





LAPORAN TUGAS

Penerapan Kecerdasan Buatan

Pada Sistem Peringatan Dini Gunung Meletus



Matakuliah	TI0263 – Kecerdasan Buatan (Grup C) - Genap 2021/2022
Dosen Pengampu	Matahari Bhakti Nendya, S.Kom., M.T
Nama Kelompok	<i>Kelompok 4</i>
Anggota Kelompok	<div> <div>1. Dewangga Yuka Pratama (71200581)</div> <div>2. Abraham David Hartanto (71200632)</div> <div>3. Andreas Nugroho (71200646)</div> <div>4. Renaldo Surya Saputra (71200670)</div> </div> <div>     </div>
Deklarasi	Dengan ini kami menyatakan bahwa tugas ini merupakan hasil karya kelompok kami, tidak ada manipulasi data serta bukan merupakan plagiasi dari karya orang lain.



UNIVERSITAS KRISTEN DUTA WACANA
Fakultas Teknologi Informasi
Program Studi Informatika



BAB I: PROTOTYPE PROGRAM

Program kami dirancang untuk melakukan mendapatkan path atau gambaran dari aktivitas gunung berapi. Karena menggunakan bentuk data tree atau graph maka kami harus membuat program Python berdasarkan bentuk data tersebut. Program tersebut nantinya akan di import kedalam program utama.

```
1 class Graph:
2     def __init__(self):
3         self.data = {}
4
5     def addVertex(self, key):
6         if key not in self._data:
7             self._data[key] = []
8
9     def vertex(self):
10        for key, value in self._data.items():
11            print(key, end=' ')
12            print()
13
14    def addEdge(self, x, y):
15        if x in self._data and y in self._data:
16            self._data[x].append(y)
17            self._data[y].append(x)
18
19    def edge(self):
20        edges = []
21        for key, value in self._data.items():
22            for key2 in self._data[key]:
23                if key+key2 not in edges and key2+key not in edges:
24                    edges.append(key+key2)
25
26        for item in edges:
27            print(item, end=' ')
28            print()
29
30    def findPath(self, x, y):
31        visited = []
32        self.dfs(x, y, visited)
33
34    def dfs(self, node, y, visited):
35        visited.append(node)
36        if node == y:
37            print(visited)
38        else:
39            for item in self._data[node]:
40                if item not in visited:
41                    self.dfs(item, y, visited)
```

Gambar 1. File Graph

```
class getNode:
    def __init__(self, data):
        self.data = data
        self.left = self.right = self.mid = None

    def hasPath(root, arr, x):
        if (not root):
            return False

        arr.append(root.data)

        if (root.data == x):
            return True

        if (hasPath(root.left, arr, x) or hasPath(root.right, arr, x) or hasPath(root.mid, arr, x)):
            return True

        arr.pop(-1)
        return False

    def printPath(root, x):
        arr = []

        if (hasPath(root, arr, x)):
            for i in range(len(arr) - 1):
                print(arr[i], end = "<")
            print(arr[len(arr) - 1])
        else:
            print("No Path")
```

Gambar 2. File Untuk Pencarian Tr

Sehingga file program akan terdiri dari beberapa file berupa folder image untuk melakukan test pada gambar yang akan dicek statusnya, file python binarysearch.py, graph.py, dan volcano_attack.py sebagai main program.



Gambar 3. Gambaran Seluruh File

```
volcano_attack.py X
volcano_attack.py > ...
1 from PIL import Image, ImageDraw, ImageFont, ImageTk
2 import tkinter as tk
3 import easygui
4 from graph import *
5 from binarySearch import *
6
```

Gambar 4. Modul Pelengkap Untuk Program Utama

Untuk proses pengolahan gambar maka program dirancang untuk mendapatkan gambar pada lokal computer.

```
# Code untuk melakukan pengambilan gambar sesuai Pilihan
input_file_png = easygui.fileopenbox(msg="EDIT GAMBAR",filetypes=["*.png"])
input_file_jpg = input_file_png.rpartition('.')[0] + ".jpg"
print(input_file_jpg)
```

Gambar 5. Pengambilan Gambar

Setelah memilih gambar maka program akan mengarahkan user untuk melakukan crop gambar atau dalam kasus kami adalah pilih area untuk diidentifikasi. Bagian ini dikerjakan oleh dua

fungsi, yaitu fungsi `mouse_event` dengan parameter “event”, fungsi `update_sel_rect` dengan parameter “event” dan fungsi `imageCut`.

