**TUGAS JARINGAN SYARAF TIRUAN**

**BAB 2**

**Pengenalan Pola Huruf dengan Perceptron**



**Disusun oleh:**

**Adrian Maulana Muhammad** **06111540000099**

**M. Kevin Adnan Murbiantoro** **06111640000045**

**Departemen Matematika**

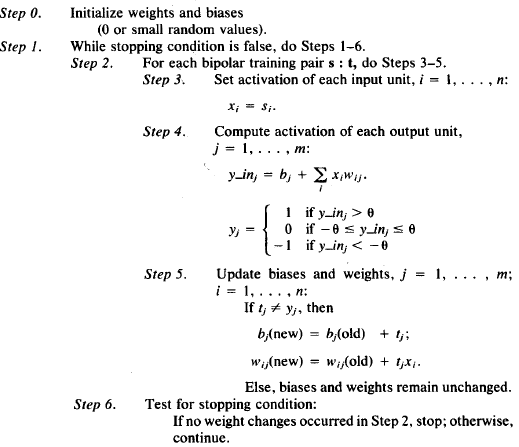
**Fakultas Sains dan Analitika Data**

**Institut Teknologi Sepuluh Nopember**

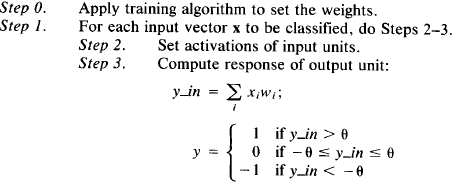
**Surabaya**

**2020**

1. **Algoritma Perceptron**
   1. **Algoritma Training**



* 1. **Algortima Testing**



1. **Implementasi Algoritma Perceptron pada Program**
   1. **Implementasi Algoritma Training**

# ----------Algoritma Training----------

def train(input):

# ----------Step 0----------

# Inisialisasi beban dan bias

weights = {}

bias = {}

# Atur nilai awal dari beban dan bias = 0

for key in input:

weights[key] = np.zeros(input\_size)

bias[key] = 0

# ----------Step 1----------

trained = False # Selama stopping condition adalah false, lakukan step 2-6

for epoch in range(max\_iterations):

trained = True

# ----------Step 2----------

for key, samples in input.items(): # Untuk setiap pasangan training S:T, lakukan step 3-5

target = {} # inisialisasi target

for letter in input:

target[letter] = 1 if letter == key else -1 # nilai dari target

# ----------Step 3----------

for sample\_index, sample in enumerate(samples): # Xi = Si (input units)

# ----------Step 4----------

for letter, t in target.items():

y\_in = bias[letter]

for i in range(input\_size):

y\_in += sample[i] \* weights[letter][i] # y\_in += b + sum(xi\*wi) (output units)

if y\_in > threshold: # jika y\_in lebih besar dari theta

y = 1

elif y\_in < -threshold: # jika y\_in kurang dari minus theta

y = -1

else:

y = 0 # jika y\_in diantara minus theta dan theta

# ----------Step 5----------

if y != t: # Updating nilai dari beban dan bias jika terdapat error

error = t - y

bias[letter] = bias[letter] + LR \* error

for i in range(input\_size):

weights[letter][i] = weights[letter][i] + LR \* sample[i] \* error

trained = False

# ----------Step 6----------

if trained: # Test kondisi stopping

break

# Output dari training

return (trained, weights, bias, epoch)

* 1. **Implementasi Algoritma Testing**

# ----------Algoritma Testing----------

def test(input):

found = []

y\_vec = []

bias\_final = []

weight\_final = []

# ----------Step 0----------

input[input == 0] = -1 # ubah input vector ke bipolar

# ----------Step 1----------

for letter, weight in weights.items(): # Beban didapat dari algoritma training

y\_in = bias[letter]

bias\_final = bias[letter]

weight\_final = weight

# ----------Step 2----------

for s, w in zip(input, weight): # Input units (Xi)

