

**TUGAS AKHIR - KS184822**

# **KLASIFIKASI LABEL GAMBAR SOAL MATEMATIKA MENGUNAKAN CONVOLUTIONAL NEURAL NETWORK**

**FADHLI AZHAR**

**NRP 06211640000049**

**Dosen Pembimbing**

**Dr. Dra. Kartika Fithriasari M.Si**

**NIP 196912121993032002**

**Program Studi Sarjana**

**Departemen Statistika**

**Fakultas Sains dan Analitika Data**

**Institut Teknologi Sepuluh Nopember**

**Surabaya**

**2023**



**TUGAS AKHIR - KS184822**

# **KLASIFIKASI LABEL GAMBAR SOAL MATEMATIKA DENGAN CONVOLUTIONAL NEURAL NETWORK**

**FADHLI AZHAR**

**NRP 06211640000049**

Dosen Pembimbing

**Dr. Dra. Kartika Fithriasari M.Si**

**NIP 196912121993032002**

**Program Studi Sarjana**

Departemen Statistika

Fakultas Sains dan Analitika Data

Institut Teknologi Sepuluh Nopember

Surabaya

2023



**FINAL PROJECT - KS184822**

# **CLASSIFICATION OF MATHEMATICS IMAGE PROBLEM LABELS USING CONVOLUTIONAL NEURAL NETWORK**

**FADHLI AZHAR**

**NRP 06211640000049**

**Advisor**

**Dr. Dra. Kartika Fithriasari M.Si**

**NIP 196912121993032002**

**Undergraduate Program of Statistics**

Department of Statistics

Faculty of Science and Data Analytics

Institut Teknologi Sepuluh Nopember

Surabaya

2023

## LEMBAR PENGESAHAN

### KLASIFIKASI LABEL GAMBAR SOAL MATEMATIKA MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK

#### TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat  
memperoleh gelar Sarjana pada  
Program Studi S-1 Sarjana  
Departemen Statistika  
Fakultas Sains dan Analitika Data  
Institut Teknologi Sepuluh Nopember

Oleh : Fadhli Azhar

NRP. 06211640000049

Tanggal Ujian : Februari 2023

Periode Wisuda : September 2023

Disetujui oleh :

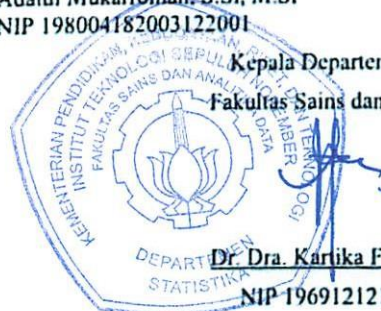
Pembimbing :

1. Dr. Dra. Kartika Fithriasari M.Si  
NIP 196912121993092002

Penguji :

1. Dr. Santi Wulan Purnami, S.Si, M.Si  
NIP 197209231998032001
2. Adatul Mukarromah, S.Si, M.Si  
NIP 198004182003122001

Kepala Departemen Statistika  
Fakultas Sains dan Analitika Data



Dr. Dra. Kartika Fithriasari, M.Si  
NIP 196912121993092002

.....  
.....  
.....

*Halaman ini sengaja dikosongkan*

## APPROVAL SHEET

### CLASSIFICATION OF MATHEMATICS IMAGE PROBLEM LABELS USING CONVOLUTIONAL NEURAL NETWORK

#### FINAL PROJECT

Submitted to fulfill one of requirements for  
Obtaining a degree Bachelor of Statistics  
Undergraduate Study Program S-1 Statistics  
Department of Statistics  
Faculty of Science and Data Analytics  
Institut Teknologi Sepuluh Nopember  
By : **Fadhli Azhar**  
NRP. 06211640000049  
Exam Date : Februari 2023  
Graduation Period : September 2023

Approved by :

Advisor :

2. Dr. Dra. Kartika Fithriasari M.Si  
NIP 196912121993092002

Examiners :

3. Dr. Santi Wulan Purnami, S.Si, M.Si  
NIP 197209231998032001

4. Adatul Mukarromah, S.Si, M.Si  
NIP 198004182003122001

Kepala Departemen Statistika  
Fakultas Sains dan Analitika Data



**Dr. Dra. Kartika Fithriasari, M.Si**  
NIP 196912121993092002

*Halaman ini sengaja dikosongkan*

## PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

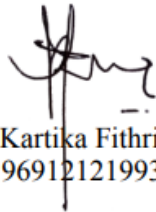
Nama mahasiswa / NRP : Fadhli Azhar / 06211640000049  
Program studi : Sarjana  
Dosen Pembimbing / NIP : Dr. Kartika Fithriasari M.Si / 196912121993032002

dengan ini menyatakan bahwa Tugas Akhir dengan judul “Klasifikasi Label Gambar Soal Matematika Menggunakan *Convolutional Neural Network*” adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 4 Januari 2023

Mengetahui  
Dosen Pembimbing



Dr. Dra. Kartika Fithriasari M.Si  
NIP. 196912121993032002

Mahasiswa



Fadhli Azhar  
NRP. 06211640000049



*Halaman ini sengaja dikosongkan*

## ABSTRAK

### Klasifikasi Label Gambar Soal Matematika Menggunakan Convolutional Neural Network

**Nama Mahasiswa / NRP** : Fadhli Azhar / 06211640000049  
**Departemen** : Statistika FSAD - ITS  
**Dosen Pembimbing** : Dr. Dra. Kartika Fithriasari M.Si

#### Abstrak

Pemahaman *learner-centered* merupakan pemahaman yang mendorong pelajar untuk berkontribusi secara aktif pada proses pembelajaran. Proses belajar aktif terdistribusi merupakan salah satu Teknik yang dapat diterapkan pada pemahaman *learner-centered*. Infrastruktur dalam menerapkan proses belajar aktif terdistribusi dibutuhkan agar memudahkan pelajar dalam melaksanakan proses belajar aktif terdistribusi seperti sistem rekomendasi. Sistem rekomendasi yang dapat dibuat untuk membantu pengalaman belajar pelajar lebih menarik antara lain pengenalan soal, rekomendasi materi belajar dan sebagainya. Sistem rekomendasi seperti pengenalan soal dapat dibuat dengan bantuan model deep learning seperti *Convolutional Neural Network*. Model *convolutional neural network* mampu mempelajari pola yang ada pada label soal matematika. Model terbaik yang didapatkan pada hasil penerapan *convolutional neural network* adalah model *convolutional neural network* dengan 2 *convolutional kernel* dan 3 *convolutional layer*. Model tersebut mampu mendapatkan nilai kebaikan klasifikasi sebesar 0,9453 pada data *training* dan 0,9375 pada data *testing*.

**Kata kunci:** *Convolutional Neural Network, Convolutional Kernel, Convolutional Layer, Learner-Centered.*

*Halaman ini sengaja dikosongkan*

## ABSTRACT

### Classification of Mathematics Image Problem Labels Using Convolutional Neural Network

**Student Name / NRP** : Fadhli Azhar / 06211640000049  
**Department** : Statistika FSAD - ITS  
**Advisor** : Dr. Dra. Kartika Fithriasari M.Si

#### Abstract

Learner-centered understanding is an understanding that encourages students to actively contribute to the learning process. Distributed active learning is a technique that can be applied to learner-centered understanding. Infrastructure in implementing a distributed active learning process is needed to make it easier for students to carry out a distributed active learning process such as a recommendation system. Recommendation systems that can be made to help students' learning experiences be more interesting include question identifiers, study material recommendations and so on. Recommendation systems such as question identifiers can be made with the help of deep learning models such as Convolutional Neural Networks. The convolutional neural network model is able to study patterns in mathematics problem labels. The best model obtained from the results of implementing a convolutional neural network is a convolutional neural network model with 2 convolutional kernels and 3 convolutional layers. The model is able to get a good classification value of 0.9453 on the training data and 0.9375 on the testing data.

**Keywords:** *Convolutional Neural Network, Distributed Active Learning, Learner-Centered.*

*Halaman ini sengaja dikosongkan*

## DAFTAR ISI

LEMBAR PENGESAHAN	Error! Bookmark not defined.
PERNYATAAN ORISINALITAS	ii
ABSTRAK	vi
ABSTRACT	ix
DAFTAR ISI	xi
DAFTAR GAMBAR	xii
DAFTAR TABEL	xv
DAFTAR SIMBOL	xvii
BAB 1 PENDAHULUAN	19
1.1 Latar Belakang	11
1.2 Rumusan Masalah	21
1.3 Tujuan	21
1.4 Batasan Masalah	13
1.5 Manfaat	21
BAB 2 TINJAUAN PUSTAKA	15
2.1 Convolutional Neural Network	15
2.2 Evaluasi Hasil Klasifikasi	18
BAB 3 METODOLOGI	27
3.1 Sumber Data	20
3.2 Variabel Penelitian	20
3.3 Langkah-Langkah Penelitiann	21
BAB 4 Hasil dan Pembahasan	29
4.1 Preprocessing Data Soal Foto	22
4.2 Deskripsi Data Hasil Preprocessing	22
4.3 Klasifikasi Gambar Soal Menggunakan CNN	23
4.4 Perbandingan Performa Klasifikasi Setiap Convolutional Kernel	25
BAB 5 Kesimpulan dan Saran	37
5.1 Kesimpulan	37
5.2 Saran	37
DAFTAR PUSTAKA	38
LAMPIRAN	39
BIODATA PENULIS	41
	xi

*Halaman ini sengaja dikosongkan*

## DAFTAR GAMBAR

<b>Gambar 2.1</b> Arsitektur <i>Convolutional Neural Network</i>	15
<b>Gambar 2.2</b> Ilustrasi operasi convolutional layer	16
<b>Gambar 2.3</b> ReLU Layer	16
<b>Gambar 2.4</b> Average pooling dan max pooling	17
<b>Gambar 3.1</b> Contoh soal integral	19
<b>Gambar 4.1 (a)</b> Foto soal sebelum preprocessing	21
<b>Gambar 4.1 (b)</b> Foto soal setelah preprocessing	21
<b>Gambar 4.2</b> Contoh soal integral	22
<b>Gambar 4.3</b> Contoh soal peluang	22



*Halaman ini sengaja dikosongkan*

## DAFTAR TABEL

<b>Tabel 3.1</b> Variabel penelitian yang digunakan.....	19
<b>Tabel 3.2</b> Struktur data.....	19
<b>Tabel 4.1</b> Hasil kebaikan klasifikasi pada data training dan testing menggunakan 2 convolutional kernel.....	22
<b>Tabel 4.2</b> Hasil kebaikan klasifikasi pada data training dan testing menggunakan 3 convolutional kernel.....	23
<b>Tabel 4.3</b> Hasil kebaikan klasifikasi pada data training dan testing menggunakan 4 convolutional kernel.....	23
<b>Tabel 4.4</b> Hasil kebaikan klasifikasi pada data training dan testing menggunakan 5 convolutional kernel.....	23
<b>Tabel 4.5</b> Hasil kebaikan klasifikasi pada data training dan testing menggunakan 6 convolutional kernel.....	24
<b>Tabel 4.6</b> Hasil kebaikan klasifikasi pada data training dan testing menggunakan 7 convolutional kernel.....	24
<b>Tabel 4.7</b> Perbandingan performansi model CNN.....	24

*Halaman ini sengaja dikosongkan*

## DAFTAR LAMPIRAN

<b>Lampiran 1.</b> Syntax <i>Pre-processing</i> .....	28
<b>Lampiran 2.</b> Syntax 2 Convolutional Kernel 1 Convolutional Layer.....	30
<b>Lampiran 3.</b> Hasil 2 Convolutional Kernel 1 Convolutional Layer.....	31
<b>Lampiran 4.</b> Syntax 2 Convolutional Kernel 2 Convolutional Layer.....	32
<b>Lampiran 5.</b> Hasil 2 Convolutional Kernel 2 Convolutional Layer.....	33
<b>Lampiran 6.</b> Syntax 2 Convolutional Kernel 3 Convolutional Layer.....	34
<b>Lampiran 7.</b> Hasil 2 Convolutional Kernel 3 Convolutional Layer.....	35
<b>Lampiran 8.</b> Syntax 3 Convolutional Kernel 1 Convolutional Layer.....	36
<b>Lampiran 9.</b> Hasil 3 Convolutional Kernel 1 Convolutional Layer.....	36
<b>Lampiran 10.</b> Syntax 3 Convolutional Kernel 2 Convolutional Layer.....	38
<b>Lampiran 11.</b> Hasil 3 Convolutional Kernel 2 Convolutional Layer.....	38
<b>Lampiran 12.</b> Syntax 3 Convolutional Kernel 3 Convolutional Layer.....	39
<b>Lampiran 13.</b> Hasil 3 Convolutional Kernel 3 Convolutional Layer.....	40
<b>Lampiran 14.</b> Syntax 4 Convolutional Kernel 1 Convolutional Layer.....	41
<b>Lampiran 15.</b> Hasil 4 Convolutional Kernel 1 Convolutional Layer.....	41
<b>Lampiran 16.</b> Syntax 4 Convolutional Kernel 2 Convolutional Layer.....	43
<b>Lampiran 17.</b> Hasil 4 Convolutional Kernel 2 Convolutional Layer.....	43
<b>Lampiran 18.</b> Syntax 4 Convolutional Kernel 3 Convolutional Layer.....	45
<b>Lampiran 19.</b> Hasil 4 Convolutional Kernel 3 Convolutional Layer.....	45
<b>Lampiran 20.</b> Syntax 5 Convolutional Kernel 1 Convolutional Layer.....	47
<b>Lampiran 21.</b> Hasil 5 Convolutional Kernel 1 Convolutional Layer.....	47
<b>Lampiran 22.</b> Syntax 5 Convolutional Kernel 2 Convolutional Layer.....	48
<b>Lampiran 23.</b> Hasil 5 Convolutional Kernel 2 Convolutional Layer.....	49
<b>Lampiran 24.</b> Syntax 5 Convolutional Kernel 3 Convolutional Layer.....	50
<b>Lampiran 25.</b> Hasil 5 Convolutional Kernel 3 Convolutional Layer.....	51
<b>Lampiran 26.</b> Syntax 6 Convolutional Kernel 1 Convolutional Layer.....	52
<b>Lampiran 27.</b> Hasil 6 Convolutional Kernel 1 Convolutional Layer.....	52
<b>Lampiran 28.</b> Syntax 6 Convolutional Kernel 2 Convolutional Layer.....	54

<b>Lampiran 29.</b>	<i>Hasil 6 Convolutional Kernel 2 Convolutional Layer</i>	54
<b>Lampiran 30.</b>	<i>Syntax 6 Convolutional Kernel 3 Convolutional Layer</i>	56
<b>Lampiran 31.</b>	<i>Hasil 6 Convolutional Kernel 3 Convolutional Layer</i>	56
<b>Lampiran 32.</b>	<i>Syntax 7 Convolutional Kernel 1 Convolutional Layer</i>	57
<b>Lampiran 33.</b>	<i>Hasil 7 Convolutional Kernel 1 Convolutional Layer</i>	58
<b>Lampiran 34.</b>	<i>Syntax 7 Convolutional Kernel 2 Convolutional Layer</i>	59
<b>Lampiran 35.</b>	<i>Hasil 7 Convolutional Kernel 2 Convolutional Layer</i>	59
<b>Lampiran 36.</b>	<i>Syntax 7 Convolutional Kernel 3 Convolutional Layer</i>	61
<b>Lampiran 37.</b>	<i>Hasil 7 Convolutional Kernel 3 Convolutional Layer</i>	61
<b>Lampiran 38.</b>	<i>Data yang digunakan</i>	62

# BAB 1 PENDAHULUAN

## 1.1 Latar Belakang

*Learner-centered* adalah sebuah pandangan yang menggabungkan fokus pada latar belakang individu dengan fokus pada pembelajaran. *Learner-centered* memiliki 2 fokus utama, yaitu pembelajar dan materi ajar (McCombs, 2000). Pelajar sebagai subjek pada *learner-centered* dianggap sudah memiliki bakat tertentu yang akan dikembangkan. Pengembangan bakat pelajar dilakukan dengan memberikan materi ajar dan praktik pengajaran yang paling efektif dalam mendorong tingkat motivasi, pembelajaran, dan pencapaian tertinggi bagi semua pelajar.

Pemahaman *learner-centered* lebih menekankan dasar prinsip psikologis sebagai representasi dari basis pengetahuan pelajar dan pembelajaran (McCombs, 2000). Penerapan *learner-centered* terikat dengan keyakinan, karakteristik, disposisi dan praktik pengajar. Orientasi *learner-centered* pada pelajar membuat praktik belajar menyertakan pelajar dalam pengambilan keputusan tentang praktik pengajaran dan pengambilan nilai pembelajaran; menghargai perspektif unik setiap pelajar; menghormati dan mengakomodasi perbedaan individu dalam latar belakang, minat, kemampuan, dan pengalaman pelajar; dan memperlakukan peserta didik sebagai rekan pencipta dan mitra dalam proses belajar mengajar.

Teknologi memiliki peran yang dapat dimanfaatkan dalam pengembangan pendidikan. Teknologi yang terintegrasi dengan kurikulum akan memberikan dukungan secara langsung kepada pendidikan. Teknologi dapat membuat alat bantu belajar yang beragam sesuai dengan tujuan praktik pendidikan. Penerapan teknologi pada pendidikan dilakukan dengan penerapan prinsip *learner-centered*. Sehingga, penerapan teknologi pada pendidikan dapat membuat tipe belajar bervariasi sesuai dengan latar belakang pelajar.

Pembelajaran interaktif terdistribusi merupakan salah satu bentuk penerapan teknologi pada pendidikan. Khalifa dan Lam (2002) menemukan bahwa pembelajaran pembelajaran interaktif terdistribusi lebih baik daripada pembelajaran pasif terdistribusi pada lingkungan pembelajaran berbasis web. Pembelajaran interaktif terdistribusi memberikan lingkungan belajar yang lebih mendukung pelajar. Pembelajaran interaktif terdistribusi membuat pelajar lebih aktif dan eksploratif dalam proses belajar dibandingkan pembelajaran pasif terdistribusi. Pelajar yang mengikuti pembelajaran interaktif terdistribusi mencapai nilai belajar yang lebih tinggi daripada pelajar yang mengikuti pembelajaran pasif terdistribusi.