Lalu program akan mengembalikan gambar dengan tambahan informasi indentifikasi cahaya sesuai besarnya crop yang dilakukan dan melakukan identifikasi terhadap gambar yang telah dicrop tersebut. Identifikasi inilah yang kami gunakan untuk mengetahui aksi apa yang harus dilakukan.

```

bukaGambar = Image.open('jpg_crop.jpg')
draw = ImageDraw.Draw(bukaGambar)
lebar = bukaGambar.size[0]
tinggi = bukaGambar.size[1]
pix = bukaGambar.load()

bukaGambar = Image.open('png_crop.png')
postPix = list(bukaGambar.getdata())

canvaslebar = lebar + 20
canvastinggi = tinggi + 20
gambarBaru = Image.new('RGB', (canvaslebar, canvastinggi), 'white')
data = list(bukaGambar.getdata())
dx = 0
dy = 0
counter = 0
for y in range(tinggi):
    for x in range(lebar):
        draw.rectangle((dx, dy, dx + 20, dy + 20), fill=data[counter])
        font = ImageFont.truetype("arial.ttf", 10)
        fillage = str(data[counter])
        if postPix[counter][0] - postPix[counter][2] > 10:
            draw.text((dx + 1, dy + 5), fillage[1:4], 'green', font=font)
        else:
            draw.text((dx + 1, dy + 5), fillage[1:4], 'red', font=font)
        draw = ImageDraw.Draw(gambarBaru)
        dx = dx + 20
        counter = counter + 1
    dy = dy + 20
    print("Identifikasi = ", dy)
    dx = 0

```

Gambar 6. Proses Penghitungan Identifikasi

```

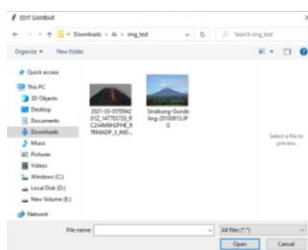
94 goal=""
95 if(dy < 500):
96     print("status gunung normal")
97     goal = "Kegiatan Normal"
98 elif(dy > 500 and dy < 1000):
99     print("status gunung siaga")
100     goal = "Masyarakat Tidak Perlu Mengungsi"
101 elif(dy > 1000 and dy < 1500):
102     print("status gunung waspada")
103     goal = "Masyarakat Mulai Mengungsi"
104 elif(dy > 1500 and dy < 2000):
105     print("status gunung awas")
106     goal = "Masyarakat Sudah Mengungsi"
107 elif(dy > 2000):
108     print("gunung meletus gan, silahkan mengungsi!")
109     goal = "Erupsi"
110 print(goal)
111

```

Gambar 7. Perhitungan Kemungkinan Berdasarkan Hasil Identifikasi

Berdasarkan program tersebut maka goal atau target sudah ditentukan. Misal hasil identifikasi melebihi angka 2000 maka pasti terjadi erupsi. Maka berikutnya algoritma dfs akan mendapatkan peristiwa yang telah terjadi berdasarkan *goal* tersebut.

