Nama : Reynaldi Rio S Nim : 17102162 Klasifikasi kanker kulit menggunakan metode Convolutional Neural Network (Studi Kasus: Melanoma)

```
import matplotlib.pyplot as plt
import pandas as pd
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Activation, Dropout
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint, Callback, CSVLogger
from keras.models import load_model

import glob
import numpy as np
from keras.preprocessing import image
from keras.models import load_model

#import classification report
from sklearn.metrics import classification_report


size_ = 128

model = Sequential()

model.add(Conv2D(32, (3, 3), input_shape = (size_, size_, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Conv2D(64, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Conv2D(64, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Conv2D(128, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(units =2))
model.add(Dense(units = 2, activation = 'softmax'))

model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])


model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
```

```
conv2d_12 (Conv2D)           (None, 126, 126, 32)      896
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 61, 61, 64)        18496
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 28, 28, 64)        36928
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 12, 12, 128)       73856
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 6, 6, 128)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_9 (Dense)              (None, 128)               589952
_________________________________________________________________
activation_3 (Activation)    (None, 128)               0
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_10 (Dense)             (None, 2)                 258
_________________________________________________________________
dense_11 (Dense)             (None, 2)                 6
=================================================================
Total params: 720,392
Trainable params: 720,392
Non-trainable params: 0
_________________________________________________________________
```

```
getData = ImageDataGenerator()
train_data = getData.flow_from_directory('C:/Users/hp/TA 2/Dataset/Train', target_size = (siz
validation_data = getData.flow_from_directory('C:/Users/hp/TA 2/Dataset/Validation', target_s
```

```
Found 14245 images belonging to 2 classes.
Found 1780 images belonging to 2 classes.
```

```
check_point = ModelCheckpoint('model_epoch-{epoch:d}.h5', period=10)
csv_logger = CSVLogger('training_history.csv')
model.fit(x=train_data, epochs=50, callbacks=[check_point,csv_logger], validation_data=valida
