PEMBELAJARAN MESIN LANJUT

(EL-7007)

Tugas Mandiri

Oleh

YULRIO BRIANORMAN

NIM: 33221012

(Program Studi Doktor Teknik Elektro dan Informatika)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

Maret 2022

Kata Pengantar

Pertama-tama penyusun panjatkan puji dan syukur atas rahmat dan ridho Allah SWT, karena tanpa rahmat dan ridhoNya, penyusun tidak akan dapat menyelesaikan tugas mandiri ini dengan baik dan selesai tepat waktu.

Tidak lupa penyusun ucapakan terima kasih kepada Prof. Ir. Dwi Hendratmo Widyantoro, M.Sc., Ph.D. selaku dosen pengampu mata kuliah Pembelajaran Mesin Lanjut (EL-7007) yang telah memberi arahan kepada penyusun untuk menyelesaikan tugas mandiri ini.

Bila terdapat kesalahan yang belum penyusun ketahui pada tugas mandiri ini, mohon kiranya saran dan kritik dari para pembaca.

Wassalamu'alaikum. Wr. Wb.

Bandung, Maret 2022

Penyusun

Daftar Isi

Daftar Tabel

Daftar Gambar

Daftar Kode Program

Tugas 1
Eksplorasi Hyperparameter CNN dan Neural Network

1.1 Pendahuluan

Persoalan penentuan arsitektur neural network yang digunakan pada proses klasifikasi merupakan permasalahan yang dinilai penting untuk dilakukan eksplorasi. Aristektur neural network yang dirancang haruslah memberikan hasil yang optimal baik dari efisiensi waktu komputasi yang diperlukan dan juga efektifitas hasil akurasi yang didapat.

Pada eksplorasi ini akan menggunakan data CIFAR10 yang telah disediakan oleh modul *tf.keras.datasets*. Pengukuran kinerja model ditentukan dengan menggunakan kumpulan tes data yang telah disedikan. Adapun kode program *digit recognition* digunakan sebagai kode program dasar untuk proses eksplorasi.

1.2 Rumusan Masalah

Adapun rumusan masalah yang ditentukan pada proses eksplorasi ini adalah:

1. Berapa banyaknya hidden unit yang optimal pada bagian *fully connected network*?
2. Berapa banyaknya *layer convolution* yang optimal?
3. Berapa ukuran filter yang optimal untuk setiap *layer convolution*?
4. Berapa banyaknya filter yang optimal untuk setiap *layer convolution*?

1.3 Panduan Eksplorasi

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). Jika ada nilai hyperparameter lainnya yang sudah ditemukan pada eksplorasi sebelumnya, gunakan nilai hyperparameter optimal tsb pada eksplorasi berikutnya Nilai optimal diambil dari percobaan yang menghasilkan kinerja terbaik.

Proses eksplorasi lanjut yang lakukan adalah eksplorasi pada *learning rate schedule* dan *probabilistic losses* untuk menghasilkan kinerja paling baik pada persoalan ini.

1.4 Tahapan Eksplorasi

Eksplorasi akan dilakukan 4 tahap yaitu:

1. Tahap 1 adalah pencarian arsitektur yang optimal pada *fully connected*.
2. Tahap 2 adalah pencarian arsitektur CNN yang optimal dengan menggunakan arsitektur hasil tahap 1 pada bagaian *fully connected*.
3. Tahap 3 adalah pencarian *optimizer* dan nilai *learning rate* yang optimal dengan menggunakan arsitektur dari tahap 3.
4. Tahap 4 adalah pencarian *probabilistic losses* yang menghasilkan kinerja paling baik dengan menggunakan arsitektur dari tahap 3.

Diharapkan dari setelah melakukan semua tahap maka akan dihasilkan sebuah arsitektur yang optimal untuk proses klasifikasi gambar pada kumpulan data CIFAR10.

1.5 Proses Eksplorasi

1.5.1 Kumpulan Data

Kumpulan data yang digunakan adalah CIFAR10, seperti yang terlihat pada Gambar 1. Kumpulan data CIFAR-10 adalah kumpulan data standar yang digunakan dalam *computer vision* dan komunitas *deep learning*. Kumpulan data ini terdiri dari 60.000 gambar berwarna dengen ukuran 32x32 dan terbagi dalam 10 kelas. Setiap kelas terdiri dari 6.000 gambar. Pembagian gambar untuk proses pembelajaran adalah 50.000 gambar untuk data latih dan 10.000 gambar untuk data uji.