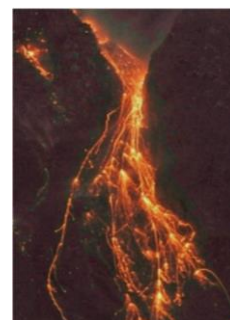
Berikut Contohnya:



Gambar 8. Pilih Gambar



Gambar 9. Lakukan Crop Pada Gambar



Gambar 10. Hasil Crop Keluar

```

Erupsi
DFS
['Start', 'Gunung Stabil', 'Status Normal', 'Kegiatan Normal', 'Masyarakat Tidak Perlu Mengungsi', 'Muncul Aktivitas', 'Aktivitas Seismik',
'Status Waspada', 'Masyarakat Perlu Waspada', 'Aktivitas Kawah', 'Status Siaga', 'Kegiatan Dikurangi', 'Masyarakat Mulai Mengungsi', 'Munc
ul Uap dan Abu', 'Status Awas', 'Kegiatan Diberhentikan', 'Masyarakat Sudah Mengungsi', 'Erupsi']

=====
Path Yang Ditemukan :
Start->Muncul Aktivitas->Muncul Uap dan Abu->Erupsi
PS C:\Users\I E N O V O\Downloads\Ai>

```

Gambar 11. Hasil Akhir Program Berisi Tahapan Sebelum Goal

Selain itu, kami juga menggunakan metode *machine learning* yang mempunyai fungsi utama untuk mendeteksi gambar yang diambil dari kondisi gunung dan dibandingkan dengan berbagai *sample* yang dibagi menjadi beberapa *class* untuk setiap kondisi gunung. Berikut adalah langkah-langkah utama demo prototype menggunakan metode ini :

1. Image Scrapping

```
pip install -U duckduckgo_search

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting duckduckgo_search
  Downloading duckduckgo_search-1.8-py3-none-any.whl (16 kB)
Collecting requests>=2.27.1
  Downloading requests-2.28.0-py3-none-any.whl (62 kB)
  |#####| 62 kB 1.3 MB/s
Collecting lxml>=4.7.1
  Downloading lxml-4.9.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_24_x86_64.whl (6.4 MB)
  |#####| 6.4 MB 37.0 MB/s
Collecting click>=8.1.3
  Downloading click-8.1.3-py3-none-any.whl (96 kB)
  |#####| 96 kB 6.4 MB/s
Collecting brotli>=1.0.9
  Downloading Brotli-1.0.9-cp37-cp37m-manylinux1_x86_64.whl (357 kB)
  |#####| 357 kB 43.6 MB/s
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from click>=8.1.3->duckduckgo_search) (4.11.4)
Requirement already satisfied: charset-normalizer<=2.0.0 in /usr/local/lib/python3.7/dist-packages (from requests>=2.27.1->duckduckgo_search) (2.0.12)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests>=2.27.1->duckduckgo_search) (2022.5.18.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests>=2.27.1->duckduckgo_search) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests>=2.27.1->duckduckgo_search) (1.24.3)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->click>=8.1.3->duckduckgo_search) (3.8.0)
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->click>=8.1.3->duckduckgo_search) (4.2.0)
Installing collected packages: requests, lxml, click, brotli, duckduckgo-search
  Attempting uninstall: requests
    Found existing installation: requests 2.23.0
    Uninstalling requests-2.23.0:
      Successfully uninstalled requests-2.23.0
  Attempting uninstall: lxml
    Found existing installation: lxml 4.2.6
    Uninstalling lxml-4.2.6:
      Successfully uninstalled lxml-4.2.6
  Attempting uninstall: click
    Found existing installation: click 7.1.2
    Uninstalling click-7.1.2:
      Successfully uninstalled click-7.1.2
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
google-colab 1.0.0 requires requests==2.23.0, but you have requests 2.28.0 which is incompatible.
flask 1.1.4 requires click<8.0,>=5.1, but you have click 8.1.3 which is incompatible.
datascience 0.10.6 requires folium==0.2.1, but you have folium 0.8.3 which is incompatible.
Successfully installed brotli-1.0.9 click-8.1.3 duckduckgo-search-1.8 lxml-4.9.0 requests-2.28.0

[ ] !pip install -q jmd_imagescraper
```

