

# BAB 1

---

## CHAPTER 1

---



## BAB 2

---

## CHAPTER 2

---



## BAB 3

---

## CHAPTER 3

---



## BAB 4

---

## CHAPTER 4

---





## BAB 5

---

## CHAPTER 4

---



## BAB 6

---

## CHAPTER 5

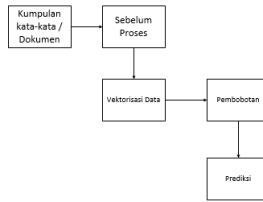
---

### 6.1 1174069 - Fanny Shafira Damayanti

#### 6.1.1 Teori

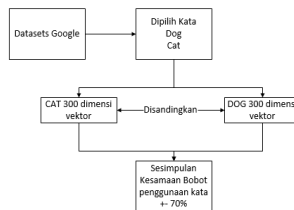
1. Jelaskan kenapa kata-kata harus di lakukan vektorisasi. Dilengkapi dengan ilustrasi atau gambar.

Kata kata harus dilakukan vektorisasi dikarenakan atau bertujuan utk mengukur nilai kemunculan suatu kata yang serupa dari sebuah kalimat sehingga kata-kata tersebut dapat di prediksi kemunculanya. atau juga di buatnya vektorisasi data digunakan untuk memprediksi bobot dari suatu kata misalkan ayam dan kucing sama-sama hewan maka akan dibuat prediksi apakah kata tersebut akan muncul pada kalimat yang kira-kira memiliki bobot yang sama.



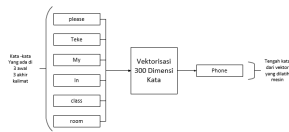
Gambar 6.1 Teori 1

2. Jelaskan mengapa dimensi dari vektor dataset google bisa sampai 300. Dilengkapidengan ilustrasi atau gambar.
- Dimensi dataset dari google bisa mencapai 300 karena dimensi dari vektor tersebut digunakan untuk membandingkan bobot dari setiap kata, misalkan terdapat kata dog dan cat pada dataset google tersebut setiap kata tersebut di buat dimensi vektor 300 untuk kata dog dan 300 dimensi vektor juga untuk kata cat kemudian kata tersebt di bandingkan bobot kesamaan katanya maka akan muncul akurasi sekitar 70 persen kesamaan bobot dikarenakan kata dog dan cat sama sama di gunakan untuk hewan priharaan.



Gambar 6.2 Teori 2

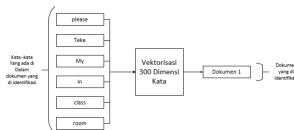
3. Jelaskan konsep vektorisasi untuk kata.dilengkapi dengan ilustrasi atau gambar.
- Vektorisasi untuk kata untuk mengetahui kata tengah dari suatu kalimat atau kata utama atau objek utama pada suatu kalimat contoh ( Jangan lupa subscribe channel saya ya sekian treimakasih ) kata tengah tersebut merupakan channel yang memiliki bobot sebagai kata tengah dari suatu kalimat atau bobot sebagai objek dari suatu kalimat. hal ini sangat berkaitan dengan dimensi vektor pada dataset google yang 300 tadi karena untuk mendapatkan nilai atau bobot dari kata tengah tersebut di dapatkan dari proses dimensi dari kata tersebut.



**Gambar 6.3** Teori 3

4. Jelaskan konsep vektorisasi untuk dokumen, dilengkapi dengan ilustrasi atau gambar.

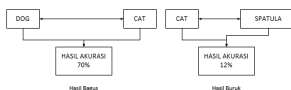
Vektorisasi untuk dokumen hampir sama seperti vektorisasi untuk kata hanya saja pemilihan kata utama atau kata tengah terdapat pada satu dokumen jadi mesin akan membuat dimensi vektor 300 untuk dokumen dan nanti kata tengahnya akan di sandingkan pada dokumen yang terdapat pada dokumen tersebut.



**Gambar 6.4** Teori 4

5. Jelaskan apa mean dan standar deviasi, dilengkapi dengan ilustrasi atau gambar.

mean merupakan petunjuk terhadap kata-kata yang di olah jika kata-kata itu akurasi tinggi berarti kata tersebut sering muncul begitu juga sebaliknya, sedangkan standar deviation merupakan standar untuk meminimalkan kesalahan. sehingga kesalahan tersebut di anggap wajar misalkan kita memperkirakan kedalaman dari dataset merupakan 2 atau 3 tapi pada kenyataannya merupakan 5 itu merupakan kesalahan tapi masih bisa dianggap wajar karna masih mendekati perkiraan awal.

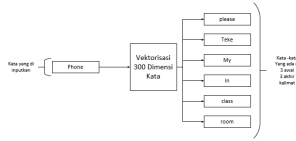


**Gambar 6.5** Teori 5

6. Jelaskan apa itu skip-gram, dilengkapi dengan ilustrasi atau gambar.

Skip-Gram adalah kebalikan dari konsep vektorisasi untuk kata dimana kata tengah menjadi acuan terhadap kata-kata pelengkap dalam suatu

kalimat.



**Gambar 6.6** Teori 6

### 6.1.2 Praktek

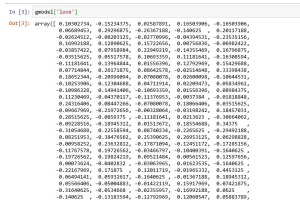
1. Cobalah dataset google, dan jelaskan vektor dari kata love, faith, fall, sick, clear, shine, bag, car, wash, motor, cycle dan cobalah untuk melakukan perbandingan similirati dari masing-masing kata tersebut.

- berikut merupakan code import gensim digunakan untuk membuat data model atau rangkangan data yang akan di buat. selanjutnya dibuat variabel gmodel yang berisi data vektor negativ. selanjutnya data tersebut di load agar data tersebut dapat di tampilkan dan di olah.



**Gambar 6.7**     Praktek 1

- berikut merupakan hasil lpengolahan kata love pada data google yang di load tadi. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.



Gambar 6.8      Praktek 1.1

- berikut merupakan hasil lpengolahan kata faith pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

[illegible]

**Gambar 6.9**      Praktek 1.2

- berikut merupakan hasil lpengolahan kata fall pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

[illegible]

Gambar 6.10      Praktek 1.3

- berikut merupakan hasil lpengolahan kata sick pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

[illegible]

**Gambar 6.11**     Praktek 1.4

- berikut merupakan hasil lpengolahan kata clear pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

- berikut merupakan hasil lpengolahan kata shine pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

[illegible]

- berikut merupakan hasil lpengolahan kata bag pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```

[0x1] [0x00] [0x01] [0x02] [0x03] [0x04] [0x05] [0x06] [0x07] [0x08] [0x09] [0x0A] [0x0B] [0x0C] [0x0D] [0x0E] [0x0F] [0x10] [0x11] [0x12] [0x13] [0x14] [0x15] [0x16] [0x17] [0x18] [0x19] [0x1A] [0x1B] [0x1C] [0x1D] [0x1E] [0x1F] [0x20] [0x21] [0x22] [0x23] [0x24] [0x25] [0x26] [0x27] [0x28] [0x29] [0x2A] [0x2B] [0x2C] [0x2D] [0x2E] [0x2F] [0x30] [0x31] [0x32] [0x33] [0x34] [0x35] [0x36] [0x37] [0x38] [0x39] [0x3A] [0x3B] [0x3C] [0x3D] [0x3E] [0x3F] [0x40] [0x41] [0x42] [0x43] [0x44] [0x45] [0x46] [0x47] [0x48] [0x49] [0x4A] [0x4B] [0x4C] [0x4D] [0x4E] [0x4F] [0x50] [0x51] [0x52] [0x53] [0x54] [0x55] [0x56] [0x57] [0x58] [0x59] [0x5A] [0x5B] [0x5C] [0x5D] [0x5E] [0x5F] [0x60] [0x61] [0x62] [0x63] [0x64] [0x65] [0x66] [0x67] [0x68] [0x69] [0x6A] [0x6B] [0x6C] [0x6D] [0x6E] [0x6F] [0x70] [0x71] [0x72] [0x73] [0x74] [0x75] [0x76] [0x77] [0x78] [0x79] [0x7A] [0x7B] [0x7C] [0x7D] [0x7E] [0x7F] [0x80] [0x81] [0x82] [0x83] [0x84] [0x85] [0x86] [0x87] [0x88] [0x89] [0x8A] [0x8B] [0x8C] [0x8D] [0x8E] [0x8F] [0x90] [0x91] [0x92] [0x93] [0x94] [0x95] [0x96] [0x97] [0x98] [0x99] [0x9A] [0x9B] [0x9C] [0x9D] [0x9E] [0x9F] [0xA0] [0xA1] [0xA2] [0xA3] [0xA4] [0xA5] [0xA6] [0xA7] [0xA8] [0xA9] [0xAA] [0xAB] [0xAC] [0xAD] [0xAE] [0xAF] [0xB0] [0xB1] [0xB2] [0xB3] [0xB4] [0xB5] [0xB6] [0xB7] [0xB8] [0xB9] [0xBA] [0xBB] [0xBC] [0xBD] [0xBE] [0xBF] [0xC0] [0xC1] [0xC2] [0xC3] [0xC4] [0xC5] [0xC6] [0xC7] [0xC8] [0xC9] [0xCA] [0xCB] [0xCC] [0xCD] [0xCE] [0xCF] [0xD0] [0xD1] [0xD2] [0xD3] [0xD4] [0xD5] [0xD6] [0xD7] [0xD8] [0xD9] [0xDA] [0xDB] [0xDC] [0xDD] [0xDE] [0xDF] [0xE0] [0xE1] [0xE2] [0xE3] [0xE4] [0xE5] [0xE6] [0xE7] [0xE8] [0xE9] [0xEA] [0xEB] [0xEC] [0xED] [0xEE] [0xEF] [0xF0] [0xF1] [0xF2] [0xF3] [0xF4] [0xF5] [0xF6] [0xF7] [0xF8] [0xF9] [0xFA] [0xFB] [0xFC] [0xFD] [0xFE] [0xFF]

```

- berikut merupakan hasil lpengolahan kata car pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.



[illegible]

**Gambar 6.15**      Praktek 1.8

- berikut merupakan hasil pengolahan kata wash pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```

[1] 0.94060222-0i, 1.4101052-0i, 1.5667008-0i, 1.3478912-0i,
[2] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[3] 1.0787818-0i, 2.2500000-0i, 1.1152320-0i, 1.1690160-0i,
[4] 1.2478812-0i, 2.6889712-0i, 1.0000000-0i, 1.0000000-0i,
[5] 1.4072818-0i, 2.0467776-0i, 1.2148464-0i, 0.8867737-0i,
[6] 1.7475112-0i, 2.1100000-0i, 1.0000000-0i, 1.0000000-0i,
[7] 2.0000000-0i, 1.2180732-0i, 1.0150376-0i, 2.7107056-0i,
[8] 1.0000000-0i, 1.1807536-0i, 1.0000000-0i, 1.0000000-0i,
[9] 1.8807136-0i, 1.7880576-0i, 1.0000000-0i, 1.1068416-0i,
[10] 1.0000000-0i, 1.7871360-0i, 1.0000000-0i, 1.0000000-0i,
[11] 1.0000000-0i, 2.2071808-0i, 1.0000000-0i, 1.0000000-0i,
[12] 1.0000000-0i, 1.4068736-0i, 1.0000000-0i, 1.0000000-0i,
[13] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[14] 1.0000000-0i, 2.2493184-0i, 1.0000000-0i, 1.0000000-0i,
[15] 1.0000000-0i, 1.7717184-0i, 1.0000000-0i, 1.0000000-0i,
[16] 1.0000000-0i, 6.6151776-0i, 2.5268512-0i, 1.4721088-0i,
[17] 1.0000000-0i, 5.3222528-0i, 2.5268512-0i, 1.4721088-0i,
[18] 1.77890112-0i, 2.74687136-0i, 2.31176896-0i, 1.83107376-0i,
[19] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[20] 1.0000000-0i, 0.8071808-0i, 1.0000000-0i, 2.7545184-0i,
[21] 1.0000000-0i, 1.4762688-0i, 1.0000000-0i, 2.7545184-0i,
[22] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[23] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[24] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[25] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[26] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[27] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[28] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[29] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[30] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[31] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[32] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[33] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[34] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[35] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[36] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[37] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[38] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[39] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[40] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[41] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[42] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[43] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[44] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[45] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[46] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[47] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[48] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[49] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[50] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[51] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[52] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[53] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[54] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[55] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[56] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[57] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[58] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[59] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[60] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[61] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[62] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[63] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[64] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[65] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[66] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[67] 1.0000000-0i, 1.0000000-0i, 1.0000000-0i, 1.0000000-0i,
[68] 1.000
```

Gambar 6.16      Praktek 1.9

- berikut merupakan hasil lpengolahan kata motor pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

[illegible]

**Gambar 6.17**     **Praktek 1.10**

- berikut merupakan hasil lpengolahan kata cycle pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```

