Methotika : Jurnal Ilmiah Teknik Informatika
ISSN : 2776 – 5792
Vol. 3, No. 1, April 2023, pp. 1-12
http://ojs.fikom-methodist.net/index.php/methotika

Deteksi Serangan DDOS Pada VPS Menggunakan Metode Deep Neural Network

Prialdi Simarmata[1], Naikson Fandier Saragih[2], Indra Kelana Jaya[3]
[1,2,3]Fakultas Ilmu Komputer, Universitas Methodist Indonesia

Info Artikel

Histori Artikel:

Received, Desember, 2022
Revised, Januari, 2023
Accepted, Januari, 2023

Keywords:

Deep Neural Network,
Deep Learning,
Batch Size,
Epoch,
Klasifikasi,
Confusion Matrix,
Virtual Private Server,
python

ABSTRAK

Perkembangan internet dan komunikasi yang pesat telah memperluas ancaman dan serangan keamanan siber. Saat ini kebutuhan manusia sangat bergantung pada keberadaan informasi atau data digital, semakin besar permintaan akan informasi, semakin tajam gangguan keamanan pada sistem jaringan. Sistem *Intrusion Detection System* dibangun menggunakan metode *Deep Neural Network* untuk mendeteksi dan mengetahui nilai akurasi deteksi serangan jaringan. Penelitian ini menggunakan Dataset hasil simulasi serangan DDOS menggunakan LOIC 1.0.8.0, selama 35 menit pada *Virtual Private Server*. Data serangan dicapture dengan Wireshark 3.6.7.0 diperoleh 39920 record data, selanjutnya data diekstraksi dengan CICFlowMeter 4.0, diperoleh ekstraksi dataset sebanyak 70343 baris data. Program metode DNN dibangun dengan pemograman python, dalam program dilakukan praproses data, normalisasi data dilakukan dengan metode min-max. Dataset sebanyak 70343 data dibagi menjadi data latih dan data uji dengan perbandingan 80 : 20, diperoleh data latih 56274 data dan data uji 14069 data. Tahapan pengujian dengan DNN dilakukan sebanyak 20 kali percobaan, dengan menggunakan parameter layer yang berbeda dengan parameter epoch 20 dan batch size 64. Hasil percobaan ke 10 dihasilkan model DNN dengan 4 layer dengan fungsi aktivasi relu dan softmax, menggunakan parameter epoch 20 dan batch size 64 mendapatkan hasil klasifikasi akurasi tertinggi dengan nilai 96,50%. Adapun pengujian dengan Dataset serangan DDOS tanggal 21-02-2018 dari situs https://www.unb.ca/cic/datasets/ids-2018.html dengan dataset CIC sebagai data latih sebanyak 838860 data dan menggunakan dataset VPS sebagai data uji sebanyak 14069. Tahapan pengujian dengan DNN dilakukan dengan percobaan sebanyak 10 kali dengan menggunakan parameter layer yang berbeda, menggunakan parameter epoch berbeda dan batch size 64. Hasil percobaan ke 7 dihasilkan model DNN dengan 4 layer dengan fungsi aktivasi relu dan softmax, menggunakan parameter epoch 7 dan batch size 64 mendapatkan hasil klasifikasi akurasi tertinggi dengan nilai 45,87% dengan kategori akurasi sedang.

Penulis Koresponden

Naikson Fandier Saragih,
Fakultas Ilmu Komputer,
Universitas Methodist Indonesia, Medan,
Jl. Hang Tuah No.8, Medan - Sumatera Utara.
Email: saragihnaikson@gmail.com

1. PENDAHULUAN

Dalam mencegah resiko terkena serangan, dibutuhkan suatu sistem pemantauan paket data dalam jaringan dan menganalisa paket tersebut untuk mengamankan jaringan agar tidak menjadi target seorang

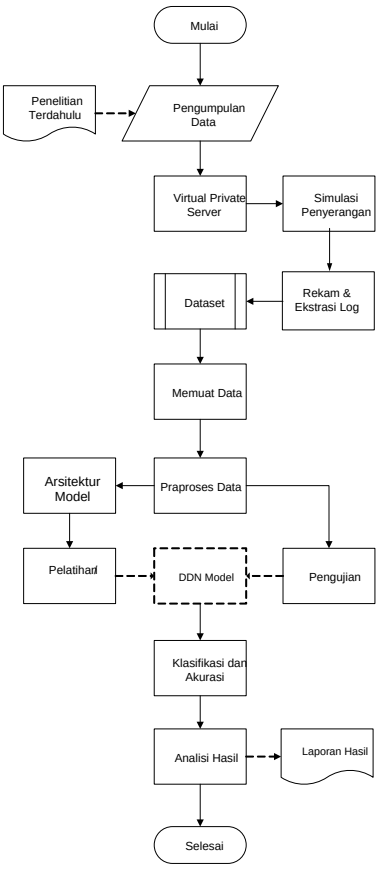
penyerang. Sistem yang dapat digunakan untuk mencegah resiko terkena serangan yaitu *Intrusion Detection System* (IDS) yang dapat mendeteksi aktivitas yang mencurigakan dalam sebuah jaringan [1]. *Virtual Private Server* atau VPS adalah sebuah metode untuk mempartisi atau membagi sumber daya / resource sebuah server menjadi beberapa server virtual. VPS merupakan implementasi dari teknologi virtualisasi, VPS banyak digunakan pada saat *traffic website* sudah tinggi dan memerlukan *resource* lebih, saat memerlukan akses IP *Private*, dan membuat aplikasi berbasis file *sharing*, *streaming*, MAP Server [2].