Gambar 12. Penginstalan Library untuk mengakses sample dari DuckDuckGo

```
from pathlib import Path
root = Path().cwd()/"gambar"

from jmd_imagescraper.core import *

duckduckgo_search(root, "Status Normal", "mountains", max_results=50)
duckduckgo_search(root, "Gunung Berasap", "mountain eruption", max_results=50)
duckduckgo_search(root, "Gunung Meletus", "volcano", max_results=50)

Duckduckgo search: mountains
Downloading results into /content/gambar/Status Normal
##### 100.00% [50/50 00:04<00:00 Images downloaded]
Duckduckgo search: mountain eruption
Downloading results into /content/gambar/Gunung Berasap
##### 100.00% [50/50 00:04<00:00 Images downloaded]
Duckduckgo search: volcano
Downloading results into /content/gambar/Gunung Meletus
##### 100.00% [50/50 00:02<00:00 Images downloaded]
[PosixPath('/content/gambar/Gunung Meletus/001_c37802a3.jpg'),
 PosixPath('/content/gambar/Gunung Meletus/002_4fa99f90.jpg'),
 PosixPath('/content/gambar/Gunung Meletus/003_e639b8b1.jpg'),
 PosixPath('/content/gambar/Gunung Meletus/004_1a1a1a1a.jpg'),
 PosixPath('/content/gambar/Gunung Meletus/005_2b2b2b2b.jpg'),
 PosixPath('/content/gambar/Gunung Meletus/006_3c3c3c3c.jpg'),
 PosixPath('/content/gambar/Gunung Meletus/007_4d4d4d4d.jpg'),
 PosixPath('/content/gambar/Gunung Meletus/008_5e5e5e5e.jpg'),
 PosixPath('/content/gambar/Gunung Meletus/009_6f6f6f6f.jpg'),
 PosixPath('/content/gambar/Gunung Meletus/010_7f7f7f7f.jpg')]
```

Gambar 13. Pengambilan sample gambar

Pengambilan sample gambar untuk setiap kondisi diambil melalui salah satu search engine yaitu DuckDuckGo dengan pembagian setiap kondisi menggunakan keyword. Keyword yang digunakan untuk setiap kondisi adalah mountains, mountain eruption, dan volcano. Pembagian setiap sample untuk setiap kondisi diatur dengan jumlah maksimal 50 gambar yang kemudian dibagi kedalam sub folder masing-masing.

▼ Kemudian memasukan hasil scrapping

```
[ ] from google.colab import drive
import shutil

drive.mount("/content/drive")
folder = Path("/content/drive/My Drive/AI")
folder.mkdir(parents=True, exist_ok=True)

shutil.copyfile(ZIP_NAME, str(folder/ZIP_NAME))

Mounted at /content/drive
'/content/drive/My Drive/AI/gunung.zip'
```

Gambar 14. Memasukkan hasil scrapping ke dalam sebuah folder di Gdrive

2. Membuat dan Train Model

```
num_classes = 5

model = Sequential([
    layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
    layers.Conv2D(16, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(num_classes)
])
```

▼ Mengcompile Model

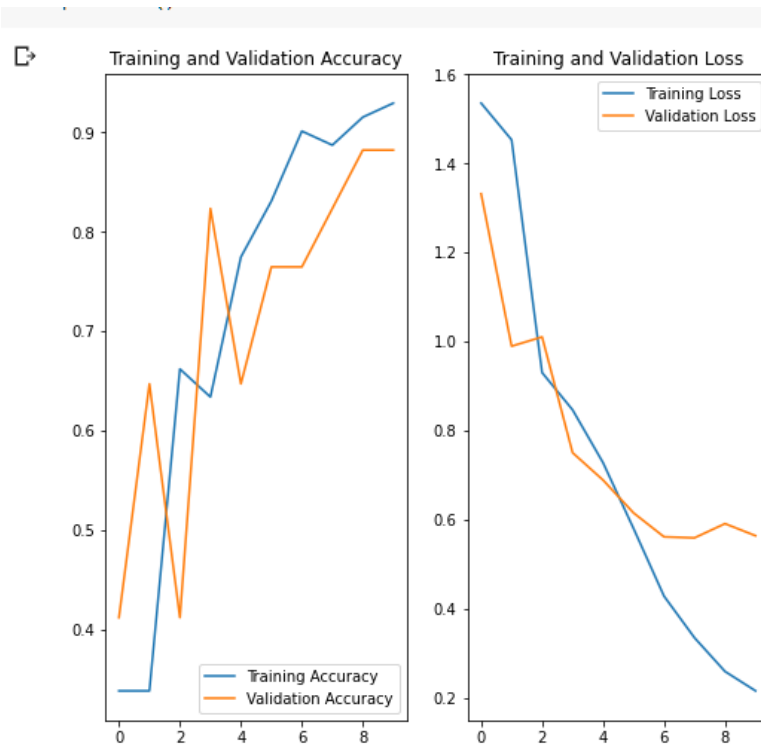
```
[ ] model.compile(optimizer='adam',
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])
```

```
[ ] model.summary()
```