# ----------Step 3----------

y\_in += s \* w # hitung nila dari y\_in = sum(Xi\*Wi)

if y\_in > threshold: # jika y\_in lebih besar dari theta

found.append(letter)

y\_vec.append(1)

elif y\_in < -threshold: # jika y\_in lebih kecil dari min theta

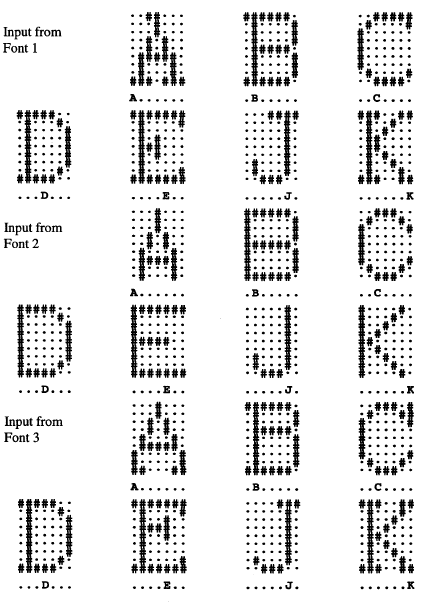
y\_vec.append(-1)

else: # selain dari yang di atas

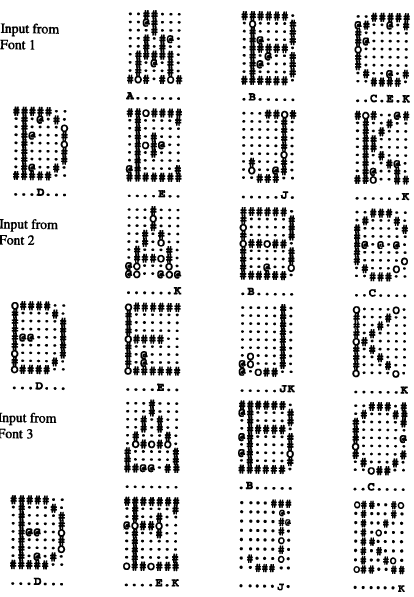
y\_vec.append(0)

return (found, y\_vec, bias\_final, weight\_final)

1. **Implementasi Data pada Program**
   1. **Data Input Training**



* 1. **Data Input Testing**



1. **Source Code pada Program**

# ----------Penggunaan Libaries----------

import numpy as np

import math

from tkinter import \*

import tkinter.filedialog as fdialog

import tkinter.messagebox as messagebox

import os

# ----------Tahap Inisialisasi----------

# Inisialisasi matriks 7x9

w = 7

h = 9

input\_size = w \* h

states = np.zeros((w, h))

# Inisialisasi beban dan bias

weights = {}

bias = {}

# Inisialisasi Threshold (teta) = 0, Learning Rate = 1, dan iterasi maksimum = 1000

threshold = 0

LR = 1

max\_iterations = 10000

# ----------Fungsi untuk Transformasi File menjadi Biner (0 & 1)----------

def open\_file(file):

result = np.zeros((w,h)) # inisialisai matriks 0

lines = [line.rstrip('\n') for line in open(file)] # isi dari tiap (x,y) dari file

for y, line in enumerate(lines):

for x, ch in enumerate(line):

result[x, y] = 1 if ch == '\*' else 0 # jika '-' maka 1, jika '\*' maka 0

return result

# ----------Algoritma Training----------

def train(input):

# ----------Step 0----------

# Inisialisasi beban dan bias

weights = {}

bias = {}

# Atur nilai awal dari beban dan bias = 0

for key in input:

weights[key] = np.zeros(input\_size)

bias[key] = 0

# ----------Step 1----------

trained = False # Selama stopping condition adalah false, lakukan step 2-6

for epoch in range(max\_iterations):

trained = True

# ----------Step 2----------

for key, samples in input.items(): # Untuk setiap pasangan training S:T, lakukan step 3-5

target = {} # inisialisasi target

for letter in input:

target[letter] = 1 if letter == key else -1 # nilai dari target

# ----------Step 3----------

for sample\_index, sample in enumerate(samples): # Xi = Si (input units)