Distributed Denial of Service (DDoS) adalah serangan yang dapat menyebabkan keamanan pada server atau sistem menjadi down. Tujuan serangan *DDoS* untuk melumpuhkan sistem target dengan cara menghabiskan jaringan atau sumber daya dari suatu sistem yang dimiliki pengguna. Serangan *DDoS* memang sering terjadi di perusahaan maupun instansi pemerintahan oleh karena itu penelitian akan terfokus pada serangan tersebut untuk mendeteksi dan mengklasifikasi melalui log data jaringan.Dalam perkembangan teknologi telah populer teknik baru yang diterapkan sebagai IDS yaitu *Deep Learning*. *Deep Learning* adalah bidang ilmu komputer yang menggunakan teknik statistika untuk memberi kemampuan sistem komputer agar dapat belajar dari data . *Deep Neural Network* adalah salah satu jenis *Deep Learning* yang digunakan sebagai IDS. *Deep Neural Network* adalah sebuah *Artificial Neural Network* dengan beberapa lapisan antara lapisan input dan output. *Deep Neural Network* dapat menemukan manipulasi matematis yang benar untuk mengubah input menjadi output, apakah itu hubungan linear atau hubungan non-linear [3]. DNN bekerja dengan mengubah data yang semula berupa log serangan jaringan menjadi teks atau angka untuk dijadikan sebagai dataset. Data log serangan DDoS tersebut akan dijadikan sebagai dataset kemudian di proses menggunakan metode DNN untuk mendapatkan nilai akurasi.

2. METODE PENELITIAN

2.1 Framework Penelitian

Framework penelitian yaitu kerangka penelitian yang akan membantu penulis dalam melakukan proses penelitian sehingga mendapatkan hasil yang diinginkan. Adapun tahapan-tahapan yang dilakukan, dapat dilihat pada Gambar 1.



Gambar 1. Framework Penelitian

2.2 Dataset

Dataset terbagi menjadi dua yaitu data latih (Training) dan data uji (Testing) dengan perbandingan split data 80% : 20%. Data uji diambil dari hasil pengujian skenario serangan terhadap VPS dan data latih diambil dari situs https://www.unb.ca/cic/datasets/ids-2018.html berikut adalah kedua dataset tersebut :

a. Data Latih (Training)
   Data latih diperoleh dari situs resmi milik *Canadian Institute for Cybersecurity.* Data yang didwonload yakni daftar serangan harian DDoS berdasarkan pada hari rabu tanggal 21-02-2018. Data yang diperoleh berjumlah 1048575 baris data, namun data yang digunakan untuk pelatihan berjumlah 387040.
b. Data uji
   Data uji yang digunakan merupakan hasil dari skenario serangan DDoS pada VPS. Pengujian serangan terhadap VPS mendapatkan jumlah 38725. Hasil ini di ekstrak menggunakan tools CICFlowMeter 4.0 file yang didapat berformat CSV.
c. Label
   Label dalam dataset diubah menjadi 2 label didalamnya yaitu Benign (Normal) dan DDOS yang kemudian ditranformasi ke bentuk array. Berikut tabel tranformasi ubah Label ke angka (*Array*) :

2.3 Skenario pembagian dataset

a. Dataset VPS

Pembagian dataset dibagi dengan split 80 : 20 data. Data *training* dengan size 80% dan data *testing* dengan size 20%. Dapat dilihat pada tabel 1:

Tabel 1 Pembagian Dataset

Data latih (80%)	**Data uji (20%)**
70343 x 0,8 = 56274 data	70343 x 0,2 = 14069 data

b. Dataset DDoS 02-21-2018 (training) dan Dataset VPS (testing)
   Pembagian dataset dibagi dengan split 80: 200 data. Data *training* dengan size 80% dan data *testing* dengan size 20%. Dapat dilihat pada tabel 2:

Tabel 2 Pembagian Dataset

Data latih (80%)	**Data uji (20%)**
838860 data	14069 data

3. HASIL DAN PEMBAHASAN

3.1 Implementasi Sistem

Langkah pertama tahapan implementasi sistem pada penelitian ini adalah menyewa layanan virtual private server dari provider herza.id. Program Deep Neural Network akan dibangun dengan bahasa pemograman python pada google colab.

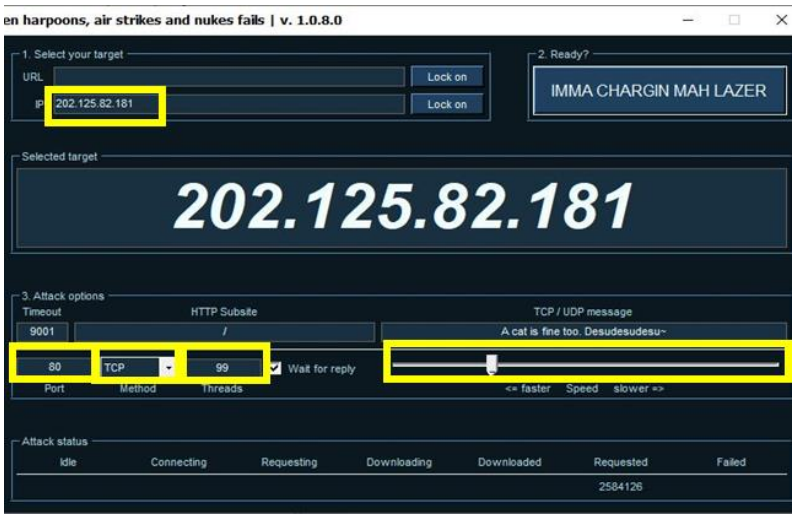
3.2 Pengujian Sistem

Pengujian serangan dilakukan dengan menggunakan tools Loic untuk menyerang VPS. Proses serangan DDOS TCP Flood berdurasi 5 menit, kedua merupakan aliran data normal berdurasi 6 menit 28 detik, ketiga dengan serangan DDOS UDP Flood berdurasi 5 menit, keempat merupakan aliran data normal berdurasi 7 menit 53 detik, dan ke lima serangan ICMP Flood berdurasi 10 menit. Total durasi waktu pengujian serangan sekitar 35 Menit. Berikut tabel durasi pengujian serangan DDOS dan aliran normal yang telah dilakukan :