```

```
WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` to specify
Epoch 1/50
100/100 [==============================] - 108s 1s/step - loss: 4.1193 - accuracy: 0.
Epoch 2/50
100/100 [==============================] - 95s 948ms/step - loss: 0.6354 - accuracy: (
Epoch 3/50
100/100 [==============================] - 80s 804ms/step - loss: 0.6197 - accuracy: (
Epoch 4/50
100/100 [==============================] - 67s 666ms/step - loss: 0.6757 - accuracy: (
Epoch 5/50
100/100 [==============================] - 58s 578ms/step - loss: 0.5875 - accuracy: (
```

```
Epoch 6/50
100/100 [==============================] - 55s 547ms/step - loss: 0.5909 - accuracy: 
Epoch 7/50
100/100 [==============================] - 53s 528ms/step - loss: 0.5330 - accuracy: 
Epoch 8/50
100/100 [==============================] - 50s 504ms/step - loss: 0.5304 - accuracy: 
Epoch 9/50
100/100 [==============================] - 50s 497ms/step - loss: 0.4987 - accuracy: 
Epoch 10/50
100/100 [==============================] - 50s 499ms/step - loss: 0.4851 - accuracy: 
Epoch 11/50
100/100 [==============================] - 48s 479ms/step - loss: 0.4738 - accuracy: 
Epoch 12/50
100/100 [==============================] - 48s 476ms/step - loss: 0.4434 - accuracy: 
Epoch 13/50
100/100 [==============================] - 48s 479ms/step - loss: 0.4563 - accuracy: 
Epoch 14/50
100/100 [==============================] - 47s 472ms/step - loss: 0.4944 - accuracy: 
Epoch 15/50
100/100 [==============================] - 47s 469ms/step - loss: 0.4578 - accuracy: 
Epoch 16/50
100/100 [==============================] - 47s 469ms/step - loss: 0.4245 - accuracy: 
Epoch 17/50
100/100 [==============================] - 46s 464ms/step - loss: 0.4064 - accuracy: 
Epoch 18/50
100/100 [==============================] - 47s 467ms/step - loss: 0.4306 - accuracy: 
Epoch 19/50
100/100 [==============================] - 47s 467ms/step - loss: 0.4509 - accuracy: 
Epoch 20/50
100/100 [==============================] - 47s 474ms/step - loss: 0.4269 - accuracy: 
Epoch 21/50
100/100 [==============================] - 47s 469ms/step - loss: 0.4211 - accuracy: 
Epoch 22/50
100/100 [==============================] - 47s 468ms/step - loss: 0.4076 - accuracy: 
Epoch 23/50
100/100 [==============================] - 47s 469ms/step - loss: 0.3996 - accuracy: 
Epoch 24/50
100/100 [==============================] - 47s 473ms/step - loss: 0.3676 - accuracy: 
Epoch 25/50
100/100 [==============================] - 47s 469ms/step - loss: 0.3418 - accuracy: 
Epoch 26/50
100/100 [==============================] - 47s 471ms/step - loss: 0.3681 - accuracy: 
