TUGAS KULIAH

PEMBELAJARAN MESIN LANJUT
(EI7007)

`

EKSPLORASI HYPERPARAMETER CONVOLUTIONAL NEURAL NETWORK DAN FULLY CONNECTED NEURAL NETWORK

NAMA : RICKY ISFANDIARI
NIM : 33221041

PROGRAM STUDI DOKTOR TEKNIK ELEKTRO DAN INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

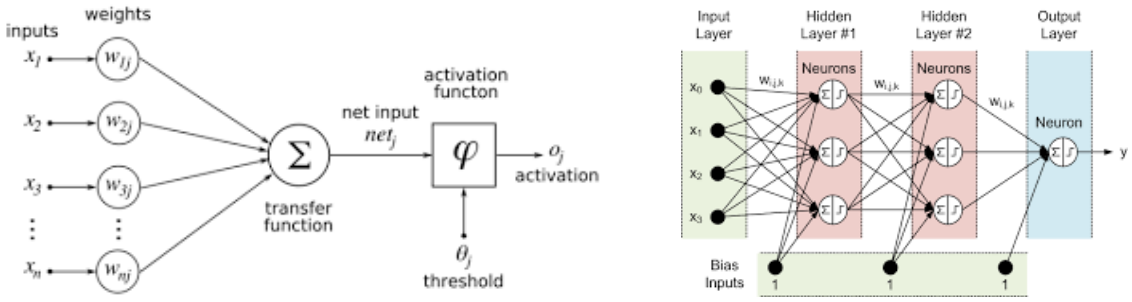
2022

EKSPLORASI HYPERPARAMETER CONVOLUTIONAL NEURAL NETWORK DAN FULLY CONNECTED NEURAL NETWORK

A. PENDAHULUAN

Artificial Neural Network Artificial (ANN) atau Jaringan Syaraf Tiruan (JST) merupakan sebuah teknik atau pendekatan pengolahan informasi yang terinspirasi oleh cara kerja sistem saraf biologis, khususnya pada sel otak manusia dalam memproses informasi. Elemen kunci dari teknik ini adalah struktur sistem pengolahan informasi yang bersifat unik dan beragam untuk tiap aplikasi. Neural Network terdiri dari sejumlah besar elemen pemrosesan informasi (neuron) yang saling terhubung dan bekerja bersama-sama untuk menyelesaikan sebuah masalah tertentu, yang pada umumnya dalah masalah klasifikasi ataupun prediksi.

Cara kerja Neural Network dapat dianalogikan sebagaimana halnya manusia belajar dengan mengunakan contoh atau yang disebut sebagai supervised learning. Sebuah Neural Network dikonfigurasi untuk aplikasi tertentu (seperti pengenalan pola, klasifikasi data, pengelompokan data, regresi data deret waktu, dan lain sebagainya) dan kemudian disempurnakan melalui proses pembelajaran. Jika proses belajar yang terjadi dalam sistem biologis melibatkan penyesuaian koneksi sinaptik yang ada antara neuron, maka pada Neural Network penyesuaian koneksi sinaptik antar neuron dilakukan dengan menyesuaikan nilai bobot yang ada pada tiap konektivitas baik dari input, neuron, maupun output. Arsitektur Neural Network secara umum dapat dilihat pada Gambar 1 di bawah ini.



Gambar 1. Arsitektur Neural Network

Neural Network memproses informasi berdasarkan cara kerja otak manusia. Dalam hal ini Neural Network terdiri dari sejumlah besar elemen pemrosesan yang saling terhubung dan bekerja secara paralel untuk memecahkan suatu masalah tertentu. Di sisi lain, komputer konvensional menggunakan pendekatan kognitif untuk memecahkan masalah; dimana cara pemecahan masalah haruslah sudah diketahui sebelumnya untuk kemudian dibuat menjadi beberapa instruksi kecil yang terstruktur. Instruksi ini kemudian dikonversi menjadi program komputer dan kemudian ke dalam kode mesin yang dapat dijalankan oleh komputer.

B. EKSPLORASI HYPERPARAMETER MODEL KLASIFIKASI MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK (CNN)

Penelitian ini akan membuat klasifikasi terhadap data gambar menggunakan metode CNN dan library Keras.

B.1. DATASET

Dataset yang digunakan adalah Fashion-MNIST yang bersumber dari Keras yaitu https://keras.io/api/datasets/fashion_mnist. Dataset tersebut berupa kumpulan data gambar yang memiliki 60.000 gambar pelatihan (training) dan 10.000 gambar pengujian (testing), di mana setiap gambar abu-abu berukuran 28x28 pixel dengan label yang terdiri dari 10 kelas. Fashion-MNIST berfungsi sebagai pengganti langsung dataset MNIST asli untuk membuat tolok ukur algoritma machine learning.

