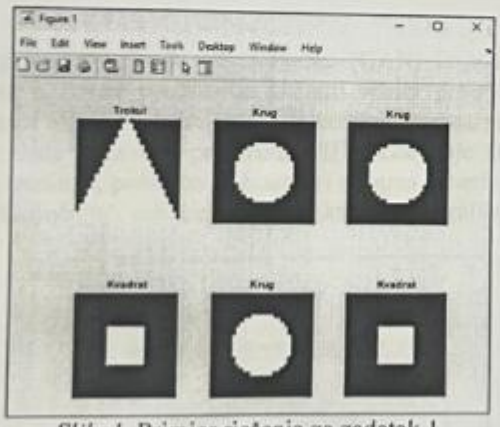
Zadatak 1

Potrebno je razviti jednostavan konvolucioni neuronski model (CNN) u MATLAB-u koji može klasifikovati geometrijske oblike na malim crno-bijelim slikama dimenzija 28×28 piksela. Potrebno je generisati ukupno 120 slika, pri čemu svaka trećina prikazuje kvadrat, krug ili trokut centralno postavljen u slici, a zatim podijeliti slike na trening i test skup. Nakon treniranja mreže na trening skupu, potrebno je testirati model na test skupu i izračunati njegovu tačnost. Dodati vizualni prilaz sa šest test slika sa prikazom predviđenih oblika "Kvadrat", "Krug" ili "Trokut". Na slici 1 je dat primjer rješenja.



Slika1. Primjer rješenja za zadatak 1

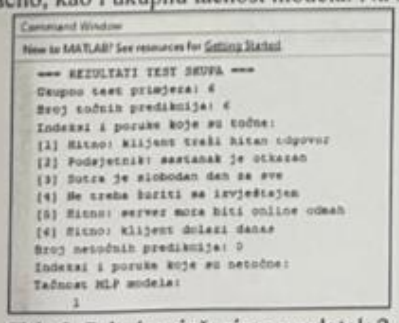
```
clc; clear; close all;

numImages = 120;
imgSize   = 28;

X = zeros(imgSize, imgSize, 1, numImages);
Y = categorical(strings(numImages,1));

[cx, cy] = meshgrid(1:imgSize, 1:imgSize);
center = (imgSize+1)/2;

for i = 1:numImages
    img = zeros(imgSize, imgSize);

    if mod(i,3) == 1
        side = 12;
        half = side/2;

        xMin = round(center - half);
        xMax = round(center + half - 1);
        yMin = round(center - half);
        yMax = round(center + half - 1);

        img(yMin:yMax, xMin:xMax) = 1;
        Y(i) = categorical("Kvadrat");
```

```
    elseif mod(i,3) == 2
        r = 6;
        mask = ((cx - center).^2 + (cy - center).^2) <= r^2;
        img(mask) = 1;
        Y(i) = categorical("Krug");

    else
        baseHalf = 7;
        height   = 12;
        yTop = round(center - height/2);
        yBot = round(center + height/2);

        for yy = yTop:yBot
            t = (yy - yTop) / (yBot - yTop);
            halfWidth = round(t * baseHalf);
            xL = round(center - halfWidth);
            xR = round(center + halfWidth);
            img(yy, xL:xR) = 1;
        end
        Y(i) = categorical("Trougao");
    end

    X(:,:,1,i) = img;
end

idx = randperm(numImages);
trainIdx = idx(1:90);
testIdx  = idx(91:end);

XTrain = X(:,:,:,trainIdx);
YTrain = Y(trainIdx);

XTest  = X(:,:,:,testIdx);
YTest  = Y(testIdx);

layers = [
    imageInputLayer([imgSize imgSize 1])
    convolution2dLayer(3,8,'Padding','same')
    reluLayer
    maxPooling2dLayer(2,'Stride',2)
    convolution2dLayer(3,16,'Padding','same')
    reluLayer
    fullyConnectedLayer(3)
```

