Lampiran 2

Coding aplikasi

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession

config = ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = 0.5
config.gpu_options.allow_growth = True
session = InteractiveSession(config=config)
```

```
from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
# Helper libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import glob
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
```

```
#Build The CNN
model = Sequential() #Create the architecture
model.add(Conv2D(64, (5, 5), activation='relu',
input_shape=(128,128,3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (5, 5), activation='relu'))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

```
# import the libraries as shown below
import tensorflow as tf
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.inception_v3 import InceptionV3
#from keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.inception_v3 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
#import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow.keras.optimizers import RMSprop
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline
import pandas as pd
from tensorflow.keras.preprocessing.image import img_to_array, load_img
import random

import tensorflow as tf
import tensorflow.keras.layers as Layers
import tensorflow.keras.activations as Activations
import tensorflow.keras.models as Models
import tensorflow.keras.optimizers as Optimizers
import tensorflow.keras.metrics as Metrics
import tensorflow.keras.utils as Utils
import pandas as pd
import tensorflow.keras.backend as K
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras import regularizers
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.callbacks import ModelCheckpoint, CSVLogger
from tensorflow.keras.optimizers import SGD
```

```
[ ] # re-size all the images to this
    IMAGE_SIZE = [100, 100]

    valid_path = '/content/drive/MyDrive/dataset/test'
    valid_path = '/content/drive/MyDrive/dataset/train'
```

```
[ ]
    # Here we will be using imagenet weights
    inception = InceptionV3(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

```
[ ] class InceptionV3():
        @staticmethod
        def build(numChannels, imgRows, imgCols, numClasses,  pooling= "max", activation= "relu"):
            # initialize the model
            model = Sequential()
            inputShape = (imgRows, imgCols, numChannels)

            # add first set of layers: Conv -> Activation -> Pool
            model.add(Conv2D(filters= 6, kernel_size= 5, input_shape= inputShape))
            model.add(Activation(activation))

            if pooling == "max":
                model.add(MaxPooling2D(pool_size= (3, 3), strides= (2, 2)))
            else:
                model.add(AveragePooling2D(pool_size= (7, 7), strides= (2, 2)))

            # add second set of layers: Conv -> Activation -> Pool
            model.add(Conv2D(filters= 16, kernel_size= 5,))
            model.add(Activation(activation))

            if pooling == "avg":
                model.add(AveragePooling2D(pool_size=(7, 7), strides=(2, 2)))
            else:
                model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))

            # Flatten -> FC 120 -> Dropout -> Activation
            model.add(Flatten())
            model.add(Dense(64))
            model.add(Dropout(0.5))
            model.add(Activation(activation))

            # FC 84 -> Dropout -> Activation
            model.add(Dense(32))
            model.add(Dropout(0.5))
            model.add(Activation(activation))

            # FC 4-> Softmax
            model.add(Dense(numClasses))
            model.add(Activation("softmax"))

            return model
```

```
[ ] # don't train existing weights
    for layer in inception.layers:
        layer.trainable = False
```

```
[ ]   # useful for getting number of output classes
    folders = glob('/content/drive/MyDrive/dataset/train/*')
```

```
[ ] folders
```

```
['/content/drive/MyDrive/dataset/train/late_blight',
 '/content/drive/MyDrive/dataset/train/early_blight']
```

```
[ ] folders = glob('/content/drive/MyDrive/dataset/test/*')
```

```
[ ] folders
```

```
['/content/drive/MyDrive/dataset/test/late_blight',
 '/content/drive/MyDrive/dataset/test/early_blight']
```

```
[ ] # our layers - you can add more if you want
    x = Flatten()(inception.output)
```

```
[ ] prediction = Dense(len(folders), activation='softmax')(x)

    # create a model object
    model = Model(inputs=inception.input, outputs=prediction)
```

```
[ ] test_set = test_datagen.flow_from_directory('/content/drive/MyDrive/dataset/test',
                                                target_size = (100, 100),
                                                batch_size = 16,
                                                class_mode = 'categorical')
```

