COMPARISON

IMAGE CLASSIFICATION MODELS

CLOUD IMAGES DATASET

* CLASSIFICATION *

CLOUD IMAGE DATASET

Data Explorer

94.53 MB

- test
  - Cloud
  - RainCloud
- train
  - Cloud
  - RainCloud

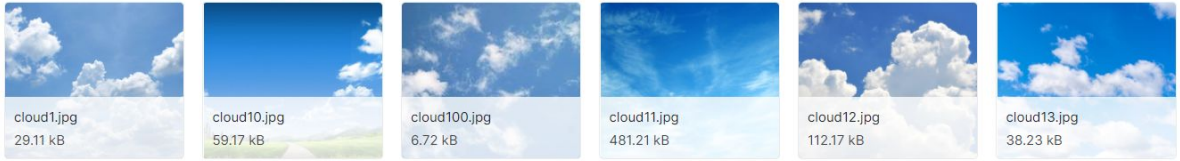
Summary

- 1200 files

Cloud (100 files)

About this directory

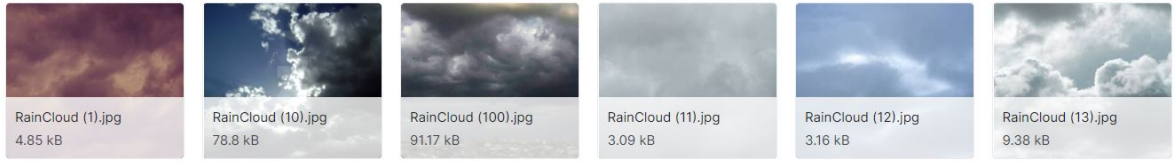
This file does not have a description yet.



RainCloud (100 files)

About this directory

This file does not have a description yet.



SSIFICATION * IMAGE * CLASSIFICATION * IMAGE * CLASSIF

PREPROCESS

ติดตั้ง Kaggle Library

```
[1]  1  !pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages (1.5.12)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from kaggle) (1.24.3)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from kaggle) (2021.5.30)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from kaggle) (2.23.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from kaggle) (4.62.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packages (from kaggle) (1.15.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-packages (from kaggle) (5.0.2)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->kaggle) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->kaggle) (3.0.4)
```

Upload kaggle.json

```
[2]  1  from google.colab import files
     2
     3  files.upload()
```

Choose files kaggle.json

- **kaggle.json**(application/json) - 75 bytes, last modified: 05/09/2021 - 100% done

```
Saving kaggle.json to kaggle.json
{'kaggle.json': b'{"username":"nuttidalapthanachai","key":"ee0a24585b49f8ef1fd3f6eab971ee01"}'}
```

PREPROCESS

สร้างโฟลเดอร์ kaggle เพื่อเก็บไฟล์ kaggle.json และเปลี่ยน Permission ของไฟล์ kaggle.json

```
[3]  1  !mkdir kaggle
```

```
[4]  1  !mv kaggle.json kaggle
```

```
[5]  1  !chmod 600 /content/kaggle/kaggle.json
```

Config Kaggle Environment

```
[6]  1  import os
     2  os.environ['KAGGLE_CONFIG_DIR'] = "/content/kaggle"
```

Download Cloud image dataset

```
[7]  1  !kaggle datasets download -d nuttidalapthanachai/cloud-image-dataset
```

```
Downloading cloud-image-dataset.zip to /content
 94% 83.0M/88.1M [00:00<00:00, 138MB/s]
100% 88.1M/88.1M [00:00<00:00, 137MB/s]
```

สร้างโฟลเดอร์ Cloud-image-dataset แล้ว Unzip Dataset

```
[8]  1  !mkdir cloud-image-dataset && unzip -q cloud-image-dataset.zip -d cloud-image-dataset
```

DATA PREPARATION

Import Library ที่จำเป็น

```
[9]  1  import tensorflow as tf
     2
     3  Adam = tf.keras.optimizers.Adam
     4  to_categorical = tf.keras.utils.to_categorical
     5  ImageDataGenerator = tf.keras.preprocessing.image.ImageDataGenerator
     6  ReduceLROnPlateau = tf.keras.callbacks.ReduceLROnPlateau
     7  load_img = tf.keras.preprocessing.image.load_img
     8  img_to_array = tf.keras.preprocessing.image.img_to_array
     9  array_to_img = tf.keras.preprocessing.image.array_to_img
    10
    11  from sklearn.model_selection import train_test_split
    12  import matplotlib.pyplot as plt
    13
    14  import plotly.graph_objs as go
    15  from plotly import subplots
    16  import plotly
    17
    18  from tensorflow.keras.models import Sequential
    19  from tensorflow.keras.layers import Dense, GRU, LSTM, Bidirectional, Embedding, Dropout, BatchNormalization
    20  from tensorflow.keras.models import load_model
    21  from tensorflow.keras.callbacks import ModelCheckpoint
    22
    23  import pandas as pd
    24  from sklearn.metrics import classification_report
    25  from sklearn.metrics import confusion_matrix
    26  import pickle as p
    27
    28  import warnings
    29  warnings.filterwarnings('ignore')
```