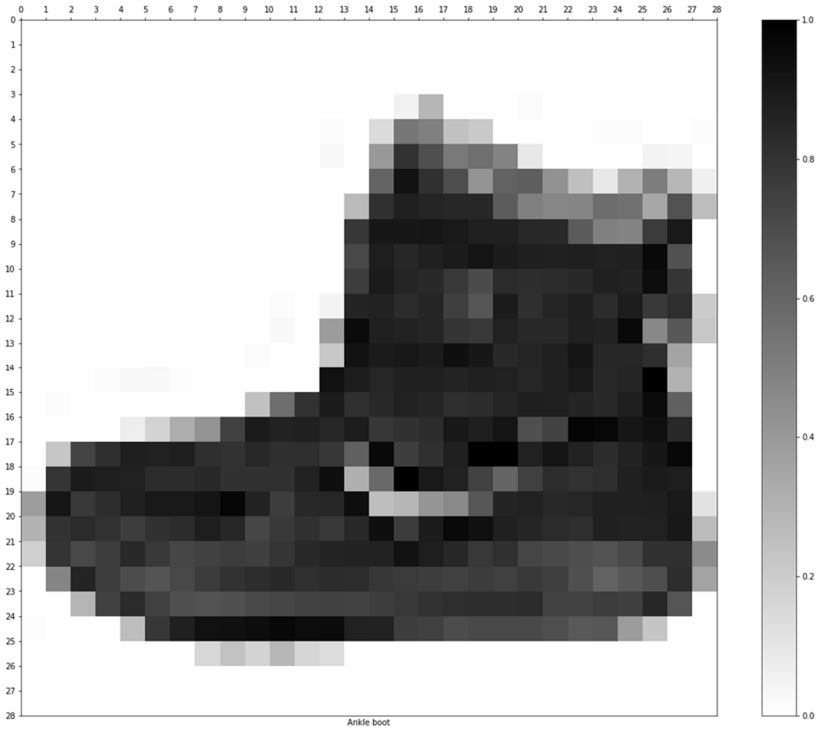
Setiap gambar memiliki panjang 28 pixel dan lebar 28 pixel, dengan total 784 pixel. Setiap pixel memiliki nilai pixel tunggal yang terkait dengannya, yang menunjukkan terang atau gelapnya pixel tersebut. Angka yang lebih tinggi mengindikasikan gambar yang lebih gelap. Nilai pixel ini adalah bilangan bulat antara 0 dan 255. Kumpulan data pelatihan dan pengujian memiliki 785 kolom. Kolom pertama terdiri dari label kelas, dan mewakili artikel pakaian. Kolom lainnya berisi nilai pixel dari gambar terkait. Sampel pada pelatihan dan pengujian tersebut memiliki label seperti yang ditunjukkan oleh Tabel 1.

Tabel 1. Label pada Dataset

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Pertama, peneliti akan mencoba untuk melihat visualisasi 20 gambar pertama dari data pelatihan. Kemudian mencoba untuk melihat visualisasi gambar dengan nilai pixel. Hasilnya seperti yang terlihat pada Gambar 2 dan Gambar 3 di bawah ini.

Gambar 2. Visualisasi 20 Gambar Pertama Dataset



Gambar 3. Visualisasi Gambar dengan Nilai Pixel

B.2. PERCOBAAN HYPERPARAMETER MODEL

Proses eksplorasi dilakukan secara eksperimen (percobaan) untuk mendapatkan nilai parameter yang bisa menghasilkan model paling optimal. Untuk mengetahui nilai yang paling optimal, harus dilakukan percobaan dengan membuat variasi nilai dari hypermarater yang sedang dieksplorasi dengan nilai hyperparameter lainnya dibuat tetap (fixed).

Beberapa pertanyaan yang mendasari untuk dilakukan eksplorasi hyperparameter adalah sebagai berikut:

1. Berapa banyaknya convolution layer yang optimal?
2. Berapa ukuran filter yang optimal untuk setiap convolution layer?
3. Berapa banyaknya filter yang optimal untuk setiap convolution layer?
4. Berapa banyaknya hidden unit yang optimal pada bagian fully connected network?
5. Dari semua pilihan optimizer yang disediakan oleh Keras, mana yang menghasilkan kinerja paling baik (pada nilai parameter default)?
6. Dari optimizer yang optimal (pada nilai parameter default), lakukan eksplorasi lebih lanjut apakah ada learning rate schedule yang menghasilkan kinerja yang lebih baik lagi?
7. Dari semua pilihan loss yang disediakan oleh Keras (probabilistik), mana yang menghasilkan kinerja paling baik?

Untuk menjawab pertanyaan-pertanyaan tersebut, peneliti mencoba melakukan berbagai percobaan terhadap hyperparameter seperti yang dtunjukkan pada Tabel 2. Model akan memisahkan data pelatihan dan data validasi. Dari dataset pelatihan yang ada, 80% akan digunakan untuk pelatihan dan 20% akan digunakan untuk validasi, kemudian mengevaluasi nilai loss dan nilai akurasi model di akhir setiap epoch.

Tabel 2. Percobaan terhadap Hyperparameter

#	Conv. Layer	Banyak Filter	Ukuran Filter	Hidden Unit	Activation Function	Optimizer Function	Learning Rate	Loss Function	Epoch	Testing Accuracy
1.	1	32	(3, 3)	128	ReLU, ReLU, Softmax	Adam	0.001	SCC *)	10	0.8979
2.	1	64	(3, 3)	128	ReLU, ReLU, Softmax	Adam	0.001	SCC	10	0.9139
3.	1	128	(3, 3)	128	ReLU, ReLU, Softmax	Adam	0.001	SCC	10	0.9142
4.	1	128	(3, 3)	256	ReLU, ReLU, Softmax	Adam	0.001	SCC	10	0.9143
5.	**1**	**128**	**(3, 3)**	**512**	**ReLU, ReLU, Softmax**	**Adam**	**0.001**	**SCC**	**10**	**0.9212**
6.	1	128	(3, 3)	1024	ReLU, ReLU, Softmax	Adam	0.001	SCC	10	0.9187
7.	1	128	(3, 3)	512	ReLU, ReLU, Softmax	Adam	0.01	SCC	10	0.9017
8.	1	128	(3, 3)	512	ReLU, ReLU, Softmax	Adam	0.005	SCC	10	0.9103
9.	1	128	(3, 3)	512	ReLU, ReLU, Softmax	Adam	0.0005	SCC	10	0.9159
10.	1	128	(3, 3)	512	ReLU, ReLU, Softmax	Adam	0.0001	SCC	10	0.8945
11.	1	128	(3, 3)	512	ReLU, ReLU, Softmax	Adam	0.001	SCC	50	0.9116
12.	1	128	(3, 3)	512	ReLU, ReLU, Softmax	Adam	0.001	SCC	100	0.9125