Gambar 1. Kumpulan data CIFAR 10

1.5.2 Tahap 1

Pada tahap ini akan lakukan pengujian proses pembelajaran dengan berbagai arsitektur yang dirancang. Arsitektur yang dirancang hanya sebatas fully connected, dalam arti tidak menggunakan layer convolution. Arsitekur yang diuji terlihat pada Tabel 1. Pada tabel diperlihatkan jumlah dense pada setiap hidden layer.

Tabel 1. Daftar arsitektur yang diuji

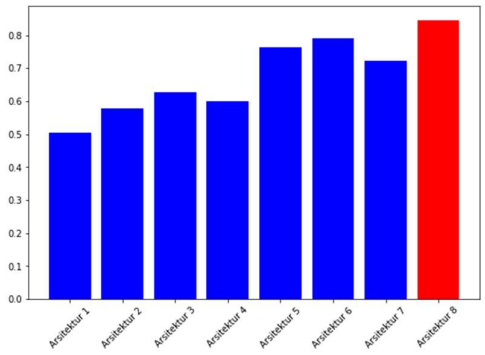
No	Arsitektur	Layer 1	Layer 2	Layer 3
1	Arsitektur 1	128	-	-
2	Arsitektur 2	256		
3	Arsitektur 3	512		
4	Arsitektur 4	128	64	
5	Arsitektur 5	256	128	
6	Arsitektur 6	512	256	
7	Arsitektur 7	256	128	64
8	Arsitektur 8	512	256	128

Setelah melakukan proses pembelajaran maka didapat hasil seperti yang ditunjukkan pada Tabel 2. Pada tabel menunjukkan bahwa akurasi tertinggi pada data latih diperoleh pada artistektur 8, sedangkan pada akurasi tertinggi pada data uji diperoleh pada arsitektur 7. Namun, untuk pengujian hasil pembelajaran pada 15 gambar secara acak diperoleh hasil tertinggi pada arsitektur 2.

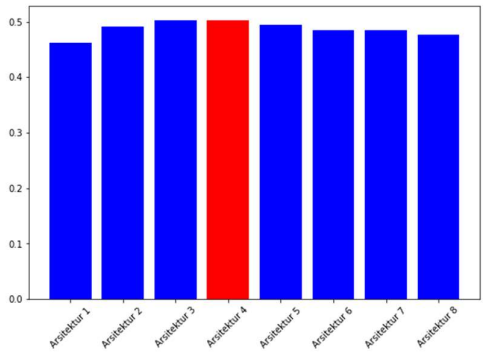
Tabel 2. Hasil pembelajaran

No	Arsitektur	Loss	Accuracy	Val. Loss	Val. Accuracy	Cek Acak
1	Arsitektur 1	1.3962	0.5040	1.5389	0.4626	9
2	Arsitektur 2	1.1683	0.5791	1.4926	0.4911	8
3	Arsitektur 3	1.0551	0.6263	1.5596	0.5019	7
4	Arsitektur 4	1.1178	0.5994	1.4892	0.5033	6
5	Arsitektur 5	0.6528	0.7648	2.1458	0.4949	8
6	Arsitektur 6	0.5826	0.7914	2.3292	0.4849	6
7	Arsitektur 7	0.7630	0.7235	1.9522	0.4849	10
8	Arsitektur 8	0.4205	0.8465	3.1541	0.4765	8