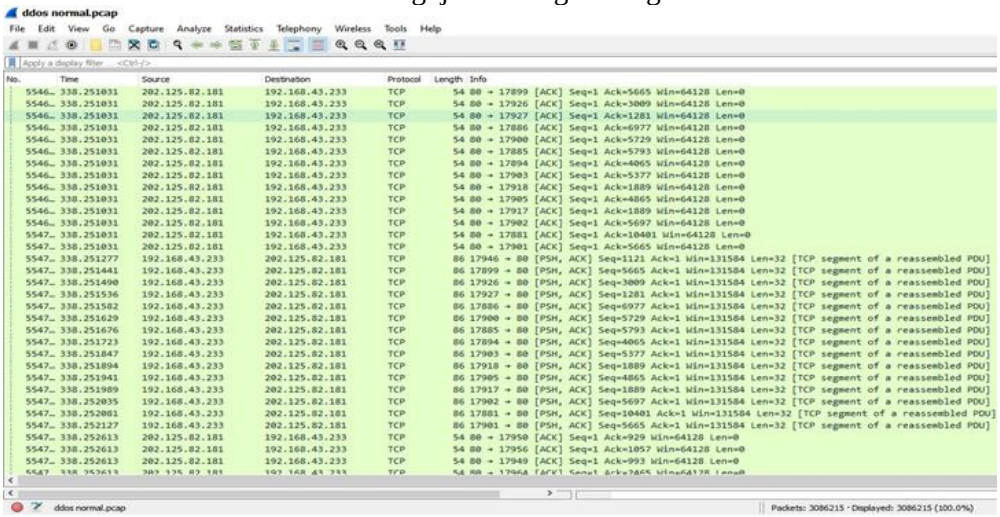
Tabel 3 Pengujian Serangan

IP Penyerang	IP Tujuan	Nama	Mulai	Selesai
192.168.43.66	103.160.63.239	DDOS attack-LOIC-TCP	03.20.00	03.25.00
192.168.43.66	103.160.63.239	Benign (Normal)	03.25.01	03.31.29
192.168.43.66	103.160.63.239	DDOS attack-LOIC-UDP	03.32.00	03.37.00
192.168.43.66	103.160.63.239	Benign (Normal)	03.37.24	03.44.17
192.168.43.66	103.160.63.239	DDOS attack-ICMP	03.45.01	03.55.02

Aplikasi *LOIC* digunakan untuk pengujian serangan *DDOS TCP Flood dan DDOS UDP Flood* sesuai dengan durasi waktu penyerangan pada tabel 5. Pengujian *TCP Flood* dan *UDP Flood* dilakukan penyerangan dengan mengirim data sebanyak mungkin sehingga server akan mengalami peningkatan kinerja *CPU* dan server akan melambat. Konfigurasi dilakukan pada software *LOIC* dengan IP VPS/target 202.125.82.181, untuk *port* yang digunakan *port* 80, menggunakan *method TCP* serangan pertama dan method UDP serangan ke dua. *Thereads* yang digunakan sebanyak 99 dan kecepatan pengiriman paket data diatur ke *Faster*. Gambar pengujian serangan disampaikan pada gambar 2



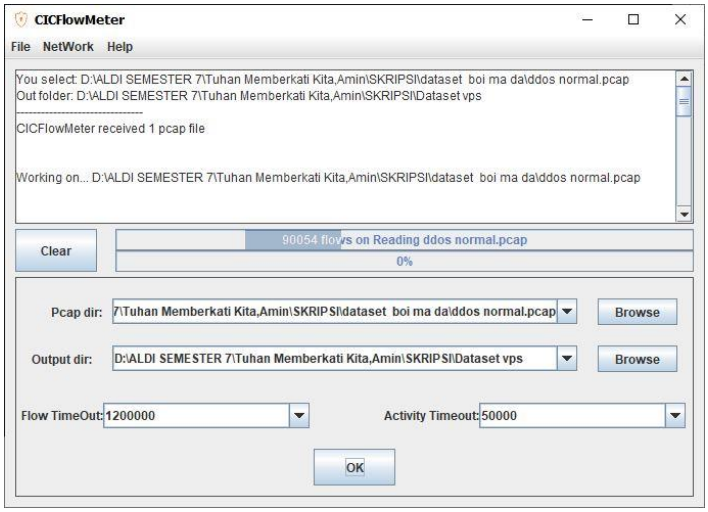
Gambar 2. Pengujian serangan dengan LOIC



Gambar 3. Perekaman paket data dengan wireshark

Pada gambar 3 merupakan proses perekaman paket data menggunakan Wireshark. Berdasarkan hasil pengujian dengan skenario serangan dan durasi waktu yang telah direncanakan, total data packet yang direkam *wireshark* berjumlah 39920 baris data. Setelah dilakukan proses perekaman lalu lintas jaringan,

semua data lalu lintas jaringan yang telah terekam oleh wireshark disimpan dalam format data PCAP. Namun format PCAP belum sesuai dengan format semestinya, sehingga perlu dilakukan ekstraksi data menjadi sebuah data yang dapat dibaca oleh sistem, yaitu format CSV. Tool yang digunakan untuk ekstrasi data menjadi CSV adalah *CICFlowMeter*. Proses ektraksi data *PCAP* ke *CSV* dilakukan dengan menggunakan *tool CICFlowmeter*. Hasil dari proses ekstraksi data dengan *CICFlowmeter* didapatkan 70433 baris data. Proses ekstraksi data disampaikan pada Gambar 4.