# ----------Step 4----------

for letter, t in target.items():

y\_in = bias[letter]

for i in range(input\_size):

y\_in += sample[i] \* weights[letter][i] # y\_in += b + sum(xi\*wi) (output units)

if y\_in > threshold: # jika y\_in lebih besar dari theta

y = 1

elif y\_in < -threshold: # jika y\_in kurang dari minus theta

y = -1

else:

y = 0 # jika y\_in diantara minus theta dan theta

# ----------Step 5----------

if y != t: # Updating nilai dari beban dan bias jika terdapat error

error = t - y

bias[letter] = bias[letter] + LR \* error

for i in range(input\_size):

weights[letter][i] = weights[letter][i] + LR \* sample[i] \* error

trained = False

# ----------Step 6----------

if trained: # Test kondisi stopping

break

# Output dari training

return (trained, weights, bias, epoch)

# ----------Implementasi fungsi training terhadap file dari folder train----------

def train\_folder():

global weights, bias

# Sesuiakan dengan data training

dir\_path = os.getcwd() + '\\data\\'

data = {}

for file in os.listdir(dir\_path):

if file.endswith('.txt'): # hanya menerima input dengan format .txt

ch = file[0].upper()

if not ch in data:

data[ch] = []

matrix = open\_file(dir\_path + file) # Ubah file kedalam bentuk matriks

matrix = matrix.reshape(input\_size) # Ubah ke 1 dimensi

matrix[matrix == 0] = -1 # Ubah ke bentuk bipolar

data[ch].append(matrix) # insert train data into dictionary

trained, weights, bias, epoch = train(data)

return (epoch)

# ----------Algoritma Testing----------

def test(input):

found = []

y\_vec = []

bias\_final = []

weight\_final = []

# ----------Step 0----------

input[input == 0] = -1 # ubah input vector ke bipolar

# ----------Step 1----------

for letter, weight in weights.items(): # Beban didapat dari algoritma training

y\_in = bias[letter]

bias\_final = bias[letter]

weight\_final = weight

# ----------Step 2----------

for s, w in zip(input, weight): # Input units (Xi)

# ----------Step 3----------

y\_in += s \* w # hitung nila dari y\_in = sum(Xi\*Wi)

if y\_in > threshold: # jika y\_in lebih besar dari theta

found.append(letter)

y\_vec.append(1)

elif y\_in < -threshold: # jika y\_in lebih kecil dari min theta

y\_vec.append(-1)

else: # selain dari yang di atas

y\_vec.append(0)

return (found, y\_vec, bias\_final, weight\_final)

# ----------Design GUI----------

# Inisialisasi root, frame, dan toolbar

root = Tk()

root.configure(background='white')

root.title('Pengenalan Pola Huruf dengan Perceptron')

root.resizable(False, False)

frame = Frame()

frame.configure(background='white')

frame.pack(padx=10, pady=10)

toolbar = Frame(frame, background='white')

toolbar.pack(fill=X)

# Posisi & Ukuran dari GUI

window\_height = 700

window\_width = 650

screen\_width = root.winfo\_screenwidth()

screen\_height = root.winfo\_screenheight()

x\_cordinate = int((screen\_width/2) - (window\_width/2))

y\_cordinate = int((screen\_height/2) - (window\_height/2))

root.geometry("{}x{}+{}+{}".format(window\_width, window\_height, x\_cordinate, y\_cordinate))

# Judul GUI

path = "judul-perceptron.png"

img = PhotoImage(file=path)

label = Label(frame, image=img, background='white').pack(side=TOP)

# Tombol Load

def load\_callback():

global states

file = fdialog.askopenfilename()

if file != '':

states = open\_file(file)

print\_grid()

Button(toolbar, text="Load", command = load\_callback).pack(side=LEFT)