[10] [10] array([ 0.0548108,  0.2167008,  0.6270066,  1.2153136,  2.2678125,  3.740125,
            5.540125,  7.640125,  10.040125,  12.740125,  15.740125,  19.040125,
            22.540125,  26.240125,  30.140125,  34.240125,  38.540125,  43.040125,
            47.540125,  52.240125,  57.040125,  61.840125,  66.740125,  71.740125,
            76.740125,  81.840125,  87.040125,  92.240125,  97.540125,  102.840125,
            108.240125,  113.640125,  119.040125,  124.440125,  129.840125,  135.240125,
            140.640125,  146.040125,  151.440125,  156.840125,  162.240125,  167.640125,
            173.040125,  178.440125,  183.840125,  189.240125,  194.640125,  200.040125,
            205.440125,  210.840125,  216.240125,  221.640125,  227.040125,  232.440125,
            237.840125,  243.240125,  248.640125,  254.040125,  259.440125,  264.840125,
            270.240125,  275.640125,  281.040125,  286.440125,  291.840125,  297.240125,
            302.640125,  308.040125,  313.440125,  318.840125,  324.240125,  329.640125,
            335.040125,  340.440125,  345.840125,  351.240125,  356.640125,  362.040125,
            367.440125,  372.840125,  378.240125,  383.640125,  389.040125,  394.440125,
            399.840125,  405.240125,  410.640125,  416.040125,  421.440125,  426.840125,
            432.240125,  437.640125,  443.040125,  448.440125,  453.840125,  459.240125,
            464.640125,  470.040125,  475.440125,  480.840125,  486.240125,  491.640125,
            497.040125,  502.440125,  507.840125,  513.240125,  518.640125,  524.040125,
            529.440125,  534.840125,  540.240125,  545.640125,  551.040125,  556.440125,
            561.840125,  567.240125,  572.640125,  578.040125,  583.440125,  588.840125,
            594.240125,  599.640125,  605.040125,  610.440125,  615.840125,  621.240125,
            626.640125,  632.040125,  637.440125,  642.840125,  648.240125,  653.640125,
            659.040125,  664.440125,  669.840125,  675.240125,  680.640125,  686.040125,
            691.440125,  696.840125,  702.240125,  707.640125,  713.040125,  718.440125,
            723.840125,  729.240125,  734.640125,  740.040125,  745.440125,  750.840125,
            756.240125,  761.640125,  767.040125,  772.440125,  777.840125,  783.240125,
            788.640125,  794.040125,  799.440125,  804.840125,  810.240125,  815.640125,
            821.040125,  826.440125,  831.840125,  837.240125,  842.640125,  848.040125,
            853.440125,  858.840125,  864.240125,  869.640125,  875.040125,  880.440125,
            885.840125,  891.240125,  896.640125,  902.040125,  907.440125,  912.840125,
            918.240125,  923.640125,  929.040125,  934.440125,  939.840125,  945.240125,
            950.640125,  956.040125,  961.440125,  966.840125,  972.240125,  977.640125,
            983.040125,  988.440125,  993.840125,  999.240125,  1004.640125,  1010.040125,
            1015.440125,  1020.840125,  1026.240125,  1031.640125,  1037.040125,  1042.440125,
            1047.840125,  1053.240125,  1058.640125,  1064.040125,  1069.440125,  1074.840125,
            1080.240125,  1085.640125,  1091.040125,  1096.440125,  1101.840125,  1107.240125,
            1112.640125,  1118.040125,  1123.440125,  1128.840125,  1134.240125,  1139.640125,
            1145.040125,  1150.440125,  1155.840125,  1161.240125,  1166.640125,  1172.040125,
            1177.440125,  1182.840125,  1188.240125,  1193.640125,  1199.040125,  1204.440125,
            1209.840125,  1215.240125,  1220.640125,  1226.040125,  1231.440125,  1236.840125,
            1242.240125,  1247.640125,  1253.040125,  1258.440125,  1263.840125,  1269.240125,
            1274.640125,  1280.040125,  1285.440125,  1290.840125,  1296.240125,  1301.640125,
            1307.040125,  1312.440125,  1317.840125,  1323.240125,  1328.640125,  1334.040125,
            1339.440125,  1344.840125,  1350.240125,  1355.640125,  1361.040125,  1366.440125,
            1371.840125,  1377.240125,  1382.640125,  1388.040125,  1393.440125,  1398.840125,
            1404.240125,  1409.640125,  1415.040125,  1420.440125,  1425.840125,  1431.240125,
            1436.640125,  1442.040125,  1447.440125,  1452.840125,  1458.240125,  1463.640125,
            1469.040125,  1474.440125,  1479.840125,  1485.240125,  1490.640125,  1496.040125,
            1501.440125,  1506.840125,  1512.240125,  1517.640125,  1523.040125,  1528.440125,
            1533.840125,  1539.240125,  1544.640125,  1550.040125,  1555.440125,  1560.840125,
            1566.240125,  1571.640125,  1577.040125,  1582.440125,  1587
```

**Gambar 6.18**     Praktek 1.11

- berikut merupakan hasil dari similaritas kata kata yang di olah menjadi matrix tadi adapun persentase untuk perbandingan setiap katanya yaitu 9 persen untuk kata wash dan clear 7 persen untuk kata bag dan love 48 persen untuk kata motor dan car 12 persen untuk kata sick dan faith dan terakhir yaitu 6 persen untuk kata cycle dan shine.

```
In [15]: gmodel.similarity('wash', 'clean')
Out[15]: 0.00019176

In [16]: gmodel.similarity('bag', 'love')
Out[16]: 0.07536006

In [17]: gmodel.similarity('motor', 'car')
Out[17]: 0.4810173

In [18]: gmodel.similarity('sick', 'faith')
Out[18]: 0.123873205

In [19]: gmodel.similarity('cycle', 'shine')
Out[19]: 0.001617922
```

Gambar 6.19      Praktek 1.12

2. Jelaskan dengan kata dan ilustrasi fungsi dari `extract words` dan `PermuteSentences`

```
In [28]: import re
         def extract_words(sent):
             sent = sent.lower()
             re_sub1 = re.sub('([^\w\s])', ' ', sent) #hapus tag html
             re_sub2 = re.sub('([^\w\s\'])([^\w\s]{2,})', ' ', sent) #hapus partik satu
             re_sub3 = re.sub('\'([^\w\s]{2,})', ' ', sent) #hapus tanda baca
             re_sub4 = re.sub('([^\w\s]{2,})', ' ', sent) #hapus spasi yang berlebihan
             return sent.split()

In [29]: import random
         class PermuteSentences(object):
             def __init__(self, sents):
                 self.sents = sents

             def __iter__(self):
                 shuffled = list(self.sents)
                 random.shuffle(shuffled)
                 for sent in shuffled:
                     yield sent
```

Gambar 6.20      Praktek 2

3. Jelaskan fungsi dari library gensim TaggedDocument dan Doc2Vec disertai praktek pemakaiannya.

```
In [22]: from gensim.models.doc2vec import TaggedDocument
         from gensim.models import Doc2Vec
```

**Gambar 6.21**    Praktek 3

4. Jelaskan dengan kata dan praktek cara menambahkan data training dari file yang dimasukkan kepada variabel dalam rangka melatih model doc2vec.

[illegible]

**Gambar 6.22**     Praktek 4

5. Jelaskan dengan kata dan praktek kenapa harus dilakukan pengocokan dan pembersihan data.

[illegible]

**Gambar 6.23**     Praktek 5

6. Jelaskan dengan kata dan praktek kenapa model harus di save dan kenapa temporari training harus dihapus.

```
In [78]: model.delete_temporary_training_data(keep_inference=True)
         model.save('haci.d2v')
```

**Gambar 6.24**     Praktek 6

7. Jalankan dengan kta dan praktek maksud dari infer code.

```
In [79]: model.infer_vector(extract_words("I will go home"))

Out[79]: array([-1.7533908e-01, 1.7054950e-01, 0.7283678e-01, 1.7399823e-02,
-3.5951584e-01, -3.2540950e-01, 0.0000000e+00, -1.9559191e-01, -1.0000000e-01,
-1.4734272e-02, 0.64738091e-01, 1.6690316e-02, 0.2400551e-02, -2.3931590e-01,
7.7021412e-02, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
-0.38780181e-01, 0.00150770e-01, -5.1534814e-02, 7.2401073e-02, -1.3899844e-02,
-2.4902820e-02, -0.2386778e-01, 7.3931477e-02, 0.19152211e-01, 0.15955802e-01,
-2.3586474e-01, 1.0000000e-01, 1.4284551e-01, -1.4341866e-01, 1.3372188e-01, -0.6272384e-02,
-2.4081554e-01, 0.8382756e-01, -1.2425344e-01, -3.1085350e-02, -0.6935650e-02,
7.1126200e-02, 0.6425000e-02, -0.9451536e-02, -0.0073090e-04, 1.0074811e-02, 0.5924744e-02, 3.3133272e-03,
5.3743320e-02, -1.9093333e-02, 0.0000000e+00, 4.9782320e-02, -2.3787551e-02,
2.5583537e-02, 0.0000000e+00, 0.0000000e+00])
```

Gambar 6.25     Praktek 7

8. Jelaskan dengan praktek dan kata maksud dari cosine similarity.

```

In [8]: from sklearn.metrics.pairwise import cosine_similarity
        cosine_similarity(
            [model_infer_vector.extract_words(sentence) for sentence in tweets, after_nltk_tokenize()]),
            [model_infer_vector.extract_words(sentence) for sentence in tweets, after_nltk_tokenize()])

Out[8]: array([[0.06740229]], dtype=float32)

In [9]: from sklearn.metrics.pairwise import cosine_similarity
        cosine_similarity(
            [model_infer_vector.extract_words(sentence) for sentence in tweets, after_nltk_tokenize()]),
            [model_infer_vector.extract_words(sentence) for sentence in tweets, after_nltk_tokenize()])

Out[9]: array([[0.06654411]], dtype=float32)

```

Gambar 6.26 Praktek 8

9. Jelaskan dengan praktek score dari cross validation masing-masing metode.

```

1 from sklearn.neighbors import KNeighborsClassifier
2 from sklearn.ensemble import RandomForestClassifier
3 from sklearn.model_selection import cross_val_score
4 import numpy as np
5
6 clf = KNeighborsClassifier(n_neighbors=9)
7 clfrf = RandomForestClassifier()
8
9 scores = cross_val_score(clf, sentvecs, sentiments, cv=5)
10 print((np.mean(scores), np.std(scores)))
11
12 scores = cross_val_score(clfrf, sentvecs, sentiments, cv=5)
13 print((np.mean(scores), np.std(scores)))
14
15 # bag-of-words comparison
16 from sklearn.pipeline import make_pipeline
17 from sklearn.feature_extraction.text import CountVectorizer,
    TfidfTransformer
18 pipeline = make_pipeline(CountVectorizer(), TfidfTransformer(),
    RandomForestClassifier())
19 scores = cross_val_score(pipeline, sentences, sentiments, cv
    =5)
20 print((np.mean(scores), np.std(scores)))

```

### 6.1.3 Penangan Error

#### 1. SS Error