DATA PREPARATION

กำหนด parameter

```
[10]  1  img_rows = 32
      2  img_cols = 32
      3  EPOCHS = 100
      4  BATCH_SIZE = 32
```

Load Dataset

```
[11]  1  import cv2
      2  import os
      3  import random
      4  import matplotlib.pyplot as plt
      5  import numpy as np
```

```
[12]  1  DIRECTORY_TRAIN = r'/content/cloud-image-dataset/train'
      2  DIRECTORY_TEST = r'/content/cloud-image-dataset/test'
      3  CATEGORIES = ['Cloud','RainCloud']
```

DATA PREPARATION

สร้างตัวแปรTrain_data และ Test_data เพื่อเก็บข้อมูล

```
[13]  1  Train_data = []
      2  Test_data = []
      3
      4  for category in CATEGORIES:
      5    folder = os.path.join(DIRECTORY_TRAIN, category)
      6    label = CATEGORIES.index(category)
      7    for img in os.listdir(folder):
      8      img_path = os.path.join(folder, img)
      9      img_arr = cv2.imread(img_path)
     10      img_arr = cv2.resize(img_arr, (img_rows, img_cols))
     11      img_arr = cv2.cvtColor(img_arr, cv2.COLOR_BGR2GRAY)
     12      Train_data.append([img_arr, label])
     13
     14  for category in CATEGORIES:
     15    folder = os.path.join(DIRECTORY_TEST, category)
     16    label = CATEGORIES.index(category)
     17    for img in os.listdir(folder):
     18      img_path = os.path.join(folder, img)
     19      img_arr = cv2.imread(img_path)
     20      img_arr = cv2.resize(img_arr, (img_rows, img_cols))
     21      img_arr = cv2.cvtColor(img_arr, cv2.COLOR_BGR2GRAY)
     22      Test_data.append([img_arr, label])
```

```
[14]  1  print(len(Train_data))
      2  print(len(Test_data))
```

```
1000
200
```

```
[15]  1  random.shuffle(Train_data)
      2  random.shuffle(Test_data)
```

DATA PREPARATION

แยกข้อมูลภาพกับผลเฉลย

```
[16]  1  train_data = []
      2  y = []
      3  test_data = []
      4  y_test = []
      5
      6  for features, labels in Train_data:
      7    train_data.append(features)
      8    y.append(labels)
      9
     10  for features, labels in Test_data:
     11    test_data.append(features)
     12    y_test.append(labels)
```

```
[17]  1  train_data = np.array(train_data)
      2  y = np.array(y)
      3  test_data = np.array(test_data)
      4  y_test = np.array(y_test)
```

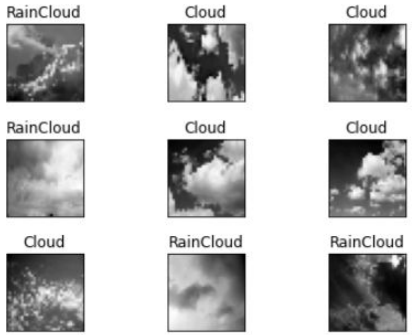
```
[18]  1  print("train dataset-rows:",train_data.shape[0]," columns:", train_data.shape[1], " rows:", train_data.shape[2])
      2  print("test dataset-rows:",test_data.shape[0]," columns:", test_data.shape[1], " rows:", train_data.shape[2])
```

```
train dataset-rows: 1000  columns: 32  rows: 32
test dataset-rows: 200  columns: 32  rows: 32
```

DATA PREPARATION

แสดงภาพตัวอย่างจาก Train Dataset

```
[19]  1  for i in range(9):
      2      ax = plt.subplot(3, 3, 1+i)
      3      ax.set_xticks([])
      4      ax.set_yticks([])
      5      ax.set_title('%s'%(CATEGORIES[int(y[i])]))
      6      plt.imshow(train_data[i], cmap=plt.get_cmap('gray'))
      7  plt.tight_layout()
      8  plt.savefig('cloud.png', dpi=300)
```