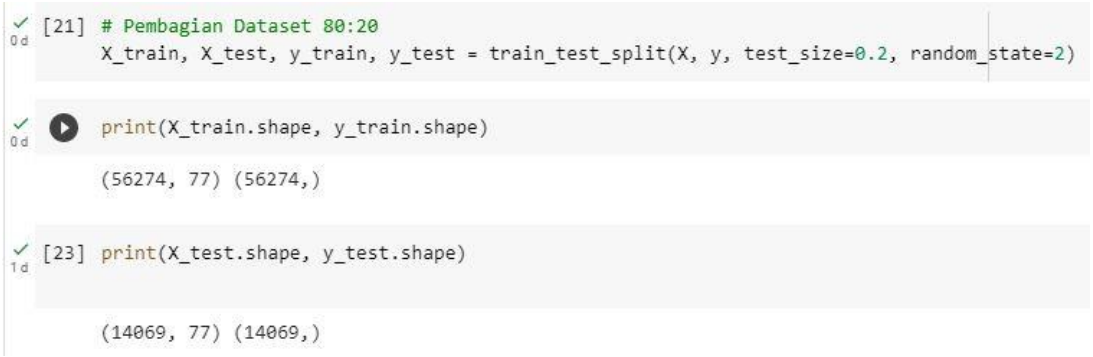


Gambar 4. Ekstraksi Dataset dengan CICFlowMeter

3.3 Tampilan Sistem

3.3.1. Dataset DDOS VPS

Dataset dengan format csv akan di input kedalam program *Deep Neural Network*. Program Deep Neural Network dibangun dengan bahasa pemograman python dan dijalankan menggunakan google colab. Data set Serangan DDoS VPS dibagi menjadi 2 yaitu data latih sebagai data pelatihan dan data uji sebagai data pengujian. Perbandingan dari pembagian data adalah 80:20, dimana 80% untuk data latih dan 20% untuk data uji. Pembagian data dilakukan dengan mengatur parameter test_size = 0.2.



Gambar 5. Pembagian Data latih dan Data uji

Pada gambar 5 Pembagian Dataset telah dilakukan, diperoleh untuk data latih (training) sebanyak 56274 data dan untuk data uji (testing) diperoleh sebanyak 14069 data.

3.3.2. Normalisasi Data

Proses normalisasi data berfungsi untuk mengatur data dalam meminimalkan duplikasi berbagai hubungan pada data dan untuk membantu meningkatkan kecepatan dan akurasi. Normalisasi data diproses

menggunakan metode minmax dengan persamaan 1. Proses normalisasi data dimulai dengan mengimport class MinmaxScaler dari library sklearn, kemudian MinMaxscaler di tampung pada variabel mm, selanjutnya data latih dan uji di transform/di normalisasi. Pada code normalisasi fit_transform method adalah dua method yang digabungkan menjadi satu yaitu method fit dan method transform. Fit bertujuan untuk melatih model. Maka dari itu, di testing dataset hanya menggunakan transform. Jika kita menggunakan fit_transform method di testing dataset, maka model akan mempelajari data baru menggunakan testing dataset. Proses normalisasi disampaikan pada gambar 6.

```
#Normalisasi data
from sklearn.preprocessing import MinMaxScaler #import class MinMaxScaler dari library sklearn.preprocessing

mm = MinMaxScaler() #class MinMaxScaler ditampung dalam variabel mm
X_train = mm.fit_transform(X_train) #untuk proses normalisasi data latih, fit transform digunakan pada data latih
X_test = mm.transform(X_test) #untuk proses normalisasi data uji, transform  digunakan pada data latih.
```

Gambar 6. Normalisasi Data

Pada gambar 7 disampaikan hasil normalisasi dari data latih dan uji, dapat dilihat bahwa data bertipe numerik sudah dalam rentang nilai antara -1,0 sampai 1,0

```
X_train #menampilkan hasil normalisasi

array([[ 0.75435443,  1.17793315,  0.78929453, ...,  0.83078236,
        -0.18266405,  0.8327157 ],
       [ 0.75435443,  1.17793315,  0.94293605, ...,  0.97431604,
        -0.18266405,  0.98937597],
       [-1.32563681, -0.84894461,  0.17528513, ...,  0.27502275,
        -0.18266405,  0.22026487],
       ...,
       [ 0.75435443,  1.17793315,  0.8024691 , ...,  0.8430902 ,
        -0.18266405,  0.84614913],
       [ 0.75435443,  1.17793315,  0.94813165, ...,  0.97916983,
        -0.18266405,  0.99467366],
       [ 0.75435443,  1.17793315,  0.83438497, ...,  0.87290638,
        -0.18266405,  0.87869208]])

X_test #menampilkan hasil normalisasi

array([[ 0.75435443,  1.17793315,  0.87242424, ...,  0.9084431 ,
        -0.18266405,  0.91747875],
       [-1.32563681, -0.84894461, -0.72318375, ..., -0.60680793,
        -0.18266405, -0.73180575],
```

Gambar 7. Hasil Normalisasi

3.3.3. Model Deep Neural Network

Model DNN yang digunakan adalah DNN dengan 4 *layer*. Untuk layer 1 merupakan layer input dengan input 77 fitur dataset dan jumlah neuron 256 menggunakan activation relu. Untuk layer 2 merupakan layer hidden layer pertama dengan jumlah neuron 124 menggunakan activation relu, layer 3 merupakan layer hidden layer kedua dengan neuron berjumlah 124 untuk hidden layer menggunakan activation relu. Untuk layer 4 merupakan layer output dengan jumlah neuron 3 menggunakan fungsi aktivasi softmax. Sequential digunakan pada model DNN supaya pemrosesan berutan dari layer 1 ke ke layer 4. Pada layer 1-3 parameter dropout ditambahkan untuk mencegah terjadinya overfitting dan mempercepat pada proses pelatihan

```
[ ]  #model DNN
     model1 = Sequential() # Sequential : pemrosesan yang berurutan layer 1-4
     model1.add(Dense(256,input_dim=77,activation='relu')) #dense : layer pada model arsitektur yang berisi neuron neuron
     model1.add(Dropout(0.01)) #Dropout : proses mencegah terjadinya overfitting dan juga mempercepat proses learning
     model1.add(Dense(124, activation='relu')) #activation : menentukan output dari sebuah neuron
     model1.add(Dropout(0.01))
     model1.add(Dense(124, activation='relu'))
     model1.add(Dropout(0.01))
     model1.add(Dense(3))
     model1.add(Activation('softmax'))
```