```
Found 452 images belonging to 2 classes.
```

```
[ ] test_set.labels
```

```
training_set.labels
```

```
[ ] # fit the model
    # Run the cell. It will take some time to execute
    r = model.fit_generator(
      training_set,
      validation_data=test_set,
      epochs=20,
      steps_per_epoch=len(training_set),
      validation_steps=len(test_set)
    )
```

```
[ ] import matplotlib.pyplot as plt
```

```
# plot the loss
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

# plot the accuracy
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```

```
# view the structure of the model
model.summary()
```

```
[ ] # tell the model what cost and optimization method to use
    model.compile(
      loss='categorical_crossentropy',
      optimizer='adam',
      metrics=['accuracy']
    )
```

```
model.compile(optimizer="sgd",
              loss="categorical_crossentropy",
              metrics=[keras.metrics.Precision(), keras.metrics.Recall(), keras.metrics.SpecificityAtSensitivity(0.5),
                       keras.metrics.SensitivityAtSpecificity(0.5), 'accuracy'])
```

```
[ ] # Use the Image Data Generator to import the images from the dataset
    from tensorflow.keras.preprocessing.image import ImageDataGenerator

    train_datagen = ImageDataGenerator(rescale = 1./255,
                                       shear_range = 0.2,
                                       zoom_range = 0.2,
                                       horizontal_flip = True)

    test_datagen = ImageDataGenerator(rescale = 1./255,
                                      shear_range = 0.2,
                                      zoom_range = 0.2,
                                      horizontal_flip = True)
```

```
[ ] # Make sure you provide the same target size as initialied for the image size
    training_set = train_datagen.flow_from_directory('/content/drive/MyDrive/dataset/train',
                                                     target_size = (100, 100),
                                                     batch_size = 16,
                                                     class_mode = 'categorical')
```

```
Found 1700 images belonging to 2 classes.
```

```
[ ] test_set = test_datagen.flow_from_directory('/content/drive/MyDrive/dataset/test',
                                                target_size = (100, 100),
                                                batch_size = 16,
```

```
from keras import metrics

model.compile(loss='mean_squared_error', optimizer='sgd',
              metrics=[metrics.mae,
                       metrics.categorical_accuracy])
```

```
[ ] from sklearn import metrics
```

```
[ ] from sklearn.metrics import confusion_matrix
    conf = metrics.confusion_matrix(test_set.labels, y_pred)
    conf
```

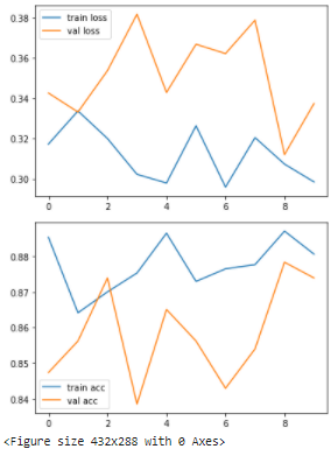
```
[ ] model.compile(optimizer="sgd",
                  loss="categorical_crossentropy",
                  metrics=[keras.metrics.Precision(name='precision'),
                           keras.metrics.Recall(name='recall'),
                           keras.metrics.SpecificityAtSensitivity(0.5, name='specificity_at_sensitivity'),
                           keras.metrics.SensitivityAtSpecificity(0.5, name='sensitivity_at_specificity'),
                           'accuracy'])
```