#	Conv. Layer	Banyak Filter	Ukuran Filter	Hidden Unit	Activation Function	Optimizer Function	Learning Rate	Loss Function	Epoch	Testing Accuracy
13.	1	128	(3, 3)	512	ReLU, ReLU, Softmax	RMSprop	0.001	SCC	10	0.9110
14.	1	128	(3, 3)	512	ReLU, ReLU, Softmax	SGD	0.001	SCC	10	0.7056
15.	1	128	(3, 3)	512	ReLU, ReLU, Softmax	Adamax	0.001	SCC	10	0.9108
16.	1	128	(3, 3)	512	ReLU, ReLU, Softmax	Nadam	0.001	SCC	10	0.9078
17.	1	128	(3, 3)	512	ReLU, ReLU, Softmax	Adagrad	0.001	SCC	10	0.7822
18.	1	128	(3, 3)	512	ReLU, ReLU, Softmax	Adadelta	0.001	SCC	10	0.6511
19.	1	128	(3, 3)	512	Sigmoid, ReLU, Softmax	Adam	0.001	SCC	10	0.8108
20.	1	128	(3, 3)	512	TanH, ReLU, Softmax	Adam	0.001	SCC	10	0.9122
21.	1	128	(3, 3)	512	TanH, TanH, Softmax	Adam	0.001	SCC	10	0.9082
22.	1	128	(3, 3)	512	SELU, SELU, Softmax	Adam	0.001	SCC	10	0.9013
23.	1	128	(3, 3)	512	ELU, ELU, Softmax	Adam	0.001	SCC	10	0.9049
24.	1	128	(3, 3)	512	Softsign, Softsign, Softmax	Adam	0.001	SCC	10	0.9107