Gambar 8. Model Deep Neural Network

3.3.4. Pelatihan

Pada gambar 9 merupakan proses untuk pembelajaran / melatih model Deep Neural Network. Pada tahapan ini model DNN dilakukan pengujian sebanyak 20 kali percobaan untuk mendapatkan hasil terbaik, dengan menggunakan parameter layer yang berbeda. Proses pelatihan dimulai dengan mengimport class keras dari library tensorflow, kemudian model dicompile dengan code model1.compile dengan loss binary crossentropy untuk mengatur tingkat kesalahan antara output prediksi dengan nilai target, dan optimizer adam digunakan untuk membantu meningkatkan akurasi dan meminimalkan nilai loss. Parameter epoch 20 adalah ketika seluruh dataset sudah melalui proses pelatihan sampai dikembalikan ke awal untuk sekali putaran kemudian dilakukan kembali proses pelatihan sesuai dengan jumlah epoch yaitu 20. Batch size 64 adalah jumlah sampel data yang akan di latih dalam satu iterasi.

```
from tensorflow import keras # import class keras dari library tensorflow
model1.compile(loss='binary_crossentropy',optimizer=keras.optimizers.Adam(learning_rate=0.01),metrics=['accuracy'])
#loss : digunakan untuk mengatur tingkat kesalahan antara keluaran prediksi dan nilai target
#optimizers : membantu meningkatkan akurasi dan meminimalkan nilai loss / cost function.
csv_logger = CSVLogger("log.csv",separator=',', append=False)

history = model1.fit(X_train, y_train, validation_data=(X_test, y_test),batch_size=64, epochs=50,callbacks=[])
model.save("DatasetVPS/dnn223_model.hdf5")
```

Gambar 9. Proses Pelatihan

Dari 20 kali percobaan yang telah dilakukan**,** pelatihan pada percobaan ke 10 dihasilkan model DNN dengan 4 layer, parameter epoch 20, size 64, fungsi loss binary crossentropy dan optimizer adam mendapatkan hasil klasifikasi akurasi tertinggi dengan nilai 96,57 %. Gambar hasil pelatihan disampaikan pada gambar 10**.**

```
Epoch 12/20
880/880 [==============================] - 4s 5ms/step - loss: 0.0989 - accuracy: 0.9655 - val_loss: 0.0981 - val_accuracy: 0.9648
Epoch 13/20
880/880 [==============================] - 6s 7ms/step - loss: 0.0991 - accuracy: 0.9656 - val_loss: 0.0961 - val_accuracy: 0.9650
Epoch 14/20
880/880 [==============================] - 5s 6ms/step - loss: 0.0975 - accuracy: 0.9656 - val_loss: 0.0958 - val_accuracy: 0.9650
Epoch 15/20
880/880 [==============================] - 4s 5ms/step - loss: 0.0978 - accuracy: 0.9657 - val_loss: 0.0975 - val_accuracy: 0.9650
Epoch 16/20
880/880 [==============================] - 6s 7ms/step - loss: 0.0974 - accuracy: 0.9656 - val_loss: 0.0958 - val_accuracy: 0.9650
Epoch 17/20
880/880 [==============================] - 4s 5ms/step - loss: 0.1904 - accuracy: 0.9649 - val_loss: 0.1911 - val_accuracy: 0.9649
Epoch 18/20
880/880 [==============================] - 5s 5ms/step - loss: 0.0991 - accuracy: 0.9657 - val_loss: 0.1047 - val_accuracy: 0.9649
Epoch 19/20
880/880 [==============================] - 7s 8ms/step - loss: 0.0996 - accuracy: 0.9657 - val_loss: 0.0984 - val_accuracy: 0.9649
Epoch 20/20
880/880 [==============================] - 4s 5ms/step - loss: 0.1034 - accuracy: 0.9657 - val_loss: 0.0961 - val_accuracy: 0.9650
```

Gambar 10. Hasil Pelatihan berupa Tingkat Nilai Akurasi

3.3.5. Pengujian

Pada gambar 11 merupakan proses pengujian model Deep Neural Network. Pada tahapan ini model DNN yang telah mengalami tahapan pelatihan akan di uji menggunakan data uji untuk melihat peforma

model DNN. Pengujian di proses dengan code model1.evaluate dengan menggunakan data testing. Hasil Pengujian dengan model DNN diperoleh akurasi tertinggi dengan nilai 96,50%

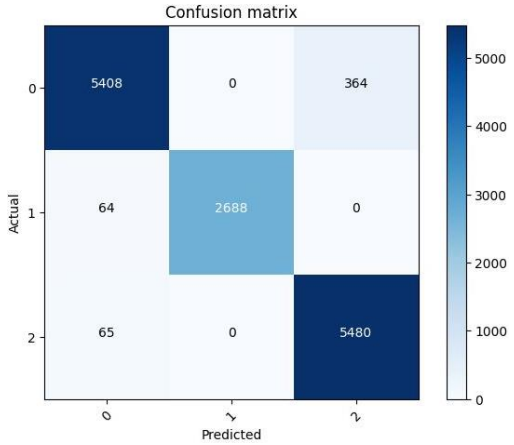
```
#pengujian
loss, accuracy = model.evaluate(X_test, y_test) #model1.evaluate : menguji data uji dengan model yang dilatih
print("\nLoss: %.3f, Accuracy: %.2f%%" % (loss*100, accuracy*100)) #menampilkan hasil akurasi dan loss
```

```
440/440 [==============================] - 3s 7ms/step - loss: 0.0961 - accuracy: 0.9650

Loss: 9.611, Accuracy: 96.50%
```

