Laporan Hasil Eksplorasi Hyperparameter CNN dan Neural Network

Nama : Sulthoni Ashiddiiqi
NIM : 33221006

1. Hasil Eksplorasi Hyperparameter CNN untuk Permasalahan Klasifikasi

1.1. Dataset

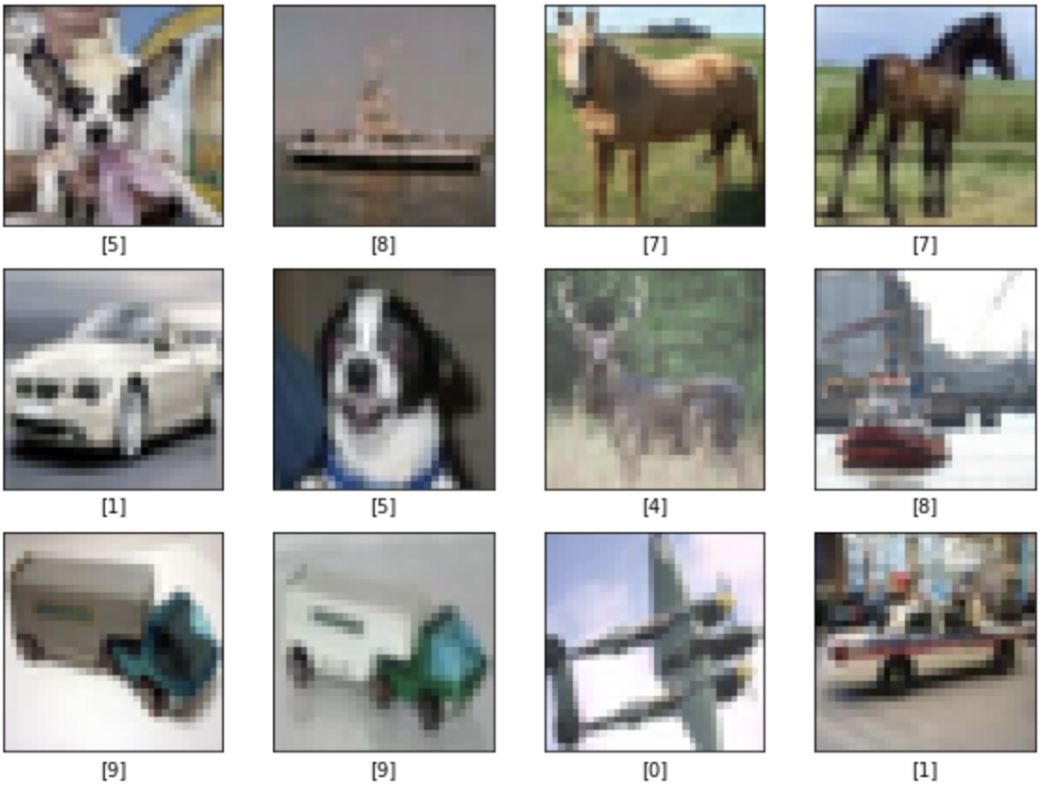
Dataset yang digunakan adalah *CIRAF10 small image* yang bisa didapatkan dari *https://keras.io/api/datasets/cifar10/*. Data ini memiliki kriteria:

- Jumlah training dataset sebanyak 50.000
- Jumlah test dataset sebanyak 10.000
- Ukuran gambar berwarna 32x32 pixel

Pada dataset ini terdapat 10 kelompok gambar. Berikut adalah label dan deskripsi gambar pada dataset.

Label	Deskripsi
0	Airplane
1	Automobile
2	Bird
3	Cat
4	Deer
5	Dog
6	Frog
7	horse
8	Ship
9	Truck

Berikut adalah contoh sampel gambar yang digunakan.



1.2. Proses Optimasi Model

Proses optimasi dilakukan untuk mencari nilai parameter yang bisa menghasilkan model paling optimal. Proses ini dilakukan secara experimental dengan mengubah-ubah parameter yang digunakan. Berikut urutan perubahan parameter yang dilakukan:

1. Menggunakan Convolution Layer sebanyak 2
2. Menggunakan filter dengan ukuran 3x3 pada kedua Convolution Layer
3. Mencari jumlah filter yang optimal.
4. Setelah didapatkan jumlah filter dengan akurasi tertinggi, selanjutnya mencari jumlah hidden unit dengan nilai akurasi tertinggi.
5. Setelah didapatkan jumlah filter dan hidden unit yang optimal, maka dilanjutkan dengan mencoba perubahan parameter jumlah convolution.