# Tombol Save

def save\_callback():

file = fdialog.asksaveasfile(mode='w', defaultextension=".txt")

for y in range(h):

for x in range(w):

file.write('.' if states[x,y] == 0 else '\*')

file.write('\n')

file.close()

Button(toolbar, text="Save", command = save\_callback).pack(side=LEFT)

# Tombol Clear

def clear\_callback():

np.ndarray.fill(states, 0)

print\_grid()

test\_result\_field\_value.set('')

y\_result\_field\_value.set(0)

Button(toolbar, text="Clear", command = clear\_callback).pack(side=LEFT)

# Entry untuk Learning Rate

Label(toolbar, text='Learning Rate', background='white').pack(side=LEFT, padx = 10)

learning\_rate\_field = Entry(toolbar, textvariable=StringVar(root, value=LR), width=8)

learning\_rate\_field.pack(side=LEFT)

# Entry untuk Threshold

Label(toolbar, text='Threshold', background='white').pack(side=LEFT, padx = 10)

threshold\_field = Entry(toolbar, textvariable=StringVar(root, value=threshold), width=8)

threshold\_field.pack(side=LEFT)

# Entry untuk angka maksimum dari iterasi

Label(toolbar, text='Iterasi Maksimum', background='white').pack(side=LEFT, padx = 10)

max\_iterations\_field = Entry(toolbar, textvariable=StringVar(root, value=max\_iterations), width=8)

max\_iterations\_field.pack(side=LEFT)

# Tombol Train

def train\_callback():

global weights, bias, threshold, LR, max\_iterations

threshold = float(threshold\_field.get())

LR = float(learning\_rate\_field.get())

max\_iterations = int(max\_iterations\_field.get())

epoch = train\_folder()

messagebox.showinfo('Hasil Training', 'Training selesai dengan %d iterasi' % epoch)

Button(toolbar, text="Train", command = train\_callback).pack(side=LEFT)

# Canvas Grid (kotak kosong)

def mouseClick(event):

x = math.floor(event.x / rect\_size)

y = math.floor(event.y / rect\_size)

if x < w and y < h: states[x, y] = 0 if states[x, y] > 0 else 1 # swap zero & one

print\_grid()

rect\_size = 50 # grid rectangles size

canvas = Canvas(frame, width=rect\_size\*w, height=rect\_size\*h, background='white')

canvas.bind("<Button-1>", mouseClick)

canvas.pack(side=TOP)

# Draw Grid (Kotak terarsir)

def print\_grid():

for i in range(w):

for j in range(h):

color = 'black' if states[i, j] > 0 else 'white'

canvas.create\_rectangle(i \* rect\_size, j \* rect\_size, (i + 1) \* rect\_size, (j + 1) \* rect\_size, outline="black", fill=color)

print\_grid();

# Bottom Bar

bottom\_bar = Frame(frame, height=50, background='white')

bottom\_bar.pack(fill=X)

# Tombol Test

def test\_callback():

input = states.copy().reshape(input\_size)

(found, y\_vec, bias\_final, weight\_final) = test(input)

if len(found) > 0:

test\_result\_field\_value.set(', '.join(found))

y\_result\_field\_value.set(y\_vec)

print("Bias yang digunakan: %s" % bias\_final)

print("Beban yang digunakan: ")

print(np.matrix(weight\_final))

print("\n")

else:

test\_result\_field\_value.set('???')

Button(bottom\_bar, text="Test", command = test\_callback).pack(side=LEFT)

# Entry dari Prediksi

Label(bottom\_bar, text='Prediksi', background='white').pack(side=LEFT, padx = 10)

test\_result\_field\_value = StringVar()

test\_result\_field = Entry(bottom\_bar, width=20, textvariable=test\_result\_field\_value)

test\_result\_field.pack(side=LEFT, padx = 10)

# Entry dari nilai y

Label(bottom\_bar, text='Kode Target (A, B, C, D, E, J, K)', background='white').pack(side=LEFT, padx = 10)

