Laporan Ujian Akhir Semester

Machine Learning Kelompok 5

Oleh:

Yunata Putra Gunawan	C14190177
Welliam Sastradipura	C14190194
Louis Margatan Subekti	C14190196

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS KRISTEN PETRA

SURABAYA

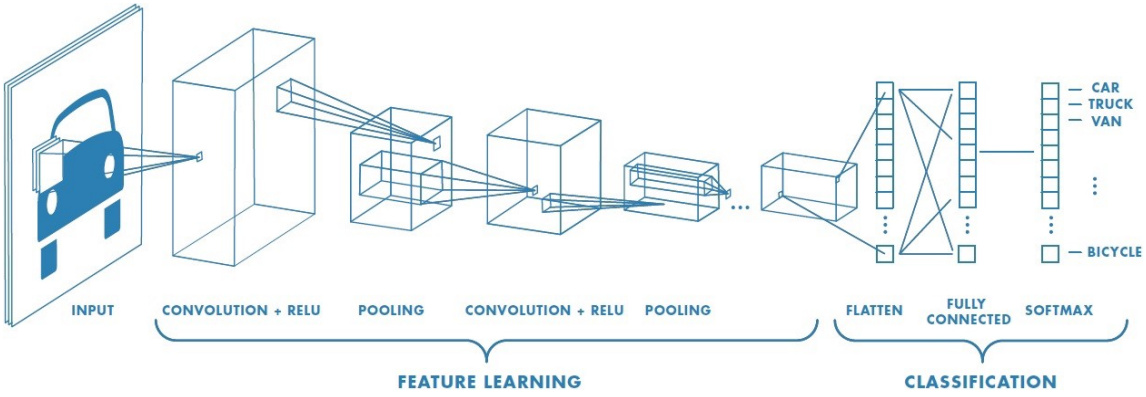
2021

A. Tugas yang dikerjakan

1. *Balancing dataset*
2. Modifikasi model
3. *Cropping* gambar yang *background* nya besar
4. Variasi *epoch*

B. Teori model yang diuji

Project ini menggunakan *CNN* (*Convolutional Neural Network*), dimana *CNN* adalah salah satu model dari *Neural Network* yang biasanya digunakan untuk menganalisa gambar visual. Kemudian arsitektur *CNN* digunakan untuk mengambil gambar visual sehingga bisa digunakan sebagai *input*, kemudian memberikan *weight* dan *bias* ke seluruh aspek gambar, sehingga gambar yang satu itu berbeda dengan yang lain.



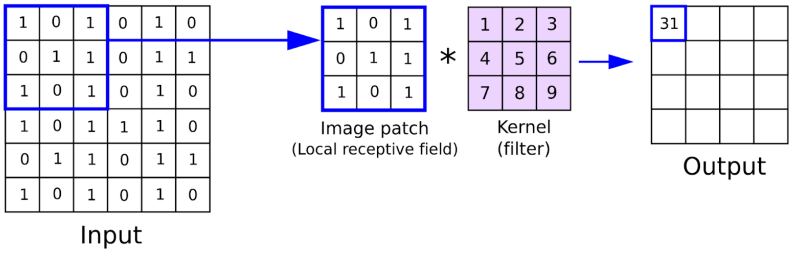
Gambar B.1 Contoh cara kerja dari *CNN*

Kelompok kami mempunyai 4 model *CNN*, dimana model pertama adalah model yang didapat dari orang lain (Model *Original*), sedangkan model kedua, ketiga, dan keempat adalah model yang dimodifikasi dari model pertama. *CNN* sendiri dilatih menggunakan kumpulan gambar yang diberi label identitas (anjing maupun kucing). Kumpulan gambar tersebut terdiri dari ribuan gambar. Pada setiap gambar yang diprediksi oleh *CNN*, hasil label akan dibandingkan dengan label data yang digunakan

dengan data latih (*train* data) dan *distance* tiap prediksi akan dievaluasi tiap *set* gambar, kemudian model akan dimodifikasi untuk meminimalisir *distance* nya sehingga kapabilitas prediksi nya meningkat.

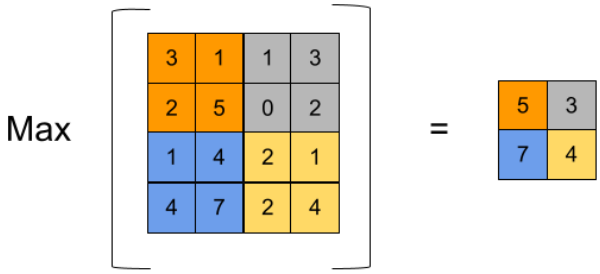
Di dalam CNN, ada beberapa istilah yang harus di kenali, yaitu:

- ***Convolutional layer*** : Merupakan layer yang memanfaatkan sesuatu yang disebut sebagai *filter*, dimana *filter* ini akan terus di *update* selama proses pembelajaran berlangsung. Di dalam layer ini terdapat juga sebuah *kernel*, dimana fungsi dari *kernel* ini adalah mengekstraksi data dari gambar