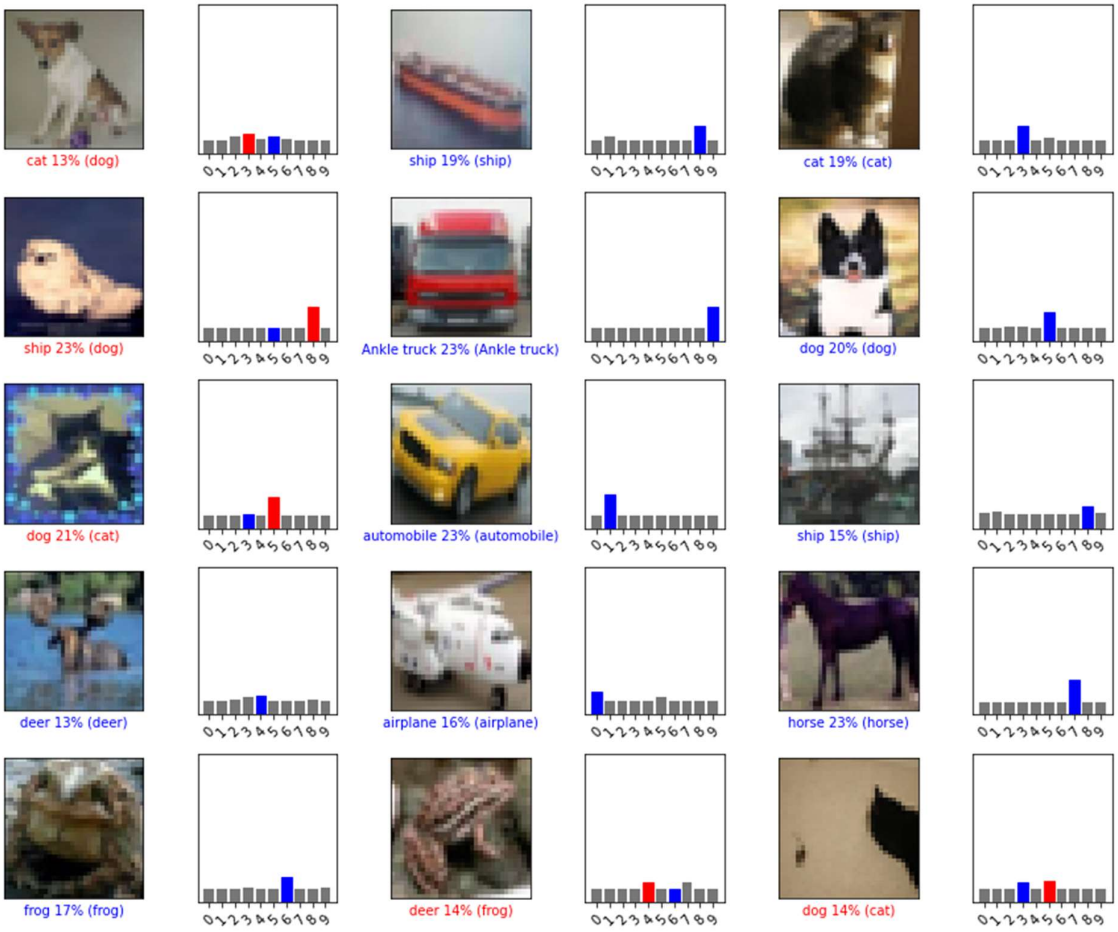
Pada Gambar 2 menunjukan grafik akurasi pada data latih dan Gambar 3 menunjukkan grafik akurasi pada data uji. Sedangkan pada Gambar 4 menunjukkan hasil pengujian pada model yang dihasilkan terhadap gambar secara acak. Pada pengujian ini menunjukkan pada arsitektur 7 memperoleh hasil paling baik. Namun, hal ini tidak menunjukkan tingkat akurasi pada hasil model lebih baik dari arsitektur lainnya, dikarenakan gambar yang diuji dilakukan secara acak.



Gambar 2. Grafik akurasi pada data latih



Gambar 3. Grafik akurasi pada data uji



Gambar 4. Pengujian model hasil pelatihan pada data acak dari arsitektur 7.

1.5.3 Tahap 2

Pada tahap 2 ini akan dicari arsitektur convolution layer yang dinilai optimal untuk ditambahkan pada aristektur 8. Jumlah convolution layer pada eksplorasi ini dibatasi 2 layer

saja. Hal ini disebabkan besar ukuran gambar yang relatif kecil yaitu 32x32 pixel. Arsitektur yang digunakan untuk proses pembelajaran seperti terlihat pada Tabel 3.

Tabel 3. Daftar arsitektur yang diuji menggunakan CNN layer

No	Arsitektur	CNN Layer 1	CNN Layer 2
1	Arsitektur 9	64	-
2	Arsitektur 10	128	-
3	Arsitektur 11	128	64

Hasil dari proses pembelajaran arsitektur 8 ditambah dengan convolution layer terlihat pada Tabel 4. Walaupun pada accuracy arsitektur 2 paling baik diantara arsitektur lainnya namun arsitektur 11 lebih baik pada validation accuracy dengan nilai yang cukup jauh yaitu 0.05. Sementara itu selisih pada nilai accuracy selisih yang didapat tidaklah terlalu jauh yakni sekitar 0.004 saja. Berdasarkan hasil ini maka diambil arsitektur yang terbaik adalah arsitektur 11.