Catatan:
*) *SCC merupakan singkatan dari Sparse Categorical Crossentropy.*

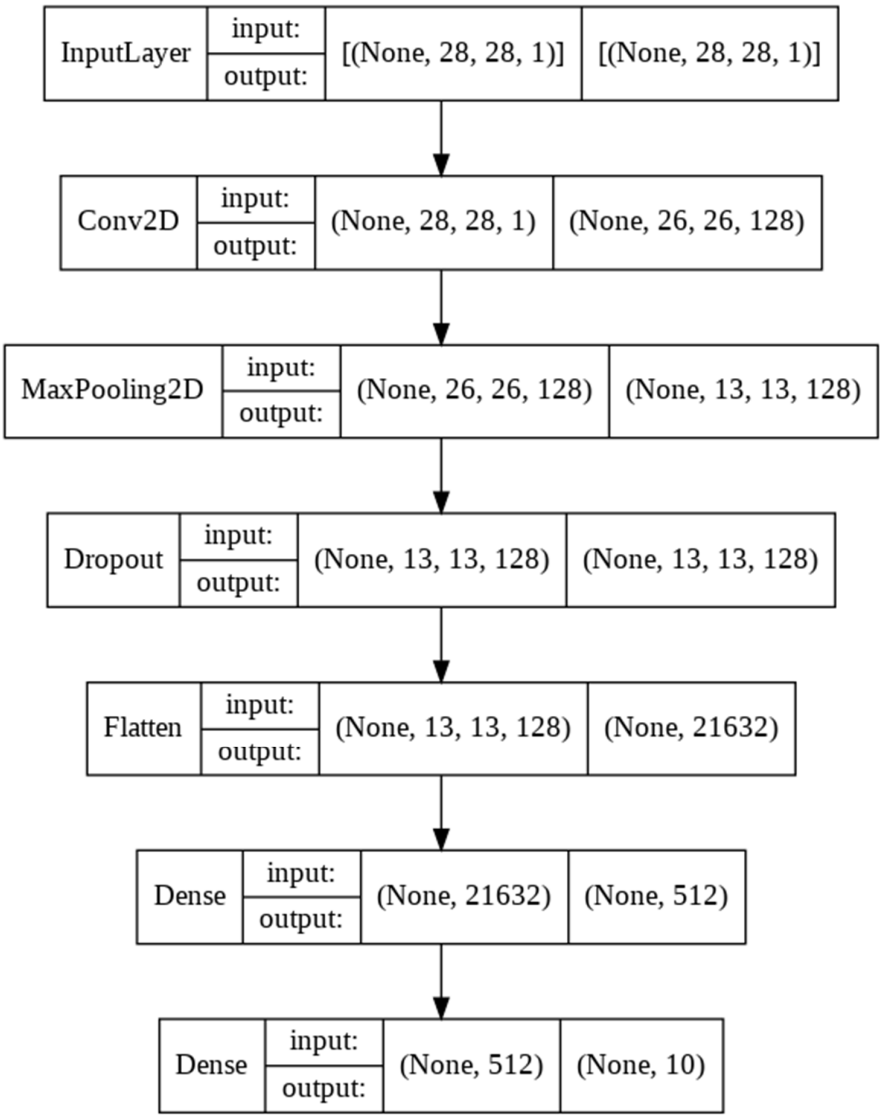
B.3. HASIL

Berdasarkan hasil eksplorasi terhadap hyperparameter (Tabel 2), didapatkan bahwa model menghasilkan kondisi yang optimal pada percobaan nomor 5. Summary dan ilustasi dari arsitektur model yang optimal ditunjukkan oleh Gambar 4 dan Gambar 5 di bawah ini.

```
# Ringkasan struktur model.
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 128)       1280

 max_pooling2d (MaxPooling2D  (None, 13, 13, 128)      0
 )

 dropout (Dropout)           (None, 13, 13, 128)       0

 flatten (Flatten)           (None, 21632)             0

 dense (Dense)               (None, 512)               11076096

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 11,082,506
Trainable params: 11,082,506
Non-trainable params: 0
_________________________________________________________________
```

Gambar 4. Summary Arsitektur Model Optimal



Gambar 5. Ilustrasi Arsitektur Model Optimal

Epoch yang digunakan berjumlah 10. Untuk hasil iterasi setiap epoch dapat dilihat seperti pada Gambar 6. Ringkasan metrik hasil pengujian dapat dilihat pada Gambar 7.

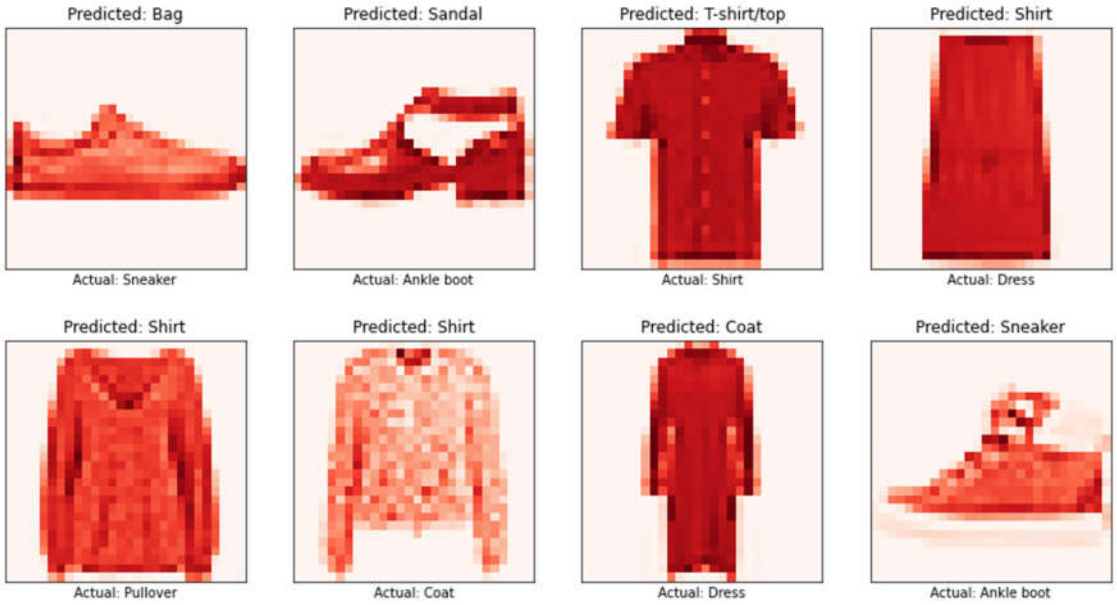
```
Epoch 1/10
188/188 [==============================] - 136s 721ms/step - loss: 0.4314 - accuracy: 0.8470 - val_loss: 0.3219 - val_accuracy: 0.8836
Epoch 2/10
188/188 [==============================] - 139s 741ms/step - loss: 0.2801 - accuracy: 0.9003 - val_loss: 0.2888 - val_accuracy: 0.8953
Epoch 3/10
188/188 [==============================] - 134s 715ms/step - loss: 0.2364 - accuracy: 0.9143 - val_loss: 0.2858 - val_accuracy: 0.8962
Epoch 4/10
188/188 [==============================] - 134s 714ms/step - loss: 0.2013 - accuracy: 0.9263 - val_loss: 0.2532 - val_accuracy: 0.9103
Epoch 5/10
188/188 [==============================] - 134s 711ms/step - loss: 0.1747 - accuracy: 0.9377 - val_loss: 0.2288 - val_accuracy: 0.9194
Epoch 6/10
188/188 [==============================] - 133s 710ms/step - loss: 0.1537 - accuracy: 0.9440 - val_loss: 0.2324 - val_accuracy: 0.9185
Epoch 7/10
188/188 [==============================] - 133s 709ms/step - loss: 0.1351 - accuracy: 0.9502 - val_loss: 0.2588 - val_accuracy: 0.9129
Epoch 8/10
188/188 [==============================] - 133s 709ms/step - loss: 0.1180 - accuracy: 0.9564 - val_loss: 0.2617 - val_accuracy: 0.9141
Epoch 9/10
188/188 [==============================] - 133s 710ms/step - loss: 0.1036 - accuracy: 0.9623 - val_loss: 0.2427 - val_accuracy: 0.9212
Epoch 10/10
188/188 [==============================] - 133s 706ms/step - loss: 0.0898 - accuracy: 0.9680 - val_loss: 0.2482 - val_accuracy: 0.9221
```