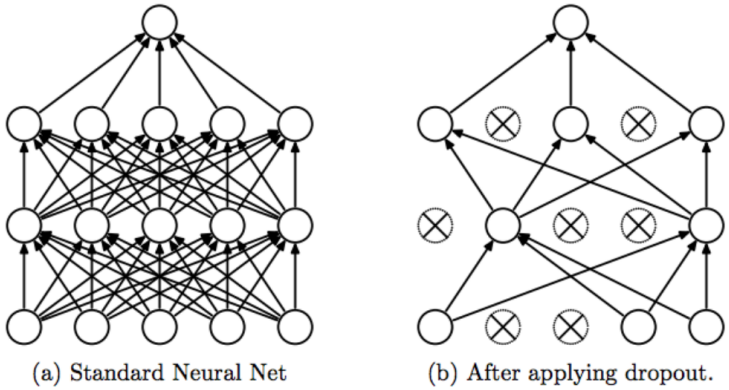
Gambar B.2 Contoh cara kerja dari ***Convolutional layer***

- ***Pooling Layer*** : Merupakan sebuah *layer* yang berfungsi mereduksi *input* agar tidak terjadi *error* atau *overfit*.



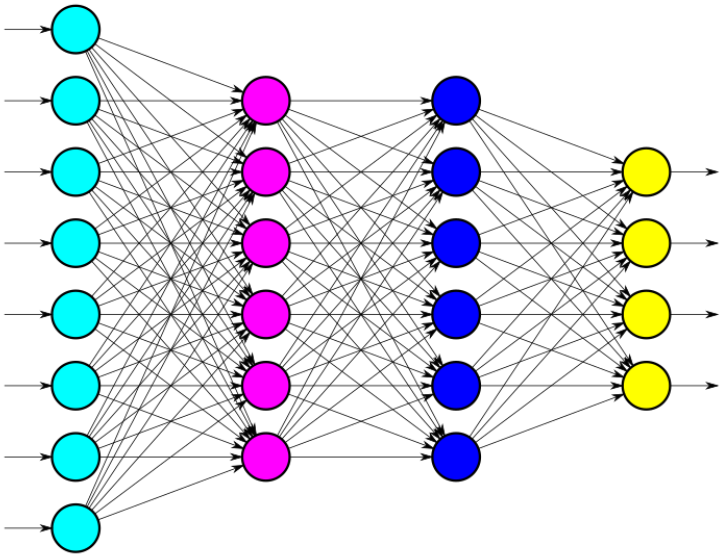
Gambar B.3 Contoh cara kerja dari ***Pooling layer***

- ***Dropout*** : Merupakan sebuah fungsi untuk mereduksi kepadatan neuron-neuron yang terhubung dari *layer* ke *layer* berikutnya.



Gambar B.4 Contoh cara kerja dari fungsi ***Dropout***

- ***Dense layers*** : Merupakan layer untuk proses klasifikasi terakhir dan untuk *output*.