Tabel 4. Hasil pembelajaran dengan convolution layer

No	Arsitektur	Loss	Accuracy	Val. Loss	Val. Accuracy	Cek Acak
1	Arsitektur 9	0.0184	0.9948	3.3159	0.6424	6
2	Arsitektur 10	0.0131	0.9964	3.3502	0.6547	6
3	Arsitektur 11	0.0132	0.9960	2.7532	0.7074	7

Berikut ini kode program arsitektur 11 yang merupakan gabungan convulotion layer dan fully connected layer.

Kode Program 1. Arsitektur 11 yang merupakan arsitektur terbaik pada proses eksplorasi

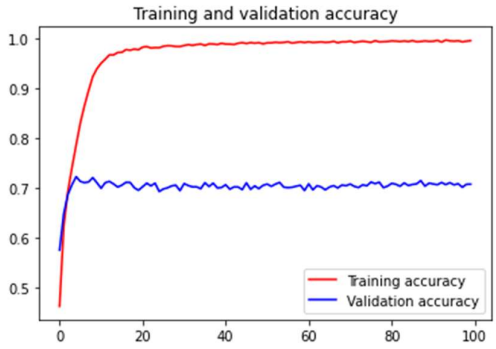
```
model = tf.keras.Sequential([
    # The first convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu', input_shape=(32, 32, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(32, 32, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10,activation='softmax')
])
```

Pada Gambar 5 adalah summary dari arsitektur yang dirancang.

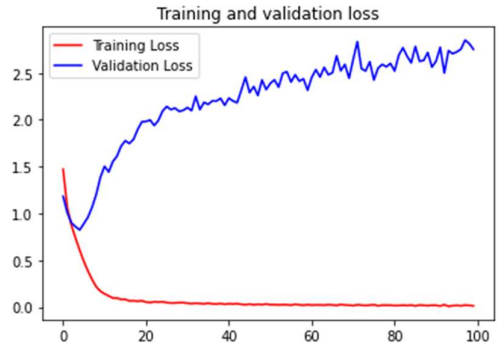
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 128)       3584

 max_pooling2d (MaxPooling2D  (None, 15, 15, 128)      0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        73792

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 flatten (Flatten)           (None, 2304)              0