Gambar 11. Pengujian dan Hasil Pengujian berupa nilai akurasi

Pada gambar 12 disampaikan hasil confusion matrix pendeteksian klasifikasi serangan DDoS dengan prediksi benar pada dataset VPS sebanyak 5408 data untuk kelas data normal, 2688 data untuk kelas data DDoS TCP, dan, 5480 data untuk kelas data UDP Flood.



Gambar 12. Hasil Confusion Matrix

3.3.6. Dataset CIC 2018 dan dan Dataset VPS

Pada tahapan dilakukan pengujian model DNN dengan Dataset CIC 2018 dan dan Dataset VPS. Dataset IDS 02-21-2018 sebagai data latih dan dataset VPS sebagai data uji dengan pembagian data 80% : 20%. Pembagian data dilakukan dengan mengatur parameter frac=0.8 untuk data latih dan frac=0.2 untuk data uji. Gambar pembagian jumlah komposisi data latih dan data uji disampaikan berikut :

```
[24] data_ids = data_ids.sample(frac=0.8, random_state=25)

[25] data_ids.shape

     (838860, 79)

[26] data_vps = data_vps.sample(frac=0.2, random_state=25)

     data_vps.shape

     (14069, 79)
```

Gambar 13 Pembagian Dataset IDS dan VPS

Pada gambar 13 disampaikan jumlah Dataset IDS / data latih sebanyak 838860 data dan jumlah data VPS / data uji sebanyak 14069 data.

3.3.7. Normalisasi Data

Proses normalisasi data berfungsi untuk mengatur data dalam meminimalkan duplikasi berbagai hubungan pada data dan untuk membantu meningkatkan kecepatan dan akurasi. Normalisasi data diproses menggunakan metode minmax dengan persamaan 1. Proses normalisasi data dimulai dengan mengimport class MinmaxScaler dari library sklearn, kemudian MinMaxscaler di tampung pada variabel mm, selanjutnya data latih dan uji di transform/di normalisasi. Pada code normalisasi fit_transform method adalah dua method yang digabungkan menjadi satu yaitu method fit dan method transform. Fit bertujuan untuk melatih model. Maka dari itu, di testing dataset hanya menggunakan transform. Jika kita menggunakan fit_transform method di testing dataset, maka model akan mempelajari data baru menggunakan testing dataset. Proses normalisasi disampaikan pada gambar 14.

```
#Normalisasi data
from sklearn.preprocessing import MinMaxScaler #import class MinMaxScaler dari library sklearn.preprocessing

mm = MinMaxScaler() #class MinMaxScaler ditampung dalam variabel mm
X_train = mm.fit_transform(X_train) #untuk proses normalisasi data latih, fit transform digunakan pada data latih
X_test = mm.transform(X_test) #untuk proses normalisasi data uji, transform  digunakan pada data latih.
```

Gambar 14. Normalisasi Data

Pada gambar 15 disampaikan hasil normalisasi dari data latih dan uji, dapat dilihat bahwa data bertipe numerik sudah dalam rentang nilai antara -1,0 sampai 1,0

```
X_train #tampilan hasil normalisasi
```

```
array([[-0.58967995, -0.05752259, -0.59561934, ..., -0.04193303,
        -0.0382938 , -0.03975383],
       [ 1.14688176, -0.05752259, -0.08249882, ..., -0.04193303,
        -0.0382938 , -0.03975383],
       [-0.58967995, -0.05752259, -0.81110219, ..., -0.04193303,
        -0.0382938 , -0.03975383],
       ...,
       [ 2.28910875, -0.05752259, -0.81633702, ..., -0.04193303,
        -0.0382938 , -0.03975383],
       [-0.58967995, -0.05752259,  1.85770143, ..., -0.04193303,
        -0.0382938 , -0.03975383],
       [-0.58967995, -0.05752259, -0.3718035 , ..., -0.04193303,
        -0.0382938 , -0.03975383]])
```

```
X_test #tampilan hasil normalisasi
```

```
array([[-5.90322016e-01, -1.68998960e+01, -9.12593859e-01, ...,
        -4.19330256e-02, -3.82938032e-02, -3.97538333e-02],
```

Gambar 15. Hasil Normalisasi Data

3.3.8. Model Deep Neural Network

Model DNN yang digunakan adalah DNN dengan 4 *layer*. Untuk layer 1 merupakan layer input dengan input 77 fitur dataset dan jumlah neuron 256 menggunakan activation relu. Untuk layer 2 merupakan layer hidden layer pertama dengan jumlah neuron 124 menggunakan activation relu, layer 3 merupakan layer hidden layer kedua dengan neuron berjumlah 124 untuk hidden layer menggunakan activation relu. Untuk layer 4 merupakan layer output dengan jumlah neuron 3 menggunakan fungsi aktivasi softmax. Sequential digunakan pada model DNN supaya pemrosesan berutan dari layer 1 ke ke layer 4. Pada layer 1-3 parameter dropout ditambahkan untuk mencegah terjadinya overfitting dan mempercepat pada proses pelatihan Model DNN yang digunakan sama dengan model DNN pada gambar 9.