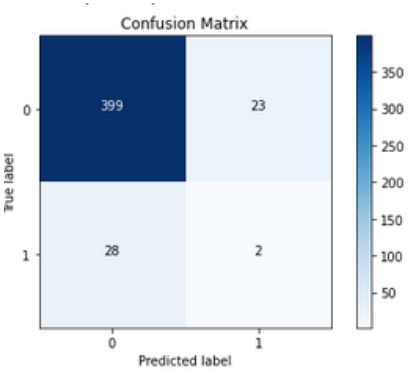
```
import itertools
classes = [0, 1, 2]
# plot confusion matrix
plt.imshow(conf, interpolation='nearest', cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.colorbar()
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes)
plt.yticks(tick_marks, classes)

fmt = 'd'
thresh = conf.max() / 2.
for i, j in itertools.product(range(conf.shape[0]), range(conf.shape[1])):
    plt.text(j, i, format(conf[i, j], fmt),
             horizontalalignment="center",
             color="black" if conf[i, j] > thresh else "black")

plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

Lampiran 3

Hasil perbandingan

a. Epoch 10, Dropout 0.5, Batch Size 16





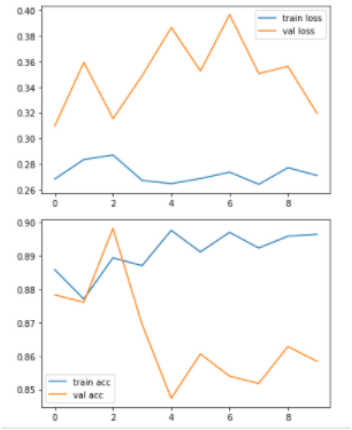
<Figure size 432x288 with 0 Axes>

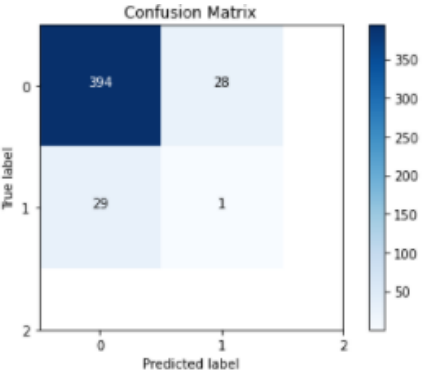
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{399 + 2}{399 + 23 + 28 + 2}$$
$$= \frac{401}{452} = 0.8871 * 100\% = 88.71\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{399}{399 + 23} = \frac{399}{422}$$
$$= 0.9454 * 100\% = 94.54\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{399}{399 + 28} = \frac{399}{427}$$
$$= 0.9344 * 100\% = 93.44\%$$

b. Epoch 10, Dropout 0.5, Batch Size 32



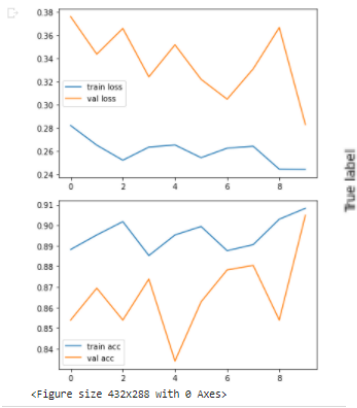


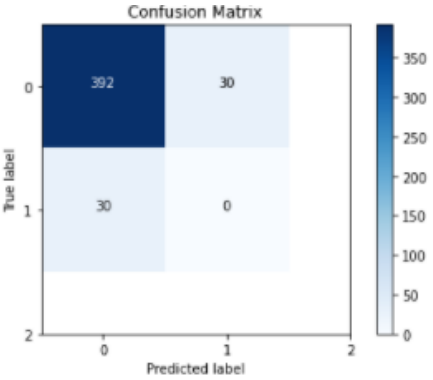
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{392 + 0}{394 + 28 + 29 + 1}$$
$$= \frac{395}{452} = 0.8738 * 100\% = 87.38\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{394}{394 + 28} = \frac{394}{422}$$
$$= 0.9945 * 100\% = 99.45\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{394}{394 + 29} = \frac{394}{427}$$
$$= 0.9314 * 100\% = 93.14\%$$