Gambar 6. Hasil Iterasi Setiap Epoch

	precision	recall	f1-score	support
T-shirt/top	0.9020	0.8560	0.8784	1000
Trouser	0.9910	0.9880	0.9895	1000
Pullover	0.9097	0.8560	0.8820	1000
Dress	0.9234	0.9160	0.9197	1000
Coat	0.8407	0.9080	0.8731	1000
Sandal	0.9908	0.9690	0.9798	1000
Shirt	0.7714	0.7930	0.7821	1000
Sneaker	0.9575	0.9680	0.9627	1000
Bag	0.9752	0.9840	0.9796	1000
Ankle boot	0.9596	0.9740	0.9667	1000
accuracy			0.9212	10000
macro avg	0.9221	0.9212	0.9214	10000
weighted avg	0.9221	0.9212	0.9214	10000

Gambar 7. Ringkasan Metrik Hasil Pengujian

Untuk melihat hasil evaluasi lebih lanjut, peneliti mencoba untuk melihat data pengujian yang salah diklasifikasikan. Sebagai contoh, 8 data gambar pertama yang salah diklasifikasikan dari data pengujian dapat dilihat pada Gambar 8 di bawah ini.



Gambar 8. Beberapa Gambar yang Salah Klasifikasi

C. EKSPLORASI HYPERPARAMETER MODEL REGRESI MENGGUNAKAN FULLY CONNECTED NEURAL NETWORK (FCNN)

Penelitian ini akan memprediksi harga rata-rata rumah di pinggiran kota Boston tertentu pada pertengahan 1970-an menggunakan metode fully connected neural network dan library Keras.

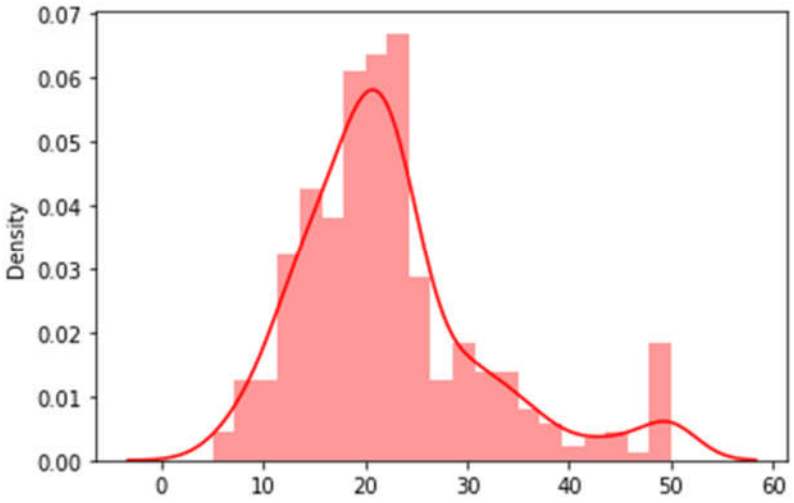
Beberapa faktor yang mempengaruhi harga rata-rata rrumah di pinggiran kota pada saat itu adalah tingkat kejahatan, tarif pajak properti lokal, dan lain sebagainya.

C.1. DATASET

Dataset yang digunakan adalah Boston Housing Price yang bersumber dari Keras yaitu https://keras.io/api/datasets/boston_housing. Dataset tersebut memiliki total series 506, dan terbagi menjadi 404 sampel pelatihan dan 102 sampel pengujian. Setiap fitur dalam data input (seperti tingkat kejahatan) memiliki skala yang berbeda-beda. Misalnya proporsi, yang mengambil nilai antara 0 dan 1, ada juga yang mengambil nilai antara 1 dan 12, dan ada juga yang mengambil nilai antara 0 dan 100. Dataset ini memiliki 13 fitur, yaitu:

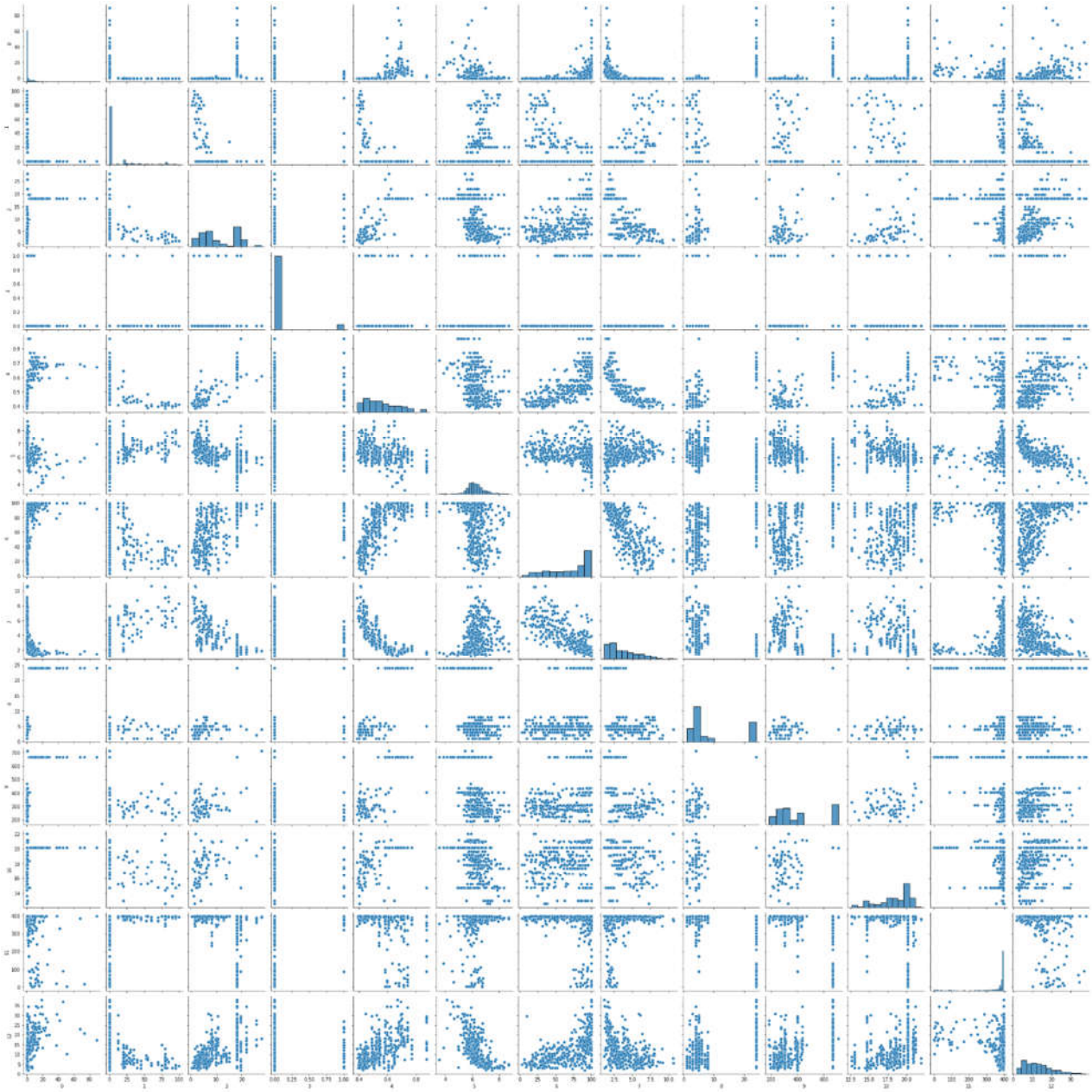
1. Tingkat kejahatan per kapita.
2. Proporsi lahan perumahan yang dikategorikan untuk kavling lebih dari 25.000 kaki persegi.
3. Proporsi hektar bisnis non-ritel per kota.
4. Variabel dummy Sungai Charles (= 1 jika saluran membatasi sungai; 0 sebaliknya).
5. Konsentrasi oksida nitrat (bagian per 10 juta).
6. Rata-rata jumlah kamar per hunian.
7. Proporsi unit yang ditempati oleh pemilik yang dibangun sebelum tahun 1940.
8. Jarak tertimbang ke lima pusat pekerjaan Boston.
9. Indeks aksesibilitas ke jalan raya radial.
10. Tarif pajak properti nilai penuh per $10.000.
11. Rasio murid-guru menurut kota.
12. 1000 * (Bk - 0,63) ** 2 yang mana Bk adalah proporsi orang kulit hitam menurut kota.
13. Persentase (%) status penduduk yang lebih rendah.

Awalnya fokus terlebih dahulu pada vaiabel terikatnya, yaitu variabel harga rumah yang juga jenis variabel kontinyu. Sifat variabel terikat sangat penting untuk pemilihan model, untuk itu kita dapat memahaminya dengan melihat histogram distribusinya terlebih dahulu (Gambar 9). Dari plot histogram tersebut dapat diketahui bahwa sebaran data variabel terikatnya membentuk kurva distribusi normal.



Gambar 9. Histogram Variabel Terikat

Kemudian dilihat sebaran variabel bebasnya sebanyak 13 variabel. Sebaran dan histogram data masing-masing fitur tersebut dapat dilihat pada Gambar 10. Grafik histogramnya terdapat pada bagian diagonal gambar tersebut. Kemudian dilakukan normalisasi fitur. Untuk setiap fitur dalam data input, peneliti mengurangi rata-rata fitur dan membaginya dengan standar deviasi, sehingga fitur akan berpusat di sekitar 0 dan memiliki simpangan baku yang standar.



Gambar 10. Plot dan Histogram Variabel Bebas

C.2. PERCOBAAN HYPERPARAMETER MODEL

Karena sampel tergolong kecil, peneliti menggunakan network yang kecil dengan 2 hidden layer, masing-masing sebanyak 128 dense unit (kondisi optimal). Umumnya, semakin sedikit

data pelatihan maka akan semakin overfitting. Untuk itu, menggunakan network kecil merupakan salah satu cara untuk mengurangi overfitting.

Proses eksplorasi dilakukan secara eksperimen (percobaan) untuk mendapatkan nilai parameter yang bisa menghasilkan model paling optimal. Untuk mengetahui nilai yang paling optimal, harus dilakukan percobaan dengan membuat variasi nilai dari hypermarater yang sedang dieksplorasi dengan nilai hyperparameter lainnya dibuat tetap (fixed).

Beberapa pertanyaan yang mendasari untuk dilakukan eksplorasi hyperparameter adalah sebagai berikut:

1. Berapa banyaknya hidden layar yang optimal?
2. Berapa banykanya hidden unit yang optimal di setiap hidden layar?
3. Apa activation function di setiap layer sehingga hasilnya optimal?
4. Dari semua pilihan optimizer, apa optimizer yang hasilnya optimal?
5. Dari semua pilihan loss function, apa yang hasilnya optimal?