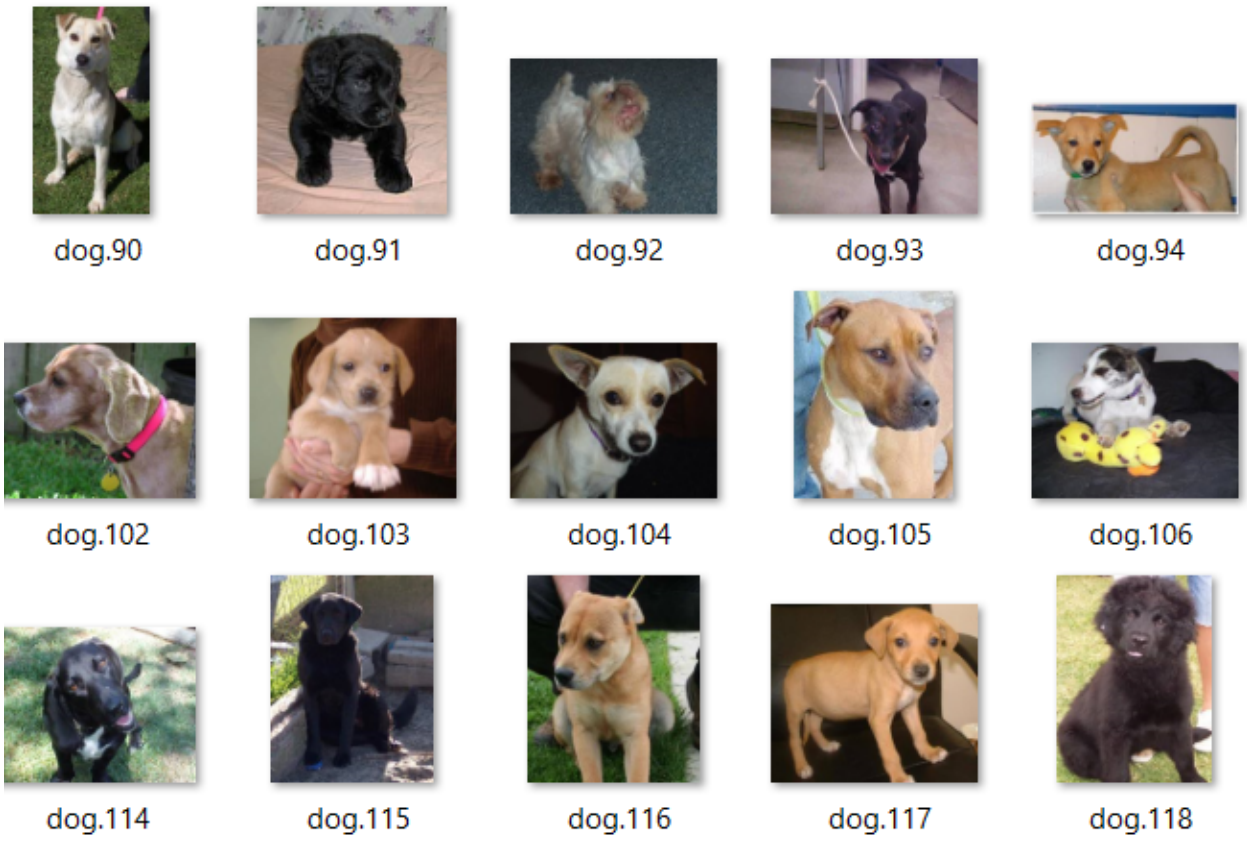


Gambar B.5 Contoh cara kerja dari ***Dense layers***

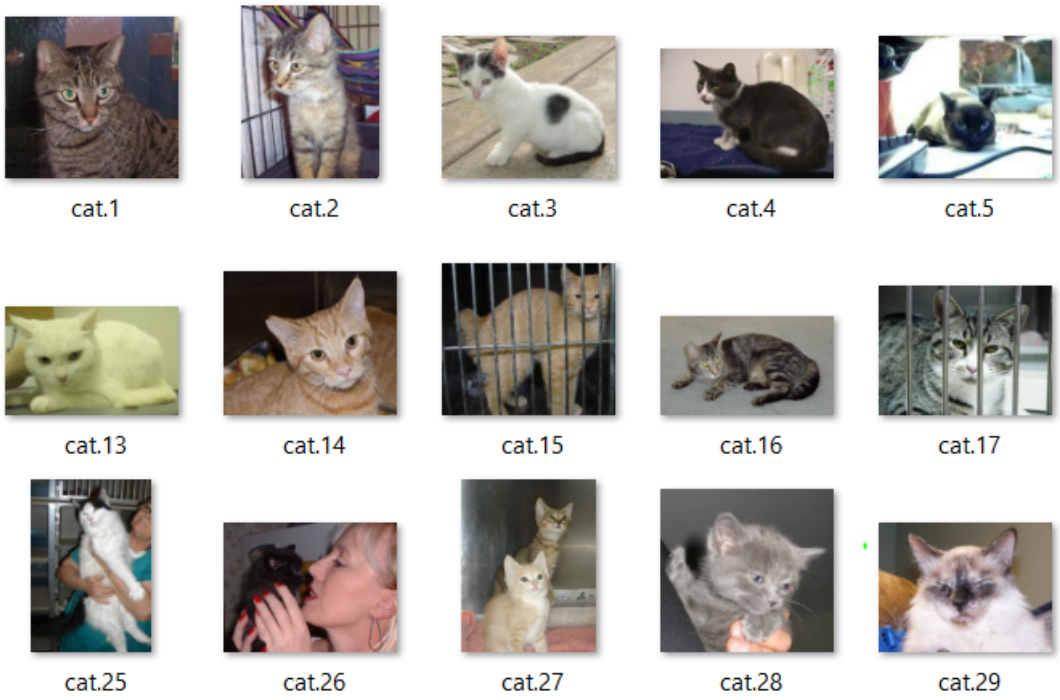
C. Penjelasan Dataset

Link Dataset : https://www.kaggle.com/biaiscience/dogs-vs-cats

Dataset yang digunakan adalah *dataset* yang berisi gambar-gambar hewan anjing dan kucing dengan total sebanyak 37.500 gambar. *Dataset* tersebut kemudian dibagi menjadi 2 *folder*, yaitu *folder Train* dan *folder Test*. Di dalam *folder Train* terdapat 25.000 gambar anjing dan kucing, sedangkan di dalam *folder Test* terdapat 12.500 gambar anjing dan kucing.



Gambar C.1 Contoh gambar *training* anjing



Gambar C.2 Contoh gambar *training* kucing

Gambar C.3 Contoh gambar *testing*

D. Hasil Pengujian dan Analisa

1. Pengujian Variasi model

Di dalam pengujian variasi model, seperti yang disampaikan pada penjelasan bagian A, kelompok kami memiliki 4 model *CNN* dan dimana kami akan menunjukan hasil percobaan *training* dan validasi dari masing masing model.

a. Model 1

```
def model_1():
    model=keras.Sequential()

    model.add(Conv2D(filters=64,kernel_size=(3,3),activation='relu',input_shape=input_shape))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.20))

    model.add(Conv2D(filters=64,kernel_size=(3,3),activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.30))

    model.add(Conv2D(filters=32,kernel_size=(3,3),activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.1))

    #Add Dense Layers on top
    model.add(Flatten())
    model.add(Dense(32,activation='relu'))
    model.add(Dense(2,activation='sigmoid'))

    return model
```