```

File "D:/Guliah/Semester 6/Keberhasilan Buktan/Bahan/src/1174027/5/1174027.py",
line 36, in <module>
    scores = cross_val_score(clf, sentvecs, sentiments, cv=5)
NameError: name 'sentvecs' is not defined

```

Gambar 6.27 Name Error

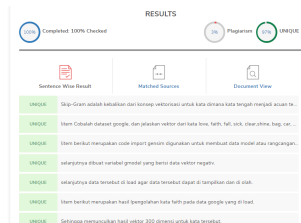
#### 2. Jenis error

- Name Error

#### 3. Cara Penanganan

Dengan Mengecek Kembali Baris Code Yang Dibuat

## 6.1.4 Bukti Tidak Melakukan Plagiat



Gambar 6.28 Bukti Tidak Melakukan Plagiat

## 6.1.5 Link Youtube

<https://youtu.be/NKurTLe8MpI>



## BAB 7

---

## CHAPTER 6

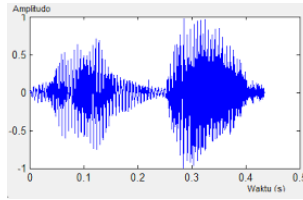
---

### 7.1 1174069 - Fanny Shafira Damayanti

#### 7.1.1 Teori

1. Kenapa file suara harus di lakukan MFCC. dilengkapi dengan ilustrasi atau gambar.

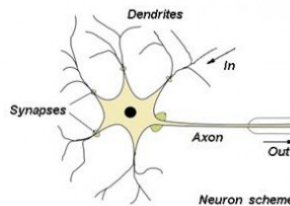
Nilai-nilai MFCC meniru pendengaran manusia dan mereka biasanya digunakan dalam aplikasi pengenalan suara serta genre musik deteksi. Nilai-nilai MFCC ini akan dimasukkan langsung ke jaringan saraf. Agar dapat diubah menjadi bentuk vektor, dan dapat digunakan pada machine learning. Disebabkan machine learning hanya mengerti bilangan vektor saja.



**Gambar 7.1** Contoh MFCC

Ilustrasinya, Ketika ingin menggunakan file suara dalam machine learning, misalnya untuk melihat jam. Machine learning tidak memahami rekaman suara melainkan vektor. Maka rekaman tersebut akan diubah kedalam bentuk vektor kemudian vektor akan menyesuaikan dengan kata-kata yang sudah disediakan. Jika cocok maka akan mengembalikan waktu yang diinginkan

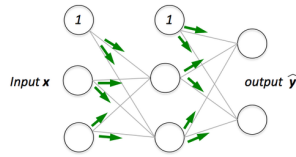
2. Konsep dasar neural network dilengkapi dengan ilustrasi atau gambar. Neural Network ini terinspirasi dari jaringan saraf otak manusia. Dimana setiap neuron terhubung ke setiap neuron di lapisan berikutnya. Lapisan pertama menerima input dan lapisan terakhir memberikan keluaran. Struktur jaringan, yang berarti jumlah neuron dan koneksinya, diputuskan sebelumnya dan tidak dapat berubah, setidaknya tidak selama training. Juga, setiap input harus memiliki jumlah nilai yang sama. Ini berarti bahwa gambar, misalnya, mungkin perlu diubah ukurannya agar sesuai dengan jumlah neuron input.



**Gambar 7.2** Contoh Pembobotan Neural Network

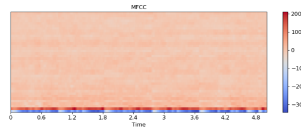
3. Konsep pembobotan dalam neural network. dilengkapi dengan ilustrasi atau gambar. Bobot mewakili kekuatan koneksi antar unit. Jika bobot dari node 1 ke node 2 memiliki besaran lebih besar, itu berarti bahwa neuron 1 memiliki pengaruh lebih besar terhadap neuron 2. Bobot penting untuk nilai input. Bobot mendekati nol berarti mengubah input ini tidak akan mengubah output. Bobot negatif berarti meningkatkan input ini akan mengurangi output. Bobot menentukan seberapa besar pengaruh input terhadap output. Seperti contoh berikut :





**Gambar 7.3** Contoh Pembobotan Neural Network

4. Konsep fungsi aktivasi dalam neural network. dilengkapi dengan ilustrasi atau gambar  
 Fungsi aktivasi digunakan untuk memperkenalkan non-linearitas ke jaringan saraf. Ini menekan nilai dalam rentang yang lebih kecil yaitu. fungsi aktivasi Sigmoid memeras nilai antara rentang 0 hingga 1. Ada banyak fungsi aktivasi yang digunakan dalam industri pembelajaran yang dalam dan ReLU, SeLU dan TanH lebih disukai daripada fungsi aktivasi sigmoid. Ilustrasinya, ketika fungsi aktivasi linier, jaringan saraf dua lapis mampu mendekati hampir semua fungsi. Namun, jika fungsi aktivasi identik dengan fungsi aktivasi  $F(X) = X$ , properti ini tidak puas, dan jika MLP menggunakan fungsi aktivasi yang sama, seluruh jaringan setara dengan jaringan saraf lapis tunggal.
5. Cara membaca hasil plot dari MFCC, dilengkapi dengan ilustrasi atau gambar  
 Berikut merupakan hasil plot dari rekaman suara :



**Gambar 7.4** Cara Membaca Hasil Plot MFCC

Dari gambar tersebut dapat diketahui :

- Terdapat 2 dimensi yaitu x sebagai waktu, dan y sebagai power atau desibel.
  - Dapat dilihat bahwa jika berwarna biru maka power dari suara tersebut rendah, dan jika merah power dari suara tersebut tinggi
  - Dibagian atas terdapat warna merah pudar yang menandakan bahwa tidak ada suara sama sekali dalam jangkauan tersebut.
6. Jelaskan apa itu one-hot encoding, dilengkapi dengan ilustrasi kode dan atau gambar.  
 One-hot encoding adalah representasi variabel kategorikal sebagai vektor

biner. Mengharuskan nilai kategorikal dipetakan ke nilai integer. Kemudian, setiap nilai integer direpresentasikan sebagai vektor biner yang semuanya bernilai nol kecuali indeks integer, yang ditandai dengan 1.

Label Encoding			One Hot Encoding			
Food Name	Categorical #	Calories				
Apple	1	95	Apple	Chicken	Broccoli	Calories
Apple	1	95	1	0	0	95
Chicken	2	231	0	1	0	231
Broccoli	3	55	0	0	1	55

**Gambar 7.5** One Hot Encoding

7. Fungsi dari `np/.unique` dan `to categorical` dalam kode program, dilengkapi dengan ilustrasi atau gambar.

Untuk `np unique` fungsinya yaitu menemukan elemen unik array. Mengembalikan elemen unik array yang diurutkan. Ada tiga output opsional selain elemen unik:

- Indeks array input yang memberikan nilai unik
- Indeks array unik yang merekonstruksi array input
- Berapa kali setiap nilai unik muncul dalam array input.

```
>>> np.unique([1, 1, 2, 2, 3, 3])
array([1, 2, 3])
>>> a = np.array([[1, 1], [2, 3]])
>>> np.unique(a)
array([1, 2, 3])
```

**Gambar 7.6** Numpy Unique

Untuk `To Categorical` fungsinya untuk mengubah vektor kelas (integer) ke matriks kelas biner.

```
# Consider an array of 5 labels out of a set of 3 classes (0, 1, 2):
> labels
array([0, 2, 1, 1, 2, 0])
# to_categorical: converts this into a matrix with as many
# columns as there are classes. The number of rows
# stays the same.
> to_categorical(labels)
array([[ 1.,  0.,  0.],
       [ 0.,  0.,  1.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.],
       [ 1.,  0.,  0.]])
```

**Gambar 7.7** To Categorical

8. Fungsi dari `Sequential` dalam kode program, dilengkapi dengan ilustrasi atau gambar.

`Sequential` berfungsi sebagai tumpukan linear lapisan. Contohnya sebagai berikut :

```

from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(2, input_dim=1))
model.add(Dense(1))

```

Gambar 7.8 Sequential

### 7.1.2 Praktek

1. Penjelasan isi data GTZAN Genre Collection dan data dari Freesound.

```

1 import librosa
2 import librosa.feature
3 import librosa.display
4 import glob
5 import numpy as np
6 import matplotlib.pyplot as plt
7 from keras.layers import Dense, Activation
8 from keras.models import Sequential
9 from keras.utils.np_utils import to_categorical
10
11 def display_mfcc(song):
12     y, _ = librosa.load(song)
13     mfcc = librosa.feature.mfcc(y)
14
15     plt.figure(figsize=(10, 4))
16     librosa.display.specshow(mfcc, x-axis='time', y-axis='mel')
17     plt.colorbar()
18     plt.title(song)
19     plt.tight_layout()
20     plt.show()

```

2. Penjelasan perbaris kode program dari display MFCC.

```

1 display_mfcc('test1.wav')
2 display_mfcc('test2.wav')
3 display_mfcc('dataset/genres/disco/disco.00069.au')
4 display_mfcc('dataset/genres/blues/blues.00069.au')
5 display_mfcc('dataset/genres/classical/classical.00069.au')
6 display_mfcc('dataset/genres/country/country.00069.au')
7 display_mfcc('dataset/genres/hiphop/hiphop.00069.au')
8 display_mfcc('dataset/genres/jazz/jazz.00069.au')
9 display_mfcc('dataset/genres/pop/pop.00069.au')
10 display_mfcc('dataset/genres/reggae/reggae.00069.au')
11 display_mfcc('dataset/genres/rock/rock.00069.au')

```

3. Penjelasan perbaris code dari Extract Feature Song.

```

1 def extract_features_song(f):
2     y, _ = librosa.load(f)
3
4     # get Mel-frequency cepstral coefficients
5     mfcc = librosa.feature.mfcc(y)

```

```

6     # normalize values between -1,1 (divide by max)
7     mfcc /= np.amax(np.absolute(mfcc))
8
9     return np.ndarray.flatten(mfcc)[:25000]

```

#### 4. Penjelasan perbaris code dari Generate Features and Labels.

```

1 def generate_features_and_labels():
2     all_features = []
3     all_labels = []
4
5     genres = ['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz', 'metal', 'pop', 'reggae', 'rock']
6     for genre in genres:
7         sound_files = glob.glob('dataset/genres/'+genre+'/*.au')
8         print('Processing %d songs in %s genre...' % (len(sound_files), genre))
9         for f in sound_files:
10             features = extract_features_song(f)
11             all_features.append(features)
12             all_labels.append(genre)
13
14     # convert labels to one-hot encoding cth blues :
15     # 1000000000 classic 0100000000
16     label_uniq_ids, label_row_ids = np.unique(all_labels, return_inverse=True)
17     #ke integer
18     label_row_ids = label_row_ids.astype(np.int32, copy=False)
19
20     onehot_labels = to_categorical(label_row_ids, len(label_uniq_ids))
21     #ke one hot
22     return np.stack(all_features), onehot_labels

```

#### 5. Penjelasan penggunaan fungsi Generate Features and Labels sangat lama ketika Meload Dataset Genre.

```

1 features, labels = generate_features_and_labels()
2 print(np.shape(features))
3 print(np.shape(labels))

```

#### 6. Kenapa harus dilakukan pemisahan data training dan dataset sebesar 80%

```

1 training_split = 0.8
2 alldata = np.column_stack((features, labels))
3 np.random.shuffle(alldata)
4 splitidx = int(len(alldata) * training_split)
5 train, test = alldata[:splitidx, :], alldata[splitidx :, :]
6 print(np.shape(train))
7 print(np.shape(test))
8 train_input = train[:, :-10]
9 train_labels = train[:, -10:]
10 test_input = test[:, :-10]
11 test_labels = test[:, -10:]
12 print(np.shape(train_input))
13 print(np.shape(train_labels))

```

## 7. Parameter dari fungsi Sequential().

```
1 model = Sequential([
2     Dense(100, input_dim=np.shape(train_input)[1]),
3     Activation('relu'),
4     Dense(10),
5     Activation('softmax'),
6     ])
```

## 8. Parameter dari fungsi Compile().

```
1 model.compile(optimizer='adam',
2               loss='categorical_crossentropy',
3               metrics=['accuracy'])
4 print(model.summary())
```

## 9. Parameter dari fungsi Fit().

```
1 model.fit(train_input, train_labels, epochs=10, batch_size
2           =32,
3           validation_split=0.2)
```

## 10. Parameter dari fungsi Evaluate().

```
1 loss, acc = model.evaluate(test_input, test_labels,
2                             batch_size=32)
3 print("Done!")
3 print("Loss: %.4f, accuracy: %.4f" % (loss, acc))
```

## 11. Parameter dari fungsi Predict().

```
1 model.predict(test_input[:1])
```

### 7.1.3 Penanganan Error

#### 1. SS Error

```
File "D:\Kuliah\Semester 6\Kecerdasan Buatan\Bahan\src\1174069\1174069.py", line 9,
in module:
ImportError:
ModuleNotFoundError: No module named 'librosa'
```

**Gambar 7.9** No Module Name error

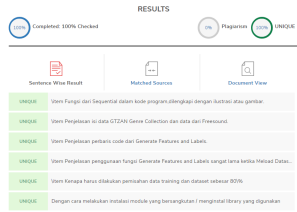
#### 2. Jenis Error

- No Module

#### 3. Cara Penanganan

Dengan cara melakukan instalasi module yang bersangkutan / menginstal library yang digunakan

## 7.1.4 Bukti Tidak Plagiat



**Gambar 7.10** Tidak Melakukan Plagiat Pada Ch 6

## 7.1.5 Link Youtube

<https://youtu.be/eeLATXvf8bE>

## BAB 8

---

## CHAPTER 7

---

### 8.1 1174006 - Kadek Diva Krishna Murti

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

```
1 @inproceedings{awangga2017colenak ,
2   title={Colenak: GPS tracking model for post-stroke
3     rehabilitation program using AES-CBC URL encryption and QR-
4     Code},
5   author={Awangga, Rolly Maulana and Fathonah, Nuraini Siti and
6     Hasanudin, Trisna Irmayadi},
7   booktitle={Information Technology, Information Systems and
8     Electrical Engineering (ICITISEE), 2017 2nd International
9     conferences on},
10  pages={255--260},
11  year={2017},
12  organization={IEEE}
13 }
```



**Gambar 8.1** Kecerdasan Buatan.

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
2. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
3. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

### 8.1.1 Teori

### 8.1.2 Praktek

### 8.1.3 Penanganan Error

### 8.1.4 Bukti Tidak Plagiat



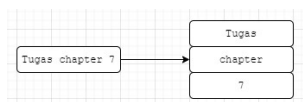
**Gambar 8.2** Kecerdasan Buatan.

## 8.2 1174066 - D.Irga B. Naufal Fakhri

### 8.2.1 Teori

#### 8.2.1.1 Kenapa file teks harus di lakukan tokenizer

Karena MTokenizer adalah proses membagi teks yang berupa kalimat, paragraf atau dokumen menjadi kata-kata atau bagian-bagian tertentu dalam kalimat tersebut. Contohnya kalimat "Tugas chapter 7", kalimat itu menjadi beberapa bagian yaitu "Tugas", "chapter", "7". Yang menjadi acuan yakni tanda baca dan spasi.



**Gambar 8.3** Teori 1

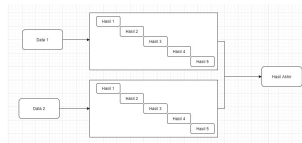


### 8.2.1.2 konsep dasar K Fold Cross Validation pada dataset komentar Youtube pada kode listing 7.1.

Konsep sederhana dari K Fold Cross Validation ialah Pada code ini:

```
1 kfold = StratifiedKfold(n_splits=5)
2 splits = kfold.split(d, d['CLASS'])
```

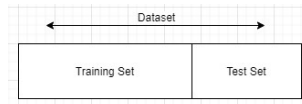
terdapat kfold yang bertujuan untuk melakukan split data menjadi 5 bagian dari dataset komentar Youtube tersebut. Sehingga dari setiap data yang sudah dibagi tersebut akan menghasilkan presentase dari setiap bagiannya, untuk menghasilkan hasil akhir dengan presentase yang cukup baik.



**Gambar 8.4** Teori 2

### 8.2.1.3 Apa maksudnya kode program for train, test in splits

For train berfungsi untuk membagi data tersebut menjadi data training. Sedangkan test in splits berfungsi untuk menguji apakah dataset tersebut sudah dibagi menjadi beberapa bagian atau masih menumpuk.



**Gambar 8.5** Teori 3

### 8.2.1.4 Apa maksudnya kode program train content = d['CONTENT'].iloc[train idx] dan test content = d['CONTENT'].iloc[test idx]

Fungsi dalam kode tersebut berfungsi untuk mengambil data pada kolom atau index CONTENT yang merupakan bagian dari train.idx dan test.idx. Contoh sederhananya ketika data telah diubah menjadi data train dan data test maka kita dapat memilihnya untuk ditampilkan pada kolom yang di inginkan.

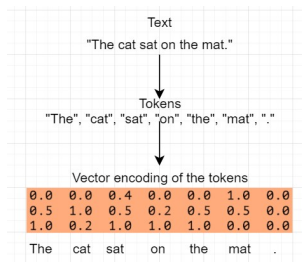
```
1 import pandas as pd
2
3 mydict = [{ 'a': 1, 'b': 2, 'c': 3, 'd': 4},
4           { 'a': 100, 'b': 200, 'c': 300, 'd': 400},
5           { 'a': 1000, 'b': 2000, 'c': 3000, 'd': 4000 } }
6 df = pd.DataFrame(mydict)
7 df
```

```
Out[2]:
a      1
b      2
c      3
d      4
Name: 0, dtype: int64
```

Gambar 8.6 Teori 4

### 8.2.1.5 Apa maksud dari fungsi `tokenizer = Tokenizer(num words=2000)` dan `tokenizer.fit on texts(train content)`

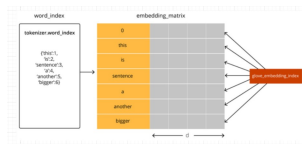
Fungsi tokenizer berfungsi untuk melakukan vektorisasi data kedalam bentuk token sebanyak 2000 kata. Dan selanjutnya akan melakukan fit tokenizer hanya untuk data training saja tidak dengan data testingnya.



Gambar 8.7 Teori 5

### 8.2.1.6 Apa maksud dari fungsi `d train inputs = tokenizer.texts to matrix(train content, mode='tfidf')` dan `d test inputs = tokenizer.texts to matrix(test content, mode='tfidf')`

Maksud dari baris diatas ialah untuk memasukkan text ke sebuah matrix dengan mode tfidf dan menginputkan data testing untuk di terjemahkan ke sebuah matriks.



Gambar 8.8 Teori 6

### 8.2.1.7 Apa maksud dari fungsi `d train inputs = d train inputs/np.amax(np.absolute(d train inputs))` dan `d test inputs = d test inputs/np.amax(np.absolute(d test inputs))`

Fungsi `np.amax` adalah nilai Maksimal. Jika sumbu tidak ada, hasilnya adalah nilai skalar. Jika sumbu diberikan, hasilnya adalah array dimensi `a.ndim - 1`.

```

>>> a = np.arange(4).reshape((2,2))
>>> a
array([[0, 1],
       [2, 3]])
>>> np.amax(a)           # Maximum of the flattened array
3
>>> np.amax(a, axis=0)   # Maxima along the first axis
array([2, 3])
>>> np.amax(a, axis=1)   # Maxima along the second axis
array([1, 3])
>>> np.amax(a, where=[False, True], initial=-1, axis=0)
array([-1, 3])
>>> b = np.arange(5, dtype=float)
>>> b[2] = np.NaN
>>> np.amax(b)
nan
>>> np.amax(b, where=~np.isnan(b), initial=-1)
4.0
>>> np.nanmax(b)
4.0

```

Gambar 8.9 Teori 7

8.2.1.8 Apa maksud fungsi dari `d train outputs = np utils.to categorical(d['CLASS'].iloc[train idx])` dan `d test outputs = np utils.to categorical(d['CLASS'].iloc[test idx])` dalam kode program

Fungsi dari baris kode tersebut ialah membuat train outputs dengan kategori dari class lalu dengan ketentuan iloc train idx. Kemudian membuat keluaran sebagai output.

```

>>> V_train
array([[1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       ...,
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.]])
>>> V_train.flatten()
array([1., 0., 0., ..., 1., 0., 0.], dtype=int64)
>>> V_train
array([[1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       ...,
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.]])
>>> np_utils.to_categorical(V_train)
array([[1., 1.],
       [1., 1.],
       [1., 1.],
       ...,
       [1., 1.],
       [1., 1.],
       [1., 1.]])

```

Gambar 8.10 Teori 8

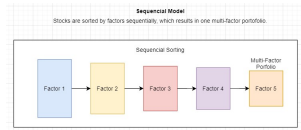
8.2.1.9 Apa maksud dari fungsi di listing 7.2.

```

1 model = Sequential()
2 model.add(Dense(512, input_shape=(2000,)))
3 model.add(Activation('relu'))
4 model.add(Dropout(0.5))
5 model.add(Dense(2))
6 model.add(Activation('softmax'))

```

Fungsi dari baris kode tersebut ialah model perlu mengetahui bentuk input apa yang harus diharapkan. Untuk alasan ini, lapisan pertama dalam model Sequential (dan hanya yang pertama, karena lapisan berikut dapat melakukan inferensi bentuk otomatis) perlu menerima informasi tentang bentuk inputnya.

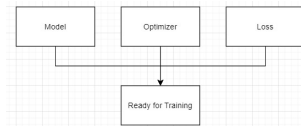


Gambar 8.11 Teori 9

#### 8.2.1.10 Apa maksud dari fungsi di listing 7.3 dengan parameter tersebut

```
1 model.compile(loss='categorical_crossentropy', optimizer='adamax',
2               metrics=['accuracy'])
```

Fungsi dari baris kode tersebut ialah bisa meneruskan nama fungsi loss yang ada, atau melewati fungsi simbolis TensorFlow yang mengembalikan skalar untuk setiap titik data dan mengambil dua argumen `y_true`: True label. dan `y_pred`: Prediksi. Tujuan yang dioptimalkan sebenarnya adalah rata-rata dari array output di semua titik data.



Gambar 8.12 Teori 10

#### 8.2.1.11 Apa itu Deep Learning

Deep Learning adalah salah satu cabang dari ilmu machine learning yang terdiri dari algoritma pemodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mendalam. Teknik dan algoritma dalam machine learning dapat digunakan baik untuk supervised learning dan unsupervised learning.

#### 8.2.1.12 Apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning

Deep Neural Network adalah salah satu algoritma berbasis jaringan saraf tiruan yang memiliki dari 1 lapisan saraf tersembunyi yang dapat digunakan untuk pengambilan keputusan. Perbedaannya dengan deep learning, yakni: Deep Neural Network dapat menentukan dan mencerna karakteristik tertentu di suatu rangkaian data, kapabilitas lebih kompleks untuk mempelajari, mencerna, dan mengklasifikasikan data, serta dibagi ke dalam berbagai lapisan dengan fungsi yang berbeda-beda.

#### 8.2.1.13 Jelaskan dengan ilustrasi gambar buatan sendiri(langkah per langkah) bagaimana perhitungan algoritma konvolusi dengan ukuran stride ( $NPM \bmod 3 + 1$ ) $\times$ ( $NPM \bmod 3 + 1$ ) yang terdapat max pooling.

Konvolusi terdapat pada operasi pengolahan citra yang mengalikan sebuah citra dengan sebuah mask atau kernel, Stride adalah parameter yang berfungsi untuk menentukan pergeseran pada filter data pixel yang terjadi. untuk contoh penggunaannya:



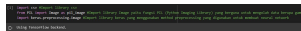
**Gambar 8.13** Teori 11

## 8.2.2 Praktek

### 8.2.2.1 Nomor 1

```
1 import csv #Import library csv
2 from PIL import Image as pil_image #Import library Image yaitu
  fungsi PIL (Python Imaging Library) yang berguna untuk
  mengolah data berupa gambar
3 import keras.preprocessing.image #Import library keras yang
  menggunakan method preprocessing yang digunakan untuk membuat
  neural network
```

Hasil:



**Gambar 8.14** Hasil No 1

### 8.2.2.2 Nomor 2

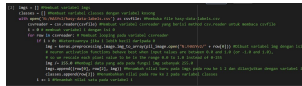
```
1 imgs = [] #Membuat variabel imgs
2 classes = [] #Membuat variabel classes dengan variabel kosong
3 with open('N:/HASYv2/hasy-data-labels.csv') as csvfile: #Membuka
  file hasy-data-labels.csv
4 csvreader = csv.reader(csvfile) #Membuat variabel csvreader
  yang berisi method csv.reader untuk membaca csvfile
5 i = 0 # membuat variabel i dengan isi 0
6 for row in csvreader: # Membuat looping pada variabel
  csvreader
7     if i > 0: #Ketentuannya jika i lebih kecil daripada 0
8         img = keras.preprocessing.image.img_to_array(
  pil_image.open("N:/HASYv2/" + row[0])) #Dibuat variabel img
  dengan isi keras untuk aktivasi neural network fungsi yang
  membaca data yang berada dalam folder HASYv2 dengan input
  nilai -1.0 dan 1.0
9         # neuron activation functions behave best when input
  values are between 0.0 and 1.0 (or -1.0 and 1.0),
10        # so we rescale each pixel value to be in the range
  0.0 to 1.0 instead of 0-255
11        img /= 255.0 #Membagi data yang ada pada fungsi img
  sebanyak 255.0
```

```

12     imgs.append((row[0], row[2], img)) #Menambah nilai
    baru pada imgs pada row ke 1 2 dan dilanjutkan dengan
    variabel img
13     classes.append(row[2]) #Menambahkan nilai pada row ke
    2 pada variabel classes
14     i += 1 #Menambah nilai satu pada variabel i

```

Hasil:



Gambar 8.15 Hasil No 2

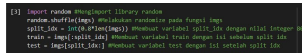
### 8.2.2.3 Nomor 3

```

1 import random #Mengimport library random
2 random.shuffle(imgs) #Melakukan randomize pada fungsi imgs
3 split_idx = int(0.8*len(imgs)) #Membuat variabel split_idx dengan
    nilai integer 80 persen dikali dari pengembalian jumlah dari
    variabel imgs
4 train = imgs[:split_idx] #Membuat variabel train dengan isi
    sebelum split_idx
5 test = imgs[split_idx:] #Membuat variabel test dengan isi setelah
    split_idx

```

Hasil:



Gambar 8.16 Hasil No 1

### 8.2.2.4 Nomor 4

```

1 import numpy as np #Mengimport library numpy dengan inisial np
2 train_input = np.asarray(list(map(lambda row: row[2], train))) #
    Membuat variabel train_input dengan np method asarray yang
    mana membuat array dengan isi row 2 dari data train
3 test_input = np.asarray(list(map(lambda row: row[2], test))) #
    Membuat test input input dengan np method asarray yang mana
    membuat array dengan isi row 2 dari data test
4 train_output = np.asarray(list(map(lambda row: row[1], train))) #
    Membuat variabel train_output dengan np method asarray yang
    mana membuat array dengan isi row 1 dari data train
5 test_output = np.asarray(list(map(lambda row: row[1], test))) #
    Membuat variabel test_output dengan np method asarray yang
    mana membuat array dengan isi row 1 dari data test

```

Hasil:

```
[4] import numpy as np #mengimport library numpy dengan initial np
train_input = np.asarray(list(map(lambda row: row[2], train))) #membuat
test_input = np.asarray(list(map(lambda row: row[2], test))) #membuat
train_output = np.asarray(list(map(lambda row: row[1], train))) #membuat
test_output = np.asarray(list(map(lambda row: row[1], test))) #membuat
```

Gambar 8.17 Hasil No 4

### 8.2.2.5 Nomor 5

```
1 from sklearn.preprocessing import LabelEncoder #Mengimport
  library LabelEncode dari sklearn
2 from sklearn.preprocessing import OneHotEncoder #Mengimport
  library OneHotEncoder dari sklearn
```

Hasil:

```
[2] from sklearn.preprocessing import LabelEncoder #Mengimport library
  LabelEncode dari sklearn
from sklearn.preprocessing import OneHotEncoder #Mengimport library
  OneHotEncoder dari sklearn
```

Gambar 8.18 Hasil No 5

### 8.2.2.6 Nomor 6

```
1 label_encoder = LabelEncoder() #Membuat variabel label_encoder
  dengan isi LabelEncoder
2 integer_encoded = label_encoder.fit_transform(classes) #Membuat
  variabel integer_encoded yang berfungsi untuk mengkonvert
  variabel classes kedalam bentuk integer
```

Hasil:

```
[6] label_encoder = LabelEncoder() #membuat variabel label_encoder dengan
  isi LabelEncoder
integer_encoded = label_encoder.fit_transform(classes) #membuat variabel
  integer_encoded
```

Gambar 8.19 Hasil No 6

### 8.2.2.7 Nomor 7

```
1 onehot_encoder = OneHotEncoder(sparse=False) #Membuat variabel
  onehot_encoder dengan isi OneHotEncoder
2 integer_encoded = integer_encoded.reshape(len(integer_encoded),
  1) #Mengisi variabel integer_encoded dengan isi
  integer_encoded yang telah di convert pada fungsi sebelumnya
3 onehot_encoder.fit(integer_encoded) #Mengkonvert variabel
  integer_encoded kedalam onehot_encoder
```

Hasil:

```
[7] onehot_encoder = OneHotEncoder(sparse=False) #membuat variabel
  onehot_encoder dengan isi OneHotEncoder
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1) #mengisi
  variabel integer_encoded dengan isi integer_encoded yang telah di
  convert pada fungsi sebelumnya
onehot_encoder.fit(integer_encoded) #mengkonvert variabel
  integer_encoded kedalam onehot_encoder
[8] onehot_encoder.get_feature_names()
Out[8]: array(['cat', 'dog', 'horse', 'sheep'], dtype=object)
```

Gambar 8.20 Hasil No 7

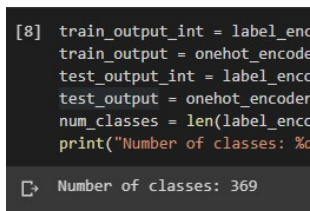
## 8.2.2.8 Nomor 8

```

1 train_output_int = label_encoder.transform(train_output) #
   Mengkonvert data train output menggunakan variabel
   label_encoder kedalam variabel train_output_int
2 train_output = onehot_encoder.transform(train_output_int.reshape(
   len(train_output_int), 1)) #Mengkonvert variabel
   train_output_int kedalam fungsi onehot_encoder
3 test_output_int = label_encoder.transform(test_output) #
   Mengkonvert data test_output menggunakan variabel
   label_encoder kedalam variabel test_output_int
4 test_output = onehot_encoder.transform(test_output_int.reshape(
   len(test_output_int), 1)) #Mengkonvert variabel
   test_output_int kedalam fungsi onehot_encoder
5 num_classes = len(label_encoder.classes_) #Membuat variabel
   num_classes dengan isi variabel label_encoder dan classess
6 print("Number of classes: %d" % num_classes) #Mencetak hasil dari
   nomer Class berupa persen

```

Hasil:



```

[8] train_output_int = label_encoder.transform(train_output)
train_output = onehot_encoder.transform(train_output_int.reshape(
test_output_int = label_encoder.transform(test_output)
test_output = onehot_encoder.transform(test_output_int.reshape(
num_classes = len(label_encoder.classes_)
print("Number of classes: %d" % num_classes)

```

Number of classes: 369

**Gambar 8.21** Hasil No 8

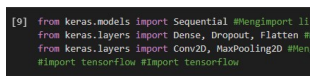
## 8.2.2.9 Nomor 9

```

1 from keras.models import Sequential #Mengimport library
   Sequential dari Keras
2 from keras.layers import Dense, Dropout, Flatten #Mengimport
   library Dense, Dropout, Flatten dari Keras
3 from keras.layers import Conv2D, MaxPooling2D #Mengimport library
   Conv2D, MaxPooling2D dari Keras

```

Hasil:



```

[9] from keras.models import Sequential #Mengimport library
   Sequential dari Keras
from keras.layers import Dense, Dropout, Flatten #Mengimport
   library Dense, Dropout, Flatten dari Keras
from keras.layers import Conv2D, MaxPooling2D #Mengimport library
   Conv2D, MaxPooling2D dari Keras

```

**Gambar 8.22** Hasil No 9

## 8.2.2.10 Nomor 10

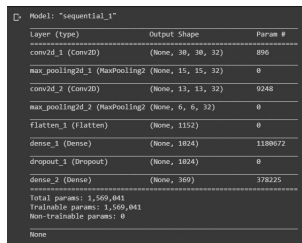


```

1 model = Sequential() #Membuat variabel model dengan isian library
  Sequential
2 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
3               input_shape=np.shape(train_input[0]))) #Variabel
  model di tambahkan library Conv2D tigapuluh dua bit dengan
  ukuran kernel 3 x 3 dan fungsi penghitungan relu dang
  menggunakan data train_input
4 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di
  tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x
  2 pixel
5 model.add(Conv2D(32, (3, 3), activation='relu')) #Variabel model
  di tambahkan dengan library Conv2D 32bit dengan kernel 3 x 3
6 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di
  tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x
  2 pixel
7 model.add(Flatten()) #Variabel model di tambahkan library Flatten
8 model.add(Dense(1024, activation='tanh')) #Variabel model di
  tambahkan library Dense dengan fungsi tanh
9 model.add(Dropout(0.5)) #Variabel model di tambahkan library
  dropout untuk memangkas data tree sebesar 50 persen
10 model.add(Dense(num_classes, activation='softmax')) #Variabel
  model di tambahkan library Dense dengan data dari num_classes
  dan fungsi softmax
11 model.compile(loss='categorical_crossentropy', optimizer='adam',
12               metrics=['accuracy']) #Mengkompile data model untuk
  mendapatkan data loss akurasi dan optimasi
13 print(model.summary()) #Mencetak variabel model kemudian
  memunculkan kesimpulan berupa data total parameter, trainable
  paremeter dan bukan trainable parameter

```

Hasil:



Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_1 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_2 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 32)	0
Flatten_1 (Flatten)	(None, 1152)	0
dense_1 (Dense)	(None, 1024)	118672
dropout_1 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 360)	378225
<b>Total params:</b>	<b>1,459,841</b>	
<b>Trainable params:</b>	<b>1,459,841</b>	
<b>Non-trainable params:</b>	<b>0</b>	

Gambar 8.23 Hasil No 10

### 8.2.2.11 Nomor 11

```

1 import keras.callbacks #Mengimport library keras dengan fungsi
  callbacks
2 tensorboard = keras.callbacks.TensorBoard(log_dir='N:/KB/hasyv2/
  logs/mnist-style') #Membuat variabel tensorboard dengan isi
  lib keras

```

Hasil:

```
[11] Import keras.callbacks module library keras dengan fungsi (callbacks,
tensorboard = keras.callbacks.TensorBoard(log_dir='./content/figs/mnist_style/
```

Gambar 8.24 Hasil No 11

## 8.2.2.12 Nomor 12

```
1 model.fit(train_input, train_output, #Fungsi model ditambahkan
    fungsi fit untuk mengetahui perhitungan dari train_input
    train_output
2     batch_size=32, #Dengan batch size 32 bit
3     epochs=10,
4     verbose=2,
5     validation_split=0.2,
6     callbacks=[tensorboard])
7
8 score = model.evaluate(test_input, test_output, verbose=2)
9 print('Test loss:', score[0])
10 print('Test accuracy:', score[1])
```

Hasil:

```
C:\train on 107668 samples, validate on 20518 samples
Epoch 1/10: 1.5342 - accuracy: 0.6288 - val_loss: 0.9789 - val_accuracy: 0.7320
Epoch 2/10: 0.9083 - accuracy: 0.7158 - val_loss: 0.8761 - val_accuracy: 0.7512
Epoch 3/10: 0.6786 - accuracy: 0.7573 - val_loss: 0.8186 - val_accuracy: 0.7651
Epoch 4/10: 0.5221 - accuracy: 0.7764 - val_loss: 0.8249 - val_accuracy: 0.7654
Epoch 5/10: 0.4118 - accuracy: 0.7796 - val_loss: 0.8478 - val_accuracy: 0.7658
Epoch 6/10: 0.3038 - accuracy: 0.7885 - val_loss: 0.8351 - val_accuracy: 0.7754
Epoch 7/10: 0.2612 - accuracy: 0.7955 - val_loss: 0.8288 - val_accuracy: 0.7751
Epoch 8/10: 0.2302 - accuracy: 0.8023 - val_loss: 0.8485 - val_accuracy: 0.7674
Epoch 9/10: 0.2081 - accuracy: 0.8071 - val_loss: 0.8511 - val_accuracy: 0.7643
Epoch 10/10: 0.1888 - accuracy: 0.8104 - val_loss: 0.8584 - val_accuracy: 0.7658
Test loss: 0.87640480777312
Test accuracy: 0.765476884577312
```

Gambar 8.25 Hasil No 12

## 8.2.2.13 Nomor 13

```
1 import time #Mengimport library time
2 results = [] #Membuat variabel result dengan array kosong
3 for conv2d_count in [1, 2]: #Melakukan looping dengan ketentuan
    konvolusi 2 dimensi 1 2
4     for dense_size in [128, 256, 512, 1024, 2048]: #Menentukan
        ukuran besaran fixcel dari data atau konvert 1 fixcel mnjadi
        data yang berada pada codigan dibawah.
5         for dropout in [0.0, 0.25, 0.50, 0.75]: #Membuat looping
            untuk memangkas masing-masing data dengan ketentuan 0 persen
            25 persen 50 persen dan 75 persen.
6             model = Sequential() #Membuat variabel model
            Sequential
7             for i in range(conv2d_count): #Membuat looping untuk
                variabel i dengan jarak dari hasil konvolusi.
8                 if i == 0: #Syarat jika i samadengan bobotnya 0
                    model.add(Conv2D(32, kernel_size=(3, 3),
9                        activation='relu', input_shape=np.shape(train_input[0]))) #
                        Menambahkan method add pada variabel model dengan konvolusi 2
                        dimensi 32 bit didalamnya dan membuat kernel dengan ukuran 3
                        x 3 dan rumus aktifasi relu dan data shape yang di hitung
                        dari data train.
```

```

10         else: #Jika tidak
11             model.add(Conv2D(32, kernel_size=(3, 3),
activation='relu')) #Menambahkan method add pada variabel
model dengan konvolusi 2 dimensi 32 bit dengan ukuran kernel
3 x3 dan fungsi aktivasi relu
12             model.add(MaxPooling2D(pool_size=(2, 2))) #
Menambahkan method add pada variabel model dengan isian
method Max pooling berdimensi 2 dengan ukuran fixcel 2 x 2.
13             model.add(Flatten()) #Merubah feature gambar menjadi
1 dimensi vektor
14             model.add(Dense(dense_size, activation='tanh')) #
Menambahkan method dense untuk pemadatan data dengan ukuran
dense di tentukan dengan rumus fungsi tanh.
15             if dropout > 0.0: #Membuat ketentuan jika pemangkasan
lebih besar dari 0 persen
16                 model.add(Dropout(dropout)) #Menambahkan method
dropout pada model dengan nilai dari dropout
17             model.add(Dense(num_classes, activation='softmax')) #
Menambahkan method dense dengan fungsi num classs dan rumus
softmax
18             model.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy']) #Mengcompile variabel
model dengan hasi loss optimasi dan akurasi matrix
19             log_dir = 'N:/KB/hasyv2/logs/conv2d_%d-dense_%d-
dropout.%.2f' % (conv2d_count, dense_size, dropout) #
Melakukan log
20             tensorboard = keras.callbacks.TensorBoard(log_dir=
log_dir) # membuat variabel tensorboard dengan isian dari
library keras dan nilai dari log_dir
21
22             start = time.time() #Membuat variabel start dengan
isian dari library time menggunakan method time
23             model.fit(train_input, train_output, batch_size=32,
epochs=10,
24                 verbose=0, validation_split=0.2, callbacks
=[tensorboard]) #Menambahkan method fit pada model dengan
data dari train input train output nilai batch nilai epoch
verbose nilai 20 persen validation split dan callback dengan
nilai tensorboard.
25             score = model.evaluate(test_input, test_output,
verbose=2) #Membuat variabel score dengan nilai evaluasi dari
model menggunakan data tes input dan tes output
26             end = time.time() #Membuat variabel end
27             elapsed = end - start #Membuat variabel elapsed
28             print("Conv2D count: %d, Dense size: %d, Dropout: %.2
f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count
, dense_size, dropout, score[0], score[1], elapsed)) #
Mencetak hasil perhitungan
29             results.append((conv2d_count, dense_size, dropout,
score[0], score[1], elapsed))

```

Hasil:

Layer	Params	Trainable	Accuracy
Conv2D (layer 1)	118	True	0.00
Conv2D (layer 2)	118	True	0.00
Conv2D (layer 3)	118	True	0.00
Conv2D (layer 4)	118	True	0.00
Conv2D (layer 5)	118	True	0.00
Conv2D (layer 6)	118	True	0.00
Conv2D (layer 7)	118	True	0.00
Conv2D (layer 8)	118	True	0.00
Conv2D (layer 9)	118	True	0.00
Conv2D (layer 10)	118	True	0.00
Conv2D (layer 11)	118	True	0.00
Conv2D (layer 12)	118	True	0.00
Conv2D (layer 13)	118	True	0.00
Conv2D (layer 14)	118	True	0.00
Conv2D (layer 15)	118	True	0.00
MaxPooling2D (layer 16)	0	False	0.00
MaxPooling2D (layer 17)	0	False	0.00
MaxPooling2D (layer 18)	0	False	0.00
MaxPooling2D (layer 19)	0	False	0.00
MaxPooling2D (layer 20)	0	False	0.00
MaxPooling2D (layer 21)	0	False	0.00
MaxPooling2D (layer 22)	0	False	0.00
MaxPooling2D (layer 23)	0	False	0.00
MaxPooling2D (layer 24)	0	False	0.00
MaxPooling2D (layer 25)	0	False	0.00
MaxPooling2D (layer 26)	0	False	0.00
MaxPooling2D (layer 27)	0	False	0.00
MaxPooling2D (layer 28)	0	False	0.00
MaxPooling2D (layer 29)	0	False	0.00
MaxPooling2D (layer 30)	0	False	0.00
MaxPooling2D (layer 31)	0	False	0.00
MaxPooling2D (layer 32)	0	False	0.00
MaxPooling2D (layer 33)	0	False	0.00
MaxPooling2D (layer 34)	0	False	0.00
MaxPooling2D (layer 35)	0	False	0.00
MaxPooling2D (layer 36)	0	False	0.00
MaxPooling2D (layer 37)	0	False	0.00
MaxPooling2D (layer 38)	0	False	0.00
MaxPooling2D (layer 39)	0	False	0.00
MaxPooling2D (layer 40)	0	False	0.00
MaxPooling2D (layer 41)	0	False	0.00
MaxPooling2D (layer 42)	0	False	0.00
MaxPooling2D (layer 43)	0	False	0.00
MaxPooling2D (layer 44)	0	False	0.00
MaxPooling2D (layer 45)	0	False	0.00
MaxPooling2D (layer 46)	0	False	0.00
MaxPooling2D (layer 47)	0	False	0.00
MaxPooling2D (layer 48)	0	False	0.00
MaxPooling2D (layer 49)	0	False	0.00
MaxPooling2D (layer 50)	0	False	0.00
MaxPooling2D (layer 51)	0	False	0.00
MaxPooling2D (layer 52)	0	False	0.00
MaxPooling2D (layer 53)	0	False	0.00
MaxPooling2D (layer 54)	0	False	0.00
MaxPooling2D (layer 55)	0	False	0.00
MaxPooling2D (layer 56)	0	False	0.00
MaxPooling2D (layer 57)	0	False	0.00
MaxPooling2D (layer 58)	0	False	0.00
MaxPooling2D (layer 59)	0	False	0.00
MaxPooling2D (layer 60)	0	False	0.00
MaxPooling2D (layer 61)	0	False	0.00
MaxPooling2D (layer 62)	0	False	0.00
MaxPooling2D (layer 63)	0	False	0.00
MaxPooling2D (layer 64)	0	False	0.00
MaxPooling2D (layer 65)	0	False	0.00
MaxPooling2D (layer 66)	0	False	0.00
MaxPooling2D (layer 67)	0	False	0.00
MaxPooling2D (layer 68)	0	False	0.00
MaxPooling2D (layer 69)	0	False	0.00
MaxPooling2D (layer 70)	0	False	0.00
MaxPooling2D (layer 71)	0	False	0.00
MaxPooling2D (layer 72)	0	False	0.00
MaxPooling2D (layer 73)	0	False	0.00
MaxPooling2D (layer 74)	0	False	0.00
MaxPooling2D (layer 75)	0	False	0.00
MaxPooling2D (layer 76)	0	False	0.00
MaxPooling2D (layer 77)	0	False	0.00
MaxPooling2D (layer 78)	0	False	0.00
MaxPooling2D (layer 79)	0	False	0.00
MaxPooling2D (layer 80)	0	False	0.00
MaxPooling2D (layer 81)	0	False	0.00
MaxPooling2D (layer 82)	0	False	0.00
MaxPooling2D (layer 83)	0	False	0.00
MaxPooling2D (layer 84)	0	False	0.00
MaxPooling2D (layer 85)	0	False	0.00
MaxPooling2D (layer 86)	0	False	0.00
MaxPooling2D (layer 87)	0	False	0.00
MaxPooling2D (layer 88)	0	False	0.00
MaxPooling2D (layer 89)	0	False	0.00
MaxPooling2D (layer 90)	0	False	0.00
MaxPooling2D (layer 91)	0	False	0.00
MaxPooling2D (layer 92)	0	False	0.00
MaxPooling2D (layer 93)	0	False	0.00
MaxPooling2D (layer 94)	0	False	0.00
MaxPooling2D (layer 95)	0	False	0.00
MaxPooling2D (layer 96)	0	False	0.00
MaxPooling2D (layer 97)	0	False	0.00
MaxPooling2D (layer 98)	0	False	0.00
MaxPooling2D (layer 99)	0	False	0.00
MaxPooling2D (layer 100)	0	False	0.00

Gambar 8.26 Hasil No 13

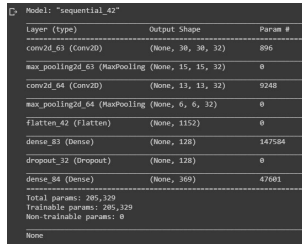
### 8.2.2.14 Nomor 14