Detail nilai eksperimen dapat dilihat pada Lampiran 1

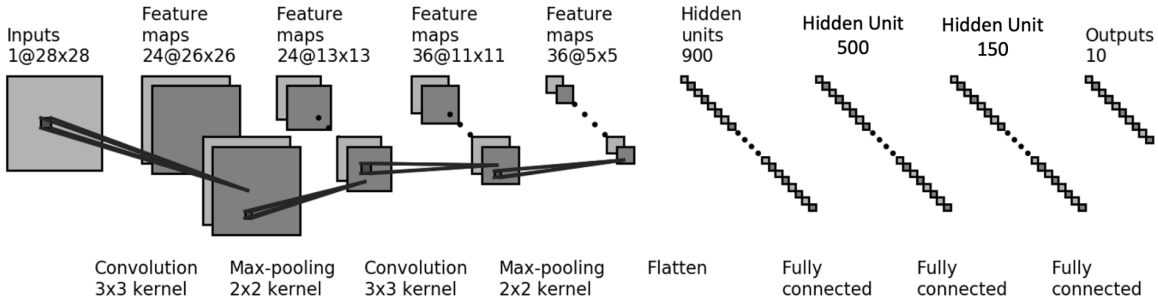
Untuk menghemat waktu training, maka *epochs* yang digunakan hanya sampai dengan 5 kali. Berdasarkan eksperimen dengan *epochs* 10 antar parameter yang berbeda perubahannya konsisten. Sehingga pada kebanyakan learning process hanya menggunakan *epochs* 5.

1.3. Hasil

Berdasarkan hasil eksperimen, berikut adalah parameter yang digunakan untuk mendapatkan hasil yang optimal:

Parameter	Nilai
Jumlah Convolution Layer	2
Ukuran Filter	Layer 1 : 3x3 Layer 2 : 3x3
Jumlah Filter	Layer 1 : 50@30x30 Layer 2 : 100@13x13
Jumlah Hidden Unit	Layer 1 : 500 Layer 2 : 150
Optimizer dengan hasil optimal	Adam
Learning Rate dengan hasil optimal	0.001
Losses	sparce_categorical_crossentropy

Berikut adalah ilustrasi model optimal yang digunakan



Hasil akhir dari model yang digunakan dengan dengan batch 36 dan epochs 5 akurasi training mencapai 81.66% dan akurasi test mencapai 71.76%. Saat ditingkatkan menjadi epochs 10 hasil akurasi training mencapai 95.56% dan akurasi test mencapai 70.39%.

1.4. Contoh hasil

Contoh dengan hasil optimal

```
# Melakukan training dengan konfigurasi model, batch_size dan jumlah epochs yang telah ditentukan sebelumnya
cnn_model.fit(train_images, train_labels, batch_size=BATCH_SIZE, epochs=EPOCHS)
```

```
Epoch 1/10
782/782 [==============================] - 117s 149ms/step - loss: 1.3671 - accuracy: 0.5057
Epoch 2/10
782/782 [==============================] - 115s 147ms/step - loss: 0.9811 - accuracy: 0.6553
Epoch 3/10
782/782 [==============================] - 114s 146ms/step - loss: 0.8112 - accuracy: 0.7160
Epoch 4/10
782/782 [==============================] - 114s 146ms/step - loss: 0.6583 - accuracy: 0.7702
Epoch 5/10
782/782 [==============================] - 114s 146ms/step - loss: 0.5273 - accuracy: 0.8166
Epoch 6/10
782/782 [==============================] - 114s 145ms/step - loss: 0.4066 - accuracy: 0.8575
Epoch 7/10
782/782 [==============================] - 114s 146ms/step - loss: 0.3006 - accuracy: 0.8960
Epoch 8/10
782/782 [==============================] - 115s 147ms/step - loss: 0.2155 - accuracy: 0.9249
Epoch 9/10
782/782 [==============================] - 116s 148ms/step - loss: 0.1600 - accuracy: 0.9445
Epoch 10/10
782/782 [==============================] - 116s 148ms/step - loss: 0.1280 - accuracy: 0.9556
<keras.callbacks.History at 0x7f941ca30f50>
```

```
# Melakukan tes pada dataset testing
test_loss, test_acc = cnn_model.evaluate(test_images, test_labels)

print('Test accuracy:', test_acc)
```

```
313/313 [==============================] - 7s 22ms/step - loss: 1.3975 - accuracy: 0.7039
Test accuracy: 0.7038999795913696
```

