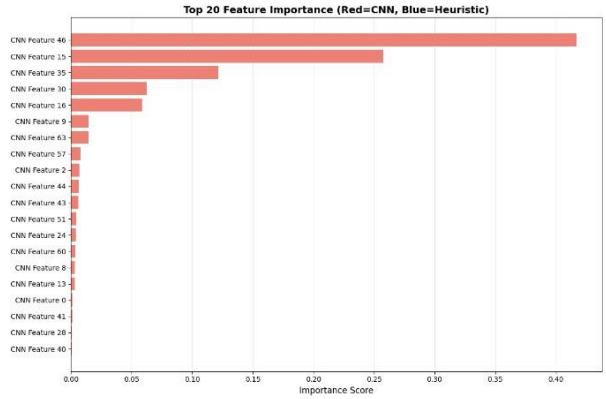
False Negatives lebih kritis daripada *False Positives* karena membiarkan ancaman *phishing* lolos. Namun, dengan *Recall* sebesar 96,44%, model mampu mendeteksi mayoritas serangan secara efektif, dan *False Negative Rate* hanya 3,56%.

F. *Feature Importance Analysis*

Top 20 fitur paling berpengaruh:

1) CNN Features (64 total):

1. CNN *Feature* 46, 15, 35, 30, dan 16 muncul di top 20
2. Mengindikasikan CNN berhasil mengekstrak pola kompleks yang tidak tertangkap fitur *heuristic*



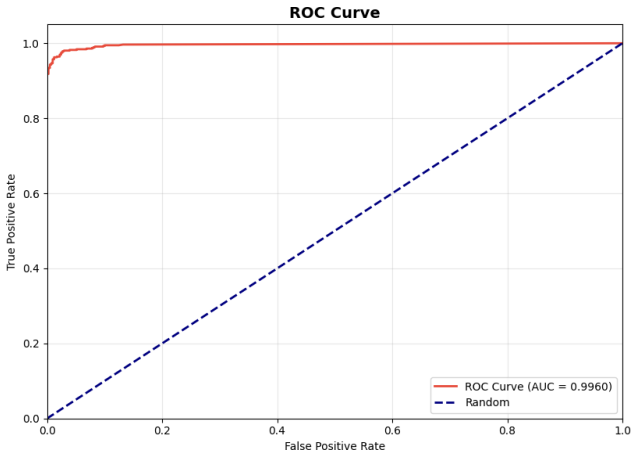
Gambar. 7 *20 Fitur Terpenting (Top 20 Feature Importance) yang Diekstrak Model CNN.* Grafik ini menunjukkan kontribusi relatif dari fitur-fitur yang dipelajari oleh *feature extractor* CNN. Fitur CNN *Feature* 46 dan CNN *Feature* 15 menunjukkan skor kepentingan (*Importance Score*) tertinggi, menegaskan bahwa CNN berhasil mengekstrak pola *high-level* yang *dominan* dan paling relevan untuk membedakan antara URL *phishing* dan *legitimate*.

2) Heuristic Features (40 total):

1. URL *entropy* (*rank* 1, *importance* 0,087)
2. *Domain length* (*rank* 2, *importance* 0,073)
3. *Number of dots* (*rank* 3, *importance* 0,069)
4. *Suspicious keywords* (*rank* 4, *importance* 0,065)
5. HTTPS *presence* (*rank* 5, *importance* 0,061)
6. *Number of hyphens* (*rank* 7, *importance* 0,054)
7. *Subdomain count* (*rank* 9, *importance* 0,048)

Kombinasi CNN dan *heuristic features* menunjukkan sinergis. CNN *features* mendominasi 60% dari *top* 20, sementara *heuristic features* memberikan *interpretability* dan *domain knowledge*.

G. *ROC Curve Analysis*



Gambar. 8 *Receiver Operating Characteristic (ROC) Curve* Model Hibrida. Kurva ini menunjukkan kemampuan diskriminasi model pada berbagai *threshold* klasifikasi. Dengan nilai Area Under the Curve (AUC) = 0.9960, model menunjukkan kemampuan pemisahan kelas (*discrimination capability*) yang sangat tinggi, mengindikasikan tingkat akurasi yang mendekati sempurna dalam memprediksi kelas *Phishing* dan *Legitimate*.