 dense (Dense)               (None, 512)               1180160

 dense_1 (Dense)             (None, 256)               131328

 dense_2 (Dense)             (None, 128)               32896

 dense_3 (Dense)             (None, 10)                1290

=================================================================
Total params: 1,423,050
Trainable params: 1,423,050
Non-trainable params: 0
_________________________________________________________________
```

Gambar 5. Summary dari arsitektur yang dirancang

Hasil pembelajaran dari arsitektur 11 terlihat pada Gambar 6 dan Gambar 7. Pada Gambar 6 menunjukkan grafik akurasi pada proses pembelajaran. Garis merah menunjukkan akurasi pada data latih dan garis biru menunjukkan akurasi pada data uji. Pada Gambar 7 menunjukkan grafik loss pada proses pembelajaran. Garis merah menunjukkan loss pada data latih dan garis biru menunjukkan loss pada data uji.



Gambar 6. Grafik akurasi pada data latih dan uji



Gambar 7. Grafik loss pada data latih dan uji

1.5.4 Tahap 3

Pada tahap 3 akan ditentukan optimizer yang dinilai paling optimal. Langkah pertama pada tahap ini adalah menentukan learning rate yang pengujian. Nilai learning rate yang akan diuji adalah 0.001, 0.0007, 0.0005, 0.0003 dan 0.0001 dengan menggunakan optimizer Adam. Proses pembelajaran akan berhenti ketika mendapatkan nilai akurasi sebesar 99.5%. Nilai learning rate yang mencapai angka tersebut akan dilihat jumlah epoch yang diperlukan. Jumlah

epoch terkecil akan dianggap sebagai learning rate yang paling optimal. Hasil dari proses pengujian terlihat pada Tabel 5.

Tabel 5. Jumlah epoch yang diperlukan dari setiap nilai LR.

No	Arsitektur	Nilai LR	Epoch
1	Arsitektur 11	0.001	68
2	Arsitektur 11	0.0007	72
3	Arsitektur 11	0.0005	63
4	Arsitektur 11	0.0003	51
5	Arsitektur 11	0,0001	49

Langkah kedua adalah menggunakan optimizer yang berbeda. Optimizer yang digunakan adalah SGD, Adam, Adadelta, Adagrad dan RMSProp. Nilai LR yang digunakan adalah 0.0001. Untuk penentuannya optimizer yang optimal digunakan metode yang sama pada saat penentuan nilai learning rate yaitu jumlah epoch terkecil untuk mencapai akurasi 99.5%. Hasil dari proses pengujian terlihat pada Tabel 6. Pada pengujian ini 3 jenis optimizer tidak mencapai akurasi 99.5% dengan learning rate 0.0001 dan 100 epoch, hanya 2 optimizer yang berhasil mencapainya yaitu Adam dan RMSProp. Adam mencapainya pada epoch ke-49 sedangakan RMSprop pada epoch ke-57.

Tabel 6. Jumlah epoch yang diperlukan dari setiap optimizer dengan nilai LR = 0.0001

No	Arsitektur	Optimizer	Epoch
1	Arsitektur 11	SGD	-
2	Arsitektur 11	Adam	49
3	Arsitektur 11	Adadelta	-
4	Arsitektur 11	Adagrad	-
5	Arsitektur 11	RMSProp	57

Untuk optimizer yang tidak mencapai nilai akurasi yang diharapakan maka pelatihan diulangi dengan menggunakan learning rate = 0.001 untuk mencapai akurasi yang dinginkan maka diperlukan lebih dari 100 epoch. Hasil yang didapat seperti terlihat pada Tabel 7.

Tabel 7. Jumlah epoch yang diperlukan dari setiap optimizer dengan nilai LR = 0.001

No	Arsitektur	Optimizer	Epoch
1	Arsitektur 11	SGD	432
2	Arsitektur 11	Adadelta	>500
3	Arsitektur 11	Adagrad	>500

Pada tahap 3 ini dapat disimpulkan bahwa optimizer dan nilai learning rate yang optimal adalah

optimizer Adam dengan nilai learning rate = 0.0001. Kode yang digunakan terlihat pada Kode Program 2.

Kode Program 2. Optimizer dan nilai learning rate yang optimal.