Salah satu teknologi yang dapat mendukung pembelajaran interaktif terdistribusi adalah sistem rekomendasi. Situs belajar berbasis web banyak mengembangkan sistem rekomendasi pada web yang digunakan. Sistem rekomendasi yang digunakan biasanya memberikan rekomendasi materi pembelajaran selanjutnya berdasarkan pengalaman pelajar pada web. Sistem rekomendasi juga dapat dibuat untuk membantu pelajar menyelesaikan tugas seperti pengenalan soal.

Saat ini, terdapat banyak *platform* belajar daring yang memanfaatkan teknologi untuk melakukan distribusi pembelajaran seperti Zenius, Ruangguru, Khan Academy dll. *Platform* belajar daring sangat mengandalkan pemahaman *learner-centered* dan Teknik pembelajaran interaktif terdistribusi dalam praktiknya. *Machine learning* dapat dimanfaatkan untuk membuat sistem rekomendasi yang dapat membantu pelajar dalam penggunaan *platform* belajar daring.

Sistem rekomendasi dibuat dengan memberikan model *machine learning* beberapa data inisial sebagai inisiasi pengenalan pola. Penelitian ini akan berfokus pada pengembangan inisiasi pengenalan pembeda label soal integral dan peluang. Pada soal integral,

model akan diarahkan untuk mengenal pola-pola soal bergambar, bersymbol, dan memiliki persamaan. Pada soal dengan label peluang, model akan lebih ditekankan pada soal yang memiliki teks.

Penelitian ini dapat dikembangkan lebih lanjut dalam pembuatan API yang mampu menjadi sistem rekomendasi yang lebih canggih seperti sistem rekomendasi soal. Pengembangan dari penelitian ini dapat melakukan pengenalan label soal lebih luas tidak hanya pada mata pelajaran matematika namun pada mata pelajaran lain juga. Tetapi untuk melakukan pengembangan diperlukan variasi data yang sangat luas agar model dapat menemukan pola yang akurat pada ciri masing-masing label.

Pengembangan label soal integral dapat dibuat untuk soal-soal yang memiliki gambar, symbol maupun persamaan pada mata pelajaran lainnya. Pengembangan tersebut perlu diinisiasi pada masalah pola sederhana, yaitu variasi data yang lebih sedikit untuk melihat apakah model dapat menemukan pola soal-soal bergambar, bersymbol, dan memiliki persamaan. Pengembangan soal dengan label peluang dapat dikombinasikan dengan *text mining* agar dapat menunjukkan label yang lebih bervariasi seperti pada label mata pelajaran.

Ignacio et al. (2017) melakukan penelitian pencocokan gambar geometri menggunakan *Convolutional Neural Network*. Penelitian yang dilakukan mencocokkan 2 gambar yang berbeda secara substansi. Penelitian dilakukan dengan menggabungkan 2 gambar secara geometri dengan pendekatan transformasi *spline affine* dan memperkirakan parameternya. Arsitektur didasarkan pada tiga komponen utama, yaitu meniru Langkah-langkah standar, pencocokan dan deteksi *inlier* simultan dan estimasi parameter model. Penelitian yang dilakukan oleh Ignacio dkk. Menghasilkan bahwa jaringan arsitektur *Convolutional Neural Network* bekerja dengan baik pada perubahan-perubahan besar gambar. Oleh karena itu, jaringan arsitektur yang *Convolutional Neural Network* dapat berkerja dengan baik pada gambar dengan perbedaan jarak dan kategori.

Arsitektur *Convolutional Neural Network* juga digunakan dalam pembuatan kendaraan bawah air otomatis. Yang et al. (2019) melakukan penelitian pada kendaraan bawah air otomatis dengan menggunakan arsitektur *Convolutional Neural Network* pada kasus pencocokan data *sonar image*. Arsitektur *Convolutional Neural Network* menggunakan fitur 7x7. Data penelitian yang digunakan menggunakan gambar dengan kerapatan 300 dpi yang di ubah menjadi 224x224 dan orientasi gambar dibuat berbeda. Arsitektur terbaik yang digunakan pada penelitian ini adalah 5 *Convolutional layers* untuk melatih fitur ekstraksi.

Gao et al. (2021) melakukan penelitian menggunakan *Convolutional Neural Network* untuk membuat model yang *robust* pada kasus dari perbedaan bentuk gambar. Dataset yang digunakan adalah dataset *Best Buddy Similarity* (BBS) dan *King's*. Dataset KTM merupakan dataset yang dikembangkan mengikuti prosedur dataset BBS sebagai pembandingan. Dataset BBS dan KTM berisi pasangan gambar yang masing-masing berjumlah 105 pasangan gambar dan 200 pasangan gambar. Arsitektur *Convolutional Neural Network* digunakan untuk melakukan ekstraksi fitur. Ekstraksi fitur yang didapat akan lebih bisa menekankan perbedaan gambar yang didapat. Arsitektur *Convolutional Neural Network* terbaik yang didapatkan adalah arsitektur *Convolutional Neural Network* dengan 4 *Convolutional layers*.

Sun et al. (2021) melakukan penelitian menggunakan metode dan *convolutional neural network* untuk mendeteksi massa payudara. Penelitian ini dilakukan dengan 2 tahapan, yaitu tahap deteksi dan klasifikasi. Tahap deteksi menggunakan metode dan *mathematical morphology method* untuk mendeteksi benjolan yang muncul pada payudara. Jarak yang digunakan untuk mengukur adalah jarak *Euclidean*. Tahap klasifikasi menggunakan *convolutional neural network* untuk mengklasifikasikan bagian yang

diduga kanker payudara. ada 3 *convolutional layer* dengan kernel 5x5, 3x3, dan 3x3. Gambar yang digunakan untuk klasifikasi diubah ukuran menjadi 200x200 pixel.

Oleh karena itu, penelitian ini bertujuan untuk melakukan klasifikasi label soal matematika menggunakan *convolutional neural network* dengan label soal integral dan peluang.

## 1.2 Rumusan Masalah

Berdasarkan uraian latar belakang di atas, terdapat permasalahan pada sistem belajar aktif terdistribusi. sistem belajar aktif terdistribusi memerlukan infrastruktur yang mendukung pada pembelajaran pelajar. Salah satu infrastruktur yang dapat mendukung sistem belajar aktif terdistribusi adalah sistem rekomendasi pengenalan soal. Sistem rekomendasi pengenalan soal dapat membantu pelajar dalam menemukan soal yang tidak dikuasai oleh pelajar. Sistem rekomendasi pengenalan soal memerlukan pengklasifikasian pada gambar. Oleh karena itu, pada penelitian kali ini akan melakukan klasifikasi pada label soal matematika menggunakan *convolutional neural network*.

## 1.3 Batasan Masalah

Batasan masalah pada penelitian ini adalah data yang digunakan merupakan label data yang digunakan adalah integral dan peluang. Gambar dibuat menjadi warna hitam putih. Data diambil menggunakan *smartphone* dan mengambil data di internet. Alat yang digunakan untuk mengambil gambar adalah *smartphone* Samsung Note10+.

## 1.4 Tujuan

Berdasarkan uraian rumusan masalah di atas, berikut tujuan yang akan dicapai pada penelitian ini.

1. Melakukan klasifikasi label gambar soal menggunakan *convolutional neural network*.
2. Mendapatkan hasil klasifikasi terbaik menggunakan metode *convolutional neural network*.

## 1.5 Manfaat

Manfaat yang diharapkan melalui penelitian ini sebagai berikut.

1. Bagi keilmuan Statistika

Dapat menjadi referensi untuk penelitian-penelitian selanjutnya dalam melakukan pengelompokan data berupa teks dan gambar dengan menggunakan metode *convolutional neural network*.

2. Bagi Pembaca

Dapat menjadi referensi bagi pembaca, khususnya yang melakukan penelitian dalam bidang *image mining*.

3. Bagi pengembang layanan belajar daring.

Dapat menjadi referensi untuk pengembangan lanjutan pada pengembang layanan belajar daring.



*Halaman ini sengaja dikosongkan*

## BAB 2 TINJAUAN PUSTAKA

### 2.1 Convolutional Neural Network

*Convolutional Neural Network* (CNN) merupakan salah satu pengembangan dari konsep jaringan syaraf tiruan (*Neural Network*) (Wu, 2017). CNN bekerja sangat baik dalam penyelesaian kasus *computer vision*. Kasus-kasus yang diselesaikan oleh CNN adalah kasus-kasus klasifikasi dalam *image mining* seperti *image recognition*, *image classification* dll. CNN memproses data gambar dalam bentuk matriks maupun tensor yang memiliki pola tertentu.

Berikut merupakan arsitektur CNN yang digambarkan secara matematis.

$$\mathbf{x}^1 \rightarrow [\mathbf{w}^1] \rightarrow \mathbf{x}^2 \rightarrow \dots \mathbf{x}^{L-1} \rightarrow [\mathbf{w}^{L-1}] \rightarrow \mathbf{x}^L \rightarrow [\mathbf{w}^L] \rightarrow z$$

CNN memproses input dalam beberapa lapisan yang dinotasikan sebagai  $\mathbf{x}$  dan  $\mathbf{w}$ .  $\mathbf{x}$  adalah matriks input dan  $\mathbf{w}$  adalah matriks output. Secara proses matematis, gambar akan diproses sebagai input lalu diproses secara kolektif dan hasil pemrosesan akan digunakan untuk lapisan kedua dan seterusnya. Setelah itu, sebuah lapisan ditambahkan untuk mempelajari nilai kebaikan parameter-parameter yang digunakan dalam CNN didapatkan melalui proses *backward error propagation*. Secara matematis, sebuah strategi yang sering digunakan adalah pengodean hasil proses kolektif lapisan digunakan sebagai hasil klasifikasi. Lapisan akhir yang akan digunakan sebagai hasil klasifikasi merupakan hasil transformasi dari lapisan sebelumnya,

$$\arg \max \mathbf{x}_i^L$$

Proses pengklasifikasian dilakukan secara kolektif akan disebut sebagai proses alur maju. Pada proses ini, semua lapisan akan mempelajari lapisan sebelumnya yang difgunakan sebagai *input*. Hasil akhir pemrosesan kolektif lapisan akan didapatkan sebagai hasil lapisan estimasi probabilitas. Hasil lapisan estimasi probabilitas akan digunakan sebagai pengklasifikasi gambar dalam bentuk *voting*. Persamaan 2 menunjukkan estimasi probabilitas yang dipilih adalah estimasi probabilitas yang terbesar.

$$(\mathbf{w}^i)^{t+1} = (\mathbf{w}^i)^t - \eta \frac{\partial z}{\partial (\mathbf{w}^i)^t} \quad (2.1)$$

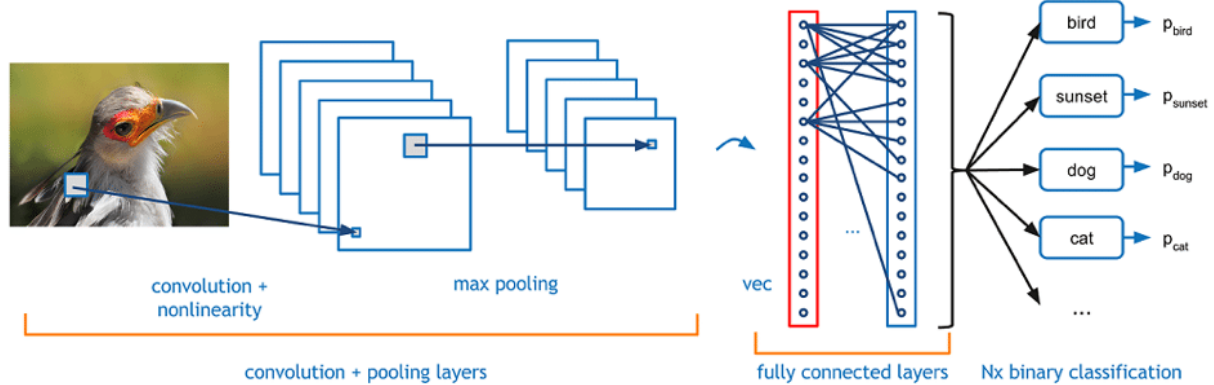
Keterangan :

- $\mathbf{W}$  : Matriks konvolusi
- $z$  : *loss layer*
- $i$  : Urutan matriks konvolusi
- $t + 1$  : Urutan matriks target konvolusi

Nilai parameter yang digunakan pada CNN bertujuan untuk meminimalisir nilai kehilangan  $z$  pada *loss layer*  $z$ . Hasil klasifikasi yang digunakan akan didapatkan dengan proses yang bekerja dengan cara mempelajari lapisan secara kolektif ke belakang. Jika suatu  $\mathbf{x}^1$  digunakan sebagai input maka proses *running* pembelajaran akan berjalan 2 arah. Pertama proses pembelajaran untuk mendapatkan hasil prediksi  $\mathbf{x}^L$  lalu pembelajaran akan dioper ke *loss layer* untuk membandingkan nilai target  $\mathbf{t}$  sesuai dengan nilai  $\mathbf{x}^1$ . Persamaan 2.1 menunjukan bahwa *loss layer* yang digunakan akan membandingkan hasil pengklasifikasian dengan nilai target.

Turunan parsial  $\frac{\partial z}{\partial \mathbf{w}^i}$  adalah laju peningkatan dari nilai kehilangan  $z$ . dalam matematika optimisasi, vector turunan parsial pada persamaan 2.1 disebut sebagai gradien. Dalam tujuan untuk meminimalisir nilai kehilangan  $z$ ,  $\mathbf{w}^i$  harus dimutakhirkan sepanjang arah berlawanan dari gradien. Turunan parsial ini melambangkan untuk perubahan dimensi  $\mathbf{w}^i$  yang berbeda. Aturan pemutakhiran ini disebut *gradient descent*.

Jika nilai gradien dipindahkan terlalu jauh maka *loss function* kemungkinan akan meningkat, oleh karena itu, pemutakhiran hanya perlu mengubah parameter-parameter melalui proporsi kecil dari gradien negatif. Hal ini dikendalikan oleh  $\eta$  (laju pembelajaran). Nilai  $\eta > 0$  biasanya disetel angka sangat kecil. Satu pemutakhiran berdasarkan ada input  $\mathbf{x}^1$  akan membuat nilai kehilangan lebih kecil ketika laju pembelajaran tidak terlalu besar.



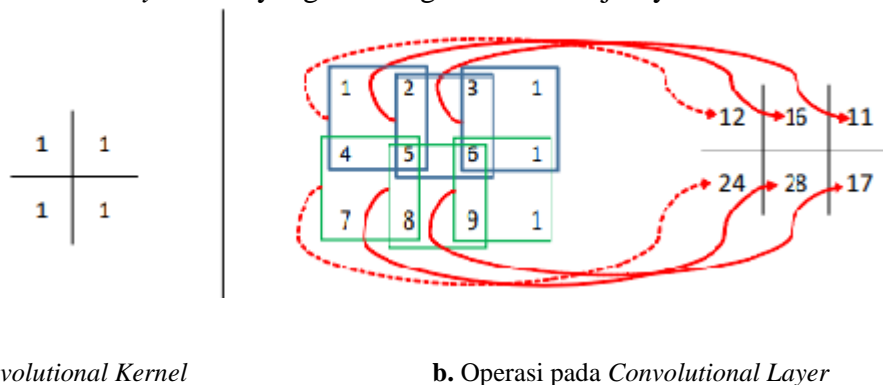
**Gambar 2.1** Arsitektur *Convolutional Neural Network*

Gambar 2.1 menunjukkan arsitektur yang digunakan oleh CNN. Arsitektur CNN memiliki beberapa lapisan penyusun, yaitu *convolutional layer*, *ReLU layer*, *pooling layer*, dan *fully connected layer*. *Convolutional layer*, *pooling layer*, dan *ReLU layer* merupakan lapisan yang melakukan fitur ekstraksi. *Fully connected layer* merupakan lapisan yang melakukan pengklasifikasian untuk menghasilkan hasil klasifikasi berdasarkan fitur ekstraksi.

### 2.3.1 *Convolutional Layer*

*Convolutional layer* adalah lapisan pertama yang akan digunakan dalam CNN (Wu, 2017). Lapisan ini terdiri dari 2 elemen, yaitu *input* dan *convolutional kernel*. *Input* adalah data gambar yang akan digunakan untuk analisis. *Convolutional kernel* adalah inti yang digunakan untuk memfilter *Convolutional layer*.

Jika kita menumpangtindihkan *convolutional kernel* di atas *input* gambar maka akan dilakukan perkalian antara nilai *convolutional kernel* dan nilai yang ada pada *input* gambar pada lokasi yang sama. Hasil perkalian dari *convolutional kernel* dan *input* gambar akan mendapatkan angka tunggal. Angka tunggal tersebut didapatkan dengan cara menjumlahkan semua hasil perkalian *convolutional kernel* dan *input* gambar. Jika operasi ini dilakukan hingga semua nilai pada *input* gambar ditumpangtindihkan dengan *convolutional kernel* maka akan didapatkan *convolutional layer* baru yang akan digunakan selanjutnya.



**a.** *Convolutional Kernel*

**b.** Operasi pada *Convolutional Layer*

**Gambar 2.2** Ilustrasi operasi *convolutional layer*

Gambar 2.2 menunjukkan ilustrasi operasi pada *convolutional layer*. Gambar a menunjukkan nilai yang digunakan dalam *convolutional kernel*. Gambar b menunjukkan operasi

yang dilakukan ketika menumpangtindihkan *convolutional kernel* pada nilai *input*. Gambar b menunjukkan bahwa didapatkan *convolutional layer* terbaru dengan nilai 12, 15, 11, 24, 28 dan 17. Convolutional layer dapat dibuat berganda agar mendapatkan hasil yang maksimal.