Model: "sequential"

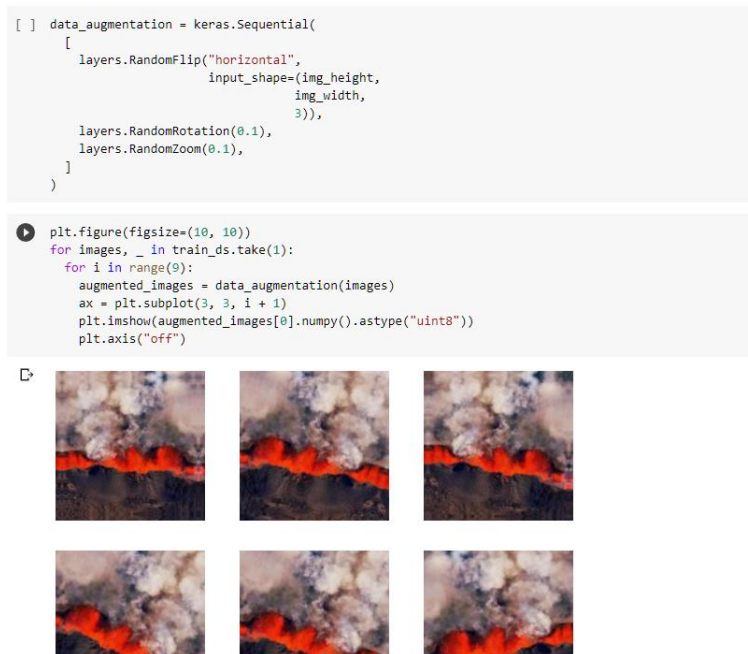
Layer (type)	Output Shape	Param #
=====		
rescaling_1 (Rescaling)	(None, 180, 180, 3)	0
conv2d (Conv2D)	(None, 180, 180, 16)	448
max_pooling2d (MaxPooling2D)	(None, 90, 90, 16)	0
)		

Gambar 15. Pembuatan model dan proses compile



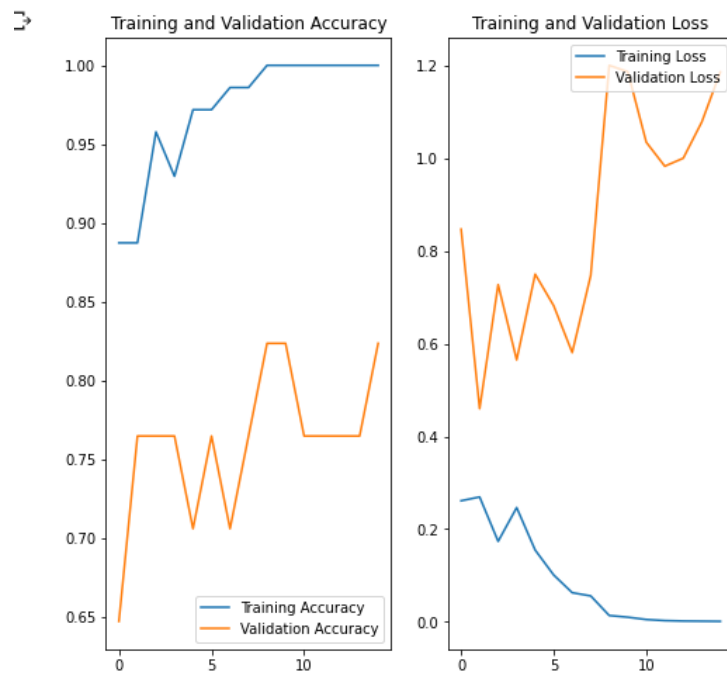
Gambar 16. Hasil Visualiasi Training Model