c. Epoch 10, Dropout 0.5, Batch Size 64



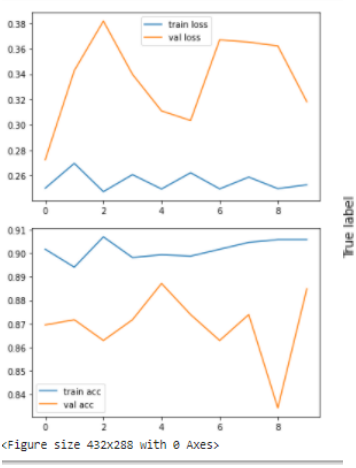


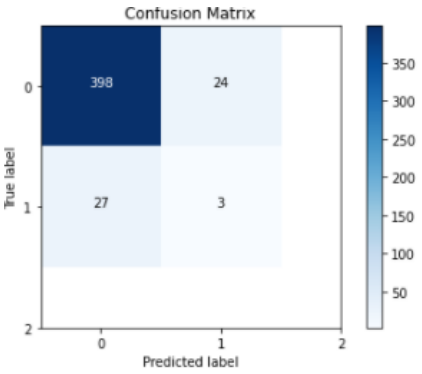
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{392 + 0}{392 + 30 + 30 + 0}$$
$$= \frac{392}{452} = 0.8672 * 100\% = 86.72\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{392}{392 + 30} = \frac{392}{422}$$
$$= 0.9289 * 100\% = 92.89\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{392}{392 + 30} = \frac{392}{422}$$
$$= 0.9289 * 100\% = 92.89\%$$

d. Epoch 10, Dropout 0.5, Batch Size 128



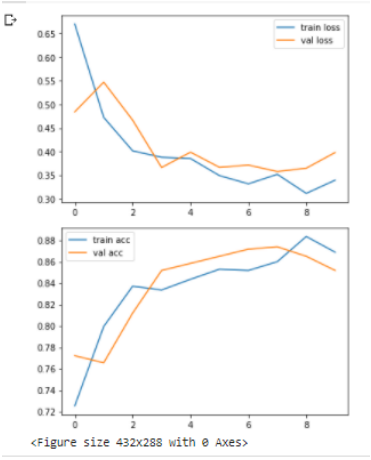


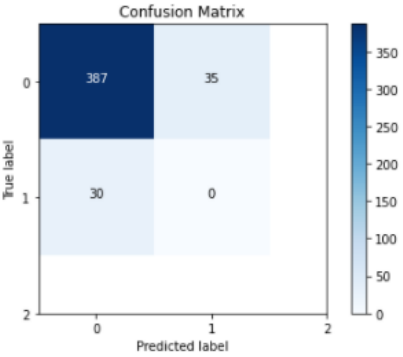
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{398 + 3}{398 + 24 + 27 + 3}$$
$$= \frac{401}{452} = 0.8871 * 100\% = 88.71\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{398}{398 + 24} = \frac{398}{422}$$
$$= 0.9431 * 100\% = 94.31\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{398}{398 + 27} = \frac{398}{425}$$
$$= 0.9364 * 100\% = 93.64\%$$

e. Epoch 10, Dropout 0.6, Batch Size 16



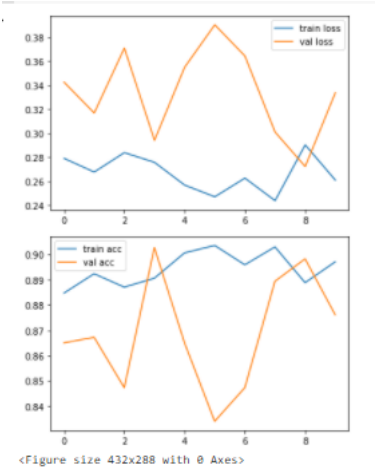


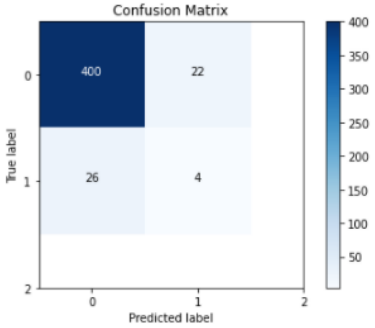
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{387 + 0}{387 + 35 + 30 + 0}$$
$$= \frac{387}{452} = 0.8561 * 100\% = 85.61\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{387}{387 + 35} = \frac{387}{422}$$
$$= 0.9148 * 100\% = 91.48\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{387}{387 + 30} = \frac{387}{427}$$
$$= 0.9063 * 100\% = 90.63\%$$

f. Epoch 10, Dropout 0.6, Batch Size 32



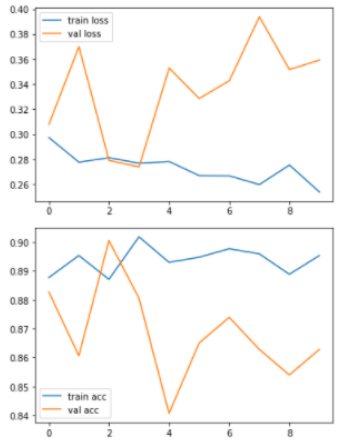


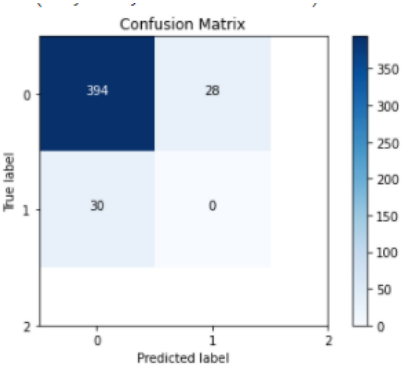
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{400 + 4}{400 + 22 + 26 + 4}$$