DATA PREPARATION

เข้ารหัสผลเฉลยแบบ One-hot Encoding

```
[22]  1  print(y.shape, y_test.shape)
      2  print(y[:9])
```

```
(1000,) (200,)
[1 0 0 1 0 0 0 1 1]
```

```
[23]  1  y = to_categorical(y)
      2  y_test = to_categorical(y_test)
      3
      4  print(y.shape, y_test.shape)
      5  y[:9]
```

```
(1000, 2) (200, 2)
array([[0., 1.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.]], dtype=float32)
```

แบ่งข้อมูลสำหรับ Train และ Validate โดยการสุ่มในสัดส่วน 80:20

```
[24]  1  x_train, x_val, y_train, y_val = train_test_split(train_data, y, test_size = 0.2 , random_state = 99)
      2  x_train.shape, x_val.shape, y_train.shape, y_val.shape
```

```
((800, 32, 32, 1), (200, 32, 32, 1), (800, 2), (200, 2))
```

CLOUD IMAGES DATASET

01 BASELINE MODEL

CLOUD IMAGES DATASET

BASELINE MODEL

Define Model

```
[25]
# model = Sequential()
model = tf.keras.Sequential()

#1. CNN LAYER
model.add(tf.keras.layers.Conv2D(filters = 32, kernel_size = (3,3), padding = 'Same', input_shape=(img_rows, img_cols, 1)))
model.add(tf.keras.layers.Activation("relu"))

#2. CNN LAYER
model.add(tf.keras.layers.Conv2D(filters = 32, kernel_size = (3,3), padding = 'Same'))
model.add(tf.keras.layers.Activation("relu"))

model.add(tf.keras.layers.MaxPool2D(pool_size=(2, 2)))

#3. CNN LAYER
model.add(tf.keras.layers.Conv2D(filters = 64, kernel_size = (3,3), padding = 'Same'))
model.add(tf.keras.layers.Activation("relu"))

#4. CNN LAYER
model.add(tf.keras.layers.Conv2D(filters = 64, kernel_size = (3,3), padding = 'Same'))
model.add(tf.keras.layers.Activation("relu"))

model.add(tf.keras.layers.MaxPool2D(pool_size=(2, 2)))

#FULLY CONNECTED LAYER
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256))
model.add(tf.keras.layers.Activation("relu"))

#OUTPUT LAYER
model.add(tf.keras.layers.Dense(2, activation='sigmoid'))
```

BASELINE MODEL

Compile Model

```
[26]  1  optimizer = Adam()
      2  model.compile(optimizer = optimizer, loss = "binary_crossentropy", metrics=["accuracy"])
      3  model.summary()
```

Callbacks function

```
[27]  1  learning_rate_reduction = ReduceLROnPlateau(monitor='val_loss', patience = 2, verbose=1,factor=0.1, min_lr = 0.000001)
      2  callbacks_list = [learning_rate_reduction]
```

Train model

```
[28]  1  history = model.fit(x_train, y_train,
      2                       batch_size= BATCH_SIZE,
      3                       epochs= EPOCHS,
      4                       callbacks=callbacks_list,
      5                       verbose = 1,
      6                       validation_data=(x_val, y_val))
```

```
Epoch 1/100
25/25 [==============================] - 3s 28ms/step - loss: 0.5328 - accuracy: 0.7300 - val_loss: 0.4107 - val_accuracy: 0.8150
Epoch 2/100
25/25 [==============================] - 0s 18ms/step - loss: 0.2832 - accuracy: 0.8913 - val_loss: 0.3620 - val_accuracy: 0.8500
Epoch 3/100
25/25 [==============================] - 0s 17ms/step - loss: 0.2475 - accuracy: 0.9050 - val_loss: 0.3781 - val_accuracy: 0.8450
Epoch 4/100
25/25 [==============================] - 0s 17ms/step - loss: 0.2620 - accuracy: 0.8975 - val_loss: 0.3497 - val_accuracy: 0.8500
Epoch 5/100
25/25 [==============================] - 0s 17ms/step - loss: 0.2286 - accuracy: 0.9150 - val_loss: 0.4267 - val_accuracy: 0.8650
```