```

1 model = Sequential() #Membuat variabel model dengan isian library
  Sequential
2 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
  input_shape=np.shape(train_input[0])))#Variabel model di
  tambahkan library Conv2D tigapuluh dua bit dengan ukuran
  kernel 3 x 3 dan fungsi penghitungan relu dang menggunakan
  data train_input
3 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di
  tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x
  2 pixel
4 model.add(Conv2D(32, (3, 3), activation='relu')) #Variabel model
  di tambahkan dengan library Conv2D 32bit dengan kernel 3 x 3
5 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di
  tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x
  2 pixel
6 model.add(Flatten()) #Variabel model di tambahkan library Flatten
7 model.add(Dense(128, activation='tanh')) #Variabel model di
  tambahkan library Dense dengan fungsi tanh
8 model.add(Dropout(0.5)) #Variabel model di tambahkan library
  dropout untuk memangkas data tree sebesar 50 persen
9 model.add(Dense(num_classes, activation='softmax')) #Variabel
  model di tambahkan library Dense dengan data dari num_classes
  dan fungsi softmax
10 model.compile(loss='categorical_crossentropy', optimizer='adam',
  metrics=['accuracy']) #Mengcompile data model untuk
  mendapatkan data loss akurasi dan optimasi
11 print(model.summary()) #Mencetak variabel model kemudian
  memunculkan kesimpulan berupa data total parameter, trainable
  paremeter dan bukan trainable parameter

```

Hasil:



Layer (type)	Output Shape	Param #
conv2d_63 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_63 (MaxPooling)	(None, 15, 15, 32)	0
conv2d_64 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_64 (MaxPooling)	(None, 6, 6, 32)	0
flatten_42 (Flatten)	(None, 1152)	0
dense_83 (Dense)	(None, 128)	147584
dropout_32 (Dropout)	(None, 128)	0
dense_84 (Dense)	(None, 360)	47601
Total params: 205,329		
Trainable params: 205,329		
Non-trainable params: 0		

Gambar 8.27 Hasil No 14

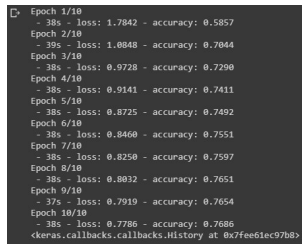
### 8.2.2.15 Nomor 15

```

1 model.fit(np.concatenate((train_input, test_input)), #Melakukan
            training yang isi datanya dari join numpy menggunakan data
            train_input test_input
2             np.concatenate((train_output, test_output)), #
            Kelanjutan data yang di gunakan pada join train_output
            test_output
3             batch_size=32, epochs=10, verbose=2) #Menggunakan
            ukuran 32 bit dan epoch 10

```

Hasil:



```

Epoch 1/10
- 30s - loss: 1.7842 - accuracy: 0.5857
Epoch 2/10
- 39s - loss: 1.0848 - accuracy: 0.7044
Epoch 3/10
- 38s - loss: 0.9728 - accuracy: 0.7298
Epoch 4/10
- 38s - loss: 0.9141 - accuracy: 0.7411
Epoch 5/10
- 38s - loss: 0.8725 - accuracy: 0.7492
Epoch 6/10
- 38s - loss: 0.8468 - accuracy: 0.7551
Epoch 7/10
- 38s - loss: 0.8258 - accuracy: 0.7597
Epoch 8/10
- 38s - loss: 0.8032 - accuracy: 0.7651
Epoch 9/10
- 37s - loss: 0.7919 - accuracy: 0.7654
Epoch 10/10
- 38s - loss: 0.7786 - accuracy: 0.7686
<keras.callbacks.callbacks.History at 0x7fee5ec97b8>

```

Gambar 8.28 Hasil No 15

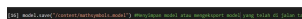
### 8.2.2.16 Nomor 16

```

1 model.save("mathsymbols.model") #Menyimpan model atau mengekspor
            model yang telah di jalan tadi

```

Hasil:



```

model.save("mathsymbols.model")

```

Gambar 8.29 Hasil No 16

### 8.2.2.17 Nomor 17

```

1 np.save('classes.npy', label_encoder.classes_) #Menyimpan label
            encoder dengan nama classes.npy

```

Hasil:

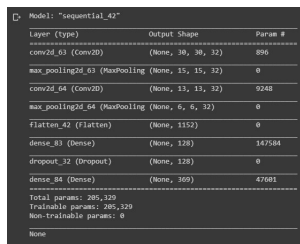


**Gambar 8.30** Hasil No 17

### 8.2.2.18 Nomor 18

```
1 import keras.models #Mengimpor library keras model
2 model2 = keras.models.load_model("mathsymbols.model") #Membuat
   variabel model2 untuk meload model yang telah di simpan tadi
3 print(model2.summary()) #Mencetak hasil model2
```

Hasil:



Layer (type)	Output Shape	Param #
conv2d_53 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_63 (MaxPooling)	(None, 15, 15, 32)	0
conv2d_54 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_64 (MaxPooling)	(None, 6, 6, 32)	0
flatten_42 (Flatten)	(None, 1152)	0
dense_81 (Dense)	(None, 128)	147584
dropout_32 (Dropout)	(None, 128)	0
dense_84 (Dense)	(None, 369)	47681
Total params: 285,329		
Trainable params: 285,329		
Non-trainable params: 0		
None		

**Gambar 8.31** Hasil No 18

### 8.2.2.19 Nomor 19

```
1 label_encoder2 = LabelEncoder() # membuat variabel label encoder
   ke 2 dengan isian fungsi label encoder.
2 label_encoder2.classes_ = np.load('classes.npy') #Menambahkan
   method classess dengan data classess yang di ekspor tadi
3 def predict(img_path): #Membuat fungsi predict dengan path img
4   newimg = keras.preprocessing.image.img_to_array(pil_image.
   open(img_path)) #Membuat variabel newimg dengan membay
   immagine menjadi array dan membuka data berdasarkan img path
5   newimg /= 255.0 #Membagi data yang terdapat pada variabel
   newimg sebanyak 255
6
7   # do the prediction
8   prediction = model2.predict(newimg.reshape(1, 32, 32, 3)) #
   Membuat variabel predivtion dengan isian variabel model2
   menggunakan fungsi predic dengan syarat variabel newimg
   dengan data reshape
9
10  # figure out which output neuron had the highest score, and
   reverse the one-hot encoding
11  inverted = label_encoder2.inverse_transform([np.argmax(
   prediction)]) #Membuat variabel inverted denagan label
   encoder2 dan menggunakan argmax untuk mencari skor keluaran
   tertinggi
```

```

12 print("Prediction: %s, confidence: %.2f" % (inverted[0], np.
    max(prediction))) #Mencetak prediksi gambar dan confidence
    dari gambar.

```

Hasil:

```

[19] label_encoder = LabelEncoder() # membuat variabel label encoder ke 1 dengan label
    label_encoder.class_ = np.load('centroid/class.npy') #memuat data ke array
    def predict(img_path): #membuat fungsi predict dengan path img
        #img = Image.open(img_path).convert('RGB')
        #img = ImageProcessing.Image(img).array[0].img_qimg(img_path) #img
        #img = cv.imread(img_path) #membuat data yang terdapat pada variabel img dengan path
        # do the prediction
        prediction = model.predict(imgs.reshape(1, 32, 32, 3)) #membuat variabel y
    # figure out which output neuron had the highest score, and reverse the output
    inverted = label_encoder.inverse_transform(np.argmax(prediction)) #membuat
    print("Prediction: %s, confidence: %.2f" % (inverted[0], np.max(prediction)))

```

Gambar 8.32 Hasil No 19

### 8.2.2.20 Nomor 20

```

1 predict("HASYv2/hasy-data/v2-00010.png") #Mencari prediksi
    menggunakan fungsi prediksi yang di buat tadi dari data di
    HASYv2/hasy-data/v2-00010.png
2 predict("HASYv2/hasy-data/v2-00500.png") #Mencari prediksi
    menggunakan fungsi prediksi yang di buat tadi dari data di
    HASYv2/hasy-data/v2-00500.png
3 predict("HASYv2/hasy-data/v2-00700.png") #Mencari prediksi
    menggunakan fungsi prediksi yang di buat tadi dari data di
    HASYv2/hasy-data/v2-00700.png

```

Hasil:

```

Prediction: A, confidence: 0.74
Prediction: \pi, confidence: 0.48
Prediction: \alpha, confidence: 0.98

```

Gambar 8.33 Hasil No 20

## 8.2.3 Penanganan Error

### 8.2.3.1 Error

- ProfilerNotRunningError

```
ProfilerNotRunningError: Cannot stop profiling. No profiler is running.
```

Gambar 8.34 ProfilerNotRunningError

### 8.2.3.2 Solusi Error

- ProfilerNotRunningError

tambahkan kode `profile=10000000` pada parameter `log_dir`

## 8.2.4 Bukti Tidak Plagiat



**Gambar 8.35** Bukti tidak plagiat

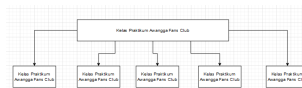
## 8.2.5 Link Youtube

[https://youtu.be/Vq\\_LZ89hTpg](https://youtu.be/Vq_LZ89hTpg)

## 8.3 1174069 - Fanny Shafira Damayanti

### 8.3.1 Teori

1. Jelaskan kenapa file teks harus di lakukan tokenizer. dilengkapi dengan ilustrasi atau gambar.



**Gambar 8.36** Ilustrasi Tokenizer

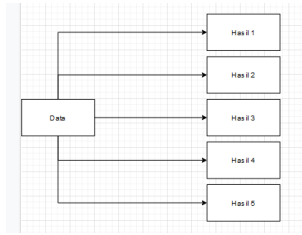
2. Jelaskan konsep dasar K Fold Cross Validation pada dataset komentar Youtube pada kode listing ??, dilengkapi dengan ilustrasi atau gambar.