Gambar D.1.a.1 Kodingan Model 1

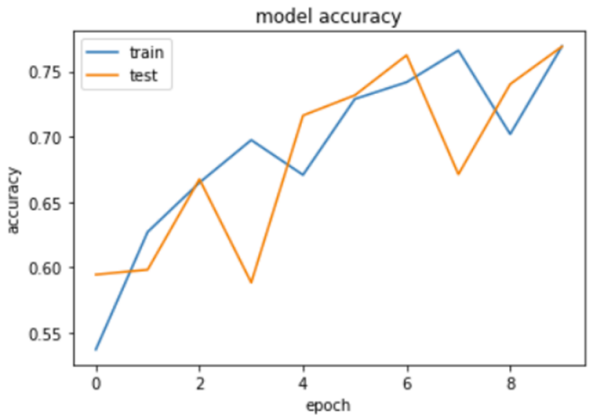
Keterangan model :

Model 1	
Convolutional layer	64x64x32
Pooling Layer	2x2
Dropout	0.2, 0.3, 0.1
Dense layers	32x2

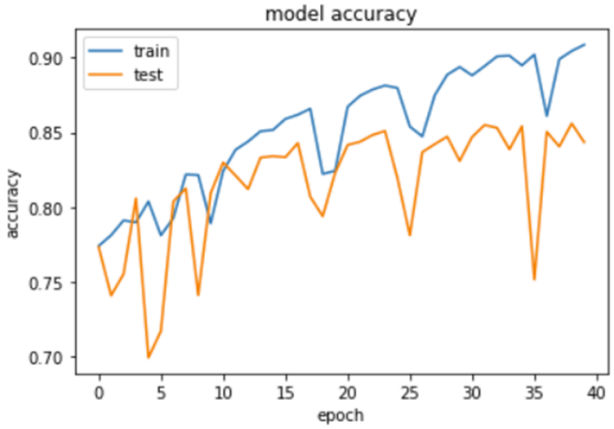
Hasil Percobaan:

Epoch	***Training Accuracy***	***Validation Accuracy***
10	76.92%	76.91%
50	90.87%	84.35%
75	93.92%	86.00%
100	94.95%	85.31%

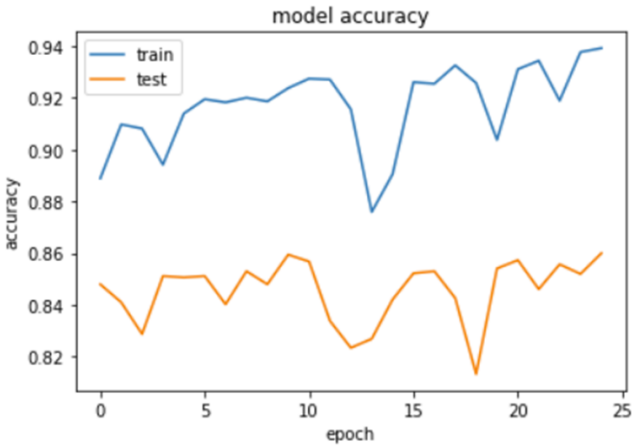
Grafik Percobaan:



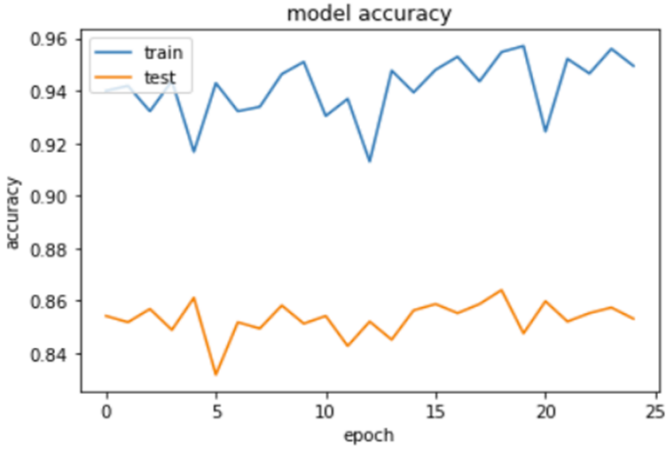
Gambar D.1.a.2 Grafik *Epoch* 10



Gambar D.1.a.3 Grafik *Epoch* 50



Gambar D.1.a.4 Grafik *Epoch* 75



Gambar D.1.a.5 Grafik *Epoch* 100

b. Model 2

```
def model_2():
    model=keras.Sequential()


    model.add(Conv2D(filters=128,kernel_size=(3,3),activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(filters=128,kernel_size=(3,3),activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.2))

    model.add(Flatten())
    model.add(Dense(16,activation='relu'))
    model.add(Dropout(0.1))
    model.add(Dense(2,activation='sigmoid'))
```

Gambar D.1.b.1 Kodingan Model 2

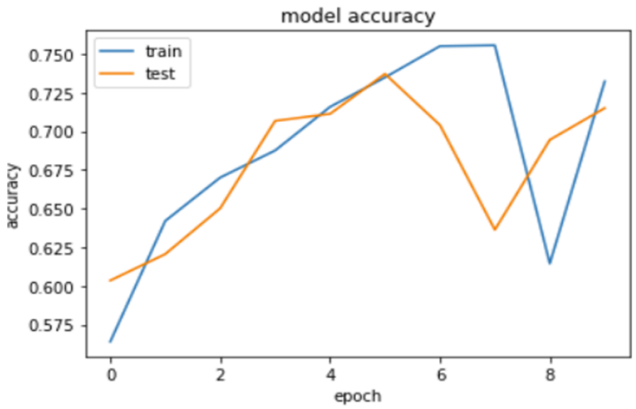
Keterangan model :

