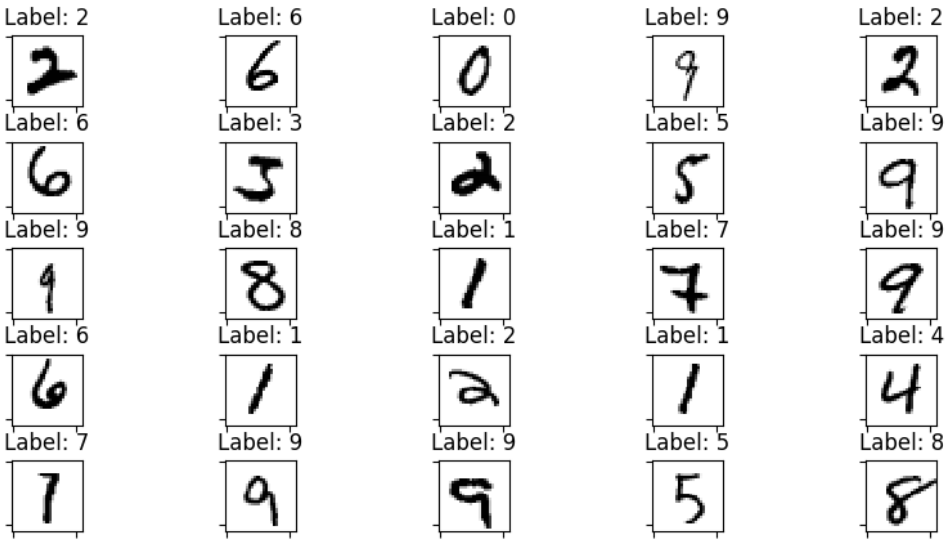
VJEŽBA 8: KLASIFIKACIJA RUKOM PISANIH BROJEVA

I. Cilj vježbe:

Primijeniti znanje stečeno o neuronskim mrežama na problemu klasifikacije rukom pisanih brojeva.

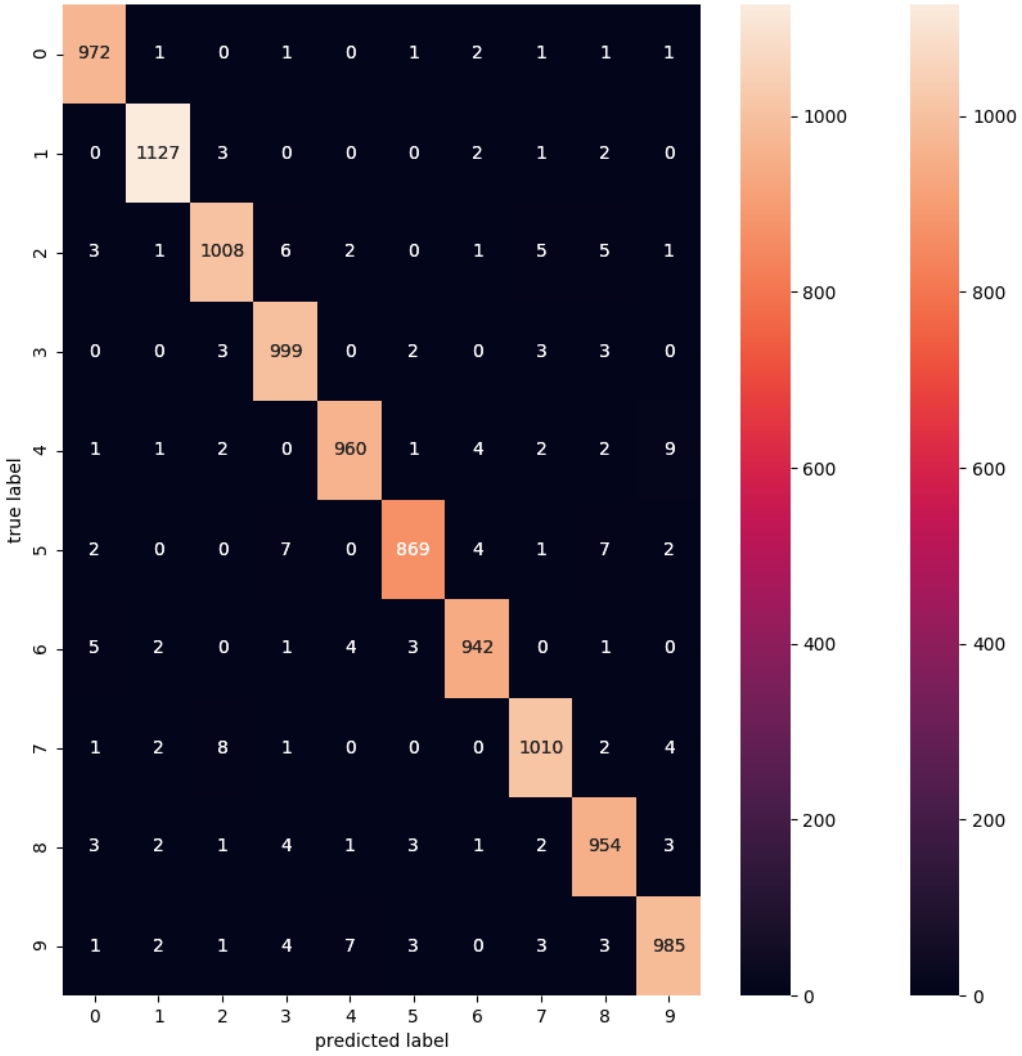
II. Opis vježbe:

U ovoj vježbi razmatra se problem klasifikacije rukom pisanih brojeva. Za izgradnju modela za klasifikaciju rukom pisanih brojeva na raspolaganju je skup podataka pod nazivom MNIST. Ovaj skup sadrži slike rukom pisanih brojeva koje su pisali zaposlenici u *United States Census Bureau* i američki studenti. Slike su zapisane u sivim tonovima odnosno svaki piksel na slici ima vrijednost u rasponu od 0 do 255. Slike su normirane na dimenziju 28 x 28 piksela. Svaka slika ima odgovarajuću oznaku tj. labelu (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). MNIST sadrži skup podataka za učenje od 60,000 slika, te skup podataka za testiranje koji sadrži 10,000 slika. Primjer slika iz skupa podataka za učenje dan je na slici 8.1.

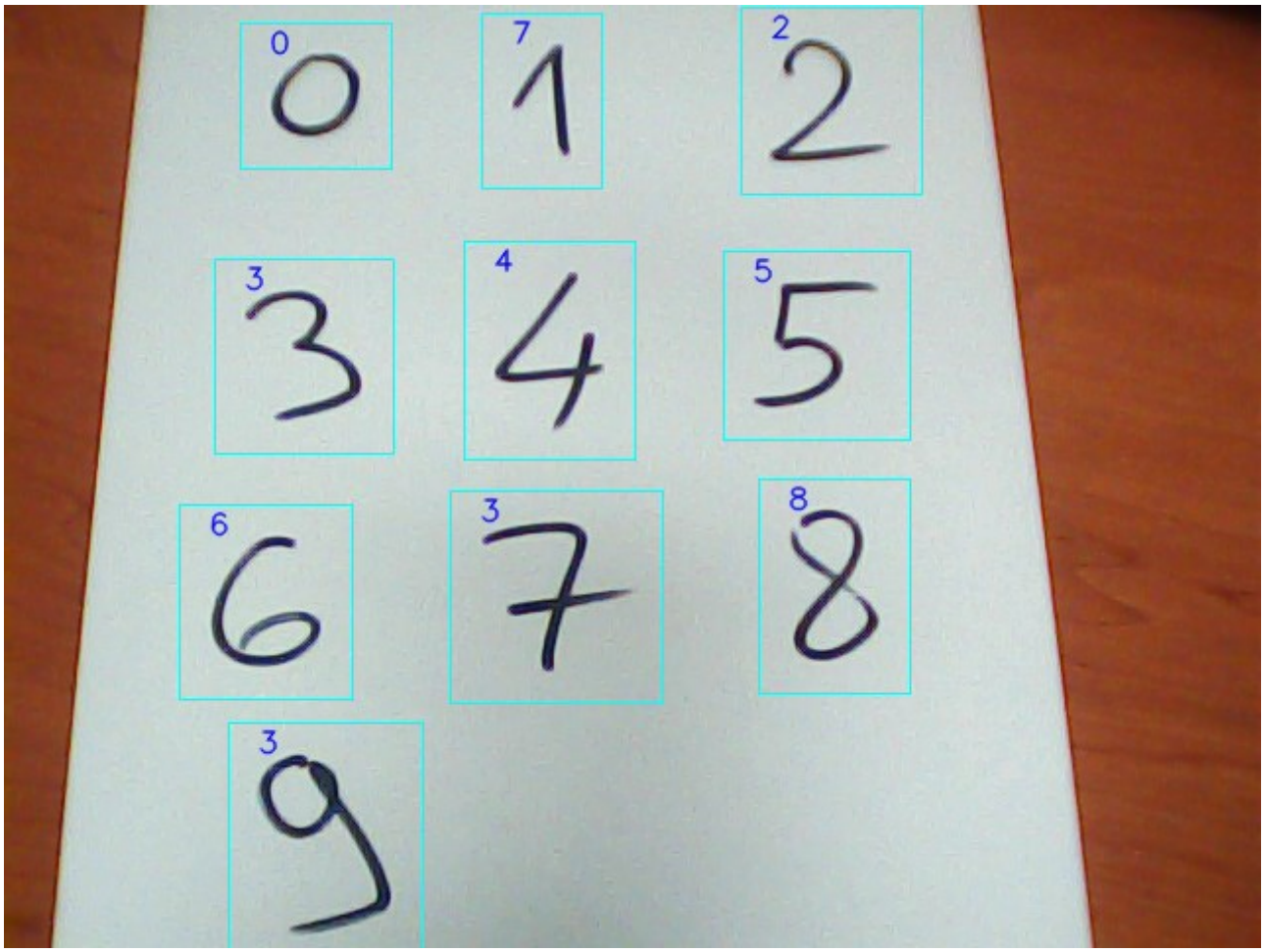


Sl. 8.1. Primjer podataka iz trening skupa MNIST.