```
model.compile(loss = 'categorical_crossentropy',
                  optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
                  metrics=['accuracy'])
```

1.5.5 Tahap 4

Pada tahap 4 akan ditentukan tipe loss yang dinilai paling optimal. Langkah yang dilakukan adalah menguji berbagai tipe loss. Tipe loss yang akan diuji adalah Categorical Crossentropy, Categorical Hinge, BinaryCrossentropy, MeanAbsoluteError dan Mean Squared Error. Pengujian ini menggunakan parameter optimizer Adam dengan nilai learning rate = 0.0001. Proses pembelajaran akan berhenti ketika mendapatkan nilai akurasi sebesar 99.5%. Hasil dari pembelajaran dapat dilihat pada Tabel 8 dan menunjukkan bahwa Categorical Crossentropy adalah yang paling optimal. Kode yang digunakan terlihat pada Kode Program 2.

Tabel 8. Jumlah epoch yang diperlukan pada setiap tipe loss yang diuji

No	Arsitektur	Loss	Epoch
1	Arsitektur 11	Categorical Crossentropy	49
2	Arsitektur 11	Categorical Hinge	>100
3	Arsitektur 11	Binary Crossentropy	53
4	Arsitektur 11	Mean Absolute Error	>100
5	Arsitektur 11	Mean Squared Error	>100

1.6 Kesimpulan

Setelah melakukan 4 tahap eksplorasi maka dapat disimpulkan arsitektur yang digunakan adalah arsitektur 11 yang dapat dilihat pada Kode Program 1. *Optimizer* yang digunakan adalah Adam dengan nilai *learning rate* yang digunakan sebesar 0.0001. Untuk tipe *loss* yang digunakan adalah *Categorical Crossentropy*. Ketiga parameter ini dapat dilihat pada Kode Program 2. Kode lengkap dapat dilihat pada url: https://bit.ly/36Cu80C.

Tugas 2
Persoalan Linear

2.1 Pendahuluan

Persoalan penentuan arsitektur neural network yang digunakan pada persoalan linear merupakan permasalahan yang dinilai penting untuk dilakukan eksplorasi. Aristektur neural network yang dirancang haruslah memberikan hasil yang optimal baik dari efisiensi waktu komputasi yang diperlukan dan juga efektifitas hasil akurasi yang didapat.

Pada eksplorasi ini akan menggunakan kumpulan data Boston Housing Price yang telah disediakan oleh modul *tf.keras.datasets*. Pengukuran kinerja model ditentukan dengan menggunakan kumpulan tes data yang telah disediakan.

2.2 Rumusan Masalah

Adapun rumusan masalah yang ditentukan pada proses eksplorasi ini adalah:

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

2.3 Panduan Eksplorasi

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). Jika ada nilai hyperparameter lainnya yang sudah ditemukan pada eksplorasi sebelumnya, gunakan nilai hyperparameter optimal tsb pada eksplorasi berikutnya Nilai optimal diambil dari percobaan yang menghasilkan kinerja terbaik.

2.4 Tahapan Eksplorasi

Eksplorasi akan dilakukan 4 tahap yaitu:

1. Tahap 1 adalah pencarian jumlah hidden layar yang optimal sekaligus pencarian hidden unit yang optimal disetiap hidden layer.
2. Tahap 2 adalah pengujian terhadap activation function di setiap layer sehingga hasilnya optimal.
3. Tahap 3 adalah pengujian terhadap optimizer di setiap layer sehingga hasilnya optimal.
4. Tahap 4 adalah pencarian *probabilistic losses* yang menghasilkan kinerja paling baik dengan menggunakan arsitektur dari tahap 3.

2.5 Proses Eksplorasi

2.5.1 Kumpulan Data

Kumpulan data yang digunakan adalah Boston housing. Kumpulan data ini berisi informasi yang dikumpulkan oleh lembaga sensus di US mengenai perumahan di wilayah Boston. Kumpulan data ini selain untuk keperluan arsip juda telah digunakan secara luas di seluruh literatur untuk benchmark algoritma. Kumpulan data ini berukuran kecil dengan 506 baris data. Contoh data serperti yang terlihat pada Tabel 9.

Tabel 9. Contoh kumpulan data Boston housing.

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	1.23247	0.0	8.14	0.0	0.538	6.142	91.7	3.9769	4.0	307.0	21.0	396.90	18.72
1	0.02177	82.5	2.03	0.0	0.415	7.610	15.7	6.2700	2.0	348.0	14.7	395.38	3.11
2	4.89822	0.0	18.10	0.0	0.631	4.970	100.0	1.3325	24.0	666.0	20.2	375.52	3.26
3	0.03961	0.0	5.19	0.0	0.515	6.037	34.5	5.9853	5.0	224.0	20.2	396.90	8.01
4	3.69311	0.0	18.10	0.0	0.713	6.376	88.4	2.5671	24.0	666.0	20.2	391.43	14.65

2.5.2 Tahap 1

Pada tahap ini akan lakukan pengujian proses pembelajaran dengan berbagai arsitektur yang dirancang. Arsitektur yang dirancang hanya sebatas fully connected. Arsitekur yang diuji terlihat pada Tabel 1. Pada tabel diperlihatkan jumlah dense pada setiap hidden layer. Proses pembelajaran dilakukan sebanyak 1000 epoch.

Tabel 10. Daftar artsitektur yang akan diuji

No	Arsitektur	Layer 1	Layer 2	Layer 3
1	Arsitektur 1	128	-	-
2	Arsitektur 2	256		
3	Arsitektur 3	512		
4	Arsitektur 4	128	64	

5	Arsitektur 5	256	128	
6	Arsitektur 6	512	256	
7	Arsitektur 7	256	128	64
8	Arsitektur 8	512	256	128

Setelah melakukan proses pembelajaran maka didapat hasil seperti yang ditunjukkan Tabel 11. Pada tabel menunjukkan bahwa loss dan MSE terendah pada pembelajaran diperoleh pada artistektur 8, sedangkan pada MAE terendah pada pembelajaran pada arsitektur 7. Sehingga dapat disimpulkan bahwa arsitektur paling optimal pada tahap ini adalah artistektur 8. Arsitektur ini terdiri dari 3 hidden layer dengan 512 hidden unit pada layer pertama, 256 hidden unit pada layer kedua dan 128 hidden unit pada layer ketiga.

Tabel 11. Hasil pembelajaran dari setiap arsitektur

No	Arsitektur	Loss	MAE	MSE	Val. Loss	Val. MAE	Val. MSE
1	Arsitektur 1	29.991846	3.939339	29.991846	29.249886	3.887825	29.249886
2	Arsitektur 2	27.834175	3.716532	27.834175	27.600368	3.704438	27.600368
3	Arsitektur 3	22.678406	3.344893	22.678406	24.356831	3.557010	24.356831
4	Arsitektur 4	10.346138	2.330476	10.346138	28.336113	3.412851	28.336113
5	Arsitektur 5	10.085157	2.306992	10.085157	28.630051	3.419175	28.630051
6	Arsitektur 6	8.859613	2.164888	8.859613	26.418644	3.272488	26.418644
7	Arsitektur 7	8.175705	2.056012	8.175705	25.556967	3.152072	25.556967
8	Arsitektur 8	8.120215	2.133101	8.120215	25.224781	3.316952	25.224781

Kode program dari arsitektur 8 terlihat pada

Kode Program 3. Arsitektur 8 untuk permasalahan linear

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=[len(train_features[0])]),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(1)
])
```