BASELINE MODEL

นิยามกราฟ

```
[29]
def create_trace(x,y,ylabel,color):
        trace = go.Scatter(x = x,y = y,name=ylabel,marker=dict(color=color),mode = "markers+lines",text=x)
        return trace

def plot_accuracy_and_loss(train_model):
    hist = train_model.history
    acc = hist['accuracy']
    val_acc = hist['val_accuracy']
    loss = hist['loss']
    val_loss = hist['val_loss']
    epochs = list(range(1,len(acc)+1))

    trace_ta = create_trace(epochs,acc,"Training accuracy", "Green")
    trace_va = create_trace(epochs,val_acc,"Validation accuracy", "Red")
    trace_tl = create_trace(epochs,loss,"Training loss", "Blue")
    trace_vl = create_trace(epochs,val_loss,"Validation loss", "Magenta")

    fig = subplots.make_subplots(rows=1,cols=2,
                                 subplot_titles=('Training and validation accuracy','Training and validation loss'))
    fig.append_trace(trace_ta,1,1)
    fig.append_trace(trace_va,1,1)
    fig.append_trace(trace_tl,1,2)
    fig.append_trace(trace_vl,1,2)
    fig['layout']['xaxis'].update(title = 'Epoch')
    fig['layout']['xaxis2'].update(title = 'Epoch')
    fig['layout']['yaxis'].update(title = 'Accuracy', range=[0,1])
    fig['layout']['yaxis2'].update(title = 'Loss', range=[0,1])

    plotly.offline.iplot(fig, filename='accuracy-loss')
```

BASELINE MODEL

Save History

```
[31]  1  filepath_history_model = 'history_model'
      2  with open(filepath_history_model, 'wb') as file :
      3    p.dump(history.history, file)
```

```
[32]  1  filepath = 'model.h5'
      2  model.save(filepath)
```

Load History

```
[33]  1  with open(filepath_history_model, 'rb') as file :
      2    his = p.load(file)
```

```
[34]  1  predict_model = load_model(filepath)
```

Evaluate Model

```
[35]  1  score = predict_model.evaluate(test_data, y_test, verbose = 0)
      2  print("Test Loss:",score[0])
      3  print("Test Accuracy:",score[1])
```

```
Test Loss: 0.2878543734550476
Test Accuracy: 0.8600000143051147
```

CLOUD
IMAGES
DATASET

02

IMAGE
AUGMENTATION

* CLASSIFICATION MODELS *

IMAGE AUGMENTATION

นิยาม Image Augmentation

```
[41]  1  datagen = ImageDataGenerator(
      2          rotation_range=0.05,    #Randomly rotate images in the range
      3          zoom_range = 0.2,       #Randomly zoom image
      4          width_shift_range=0.1,  #Randomly shift images horizontally
      5          height_shift_range=0.1, #Randomly shift images vertically
      6          shear_range=0.05        #Randomly shear images
      7          )
      8
      9  datagen.fit(x_train)
```

ลดมิติภาพ

```
[42]  1  x_batch = datagen.flow(x_train, y_train, batch_size=9).next()
      2  print(x_batch[0].shape)
      3  x_batch = x_batch[0].reshape((x_batch[0].shape[0], img_rows, img_cols))
      4  print(x_batch.shape)

     (9, 32, 32, 1)
     (9, 32, 32)
```

IMAGE AUGMENTATION

นิยาม Model

```
[43]
# model = Sequential()
model = tf.keras.Sequential()

#1. CNN LAYER
model.add(tf.keras.layers.Conv2D(filters = 32, kernel_size = (3,3), padding = 'Same', input_shape=(img_rows, img_cols, 1)))
model.add(tf.keras.layers.Activation("relu"))

#2. CNN LAYER
model.add(tf.keras.layers.Conv2D(filters = 32, kernel_size = (3,3), padding = 'Same'))
model.add(tf.keras.layers.Activation("relu"))

model.add(tf.keras.layers.MaxPool2D(pool_size=(2, 2)))

#3. CNN LAYER
model.add(tf.keras.layers.Conv2D(filters = 64, kernel_size = (3,3), padding = 'Same'))
model.add(tf.keras.layers.Activation("relu"))

#4. CNN LAYER
model.add(tf.keras.layers.Conv2D(filters = 64, kernel_size = (3,3), padding = 'Same'))
model.add(tf.keras.layers.Activation("relu"))

model.add(tf.keras.layers.MaxPool2D(pool_size=(2, 2)))

#FULLY CONNECTED LAYER
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256))
model.add(tf.keras.layers.Activation("relu"))

#OUTPUT LAYER
model.add(tf.keras.layers.Dense(2, activation='sigmoid'))
```

IMAGE AUGMENTATION

Compile Model

```
[44]  1  optimizer = Adam()
      2  model.compile(optimizer = optimizer, loss = "binary_crossentropy", metrics=["accuracy"])
      3  model.summary()
```