Untuk menjawab pertanyaan-pertanyaan tersebut, peneliti mencoba melakukan berbagai percobaan terhadap hyperparameter seperti yang dtunjukkan pada Tabel 3.

Tabel 3. Percobaan terhadap Hyperparameter

#	Hidden Layer	Hidden Unit	K-Fold (Validasi)	Epoch	Activation Function	Optimizer	Akurasi (MAE) Training	Akurasi (MAE) Testing	Loss Function (MSE) Testing
1.	2	32	4	100	ReLU	RMSprop	2.8060	2.5351	17.9388
2.	2	32	5	100	ReLU	RMSprop	2.7969	2.6935	16.3910
3.	2	64	4	100	ReLU	RMSprop	2.5824	2.4809	16.6616
4.	2	64	5	100	ReLU	RMSprop	2.5635	2.6792	15.2886
5.	2	128	4	100	ReLU	RMSprop	2.4763	2.3946	11.9718
6.	**2**	**128**	**5**	**100**	**ReLU**	**RMSprop**	**2.4926**	**2.1979**	**9.9351**
7.	3	32	4	100	ReLU	RMSprop	2.6382	2.3786	14.0495
8.	3	32	5	100	ReLU	RMSprop	2.5899	2.4284	11.5544
9.	3	64	4	100	ReLU	RMSprop	2.3536	2.6404	11.1439
10.	3	64	5	100	ReLU	RMSprop	2.4745	2.6389	16.8576
11.	3	128	4	100	ReLU	RMSprop	2.5411	2.6784	14.0934
12.	3	128	5	100	ReLU	RMSprop	2.5129	2.6990	12.9168
13.	2	32	4	500	ReLU	RMSprop	2.4962	2.6498	12.6594
14.	2	32	5	500	ReLU	RMSprop	2.4810	2.5407	11.9042
15.	2	64	4	500	ReLU	RMSprop	2.6028	2.6005	16.0494
16.	2	64	5	500	ReLU	RMSprop	2.5988	2.8673	20.3837
17.	2	128	4	500	ReLU	RMSprop	2.5188	2.2670	10.1312
18.	2	128	5	500	ReLU	RMSprop	2.5248	2.4395	14.1261
19.	3	32	4	500	ReLU	RMSprop	2.5798	2.5481	11.6921
20.	3	32	5	500	ReLU	RMSprop	2.5467	2.4539	11.7663
21.	3	64	4	500	ReLU	RMSprop	2.4946	2.4680	15.8437
22.	3	64	5	500	ReLU	RMSprop	2.4816	2.3241	11.7026
23.	3	128	4	500	ReLU	RMSprop	2.4726	2.5244	11.9146
24.	3	128	5	500	ReLU	RMSprop	2.4093	2.2654	11.9363

C.3. HASIL

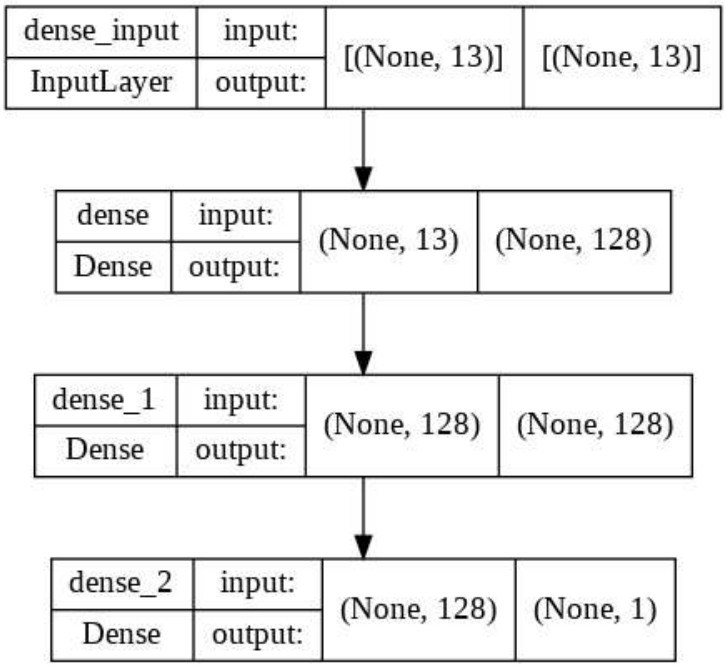
Berdasarkan hasil eksplorasi terhadap hyperparameter (Tabel 3), didapatkan bahwa model menghasilkan kondisi yang optimal pada percobaan nomor 6. Summary dan ilustrasi dari arsitektur model yang optimal ditunjukkan oleh Gambar 11 dan Gambar 12 di bawah ini.

```
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               1792

 dense_1 (Dense)             (None, 128)               16512

 dense_2 (Dense)             (None, 1)                 129

=================================================================
Total params: 18,433
Trainable params: 18,433
Non-trainable params: 0
_________________________________________________________________
None
```