$$= \frac{404}{452} = 0.8938 * 100\% = 89.38\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{400}{400 + 22} = \frac{400}{422}$$
$$= 0.93.45 * 100\% = 93.45\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{400}{400 + 2} = \frac{400}{402}$$
$$= 0.94.78 * 100\% = 94.78\%$$

g. Epoch 10, Dropout 0.6, Batch Size 64



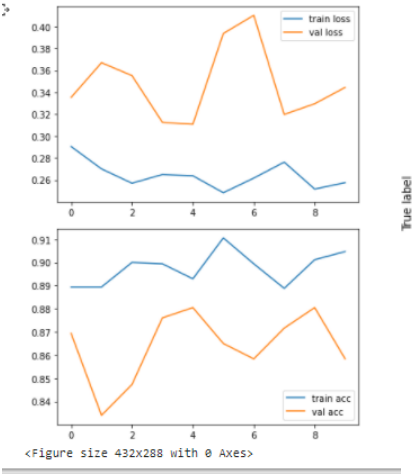


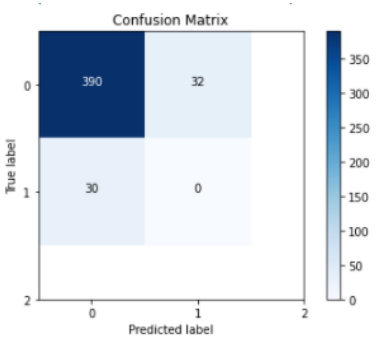
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{394 + 0}{394 + 28 + 30 + 0}$$
$$= \frac{394}{452} = 0.87.16 * 100\% = 87.16\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{394}{394 + 28} = \frac{394}{422}$$
$$= 0.93.36 * 100\% = 93.36\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{394}{394 + 30} = \frac{394}{424}$$
$$= 0.9292 * 100\% = 92.92\%$$

h. Epoch 10, Dropout 0.6, Batch Size 128



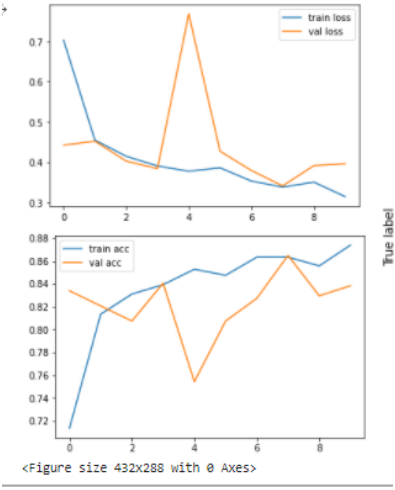


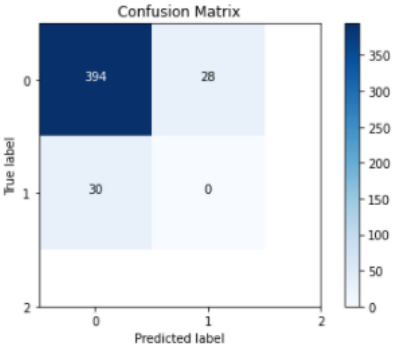
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{390 + 0}{390 + 32 + 30 + 0}$$
$$= \frac{390}{452} = 0.8628 * 100\% = 86.28\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{390}{390 + 32} = \frac{390}{422}$$
$$= 0.9241 * 100\% = 92.41\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{390}{390 + 30} = \frac{390}{420}$$
$$= 0.9285 * 100\% = 92.85\%$$

i. Epoch 10, Dropout 0.7, Batch Size 16



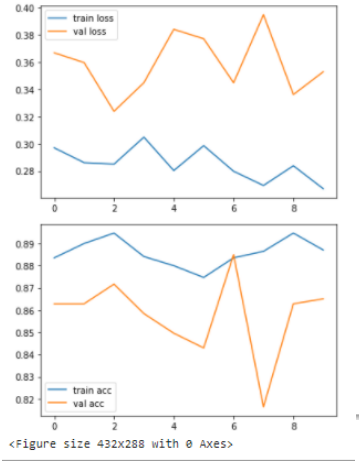


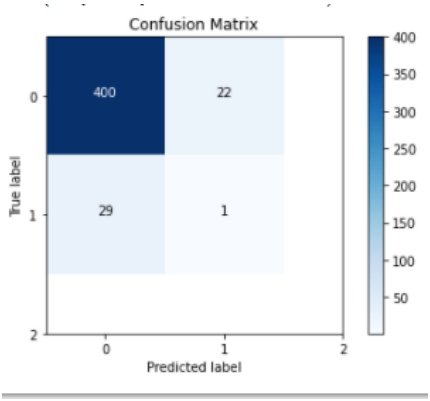
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{394 + 0}{394 + 28 + 30 + 0}$$
$$= \frac{394}{452} = 0.8716 * 100\% = 87.16\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{394}{394 + 28} = \frac{394}{422}$$
$$= 0.9099 * 100\% = 90.99\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{394}{394 + 30} = \frac{394}{424}$$