Model 2	
Convolutional layer	128x128
Pooling Layer	2x2
Dropout	0.2, 0.2, 0.1
Dense layers	16x2

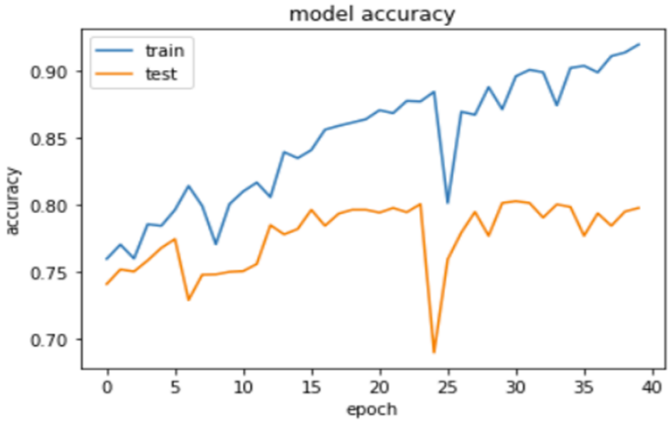
Hasil Percobaan:

Epoch	***Training Accuracy***	***Validation Accuracy***
10	73.22%	71.49%
50	92.00%	79.79%
75	93.50%	79.81%
100	94.10%	79.55%

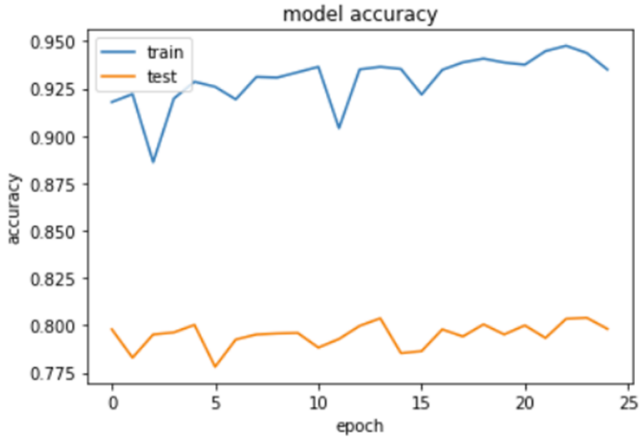
Grafik Percobaan:



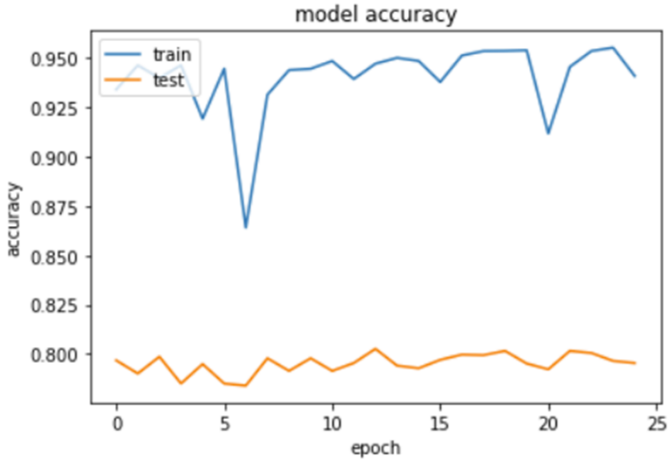
Gambar D.1.b.2 Grafik *Epoch* 10



Gambar D.1.b.3 Grafik *Epoch* 50



Gambar D.1.b.4 Grafik *Epoch* 75



Gambar D.1.b.5 Grafik *Epoch* 100

c. Model 3

```
def model_3():
    model=keras.Sequential()

    model.add(Conv2D(filters=128,kernel_size=(3,3),activation='relu',input_shape=input_shape))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.1))

    model.add(Conv2D(filters=64,kernel_size=(3,3),activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(filters=32,kernel_size=(3,3),activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.3))

    model.add(Conv2D(filters=16,kernel_size=(3,3),activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.3))

    model.add(Flatten())
    model.add(Dense(64,activation='relu'))
    model.add(Dense(2,activation='sigmoid'))
```

Gambar D.1.c.1 Kodingan Model 3

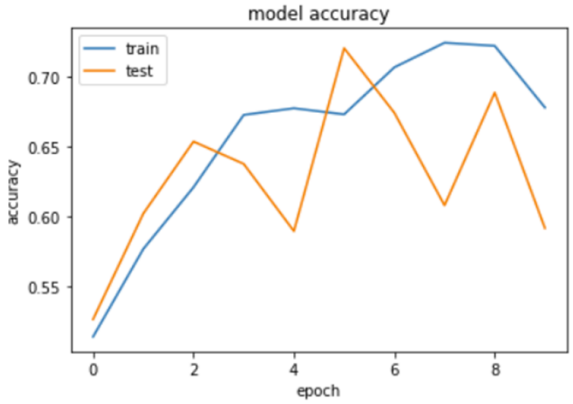
Keterangan model :

Model 3	
Convolutional layer	128x64x32x16
Pooling Layer	2x2
Dropout	0.1, 0.2, 0.3, 0.4
Dense layers	64x2