Train Model

```
[45]  1  history = model.fit(datagen.flow(x_train, y_train, batch_size=BATCH_SIZE),
      2                               shuffle=True,
      3                               epochs=EPOCHS,
      4                               callbacks=callbacks_list,
      5                               validation_data = (x_val, y_val),
      6                               verbose=1,
      7                               steps_per_epoch=x_train.shape[0] // BATCH_SIZE)
```

```
Epoch 1/100
25/25 [==============================] - 2s 34ms/step - loss: 0.6106 - accuracy: 0.6263 - val_loss: 0.4483 - val_accuracy: 0.7900
Epoch 2/100
25/25 [==============================] - 1s 25ms/step - loss: 0.4074 - accuracy: 0.8350 - val_loss: 0.3566 - val_accuracy: 0.8500
Epoch 3/100
25/25 [==============================] - 1s 28ms/step - loss: 0.2935 - accuracy: 0.8813 - val_loss: 0.3484 - val_accuracy: 0.8300
Epoch 4/100
25/25 [==============================] - 1s 26ms/step - loss: 0.2829 - accuracy: 0.8875 - val_loss: 0.3507 - val_accuracy: 0.8300
Epoch 5/100
25/25 [==============================] - 1s 25ms/step - loss: 0.2798 - accuracy: 0.8913 - val_loss: 0.3715 - val_accuracy: 0.8550

Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
Epoch 6/100
25/25 [==============================] - 1s 27ms/step - loss: 0.2390 - accuracy: 0.9087 - val_loss: 0.3754 - val_accuracy: 0.8550
Epoch 7/100
25/25 [==============================] - 1s 27ms/step - loss: 0.2536 - accuracy: 0.8963 - val_loss: 0.3820 - val_accuracy: 0.8600
```

IMAGE AUGMENTATION

Save History

```
[47]  1  filepath_history_model = 'history_model'
      2  with open(filepath_history_model, 'wb') as file :
      3    p.dump(history.history, file)
```

```
[48]  1  filepath = 'model.h5'
      2  model.save(filepath)
```

Load History

```
[49]  1  with open(filepath_history_model, 'rb') as file :
      2    his = p.load(file)
```

```
[50]  1  predict_model = load_model(filepath)
```

Evaluate Model

```
[51]  1  score = predict_model.evaluate(test_data, y_test, verbose = 0)
      2  print("Test Loss:",score[0])
      3  print("Test Accuracy:",score[1])
```

```
Test Loss: 0.29108744859695435
Test Accuracy: 0.8700000047683716
```

03

CLOUD IMAGES DATASET

BATCH NORMALIZATION

* CLASSIFICATION MODELS *

BATCH NORMALIZATION

นิยาม Image Augmentation

```
[57]  1  datagen = ImageDataGenerator(
      2                   rotation_range=0.05,    #Randomly rotate images in the range
      3                   zoom_range = 0.2,       #Randomly zoom image
      4                   width_shift_range=0.1,  #Randomly shift images horizontally
      5                   height_shift_range=0.1, #Randomly shift images vertically
      6                   shear_range=0.05        #Randomly shear images
      7                   )
      8
      9  datagen.fit(x_train)
```

BATCH NORMALIZATION

นิยาม Model

```
[58]  1  # model = Sequential()
      2  model = tf.keras.Sequential()
      3
      4  #1. CNN LAYER
      5  model.add(tf.keras.layers.Conv2D(filters = 32, kernel_size = (3,3), padding = 'Same', input_shape=(img_rows, img_cols, 1)))
      6  model.add(tf.keras.layers.BatchNormalization())
      7  model.add(tf.keras.layers.Activation("relu"))
      8
      9  #2. CNN LAYER
     10  model.add(tf.keras.layers.Conv2D(filters = 32, kernel_size = (3,3), padding = 'Same'))
     11  model.add(tf.keras.layers.BatchNormalization())
     12  model.add(tf.keras.layers.Activation("relu"))
     13
     14  model.add(tf.keras.layers.MaxPool2D(pool_size=(2, 2)))
     15
     16  #3. CNN LAYER
     17  model.add(tf.keras.layers.Conv2D(filters = 64, kernel_size = (3,3), padding = 'Same'))
     18  model.add(tf.keras.layers.BatchNormalization())
     19  model.add(tf.keras.layers.Activation("relu"))
     20
     21  #4. CNN LAYER
     22  model.add(tf.keras.layers.Conv2D(filters = 64, kernel_size = (3,3), padding = 'Same'))
     23  model.add(tf.keras.layers.BatchNormalization())
     24  model.add(tf.keras.layers.Activation("relu"))
     25
     26  model.add(tf.keras.layers.MaxPool2D(pool_size=(2, 2)))
     27
     28  #FULLY CONNECTED LAYER
     29  model.add(tf.keras.layers.Flatten())
     30  model.add(tf.keras.layers.Dense(256))
     31  model.add(tf.keras.layers.BatchNormalization())
     32  model.add(tf.keras.layers.Activation("relu"))
     33
     34  #OUTPUT LAYER
     35  model.add(tf.keras.layers.Dense(2, activation='sigmoid'))
```