$$= 0.9292 * 100\% = 92.92\%$$

j. Epoch 10, Dropout 0.7, Batch Size 32



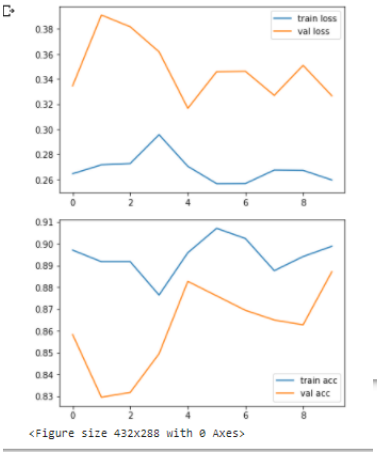


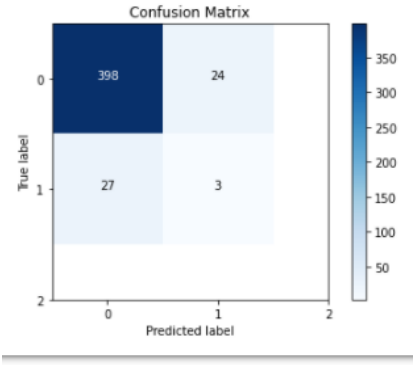
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{400 + 1}{400 + 22 + 29 + 1}$$
$$= \frac{401}{452} = 0.8871 * 100\% = 88.71\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{400}{400 + 22} = \frac{400}{422}$$
$$= 0.93.45 * 100\% = 93.45\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{400}{400 + 29} = \frac{400}{429}$$
$$= 0.9324 * 100\% = 93.24\%$$

k. Epoch 10, Dropout 0.7, Batch Size 64



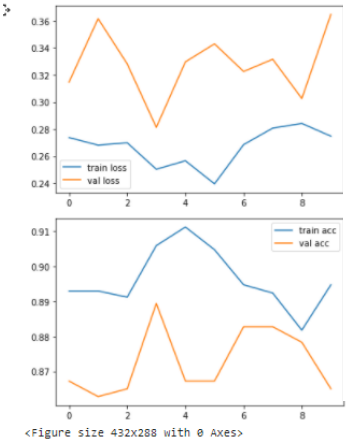


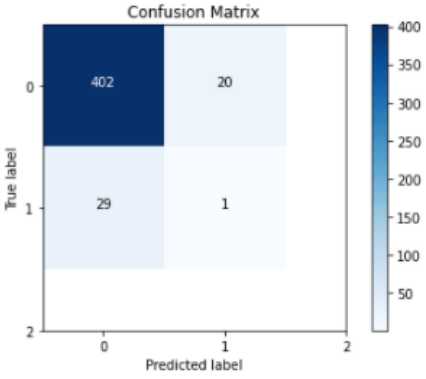
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{398 + 3}{398 + 24 + 27 + 3}$$
$$= \frac{401}{452} = 0.8871 * 100\% = 88.71\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{398}{398 + 24} = \frac{398}{422}$$
$$= 0.9431 * 100\% = 94.31\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{398}{398 + 27} = \frac{398}{435}$$
$$= 0.9364 * 100\% = 93.64\%$$

I. Epoch 10, Dropout 0.7, Batch Size 128





$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{402 + 1}{402 + 20 + 29 + 1}$$
$$= \frac{403}{452} = 0.8915 * 100\% = 89.15\%$$

$$Sensitifitas = \frac{TP}{TP + FP} = \frac{402}{402 + 20} = \frac{402}{422}$$
$$= 0.9526 * 100\% = 95.26\%$$

$$Spesifisitas = \frac{TP}{TP + FN} = \frac{402}{402 + 29} = \frac{402}{431}$$
$$= 0.9327 * 100\% = 93.27\%$$

Lampiran 4

Bukti kartu seminar

KARTU SEMINAR

Nama : DAHLIATUL FITRIYAH NINGSIH
Nim : 201769040016
Prodi : TEKNIK INFORMATIKA
Fakultas : TEKNIK

NO	Tanggal	Judul Seminar yang diikuti	Dosen Pendamping	Tanda Tangan	Keterangan