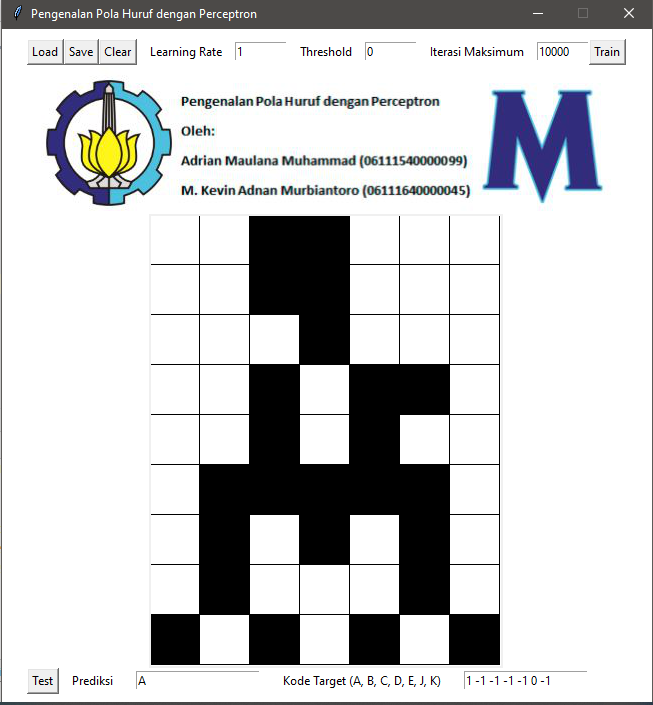
y\_result\_field\_value = IntVar()

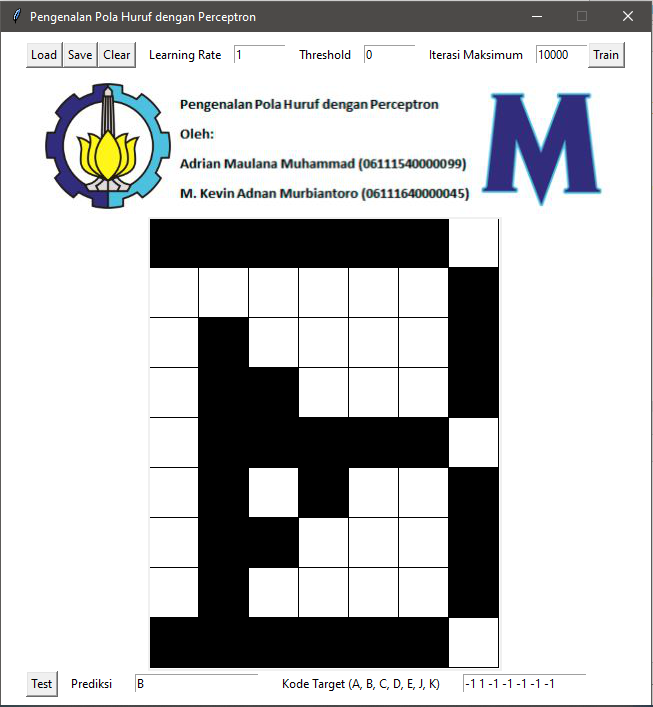
y\_result\_field = Entry(bottom\_bar, width=20, textvariable=y\_result\_field\_value)