BATCH NORMALIZATION

Compile Model

```
[59]  1  optimizer = Adam()
      2  model.compile(optimizer = optimizer, loss = "binary_crossentropy", metrics=["accuracy"])
      3
      4  model.summary()
```

Train Model

```
[60]  1  history = model.fit(datagen.flow(x_train, y_train, batch_size=BATCH_SIZE),
      2                               shuffle=True,
      3                               epochs= EPOCHS,
      4                               callbacks=callbacks_list,
      5                               validation_data = (x_val, y_val),
      6                               verbose=1,
      7                               steps_per_epoch=x_train.shape[0] // BATCH_SIZE)
```

```
Epoch 1/100
25/25 [==============================] - 2s 40ms/step - loss: 0.2869 - accuracy: 0.8888 - val_loss: 0.8264 - val_accuracy: 0.5250
Epoch 2/100
25/25 [==============================] - 1s 28ms/step - loss: 0.1921 - accuracy: 0.9212 - val_loss: 1.1771 - val_accuracy: 0.5250
Epoch 3/100
25/25 [==============================] - 1s 29ms/step - loss: 0.1938 - accuracy: 0.9250 - val_loss: 1.3864 - val_accuracy: 0.5250

Epoch 00003: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
Epoch 4/100
25/25 [==============================] - 1s 29ms/step - loss: 0.1536 - accuracy: 0.9425 - val_loss: 1.4747 - val_accuracy: 0.5250
Epoch 5/100
25/25 [==============================] - 1s 29ms/step - loss: 0.1457 - accuracy: 0.9425 - val_loss: 1.6145 - val_accuracy: 0.5250

Epoch 00005: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
Epoch 6/100
25/25 [==============================] - 1s 28ms/step - loss: 0.1438 - accuracy: 0.9500 - val_loss: 1.7038 - val_accuracy: 0.5250
Epoch 7/100
25/25 [==============================] - 1s 28ms/step - loss: 0.1421 - accuracy: 0.9488 - val_loss: 1.7803 - val_accuracy: 0.5250
```

BATCH NORMALIZATION

Save History

```
[62]  1  filepath_history_model = 'history_model'
      2  with open(filepath_history_model, 'wb') as file :
      3    p.dump(history.history, file)
```

```
[63]  1  filepath = 'model.h5'
      2  model.save(filepath)
```

Load History

```
[64]  1  with open(filepath_history_model, 'rb') as file :
      2    his = p.load(file)
```

```
[65]  1  predict_model = load_model(filepath)
```

Evaluate Model

```
[66]  1  score = predict_model.evaluate(test_data, y_test, verbose = 0)
      2  print("Test Loss:",score[0])
      3  print("Test Accuracy:",score[1])
```

```
Test Loss: 0.194719135761261
Test Accuracy: 0.9150000214576721
```

* CLASSIFICATION MODELS *

CLOUD IMAGES DATASET

DROPOUT 04

CLOUD IMAGES DATASET

* CLASSIFICATION MODELS *

DROPOUT MODEL

นิยาม Image Augmentation

```
[72]  1  datagen = ImageDataGenerator(
      2                  rotation_range=0.05,    #Randomly rotate images in the range
      3                  zoom_range=0.2,         #Randomly zoom image
      4                  width_shift_range=0.1,  #Randomly shift images horizontally
      5                  height_shift_range=0.1, #Randomly shift images vertically
      6                  shear_range=0.05        #Randomly shear images
      7                  )
      8
      9  datagen.fit(x_train)
```