Epoch 27/50
100/100 [==============================] - 47s 466ms/step - loss: 0.3313 - accuracy: 
Epoch 28/50
100/100 [==============================] - 47s 474ms/step - loss: 0.3322 - accuracy: 
Epoch 29/50
```

```
data_history = pd.read_csv('training_history.csv', sep=',')
```

```
data_history
```

	epoch	accuracy	loss	val_accuracy	val_loss
0	0	0.558777	4.119327	0.709375	0.626683
1	1	0.617813	0.635407	0.712500	0.549132
2	2	0.656563	0.619734	0.549375	0.672475
3	3	0.564135	0.675670	0.709375	0.625580
4	4	0.668750	0.587487	0.708750	0.555900
5	5	0.682004	0.590941	0.746250	0.522380
6	6	0.724236	0.533026	0.751250	0.499875
7	7	0.732813	0.530447	0.716250	0.566461
8	8	0.764063	0.498697	0.786875	0.441363
9	9	0.770000	0.485131	0.784375	0.459627
10	10	0.773438	0.473755	0.793750	0.448876
11	11	0.789159	0.443391	0.812500	0.446043
12	12	0.777500	0.456328	0.808750	0.406082
13	13	0.746250	0.494443	0.794375	0.491336
14	14	0.774976	0.457795	0.780625	0.450826
15	15	0.799874	0.424497	0.816250	0.396007
16	16	0.809687	0.406413	0.803125	0.415452
17	17	0.802813	0.430616	0.818125	0.389180
18	18	0.796092	0.450916	0.801250	0.429773
19	19	0.796562	0.426887	0.798750	0.432219
20	20	0.807813	0.421115	0.822500	0.376552
21	21	0.820625	0.407608	0.828750	0.378875
22	22	0.823750	0.399614	0.843750	0.338335
23	23	0.835625	0.367583	0.863750	0.327217
24	24	0.843750	0.341826	0.863125	0.322203
25	25	0.838750	0.368095	0.858125	0.317879
26	26	0.854396	0.331329	0.877500	0.299320
27	27	0.851875	0.332233	0.867500	0.304827
28	28	0.829184	0.396759	0.828125	0.386869
29	29	0.817523	0.387297	0.822500	0.450815

30	30	0.856875	0.333955	0.875000	0.290395
31	31	0.859439	0.330269	0.869375	0.292049
32	32	0.878750	0.298082	0.878125	0.296556
33	33	0.884375	0.269267	0.865000	0.339827
34	34	0.876562	0.295877	0.855625	0.357001
35	35	0.882812	0.279775	0.888125	0.299450
36	36	0.886543	0.284780	0.901250	0.231007
37	37	0.898204	0.242503	0.875000	0.310842
38	38	0.897500	0.244296	0.880625	0.293633
39	39	0.879609	0.265896	0.898750	0.227620
40	40	0.905767	0.232334	0.890000	0.261613
41	41	0.903750	0.228760	0.894375	0.253151
42	42	0.897888	0.241994	0.894375	0.229884
43	43	0.893750	0.247978	0.908125	0.225517
44	44	0.897500	0.252260	0.903125	0.225084
45	45	0.914688	0.225868	0.877500	0.281045
46	46	0.874251	0.295150	0.881250	0.274773
47	47	0.912500	0.217293	0.900000	0.231760

```
data_history = data_history.loc[[9,19,29,39,49]]

data_history.to_csv('training_history.csv', index=False)
#menyimpan ke csv


data_history = pd.read_csv('training_history.csv', sep=',')

data_history
```

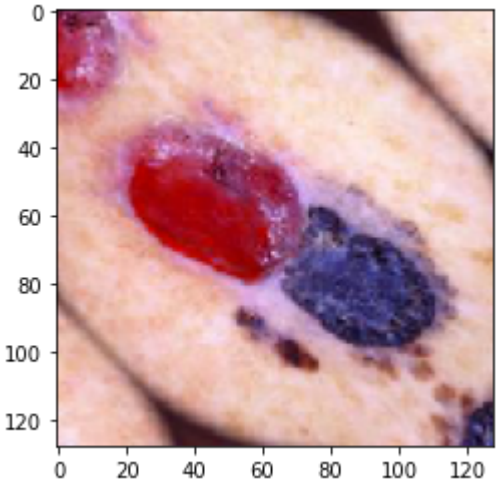
	epoch	accuracy	loss	val_accuracy	val_loss