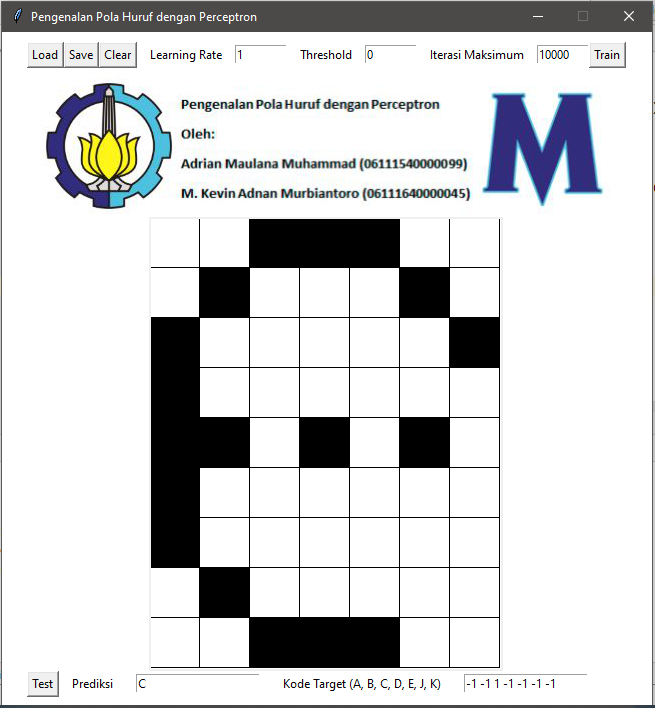
y\_result\_field.pack(side=LEFT, padx = 10)

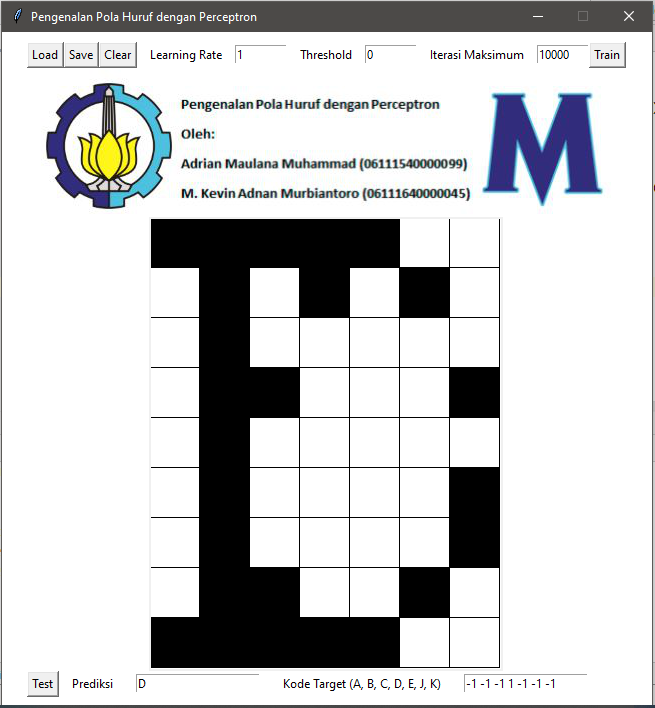
root.mainloop()