DROPOUT MODEL

นิยาม Model

```
[73]  1  # model = Sequential()
      2  model = tf.keras.Sequential()
      3
      4  #1. CNN LAYER
      5  model.add(tf.keras.layers.Conv2D(filters = 32, kernel_size = (3,3), padding = 'Same', input_shape=(img_rows, img_cols, 1)))
      6  model.add(tf.keras.layers.BatchNormalization())
      7  model.add(tf.keras.layers.Activation("relu"))
      8  model.add(tf.keras.layers.Dropout(0.1))
      9
     10  #2. CNN LAYER
     11  model.add(tf.keras.layers.Conv2D(filters = 32, kernel_size = (3,3), padding = 'Same'))
     12  model.add(tf.keras.layers.BatchNormalization())
     13  model.add(tf.keras.layers.Activation("relu"))
     14  model.add(tf.keras.layers.MaxPool2D(pool_size=(2, 2)))
     15  model.add(tf.keras.layers.Dropout(0.1))
     16
     17  #3. CNN LAYER
     18  model.add(tf.keras.layers.Conv2D(filters = 64, kernel_size = (3,3), padding = 'Same'))
     19  model.add(tf.keras.layers.BatchNormalization())
     20  model.add(tf.keras.layers.Activation("relu"))
     21  model.add(tf.keras.layers.Dropout(0.1))
     22
     23  #4. CNN LAYER
     24  model.add(tf.keras.layers.Conv2D(filters = 64, kernel_size = (3,3), padding = 'Same'))
     25  model.add(tf.keras.layers.BatchNormalization())
     26  model.add(tf.keras.layers.Activation("relu"))
     27
     28  model.add(tf.keras.layers.MaxPool2D(pool_size=(2, 2)))
     29  model.add(tf.keras.layers.Dropout(0.1))
     30
     31  #FULLY CONNECTED LAYER
     32  model.add(tf.keras.layers.Flatten())
     33  model.add(tf.keras.layers.Dense(256))
     34  model.add(tf.keras.layers.BatchNormalization())
     35  model.add(tf.keras.layers.Activation("relu"))
     36  model.add(tf.keras.layers.Dropout(0.1))
     37
     38  #OUTPUT LAYER
     39  model.add(tf.keras.layers.Dense(2, activation='sigmoid'))
```

DROPOUT MODEL

Compile Model

```
[74]  1  optimizer = Adam()
      2  model.compile(optimizer = optimizer, loss = "binary_crossentropy", metrics=["accuracy"])
      3  model.summary()
```

Train Model

```
[75]  1  history = model.fit(datagen.flow(x_train, y_train, batch_size=BATCH_SIZE),
      2                               shuffle=True,
      3                               epochs=EPOCHS,
      4                               callbacks=callbacks_list,
      5                               validation_data = (x_val, y_val),
      6                               verbose = 1,
      7                               steps_per_epoch=x_train.shape[0] // BATCH_SIZE)
```

```
Epoch 1/100
25/25 [==============================] - 2s 39ms/step - loss: 0.3659 - accuracy: 0.8500 - val_loss: 0.7357 - val_accuracy: 0.5250
Epoch 2/100
25/25 [==============================] - 1s 31ms/step - loss: 0.2289 - accuracy: 0.9038 - val_loss: 1.2812 - val_accuracy: 0.5250
Epoch 3/100
25/25 [==============================] - 1s 31ms/step - loss: 0.2078 - accuracy: 0.9250 - val_loss: 1.6813 - val_accuracy: 0.5250

Epoch 00003: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
Epoch 4/100
25/25 [==============================] - 1s 29ms/step - loss: 0.1698 - accuracy: 0.9413 - val_loss: 1.8992 - val_accuracy: 0.5250
Epoch 5/100
25/25 [==============================] - 1s 29ms/step - loss: 0.1573 - accuracy: 0.9400 - val_loss: 2.1175 - val_accuracy: 0.5250

Epoch 00005: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
Epoch 6/100
25/25 [==============================] - 1s 28ms/step - loss: 0.1561 - accuracy: 0.9375 - val_loss: 2.2541 - val_accuracy: 0.5250
Epoch 7/100
25/25 [==============================] - 1s 29ms/step - loss: 0.1567 - accuracy: 0.9400 - val_loss: 2.3562 - val_accuracy: 0.5250
```

DROPOUT MODEL

Save History

```
[77]  1  filepath_history_model = 'history_model'
      2  with open(filepath_history_model, 'wb') as file :
      3    p.dump(history.history, file)
```

```
[78]  1  filepath = 'model.h5'
      2  model.save(filepath)
```

Load History

```
[79]  1  with open(filepath_history_model, 'rb') as file :
      2    his = p.load(file)
```

```
[80]  1  predict_model = load_model(filepath)
```

Evaluate Model

```
[81]  1  score = predict_model.evaluate(test_data, y_test, verbose = 0)
      2  print("Test Loss:",score[0])
      3  print("Test Accuracy:",score[1])
```

```
Test Loss: 0.17420116066932678
Test Accuracy: 0.9200000166893005
```