2.5.3 Tahap 2

Pada tahap ini akan diujicobakan tipe fungsi aktifasi yang digunakan. Fungsi aktifasi yang akan digunakan adalah linear, relu, sigmoid dan tanh. Arsitektur yang digunakan adalah arsitektur 8. Hasil pembelajaran ditunjukkan pada Tabel 12. Pada tabel terlihat bahwa fungsi aktifasi tanh memperoleh hasil loss terkecil yaitu 3.839676.

Tabel 12. Hasil pembelajaran berdasarkan fungsi aktifiasi

No	Fungsi Aktifasi	Loss	MAE	MSE	Val. Loss	Val. MAE	Val. MSE
1	linear	3.861384	1.452860	3.861384	31.535276	3.466547	31.535276
2	relu	8.120215	2.133101	8.120215	25.224781	3.316952	25.224781
3	sigmoid	14.625973	2.826053	14.625973	32.291466	3.909680	32.291466
4	tanh	3.839676	1.452180	3.839676	34.192924	3.642324	34.192924

Fungsi aktifasi yang digunakan adalah tanh seperti yang terlihat pada Kode Program 4.

Kode Program 4. Penggunaan fungsi aktifasi tanh

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=[len(train_features[0])]),
    tf.keras.layers.Dense(512, activation='tanh'),
    tf.keras.layers.Dense(256, activation='tanh'),
    tf.keras.layers.Dense(128, activation='tanh'),
    tf.keras.layers.Dense(1)
])
```

2.5.4 Tahap 3

Pada tahap 3 akan ditentukan optimizer yang dinilai paling optimal. Langkah pertama pada tahap ini adalah menentukan learning rate yang pengujian. Nilai learning rate yang akan diuji adalah 0.001, 0.0007, 0.0005, 0.0003 dan 0.0001 dengan menggunakan optimizer Adam. Arsitektur yang digunakan adalah arsitektur 8 dengan fungsi aktifasi tanh dan proses pembelajaran sebanyak 1000 epoch. Pembelajaran akan dihentikan ketika MAE lebih kecil dari 1.5.

Tabel 13. Hasil pembelajaran dari setiap nilai learning rate

No	Arsitektur	Optimizer	Nilai LR	Epoch
1	Arsitektur 8	Adam	0.001	983
2	Arsitektur 8	Adam	0.0007	1305
3	Arsitektur 8	Adam	0.0005	1633
4	Arsitektur 8	Adam	0.0003	>2000
5	Arsitektur 8	Adam	0,0001	>2000

Langkah kedua adalah menggunakan optimizer yang berbeda. Optimizer yang digunakan adalah SGD, Adam, Adadelta, Adagrad dan RMSProp. Nilai LR yang digunakan adalah 0.001. Untuk penentuannya optimizer yang optimal digunakan metode yang sama pada saat penentuan nilai learning rate yaitu jumlah epoch terkecil untuk mencapai MAE lebih kecil dari 1.5. Hasil dari proses pengujian terlihat pada Tabel 14.

Tabel 14. Hasil pembelajaran dari setiap optimizer

No	Arsitektur	Optimizer	Nilai LR	Epoch
1	Arsitektur 8	SGD	0.001	>2000
2	Arsitektur 8	Adam	0.001	1003
3	Arsitektur 8	Adadelta	0.001	>2000
4	Arsitektur 8	Adagrad	0.001	>2000
5	Arsitektur 8	RMSProp	0.001	1423

Pada tahap 3 ini dapat disimpulkan bahwa optimizer dan nilai learning rate yang optimal adalah optimizer Adam dengan nilai learning rate = 0.001. Kode yang digunakan terlihat pada Kode Program 5.

Kode Program 5. Optimizer Adam dengan nilai learning rate = 0.001

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
                  loss='mse', metrics=['mae', 'mse'])
```