### 2.3.2 ReLU Layer

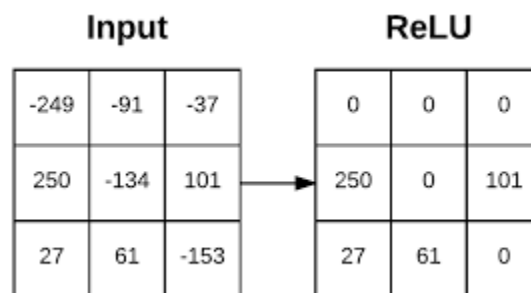
*ReLU layer (Rectified Linear unit)* adalah lapisan yang digunakan untuk meloloskan nilai yang lebih besar dari nol di *convolutional layer* (Wu, 2017).

$$y_{i,j,d} = \max\{0, x_{i,j,d}^l\} \quad (2.2)$$

Keterangan :

- y : Matriks ReLU
- x : Matriks konvolusi
- i : Baris
- j : Kolom
- d : *channel* warna

Berdasarkan persamaan 2.2, Apabila nilai pada *convolutional layer* lebih kecil dari 0 maka nilai tersebut akan diganti menjadi 0. Tujuan dari *ReLU Layer* adalah untuk meningkatkan nonlinearitas dari CNN. Kasus gambar semantic biasanya memiliki pemetaan yang sangat nonlinear pada nilai input *pixel*. Secara intuitif, *ReLU layer* sangat berguna untuk mengena pola-pola kompleks maupun objek-objek.



Gambar 2.3 ReLU Layer

*ReLU layer* membuat gradien dari fitur di layer ke-1 menjadi 0 apabila kurang dari 0. Hal ini bermakna tidak menarik atau tidak aktif. Untuk fitur-fitur yang diaktivasi, gradien adalah proses *back propagated* tanpa perubahan apapun yang mengundungkan untuk *stochastics gradient descent learning*. *ReLU* menjadi sangat penting di CNN karena mereduksi kesulitan pembelajaran parameter-paramater CNN dan memperbaiki akurasi CNN.

### 2.3.4 Pooling Layer

*Pooling layer* merupakan salah satu *layer* yang akan digunakan dalam arsitektur CNN (O'Shea dkk., 2015). *Pooling layer* tidak membutuhkan parameter dalam beroperasi. Sebuah *pooling layer* beroperasi atas input kanal  $\mathbf{x}^l$  secara independen. Operator *pooling layer* akan memetakan sebuah subregion menjadi sebuah angka tunggal.



Gambar 2.4 Average Pooling dan Max Pooling

Dua tipe operator *pooling layer* yang sering digunakan adalah *max pooling* dan *average pooling*. Operator *max pooling* akan memetakan sebuah sub region menjadi nilai terbesar.

Operator *average pooling* akan memetakan sebuah subregion menjadi nilai rata-rata. Berikut persamaan matematis yang digunakan untuk kedua operator tersebut.

$$\text{Max} : y_{i^{l+1},j^{l+1},d} = \max_{0 \leq i < H, 0 \leq j < W} x_{i^{l+1} \times H + i, j^{l+1} \times W + j, d}^l \quad (2.3)$$

$$\text{Average} : y_{i^{l+1},j^{l+1},d} = \frac{1}{HW} \sum_{0 \leq i < H, 0 \leq j < W} x_{i^{l+1} \times H + i, j^{l+1} \times W + j, d}^l \quad (2.4)$$

Keterangan :

- y : Matriks konvolusi terbaru
- x : Matriks ReLU
- i : Baris
- j : Kolom
- d : *Channel* warna
- H : Nilai maksimum baris
- W : Nilai maksimum kolom

### 2.3.4 Fully Connected layer

*Fully connected layer* adalah hasil dari *output* operasi lokal yang sudah dilakukan pada output  $\mathbf{x}^{l+1}$  (atau  $\mathbf{y}$ ) (O'Shea dkk., 2015). *Fully connected layer* berisi hasil *output* ReLU dan *Pooling* layer yang berisi perwakilan terdistribusi dari input. *Fully connected layer* memiliki kapabilitas yang lebih kuat sebagai perwakilan *input* untuk pengklasifikasian. Anggaplah *input layer*  $\mathbf{x}^1$  berbentuk tensor ordo 3 dan kernel konvolusi berbentuk tensor ordo 3. Lalu, sejumlah kernel itu membentuk sebuah tensor ordo 4 sebagai output maka sangat jelas bahwa semua *output*  $\mathbf{y}$  berasal dari input  $\mathbf{x}^1$ .

## 2.2 Evaluasi Hasil Klasifikasi

Evaluasi hasil klasifikasi adalah alat pengukuran yang digunakan untuk menilai hasil pengklasifikasi (Han dkk., 2012). Ada dua jenis tipe kasus yang harus dicermati ketika mengevaluasi hasil klasifikasi, yaitu data *balance* dan data *imbalance*. Data *balance* bisa diukur menggunakan akurasi. Data *imbalance* diukur menggunakan sensitivity, specificity, precision, nilai ukuran  $F$ .

Evaluasi hasil klasifikasi akan menilai pengklasifikasi dengan data yang memiliki hasil sebagai berikut.

1. **True positif** adalah data yang *positive* dinilai benar *positive* oleh pengklasifikasi.
2. **True negatif** adalah data yang *negative* dinilai benar *negative* oleh pengklasifikasi.
3. **False positive** adalah data yang *negative* yang dinilai salah *positive* oleh pengklasifikasi.
4. **False negative** adalah data yang *negative* yang dinilai salah *positive* oleh pengklasifikasi.

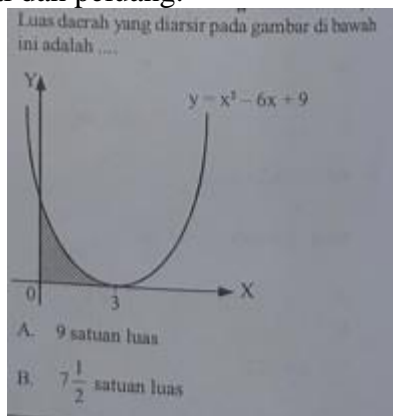
Akurasi adalah alat pengukuran evaluasi klasifikasi yang menilai data yang benar dinilai oleh pengklasifikasi. Dalam kasus pengenalan pola, akurasi dikenal dengan sebutan laju pengenalan. Akurasi akan mengukur *true positive* dan *true negative* dibandingkan dengan total data. Berikut merupakan rumus yang digunakan untuk menghitung akurasi.

$$\text{accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total Data yang Digunakan}} \quad (2.6)$$

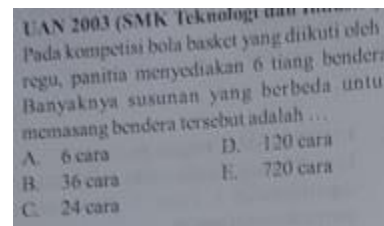
## BAB 3 METODOLOGI

### 3.1 Sumber Data

Data yang digunakan pada penelitian tugas akhir ini adalah data primer, yaitu gambar yang didapat melalui smartphone Samsung note10+ dengan file berformat jpeg dan gambar dari website defantri.com. Gambar akan diekstraksi menjadi warna hitam putih. Label soal yang digunakan, yaitu soal integral dan peluang. Jumlah gambar soal yang diambil sebanyak 160 dengan komposisi 80 soal integral dan peluang. Berikut contoh soal integral dan peluang.



(a)



(b)

**Gambar 3.1 (a)** Contoh soal integral **(b)** Contoh soal peluang

Gambar 3.1 (a) menunjukkan bahwa gambar soal integral memiliki beberapa karakteristik. Soal integral memiliki gambar seperti yang ditunjukkan pada kurva garis, memiliki persamaan, memiliki simbol dan teks. Karakteristik tersebut akan digunakan untuk menunjukkan kekhasan gambar soal integral saat membentuk model. Gambar 3.1 (b) menunjukkan contoh soal peluang. Karakteristik soal peluang hanya memiliki teks yang disusun oleh huruf dan angka saja. Karakteristik tersebut akan digunakan untuk menjadi ciri khas gambar soal peluang dalam pemodelan.

### 3.2 Variabel Penelitian

Variabel yang akan digunakan merupakan ekstraksi fitur pixel pada gambar dengan warna hitam putih.

**Tabel 3.1** Variabel penelitian yang digunakan

Variabel	Keterangan	Skala	Nilai
X	Nilai ekstraksi warna	Interval	0-255
Y	Label gambar soal	Nominal	Integral dan peluang

Variabel X merupakan nilai ekstraksi warna foto yang digunakan. Nilai ekstraksi warna foto memiliki skala interval dengan interval 0-255. Variabel Y merupakan label gambar. Label foto memiliki skala nominal dengan 2 label, yaitu label integral dan peluang. Berikut merupakan struktur data yang akan digunakan pada penelitian kali ini.

**Tabel 3.2** Struktur data

Gambar ke-	No.	X <sub>1</sub>	X <sub>2</sub>	...	X <sub>500</sub>	Y
1	1.	X <sub>1,1</sub>	X <sub>1,2</sub>	...	X <sub>1,500</sub>	Y <sub>1</sub>
	2.	X <sub>2,1</sub>	X <sub>2,2</sub>	...	X <sub>2,500</sub>	
	...	...	...	...	...	
	500	X <sub>500,1</sub>	X <sub>500,2</sub>	...	X <sub>500,500</sub>	
2	501	X <sub>501,1</sub>	X <sub>501,2</sub>	...	X <sub>501,500</sub>	Y <sub>2</sub>
	502	X <sub>502,1</sub>	X <sub>502,2</sub>	...	X <sub>502,500</sub>	
	...	...	...	...	...	
	1000	X <sub>1000,1</sub>	X <sub>1000,2</sub>	...	X <sub>1000,500</sub>	
...	...	...	...	...	...	...
160	79840	X <sub>79841,1</sub>	X <sub>79841,2</sub>	...	X <sub>79840,500</sub>	Y <sub>160</sub>
	79841	X <sub>79842,1</sub>	X <sub>79842,2</sub>	...	X <sub>79841,500</sub>	
	...	...	...	...	...	
	80000	X <sub>80000,1</sub>	X <sub>80000,2</sub>	...	X <sub>80000,500</sub>	

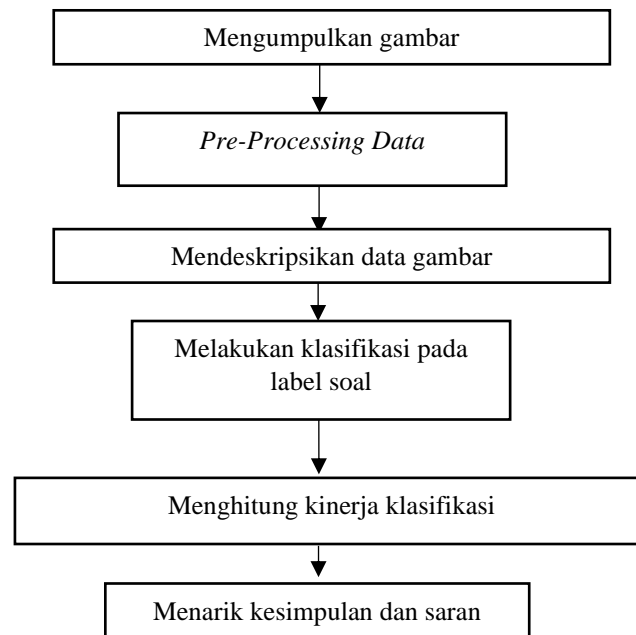
Jumlah foto yang digunakan sebanyak 160 foto dengan ukuran *pixel* 500x500 sehingga didapatkan 80000x500 fitur yang akan digunakan sebagai data input. Variabel X<sub>1</sub> merupakan fitur ke-1 dan X<sub>1,1</sub> merupakan nilai pixel yang didapatkan dari fitur ekstraksi. Y<sub>1</sub> merupakan label ke-1 yang digunakan dalam data.

### 3.3 Langkah Penelitian

Langkah penelitian yang akan digunakan dalam penelitian tugas akhir ini adalah sebagai berikut.

1. Melakukan studi literatur dan referensi terkait topik dan metode dalam penelitian.
2. Mengumpulkan data dengan cara mengambil gambar.
3. Melakukan pre-processing data sebagai berikut.
  - a. Melakukan ekstraksi data gambar.
  - b. Mengubah gambar menjadi warna hitam putih.
  - c. Mengubah ukuran pixel gambar menjadi 500x500.
4. Mengulangi Langkah 3a hingga 3c pada semua gambar yang digunakan.
5. Membuat gambaran umum data gambar.
6. Melakukan klasifikasi label soal.
7. Menghitung nilai kebaikan hasil klasifikasi.
8. Memberikan kesimpulan dan saran berdasarkan hasil penelitian.

Berikut merupakan Langkah penelitian pada penelitian ini.



**Gambar 3.2** Diagram langkah penelitian



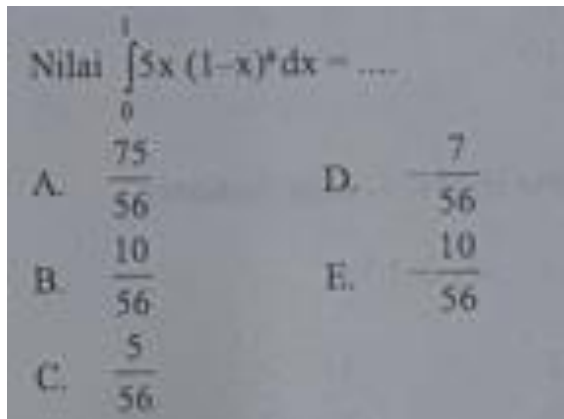
*Halaman ini sengaja dikosongkan*

## BAB 4 Hasil dan Pembahasan

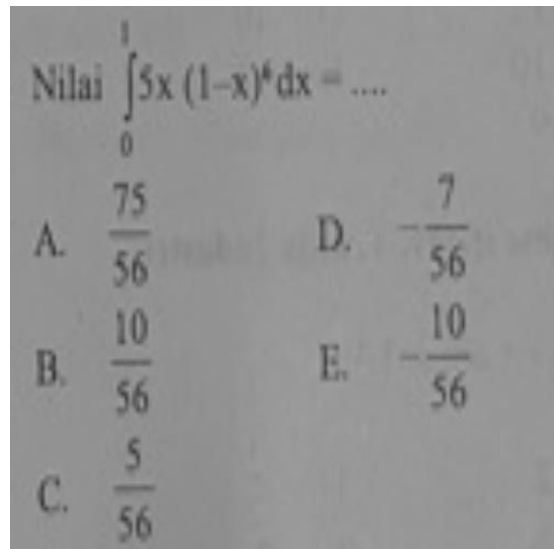
Pada penelitian ini dilakukan klasifikasi label soal integral dan peluang menggunakan metode *Convolutional Neural Network* (CNN). Kebaikan hasil klasifikasi pada penelitian ini akan diukur menggunakan *accuracy*. Penelitian ini akan menarik kesimpulan berupa jumlah *convolutional layer* terbaik yang digunakan dalam klasifikasi label soal.

### 4.1 Preprocessing Data Foto Soal

Pada penelitian ini, tahap *preprocessing* yang dilakukan adalah mengubah warna pada fitur ekstraksi yang digunakan menjadi berwarna hitam putih. Berikut merupakan salah satu contoh fitur ekstraksi yang sudah diubah menjadi berwarna hitam putih.



(a)



(b)

**Gambar 4.1** (a) Foto soal sebelum preprocessing (b) Foto soal setelah preprocessing

Pada proses *preprocessing*, foto soal diubah menjadi berwarna hitam putih dan ukuran pixel diubah menjadi 500x500. Tahap *preprocessing* dilakukan pada semua data foto yang digunakan sehingga data input yang didapatkan sebesar 80000x500 fitur dengan 2 label, yaitu label integral dan peluang. Berikut merupakan matriks data ke-1 yang sudah diubah dari berwarna menjadi hitam putih.

$$\begin{pmatrix} 254 & 253 & 255 \\ \vdots & \vdots & \vdots \\ 251 & 253 & 254 \\ 254 & 254 & 255 \\ \vdots & \vdots & \vdots \\ 251 & 253 & 254 \\ 253 & 252 & 254 \\ \vdots & \vdots & \vdots \\ 251 & 253 & 254 \end{pmatrix}$$

(a)

$$\begin{pmatrix} 254 & 251 & 253 \\ \vdots & \vdots & \vdots \\ 253 & 253 & 253 \end{pmatrix}$$

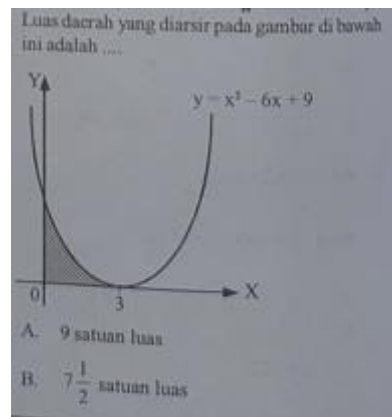
(b)

**Gambar 4.2** (a) Matriks RGB sebelum praproses (b) Matriks RGB setelah praproses

### 4.2 Deskripsi Data Hasil Preprocessing

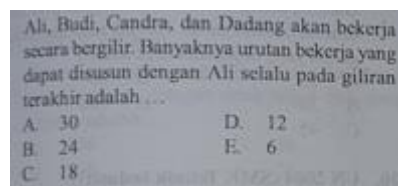
Data yang didapat dari *preprocessing* adalah data hasil *feature extraction* berwarna hitam putih dengan ukuran pixel 500x500. Nilai *feature* ekstraksi yang semakin besar menunjukkan bahwa warna pada gambar yang dilakukan fitur ekstraksi semakin putih. Nilai fitur ekstraksi

yang semakin kecil menunjukkan bahwa warna pada gambar yang dilakukan fitur ekstraksi semakin hitam. Label yang digunakan sebanyak 2 label, yaitu label bergambar dan label tidak bergambar. Jumlah label yang digunakan sebanyak 80 label integral dan 80 label peluang. Berikut merupakan contoh gambar soal integral.



**Gambar 4.2** contoh soal integral

Gambar 4.2 menunjukkan bahwa gambar soal integral memiliki beberapa karakteristik. Soal integral memiliki gambar seperti yang ditunjukkan pada kurva garis, memiliki persamaan, memiliki simbol dan teks. Karakteristik tersebut akan digunakan untuk menunjukkan kekhasan gambar soal integral saat membentuk model. Berikut merupakan contoh soal peluang.

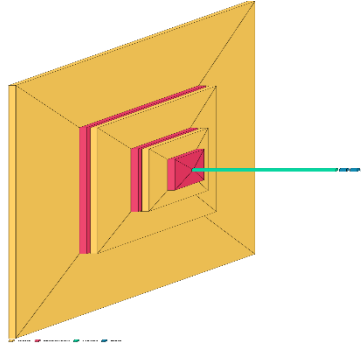


**Gambar 4.3** Contoh soal peluang

Gambar 4.3 menunjukkan contoh soal peluang. Karakteristik soal peluang hanya memiliki teks yang disusun oleh huruf dan angka saja. Karakteristik tersebut akan digunakan untuk menjadi ciri khas gambar soal peluang dalam pemodelan.