0	9	0.770000	0.485131	0.784375	0.459627
1	19	0.796562	0.426887	0.798750	0.432219
2	29	0.817523	0.387297	0.822500	0.450815
3	39	0.879609	0.265896	0.898750	0.227620
4	49	0.897187	0.249173	0.908750	0.246174

```
#TestingMulaiDariSini
```

```
size_ = 128

input_melanoma = "C:/Users/hp/TA 2/Dataset/Test/Melanoma"
input_not_melanoma = "C:/Users/hp/TA 2/Dataset/Test/Not Melanoma"

model_e10 = load_model('model_epoch-10.h5')
model_e20 = load_model('model_epoch-20.h5')
model_e30 = load_model('model_epoch-30.h5')
model_e40 = load_model('model_epoch-40.h5')
model_e50 = load_model('model_epoch-50.h5')


#Prediksi 1 gambar melanoma

path = "C:/Users/hp/TA 2/Dataset/Test/Melanoma/AUG_0_1721.jpeg"

img_a = image.load_img(path, target_size = (size_, size_))

img = image.img_to_array(img_a)
img = np.expand_dims(img, axis = 0)

hasil_e10 = np.argmax((model_e10.predict(img)), axis=-1)
hasil_e20 = np.argmax((model_e20.predict(img)), axis=-1)
hasil_e30 = np.argmax((model_e30.predict(img)), axis=-1)
hasil_e40 = np.argmax((model_e40.predict(img)), axis=-1)
hasil_e50 = np.argmax((model_e50.predict(img)), axis=-1)

 #nampilno gambar
plt.imshow(img_a)
plt.show()

#nampilno prediksine maring kelas
if(hasil_e10 == 0):
    print("Diprediksi Melanoma")
elif(hasil_e10 == 1):
    print("Diprediksi Bukan Melanoma")

if(hasil_e20 == 0):
    print("Diprediksi Melanoma")
elif(hasil_e20 == 1):
    print("Diprediksi Bukan Melanoma")

if(hasil_e30 == 0):
    print("Diprediksi Melanoma")
elif(hasil_e30 == 1):
    print("Diprediksi Bukan Melanoma")

if(hasil_e40 == 0):
    print("Diprediksi Melanoma")
elif(hasil_e40 == 1):
    print("Diprediksi Bukan Melanoma")
```

```
if(hasil_e50 == 0):
    print("Diprediksi Melanoma")
elif(hasil_e50 == 1):
    print("Diprediksi Bukan Melanoma")
```

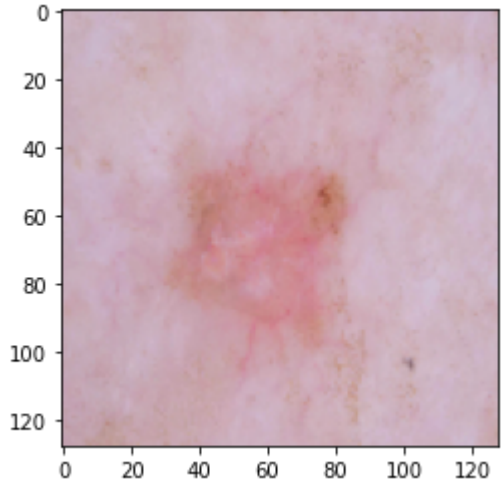
```
 WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<l
```




```
Diprediksi Melanoma
Diprediksi Melanoma
Diprediksi Bukan Melanoma
Diprediksi Melanoma
Diprediksi Melanoma
```

```
#Prediksi 1 gambar bukan melanoma

path = "C:/Users/hp/TA 2/Dataset/Test/Not Melanoma/ISIC_0024946.jpeg"

img_a = image.load_img(path, target_size = (size_, size_))

img = image.img_to_array(img_a)
img = np.expand_dims(img, axis = 0)

#prediksi
hasil_e10 = np.argmax((model_e10.predict(img)), axis=-1)
hasil_e20 = np.argmax((model_e20.predict(img)), axis=-1)
hasil_e30 = np.argmax((model_e30.predict(img)), axis=-1)
hasil_e40 = np.argmax((model_e40.predict(img)), axis=-1)
hasil_e50 = np.argmax((model_e50.predict(img)), axis=-1)

 #nampilno gambar
plt.imshow(img_a)
plt.show()

#nampilno prediksine maring kelas