1	kamis, 19 April 2018	Aplikasi mobile Augmented Reality berbasis android sebagai media pendukung pembelajaran di SMK Pasuruan TKJ	M.Lutfi M.kom		Ulfatunnihdlyah
2	kamis, 19 April 2018	Sigmentasi warna Jaringan Syaraf tiruan untuk menentukan kualitas susu sapi dengan metode BackPropagation	M.Lutfi M.kom.		Fitriatul Laili
3	kamis, 19 April 2018	Aplikasi Pembelajaran ilmu tajwid Berbasis Android menggunakan speech Recognition.	M.Lutfi M.kom		Risyandi Anugrah Hardi
4	kamis, 19 April 2018	Optimasi Parameter Support Vector Machine dengan relief untuk Prediksi Penyakit kanker servics	M. Lutfi M.kom		Azizah Erma Saraswati
5	Rabu, 22 mei 2019	Perancangan Aplikasi Pengaduan Mahasiswa berbasis Android di Univ. Yudharta pasuruan	M. Imron Rosyadi S.kom M.kom		Umi Fatimatuz Zuhdiyah
6	Rabu, 22 mei 2019	Penerapan Algoritma Depth first search dan Backtracking sebagai maze generator pd game fokus labirin	M. Imron Rosyadi S.kom M.kom		Khoirul Habibi
7	Rabu, 22 mei 2019	Implementasi forecasting persediaan bahan baku kertas art paper pada Yudharta advertising menggunakan Arima	— " —		M. Aenur Rofiq
8	7 mei 2020	Implementasi sistem pakan lele menggunakan IoT untuk meningkatkan produk Pokdakan Budaya.	M. Faishol Amrullah, M.kom		Siti Salamah.
9	8 mei 2020	Rancang bangun Game 2 dimensi sejarah IPNU IPPNU menggunakan metode multimedia development life cycle ~~berbasis android~~	Waiidini Syaikhul Huda S.kom., M.kom		Naafilah Rodliyah
10	8 mei 2020	Pengembangan game edukasi tumbuhan 3D Android menggunakan Metode water fall	Walidini Syaikhul H. S.kom. M.kom.		Eni Fadila

Catatan : kartu ini digandakan dan di lampirkan sebagai syarat ujian skripsi
Syarat ujian skripsi Minimal Mengikuti 5 kali Seminar