Gambar 11. Summary Arsitektur Model Optimal

```
from keras.utils.vis_utils import plot_model
plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```

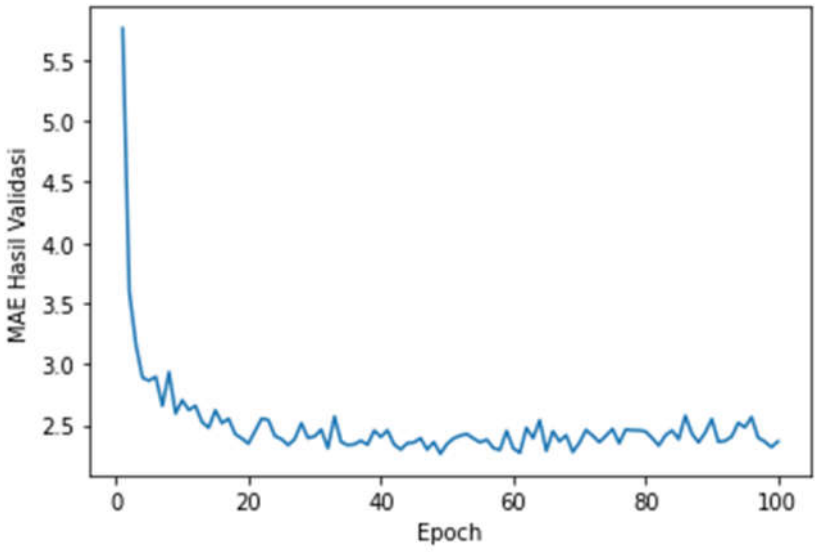


Gambar 12. Ilustrasi Arsitektur Model Optimal

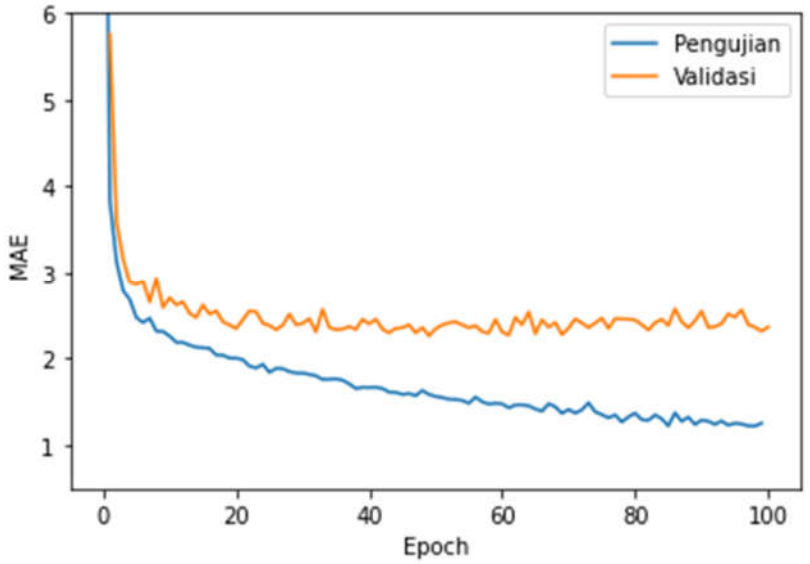
Untuk source code yang digunakan dalam membuat model yang optimal ditampilkan oleh Gambar 13. Grafik MAE untuk validasi data pelatihannya dengan jumlah epoch 100 ditunjukkan oleh Gambar 14. Grafik perbandingan MAE untuk data pengujian versus data validasi dengan jumlah epoch 100 ditunjukkan oleh Gambar 15. Plot perbandingan nilai riil versus nilai prediksi ditunjukkan oleh Gambar 16.

```
# Peneliti menggunakan fungsi dari library Keras untuk membangun model.
# Optimizer yang digunakan adalah RMSprop dari library Keras.
from keras import models
from keras import layers

def build_model():
    model = models.Sequential()
    model.add(layers.Dense(128, activation='relu',
                           input_shape=(train_data.shape[1],)))
    model.add(layers.Dense(128, activation='relu'))
    model.add(layers.Dense(1))
    model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
    return model
```

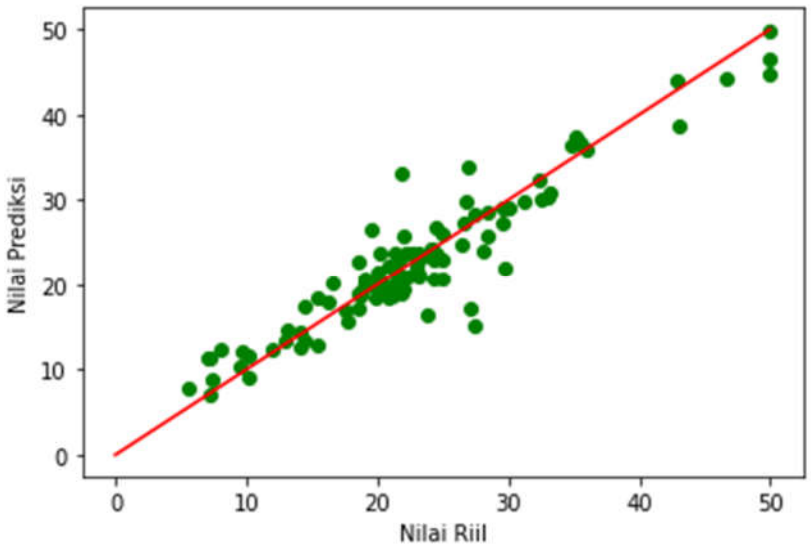
Gambar 13. Source Code Model yang Optimal



Gambar 14. Grafik MAE Hasil Validasi Data Pelatihan



Gambar 15. Grafik MAE Data Pengujian Versus Data Validasi



Gambar 16. Plot Nilai Riil Versus Nilai Prediksi

DAFTAR REFERENSI

1. Materi Kuliah: https://stei19.kuliah.itb.ac.id/course/view.php?id=840
2. https://keras.io
3. https://www.tensorflow.org
4. http://introtodeeplearning.com