2. Hasil Eksplorasi Neural Network untuk Permasalahan Regresi

2.1. Dataset

Dataset yang digunakan adalah *Boston Housing Price small image* yang bisa didapatkan dari *https://keras.io/api/datasets/boston_housing/.* Data ini memiliki kriteria:

- Jumlah training dataset sebanyak 404
- Jumlah test dataset sebanyak 102
- Pada independen variabel terdapat 13 variabel (x_1, x_2, … x_{13})
- Dependen variabel yaitu harga rumah (y)

2.2. Optimasi Model

Sebelum proses optimasi model dilakukan, hal pertama yang dilakukan adalah menyiapkan datanya terlebih dahulu. Supaya antar nilai pada independen

variabel tidak terlalu jauh dan hasil pengolahan bisa lebih baik, maka dataset pada independen variabel (baik training maupun tes) harus dinormalisasi terlebih dahulu. Untuk menormalisasi data, yang dilakukan adalah mengurangi masing-masing nilai dengan nilai rata-ratanya dan membagi hasilnya dengan standar deviasinya.

Langkah-langkah yang dilakukan untuk mencari nilai optimal adalah dengan:

1. Mencari jumlah hidden layer dan hidden unit per layer yang optimal
2. Mencari optimizer yang optimal
3. Mencari activation function yang optimal
4. Mencari Learning rate yang optimal

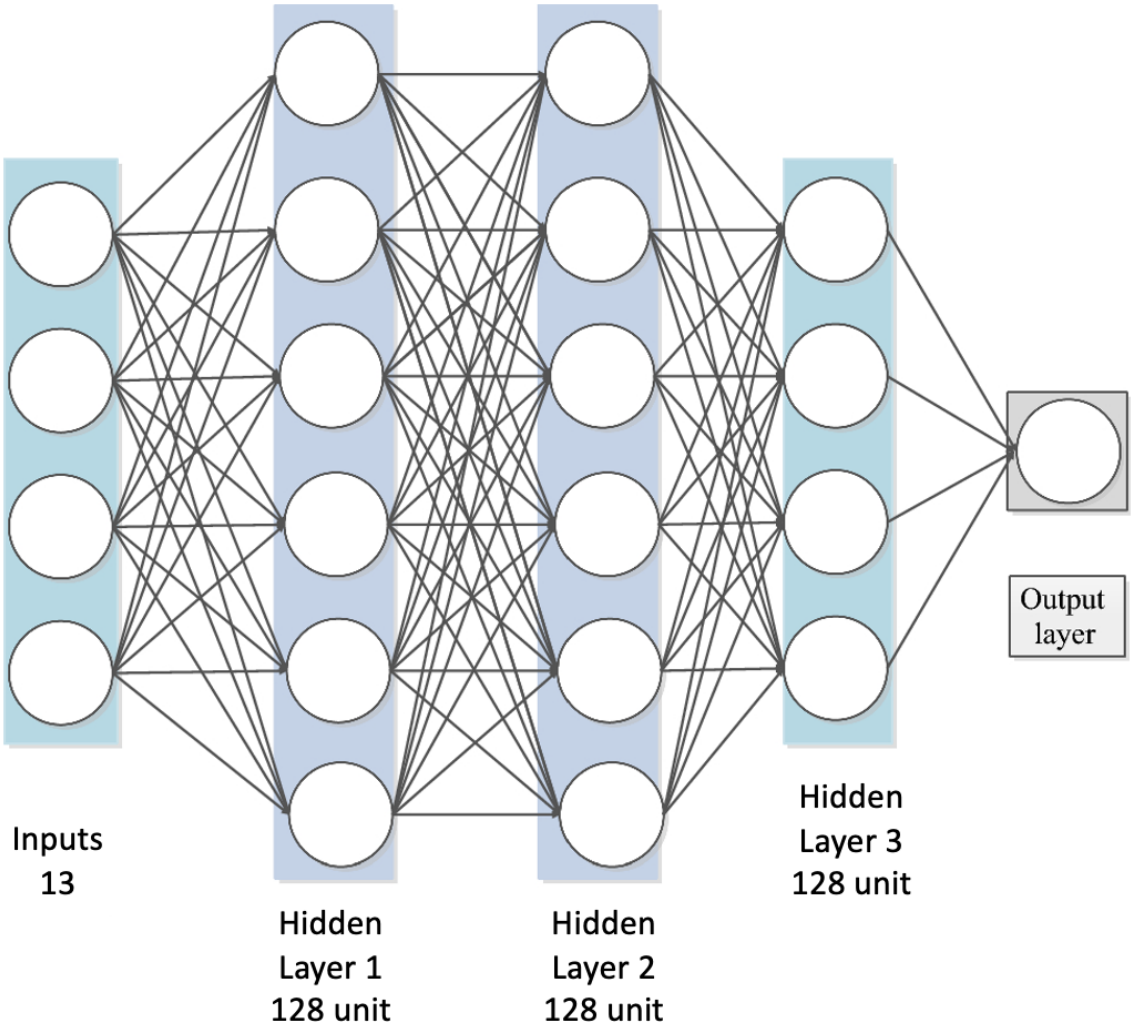
Detail nilai eksperimen dapat dilihat pada Lampiran 2