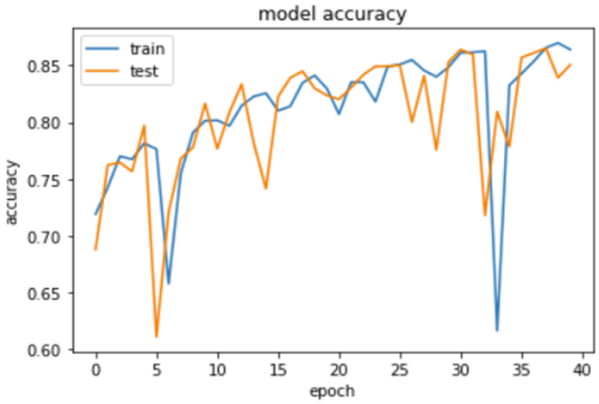
Hasil Percobaan:

Epoch	***Training Accuracy***	***Validation Accuracy***
10	67.79%	59.17%
50	86.41%	85.07%
75	88.82%	87.23%
100	91.14%	88.29%

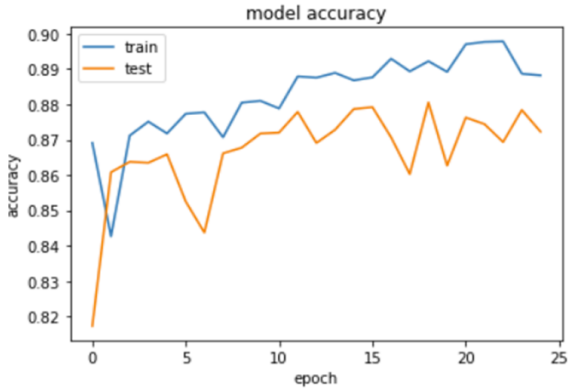
Grafik Percobaan:



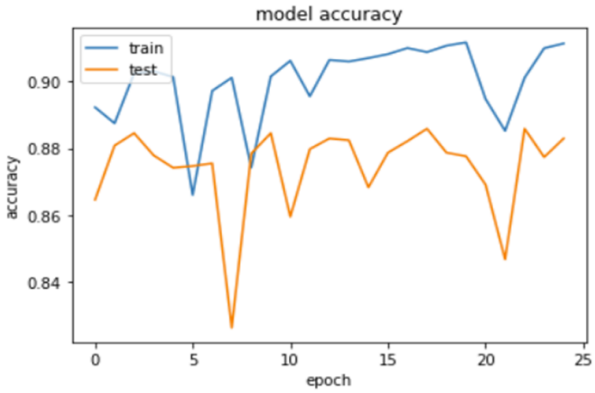
Gambar D.1.c.2 Grafik *Epoch* 10



Gambar D.1.c.3 Grafik *Epoch* 50



Gambar D.1.c.4 Grafik *Epoch* 75



Gambar D.1.c.5 Grafik *Epoch* 100

d. Model 4

```
def model_4():
    model=keras.Sequential()

    model.add(Conv2D(filters=32,kernel_size=(3,3),activation='relu',input_shape=input_shape))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.1))

    model.add(Conv2D(filters=64,kernel_size=(3,3),activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(filters=128,kernel_size=(3,3),activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.3))

    model.add(Flatten())
    model.add(Dense(32,activation='relu'))
    model.add(Dropout(0.1))
    model.add(Dense(2,activation='sigmoid'))
```

Gambar D.1.d.1 Kodingan Model 4

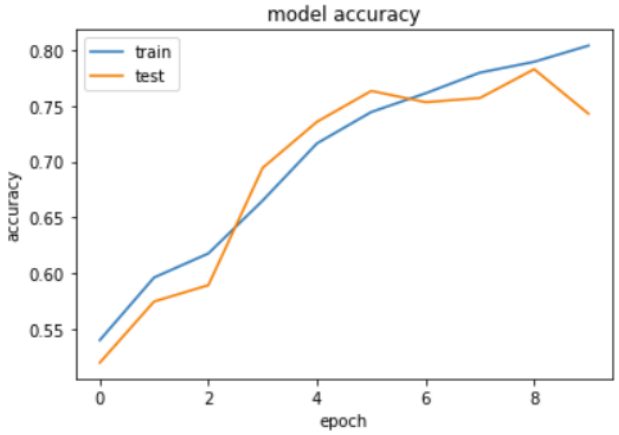
Keterangan model :

Model 4	
Convolutional layer	32x64x128
Pooling Layer	2x2
Dropout	0.1, 0.2, 0.3, 0.1
Dense layers	32x2

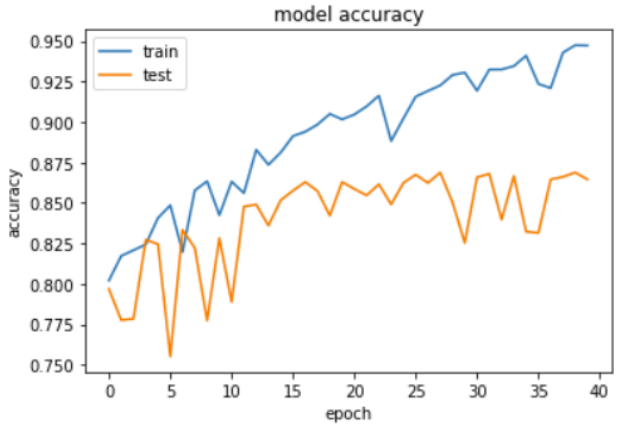
Hasil Percobaan:

Epoch	***Training Accuracy***	***Validation Accuracy***
10	80.41%	74.29%
50	94.73%	86.45%
75	96.85%	86.53%
100	97.69%	84.92%

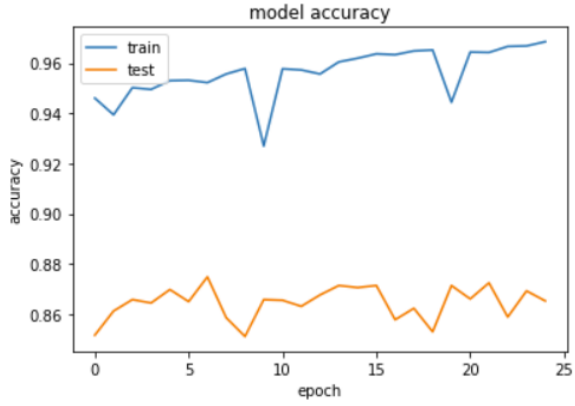
Grafik Percobaan:



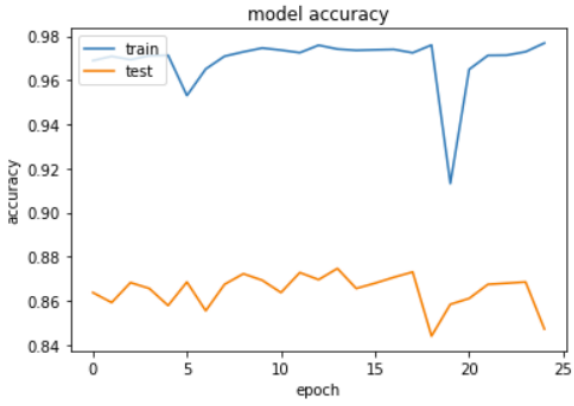
Gambar D.1.d.2 Grafik *Epoch* 10



Gambar D.1.d.3 Grafik *Epoch* 50



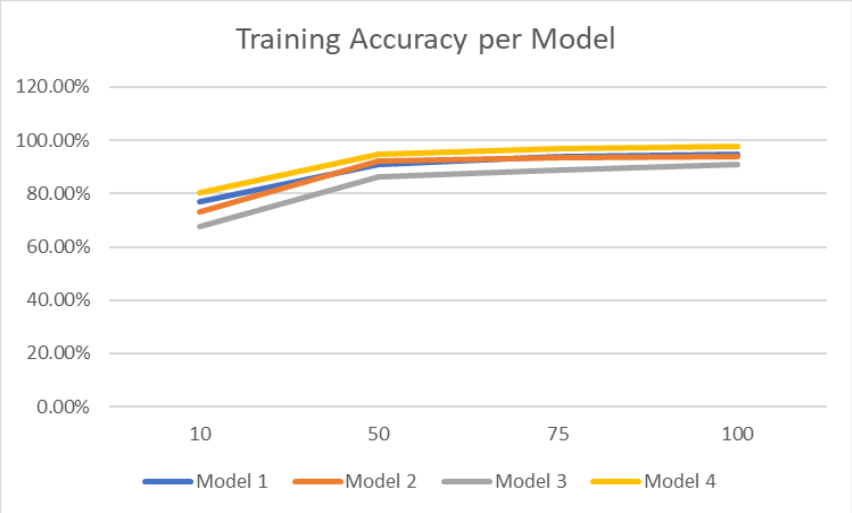
Gambar D.1.d.4 Grafik *Epoch* 75



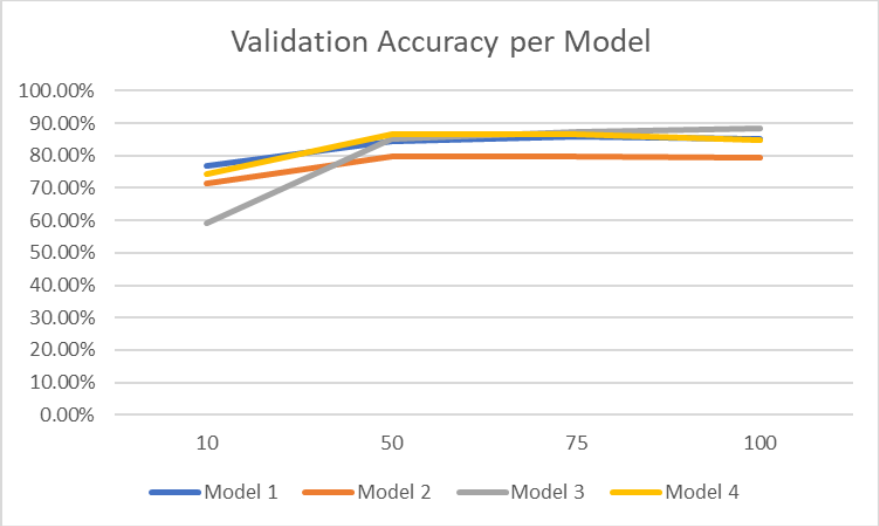
Gambar D.1.d.5 Grafik *Epoch* 100

e. Tabel Perbandingan akhir

Epoch	*Training Accuracy*			
	Model 1	Model 2	Model 3	Model 4
10	76.92%	73.22%	67.79%	80.41%
50	90.87%	92.00%	86.41%	94.73%
75	93.92%	93.50%	88.82%	96.85%
100	94.95%	94.10%	91.14%	97.69%