```
1 kfold = StratifiedKFold(n_splits=5)
2 splits = kfold.split(d, d['CLASS'])
3
```

**Listing 8.1** K Fold Cross Validation

Pada koding diatas terdapat variabel kfold yang didalamnya berisi parameter split yang diisikan nilai 5. hal tersebut dimaksudkan untuk membuat pengolahan data akan diulang setiap datanya sebanyak lima kali dengan atribut class sebagai acuan pengolahan datanya. Lalu kemudian akan di hasilkan akurasi dari pengulangan data tersebut sebesar sekian persen tergantung datanya





**Gambar 8.37** Ilustrasi K Fold Cross Validation

3. Jelaskan apa maksudnya kode program *for train, test in splits*. dilengkapi dengan ilustrasi atau gambar.

For train digunakan untuk melakukan training atau pelatihan pada data yang sudah dideklarasikan sebelumnya. Sedangkan test in split digunakan untuk membatasi jumlah data yang akan diinputkan atau data yang akan digunakan.

```
In [3]: import numpy as np
...: from sklearn.model_selection import train_test_split
...: # randomize the data
...: p = range(5)
...: print(p)
[0 1 2 3 4]

In [4]: list(zip(
...:     x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3, random_state=0)
...:     print(x_train)
...:     print(y_train)
...:     print(x_test)
...:     print(y_test)
[0 1 2 3 4]
[0 1 2 3 4]
```

**Gambar 8.38** Ilustrasi For train dan test in split

4. Jelaskan apa maksudnya kode program *train\_content = d['CONTENT'].iloc[train\_idx]* dan *test\_content = d['CONTENT'].iloc[test\_idx]*. dilengkapi dengan ilustrasi atau gambar.

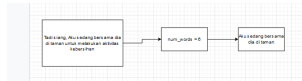
Maksud dari kode program tersebut adalah membaca isian kolom pada field yang bernama CONTENT sebagai data training dan data testing untuk program

No	Nama	Content
1	Mobil	Kendaraan darat yang biasanya memiliki roda 4
2	Motor	Kendaraan darat yang biasanya memiliki roda 2
3	Traktor	Kendaraan darat yang dipakai untuk membajak sawah
4	Helikopter	Kendaraan udara yang memiliki baling-baling untuk terbang

**Gambar 8.39** Ilustrasi penggunaan kolom Content

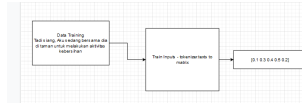
Jelaskan apa maksud dari fungsi *tokenizer = Tokenizer(num\_words=2000)* dan *tokenizer.fit\_on\_texts(train\_content)*, dilengkapi dengan ilustrasi atau gambar.

- *tokenizer = Tokenizer(num\_words=2000)* digunakan untuk membaca kalimat yang telah dibuat menjadi token sebanyak 2000 kata
- *fit\_on\_texts* digunakan untuk membuat membaca data token teks yang telah dimasukkan kedalam fungsi yaitu fungsi *train.konten*



**Gambar 8.40** Ilustrasi fit tokenizer dan num\_word=2000

5. Jelaskan apa maksud dari fungsi `d_train_inputs = tokenizer.texts_to_matrix(train_content, mode='tfidf')` dan `d_test_inputs = tokenizer.texts_to_matrix(test_content, mode='tfidf')`, dilengkapi dengan ilustrasi kode dan atau gambar. Untuk digunakan sebagai pengubah urutan teks yang tadi telah dilakukan tokenizer menjadi matriks yang berurutan seperti tf idf



**Gambar 8.41** Ilustrasi `d_train_inputs = tokenizer.texts to matrix`

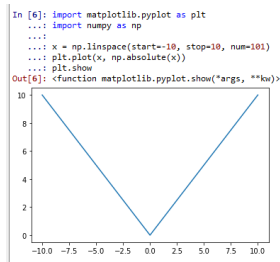
6. Jelaskan apa maksud dari fungsi `d_train_inputs = d_train_inputs/np.amax(np.absolute(d_train_inputs))` dan `d_test_inputs = d_test_inputs/np.amax(np.absolute(d_test_inputs))`, dilengkapi dengan ilustrasi atau gambar. Fungsi tersebut digunakan untuk membagi matriks tfidf dengan penentuan maksimum array sepanjang sumbu sehingga akan menimbulkan garis ke bawah dan ke atas yang membentuk gambar v. Lalu hasil tersebut akan dimasukkan ke variabel d train input dan d test input dengan metode absolute. Yang berarti tanpa bilangan negatif.
7. Jelaskan apa maksud fungsi dari `d_train_outputs = np_utils.to_categorical(d['CLASS'])` dan `d_test_outputs = np_utils.to_categorical(d['CLASS'].iloc[test_idx])` dalam kode program, dilengkapi dengan ilustrasi atau gambar. Maksud dari fungsi tersebut yaitu untuk merubah nilai vektor yang ada pada atribut class menjadi bentuk matrix dengan pengurutan berdasarkan data index training dan testing.
8. Jelaskan apa maksud dari fungsi di listing ???. Gambarkan ilustrasi Neural Network nya dari model kode tersebut.

```

1  model = Sequential()
2  model.add(Dense(512, input_shape=(2000,)))
3  model.add(Activation('relu'))
4  model.add(Dropout(0.5))
5  model.add(Dense(2))
6  model.add(Activation('softmax'))
7
  
```

**Listing 8.2** Membuat model Neural Network

model = sequential berarti variabel model berisi method sequential yang berguna untuk searching data dengan menerima parameter atau argumen kunci dengan langkah tertentu untuk mencari data yang telah diolah. Kemudian model akan ditambahkan method add dengan dense yang berarti data - data yang diinputkan akan terhubung, dengan data 612 dan 2000 data kata atau word kemudian model tersebut di masukan fungsi activation dengan rumus atau metode relu. setelah itu data akan di dropout 0.5 atau dipangkas sebanyak 50 persen dikarenakan pada pohon bobot terlalu akurat terhadap data.



**Gambar 8.42** Ilustrasi d train inputs

```

In [10]: labels = [0,2,1,2,0,1]
...: keras.utils.to_categorical(labels)Out[11]:
array([[1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.]], dtype=float32)

```

**Gambar 8.43** Ilustrasi train outputs = np utils.to categorical

9. Jelaskan apa maksud dari fungsi di listing ?? dengan parameter tersebut.

```

1 model.compile(loss='categorical_crossentropy', optimizer=
2 'adamax',
3               metrics=['accuracy'])

```

**Listing 8.3** Compile model

model tersebut kemudian di compile atau di kembalikan kembali fungsi nilainya yangmana akan mengembalikan fungsi nilai loss nya berapa yang diambil dari fungsi adamax yang berguna untuk mengetahui nilai loss-nya kemudian metrics = acuracy merupakan akurasi dari nilai matrixnya. kemudian terdiri atas beberapa layer atau hidden layer. perbedaan antara deep learning dan DNN atau Deep Neural Network yaitu deep lerning merupakan pemakai algoritma dari DNN dan DNN merupakan algoritma yang ada pada deep learning.



### 8.3.2 Praktek

1. Jelaskan kode program pada blok # In[1]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[1]:import lib
2 # menimport libtari CSV untuk mengolah data ber ekstensi csv
3 import csv
4 #kemudian Melakukan import library Image yang berguna untuk
   dari PIL atau Python Imaging Library yang berguna untuk
   mengolah data berupa gambar
5 from PIL import Image as pil_image
6 # kemudian Melakukan import library keras yang menggunakan
   method preprocessing yang digunakan untuk membuat neural
   network
7 import keras.preprocessing.image

```

2. Jelaskan kode program pada blok # In[2]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[2]:load all images (as numpy arrays) and save their
   classes
2 #Menginisiasi variabel imgs dan classes dengan variabel array
   kosong
3 imgs = []
4 classes = []
5 #membuaka file hasy-data-labels.csv yang berada di foleda
   HASYv2 yang di inisialisasi menjadi csvfile
6 with open('F://Semester 6/Artificial Intelligence/Tugas 7/New
   Folder/hasy-data-labels.csv') as csvfile:
7     #Menginisiasi variabel csvreader yang berisi method csv.
   reader yang membaca variabel csvfile
8     csvreader = csv.reader(csvfile)
9     # Menginisiasi variabel i dengan isi 0
10    i = 0
11    # membuat looping pada variabel csvreader
12    for row in csvreader:
13        # dengan ketentuan jika i lebihkecil daripada o
14        if i > 0:
15            # dibuat variabel img dengan isi keras untuk
   aktivasi neural network fungsi yang membaca data yang
   berada dalam folder HASYv2 dengan input nilai -1.0 dan 1.0
16            img = keras.preprocessing.image.img_to_array(
   pil_image.open("F://Semester 6/Artificial Intelligence/
   Tugas 7/New Folder/" + row[0]))
17            #Pembagian data yang ada pada fungsi img sebanyak
   255.0
18            img /= 255.0
19            # Penambahan nilai baru pada imgs pada row ke 1 2
   dan dilanjutkan dengan variabel img
20            imgs.append((row[0], row[2], img))
21            # Penambahan nilai pada row ke 2 pada variabel
   classes
22            classes.append(row[2])

```

```

23         # penambahan nilai satu pada variabel i
24         i += 1

```

3. Jelaskan kode program pada blok # In[3]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[3]:shuffle the data, split into 80% train, 20% test
2 # Melakukan import library random
3 import random
4 # melakukan random pada vungsi imgs
5 random.shuffle(imgs)
6 # Menginisiasi variabel split_idx dengan nilai integer 80
   persen dikali dari pengembalian jumlah dari variabel imgs
7 split_idx = int(0.8*len(imgs))
8 # Menginisiasi variabel train dengan isi lebih besar split
   idx
9 train = imgs[:split_idx]
10 # Menginisiasi variabel test dengan isi lebih kecil split idx
11 test = imgs[split_idx:]

```

4. Jelaskan kode program pada blok # In[4]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[4]:
2 # Melakukan import library numpy dengan inisial np
3 import numpy as np
4 # Menginisiasi variabel train input dengan np method asarray
   yang mana membuat array dengan isi row 2 dari data train
5 train_input = np.asarray(list(map(lambda row: row[2], train))
   )
6 # membuat test input input dengan np method asarray yang mana
   membuat array dengan isi row 2 dari data test
7 test_input = np.asarray(list(map(lambda row: row[2], test)))
8 # Menginisiasi variabel train_output dengan np method asarray
   yang mana membuat array dengan isi row 1 dari data train
9 train_output = np.asarray(list(map(lambda row: row[1], train))
   )
10 # Menginisiasi variabel test_output dengan np method asarray
   yang mana membuat array dengan isi row 1 dari data test
11 test_output = np.asarray(list(map(lambda row: row[1], test)))

```

5. Jelaskan kode program pada blok # In[5]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[5]: import encoder and one hot
2 # Melakukan import library LabelEncode dari sklearn
3 from sklearn.preprocessing import LabelEncoder
4 # Melakukan import library OneHotEncoder dari sklearn
5 from sklearn.preprocessing import OneHotEncoder

```

6. Jelaskan kode program pada blok # In[6]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[6]:convert class names into one-hot encoding
2
3 # Menginisiasi variabel label_encoder dengan isi LabelEncoder
4 label_encoder = LabelEncoder()
5 # Menginisiasi variabel integer_encoded yang berfungsi untuk
  Menconvert variabel classes kedalam bentuk integer
6 integer_encoded = label_encoder.fit_transform(classes)

```

7. Jelaskan kode program pada blok # In[7]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[7]:then convert integers into one-hot encoding
2 # Menginisiasi variabel onehot_encoder dengan isi
  OneHotEncoder
3 onehot_encoder = OneHotEncoder(sparse=False)
4 # mengisi variabel integer_encoded dengan isi integer_encoded
  yang telah di convert pada fungsi sebelumnya
5 integer_encoded = integer_encoded.reshape(len(integer_encoded)
  ), 1)
6 # Menconvert variabel integer_encoded kedalam onehot_encoder
7 onehot_encoder.fit(integer_encoded)

```

8. Jelaskan kode program pada blok # In[8]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[8]:convert train and test output to one-hot
2 # Menconvert data train output menggunakan variabel
  label_encoder kedalam variabel train_output_int
3 train_output_int = label_encoder.transform(train_output)
4 # Menconvert variabel train_output_int kedalam fungsi
  onehot_encoder
5 train_output = onehot_encoder.transform(train_output_int.
  reshape(len(train_output_int), 1))
6 # Menconvert data test_output menggunakan variabel
  label_encoder kedalam variabel test_output_int
7 test_output_int = label_encoder.transform(test_output)
8 # Menconvert variabel test_output_int kedalam fungsi
  onehot_encoder
9 test_output = onehot_encoder.transform(test_output_int.
  reshape(len(test_output_int), 1))
10 # Menginisiasi variabel num_classes dengan isi variabel
  label_encoder dan classess
11 num_classes = len(label_encoder.classes_)
12 # mencetak hasil dari nomer Class berupa persen
13 print("Number of classes: %d" % num_classes)

```

9. Jelaskan kode program pada blok # In[9]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[9]: import sequential
2 # Melakukan import library Sequential dari Keras
3 from keras.models import Sequential
4 # Melakukan import library Dense, Dropout, Flatten dari Keras
5 from keras.layers import Dense, Dropout, Flatten
6 # Melakukan import library Conv2D, MaxPooling2D dari Keras
7 from keras.layers import Conv2D, MaxPooling2D