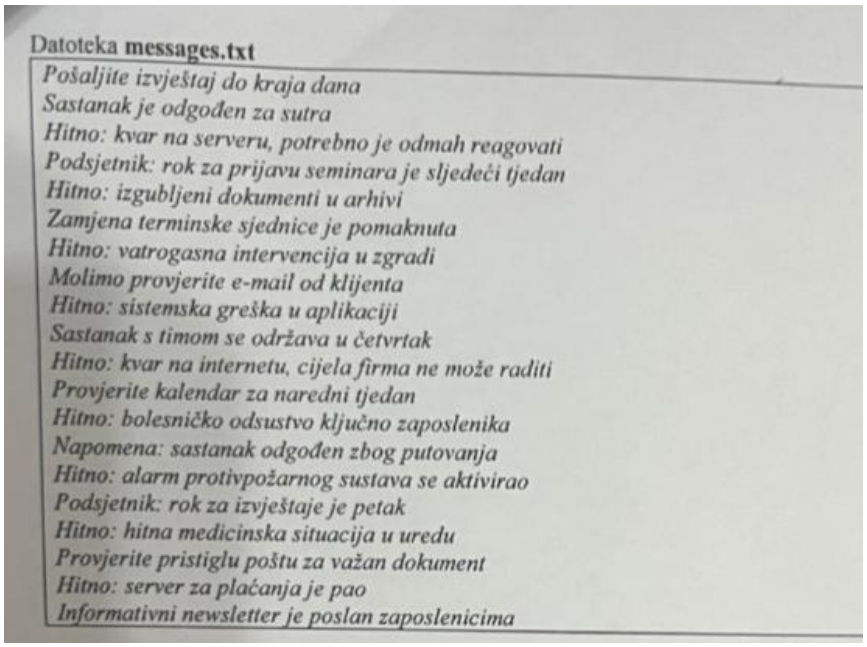
```
    softmaxLayer
    classificationLayer
];

options = trainingOptions('adam', ...
    'MaxEpochs',10, ...
    'MiniBatchSize',16, ...
    'Verbose',false);

net = trainNetwork(XTrain, YTrain, layers, options);

YPred = classify(net, XTest);
accuracy = sum(YPred == YTest)/numel(YTest)*100;

fprintf('Tacnost CNN-a: %.2f %%\n', accuracy);

figure;
for i = 1:6
    subplot(2,3,i);
    imshow(XTest(:,:,1,i));
    title(['Predikcija: ' char(YPred(i))]);
end
```

Zadatak 2

Razviti i implementirati višeslojnu perceptronsku (MLP) neuronsku mrežu za binarnu klasifikaciju tekstualnih poruka koje predstavljaju hitne i nehitne situacije. Ulaz u sistem predstavlja skup od 20 tekstualnih poruka pohranjenih u datoteci *messages.txt*. Cilj modela je da na osnovu sadržaja poruke izvrši procjenu hitnosti i generira binarni izlaz prema sljedećoj klasifikaciji: 1 – hitna poruka, 0 – nehitna poruka. Rezultati klasifikacije trebaju se prikazati u komandnom prozoru, uključujući ukupan broj tačnih i netačnih predikcija, indekse i tekstove poruka koje su klasificirane tačno ili netačno, kao i ukupnu tačnost modela. Na slici 2 je dat primjer rješenja.



Slika2. Primjer rješenja za zadatak 2



Datoteka **messages.txt**

Pošaljite izvještaj do kraja dana
Sastanak je odgođen za sutra
Hitno: kvar na serveru, potrebno je odmah reagovati
Podsjetnik: rok za prijavu seminara je sljedeći tjedan
Hitno: izgubljeni dokumenti u arhivi
Zamjena terminske sjednice je pomaknuta
Hitno: vatrogasna intervencija u zgradi
Molimo provjerite e-mail od klijenta
Hitno: sistemska greška u aplikaciji
Sastanak s timom se održava u četvrtak
Hitno: kvar na internetu, cijela firma ne može raditi
Provjerite kalendar za naredni tjedan
Hitno: bolesničko odsustvo ključno zaposlenika
Napomena: sastanak odgođen zbog putovanja
Hitno: alarm protivpožarnog sustava se aktivirao
Podsjetnik: rok za izvještaje je petak
Hitno: hitna medicinska situacija u uredu
Provjerite pristiglu poštu za važan dokument
Hitno: server za plaćanja je pao
Informativni newsletter je poslan zaposlenicima

```
clc; clear; close all;

%% 1) Ucitavanje poruka iz messages.txt
fid = fopen('messages.txt','r');
C = textscan(fid,'%s','Delimiter','\n');
fclose(fid);

messagesRaw = C{1};

N = length(messagesRaw);

%% 2) Definicija rijeci (hitno / nehitno)
urgentWords = { ...

'hitno','odmah','kvar','server','sistemska','greška','greska
','alarm','protivpožarnog','protivpozarnog', ...

'vatrogasna','intervencija','internet','ne','može','moze','r
aditi','pao','pala','medicinska','situacija', ...

'bolesni?ko','bolesnicko','odsustvo','klju?nog','kljucnog','
zaposlenika','pla?anje','placanje','online'};

nonUrgentWords = { ...
```