Epoch	*Validation Accuracy*			
	Model 1	Model 2	Model 3	Model 4
10	76.91%	71.49%	59.17%	74.29%
50	84.35%	79.79%	85.07%	86.45%
75	86.00%	79.81%	87.23%	86.53%
100	85.31%	79.55%	88.29%	84.72%



2. *Balancing Dataset*

Setelah mencoba variasi model *CNN*, adapun langkah selanjutnya yaitu memilih model untuk percobaan selanjutnya, disini kami memilih model 3 dan 4 untuk digunakan pada percobaan selanjutnya. Kemudian pada percobaan berikut, hal yang diuji adalah *balancing dataset*, dimana percobaan ini kami akan menghapus beberapa dari gambar kucing dan anjing dan melihat hasilnya.

a. Percobaan data kucing

Slicing	***Epoch***	***Training Accuracy***		***Validation Accuracy***	
		Model 3	**Model 4**	**Model 3**	**Model 4**
25%	10	79.54%	82.63%	82.21%	83.73%
	50	89.81%	95.50%	88.39%	88.42%
	75	91.75%	96.86%	88.27%	87.84%
	100	92.74%	97.83%	88.24%	88.18%
50%	10	79.34%	82.91%	79.83%	82.45%
	50	90.33%	96.12%	88.79%	87.26%
	75	92.84%	97.43%	89.79%	87.72%
	100	93.79%	98.07%	90.13%	88.39%
75%	10	81.12%	84.90%	81.78%	85.67%
	50	91.45%	97.27%	89.76%	88.61%
	75	93.34%	98.64%	89.38%	90.06%
	100	94.89%	98.70%	90.40%	88.95%

b. Percobaan data anjing

Slicing	***Epoch***	***Training Accuracy***		***Validation Accuracy***	
		Model 3	**Model 4**	**Model 3**	**Model 4**
25%	10	74.04%	81.50%	77.18%	82.69%
	50	89.12%	96.05%	88.33%	87.17%
	75	90.93%	97.48%	88.85%	87.66%
	100	92.39%	97.84%	88.91%	87.75%
50%	10	73.01%	81.26%	74.72%	81.41%
	50	87.34%	95.35%	84.57%	85.67%
	75	89.52%	97.82%	86.28%	85.50%
	100	91.11%	98.29%	86.63%	86.35%
75%	10	80.28%	84.14%	79.22%	82.81%
	50	89.59%	96.44%	88.23%	87.50%
	75	92.07%	98.39%	88.91%	88.44%
	100	93.56%	98.73%	89.89%	88.35%

PS : Untuk hasil grafik nya bisa dilihat di github dengan nama folder "*Balancing Experiment Graph*"

E. Kesimpulan

1. Di percobaan tentang variasi model, kami mempunyai 4 model dimana masing-masing dari model tersebut di training dengan *epoch* 10, 50, 75, 100. Hasil pengujian yang kami dapat dari model-model tersebut, terlihat bahwa model yang memiliki akurasi tertinggi saat *training* adalah model 4 dengan nilai sebesar 98%, dan model yang memiliki akurasi tertinggi saat validasi adalah model 3 dengan nilai sebesar 88%.

2. Dari hasil percobaan kami tentang *balancing dataset, imbalance dataset* bisa jadi mempengaruhi akurasi sebuah model, dikarenakan gambar yang di *train* tidak seimbang sehingga model lebih mengetahui salah satu gambar saja. Alasan kami mengatakan bisa jadi adalah karena hasil percobaan yang kami dapat tidak sesuai dengan ekspektasi kami. Kami mengira bahwa semakin data nya berkurang, semakin rendah akurasi saat validasi, tetapi hasil nya kurang lebih sama dengan dataset yang tidak diapa-apakan. Mungkin perbedaan yang mencolok ada di grafik yang dihasilkan saat proses training selesai (di github *folder* "*Balancing Experiment Graph*"), dimana semakin berkurang data nya, semakin lama akurasi dari data validasi untuk meningkat (biasa nya di awal-awal/*epoch* 10). Dugaan kami, kenapa akurasi nya tidak menurun dan malah meningkat, semua nya berasal dari model yang kami buat dan percobaan jumlah *epoch* nya.

3. Jika jumlah *epoch* nya terlalu sedikit, maka terjadilah ***Underfit*** dimana akurasi dari model saat *training* kurang dan saat *testing* hasil prediksi nya juga kurang bagus. Jika jumlah *epoch* nya terlalu banyak, maka terjadilah ***Overfit*** dimana akurasi dari model saat *training* tinggi/bagus, tetapi saat *testing* hasil prediksi nya kurang bagus.

4. Apabila menggunakan parameter nilai *'dropout'* yang terlalu besar, maka akurasi dari prediksi dengan menggunakan gambar *test* akan berkurang. Sama halnya pada saat modifikasi layer, dimana jika *layer-layer* yang dibuat terlalu banyak atau *filter* yang dipakai berlebihan, maka proses *training* akan menjadi lambat dan

memiliki akurasi prediksi yang buruk (dengan kata lain jumlah layer yang digunakan adalah seperlunya saja).