2.3. Hasil

Berdasarkan hasil eksperimen, berikut adalah parameter yang digunakan untuk mendapatkan hasil yang optimal:

Parameter	Nilai
Jumlah Hidden Layer	3
Jumlah Hidden Unit	Layer 1 : 128 Layer 2 : 128 Layer 3 : 128
Activation function	Relu
Optimizer	Adam
Loss function	mean_square_error
Learning Rate dengan hasil optimal	0.01

Berikut adalah ilustrasi model optimal yang digunakan



2.4. Contoh Hasil

Berikut contoh hasil yang optimal

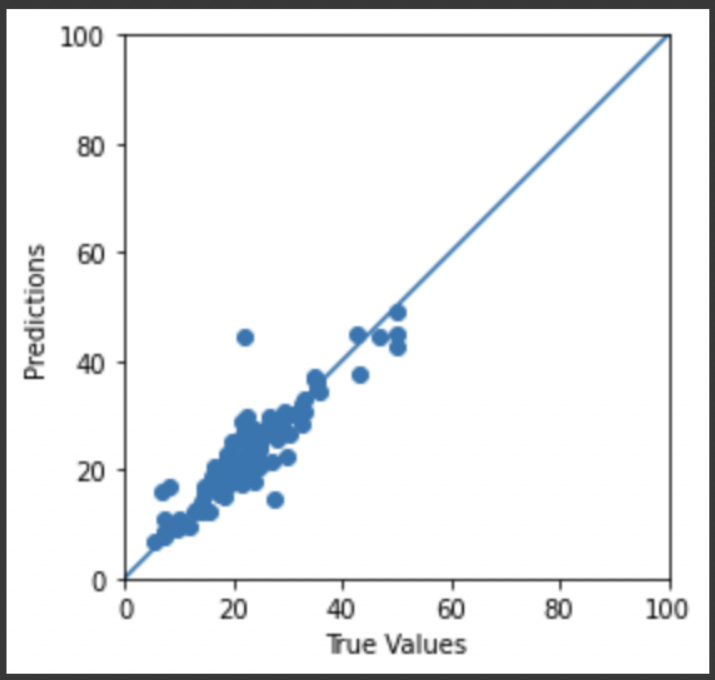
```
# Melakukan tes pada dataset testing
test_loss, test_acc = nn_model.evaluate(x_test, y_test)

print('test_loss:', test_loss)
print('Test accuracy:', test_acc)
print('x_test', x_test[0:2])
```

```
4/4 [==============================] - 0s 3ms/step - loss: 16.0581 - mae: 2.6080
test_loss: 16.058120727539062
Test accuracy: 2.608041763305664
```

```
test_predictions = nn_model.predict(x_test).flatten()

a = plt.axes(aspect='equal')
plt.scatter(y_test, test_predictions)
plt.xlabel('True Values')
plt.ylabel('Predictions')
lims = [0, 100]
plt.xlim(lims)
plt.ylim(lims)
_ = plt.plot(lims, lims)
```



Source Code

https://github.com/sulthoni/cnn-task

Lampiran 1. Hasil Eksperimen untuk Mencari Hyperparameter yang Optimal untuk Permasalahan Klasifikasi