Menggunakan teknik augmentasi untuk mengurangi overfitting



Gambar 17. Augmentasi untuk menambah akurasi

Teknik augmentasi tersebut akan meningkatkan akurasi karena melihat sample dari berbagai sudut pandang, sehingga dapat mengurangi overfitting untuk setiap gambar yang digunakan. Setelah penggunaan teknik augmentasi tersebut, akan diulang proses modelling sebelumnya.



Gambar 18. Hasil Visualiasi setelah proses augmentasi

3. Percobaan



Gambar 19 . Gambar yang dijadikan percobaan

```
import queue
flow_path = tf.keras.utils.get_file(fname=None, origin='https://upload.wikimedia.org/wikipedia/commons/9/93/Lava_fountain_USGS_page_30424305-068_large.JPG')

img = tf.keras.utils.load_img(
    flow_path, target_size=(img_height, img_width)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)

Downloading data from https://upload.wikimedia.org/wikipedia/commons/9/93/Lava_fountain_USGS_page_30424305-068_large.JPG
73728/72321 [=====] - 0s 0us/step
81920/72321 [=====] - 0s 0us/step
This image most likely belongs to Gunung Meletus with a 99.76 percent confidence.
```

Gambar 20. Hasil

Gambar yang dijadikan percobaan dimasukan ke dalam sistem sehingga dikeluarkan output yang mendeteksi bahwa gambar tersebut masuk ke dalam kondisi Gunung Meletus dengan ketepatan 99.76% menurut sistem .

BAB II: CONTOH KASUS YANG MENJELASKAN METODE REPRESENTASI

Metode representasi pengetahuan yang kami gunakan adalah *production system*. Pada metode ini kami mengambil kasus gunung erupsi. Hal ini diimplementasikan dalam bentuk scanner yang digunakan untuk menangkap status gunung dengan melihat ciri-ciri gunung tersebut. Misalnya jika **(IF)** gunung tersebut tidak mengeluarkan asap dan kondisi disekitar kawah tidak berubah, maka **(THEN)** gunung tersebut berstatus normal atau jika **(IF)** gunung tersebut mengeluarkan lava dan muncul uap beserta awan kelabu, maka **(THEN)** gunung tersebut sedang terjadi erupsi. Hal ini dapat dilihat dari hasil visual data gunung tersebut dan mengolahnya kedalam bentuk perbandingan pixel untuk dihitung dengan tujuan mengidentifikasi kondisi gunung tersebut.

BAB III: CONTOH KASUS YANG MENJELASKAN METODE ALGORITMA

Pada kasus ini metode yang kami gunakan berupa *uninformed search*. Algoritma yang kami pilih adalah Depth First Search (DFS) atau pencarian mendalam. Pada kasus gunung erupsi, algoritma DFS ini diimplementasikan sebagai langkah atau pendeteksi status dari gunung yang diamati. Hal ini dimaksudkan untuk mengantisipasi kejadian yang akan terjadi jika mampu mengidentifikasi status gunung tersebut. Misalnya setelah menangkap/mendeteksi kondisi gunung yang bersangkutan berupa “status awas” maka pengimplementasian algoritma DFS ini berupa langkah yang harus dilakukan pada saat “status awas” diberlakukan, sehingga jika gunung terdeteksi pada “status awas” maka masyarakat diharuskan berhenti aktivitas dari radius tertentu dan segera mengungsi.