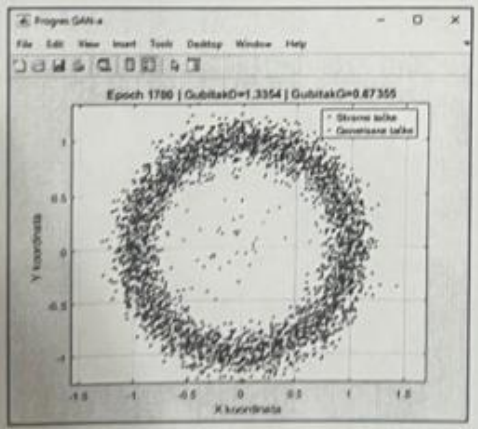
```
'pošaljite','posaljite','izvještaj','izvjestaj','sastanak','
odgo?en','odgodjen','sutra','podsjetnik','rok', ...

'prijavu','seminara','sljede?i','sljedeci','tjedan','?etvrta
k','cetvrtak','petak','napomena', ...

'informativni','bilten','poslan','zaposlenicima','kalendar',
'putovanja','poštu','postu','e-poštu','e-postu', ...

'dokument','dokumenti','arhivi','terminala','zamjena','dan'}
;

%% 3) Kreiranje ulaza X i cilja Y
% X: 2 ulaza -> broj hitnih rijeci + broj nehitnih rijeci
X = zeros(2, N);

% Y: stvarna klasa iz prefiksa "Hitno:"
Y = zeros(1, N);

for i = 1:N
    original = strtrim(messagesRaw{i});
    low = lower(original);

    % Stvarna klasa (ground truth)
    if startsWith(low, "hitno:")
        Y(i) = 1;
        low = strrep(low, "hitno:", "");
    else
        Y(i) = 0;
    end

    % ocisti interpunkciju
    low = regexprep(low, '[^\p{L}0-9\s]', ' ');
    words = split(strtrim(low));
    words = words(~strcmp(words,""));

    urgentCount = sum(ismember(words, urgentWords));
    nonUrgentCount = sum(ismember(words, nonUrgentWords));

    X(:, i) = [urgentCount; nonUrgentCount];
end

%% 4) Kreiranje i trening MLP mreze
```

```
net = feedforwardnet(5);  % 1 skriveni sloj sa 5 neurona
net.trainParam.epochs = 500;

% Podjela (interno)
net.divideParam.trainRatio = 0.8;
net.divideParam.valRatio   = 0.1;
net.divideParam.testRatio  = 0.1;

net = train(net, X, Y);

%% 5) Predikcija
Y_pred  = net(X);
Y_class = double(Y_pred > 0.5);

%% 6) ISPIS REZULTATA (FORMAT KAO U PRIMJERU)

isCorrect  = (Y_class == Y);
numCorrect = sum(isCorrect);
numWrong   = N - numCorrect;
accuracy   = numCorrect / N;    % 0-1

correctIdx = find(isCorrect);
wrongIdx   = find(~isCorrect);

disp('REZULTATI TEST SKUP')
disp(' ')
fprintf('Ukupno test primjera: %d\n\n', N);

fprintf('Broj ta?nih predikcija: %d\n\n', numCorrect);

disp('Indeksi i poruke koje su ta?ne:')
disp(' ')

for k = 1:length(correctIdx)
    i = correctIdx(k);
    fprintf('[%d] %s\n\n', i, strtrim(messagesRaw{i}));
end

fprintf('Broj neto?nih predvi?anja: %d\n\n', numWrong);

disp('Indeks i poruke koje su neto?ne:')
disp(' ')

for k = 1:length(wrongIdx)
    i = wrongIdx(k);
```

```
    fprintf('[%d] %s\n\n', i, strtrim(messagesRaw{i}));
end

disp('Ta?nost MLP modela:')
disp(' ')
disp(accuracy)

%% 7) Graf (opcionalno)
figure;
stem(1:N, Y, 'k', 'LineWidth', 2); hold on;
stem(1:N, Y_class, 'r--', 'LineWidth', 1.5);
xlabel('Poruka');
ylabel('Klasa (0=nehitno, 1=hitno)');
title('Stvarno stanje vs predikcija MLP');
legend('Stvarna klasa','Predikcija MLP','Location','best');
grid on;
```

Zadatak 3

Razviti i implementirati generativnu suparničku mrežu (GAN) koja uči distribuciju 2D tačaka raspoređenih u obliku kruga. Ulaz u sistem predstavlja latentni vektor proizvoljne dimenzije, dok izlaz generatora predstavlja 2D koordinatu tačke. Diskriminator uči razlikovati stvarne tačke od generisanih. Cilj je da generator nakon treniranja bude sposoban proizvoditi 2D tačke koje slijede istu distribuciju kao originalne (stvarne) tačke. Tokom treninga, potrebno je prikazati progres generisanih i stvarnih tačaka u 2D prostoru koristeći *figure* naredbu, uključujući oznake i legende, kako bi se pratila konvergencija GAN-a. Na slici 3 je dat primjer rješenja.