### 4.3 Klasifikasi Label Gambar Soal Menggunakan CNN

Setelah melakukan preprocessing data pada setiap *feature* yang digunakan, akan dilakukan pengklasifikasian label soal integral dan peluang dengan menggunakan metode *Convolutional Neural Network* (CNN). Jumlah *convolution kernel* yang digunakan pada tahap ini sejumlah 2,3,4,5,6, dan 7 *convolutional kernel size* dengan masing-masing *convolutional layer* maksimal 3 *convolutional layer*. Berikut merupakan contoh arsitektur CNN yang digunakan dalam penelitian ini.



**Gambar 4.4** Contoh arsitektur CNN

Gambar 4.4 menunjukkan contoh visualisasi arsitektur CNN yang digunakan dalam penelitian ini. Arsitektur yang dijadikan contoh pada gambar 4.4 adalah arsitektur dengan 2 *Convolutional kernel* dan 3 *convolutional layer*. Berikut merupakan contoh perhitungan matriks konvolusi pada penelitian ini.

$$\begin{pmatrix} 254 & 251 & \dots & 253 \\ \vdots & \vdots & \ddots & \vdots \\ 253 & 253 & \dots & 253 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \quad \begin{pmatrix} 6 & -6 & \dots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

(a) (b) (c)

**Gambar 4.5** (a) Matriks konvolusi (b) matriks kernel konvolusi (c) Matriks konvolusi terbaru

Gambar 4.5 (a) merupakan matriks konvolusi yang digunakan sebagai input pada penelitian ini. Matriks kernel konvolusi pada gambar 4.5 (b) merupakan salah satu matriks pencari yang digunakan dalam penelitian ini. Matriks konvolusi akan dikalikan dengan matriks kernel konvolusi sehingga menghasilkan matriks konvolusi terbaru pada gambar 4.5 (c). berikut merupakan gambaran pada proses matriks ReLU.

$$\begin{pmatrix} 6 & -6 & \dots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix} \quad \begin{pmatrix} 6 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

(a) (b)

**Gambar 4.6** (a) matriks konvolusi (b) matriks ReLU

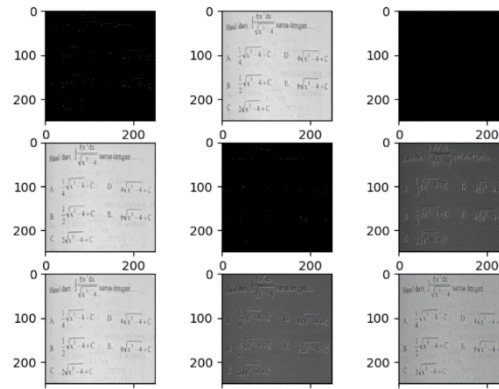
Gambar 4.6 merupakan gambar yang menunjukkan proses perubahan dari matriks konvolusi menjadi matriks ReLU. Pada gambar 4.6 (a) masih terdapat nilai negative seperti -6 dan -1. Gambar 4.6 (b) menunjukkan bahwa matriks konvolusi sudah diubah menjadi matriks ReLU karena sudah tidak memiliki nilai negatif. Berikut merupakan ilustrasi dari pengubahan matriks ReLU menjadi matriks *pooling* menggunakan *max pooling*.

$$\begin{pmatrix} 6 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix} \quad \begin{pmatrix} 6 & 0 & 5 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

(a) (b)

**Gambar 4.7** (a) matriks ReLU (b) matriks konvolusi terbaru

Gambar 4.7 (a) menunjukkan matriks ReLU yang didapatkan dari proses ReLU. Gambar 4.7 (b) menunjukkan bahwa matriks ReLU sudah diubah menjadi matriks konvolusi terbaru menggunakan metode *max pooling*. Matriks tersebut akan digunakan pada tahapan selanjutnya dalam proses *neural network*. Berikut merupakan bentuk visualisasi pencirian yang terjadi pada gambar yang digunakan pada penelitian ini.



**Gambar 4.8** Proses pencirian pada salah gambar

Berikut pembahasan lebih lanjut untuk masing-masing model yang dibentuk dari arsitektur.

#### 4.3.1 Klasifikasi Label Gambar Soal menggunakan CNN dengan 2 *convolutional kernel*

Berikut merupakan tabel hasil klasifikasi model CNN dengan 2 *convolutional kernel* pada arsitektur CNN dengan maksimal 3 *convolutional layer*.

**Tabel 4.1** Hasil kebaikan klasifikasi pada data *training* dan *testing* menggunakan 2 *convolutional kernel*

No.	Convolutional Layer	Akurasi	
		Training	Testing
1.	1	0,5000	0,5000
2.	2	0,9375	0,8125
3.	3	<b>0,9453</b>	<b>0,9375</b>

Tabel 4.1 menunjukkan hasil kebaikan klasifikasi model CNN dengan 2 *convolutional kernel* pada arsitektur CNN dengan maksimal 3 *convolutional layer*. Pada model-model tersebut, didapatkan bahwa model CNN dengan 2 *convolutional kernel* dan 3 *convolutional layer* merupakan model terbaik dengan nilai akurasi pada data training dan testing masing-masing sebesar 0,9453 dan 0,9375.

#### 4.3.2 Klasifikasi Label Gambar Soal menggunakan CNN dengan 3 *convolutional kernel*

Berikut merupakan tabel hasil klasifikasi model CNN dengan 3 *convolutional kernel* pada arsitektur CNN dengan maksimal 3 *convolutional layer*.

**Tabel 4.2** Hasil kebaikan klasifikasi pada data *training* dan *testing* menggunakan 3 *convolutional kernel*

No.	Convolutional Layer	Akurasi	
		Training	Testing
1.	1	0,7188	0,5000
2.	2	0,5156	<b>0,7500</b>
3.	3	<b>0,8750</b>	0,6875

Tabel 4.2 menunjukkan bahwa nilai hasil kebaikan klasifikasi pada model CNN dengan 3 *convolutional kernel* pada arsitektur dengan maksimal 3 *convolutional layer* mendapatkan nilai akurasi pada data *training* dan *testing* masing-masing sebesar 0,7188 dan 0,5000 pada arsitektur dengan 1 *convolutional layer*. Model CNN 3 *convolutional kernel* dengan 2 *convolutional layer* mendapatkan hasil kebaikan klasifikasi sebesar 0,5156 pada data *training* dan 0,7500 pada data testing. Sedangkan model CNN 3 *convolutional kernel* dengan 3 *convolutional layer* mendapatkan nilai hasil kebaikan klasifikasi sebesar 0,8750 pada data *training* dan 0,6875 pada data *testing*.

#### 4.3.3 Klasifikasi Label Gambar Soal menggunakan CNN dengan 4 *convolutional kernel*

Berikut merupakan tabel hasil klasifikasi model CNN pada data label soal integral dan peluang dengan 4 *convolutional kernel* pada arsitektur maksimal 3 *convolutional layer*.

**Tabel 4.3** Hasil kebaikan klasifikasi pada data *training* dan *testing* menggunakan 4 *convolutional kernel*

No.	Convolutional Layer	Akurasi	
		Training	Testing
1.	1	0,6172	<b>0,7812</b>
2.	2	<b>0,6406</b>	0,5000
3.	3	0,5391	0,5000

Tabel 4.3 menunjukkan nilai hasil kebaikan klasifikasi model CNN dengan 4 *convolutional kernel* pada arsitektur maksimal 3 *convolutional layer*. Model CNN 4 *convolutional kernel* dengan 1 *convolutional layer* mendapatkan hasil kebaikan klasifikasi pada data *training* dan *testing* masing-masing sebesar 0,6172 dan 0,7812. Model CNN dengan 4 *convolutional kernel* dengan 2 *convolutional layer* mendapatkan hasil kebaikan klasifikasi pada data *training* dan data *testing* masing-masing sebesar 0,6406 dan 0,5000. Model CNN dengan 4 *convolutional kernel* dengan 3 *convolutional layer* mendapatkan hasil klasifikasi pada data *training* dan *testing* masing-masing sebesar 0,5391 dan 0,5000. Model CNN terbaik pada data *training* didapatkan oleh model CNN 4 *convolutional kernel* dengan 2 *convolutional layer* dan model CNN terbaik pada data *testing* didapatkan oleh model CNN dengan 4 *convolutional kernel* dengan 1 *convolutional layer*.

#### 4.3.4 Klasifikasi Gambar Soal menggunakan CNN dengan 5 *convolutional kernel*

Berikut merupakan hasil kebaikan klasifikasi model CNN dengan 5 *convolutional kernel* dengan maksimal 3 *convolutional layer* pada data berlabel integral dan peluang.

**Tabel 4.4** Hasil kebaikan klasifikasi pada data *training* dan *testing* menggunakan 5 *convolutional kernel*

No.	Convolutional Layer	Akurasi	
		Training	Testing
1.	1	0,7031	<b>0,5312</b>
2.	2	0,5000	0,5000
3.	3	<b>0,8047</b>	0,5000

Tabel 4.4 menunjukkan hasil kebaikan klasifikasi pada data *training* dan data *testing* label soal integral dan peluang menggunakan 5 *convolutional kernel* dengan maksimal 3 *convolutional layer*. Model CNN terbaik yang didapatkan pada pengklasifikasian data *training* adalah model CNN menggunakan 5 *convolutional kernel* dengan 3 *convolutional layer* yang mendapatkan nilai akurasi 0,8047. Model CNN terbaik yang mengklasifikasi data *testing* adalah model CNN menggunakan 5 *convolutional kernel* dengan 1 *convolutional layer*.

#### 4.3.5 Klasifikasi Gambar Soal menggunakan CNN dengan 6 *convolutional kernel*

Berikut merupakan tabel nilai hasil kebaikan klasifikasi model CNN menggunakan 6 *convolutional kernel* dengan maksimal menggunakan 3 *convolutional layer* pada data label soal integral dan peluang.

**Tabel 4.5** Hasil kebaikan klasifikasi pada data *training* dan *testing* menggunakan 6 *convolutional kernel*

No.	Convolutional Layer	Akurasi	
		Training	Testing
1.	1	0,6406	0,7500
2.	2	0,5000	0,5000
3.	3	<b>0,8984</b>	<b>0,8125</b>

Tabel 4.5 menunjukkan nilai hasil kebaikan klasifikasi pada data *training* dan *testing* menggunakan 6 *convolutional kernel* dengan maksimal 3 *convolutional layer*. Model terbaik pada pengklasifikasian data *training* dan data *testing* didapatkan oleh model CNN dengan 6 *convolutional kernel* dengan 3 *convolutional layer* dengan nilai akurasi sebesar 0,8984 untuk data *training* dan 0,8125 untuk data *testing*.

#### 4.3.6 Klasifikasi Gambar Soal menggunakan CNN dengan 7 *convolutional kernel*

Berikut merupakan tabel hasil kebaikan klasifikasi model CNN dengan 7 *convolutional kernel* dengan maksimal 3 *convolutional layer* pada data label soal integral peluang.

**Tabel 4.6** Hasil kebaikan klasifikasi pada data *training* dan *testing* menggunakan 7 *convolutional kernel*

No.	<i>Convolutional Layer</i>	Akurasi	
		<i>Training</i>	<i>Testing</i>
1.	1	0,5000	0,5000
2.	2	<b>0,5781</b>	<b>0,8750</b>
3.	3	0,5000	0,5625

Tabel 4.6 menunjukkan bahwa nilai hasil kebaikan klasifikasi model CNN dengan 7 *convolutional kernel* terbaik didapatkan oleh model dengan arsitektur 2 *convolutional layer* yang mendapatkan nilai kebaikan pada data *training* sebesar 0,5781 dan nilai kebaikan hasil klasifikasi pada data *testing* sebesar 0,8750.

#### 4.4 Perbandingan Performa Klasifikasi Setiap *Convolutional Kernel*

Setelah dilakukan pengklasifikasian menggunakan model *Convolutional Neural Network* dengan jumlah *convolution kernel* sejumlah 2,3,4,5,6 dan 7 dalam jumlah *convolutional layer* sejumlah maksimal 3 dan didapatkan nilai kebaikan klasifikasi berupa akurasi pada masing-masing model CNN maka akan dilakukan perbandingan model untuk mendapatkan model terbaik. Berikut merupakan tabel perbandingan performansi model CNN.

**Tabel 4.7** Perbandingan performansi model CNN

No.	<i>Convolutional Kernel</i>	<i>Convolutional layer</i>	Akurasi	
			<i>Training</i>	<i>Testing</i>
1.	2	3	<b>0,9453</b>	<b>0,9375</b>
2.	3	3	0,8750	0,6875
3.	4	1	0,6172	0,7812
4.	5	3	0,8047	0,5000
5.	6	3	0,8984	0,8125
6.	7	2	0,5781	0,8750

Tabel 4.7 menunjukkan bahwa model *convolutional neural network* dengan *convolution kernel* dan *convolutional layer* masing-masing sebanyak 2 dan 3 merupakan model terbaik pada data *training* dan *testing* dengan nilai akurasi masing-masing 0,9453 dan 0,9375.

## BAB 5 Kesimpulan dan Saran

### 5.1 Kesimpulan

Berdasarkan hasil dan pembahasan didapatkan beberapa kesimpulan yang dapat ditarik sebagai berikut.

1. Soal dengan label integral memiliki persamaan, simbol, dan gambar. Soal dengan label peluang hanya memiliki teks.
2. Model terbaik pada *convolutional kernel 2* adalah model CNN dengan 3 *convolutional layer*, Model terbaik pada *convolutional kernel 3* adalah model CNN dengan 3 *convolutional layer*, Model terbaik pada *convolutional kernel 4* adalah model CNN dengan 1 *convolutional layer*, Model terbaik pada *convolutional kernel 5* adalah model CNN dengan 3 *convolutional layer*, Model terbaik pada *convolutional kernel 6* adalah model CNN dengan 3 *convolutional layer*, Model terbaik pada *convolutional kernel 7* adalah model CNN dengan 2 *convolutional layer*.
3. Model CNN terbaik yang didapatkan dari penelitian ini adalah model CCN dengan *convolutional kernel 2* dan 3 *convolutional layer* dengan nilai akurasi 0,9453 dan 0,9375 pada data training dan data testing.

### 5.2 Saran

Berdasarkan kesimpulan yang diperoleh, berikut saran yang dipertimbangkan untuk penelitian selanjutnya.

1. Menambah jumlah data dan variasi data yang digunakan agar mendapatkan hasil yang lebih baik karena dengan jumlah data dan variasi data yang banyak dapat membuat model dapat mendeteksi pola yang lebih kompleks.
2. Mencoba menggunakan metode klasifikasi yang lain dan menjadikan model yang didapatkan pada penelitian ini sebagai pembandingan.



## DAFTAR PUSTAKA

- Bo Gao dan Michael W Spratling. (2021). *Robust via Hierarchical Convolutional Features from a Shape Biased CNN*. Department of Informatics, King's College London, London, UK.
- Greg Pass dan Ramin Zabih. (1996). *Histogram Refinement for Content Based Image Retrieval*. Computer Science, Cornell University, Ithaca, New York.
- Ignacio Rocco, Relja Arandjelovic, dan Josef Sivic. (2017). *Convolutional Neural Network for Geometric Matching*. CVF.
- Jianxin Wu. (2017). *Introduction to Convolutional Neural Network*. Lamda Group.
- Jiawei Han, Michael Kamber, dan Jian Pei. (2012). *Data Mining Concept and Technique*. Morgan Kaufmann
- Keiron O'Shea dan Ryan Nash. (2015). *An Introduction to Convolutional Neural Network*. arXiv:1511.98458v2 [cs.NE]
- Beibei Huo, Guxia Kang, Kui Liu, dan Ningbo Zhang. (2018). *Breast Cancer Classification Based on Fully Connected Layer First Convolutional Neural Network*. IEEE.
- Lilei Sun, Huijie Sun, Junqian Wang, Shuai Wu, Yong Zhao, dan Yong Xu. (2021). *Breast Mass Detection in Mammography Based on Image and CNN*. MDPI
- McCombs dan Barbara L.(2000). *Assessing the Role of Educational Technology in the Teaching and Learning Porcess: A Learner-Centered Perspective*. Eric.
- Michael Ryan dan Novita Hanifah. (2015). *An Examination of Character Recognition on ID Card Using Approach*. ICCSCI 2015.
- Mohamed Khalifa dan Rinky Lam. (2002). *Web-Based Learning : Effects on Learning Process and Outcome*. IEEE.
- Wenli Yang, Shuangshuang Fan, Shuxiang Xu, Peter King, Byeong Kang, dan Eonjoo Kim. (2019). *Autonomous Underwater Vehicle Navigation Using Sonar Image Matching based on Convolutional Neural Network*. IFAC.