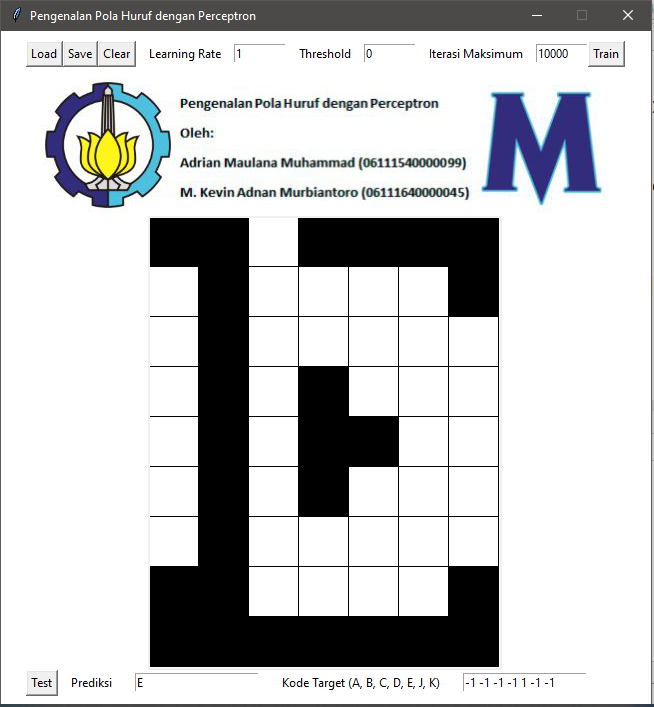
1. **Output dari Program**

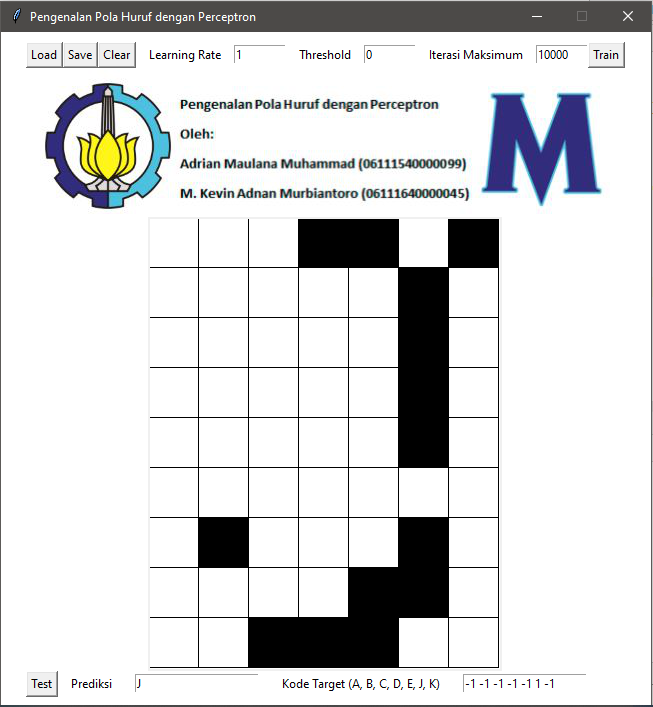


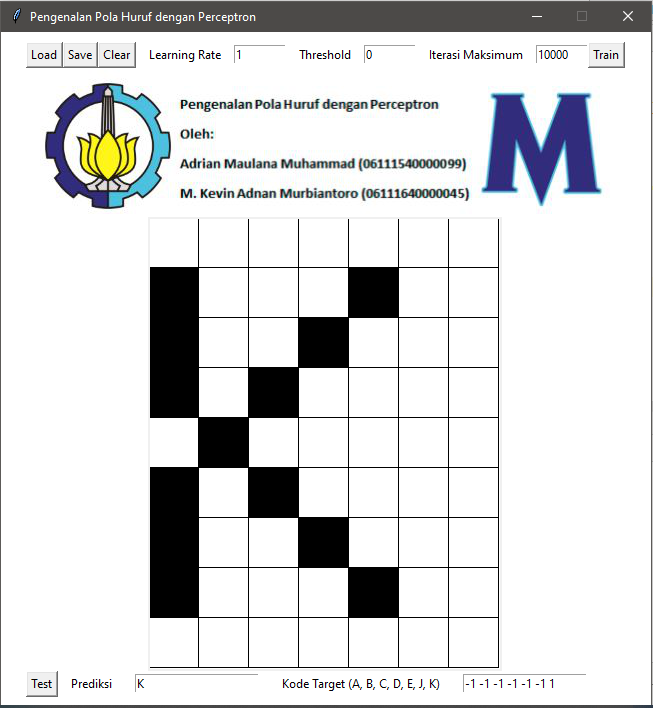


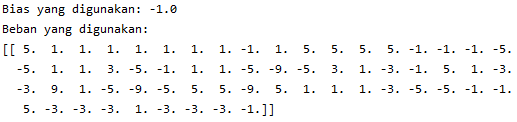












1. **Referensi**

# [Laurene Fausett](https://www.google.co.id/search?tbo=p&tbm=bks&q=inauthor:%22Laurene+Fausett%22), [Laurene V. Fausett](https://www.google.co.id/search?tbo=p&tbm=bks&q=inauthor:%22Laurene+V.+Fausett%22), 1994, “*Fundamentals of Neural Networks: Architectures, Algorithms, and Applications”.*