2.5.5 Tahap 4

Pada tahap 4 akan ditentukan tipe loss yang dinilai paling optimal. Langkah yang dilakukan adalah menguji berbagai tipe loss. Tipe loss yang akan diuji adalah Huber, MeanAbsoluteError dan Mean Squared Error. Pengujian ini menggunakan parameter optimizer Adam dengan nilai learning rate = 0.001 dan fungsi aktifasi tanh. Proses pembelajaran akan berhenti ketika mendapatkan nilai MAE lebih kecil dari 1.5.

Tabel 15. Jumlah epoch yang diperlukan pada setiap tipe loss yang diuji

No	Arsitektur	Loss	Epoch
1	Arsitektur 8	Huber	795
2	Arsitektur 8	Mean Absolute Error	920
3	Arsitektur 8	Mean Squared Error	968

Hasil pembelajaran menunjukkan pada perhitungan menggunakan tipe loss Huber merupakan yang paling optimal diantara yang lain.

Kode Program 6. Optimizer Adam, nilai learning rate = 0.001 dan loss Huber

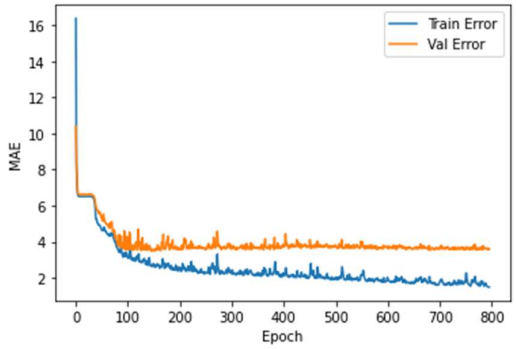
```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
                  loss='huber_loss', metrics=['mae', 'mse'])
```

2.6 Kesimpulan

Setelah melakukan 4 tahap eksplorasi maka dapat disimpulkan arsitektur yang digunakan adalah arsitektur 8 yang dapat dilihat pada Kode Program 4. *Optimizer* yang digunakan adalah Adam dengan nilai *learning rate* yang digunakan sebesar 0.001. Untuk tipe *loss* yang digunakan adalah Huber. Ketiga parameter ini dapat dilihat pada Kode Program 6. Kode lengkap dapat dilihat pada url: https://bit.ly/3tLh3es.

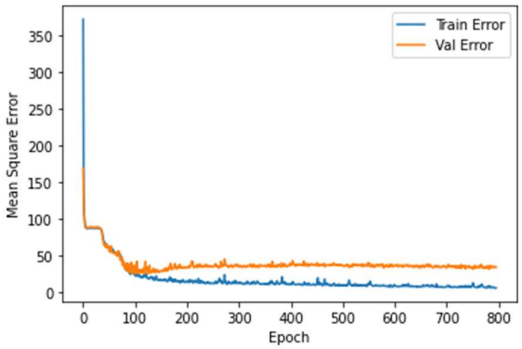
Hasil evaluasi Mean Absolute Error yang dilakukan terlihat pada Gambar 8.

Gambar 8. Grafik Mean Absolute Error



Hasil evaluasi grafik Mean Squared Error yang dilakukan terlihat pada Gambar 9.

Gambar 9. Grafik Mean Squared Error



Sedangkan untuk grafik prediksi model terlihat pada Gambar 10.

Gambar 10. Grafik prediksi model