if(hasil_e10 == 0):
    print("Diprediksi Melanoma")
elif(hasil_e10 == 1):
    print("Diprediksi Bukan Melanoma")
```

```
if(hasil_e20 == 0):
    print("Diprediksi Melanoma")
elif(hasil_e20 == 1):
    print("Diprediksi Bukan Melanoma")

if(hasil_e30 == 0):
    print("Diprediksi Melanoma")
elif(hasil_e30 == 1):
    print("Diprediksi Bukan Melanoma")

if(hasil_e40 == 0):
    print("Diprediksi Melanoma")
elif(hasil_e40 == 1):
    print("Diprediksi Bukan Melanoma")

if(hasil_e50 == 0):
    print("Diprediksi Melanoma")
elif(hasil_e50 == 1):
    print("Diprediksi Bukan Melanoma")
```




```
Diprediksi Bukan Melanoma
Diprediksi Bukan Melanoma
Diprediksi Bukan Melanoma
Diprediksi Bukan Melanoma
Diprediksi Bukan Melanoma
```

```
#PREDICT MELANOMA
e10=[]
e20=[]
e30=[]
e40=[]
e50=[]

for img in glob.glob(input_melanoma + "/*.jpeg"):

    img_a = image.load_img(img, target_size = (size_, size_))

    img_ = image.img_to_array(img_a)
```

```
        img = image.img_to_array(img_a)
        img = np.expand_dims(img, axis = 0)

        hasil_e10 = np.argmax((model_e10.predict(img)), axis=-1)
        hasil_e20 = np.argmax((model_e20.predict(img)), axis=-1)
        hasil_e30 = np.argmax((model_e30.predict(img)), axis=-1)
        hasil_e40 = np.argmax((model_e40.predict(img)), axis=-1)
        hasil_e50 = np.argmax((model_e50.predict(img)), axis=-1)

        e10.append(hasil_e10)
        e20.append(hasil_e20)
        e30.append(hasil_e30)
        e40.append(hasil_e40)
        e50.append(hasil_e50)

    pred_melanoma = [e10,e20,e30,e40,e50]


    #PREDICT NOT MELANOMA
    e10=[]
    e20=[]
    e30=[]
    e40=[]
    e50=[]

    for img in glob.glob(input_not_melanoma + "/*.jpeg"):

        img_a = image.load_img(img, target_size = (size_, size_))

        img = image.img_to_array(img_a)
        img = np.expand_dims(img, axis = 0)

        hasil_e10 = np.argmax((model_e10.predict(img)), axis=-1)
        hasil_e20 = np.argmax((model_e20.predict(img)), axis=-1)
        hasil_e30 = np.argmax((model_e30.predict(img)), axis=-1)
        hasil_e40 = np.argmax((model_e40.predict(img)), axis=-1)
        hasil_e50 = np.argmax((model_e50.predict(img)), axis=-1)

        e10.append(hasil_e10)
        e20.append(hasil_e20)
        e30.append(hasil_e30)
        e40.append(hasil_e40)
        e50.append(hasil_e50)

    pred_not_melanoma = [e10,e20,e30,e40,e50]


    data_predict = pd.DataFrame(index=["10-Epoch","20-Epoch","30-Epoch","40-Epoch","50-Epoch"])

    data_predict_melanoma = pd.DataFrame(pred_melanoma, index=["10-Epoch","20-Epoch","30-Epoch","

    data_predict_melanoma
```

	0	1	2	3	4	5	6	7	8	9	...	880	881	882	883	884	885	886	8