Slika3. Primjer rješenja za zadatak 3

```
clear; clc; close all;
%% 1) Dataset: 2D ta?ke u obliku kruga (REAL DATA)
numReal = 5000;
theta = 2*pi*rand(numReal,1);
r = 1 + 0.1*randn(numReal,1);
```

```
XReal = [r.*cos(theta), r.*sin(theta)]; % 2D kružnica
%% 2) DEFINICIJA GENERATORA
latentDim = 10; % dimenzija latentnog vektora
layersG = [
 featureInputLayer(latentDim, 'Normalization','none')
 fullyConnectedLayer(32)
 reluLayer
 fullyConnectedLayer(16)
 reluLayer
 fullyConnectedLayer(2) % generisana 2D ta?ka
];
lgraphG = layerGraph(layersG);
dlnetG = dlnetwork(lgraphG)
%% 3) DEFINICIJA DISKRIMINATORA
layersD = [
 featureInputLayer(2, 'Normalization','none')
 fullyConnectedLayer(32)
 leakyReluLayer(0.2)
 dropoutLayer(0.3)
 fullyConnectedLayer(16)
 leakyReluLayer(0.2)
 fullyConnectedLayer(1)
 sigmoidLayer % izlaz = vjerovatno?a real/fake
];
lgraphD = layerGraph(layersD);
dlnetD = dlnetwork(lgraphD);
%% 4) OPCIJE TRENIRANJA
numEpochs = 3000;
learningRate = 0.0005;
miniBatchSize = 64;
%% 5) TRENIRANJE GAN-A
figure;
for epoch = 1:numEpochs
 % === MINI-BATCH REALNIH PODATAKA ===
 idx = randperm(numReal, miniBatchSize);
 XBatch = XReal(idx,:);
 dlXReal = dlarray(single(XBatch'), 'CB'); % C = feature, B
= batch
 % === GENERISANJE LATENT VEKtORA ===
 Z = randn(latentDim, miniBatchSize, 'single');
 dlZ = dlarray(Z, 'CB'); % C = latentDim, B = batch
 % === Izra?un gradijenata sa dlfeval ===
 [gradientsD, gradientsG, lossD, lossG] =
dlfeval(@modelGradients, dlnetD,
```

```
dlnetG, dlXReal, dlZ);
 % === Update mreža ===
 dlnetD = dlupdate(@(p,g) p - learningRate*g, dlnetD,
gradientsD);
 dlnetG = dlupdate(@(p,g) p - learningRate*g, dlnetG,
gradientsG);
 % === PRIKAZ PROGRESIJE ===
 if mod(epoch, 100) == 0
 dlXFake = forward(dlnetG, dlZ);
 fake = extractdata(dlXFake)';
 plot(XReal(:,1), XReal(:,2), 'b.'); hold on;
 plot(fake(:,1), fake(:,2), 'r.');
 legend('Real','Fake');
 title("GAN Epoch " + epoch + ...
 " | LossD=" + gather(extractdata(lossD)) + ...
 " | LossG=" + gather(extractdata(lossG)));
grid on; axis equal;
 drawnow;
 hold off;
 end
end
disp('Trening GAN-a je završen.');
%%
=============================================================
% Funkcija za gradijente generatora i diskriminatora
%%
=============================================================
function [gradientsD, gradientsG, lossD, lossG] =
modelGradients(dlnetD,
dlnetG, dlXReal, dlZ)
 % Forward pass generatora
 dlXFake = forward(dlnetG, dlZ);
 % Forward pass discriminatora
 dReal = forward(dlnetD, dlXReal);
 dFake = forward(dlnetD, dlXFake);
 % Loss funkcije
 lossD = -mean(log(dReal + 1e-8) + log(1 - dFake + 1e-8));
 lossG = -mean(log(dFake + 1e-8));
 % Gradijenti
 gradientsD = dlgradient(lossD, dlnetD.Learnables);
end
```