## LAMPIRAN

### Lampiran 1. Syntax Pre-processing

```
import cv2
import os
import numpy as np
from random import shuffle
from tqdm import tqdm
import pandas as pd
TRAIN_DIR = 'D:/Kuliah/Tugas Akhir/Gambar/train'
TEST_DIR = 'D:/Kuliah/Tugas Akhir/Gambar/test'
IMG_SIZE = 500
LR = 1e-3
# Function to Get the current
# working directory
def current_path():
    print("Current working directory before")
    print(os.getcwd())
    print()
current_path()
os.chdir('D:\Kuliah\Tugas Akhir\Referensi\Fix\Model')
current_path()
"""Setting up the model which will help with tensorflow models"""
MODEL_NAME = 'IntegralVSPeluang-{}-{}.model'.format(LR, '6conv-basic')
def func():
    return 0.5
"""Labelling the dataset"""
def label_img(img):
    word_label = img.split('.')[0]
    # DIY One hot encoder
    if word_label == 'Integral': return ['Integral']
    elif word_label == 'Peluang': return ['Peluang']
"""Creating the training data"""
def create_train_data():
    # Creating an empty list where we should store the training data
    # after a little preprocessing of the data
    training_data = []

    # tqdm is only used for interactive loading
    # loading the training data
    for img in tqdm(os.listdir(TRAIN_DIR)):

        # labeling the images
        label = label_img(img)

        path = os.path.join(TRAIN_DIR, img)
```

## Lampiran 1. Syntax pre-processing

```
# loading the image from the path and then converting them into
# grayscale for easier covnet prob
img = cv2.imread(path)

# resizing the image for processing them in the covnet
img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))

## grayscale for easier covnet prob
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# final step-forming the training data list with numpy array of the images
training_data.append([img, label])

# shuffling of the training data to preserve the random state of our data
shuffle(training_data, func)

# saving our trained data for further uses if required
np.save('train_data.npy', training_data)
return training_data

"""Processing the given test data"""
# Almost same as processing the training data but
# we dont have to label it.
def process_test_data():
    testing_data = []
    for img in tqdm(os.listdir(TEST_DIR)):
        path = os.path.join(TEST_DIR, img)
        img_num = img.split('.')[0]
        img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        testing_data.append([np.array(img), img_num])

    shuffle(testing_data)
    np.save('test_data.npy', testing_data)
    return testing_data

"""Running the training and the testing in the dataset for our model"""
train_data = create_train_data()
test_data = process_test_data()
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
```

## Lampiran 1. Syntax pre-processing

```
"""Setting up the features and labels"""
# X-Features & Y-Labels
labels = LabelEncoder()
X = np.array([i[0] for i in train_data])/255
Y = [i[1] for i in train_data]
labels.fit(Y)
Y = labels.transform(Y)
X.shape
train_images, test_images, train_labels, test_labels = train_test_split(X,Y,
test_size=0.2, random_state=123)
```

## Lampiran 2. Syntax Prediksi 2 Convolutional Kernel 1 Convolutional Layer

```
model = models.Sequential()
model.add(layers.Conv2D(32, (2, 2), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
"""Fitting the data into our model"""
#es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(train_images, np.asarray(train_labels), epochs = 20,
                    validation_data =(test_images, np.asarray(test_labels)),
                    batch_size = 200)
```

### Lampiran 3. Hasil Prediksi 2 Convolutional Kernel 1 Convolutional Layer

```
Epoch 1/20
1/1 [=====] - 64s 64s/step - loss: 0.7001 - accuracy: 0.5000 - val_loss: 5.4743 - val_accuracy: 0.5000
Epoch 2/20
1/1 [=====] - 32s 32s/step - loss: 4.9393 - accuracy: 0.5000 - val_loss: 44.6750 - val_accuracy: 0.5000
Epoch 3/20
1/1 [=====] - 35s 35s/step - loss: 46.2739 - accuracy: 0.5000 - val_loss: 28.9080 - val_accuracy: 0.5000
Epoch 4/20
1/1 [=====] - 32s 32s/step - loss: 29.9103 - accuracy: 0.5000 - val_loss: 2.3713 - val_accuracy: 0.5000
Epoch 5/20
1/1 [=====] - 42s 42s/step - loss: 2.3824 - accuracy: 0.5000 - val_loss: 29.1493 - val_accuracy: 0.5000
Epoch 6/20
1/1 [=====] - 30s 30s/step - loss: 26.5169 - accuracy: 0.5000 - val_loss: 25.3417 - val_accuracy: 0.5000
Epoch 7/20
1/1 [=====] - 37s 37s/step - loss: 23.0344 - accuracy: 0.5000 - val_loss: 16.4911 - val_accuracy: 0.5000
Epoch 8/20
1/1 [=====] - 54s 54s/step - loss: 14.9720 - accuracy: 0.5000 - val_loss: 7.8888 - val_accuracy: 0.5000
Epoch 9/20
1/1 [=====] - 51s 51s/step - loss: 7.1580 - accuracy: 0.5000 - val_loss: 6.7961 - val_accuracy: 0.5000
Epoch 10/20
1/1 [=====] - 39s 39s/step - loss: 6.1592 - accuracy: 0.5000 - val_loss: 5.7416 - val_accuracy: 0.5000
Epoch 11/20
1/1 [=====] - 36s 36s/step - loss: 5.1975 - accuracy: 0.5000 - val_loss: 4.6913 - val_accuracy: 0.5000
Epoch 12/20
1/1 [=====] - 61s 61s/step - loss: 4.2411 - accuracy: 0.5000 - val_loss: 3.6642 - val_accuracy: 0.5000
Epoch 13/20
1/1 [=====] - 28s 28s/step - loss: 3.3076 - accuracy: 0.5000 - val_loss: 2.3418 - val_accuracy: 0.5000
Epoch 14/20
1/1 [=====] - 46s 46s/step - loss: 2.1116 - accuracy: 0.5000 - val_loss: 1.1933 - val_accuracy: 0.5000
Epoch 15/20
1/1 [=====] - 51s 51s/step - loss: 1.0921 - accuracy: 0.5000 - val_loss: 0.6918 - val_accuracy: 0.5000
Epoch 16/20
1/1 [=====] - 36s 36s/step - loss: 0.6797 - accuracy: 0.5547 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 17/20
1/1 [=====] - 21s 21s/step - loss: 0.6932 - accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
```

### Lampiran 3. Hasil Prediksi 2 Convolutional Kernel 1 Convolutional Layer

```
Epoch 18/20
1/1 [=====] - 20s 20s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 19/20
1/1 [=====] - 23s 23s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 20/20
1/1 [=====] - 23s 23s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
```

### Lampiran 4. Syntax Prediksi 2 Convolutional Kernel 2 Convolutional Layer

```
model = models.Sequential()
model.add(layers.Conv2D(32, (2, 2), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (2, 2), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
"Fitting the data into our model"
#es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(train_images, np.asarray(train_labels), epochs = 20,
                    validation_data =(test_images, np.asarray(test_labels)),
                    batch_size = 200)
```

## Lampiran 5. Hasil Prediksi 2 Convolutional Kernel 2 Convolutional Layer

```
Epoch 1/20
1/1 [=====] - 118s 118s/step - loss: 0.6934 -
accuracy: 0.5000 - val_loss: 2.0096 - val_accuracy: 0.5000
Epoch 2/20
1/1 [=====] - 71s 71s/step - loss: 2.0626 -
accuracy: 0.5000 - val_loss: 0.7586 - val_accuracy: 0.5000
Epoch 3/20
1/1 [=====] - 51s 51s/step - loss: 0.7714 -
accuracy: 0.5000 - val_loss: 2.7303 - val_accuracy: 0.5000
Epoch 4/20
1/1 [=====] - 68s 68s/step - loss: 2.4856 -
accuracy: 0.5000 - val_loss: 0.9468 - val_accuracy: 0.5000
Epoch 5/20
1/1 [=====] - 59s 59s/step - loss: 0.8891 -
accuracy: 0.5000 - val_loss: 1.5446 - val_accuracy: 0.5000
Epoch 6/20
1/1 [=====] - 37s 37s/step - loss: 1.5870 -
accuracy: 0.5000 - val_loss: 1.3341 - val_accuracy: 0.5000
Epoch 7/20
1/1 [=====] - 50s 50s/step - loss: 1.3646 -
accuracy: 0.5000 - val_loss: 0.7329 - val_accuracy: 0.5000
Epoch 8/20
1/1 [=====] - 62s 62s/step - loss: 0.7280 -
accuracy: 0.5000 - val_loss: 0.8270 - val_accuracy: 0.5000
Epoch 9/20
1/1 [=====] - 52s 52s/step - loss: 0.7714 -
accuracy: 0.5000 - val_loss: 1.0634 - val_accuracy: 0.5000
Epoch 10/20
1/1 [=====] - 51s 51s/step - loss: 0.9713 -
accuracy: 0.5000 - val_loss: 0.7482 - val_accuracy: 0.5000
Epoch 11/20
1/1 [=====] - 60s 60s/step - loss: 0.6926 -
accuracy: 0.5000 - val_loss: 0.6880 - val_accuracy: 0.5000
Epoch 12/20
1/1 [=====] - 55s 55s/step - loss: 0.6382 -
accuracy: 0.5312 - val_loss: 0.6876 - val_accuracy: 0.5000
Epoch 13/20
1/1 [=====] - 64s 64s/step - loss: 0.6236 -
accuracy: 0.5547 - val_loss: 0.6181 - val_accuracy: 0.8125
Epoch 14/20
1/1 [=====] - 57s 57s/step - loss: 0.5535 -
```

## Lampiran 5. Hasil Prediksi 2 Convolutional Kernel 2 Convolutional Layer

```
accuracy: 0.8203 - val_loss: 0.6107 - val_accuracy: 0.6562
Epoch 15/20
1/1 [=====] - 66s 66s/step - loss: 0.5391 -
accuracy: 0.6406 - val_loss: 0.6176 - val_accuracy: 0.5000
Epoch 16/20
1/1 [=====] - 49s 49s/step - loss: 0.5078 -
accuracy: 0.6484 - val_loss: 0.5676 - val_accuracy: 0.9062
Epoch 17/20
1/1 [=====] - 52s 52s/step - loss: 0.4655 -
accuracy: 0.9766 - val_loss: 0.5385 - val_accuracy: 0.7188
Epoch 18/20
1/1 [=====] - 63s 63s/step - loss: 0.4547 -
accuracy: 0.7422 - val_loss: 0.6189 - val_accuracy: 0.5000
Epoch 19/20
1/1 [=====] - 65s 65s/step - loss: 0.4527 -
accuracy: 0.6797 - val_loss: 0.5128 - val_accuracy: 0.8750
Epoch 20/20
1/1 [=====] - 60s 60s/step - loss: 0.4065 -
accuracy: 0.9375 - val_loss: 0.4984 - val_accuracy: 0.8125
```

## Lampiran 6. Syntax Prediksi 2 Convolutional Kernel 3 Convolutional Layer

```
model = models.Sequential()
model.add(layers.Conv2D(32, (2, 2), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (2, 2), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (2, 2), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
"Fitting the data into our model"
#es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(train_images, np.asarray(train_labels), epochs = 20,
                    validation_data =(test_images, np.asarray(test_labels)),
                    batch_size = 200)
```



## Lampiran 7. Hasil Prediksi 2 Convolutional Kernel 3 Convolutional Layer

```
Epoch 1/20
1/1 [=====] - 118s 118s/step - loss: 0.6934 -
accuracy: 0.5000 - val_loss: 2.0096 - val_accuracy: 0.5000
Epoch 2/20
1/1 [=====] - 71s 71s/step - loss: 2.0626 -
accuracy: 0.5000 - val_loss: 0.7586 - val_accuracy: 0.5000
Epoch 3/20
1/1 [=====] - 51s 51s/step - loss: 0.7714 -
accuracy: 0.5000 - val_loss: 2.7303 - val_accuracy: 0.5000
Epoch 4/20
1/1 [=====] - 68s 68s/step - loss: 2.4856 -
accuracy: 0.5000 - val_loss: 0.9468 - val_accuracy: 0.5000
Epoch 5/20
1/1 [=====] - 59s 59s/step - loss: 0.8891 -
accuracy: 0.5000 - val_loss: 1.5446 - val_accuracy: 0.5000
Epoch 6/20
1/1 [=====] - 37s 37s/step - loss: 1.5870 -
accuracy: 0.5000 - val_loss: 1.3341 - val_accuracy: 0.5000
Epoch 7/20
1/1 [=====] - 50s 50s/step - loss: 1.3646 -
accuracy: 0.5000 - val_loss: 0.7329 - val_accuracy: 0.5000
Epoch 8/20
1/1 [=====] - 62s 62s/step - loss: 0.7280 -
accuracy: 0.5000 - val_loss: 0.8270 - val_accuracy: 0.5000
Epoch 9/20
1/1 [=====] - 52s 52s/step - loss: 0.7714 -
accuracy: 0.5000 - val_loss: 1.0634 - val_accuracy: 0.5000
Epoch 10/20
1/1 [=====] - 51s 51s/step - loss: 0.9713 -
accuracy: 0.5000 - val_loss: 0.7482 - val_accuracy: 0.5000
Epoch 11/20
1/1 [=====] - 60s 60s/step - loss: 0.6926 -
accuracy: 0.5000 - val_loss: 0.6880 - val_accuracy: 0.5000
Epoch 12/20
1/1 [=====] - 55s 55s/step - loss: 0.6382 -
accuracy: 0.5312 - val_loss: 0.6876 - val_accuracy: 0.5000
Epoch 13/20
1/1 [=====] - 64s 64s/step - loss: 0.6236 -
accuracy: 0.5547 - val_loss: 0.6181 - val_accuracy: 0.8125
Epoch 14/20
1/1 [=====] - 57s 57s/step - loss: 0.5535 -
accuracy: 0.8203 - val_loss: 0.6107 - val_accuracy: 0.6562
Epoch 15/20
1/1 [=====] - 66s 66s/step - loss: 0.5391 -
accuracy: 0.6406 - val_loss: 0.6176 - val_accuracy: 0.5000
Epoch 16/20
1/1 [=====] - 49s 49s/step - loss: 0.5078 -
accuracy: 0.6484 - val_loss: 0.5676 - val_accuracy: 0.9062
Epoch 17/20
1/1 [=====] - 52s 52s/step - loss: 0.4655 -
accuracy: 0.9766 - val_loss: 0.5385 - val_accuracy: 0.7188
Epoch 18/20
1/1 [=====] - 63s 63s/step - loss: 0.4547 -
accuracy: 0.7422 - val_loss: 0.6189 - val_accuracy: 0.5000
```

## Lampiran 7. Hasil Prediksi 2 Convolutional Kernel 3 Convolutional Layer

```
Epoch 19/20
1/1 [=====] - 65s 65s/step - loss: 0.4527 -
accuracy: 0.6797 - val_loss: 0.5128 - val_accuracy: 0.8750
Epoch 20/20
1/1 [=====] - 60s 60s/step - loss: 0.4065 -
accuracy: 0.9375 - val_loss: 0.4984 - val_accuracy: 0.8125
```

## Lampiran 8. Syntax Prediksi 3 Convolutional Kernel 1 Convolutional Layer

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
"Fitting the data into our model"
#es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(train_images, np.asarray(train_labels), epochs = 20,
                    validation_data=(test_images, np.asarray(test_labels)),
                    batch_size = 200)
```

## Lampiran 9. Hasil Prediksi 3 Convolutional Kernel 1 Convolutional Layer

```
Epoch 1/20
1/1 [=====] - 51s 51s/step - loss: 0.6935 -
accuracy: 0.5000 - val_loss: 28.2534 - val_accuracy: 0.5000
Epoch 2/20
1/1 [=====] - 24s 24s/step - loss: 29.1331 -
accuracy: 0.5000 - val_loss: 1.1582 - val_accuracy: 0.5000
Epoch 3/20
1/1 [=====] - 42s 42s/step - loss: 1.2005 -
accuracy: 0.5000 - val_loss: 2.1637 - val_accuracy: 0.5000
Epoch 4/20
1/1 [=====] - 48s 48s/step - loss: 2.2409 -
accuracy: 0.5000 - val_loss: 3.8782 - val_accuracy: 0.5000
Epoch 5/20
1/1 [=====] - 33s 33s/step - loss: 3.5311 -
accuracy: 0.5000 - val_loss: 0.7031 - val_accuracy: 0.5000
Epoch 6/20
1/1 [=====] - 47s 47s/step - loss: 0.7066 -
accuracy: 0.5000 - val_loss: 1.6243 - val_accuracy: 0.5000
Epoch 7/20
1/1 [=====] - 37s 37s/step - loss: 1.6535 -
accuracy: 0.5000 - val_loss: 3.1609 - val_accuracy: 0.5000
Epoch 8/20
1/1 [=====] - 48s 48s/step - loss: 2.8641 -
accuracy: 0.5000 - val loss: 1.7878 - val accuracy: 0.5000
```

## Lampiran 9. Hasil Prediksi 3 Convolutional Kernel 1 Convolutional Layer

```
Epoch 9/20
1/1 [=====] - 29s 29s/step - loss: 1.6130 -
accuracy: 0.5000 - val_loss: 3.0345 - val_accuracy: 0.5000
Epoch 10/20
1/1 [=====] - 45s 45s/step - loss: 3.0257 -
accuracy: 0.5000 - val_loss: 3.0792 - val_accuracy: 0.5000
Epoch 11/20
1/1 [=====] - 46s 46s/step - loss: 3.0309 -
accuracy: 0.5000 - val_loss: 0.5997 - val_accuracy: 0.6562
Epoch 12/20
1/1 [=====] - 38s 38s/step - loss: 0.5298 -
accuracy: 0.6406 - val_loss: 2.7834 - val_accuracy: 0.5000
Epoch 13/20
1/1 [=====] - 29s 29s/step - loss: 2.5077 -
accuracy: 0.5000 - val_loss: 1.3129 - val_accuracy: 0.5000
Epoch 14/20
1/1 [=====] - 32s 32s/step - loss: 1.1462 -
accuracy: 0.5078 - val_loss: 1.6387 - val_accuracy: 0.5000
Epoch 15/20
1/1 [=====] - 26s 26s/step - loss: 1.3394 -
accuracy: 0.5000 - val_loss: 1.9746 - val_accuracy: 0.5000
Epoch 16/20
1/1 [=====] - 30s 30s/step - loss: 1.6291 -
accuracy: 0.5000 - val_loss: 0.6334 - val_accuracy: 0.7812
Epoch 17/20
1/1 [=====] - 44s 44s/step - loss: 0.4361 -
accuracy: 0.9062 - val_loss: 1.1673 - val_accuracy: 0.5000
Epoch 18/20
1/1 [=====] - 46s 46s/step - loss: 1.0374 -
accuracy: 0.5781 - val_loss: 1.1590 - val_accuracy: 0.5312
Epoch 19/20
1/1 [=====] - 30s 30s/step - loss: 1.0342 -
accuracy: 0.5938 - val_loss: 0.5237 - val_accuracy: 0.6875
Epoch 20/20
1/1 [=====] - 32s 32s/step - loss: 0.4046 -
accuracy: 0.7188 - val loss: 1.4467 - val accuracy: 0.5000
```

## Lampiran 10. Syntax Prediksi 3 Convolutional Kernel 2 Convolutional Layer

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(IMG_SIZE, IMG_SIZE, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
"Fitting the data into our model"
#es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(train_images, np.asarray(train_labels), epochs = 20,
                    validation_data =(test_images, np.asarray(test_labels)),
                    batch_size = 200)
```