```

10. Jelaskan kode program pada blok # In[10]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[10]: desain jaringan
2 # Menginisiasi variabel model dengan isian library Sequential
3 model = Sequential()
4 # variabel model di tambahkan library Conv2D tigapuluh dua
  bit dengan ukuran kernel 3 x 3 dan fungsi penghitungan
  relu dang menggunakan data train_input
5 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
6                  input_shape=np.shape(train_input[0])))
7 # variabel model di tambahkan dengan lib MaxPooling2D dengan
  ketentuan ukuran 2 x 2 pixel
8 model.add(MaxPooling2D(pool_size=(2, 2)))
9 # variabel model di tambahkan dengan library Conv2D 32bit
  dengan kernel 3 x 3
10 model.add(Conv2D(32, (3, 3), activation='relu'))
11 # variabel model di tambahkan dengan lib MaxPooling2D dengan
  ketentuan ukuran 2 x 2 pixel
12 model.add(MaxPooling2D(pool_size=(2, 2)))
13 # variabel model di tambahkan library Flatten
14 model.add(Flatten())
15 # variabel model di tambahkan library Dense dengan fungsi
  tanh
16 model.add(Dense(1024, activation='tanh'))
17 # variabel model di tambahkan library dropout untuk memangkas
  data tree sebesar 50 persen
18 model.add(Dropout(0.5))
19 # variabel model di tambahkan library Dense dengan data dari
  num_classes dan fungsi softmax
20 model.add(Dense(num_classes, activation='softmax'))
21 # mengkompile data model untuk mendapatkan data loss akurasi
  dan optimasi
22 model.compile(loss='categorical_crossentropy', optimizer='
  adam',
23               metrics=['accuracy'])
24 # mencetak variabel model kemudian memunculkan kesimpulan
  berupa data total parameter, trainable parameter dan bukan
  trainable parameter
25 print(model.summary())

```

11. Jelaskan kode program pada blok # In[11]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[11]: import sequential

```



```

2 # Melakukan import library keras callbacks
3 import keras.callbacks
4 # Menginisiasi variabel tensorboard dengan isi lib keras
5 tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/
    mnist-style')

```

12. Jelaskan kode program pada blok # In[12]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[12]: 5menit kali 10 epoch = 50 menit
2 # fungsi model titambahkan metod fit untuk mengetahui
    perhitungan dari train_input train_output
3 model.fit(train_input, train_output,
4 # dengan batch size 32 bit
5         batch_size=32,
6         epochs=10,
7         verbose=2,
8         validation_split=0.2,
9         callbacks=[tensorboard])
10
11 score = model.evaluate(test_input, test_output, verbose=2)
12 print('Test loss:', score[0])
13 print('Test accuracy:', score[1])

```

13. Jelaskan kode program pada blok # In[13]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[13]:try various model configurations and parameters to
    find the best
2 # Melakukan import library time
3 import time
4 #Menginisiasi variabel result dengan array kosong
5 results = []
6 # melakukan looping dengan ketentuan konvolusi 2 dimensi 1 2
7 for conv2d_count in [1, 2]:
8     # menentukan ukuran besaran fixcel dari data atau konvert
        1 fixcel mnjadi data yang berada pada codigan dibawah.
9     for dense_size in [128, 256, 512, 1024, 2048]:
10         # membuat looping untuk memangkas masing-masing data
            dengan ketentuan 0 persen 25 persen 50 persen dan 75
            persen.
11         for dropout in [0.0, 0.25, 0.50, 0.75]:
12             # Menginisiasi variabel model Sequential
13             model = Sequential()
14             #membuat looping untuk variabel i dengan jarak
                dari hasil konvolusi.
15             for i in range(conv2d_count):
16                 # syarat jika i samadengan bobotnya 0
17                 if i == 0:
18                     # Penambahan method add pada variabel
                        model dengan konvolusi 2 dimensi 32 bit didalamnya dan
                        membuat kernel dengan ukuran 3 x 3 dan rumus aktivasi relu
                        dan data shape yang di hitung dari data train.

```

```

19         model.add(Conv2D(32, kernel_size=(3, 3),
20             activation='relu', input_shape=np.shape(train_input[0])))
21         # jika tidak
22         else:
23             # Penambahan method add pada variabel
24             model dengan konvolusi 2 dimensi 32 bit dengan ukuran
25             kernel 3 x3 dan fungsi aktivasi relu
26             model.add(Conv2D(32, kernel_size=(3, 3),
27                 activation='relu'))
28             # Penambahan method add pada variabel model
29             dengan isian method Max pooling berdimensi 2 dengan
30             ukuran fixcel 2 x 2.
31             model.add(MaxPooling2D(pool_size=(2, 2)))
32             # merubah feature gambar menjadi 1 dimensi vektor
33             model.add(Flatten())
34             # Penambahan method dense untuk pemadatan data
35             dengan ukuran dense di tentukan dengan rumus fungsi tanh.
36             model.add(Dense(dense_size, activation='tanh'))
37             # membuat ketentuan jika pemangkasan lebih besar
38             dari 0 persen
39             if dropout > 0.0:
40                 # Penambahan method dropout pada model dengan
41                 nilai dari dropout
42                 model.add(Dropout(dropout))
43                 # Penambahan method dense dengan fungsi num
44                 classs dan rumus softmax
45                 model.add(Dense(num_classes, activation='softmax'
46                     ))
47                 # mongkompile variabel model dengan hasil loss
48                 optimasi dan akurasi matrix
49                 model.compile(loss='categorical_crossentropy',
50                     optimizer='adam', metrics=['accuracy'])
51                 # melakukan log pada dir
52                 log_dir = './logs/conv2d_%d-dense_%d-dropout_%2f'
53                 '% (conv2d_count, dense_size, dropout)
54                 # Menginisiasi variabel tensorboard dengan isian
55                 dari library keras dan nilai dari log_dir
56                 tensorboard = keras.callbacks.TensorBoard(log_dir
57                     =log_dir)
58                 # Menginisiasi variabel start dengan isian dari
59                 library time menggunakan method time
60
61                 start = time.time()
62                 # Penambahan method fit pada model dengan data
63                 dari train input train output nilai batch nilai epoch
64                 verbose nilai 20 persen validation split dan callback
65                 dengan nilai tnsorboard.
66                 model.fit(train_input, train_output, batch_size
67                     =32, epochs=10,
68                     verbose=0, validation_split=0.2,
69                     callbacks=[tensorboard])
70                 # Menginisiasi variabel score dengan nilai
71                 evaluasi dari model menggunakan data tes input dan tes
72                 output
73                 score = model.evaluate(test_input, test_output,
74                     verbose=2)

```

```

50         # Menginisiasi variabel end
51         end = time.time()
52         # Menginisiasi variabel elapsed
53         elapsed = end - start
54         # mencetak hasil perhitungan
55         print("Conv2D count: %d, Dense size: %d, Dropout:
           %.2f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (
           conv2d_count, dense_size, dropout, score[0], score[1],
           elapsed))
56         results.append((conv2d_count, dense_size, dropout
           , score[0], score[1], elapsed))

```

14. Jelaskan kode program pada blok # In[14]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarnya dari komputer sendiri.

```

1  # In[14]:rebuild/retrain a model with the best parameters (
           from the search) and use all data
2  # Menginisiasi variabel model dengan isian library Sequential
3  model = Sequential()
4  # variabel model di tambahkan library Conv2D tigapuluh dua
           bit dengan ukuran kernel 3 x 3 dan fungsi penghitungan
           relu dang menggunakan data train_input
5  model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
           input_shape=np.shape(train_input[0])))
6  # variabel model di tambahkan dengan lib MaxPooling2D dengan
           ketentuan ukuran 2 x 2 pixel
7  model.add(MaxPooling2D(pool_size=(2, 2)))
8  # variabel model di tambahkan dengan library Conv2D 32bit
           dengan kernel 3 x 3
9  model.add(Conv2D(32, (3, 3), activation='relu'))
10 # variabel model di tambahkan dengan lib MaxPooling2D dengan
           ketentuan ukuran 2 x 2 pixel
11 model.add(MaxPooling2D(pool_size=(2, 2)))
12 # variabel model di tambahkan library Flatten
13 model.add(Flatten())
14 # variabel model di tambahkan library Dense dengan fungsi
           tanh
15 model.add(Dense(128, activation='tanh'))
16 # variabel model di tambahkan library dropout untuk memangkas
           data tree sebesar 50 persen
17 model.add(Dropout(0.5))
18 # variabel model di tambahkan library Dense dengan data dari
           num_classes dan fungsi softmax
19 model.add(Dense(num_classes, activation='softmax'))
20 # mengcompile data model untuk mendapatkan data loss akurasi
           dan optimasi
21 model.compile(loss='categorical_crossentropy', optimizer='
           adam', metrics=['accuracy'])
22 # mencetak variabel model kemudian memunculkan kesimpulan
           berupa data total parameter, trainable parameter dan bukan
           trainable parameter
23 print(model.summary())

```

15. Jelaskan kode program pada blok # In[15]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[15]:join train and test data so we train the network on
   all data we have available to us
2 # melakukan join numpy menggunakan data train_input
   test_input
3 model.fit(np.concatenate((train_input , test_input)),
4           # kelanjutan data yang di gunakan pada join
   train_output test_output
5           np.concatenate((train_output , test_output)),
6           #menggunakan ukuran 32 bit dan epoch 10
7           batch_size=32, epochs=10, verbose=2)

```

16. Jelaskan kode program pada blok # In[16]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[16]:save the trained model
2 #menyimpan model atau mengekspor model yang telah di
   jalantadi
3 model.save("mathsymbols.model")

```

17. Jelaskan kode program pada blok # In[17]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[17]:save label encoder (to reverse one-hot encoding)
2 # menyompan label encoder dengan nama classes.npy
3 np.save('classes.npy', label_encoder.classes_)

```

18. Jelaskan kode program pada blok # In[18]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[18]:load the pre-trained model and predict the math
   symbol for an arbitrary image;
2 # the code below could be placed in a separate file
3 # mengimpor library keras model
4 import keras.models
5 # Menginisiasi variabel model2 untuk meload model yang telah
   di simpan tadi
6 model2 = keras.models.load_model("mathsymbols.model")
7 # mencetak hasil model2
8 print(model2.summary())

```

19. Jelaskan kode program pada blok # In[19]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[19]:restore the class name to integer encoder
2 # Menginisiasi variabel label encoder ke 2 dengan isian
   fungsi label encoder.
3 label_encoder2 = LabelEncoder()
4 # Penambahan method classess dengan data classess yang di
   ekspor tadi
5 label_encoder2.classes_ = np.load('classes.npy')
6 # membuat fungsi predict dengan path img
7 def predict(img_path):
8     # Menginisiasi variabel newimg dengan membuay image
   menjadi array dan membuka data berdasarkan img path
9     newimg = keras.preprocessing.image.img_to_array(pil.image
   .open(img_path))
10    # membagi data yang terdapat pada variabel newimg
   sebanyak 255
11    newimg /= 255.0
12
13    # do the prediction
14    # Menginisiasi variabel predivtion dengan isian variabel
   model2 menggunakan fungsi predic dengan syarat variabel
   newimg dengan data reshape
15    prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
16
17    # figure out which output neuron had the highest score ,
   and reverse the one-hot encoding
18    # Menginisiasi variabel inverted denagan label encoder2
   dan menggunakan argmax untuk mencari skor luaran
   tertinggi
19    inverted = label_encoder2.inverse_transform([np.argmax(
   prediction)])
20    # mencetak prediksi gambar dan confidence dari gambar.
21    print("Prediction: %s, confidence: %.2f" % (inverted[0],
   np.max(prediction)))

```

20. Jelaskan kode program pada blok # In[20]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[20]: grab an image (we'll just use a random training
   image for demonstration purposes)
2 # mencari prediksi menggunakan fungsi prediksi yang di buat
   tadi dari data di HASYv2/hasy-data/v2-00010.png
3 predict("F://Semester 6/Artificial Intelligence/Tugas 7/New
   Folder/hasy-data/v2-00010.png")
4 # mencari prediksi menggunakan fungsi prediksi yang di buat
   tadi dari data di HASYv2/hasy-data/v2-00500.png
5 predict("F://Semester 6/Artificial Intelligence/Tugas 7/New
   Folder/hasy-data/v2-00500.png")
6 # mencari prediksi menggunakan fungsi prediksi yang di buat
   tadi dari data di HASYv2/hasy-data/v2-00700.png
7 predict("F://Semester 6/Artificial Intelligence/Tugas 7/New
   Folder/hasy-data/v2-00700.png")

```

8.3.3 Penanganan Error

1. SS Error

```
File "D:\kuliah\semester 8\kecerdasan buatan\bahan\src\1174027\8\1174027.py", line 9,
in <module>
  import librosa
librosaNotFoundError: No module named 'librosa'
```

Gambar 8.46 No Module Name error

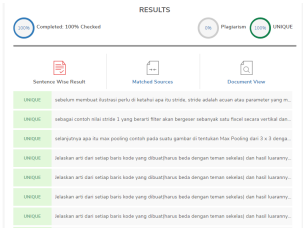
2. Jenis Error

- No Module

3. Cara Penanganan

Dengan cara melakukan instalasi module yang bersangkutan / menginstal library yang digunakan

8.3.4 Bukti Tidak Plagiat



Gambar 8.47 Tidak Melakukan Plagiat Pada Ch 7