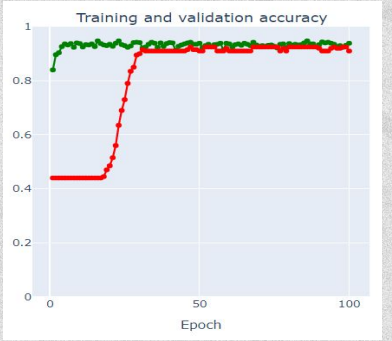
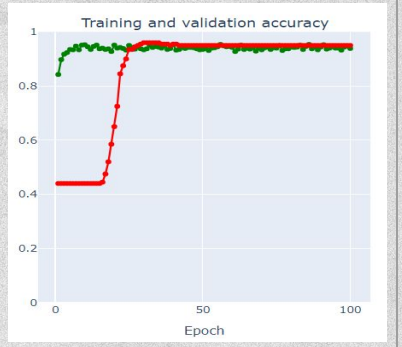
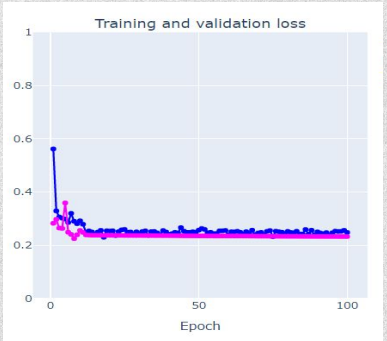
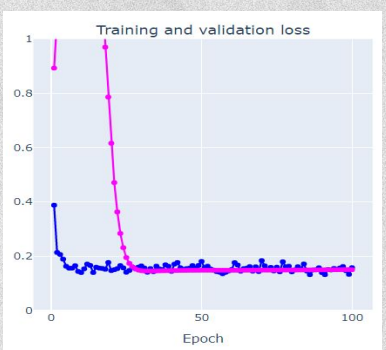
CLOUD IMAGES DATASET

COMPARE MODELS

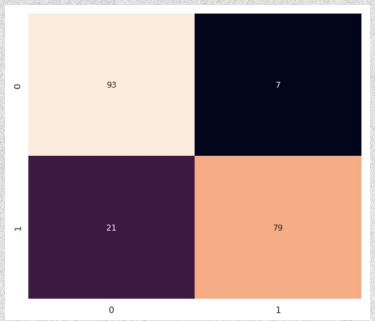
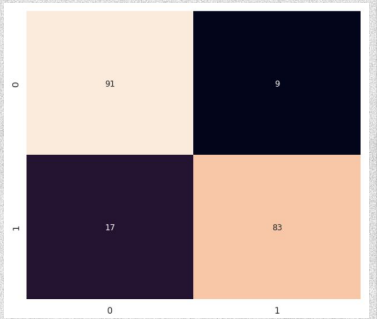
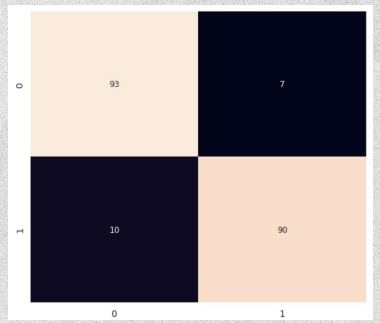
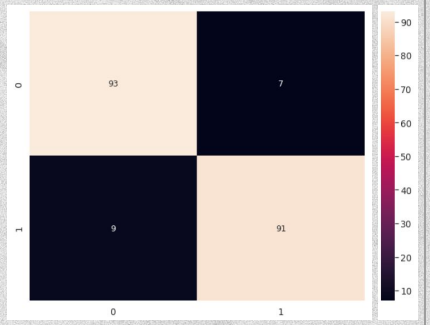
* CLASSIFICATION *

* CLASSIFICATION *

COMPARE MODELS

Models	Baseline Model	Image Augmentation	Batch Normalization	Dropout
Plot accuracy				
Plot loss				

COMPARE MODELS

Models	Baseline Model	Image Augmentation	Batch Normalization	Dropout
Confusion matrix				
Classification Report	precision recall f1-score Cloud 0.8158 0.9300 0.8692 RainCloud 0.9186 0.7900 0.8495 accuracy 0.8600 macro avg 0.8672 0.8600 0.8593 weighted avg 0.8672 0.8600 0.8593	precision recall f1-score Cloud 0.8426 0.9100 0.8750 RainCloud 0.9022 0.8300 0.8646 accuracy 0.8700 macro avg 0.8724 0.8700 0.8698 weighted avg 0.8724 0.8700 0.8698	precision recall f1-score Cloud 0.9029 0.9300 0.9163 RainCloud 0.9278 0.9000 0.9137 accuracy 0.9150 macro avg 0.9154 0.9150 0.9150 weighted avg 0.9154 0.9150 0.9150	precision recall f1-score Cloud 0.9118 0.9300 0.9208 RainCloud 0.9286 0.9100 0.9192 accuracy 0.9200 macro avg 0.9202 0.9200 0.9200 weighted avg 0.9202 0.9200 0.9200

THANKS!

CLOUD IMAGES DATASET

PRESENTED BY

Group : Today is sunday!

620710405 นางสาวณัฐธิดา ลาภธนชัย
620710745 นางสาวอาทิตยา ชมทอง

* CLASSIFICATION *

* CLASSIFICATION *