3.3.9. Pelatihan

Pada gambar 176 merupakan proses untuk pembelajaran / melatih model Deep Neural Network. Pada tahapan ini model DNN dilakukan pengujian sebanyak 10 kali percobaan untuk mendapatkan hasil terbaik, dengan menggunakan parameter layer yang berbeda dengan parameter epoch 20, batch size 64, fungsi loss

binary crossentropy, dan optimizer adam. Proses pelatihan dimulai dengan mengimport class keras dari library tensorflow, kemudian model dicompile dengan code model1.compile dengan loss binary crossentropy untuk mengatur tingkat kesalahan antara output prediksi dengan nilai target, dan optimizer adam digunakan untuk membantu meningkatkan akurasi dan meminimalkan nilai loss. Parameter epoch 20 adalah ketika seluruh dataset sudah melalui proses pelatihan sampai dikembalikan ke awal untuk sekali putaran kemudian dilakukan kembali proses pelatihan sesuai dengan jumlah epoch yaitu 20. Batch size 64 adalah jumlah sampel data yang akan di latih dalam satu iterasi.

```
# pelatihan
from tensorflow import keras # import class keras dari library tensorflow
model1.compile(loss='binary_crossentropy',optimizer=keras.optimizers.Adam(learning_rate=0.01),metrics=['accuracy'])
#loss : digunakan untuk mengatur tingkat kesalahan antara keluaran prediksi dan nilai target
#optimizers : membantu meningkatkan akurasi dan meminimalkan nilai loss / cost function.
csv_logger = CSVLogger("log.csv",separator=',', append=False)

history = model1.fit(X_train, y_train, validation_data=(X_test, y_test),batch_size=64, epochs=7,callbacks=[])
model.save("DatasetVPS/dnn223_model.hdf5")
```

Gambar 16. Proses Pelatihan

Dari 10 kali percobaan yang telah dilakukan, pelatihan pada percobaan ke 7 dihasilkan model DNN dengan 4 layer, parameter epoch 7, size 64, fungsi loss binary crossentropy dan optimizer adam mendapatkan hasil klasifikasi akurasi tertinggi dengan nilai 99,57 %

```
Epoch 1/7
13108/13108 [==============================] - 44s 3ms/step - loss: 0.0145 - accuracy: 0.9998 - val_loss: 322217.1250 - val_accuracy: 0.4206
Epoch 2/7
13108/13108 [==============================] - 43s 3ms/step - loss: 0.0111 - accuracy: 1.0000 - val_loss: 471888.7500 - val_accuracy: 0.4405
Epoch 3/7
13108/13108 [==============================] - 42s 3ms/step - loss: 0.0080 - accuracy: 1.0000 - val_loss: 1049317.7500 - val_accuracy: 0.4091
Epoch 4/7
13108/13108 [==============================] - 43s 3ms/step - loss: 4.8137e-07 - accuracy: 1.0000 - val_loss: 1053706.0000 - val_accuracy: 0.4091
Epoch 5/7
13108/13108 [==============================] - 41s 3ms/step - loss: 1.4412e-09 - accuracy: 1.0000 - val_loss: 1018218.6250 - val_accuracy: 0.4091
Epoch 6/7
13108/13108 [==============================] - 43s 3ms/step - loss: 2.9557e-12 - accuracy: 1.0000 - val_loss: 1018151.5625 - val_accuracy: 0.4091
Epoch 7/7
13108/13108 [==============================] - 43s 3ms/step - loss: 0.0594 - accuracy: 1.0000 - val_loss: 704025.6875 - val_accuracy: 0.4587
```

Gambar 17. Hasil Pelatihan berupa tingkat nilai akurasi

3.3.10. Pengujian

Pada gambar 18 merupakan proses pengujian model Deep Neural Network. Pengujian di proses dengan code model1.evaluate dengan menggunakan data testing. Hasil Pengujian dengan model DNN diperoleh akurasi tertinggi dengan nilai 45,87%.

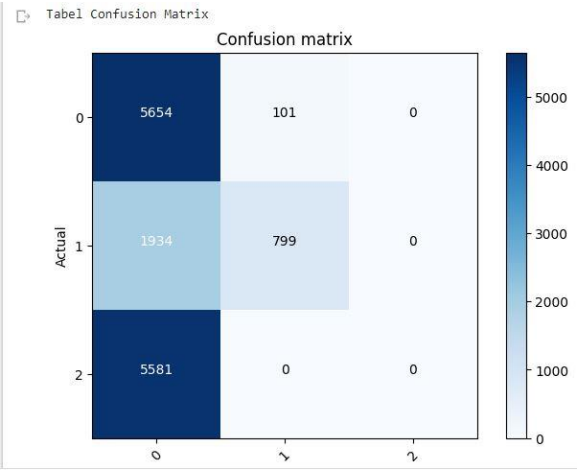
```
#pengujian
loss, accuracy = model1.evaluate(X_test, y_test) #model1.evaluate : menguji data uji dengan model yang dilatih
print("\nLoss: %.3f, Accuracy: %.2f%%" % (loss*100, accuracy*100)) #menampilkan hasil akurasi dan loss

440/440 [==============================] - 0s 1ms/step - loss: 704025.8125 - accuracy: 0.4587

Loss: 70402581.250, Accuracy: 45.87%
```

Gambar 18. Pengujian Pada Dataset VPS

Pada gambar 19 disampaikan hasil confusion matrix pendeteksian klasifikasi serangan DDoS dengan prediksi benar pada dataset VPS sebanyak 5654 data untuk kelas data normal, 799 data untuk kelas data DDoS TCP, dan, 0 data untuk kelas data UDP Flood.