## Lampiran 11. Hasil Prediksi 3 Convolutional Kernel 2 Convolutional Layer

```
Epoch 1/20
1/1 [=====] - 77s 77s/step - loss: 0.6987 -
accuracy: 0.5000 - val_loss: 8.1370 - val_accuracy: 0.5000
Epoch 2/20
1/1 [=====] - 58s 58s/step - loss: 8.4450 -
accuracy: 0.5000 - val_loss: 2.5592 - val_accuracy: 0.5000
Epoch 3/20
1/1 [=====] - 54s 54s/step - loss: 2.6585 -
accuracy: 0.5000 - val_loss: 2.3322 - val_accuracy: 0.5000
Epoch 4/20
1/1 [=====] - 56s 56s/step - loss: 2.4264 -
accuracy: 0.5000 - val_loss: 1.8587 - val_accuracy: 0.5000
Epoch 5/20
1/1 [=====] - 59s 59s/step - loss: 1.9345 -
accuracy: 0.5000 - val_loss: 1.2960 - val_accuracy: 0.5000
Epoch 6/20
1/1 [=====] - 60s 60s/step - loss: 1.3475 -
accuracy: 0.5000 - val_loss: 0.8429 - val_accuracy: 0.5000
Epoch 7/20
1/1 [=====] - 55s 55s/step - loss: 0.8763 -
accuracy: 0.5000 - val_loss: 0.7159 - val_accuracy: 0.5000
Epoch 8/20
1/1 [=====] - 54s 54s/step - loss: 0.7351 -
accuracy: 0.5000 - val_loss: 0.6902 - val_accuracy: 0.5000
Epoch 9/20
1/1 [=====] - 56s 56s/step - loss: 0.6965 -
accuracy: 0.5000 - val_loss: 0.6898 - val_accuracy: 0.5000
Epoch 10/20
1/1 [=====] - 56s 56s/step - loss: 0.6967 -
accuracy: 0.5000 - val_loss: 0.6880 - val_accuracy: 0.5000
Epoch 11/20
1/1 [=====] - 46s 46s/step - loss: 0.6936 -
accuracy: 0.5000 - val_loss: 0.6873 - val_accuracy: 0.5000
```

## Lampiran 11. Hasil Prediksi 3 Convolutional Kernel 2 Convolutional Layer

```
Epoch 12/20
1/1 [=====] - 56s 56s/step - loss: 0.6919 -
accuracy: 0.5000 - val_loss: 0.6864 - val_accuracy: 0.5000
Epoch 13/20
1/1 [=====] - 61s 61s/step - loss: 0.6899 -
accuracy: 0.5000 - val_loss: 0.6858 - val_accuracy: 0.5000
Epoch 14/20
1/1 [=====] - 58s 58s/step - loss: 0.6884 -
accuracy: 0.5000 - val_loss: 0.6846 - val_accuracy: 0.5000
Epoch 15/20
1/1 [=====] - 63s 63s/step - loss: 0.6868 -
accuracy: 0.5000 - val_loss: 0.6831 - val_accuracy: 0.5000
Epoch 16/20
1/1 [=====] - 63s 63s/step - loss: 0.6849 -
accuracy: 0.5000 - val_loss: 0.6821 - val_accuracy: 0.5000
Epoch 17/20
1/1 [=====] - 58s 58s/step - loss: 0.6835 -
accuracy: 0.5000 - val_loss: 0.6805 - val_accuracy: 0.5000
Epoch 18/20
1/1 [=====] - 55s 55s/step - loss: 0.6816 -
accuracy: 0.5000 - val_loss: 0.6785 - val_accuracy: 0.5000
Epoch 19/20
1/1 [=====] - 50s 50s/step - loss: 0.6790 -
accuracy: 0.5000 - val_loss: 0.6762 - val_accuracy: 0.5000
Epoch 20/20
1/1 [=====] - 43s 43s/step - loss: 0.6756 -
accuracy: 0.5156 - val_loss: 0.6736 - val_accuracy: 0.7500
```

## Lampiran 12. Syntax Prediksi 3 Convolutional Kernel 3 Convolutional Layer

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
"Fitting the data into our model"
#es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(train_images, np.asarray(train_labels), epochs = 20,
                    validation_data =(test_images, np.asarray(test_labels)),
                    batch_size = 200)
```

### Lampiran 13. Hasil Prediksi 3 Convolutional Kernel 3 Convolutional Layer

```
Epoch 1/20
1/1 [=====] - 49s 49s/step - loss: 0.6948 -
accuracy: 0.5000 - val_loss: 0.7809 - val_accuracy: 0.5000
Epoch 2/20
1/1 [=====] - 42s 42s/step - loss: 0.8086 -
accuracy: 0.5000 - val_loss: 0.7249 - val_accuracy: 0.5000
Epoch 3/20
1/1 [=====] - 48s 48s/step - loss: 0.7450 -
accuracy: 0.5000 - val_loss: 0.6952 - val_accuracy: 0.5000
Epoch 4/20
1/1 [=====] - 44s 44s/step - loss: 0.7057 -
accuracy: 0.5000 - val_loss: 0.6909 - val_accuracy: 0.5000
Epoch 5/20
1/1 [=====] - 48s 48s/step - loss: 0.6938 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 6/20
1/1 [=====] - 32s 32s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 7/20
1/1 [=====] - 34s 34s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 8/20
1/1 [=====] - 36s 36s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6935 - val_accuracy: 0.5000
Epoch 9/20
1/1 [=====] - 51s 51s/step - loss: 0.6926 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 10/20
1/1 [=====] - 31s 31s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 11/20
1/1 [=====] - 31s 31s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6915 - val_accuracy: 0.5938
Epoch 12/20
1/1 [=====] - 37s 37s/step - loss: 0.6913 -
accuracy: 0.6328 - val_loss: 0.6889 - val_accuracy: 0.5000
Epoch 13/20
1/1 [=====] - 38s 38s/step - loss: 0.6926 -
accuracy: 0.5000 - val_loss: 0.6898 - val_accuracy: 0.5000
Epoch 14/20
1/1 [=====] - 35s 35s/step - loss: 0.6959 -
accuracy: 0.5000 - val_loss: 0.6878 - val_accuracy: 0.5000
Epoch 15/20
1/1 [=====] - 46s 46s/step - loss: 0.6933 -
accuracy: 0.5000 - val_loss: 0.6840 - val_accuracy: 0.5000
Epoch 16/20
1/1 [=====] - 56s 56s/step - loss: 0.6856 -
accuracy: 0.5000 - val_loss: 0.6781 - val_accuracy: 0.5000
Epoch 17/20
1/1 [=====] - 49s 49s/step - loss: 0.6770 -
accuracy: 0.5000 - val_loss: 0.6761 - val_accuracy: 0.5000
Epoch 18/20
1/1 [=====] - 47s 47s/step - loss: 0.6726 -
accuracy: 0.5078 - val_loss: 0.6742 - val_accuracy: 0.5000
```

### Lampiran 13. Hasil Prediksi 3 Convolutional Kernel 3 Convolutional Layer

```
Epoch 19/20
1/1 [=====] - 49s 49s/step - loss: 0.6682 -
accuracy: 0.6016 - val_loss: 0.6654 - val_accuracy: 0.8125
Epoch 20/20
1/1 [=====] - 37s 37s/step - loss: 0.6607 -
accuracy: 0.8750 - val_loss: 0.6553 - val_accuracy: 0.6875
```

### Lampiran 14. Syntax Prediksi 4 Convolutional Kernel 1 Convolutional layer

```
model = models.Sequential()
model.add(layers.Conv2D(32, (4, 4), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
"Fitting the data into our model"
#es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(train_images, np.asarray(train_labels), epochs = 20,
                    validation_data=(test_images, np.asarray(test_labels)),
                    batch_size = 200)
```

### Lampiran 15. Hasil Prediksi 4 Convolutional Kernel 1 Convolutional Layer

```
Epoch 1/20
1/1 [=====] - 54s 54s/step - loss: 0.6929 -
accuracy: 0.5000 - val_loss: 87.7366 - val_accuracy: 0.5000
Epoch 2/20
1/1 [=====] - 20s 20s/step - loss: 90.3388 -
accuracy: 0.5000 - val_loss: 25.8987 - val_accuracy: 0.5000
Epoch 3/20
1/1 [=====] - 30s 30s/step - loss: 23.5370 -
accuracy: 0.5000 - val_loss: 16.3091 - val_accuracy: 0.5000
Epoch 4/20
1/1 [=====] - 39s 39s/step - loss: 14.8197 -
accuracy: 0.5000 - val_loss: 13.9189 - val_accuracy: 0.5000
Epoch 5/20
1/1 [=====] - 53s 53s/step - loss: 14.2911 -
accuracy: 0.5000 - val_loss: 11.9844 - val_accuracy: 0.5000
Epoch 6/20
1/1 [=====] - 47s 47s/step - loss: 12.2313 -
accuracy: 0.5000 - val_loss: 3.3047 - val_accuracy: 0.5000
Epoch 7/20
1/1 [=====] - 40s 40s/step - loss: 3.1855 -
accuracy: 0.5000 - val_loss: 10.4352 - val_accuracy: 0.5000
Epoch 8/20
1/1 [=====] - 35s 35s/step - loss: 9.4215 -
accuracy: 0.5000 - val_loss: 13.1943 - val_accuracy: 0.5000
```

## Lampiran 15 Hasil Prediksi 4 Convolutional Kernel 1 Convolutional Layer

```
Epoch 9/20
1/1 [=====] - 43s 43s/step - loss: 11.9116 -
accuracy: 0.5000 - val_loss: 10.2125 - val_accuracy: 0.5000
Epoch 10/20
1/1 [=====] - 43s 43s/step - loss: 9.1776 -
accuracy: 0.5000 - val_loss: 3.4707 - val_accuracy: 0.5000
Epoch 11/20
1/1 [=====] - 39s 39s/step - loss: 3.0232 -
accuracy: 0.5000 - val_loss: 5.7315 - val_accuracy: 0.5000
Epoch 12/20
1/1 [=====] - 59s 59s/step - loss: 5.4327 -
accuracy: 0.5000 - val_loss: 7.3649 - val_accuracy: 0.5000
Epoch 13/20
1/1 [=====] - 45s 45s/step - loss: 7.0505 -
accuracy: 0.5000 - val_loss: 6.6627 - val_accuracy: 0.5000
Epoch 14/20
1/1 [=====] - 54s 54s/step - loss: 6.2661 -
accuracy: 0.5000 - val_loss: 4.0894 - val_accuracy: 0.5000
Epoch 15/20
1/1 [=====] - 49s 49s/step - loss: 3.5459 -
accuracy: 0.5000 - val_loss: 0.7455 - val_accuracy: 0.8438
Epoch 16/20
1/1 [=====] - 56s 56s/step - loss: 0.4259 -
accuracy: 0.9219 - val_loss: 2.4410 - val_accuracy: 0.5000
Epoch 17/20
1/1 [=====] - 42s 42s/step - loss: 2.2016 -
accuracy: 0.5469 - val_loss: 3.5581 - val_accuracy: 0.5000
Epoch 18/20
1/1 [=====] - 23s 23s/step - loss: 3.0841 -
accuracy: 0.5078 - val_loss: 3.0067 - val_accuracy: 0.5000
Epoch 19/20
1/1 [=====] - 22s 22s/step - loss: 2.5749 -
accuracy: 0.5312 - val_loss: 1.4148 - val_accuracy: 0.5625
Epoch 20/20
1/1 [=====] - 31s 31s/step - loss: 1.3642 -
accuracy: 0.6172 - val_loss: 0.5587 - val_accuracy: 0.7812
```



## Lampiran 16. Syntax Prediksi 4 Convolutional Kernel 2 Convolutional Layer

```
model = models.Sequential()
model.add(layers.Conv2D(32, (4, 4), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (4, 4), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
"Fitting the data into our model"
#es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(train_images, np.asarray(train_labels), epochs = 20,
                    validation_data =(test_images, np.asarray(test_labels)),
                    batch_size = 200)
```

## Lampiran 17. Hasil Prediksi 4 Convolutional Kernel 2 Convolutional Layer

```
Epoch 1/20
1/1 [=====] - 118s 118s/step - loss: 0.6933
- accuracy: 0.5000 - val_loss: 6.7803 - val_accuracy: 0.5000
Epoch 2/20
1/1 [=====] - 75s 75s/step - loss: 6.2024 -
accuracy: 0.5000 - val_loss: 6.5585 - val_accuracy: 0.5000
Epoch 3/20
1/1 [=====] - 76s 76s/step - loss: 6.7618 -
accuracy: 0.5000 - val_loss: 0.9383 - val_accuracy: 0.5000
Epoch 4/20
1/1 [=====] - 79s 79s/step - loss: 0.9243 -
accuracy: 0.5000 - val_loss: 0.8977 - val_accuracy: 0.5000
Epoch 5/20
1/1 [=====] - 75s 75s/step - loss: 0.8259 -
accuracy: 0.5000 - val_loss: 0.6192 - val_accuracy: 0.5312
Epoch 6/20
1/1 [=====] - 77s 77s/step - loss: 0.5744 -
accuracy: 0.7344 - val_loss: 0.7534 - val_accuracy: 0.5000
Epoch 7/20
1/1 [=====] - 80s 80s/step - loss: 0.5463 -
accuracy: 0.6094 - val_loss: 0.6598 - val_accuracy: 0.8750
Epoch 8/20
1/1 [=====] - 72s 72s/step - loss: 0.4528 -
accuracy: 0.9375 - val_loss: 0.6161 - val_accuracy: 0.6875
Epoch 9/20
1/1 [=====] - 67s 67s/step - loss: 0.4213 -
accuracy: 0.8125 - val_loss: 1.2498 - val_accuracy: 0.5000
Epoch 10/20
1/1 [=====] - 61s 61s/step - loss: 0.6413 -
accuracy: 0.6172 - val_loss: 1.5342 - val_accuracy: 0.5000
Epoch 11/20
1/1 [=====] - 75s 75s/step - loss: 0.4105 -
accuracy: 0.9375 - val_loss: 0.6161 - val_accuracy: 0.6875
```

## Lampiran 17. Hasil Prediksi 4 Convolutional Kernel 2 Convolutional Layer

```
Epoch 11/20
1/1 [=====] - 75s 75s/step - loss: 1.4125 -
accuracy: 0.5312 - val_loss: 0.6163 - val_accuracy: 0.6875
Epoch 12/20
1/1 [=====] - 66s 66s/step - loss: 0.5071 -
accuracy: 0.6484 - val_loss: 0.6884 - val_accuracy: 0.5000
Epoch 13/20
1/1 [=====] - 49s 49s/step - loss: 0.4619 -
accuracy: 0.7188 - val_loss: 0.6970 - val_accuracy: 0.5000
Epoch 14/20
1/1 [=====] - 61s 61s/step - loss: 0.4883 -
accuracy: 0.6719 - val_loss: 0.6781 - val_accuracy: 0.5625
Epoch 15/20
1/1 [=====] - 52s 52s/step - loss: 0.4739 -
accuracy: 0.7344 - val_loss: 0.9357 - val_accuracy: 0.5000
Epoch 16/20
1/1 [=====] - 65s 65s/step - loss: 0.6124 -
accuracy: 0.5938 - val_loss: 0.6678 - val_accuracy: 0.8750
Epoch 17/20
1/1 [=====] - 48s 48s/step - loss: 0.4745 -
accuracy: 0.9766 - val_loss: 0.6713 - val_accuracy: 0.8438
Epoch 18/20
1/1 [=====] - 46s 46s/step - loss: 0.5009 -
accuracy: 0.9922 - val_loss: 0.6795 - val_accuracy: 0.8438
Epoch 19/20
1/1 [=====] - 66s 66s/step - loss: 0.4716 -
accuracy: 0.9688 - val_loss: 0.6511 - val_accuracy: 0.6562
Epoch 20/20
1/1 [=====] - 66s 66s/step - loss: 0.4906 -
accuracy: 0.6406 - val_loss: 0.8946 - val_accuracy: 0.5000
```

## Lampiran 18. Syntax Prediksi 4 Convolutional Kernel 3 Convolutional Layer

```
model = models.Sequential()
model.add(layers.Conv2D(32, (4, 4), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (4, 4), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (4, 4), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
"Fitting the data into our model"
#es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(train_images, np.asarray(train_labels), epochs = 20,
                    validation_data =(test_images, np.asarray(test_labels)),
                    batch_size = 200)
```

## Lampiran 19. Hasil Prediksi 4 Convolutional Kernel 3 Convolutional Layer

```
Epoch 1/20
1/1 [=====] - 73s 73s/step - loss: 0.6950 -
accuracy: 0.5000 - val_loss: 5.8158 - val_accuracy: 0.5000
Epoch 2/20
1/1 [=====] - 76s 76s/step - loss: 6.0591 -
accuracy: 0.5000 - val_loss: 0.6943 - val_accuracy: 0.5000
Epoch 3/20
1/1 [=====] - 65s 65s/step - loss: 0.6897 -
accuracy: 0.5000 - val_loss: 0.7261 - val_accuracy: 0.5000
Epoch 4/20
1/1 [=====] - 67s 67s/step - loss: 0.7112 -
accuracy: 0.5000 - val_loss: 0.6919 - val_accuracy: 0.5000
Epoch 5/20
1/1 [=====] - 78s 78s/step - loss: 0.6915 -
accuracy: 0.5156 - val_loss: 0.6947 - val_accuracy: 0.5000
Epoch 6/20
1/1 [=====] - 67s 67s/step - loss: 0.7005 -
accuracy: 0.5000 - val_loss: 0.6920 - val_accuracy: 0.5000
Epoch 7/20
1/1 [=====] - 62s 62s/step - loss: 0.6903 -
accuracy: 0.5391 - val_loss: 0.7368 - val_accuracy: 0.5000
Epoch 8/20
1/1 [=====] - 64s 64s/step - loss: 0.7184 -
accuracy: 0.5000 - val_loss: 0.7133 - val_accuracy: 0.5000
Epoch 9/20
1/1 [=====] - 63s 63s/step - loss: 0.7005 -
accuracy: 0.5000 - val_loss: 0.6893 - val_accuracy: 0.5938
```

## Lampiran 19. Hasil Prediksi 4 Convolutional Kernel 3 Convolutional Layer