10-Epoch	[0]	[0]	[0]	[0]	[0]	[0]	[0]	[1]	[0]	[0]	...	[0]	[0]	[1]	[0]	[1]	[0]	[0]	
20-Epoch	[1]	[0]	[0]	[1]	[0]	[0]	[0]	[1]	[0]	[0]	...	[0]	[1]	[1]	[0]	[1]	[1]	[0]	
30-Epoch	[0]	[0]	[0]	[0]	[0]	[0]	[0]	[1]	[0]	[0]	...	[0]	[1]	[1]	[0]	[1]	[1]	[1]	
40-Epoch	[0]	[0]	[0]	[0]	[0]	[0]	[0]	[0]	[0]	[0]	...	[0]	[0]	[1]	[0]	[1]	[0]	[1]	

```
data_predict_not_melanoma = pd.DataFrame(pred_not_melanoma, index=["10-Epoch","20-Epoch","30-

data_predict_not_melanoma
```

	0	1	2	3	4	5	6	7	8	9	...	880	881	882	883	884	885	886	8
10-Epoch	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	...	[0]	[0]	[0]	[1]	[0]	[1]	[1]	
20-Epoch	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	...	[1]	[0]	[0]	[1]	[1]	[1]	[1]	
30-Epoch	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	...	[0]	[1]	[1]	[1]	[1]	[1]	[1]	
40-Epoch	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	...	[0]	[1]	[1]	[1]	[1]	[1]	[1]	

```
#Diprediksi Melanoma
data_predict['TP'] = (data_predict_melanoma == 0).sum(axis=1)
data_predict['FP'] = (data_predict_melanoma == 1).sum(axis=1)

data_predict
```

	TP	FP
10-Epoch	713	177
20-Epoch	611	279
30-Epoch	633	257
40-Epoch	833	57
50-Epoch	798	92

```
#Diprediksi Bukan Melanoma
data_predict['TN'] = (data_predict_not_melanoma == 1).sum(axis=1)
data_predict['FN'] = (data_predict_not_melanoma == 0).sum(axis=1)
```

```
data_predict
```

	TP	FP	TN	FN
10-Epoch	713	177	708	182
20-Epoch	611	279	818	72
30-Epoch	633	257	844	46
40-Epoch	833	57	778	112
50-Epoch	798	92	830	60

```
#AKURASI CONFUSION MATRIX
data_predict['Akurasi'] = (data_predict['TN'] + data_predict['TP'])/(data_predict['FP'] + dat

data_predict
```

	TP	FP	TN	FN	Akurasi
10-Epoch	713	177	708	182	0.798315
20-Epoch	611	279	818	72	0.802809
30-Epoch	633	257	844	46	0.829775
40-Epoch	833	57	778	112	0.905056
50-Epoch	798	92	830	60	0.914607

```
data_predict.to_csv('training_pred_conf.csv', index=False)

data_predict = pd.read_csv('training_pred_conf.csv', sep=',')

data_predict
```

	TP	FP	TN	FN	Akurasi
0	713	177	708	182	0.798315
1	611	279	818	72	0.802809
2	633	257	844	46	0.829775
3	833	57	778	112	0.905056
4	798	92	830	60	0.914607

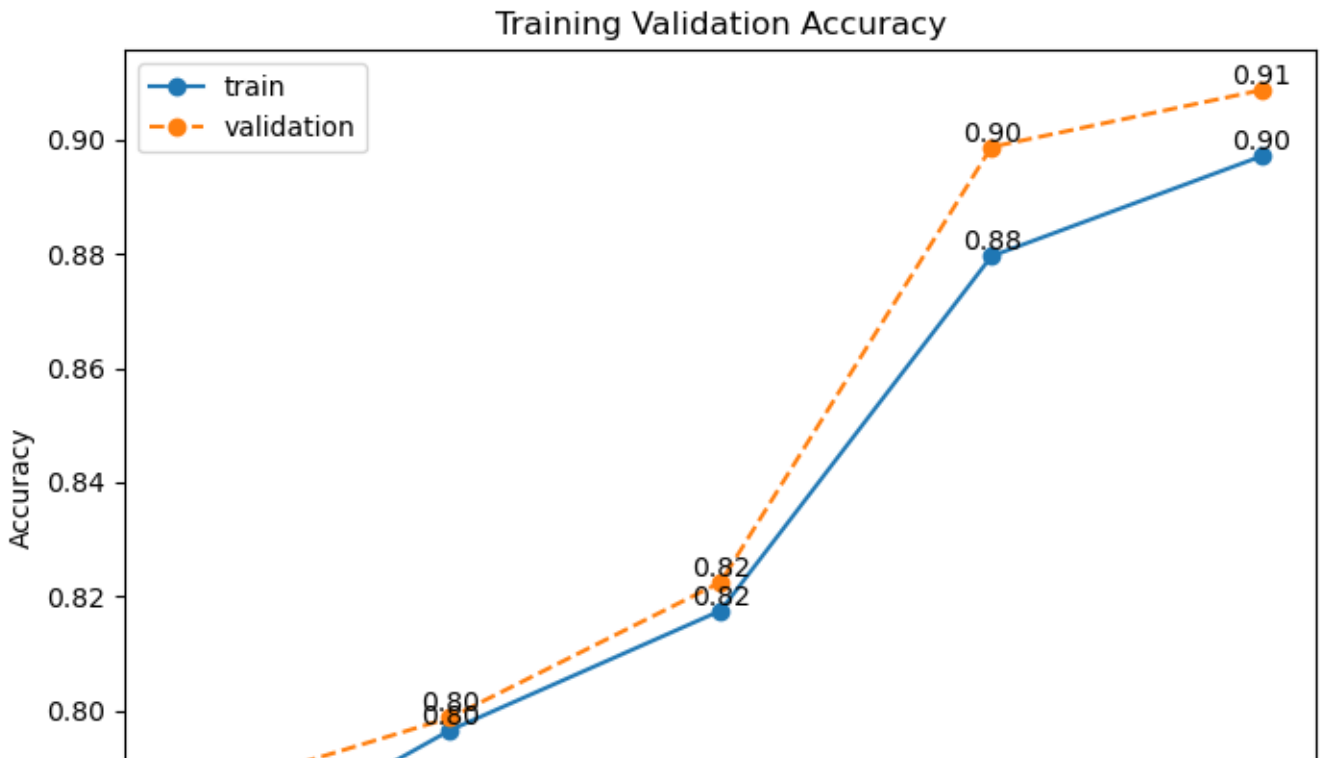
```
data_history = pd.read_csv('training_history.csv', sep=',')
prediction_data = data_history.merge(data_predict, left_index=True, right_index=True)
prediction_data.insert(loc=0,column='Epoch', value=["10-Epoch","20-Epoch","30-Epoch","40-Epoc
prediction_data = prediction_data.drop(labels='epoch', axis=1)
```

```
prediction_data = prediction_data.drop(labels= 'Epoch', axis=1)
```

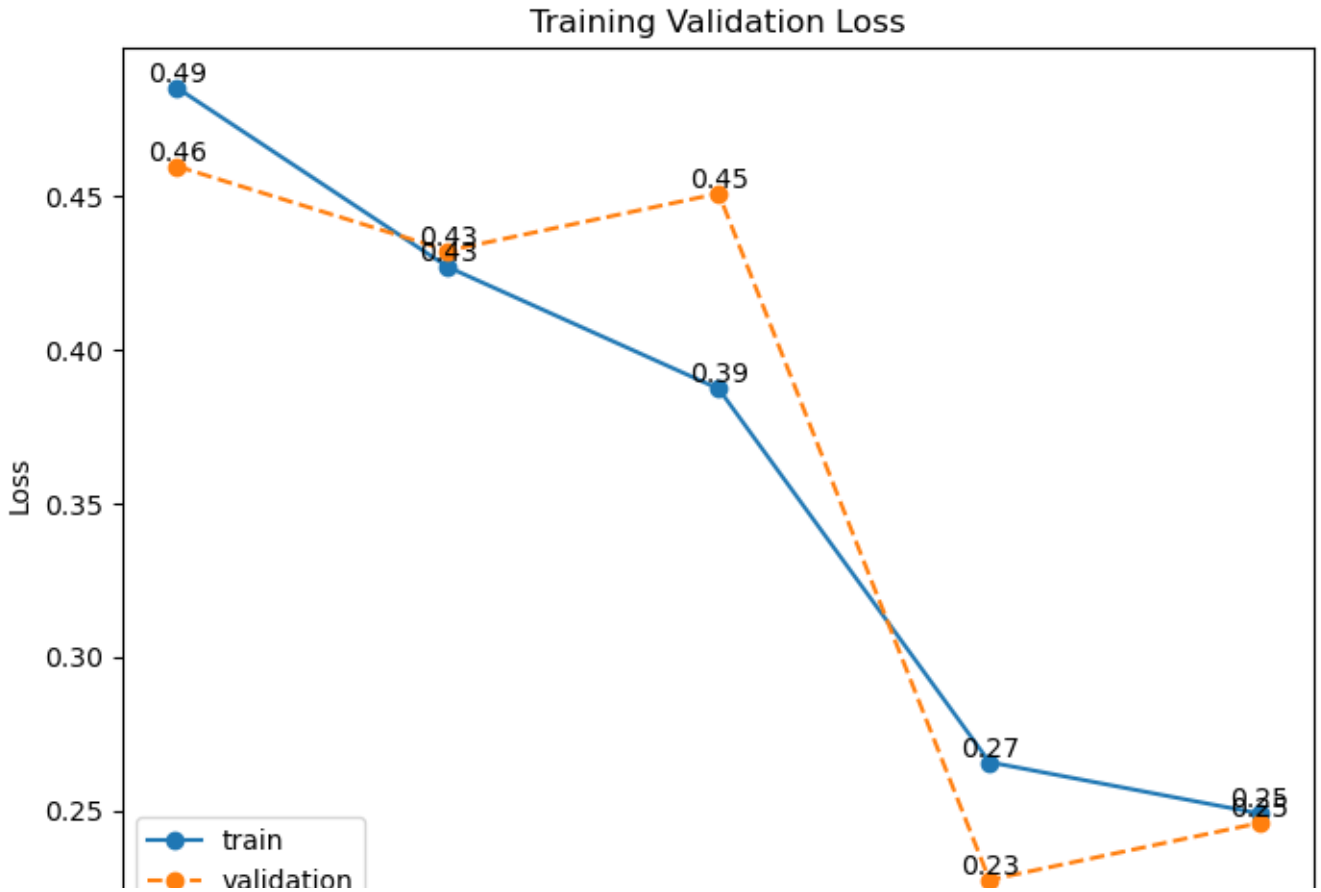
```
prediction_data
```