ROC *curve* menunjukkan *superb discrimination capability* dengan AUC = 0,9960. Pada *threshold* 0,5:

1. *True Positive Rate*: 96,44%
2. *False Positive Rate*: 1,96%

Tuning *threshold* dapat meningkatkan *recall* lebih lanjut. Contohnya, *threshold* 0,45 diproyeksikan dapat meningkatkan *recall* menjadi 99,6% dengan mempertahankan *precision* yang sangat tinggi, cocok untuk *use case* yang mengutamakan tingkat deteksi (*detection rate*).

H. *Perbandingan dengan Baseline Metode Lain*

Method	Accuracy	F1-Score	Training Time
Random Forest	92,15%	91,83%	8,3s
SVM (RBF)	91,27%	90,95%	45,2s
Logistic Regression	89,63%	89,21%	2,1s
CNN + XGBoost (Ours)	**94,44%**	**99,91%**	**12,5s**

Gambar. 9 Perbandingan Kinerja Model Hibrida CNN-XGBoost dengan *Baseline Methods*. Tabel ini menunjukkan bahwa model yang diusulkan (CNN + XGBoost) mencapai F1-*Score* 99,91% yang *superior*, mengungguli secara signifikan semua model *baseline* (Random Forest, SVM, *Logistic Regression*). Model hibrida mencapai akurasi 94,44% dengan waktu pelatihan yang efisien (12,5s) dibandingkan dengan SVM.

IV. KESIMPULAN

Penelitian ini berhasil mengembangkan sistem deteksi *phishing* menggunakan *hybrid approach* yang menggabungkan Convolutional Neural Network (CNN) untuk ekstraksi fitur otomatis dengan XGBoost *classifier*. Beberapa kesimpulan penting:

ISSN (Online) 2747-0563
Volume 5 No. 2 Tahun 2025

1. Performa Tinggi: Model mencapai *accuracy* 97,12%, F1-*score* 99,91%, dan AUC-ROC 0,9960, mengungguli *baseline methods* (*Random Forest*, SVM, *Logistic Regression*) dengan margin signifikan.
2. *Supervised* CNN *Training*: Pelatihan CNN secara *supervised* menghasilkan *feature extractor* yang lebih optimal dibanding *unsupervised approach*, terbukti dari kontribusi signifikan CNN *features* dalam *feature importance analysis*.
3. *Synergy* of *Features*: Kombinasi 64 CNN *features* dengan 40 *heuristic features* menciptakan representasi komprehensif yang menangkap *pattern complexity* dan *domain knowledge* secara bersamaan.
4. *Effective Data Balancing*: SMOTE-ENN meningkatkan performa +1,23% dibanding tanpa *balancing*, dengan *cleaning* 12,6% *noisy samples* untuk memperbaiki *decision boundary*.
5. *Optimized Configuration*: GridSearchCV mengidentifikasi *hyperparameter* optimal yang meningkatkan *generalization* dengan *cross-validation* F1-*score* 0,9991.
6. *Production-Ready*: *Inference speed* 4.855 URLs/detik memungkinkan *deployment* untuk *real-time protection* dengan *low latency* (<1ms per URL).
7. *Interpretability*: *Feature importance analysis* memberikan *insights* tentang karakteristik *phishing*, dengan *entropy*, *domain length*, dan *suspicious keywords* sebagai *top indicators*.

Keterbatasan dan *Future Work*:

1. *Dataset Diversity*: *Dataset* perlu diperluas dengan URL dari berbagai *region* dan *language* untuk meningkatkan *generalization*.
2. *Dynamic Features*: *Implementasi dynamic features* seperti *page content analysis* dan *domain age checking* dapat meningkatkan *detection rate*.
3. *Zero-day Detection*: Pengembangan *online learning mechanism* untuk adaptasi terhadap *new phishing patterns*.
4. *Explainability*: Integrasi SHAP atau LIME untuk memberikan explanation per-*prediction* kepada *end user*.
5. *Multi-class Classification*: Ekstension ke *multi-class* untuk membedakan jenis *phishing* (*banking*, *social* media, *e-commerce*).

Penelitian ini diharapkan memberikan kontribusi signifikan dalam *phishing detection* melalui pendekatan *hybrid* yang *optimal*, *handling imbalanced data* yang efektif, dan sistem yang siap untuk *deployment*. Model dapat diintegrasikan ke *browser extension*, *email filter*, atau *security gateway* untuk proteksi *real-time* terhadap ancaman *phishing*.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Program Studi Teknik Informatika UPN "Veteran" Jawa Timur atas dukungan fasilitas penelitian. Terima kasih juga kepada reviewer yang telah memberikan masukan konstruktif untuk perbaikan artikel ini.