```
Epoch 10/20
1/1 [=====] - 65s 65s/step - loss: 0.6864 -
accuracy: 0.6250 - val_loss: 0.6905 - val_accuracy: 0.5000
Epoch 11/20
1/1 [=====] - 69s 69s/step - loss: 0.6959 -
accuracy: 0.5000 - val_loss: 0.6853 - val_accuracy: 0.5000
Epoch 12/20
1/1 [=====] - 63s 63s/step - loss: 0.6885 -
accuracy: 0.5000 - val_loss: 0.7005 - val_accuracy: 0.5000
Epoch 13/20
1/1 [=====] - 45s 45s/step - loss: 0.6880 -
accuracy: 0.5000 - val_loss: 0.6973 - val_accuracy: 0.5000
Epoch 14/20
1/1 [=====] - 51s 51s/step - loss: 0.6842 -
accuracy: 0.5000 - val_loss: 0.6802 - val_accuracy: 0.7188
Epoch 15/20
1/1 [=====] - 71s 71s/step - loss: 0.6748 -
accuracy: 0.7266 - val_loss: 0.6772 - val_accuracy: 0.5000
Epoch 16/20
1/1 [=====] - 68s 68s/step - loss: 0.6749 -
accuracy: 0.5000 - val_loss: 0.6748 - val_accuracy: 0.5000
Epoch 17/20
1/1 [=====] - 80s 80s/step - loss: 0.6712 -
accuracy: 0.5078 - val_loss: 0.6698 - val_accuracy: 0.6875
Epoch 18/20
1/1 [=====] - 71s 71s/step - loss: 0.6603 -
accuracy: 0.6562 - val_loss: 0.6669 - val_accuracy: 0.6562
Epoch 19/20
1/1 [=====] - 64s 64s/step - loss: 0.6548 -
accuracy: 0.6250 - val_loss: 0.6607 - val_accuracy: 0.5000
Epoch 20/20
1/1 [=====] - 76s 76s/step - loss: 0.6480 -
accuracy: 0.5391 - val_loss: 0.6637 - val_accuracy: 0.5000
```

## Lampiran 20. Syntax Prediksi 5 Convolutional Kernel 1 Convolutional Layer

```
model = models.Sequential()
model.add(layers.Conv2D(32, (5, 5), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
"Fitting the data into our model"
#es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(train_images, np.asarray(train_labels), epochs = 20,
                    validation_data =(test_images, np.asarray(test_labels)),
                    batch_size = 200)
```

## Lampiran 21. Hasil Prediksi 5 Convolutional Kernel 1 Convolutional Layer

```
Epoch 1/20
1/1 [=====] - 73s 73s/step - loss: 0.7060 -
accuracy: 0.5000 - val_loss: 37.1617 - val_accuracy: 0.5000
Epoch 2/20
1/1 [=====] - 57s 57s/step - loss: 38.5539 -
accuracy: 0.5000 - val_loss: 11.7985 - val_accuracy: 0.5000
Epoch 3/20
1/1 [=====] - 42s 42s/step - loss: 10.7392 -
accuracy: 0.5000 - val_loss: 9.3325 - val_accuracy: 0.5000
Epoch 4/20
1/1 [=====] - 48s 48s/step - loss: 8.4848 -
accuracy: 0.5000 - val_loss: 1.0595 - val_accuracy: 0.5000
Epoch 5/20
1/1 [=====] - 44s 44s/step - loss: 1.0957 -
accuracy: 0.5000 - val_loss: 1.6493 - val_accuracy: 0.5000
Epoch 6/20
1/1 [=====] - 33s 33s/step - loss: 1.4986 -
accuracy: 0.5000 - val_loss: 9.6178 - val_accuracy: 0.5000
Epoch 7/20
1/1 [=====] - 79s 79s/step - loss: 9.9813 -
accuracy: 0.5000 - val_loss: 5.6838 - val_accuracy: 0.5000
Epoch 8/20
1/1 [=====] - 39s 39s/step - loss: 5.8590 -
accuracy: 0.5000 - val_loss: 1.1485 - val_accuracy: 0.5000
Epoch 9/20
1/1 [=====] - 41s 41s/step - loss: 1.1153 -
accuracy: 0.5000 - val_loss: 3.3682 - val_accuracy: 0.5000
Epoch 10/20
1/1 [=====] - 69s 69s/step - loss: 2.9971 -
accuracy: 0.5000 - val_loss: 3.1958 - val_accuracy: 0.5000
Epoch 11/20
1/1 [=====] - 56s 56s/step - loss: 2.8132 -
accuracy: 0.5000 - val_loss: 1.8281 - val_accuracy: 0.5000
```

## Lampiran 21. Hasil Prediksi 5 Convolutional Kernel 1 Convolutional Layer

```
Epoch 12/20
1/1 [=====] - 56s 56s/step - loss: 1.6406 -
accuracy: 0.5000 - val_loss: 1.2783 - val_accuracy: 0.5000
Epoch 13/20
1/1 [=====] - 45s 45s/step - loss: 1.0761 -
accuracy: 0.5391 - val_loss: 0.5426 - val_accuracy: 0.6875
Epoch 14/20
1/1 [=====] - 35s 35s/step - loss: 0.4809 -
accuracy: 0.6406 - val_loss: 1.8660 - val_accuracy: 0.5000
Epoch 15/20
1/1 [=====] - 29s 29s/step - loss: 1.3315 -
accuracy: 0.5078 - val_loss: 0.6627 - val_accuracy: 0.5312
Epoch 16/20
1/1 [=====] - 41s 41s/step - loss: 0.4331 -
accuracy: 0.7422 - val_loss: 0.6707 - val_accuracy: 0.6562
Epoch 17/20
1/1 [=====] - 46s 46s/step - loss: 0.6487 -
accuracy: 0.6406 - val_loss: 0.5409 - val_accuracy: 0.6875
Epoch 18/20
1/1 [=====] - 42s 42s/step - loss: 0.4704 -
accuracy: 0.6484 - val_loss: 0.5604 - val_accuracy: 0.8438
Epoch 19/20
1/1 [=====] - 34s 34s/step - loss: 0.3896 -
accuracy: 0.9453 - val_loss: 0.6925 - val_accuracy: 0.5000
Epoch 20/20
1/1 [=====] - 51s 51s/step - loss: 0.4236 -
accuracy: 0.7031 - val_loss: 0.6392 - val_accuracy: 0.5312
```

## Lampiran 22. Syntax Prediksi 5 Convolutional Kernel 2 Convolutional Layer

```
model = models.Sequential()
model.add(layers.Conv2D(32, (5, 5), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (5, 5), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
"Fitting the data into our model"
#es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(train_images, np.asarray(train_labels), epochs = 20,
                    validation_data =(test_images, np.asarray(test_labels)),
                    batch_size = 200)
```

### Lampiran 23. Hasil Prediksi 5 Convolutional Kernel 2 Convolutional Layer

```
Epoch 1/20
1/1 [=====] - 79s 79s/step - loss: 0.6967 -
accuracy: 0.5000 - val_loss: 1.8329 - val_accuracy: 0.5000
Epoch 2/20
1/1 [=====] - 95s 95s/step - loss: 1.6779 -
accuracy: 0.5000 - val_loss: 0.9053 - val_accuracy: 0.5000
Epoch 3/20
1/1 [=====] - 95s 95s/step - loss: 0.8595 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 4/20
1/1 [=====] - 55s 55s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 5/20
1/1 [=====] - 62s 62s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 6/20
1/1 [=====] - 61s 61s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 7/20
1/1 [=====] - 68s 68s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 8/20
1/1 [=====] - 61s 61s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 9/20
1/1 [=====] - 59s 59s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 10/20
1/1 [=====] - 55s 55s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 11/20
1/1 [=====] - 45s 45s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 12/20
1/1 [=====] - 42s 42s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 13/20
1/1 [=====] - 42s 42s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 14/20
1/1 [=====] - 53s 53s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 15/20
1/1 [=====] - 51s 51s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 16/20
1/1 [=====] - 58s 58s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 17/20
1/1 [=====] - 43s 43s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 18/20
1/1 [=====] - 44s 44s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 19/20
1/1 [=====] - 52s 52s/step - loss: 0.6932 -
```

### Lampiran 23. Hasil Prediksi 5 Convolutional Kernel 2 Convolutional Layer

```
Epoch 19/20
1/1 [=====] - 52s 52s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 20/20
1/1 [=====] - 59s 59s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
```

### Lampiran 24. Syntax Prediksi 5 Convolutional Kernel 3 Convolutional Layer

```
model = models.Sequential()
model.add(layers.Conv2D(32, (5, 5), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (5, 5), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (5, 5), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
"Fitting the data into our model"
#es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(train_images, np.asarray(train_labels), epochs = 20,
                    validation_data =(test_images, np.asarray(test_labels)),
                    batch_size = 200)
```



## Lampiran 25. Hasil Prediksi 5 Convolutional Kernel 3 Convolutional Layer

```
Epoch 1/20
1/1 [=====] - 99s 99s/step - loss: 0.6957 -
accuracy: 0.5000 - val_loss: 3.3610 - val_accuracy: 0.5000
Epoch 2/20
1/1 [=====] - 90s 90s/step - loss: 3.5086 -
accuracy: 0.5000 - val_loss: 0.6889 - val_accuracy: 0.5000
Epoch 3/20
1/1 [=====] - 86s 86s/step - loss: 0.6868 -
accuracy: 0.6484 - val_loss: 0.9594 - val_accuracy: 0.5000
Epoch 4/20
1/1 [=====] - 79s 79s/step - loss: 0.9062 -
accuracy: 0.5000 - val_loss: 0.6914 - val_accuracy: 0.5625
Epoch 5/20
1/1 [=====] - 83s 83s/step - loss: 0.6895 -
accuracy: 0.5859 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 6/20
1/1 [=====] - 82s 82s/step - loss: 0.7008 -
accuracy: 0.5000 - val_loss: 0.6974 - val_accuracy: 0.5000
Epoch 7/20
1/1 [=====] - 85s 85s/step - loss: 0.6868 -
accuracy: 0.5000 - val_loss: 0.8635 - val_accuracy: 0.5000
Epoch 8/20
1/1 [=====] - 85s 85s/step - loss: 0.8894 -
accuracy: 0.5000 - val_loss: 0.6976 - val_accuracy: 0.5000
Epoch 9/20
1/1 [=====] - 71s 71s/step - loss: 0.6897 -
accuracy: 0.5000 - val_loss: 0.6993 - val_accuracy: 0.5000
Epoch 10/20
1/1 [=====] - 88s 88s/step - loss: 0.6911 -
accuracy: 0.5000 - val_loss: 0.7317 - val_accuracy: 0.5000
Epoch 11/20
1/1 [=====] - 77s 77s/step - loss: 0.7440 -
accuracy: 0.5000 - val_loss: 0.6844 - val_accuracy: 0.7500
Epoch 12/20
1/1 [=====] - 69s 69s/step - loss: 0.6824 -
accuracy: 0.6797 - val_loss: 0.7195 - val_accuracy: 0.5000
Epoch 13/20
1/1 [=====] - 79s 79s/step - loss: 0.7050 -
accuracy: 0.5000 - val_loss: 0.6929 - val_accuracy: 0.5000
Epoch 14/20
1/1 [=====] - 81s 81s/step - loss: 0.6852 -
accuracy: 0.5000 - val_loss: 0.6827 - val_accuracy: 0.5000
Epoch 15/20
1/1 [=====] - 83s 83s/step - loss: 0.6849 -
accuracy: 0.5000 - val_loss: 0.6895 - val_accuracy: 0.5000
Epoch 16/20
1/1 [=====] - 78s 78s/step - loss: 0.6936 -
accuracy: 0.5000 - val_loss: 0.6749 - val_accuracy: 0.6875
Epoch 17/20
1/1 [=====] - 82s 82s/step - loss: 0.6703 -
accuracy: 0.6250 - val_loss: 0.6840 - val_accuracy: 0.5000
Epoch 18/20
1/1 [=====] - 83s 83s/step - loss: 0.6734 -
accuracy: 0.5156 - val_loss: 0.6757 - val_accuracy: 0.5312
Epoch 19/20
1/1 [=====] - 77s 77s/step - loss: 0.6666 -
```

## Lampiran 25. Hasil Prediksi 5 Convolutional Kernel 3 Convolutional Layer

```
Epoch 19/20
1/1 [=====] - 77s 77s/step - loss: 0.6666 -
accuracy: 0.5703 - val_loss: 0.6675 - val_accuracy: 0.7500
Epoch 20/20
1/1 [=====] - 79s 79s/step - loss: 0.6631 -
accuracy: 0.8047 - val_loss: 0.6697 - val_accuracy: 0.5000
```

## Lampiran 26. Syntax Prediksi 6 Convolutional Kernel 1 Convolutional Layer

```
model = models.Sequential()
model.add(layers.Conv2D(32, (6, 6), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
"Fitting the data into our model"
#es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(train_images, np.asarray(train_labels), epochs = 20,
                    validation_data=(test_images, np.asarray(test_labels)),
                    batch_size = 200)
```

## Lampiran 27. Hasil Prediksi 6 Convolutional Kernel 1 Convolutional Layer

```
Epoch 1/20
1/1 [=====] - 100s 100s/step - loss: 0.7065
- accuracy: 0.5000 - val_loss: 48.4477 - val_accuracy: 0.5000
Epoch 2/20
1/1 [=====] - 41s 41s/step - loss: 50.1927 -
accuracy: 0.5000 - val_loss: 14.9074 - val_accuracy: 0.5000
Epoch 3/20
1/1 [=====] - 46s 46s/step - loss: 15.4483 -
accuracy: 0.5000 - val_loss: 12.2558 - val_accuracy: 0.5000
Epoch 4/20
1/1 [=====] - 80s 80s/step - loss: 11.1603 -
accuracy: 0.5000 - val_loss: 11.7369 - val_accuracy: 0.5000
Epoch 5/20
1/1 [=====] - 44s 44s/step - loss: 10.6727 -
accuracy: 0.5000 - val_loss: 1.2018 - val_accuracy: 0.5000
Epoch 6/20
1/1 [=====] - 47s 47s/step - loss: 1.2105 -
accuracy: 0.5000 - val_loss: 1.5447 - val_accuracy: 0.5000
Epoch 7/20
1/1 [=====] - 52s 52s/step - loss: 1.5567 -
accuracy: 0.5000 - val_loss: 4.0178 - val_accuracy: 0.5000
Epoch 8/20
1/1 [=====] - 37s 37s/step - loss: 3.6165 -
accuracy: 0.5000 - val_loss: 1.7713 - val_accuracy: 0.5000
```

## Lampiran 27. Hasil Prediksi 6 Convolutional Kernel 1 Convolutional Layer

```
Epoch 9/20
1/1 [=====] - 46s 46s/step - loss: 1.5706 -
accuracy: 0.5000 - val_loss: 3.9613 - val_accuracy: 0.5000
Epoch 10/20
1/1 [=====] - 41s 41s/step - loss: 4.0043 -
accuracy: 0.5000 - val_loss: 3.8648 - val_accuracy: 0.5000
Epoch 11/20
1/1 [=====] - 43s 43s/step - loss: 3.8569 -
accuracy: 0.5000 - val_loss: 0.8986 - val_accuracy: 0.5000
Epoch 12/20
1/1 [=====] - 41s 41s/step - loss: 0.7472 -
accuracy: 0.5156 - val_loss: 3.1423 - val_accuracy: 0.5000
Epoch 13/20
1/1 [=====] - 55s 55s/step - loss: 2.7243 -
accuracy: 0.5000 - val_loss: 3.6557 - val_accuracy: 0.5000
Epoch 14/20
1/1 [=====] - 60s 60s/step - loss: 3.1720 -
accuracy: 0.5000 - val_loss: 1.3092 - val_accuracy: 0.5000
Epoch 15/20
1/1 [=====] - 83s 83s/step - loss: 1.1240 -
accuracy: 0.5547 - val_loss: 3.9413 - val_accuracy: 0.5000
Epoch 16/20
1/1 [=====] - 49s 49s/step - loss: 3.5133 -
accuracy: 0.5000 - val_loss: 2.0623 - val_accuracy: 0.5000
Epoch 17/20
1/1 [=====] - 39s 39s/step - loss: 1.5137 -
accuracy: 0.5000 - val_loss: 0.6670 - val_accuracy: 0.6562
Epoch 18/20
1/1 [=====] - 57s 57s/step - loss: 0.6659 -
accuracy: 0.6406 - val_loss: 0.5644 - val_accuracy: 0.6875
Epoch 19/20
1/1 [=====] - 56s 56s/step - loss: 0.5428 -
accuracy: 0.6406 - val_loss: 0.8586 - val_accuracy: 0.5000
Epoch 20/20
1/1 [=====] - 37s 37s/step - loss: 0.5119 -
accuracy: 0.6406 - val_loss: 0.6276 - val_accuracy: 0.7500
```

## Lampiran 28. Syntax Prediksi 6 Convolutional Kernel 2 Convolutional Layer

```
model = models.Sequential()
model.add(layers.Conv2D(32, (6, 6), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (6, 6), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
"Fitting the data into our model"
#es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(train_images, np.asarray(train_labels), epochs = 20,
                    validation_data=(test_images, np.asarray(test_labels)),
                    batch_size = 200)
```

## Lampiran 29. Syntax Prediksi 6 Convolutional Kernel 2 Convolutional Layer

```
Epoch 1/20
1/1 [=====] - 106s 106s/step - loss: 0.7012
- accuracy: 0.5000 - val_loss: 3.7223 - val_accuracy: 0.5000
Epoch 2/20
1/1 [=====] - 106s 106s/step - loss: 3.8646
- accuracy: 0.5000 - val_loss: 1.0588 - val_accuracy: 0.5000
Epoch 3/20
1/1 [=====] - 118s 118s/step - loss: 0.9891
- accuracy: 0.5000 - val_loss: 0.6969 - val_accuracy: 0.5000
Epoch 4/20
1/1 [=====] - 116s 116s/step - loss: 0.6922
- accuracy: 0.5000 - val_loss: 2.0167 - val_accuracy: 0.5000
Epoch 5/20
1/1 [=====] - 117s 117s/step - loss: 1.8483
- accuracy: 0.5000 - val_loss: 0.6965 - val_accuracy: 0.5000
Epoch 6/20
1/1 [=====] - 115s 115s/step - loss: 0.6949
- accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 7/20
1/1 [=====] - 85s 85s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 8/20
1/1 [=====] - 76s 76s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 9/20
1/1 [=====] - 83s 83s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 10/20
1/1 [=====] - 74s 74s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 11/20
```

## Lampiran 29. Syntax Prediksi 6 Convolutional Kernel 2 Convolutional Layer