	Epoch	accuracy	loss	val_accuracy	val_loss	TP	FP	TN	FN	Akurasi
0	10-Epoch	0.770000	0.485131	0.784375	0.459627	713	177	708	182	0.798315
1	20-Epoch	0.796562	0.426887	0.798750	0.432219	611	279	818	72	0.802809
2	30-Epoch	0.817523	0.387297	0.822500	0.450815	633	257	844	46	0.829775
3	40-Epoch	0.879609	0.265896	0.898750	0.227620	833	57	778	112	0.905056
4	50-Epoch	0.897187	0.249173	0.908750	0.246174	798	92	830	60	0.914607

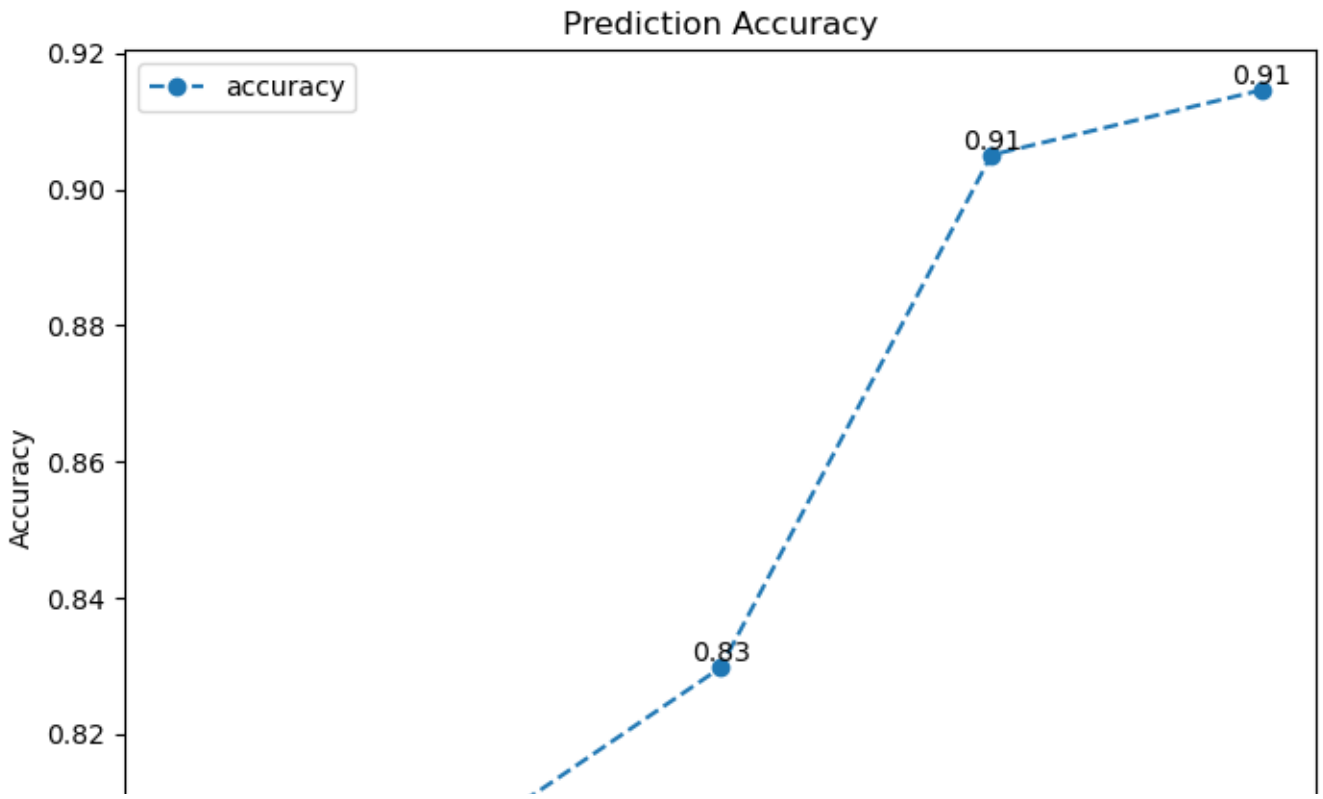
```
prediction_data.to_csv('training_history_pred_conf.csv', index=False)


#GrafikMulaiDariSini
data = pd.read_csv('training_history_pred_conf.csv', sep=',')

plt.figure(figsize=(8,6),dpi=100)

accuracy = data['accuracy']
val_accuracy = data['val_accuracy']
labels = data['Epoch']

plt.plot(accuracy, linestyle='-', marker='o')
plt.plot(val_accuracy, linestyle='--', marker='o')
plt.title('Training Validation Accuracy')
plt.ylabel('Accuracy')
plt.xticks(data.index,labels, rotation=45)
plt.legend(['train','validation'], loc='upper left')

for index,data in enumerate(accuracy):
    plt.text(x = index , y = data , s=f"{'{:.2f}'.format(data)}", va = 'bottom', ha='center')

for index,data in enumerate(val_accuracy):
    plt.text(x = index , y = data , s=f"{'{:.2f}'.format(data)}", va = 'bottom', ha='center')

plt.show()
```




```
data = pd.read_csv('training_history_pred_conf.csv', sep=',')

plt.figure(figsize=(8,6),dpi=100)

loss = data['loss']
val_loss = data['val_loss']
labels = data['Epoch']

plt.plot(loss, linestyle='-', marker='o')
plt.plot(val_loss, linestyle='--', marker='o')
plt.title('Training Validation Loss')
plt.ylabel('Loss')
plt.xticks(data.index,labels, rotation=45)
plt.legend(['train','validation'], loc='lower left')

for index,data in enumerate(loss):
    plt.text(x = index , y = data , s=f"{'{:.2f}'.format(data)}", va = 'bottom', ha='center')

for index,data in enumerate(val_loss):
    plt.text(x = index , y = data , s=f"{'{:.2f}'.format(data)}", va = 'bottom', ha='center')


plt.show()
```




```
data = pd.read_csv('training_history_pred_conf.csv', sep=',')

plt.figure(figsize=(8,6),dpi=100)

predict_acc = data['Akurasi']
labels = data['Epoch']

plt.plot(predict_acc, linestyle='--', marker='o')
plt.title('Prediction Accuracy')
plt.ylabel('Accuracy')
plt.xticks(data.index,labels, rotation=45)
plt.legend(['accuracy'], loc='upper left')

for index,data in enumerate(predict_acc):
    plt.text(x = index , y = data , s=f"{'{:.2f}'.format(data)}", va = 'bottom', ha='center')

plt.show()
```




```
#Classification Report

datagen = ImageDataGenerator()
data_test = datagen.flow_from_directory('C:/Users/hp/TA 2/Dataset/Test', shuffle=False, batch
```