	1	2	3	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Convolution Layer	2	2	2	3	2	2	2	2	2	2	2	2	3	2	2	3	2	2	2		2
Ukuran Filter	[(3,3),(3,3)]	[(3,3),(3,3)]	[(3,3),(3,3)]	[(3,3),(3,3)]	[(3,3),(3,3)]	[(3,3),(3,3)]	[(3,3),(3,3)]	[(3,3),(3,3)]	[(3,3),(3,3)]	[(3,3),(3,3)]	[(3,3),(3,3)]	[(3,3),(3,3)]	[(3,3),(3,3),(3,3)]	[(3,3),(5,5)]	[(3,3),(3,3)]	[(3,3),(3,3),(3,3)]	[(3,3),(3,3)]	[(3,3),(3,3)]	[(3,3),(3,3)]	[(3,3),(3,3)]	[(3,3),(3,3)]
Banyak Filter	(24,36)	(50, 75)	(50, 75)	(50, 75, 125)	(50, 75)	(50, 90)	(50, 100)	(40, 100)	(50, 100)	(50, 100)	(50, 100)	(50, 100)	(50, 100, 150)	(50, 100)	(40, 90)	(32, 64, 128)	(64, 128)	(64, 128)	(64, 128)	(64, 128)	(50, 100)
Hidden Unit	128	(500, 150)	(500, 250)	(500, 150)	(500, 150)	(500, 150)	(500, 150)	(500, 150)	(300, 150)	(400, 150)	(500, 250)	(500, 150)	(500, 150)	(500, 150)	(500, 150)	(512, 128)	(512, 128)	(512, 128)	(512, 128, 64, 32	(512, 128, 64, 32	(500, 150)
Activation	relu, softmax	relu, softmax	relu, softmax	relu, softmax	relu, softmax	relu, softmax	relu, softmax	relu, softmax	relu, softmax	relu, softmax	relu, softmax	relu, softmax	relu, softmax	relu, softmax	relu, softmax	relu, softmax	relu, softmax	relu, softmax	relu, softmax	relu, softmax	relu, softmax
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	SGD	Adamax
Learning rate schedule	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3
Probabilistic Losses	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy	sparse_categorical_crossentropy
Epochs	5	5	5	5	5	5	5	5	5	5	5	10	5	5	5	5	5	10	5	5	5
Training Accuracy	0.6757	0.8043	0.8006	0.7380	0.7892	0.7883	0.8166	0.8062	0.7778	0.8031	0.8053	0.9532	0.7556	0.75	0.8043	0.7355	0.819	0.9304	0.7987	0.2179	0.726
Test Accuracy		0.7222		0.6887	0.7074	0.6994	0.7176	0.7117	0.7201	0.7203	0.7072	0.7063	0.6921	0.6929	0.7295	0.7024	0.72	0.699	0.7145	0.2167	0.7

Legenda : Nilai parameter paling optimal

Lampiran 2. Hasil Eksperimen untuk Mencari Hyperparameter yang Optimal untuk Permasalahan Regresi

	1	2	3	4	5	6	7	8	9	10	11	12	13	
Jumlah Hidden layer	2	3	3	3	3	3	2	3	3	3	3	3	3	
Jumlah hidden unit per layer	(128, 64)	(256, 128, 64)	(256, 256, 256)	(128, 128, 128)	(64, 64, 64)	(32, 32, 32)	(64, 64)	(128, 128, 128)	(128, 128, 128)	(128, 128, 128)	(128, 128, 128)	(128, 128, 128)	(128, 128, 128)	
Activation function	relu	relu	relu	relu	relu	relu	relu	relu	relu	relu	relu	relu	tanh	
Optimizer yang optimal	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	RMSProp	Adamax	Adam	Adam	Adam	
Loss function yang optimal	mean_squared_e	mean_squared_e	mean_squared_e	mean_squared_e	mean_squared_e	mean_squared_e	mean_squared_e	mean_absolute_	mean_squared_e	mean_squared_e	mean_squared_e	mean_squared_e	mean_squared_error	
Learning Rate	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.0001	0.01	0.01	
Epochs	100	100	100	100	100	100	100	100	100	100	100	100	100	
Test Accuracy 'mae'	2.910553694	2.676285505	2.768587112	2.648874998	2.810562372	2.950180292	3.10383296	2.932986498	3.159449577	3.084186316	3.280569077	2.608041763	3.230306625	
				x								x		

Legenda : Nilai parameter paling optimal