```
Epoch 11/20
1/1 [=====] - 83s 83s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 12/20
1/1 [=====] - 74s 74s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 13/20
1/1 [=====] - 84s 84s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 14/20
1/1 [=====] - 65s 65s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 15/20
1/1 [=====] - 51s 51s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 16/20
1/1 [=====] - 71s 71s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 17/20
1/1 [=====] - 83s 83s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 18/20
1/1 [=====] - 83s 83s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 19/20
1/1 [=====] - 80s 80s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 20/20
1/1 [=====] - 82s 82s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
```

### Lampiran 30. Syntax Prediksi 6 Convolutional Kernel 3 Convolutional Layer

```
model = models.Sequential()
model.add(layers.Conv2D(32, (6, 6), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (6, 6), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (6, 6), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
"Fitting the data into our model"
#es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(train_images, np.asarray(train_labels), epochs = 20,
                    validation_data =(test_images, np.asarray(test_labels)),
                    batch_size = 200)
```

### Lampiran 31. Syntax Prediksi 6 Convolutional Kernel 3 Convolutional Layer

```
Epoch 1/20
1/1 [=====] - 94s 94s/step - loss: 0.6934 -
accuracy: 0.5234 - val_loss: 3.7132 - val_accuracy: 0.5000
Epoch 2/20
1/1 [=====] - 123s 123s/step - loss: 3.4117
- accuracy: 0.5000 - val_loss: 0.8269 - val_accuracy: 0.5000
Epoch 3/20
1/1 [=====] - 124s 124s/step - loss: 0.8481
- accuracy: 0.5000 - val_loss: 1.0250 - val_accuracy: 0.5000
Epoch 4/20
1/1 [=====] - 112s 112s/step - loss: 0.9627
- accuracy: 0.5000 - val_loss: 0.7204 - val_accuracy: 0.5000
Epoch 5/20
1/1 [=====] - 107s 107s/step - loss: 0.7058
- accuracy: 0.5000 - val_loss: 0.6976 - val_accuracy: 0.5000
Epoch 6/20
1/1 [=====] - 128s 128s/step - loss: 0.7074
- accuracy: 0.5000 - val_loss: 0.6948 - val_accuracy: 0.5000
Epoch 7/20
1/1 [=====] - 114s 114s/step - loss: 0.7033
- accuracy: 0.5000 - val_loss: 0.6908 - val_accuracy: 0.5000
Epoch 8/20
1/1 [=====] - 132s 132s/step - loss: 0.6954
- accuracy: 0.5000 - val_loss: 0.6918 - val_accuracy: 0.6562
Epoch 9/20
1/1 [=====] - 124s 124s/step - loss: 0.6910
- accuracy: 0.6250 - val_loss: 0.6996 - val_accuracy: 0.5000
Epoch 10/20
1/1 [=====] - 95s 95s/step - loss: 0.6935 -
accuracy: 0.5000 - val_loss: 0.7017 - val_accuracy: 0.5000
```

### Lampiran 31. Syntax Prediksi 6 Convolutional Kernel 3 Convolutional Layer

```
Epoch 10/20
1/1 [=====] - 95s 95s/step - loss: 0.6935 -
accuracy: 0.5000 - val_loss: 0.7017 - val_accuracy: 0.5000
Epoch 11/20
1/1 [=====] - 96s 96s/step - loss: 0.6935 -
accuracy: 0.5000 - val_loss: 0.6947 - val_accuracy: 0.5000
Epoch 12/20
1/1 [=====] - 104s 104s/step - loss: 0.6875
- accuracy: 0.5000 - val_loss: 0.6907 - val_accuracy: 0.5312
Epoch 13/20
1/1 [=====] - 117s 117s/step - loss: 0.6833
- accuracy: 0.5625 - val_loss: 0.6876 - val_accuracy: 0.7188
Epoch 14/20
1/1 [=====] - 122s 122s/step - loss: 0.6794
- accuracy: 0.7422 - val_loss: 0.6850 - val_accuracy: 0.6875
Epoch 15/20
1/1 [=====] - 117s 117s/step - loss: 0.6727
- accuracy: 0.7031 - val_loss: 0.6766 - val_accuracy: 0.6250
Epoch 16/20
1/1 [=====] - 111s 111s/step - loss: 0.6507
- accuracy: 0.6250 - val_loss: 0.7339 - val_accuracy: 0.5000
Epoch 17/20
1/1 [=====] - 108s 108s/step - loss: 0.6985
- accuracy: 0.5000 - val_loss: 0.6596 - val_accuracy: 0.6562
Epoch 18/20
1/1 [=====] - 126s 126s/step - loss: 0.6226
- accuracy: 0.7344 - val_loss: 0.6721 - val_accuracy: 0.5312
Epoch 19/20
1/1 [=====] - 131s 131s/step - loss: 0.6432
- accuracy: 0.6875 - val_loss: 0.6602 - val_accuracy: 0.8438
Epoch 20/20
1/1 [=====] - 114s 114s/step - loss: 0.6261
- accuracy: 0.8984 - val_loss: 0.6348 - val_accuracy: 0.8125
```

### Lampiran 32. Syntax Prediksi 7 Convolutional Kernel 1 Convolutional Layer

```
model = models.Sequential()
model.add(layers.Conv2D(32, (7, 7), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
"Fitting the data into our model"
#es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(train_images, np.asarray(train_labels), epochs = 20,
                    validation_data =(test_images, np.asarray(test_labels)),
                    batch_size = 200)
```

### Lampiran 33. Hasil Prediksi 7 Convolutional Kernel 1 Convolutional Layer

```
Epoch 1/20
1/1 [=====] - 55s 55s/step - loss: 0.6989 -
accuracy: 0.5000 - val_loss: 141.3185 - val_accuracy: 0.5000
Epoch 2/20
1/1 [=====] - 22s 22s/step - loss: 128.8064
- accuracy: 0.5000 - val_loss: 20.8270 - val_accuracy: 0.5000
Epoch 3/20
1/1 [=====] - 33s 33s/step - loss: 18.8809 -
accuracy: 0.5000 - val_loss: 12.4911 - val_accuracy: 0.5000
Epoch 4/20
1/1 [=====] - 63s 63s/step - loss: 12.9406 -
accuracy: 0.5000 - val_loss: 3.9956 - val_accuracy: 0.5000
Epoch 5/20
1/1 [=====] - 50s 50s/step - loss: 4.1649 -
accuracy: 0.5000 - val_loss: 0.8066 - val_accuracy: 0.5000
Epoch 6/20
1/1 [=====] - 31s 31s/step - loss: 0.7766 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 7/20
1/1 [=====] - 26s 26s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 8/20
1/1 [=====] - 26s 26s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 9/20
1/1 [=====] - 25s 25s/step - loss: 0.6937 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 10/20
1/1 [=====] - 27s 27s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 11/20
1/1 [=====] - 25s 25s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 12/20
1/1 [=====] - 25s 25s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 13/20
1/1 [=====] - 24s 24s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 14/20
1/1 [=====] - 24s 24s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 15/20
1/1 [=====] - 30s 30s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 16/20
1/1 [=====] - 28s 28s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 17/20
1/1 [=====] - 26s 26s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 18/20
1/1 [=====] - 23s 23s/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.7488 - val_accuracy: 0.5000
```



### Lampiran 33. Hasil Prediksi 7 Convolutional Kernel 1 Convolutional Layer

```
Epoch 19/20
1/1 [=====] - 31s 31s/step - loss: 0.7721 -
accuracy: 0.5000 - val_loss: 0.7033 - val_accuracy: 0.5000
Epoch 20/20
1/1 [=====] - 28s 28s/step - loss: 0.7244 -
accuracy: 0.5000 - val_loss: 0.7598 - val_accuracy: 0.5000
```

### Lampiran 34. Syntax Prediksi 7 Convolutional Kernel 2 Convolutional Layer

```
model = models.Sequential()
model.add(layers.Conv2D(32, (7, 7), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (7, 7), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
"Fitting the data into our model"
#es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(train_images, np.asarray(train_labels), epochs = 20,
                    validation_data =(test_images, np.asarray(test_labels)),
                    batch_size = 200)
```

### Lampiran 35. Hasil Prediksi 7 Convolutional Kernel 2 Convolutional Layer

```
Epoch 1/20
1/1 [=====] - 118s 118s/step - loss: 0.6933
- accuracy: 0.5000 - val_loss: 5.2166 - val_accuracy: 0.5000
Epoch 2/20
1/1 [=====] - 90s 90s/step - loss: 5.3958 -
accuracy: 0.5000 - val_loss: 1.1311 - val_accuracy: 0.5000
Epoch 3/20
1/1 [=====] - 83s 83s/step - loss: 1.0524 -
accuracy: 0.5000 - val_loss: 0.7240 - val_accuracy: 0.5000
Epoch 4/20
1/1 [=====] - 84s 84s/step - loss: 0.7088 -
accuracy: 0.5000 - val_loss: 0.7007 - val_accuracy: 0.5000
Epoch 5/20
1/1 [=====] - 82s 82s/step - loss: 0.6952 -
accuracy: 0.5000 - val_loss: 0.6933 - val_accuracy: 0.4062
Epoch 6/20
1/1 [=====] - 73s 73s/step - loss: 0.6926 -
accuracy: 0.5234 - val_loss: 0.6932 - val_accuracy: 0.4688
Epoch 7/20
1/1 [=====] - 77s 77s/step - loss: 0.6931 -
accuracy: 0.5391 - val_loss: 0.7160 - val_accuracy: 0.5000
```

### Lampiran 35. Hasil Prediksi 7 Convolutional Kernel 2 Convolutional Layer

```
Epoch 8/20
1/1 [=====] - 86s 86s/step - loss: 0.7039 -
accuracy: 0.5000 - val_loss: 0.6899 - val_accuracy: 0.5000
Epoch 9/20
1/1 [=====] - 81s 81s/step - loss: 0.6905 -
accuracy: 0.5000 - val_loss: 0.6912 - val_accuracy: 0.5000
Epoch 10/20
1/1 [=====] - 84s 84s/step - loss: 0.6899 -
accuracy: 0.5000 - val_loss: 0.6850 - val_accuracy: 0.5000
Epoch 11/20
1/1 [=====] - 69s 69s/step - loss: 0.6742 -
accuracy: 0.6406 - val_loss: 0.6954 - val_accuracy: 0.5000
Epoch 12/20
1/1 [=====] - 69s 69s/step - loss: 0.6809 -
accuracy: 0.5000 - val_loss: 0.6883 - val_accuracy: 0.5938
Epoch 13/20
1/1 [=====] - 75s 75s/step - loss: 0.6820 -
accuracy: 0.8359 - val_loss: 0.6873 - val_accuracy: 0.6562
Epoch 14/20
1/1 [=====] - 80s 80s/step - loss: 0.6750 -
accuracy: 0.6484 - val_loss: 0.6761 - val_accuracy: 0.5000
Epoch 15/20
1/1 [=====] - 81s 81s/step - loss: 0.6565 -
accuracy: 0.5781 - val_loss: 0.6368 - val_accuracy: 0.5938
Epoch 16/20
1/1 [=====] - 75s 75s/step - loss: 0.5959 -
accuracy: 0.6172 - val_loss: 0.6497 - val_accuracy: 0.7188
Epoch 17/20
1/1 [=====] - 71s 71s/step - loss: 0.6111 -
accuracy: 0.7656 - val_loss: 0.6310 - val_accuracy: 0.5312
Epoch 18/20
1/1 [=====] - 84s 84s/step - loss: 0.5513 -
accuracy: 0.6719 - val_loss: 0.6576 - val_accuracy: 0.5000
Epoch 19/20
1/1 [=====] - 80s 80s/step - loss: 0.5360 -
accuracy: 0.6562 - val_loss: 0.6979 - val_accuracy: 0.5625
Epoch 20/20
1/1 [=====] - 77s 77s/step - loss: 0.6852 -
accuracy: 0.5781 - val_loss: 0.5805 - val_accuracy: 0.8750
```

### Lampiran 36. Syntax Prediksi 7 Convolutional Kernel 3 Convolutional Layer

```
model = models.Sequential()
model.add(layers.Conv2D(32, (7, 7), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (7, 7), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (7, 7), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
"Fitting the data into our model"
#es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
history = model.fit(train_images, np.asarray(train_labels), epochs = 20,
                    validation_data =(test_images, np.asarray(test_labels)),
                    batch_size = 200)
```

### Lampiran 37. Hasil Prediksi 7 Convolutional Kernel 3 Convolutional Layer

```
Epoch 1/20
1/1 [=====] - 80s 80s/step - loss: 0.6947 -
accuracy: 0.5000 - val_loss: 4.7920 - val_accuracy: 0.5000
Epoch 2/20
1/1 [=====] - 82s 82s/step - loss: 4.3952 -
accuracy: 0.5000 - val_loss: 0.6915 - val_accuracy: 0.5000
Epoch 3/20
1/1 [=====] - 92s 92s/step - loss: 0.6940 -
accuracy: 0.5000 - val_loss: 0.6914 - val_accuracy: 0.5000
Epoch 4/20
1/1 [=====] - 96s 96s/step - loss: 0.6936 -
accuracy: 0.5000 - val_loss: 0.6912 - val_accuracy: 0.5000
Epoch 5/20
1/1 [=====] - 92s 92s/step - loss: 0.6910 -
accuracy: 0.5000 - val_loss: 0.7083 - val_accuracy: 0.5000
Epoch 6/20
1/1 [=====] - 90s 90s/step - loss: 0.6994 -
accuracy: 0.5000 - val_loss: 0.6927 - val_accuracy: 0.5000
Epoch 7/20
1/1 [=====] - 92s 92s/step - loss: 0.6932 -
accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 8/20
1/1 [=====] - 90s 90s/step - loss: 0.6935 -
accuracy: 0.5000 - val_loss: 0.6917 - val_accuracy: 0.5000
Epoch 9/20
1/1 [=====] - 88s 88s/step - loss: 0.6953 -
accuracy: 0.5000 - val_loss: 0.6935 - val_accuracy: 0.5000
Epoch 10/20
1/1 [=====] - 87s 87s/step - loss: 0.6928 -
```

### Lampiran 37. Hasil Prediksi 7 Convolutional Kernel 3 Convolutional Layer

```






































































Epoch 10/20
1/1 [=====] - 87s 87s/step - loss: 0.6928 -
accuracy: 0.5000 - val_loss: 0.6947 - val_accuracy: 0.5000
Epoch 11/20
1/1 [=====] - 90s 90s/step - loss: 0.6926 -
accuracy: 0.5000 - val_loss: 0.6933 - val_accuracy: 0.5000
Epoch 12/20
1/1 [=====] - 88s 88s/step - loss: 0.6925 -
accuracy: 0.5000 - val_loss: 0.6996 - val_accuracy: 0.5000
Epoch 13/20
1/1 [=====] - 92s 92s/step - loss: 0.6943 -
accuracy: 0.5000 - val_loss: 0.6976 - val_accuracy: 0.5000
Epoch 14/20
1/1 [=====] - 92s 92s/step - loss: 0.6934 -
accuracy: 0.5000 - val_loss: 0.6944 - val_accuracy: 0.5000
Epoch 15/20
1/1 [=====] - 66s 66s/step - loss: 0.6921 -
accuracy: 0.5000 - val_loss: 0.6929 - val_accuracy: 0.5000
Epoch 16/20
1/1 [=====] - 80s 80s/step - loss: 0.6919 -
accuracy: 0.6016 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 17/20
1/1 [=====] - 85s 85s/step - loss: 0.6921 -
accuracy: 0.7031 - val_loss: 0.6927 - val_accuracy: 0.5000
Epoch 18/20
1/1 [=====] - 88s 88s/step - loss: 0.6917 -
accuracy: 0.6172 - val_loss: 0.6937 - val_accuracy: 0.5000
Epoch 19/20
1/1 [=====] - 86s 86s/step - loss: 0.6910 -
accuracy: 0.5000 - val_loss: 0.6941 - val_accuracy: 0.5000
Epoch 20/20
1/1 [=====] - 87s 87s/step - loss: 0.6896 -
accuracy: 0.5000 - val_loss: 0.6918 - val_accuracy: 0.5625

```

### Lampiran 38. Data yang digunakan



## Lampiran 38. Data yang digunakan

												
Peluang.12	Peluang.13	Peluang.14	Peluang.15	Peluang.16	Peluang.17	Peluang.18	Peluang.19	Peluang.20	Peluang.21	Peluang.22	Peluang.23	Peluang.24
												
Peluang.25	Peluang.26	Peluang.27	Peluang.28	Peluang.29	Peluang.30	Peluang.31	Peluang.32	Peluang.33	Peluang.34	Peluang.35	Peluang.36	Peluang.37
												
Peluang.38	Peluang.39	Peluang.40	Peluang.41	Peluang.42	Peluang.43	Peluang.44	Peluang.45	Peluang.46	Peluang.47	Peluang.48	Peluang.49	Peluang.50
												
Peluang.51	Peluang.52	Peluang.53	Peluang.54	Peluang.55	Peluang.56	Peluang.57	Peluang.58	Peluang.59	Peluang.60	Peluang.61	Peluang.62	Peluang.63
												
Peluang.64	Peluang.65	Peluang.66	Peluang.67	Peluang.68	Peluang.69	Peluang.70	Peluang.71	Peluang.72	Peluang.73	Peluang.74	Peluang.75	Peluang.76
												
Peluang.77	Peluang.78	Peluang.79	Peluang.80									

## BIODATA PENULIS

Photo closed-up



Penulis dilahirkan di Sekayu, 2 November 1999, merupakan anak pertama dari 2 bersaudara. Penulis telah menempuh pendidikan formal yaitu di TK Pembina Sekayu, MI Istiqomah Sekayu, SMPIT Al-Furqon Palembang dan MAN 3 Palembang. Setelah lulus dari SMAN tahun 2016, Penulis mengikuti SBMPTN dan diterima di Departemen Statistika FSAD - ITS pada tahun 2016 dan terdaftar dengan NRP 06211640000049.

Di ITS Penulis sempat aktif di beberapa organisasi, yaitu Himpunan Mahasiswa Statistika ITS (HIMASTA ITS) sebagai staff Departemen Dalam Negeri dan BEM Fakultas Matematika, Komputasi, dan Sains Data sebagai staff Departemen Pengembangan Sumber Daya Mahasiswa dan wakil ketua.