```
Found 1780 images belonging to 2 classes.
```

```
probas = model_e10.predict(data_test)
y_pred = np.argmax(probas, axis=1)
y_true = data_test.classes
print(classification_report(y_true,y_pred))
```

```
                  precision    recall  f1-score   support

               0       0.80      0.80      0.80       890
               1       0.80      0.80      0.80       890

        accuracy                           0.80      1780
       macro avg       0.80      0.80      0.80      1780
    weighted avg       0.80      0.80      0.80      1780
```

```
probas = model_e20.predict(data_test)
y_pred = np.argmax(probas, axis=1)
y_true = data_test.classes
print(classification_report(y_true,y_pred))
```

```
                  precision    recall  f1-score   support
```

```
                0       0.89      0.69      0.78       890
                1       0.75      0.92      0.82       890

         accuracy                           0.80      1780
        macro avg       0.82      0.80      0.80      1780
     weighted avg       0.82      0.80      0.80      1780
```

```
probas = model_e30.predict(data_test)
y_pred = np.argmax(probas, axis=1)
y_true = data_test.classes
print(classification_report(y_true,y_pred))
```

```
                  precision    recall  f1-score   support

                0       0.93      0.71      0.81       890
                1       0.77      0.95      0.85       890

         accuracy                           0.83      1780
        macro avg       0.85      0.83      0.83      1780
     weighted avg       0.85      0.83      0.83      1780
```

```
probas = model_e40.predict(data_test)
y_pred = np.argmax(probas, axis=1)
y_true = data_test.classes
print(classification_report(y_true,y_pred))
```

```
                  precision    recall  f1-score   support

                0       0.88      0.94      0.91       890
                1       0.93      0.87      0.90       890

         accuracy                           0.91      1780
        macro avg       0.91      0.91      0.90      1780
     weighted avg       0.91      0.91      0.90      1780
```

```
probas = model_e50.predict(data_test)
y_pred = np.argmax(probas, axis=1)
y_true = data_test.classes
print(classification_report(y_true,y_pred))
```

```
                  precision    recall  f1-score   support

                0       0.93      0.90      0.91       890
                1       0.90      0.93      0.92       890

         accuracy                           0.91      1780
        macro avg       0.92      0.91      0.91      1780
     weighted avg       0.92      0.91      0.91      1780
```