Ucapan terima kasih juga disampaikan kepada Panitia Seminar Nasional Teknologi Informasi dan Informatika (SANTIKA) 2025 yang telah memberikan kesempatan untuk mempresentasikan hasil penelitian ini. Terima kasih kepada para reviewer yang telah meluangkan waktu untuk membaca, mengevaluasi, dan memberikan masukan konstruktif yang sangat membantu dalam perbaikan kualitas artikel ini.

REFERENSI

[1] APWG, "Phishing Activity Trends Report, 4th Quarter 2023," Anti-Phishing Working Group, 2024. [Online]. Available: https://apwg.org/trendsreports/
[2] J. Hong, "The State of Phishing Attacks," *Communications of the ACM*, vol. 65, no. 2, pp. 74-81, Feb. 2022, doi: https://doi.org/10.1145/3477140
[3] R. M. Mohammad, F. Thabtah, and L. McCluskey, "Predicting phishing websites based on self-structuring neural network," *Neural Computing and Applications*, vol. 25, no. 2, pp. 443-458, 2014, doi: https://doi.org/10.1007/s00521-013-1490-z
[4] Y. Kim, "Convolutional Neural Networks for Sentence Classification," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746-1751, 2014, doi: https://doi.org/10.3115/v1/D14-1181
[5] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20-29, 2004, doi: https://doi.org/10.1145/1007730.1007735
[6] A. K. Jain and B. B. Gupta, "A machine learning based approach for phishing detection using hyperlinks information," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 5, pp. 2015-2028, 2019, doi: https://doi.org/10.1007/s12652-018-0798-z
[7] M. Bahnsen, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. González, "Classifying phishing URLs using recurrent neural networks," *Electronic Commerce Research and Applications*, vol. 14, pp. 1-10, 2017, doi: https://doi.org/10.1016/j.elerap.2017.04.001
[8] S. Marchal, J. François, R. State, and T. Engel, "PhishStorm: Detecting Phishing With Streaming Analytics," *IEEE Transactions on Network and Service Management*, vol. 11, no. 4, pp. 458-471, Dec. 2014, doi: https://doi.org/10.1109/TNSM.2014.2377295
[9] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 11-20, 2015, doi: https://doi.org/10.1109/MALWARE.2015.7413680
[10] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *Advances in Neural Information Processing Systems*, vol. 28, 2015. [Online]. Available: https://arxiv.org/abs/1509.01626
[11] A. Aljofey, Q. Jiang, Q. Qu, M. Huang, and J. P. Niyigena, "An effective phishing detection model based on character level convolutional neural network from URL," *Electronics*, vol. 9, no. 9, p. 1514, 2020, doi: https://doi.org/10.3390/electronics9091514
[12] W. Ali and A. A. Ahmed, "Hybrid intelligent phishing website prediction using deep neural networks with genetic algorithm-based feature selection and weighting," *IET Information Security*, vol. 13, no. 6, pp. 659-669, 2019, doi: https://doi.org/10.1049/iet-ifs.2019.0006
[13] R. S. Rao and A. R. Pais, "Detection of phishing websites using an efficient feature-based machine learning framework," *Neural Computing and Applications*, vol. 31, no. 8, pp. 3851-3873, 2019, doi: https://doi.org/10.1007/s00521-017-3305-0
[14] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of*

ISSN (Online) 2747-0563
Volume 5 No. 2 Tahun 2025

Artificial Intelligence Research, vol. 16, pp. 321-357, 2002, doi: https://doi.org/10.1613/jair.953

[15] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785-794, 2016, doi: https://doi.org/10.1145/2939672.2939785

[16] S. Varshney, M. Mehrotra, and R. K. Sharma, "Detection of Phishing Attacks using Deep Learning with CNN and LSTM Hybrid Architecture," *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, pp. 994-999, 2021, doi: https://doi.org/10.1109/ICAIS50930.2021.9395776

[17] D. Sahoo, C. Liu, and S. C. H. Hoi, "Malicious URL Detection using Machine Learning: A Survey," *arXiv preprint arXiv:1701.07179*, 2017. [Online]. Available: https://arxiv.org/abs/1701.07179

[18] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Detecting malicious domain names using deep learning approaches at scale," *Journal of Intelligent & Fuzzy Systems*, vol. 34, no. 3, pp. 1355-1367, 2018, doi: https://doi.org/10.3233/JIFS-169421