Gambar 19. Hasil Confusion Matrix

4. KESIMPULAN

Berdasarkan hasil yang telah diperoleh dari percobaan mengenai proses klasifikasi serangan DDoS menggunakan metode DNN diantaranya 1. Hasil pengujian sistem menggunakan dataset serangan DDoS pada VPS yakni dengan split pembagian data 80 : 20 yaitu sebanyak 56274 data *training* dan 14069 data *testing*. Dataset diproses menggunakan Model Deep Neural Network dengan DNN 4 layer, dengan mengatur jumlah epoch 20, batch size 64, fungsi *loss binary crossentropy*, dan *optimizer adam* didapatkan tingkat akurasi pengujian menggunakan confusion matrix yaitu 96.50%. Dari hasil tersebut dapat disimpulkan penggunaan model DNN yang dirancang dikategorikan sangat baik. 2. Hasil pengujian sistem menggunakan Dataset DDoS 02-21-2018 sebagai data *training* dan dataset serangan DDoS VPS sebagai data *testing* yakni dengan split pembagian data 80:20 sebanyak 838860 data training dan 14069 data testing menggunakan Model Deep Neural Network dengan DNN 4 layer, dengan mengatur jumlah epoch 7, batch size 64, fungsi *loss binary crossentropy, dan, optimizer adam* didapatkan tingkat akurasi menggunakan confusion matrix yaitu 45.87% dapat disimpulkan hasil yang didapat masuk sebagai predikat sedang.

REFERENSI

[1] Y. Ariyanto, V. Al, H. Firdaus, and H. Pramana, "Klasifikasi Jenis serangan DOS dan Probing pada IDS menggunakan metode K- Nearest Neighbor," *Semin. Inform. Apl. Polinema 2020*, vol. 3, no. ISSN 2460-1160, pp. 1–5, 2020.

[2] R. Eka, A. Rachman, and T. Wahyu, "Virtual Private Server (VPS) Sebagai Alternatif Pengganti Dedicated Server," *Semin. Intell. Technol. Its Appl. SITIA*, pp. 2–7, 2010.

[3] R. . Vigneswaran, Vinayakumar.R, S. .KP, and P. Poornachandran, "Evaluating Shallow and Deep Neural Networks for Network Intrusion Detection Systems in Cyber Security," *Lect. Notes Data Eng. Commun. Technol.*, 2018, doi: 10.1007/978-981-16-3728-5_52.

[4] M. Z. Khairallah, "Network Attacks Detection using Deep neural network."

[5] S. Goutama *et al.*, "Simulasi Aplikasi untuk Mendeteksi dan Mencegah Serangan DDoS pada Jaringan Berbasis Software Defined Network," vol. 10, no. 1, 2022.

[6] W. I. Thenu, "DESAIN DAN IMPLEMENTASI HYBRID NETWORK INTRUSION DETECTION SYSTEM MENGGUNAKAN SNORT DAN DEEP NEURAL NETWORK PADA HOME AREA NETWORK," 2020.

[7] D. Hakim, "Optical Music Recognition Pada Citra Notasi Musik Menggunakan Convolutional Neural Network," *Elibrary.Unikom*, pp. 7–30, 2019.

[8] Certsi, "Design and Configuration of IPS , IDS and SIEM in Industrial Control Systems," p. 56, 2017, [Online]. Available: https://www.certsi.es/sites/default/files/contenidos/guias/doc/certsi_design_configuration_ips_ids_siem_in_ics.pdf.

[9] G. C. P. Amda, "PENDEKTESIAN SLOT PARKIR MENGGUNAKAN IMAGE STITCHING DAN DEEP NEURAL NETWORK," Universitas Sumatera Utara, 2019.

[10] A. Abubakar and A. B. Garko, "A Predictive Model for Network Intrusion Detection System Using Deep Neural Network," *J. Phys. Conf. Ser.*, vol. 1804, no. 1, pp. 113–128, 2021, doi: 10.1088/1742-6596/1804/1/012138.

[11] A. A. Kurniawan, "Intrusion Detection System Menggunakan Deep Learning Untuk Deteksi Serangan DoS," *Intrusion Detect. Syst. Menggunakan Deep Learn. Untuk Deteksi Serangan DoS*, pp. 39–57, 2020.

[12] M. Maithem and G. A. Al-Sultany, "Network intrusion detection system using deep neural networks," *J. Phys. Conf. Ser.*, vol. 1804, no. 1, 2021, doi: 10.1088/1742-6596/1804/1/012138.

[13] O. Prof and Z. A. Hasibuan, "Deep Learning : Concept , Model , Algorithm , and Application," 2020.

[14] M. A. Ridho and M. Arman, "Analisis Serangan DDoS Menggunakan Metode Jaringan Saraf Tiruan," *J. Sisfokom (Sistem Inf. dan Komputer)*, vol. 9, no. 3, pp. 373–379, 2020, doi: 10.32736/sisfokom.v9i3.945.

[15] M. W. Sari, "Analisis Keamanan Jaringan Virtual Private Network (VPN) pada Sistem Online Microbanking," *J. Tek. Inform.*, pp. 1–12, 2020.

[16] K. A. Fkadie, "ANOMALY- BASED INTRUSION DETECTION USING GENERATIVE ADVERSERIAL NETWORKS," 2021.

[17] H. C. Telaumbanua, "IMPLEMENTASI INTRUSION DETECTION AND PREVENTION SYSTEM MENGGUNAKAN SURICATA, BARNYARD2, DAN SNORBY PADA LINUX UBUNTU SERVER DI DISKOMINFO KABUPATEN GARUT," UNIVERSITAS KOMPUTER INDONESIA, 2020.

[18] J. H. Woo, J. Y. Song, and Y. J. Choi, "Performance Enhancement of Deep Neural Network Using Feature Selection and Preprocessing for Intrusion Detection," *1st Int. Conf. Artif. Intell. Inf. Commun. ICAIIC 2019*, pp. 415–417, 2019, doi: 10.1109/ICAIIC.2019.8668995.