U okviru vježbe potrebno je izgraditi neuronsku mrežu na MNIST skupu podataka te izvršiti njenu evaluaciju na skupu podatka za testiranje. Nadalje neuronsku mrežu je potrebno iskoristiti za klasifikaciju rukom pisanih brojeva u video signalu koji dolazi sa kamere spojene na računalo. Na slici 8.2. dan je primjer evaluacije izgrađene mreže pomoću matrice zabune na testnom skupu podatka. Na slici 8.3. dan je primjer upotrebe mreže za klasifikaciju rukom pisanih brojeva u video signalu.



Sl. 8.2. Evaluacija neuronske mreže na testnom skupu podataka.



Sl. 8.3. Upotreba neuronske mreže za klasifikaciju rukom pisanih brojeva u video signalu.

Podsjetnik
Generalni postupak učenja MLP neuronske mreže na određenom problemu u *scikit learn* okruženju može se sažeti u sljedeće korake:

1. Učitati raspoložive podatke. Dio podataka koristi se kao skup podataka za učenje dok se preostali podaci koriste za testiranje izgrađene mreže. Provesti predobradbu podataka (standardizacija, skaliranje, centriranje i sl.).
2. Strukturirati neuronsku mreže (koristiti klasu *MLPClassifier*): odabir broja slojeva, odabir broja neurona u pojedinom sloju, odabir tipa aktivacijske funkcije neurona te eventualno podesiti neke dodatne parametre vezane za postupak učenja neuronske mreže poput željenog numeričkog postupka, dodatni parametre optimizacije, poput koeficijenta regularizacije, broj iteracija numeričkog postupka (tzv. epohe) i sl.
3. Na temelju skupa za učenje odrediti parametre (težine) mreže odgovarajućim numeričkim postupkom.
4. Testiranje mreže kako bi se pokazale predikcijska svojstva mreže (npr. srednja kvadratna pogreška na testnim podacima, matrica zabune i sl.).

III. Priprema za vježbu:

Ponovite gradivo vezano za neuronske mreže.

IV. Rad na vježbi:

1. Klonirajte vaš repozitorij `rusu_lv_2019_20` na računalo pomoću gitbash. Zatim povucite moguće promjene iz izvornog repozitorija pomoću naredbi:

```
git remote add upstream https://gitlab.com/rgrbic/rusu_lv_2019_2020
git fetch upstream
```

```
git merge upstream/master
```

2. Riješite dane zadatke, pri čemu Python skripte trebaju imati naziv `zad_x.py` (gdje je `x` broj zadatka) i trebaju biti pohranjene u direktorij `rusu_lv_2019_20/LV8/solutions/`. Svaki zadatak rješavajte u zasebnoj *git* grani koju spojite s glavnom granom kada riješite pojedini zadatak. Pohranite skripte u lokalnu *git* bazu kao i u `rusu_lv_2019_20` repozitorij na vašem korisničkom računu. Svaki puta kada načinite promjene koje se spremaju u *git* sustav napišite i odgovarajuću poruku prilikom izvršavanja `commit` naredbe.
3. Nadopunite postojeću tekstualnu datoteku `rusu_lv_2019_20/LV8/Readme.md` s kratkim opisom vježbe i kratkim opisom rješenja vježbe te pohranite promjene u lokalnu bazu. Na kraju pohranite promjene u udaljeni repozitorij.

Zadatak 1

U prilogu vježbe nalazi se skripta 8.1. koja učitava MNIST skup podataka. Dopunite skriptu na odgovarajućim mjestima:
1) Naučite model neuronske mreže pomoću *scikitlearn* bibilioteke. Model treba imati naziv *mlp_mnist*.
2) Izračunajte točnost izgrađene mreže na skupu podataka za učenje i skupu podataka za testiranje.
3) Prikažite matricu zabune na skupu podataka za učenje i na skupu podataka za testiranje. Komentirajte dobivene rezultate.

Zadatak 2

U prilogu vježbe nalazi se funkcija 8.2. koja učitava video signal kamere koja je priključena na računalo. Svaki video okvir video signala se pretprocesira i izdvajaju se rubovi objekata s ciljem izdvajanja pojedinačnih brojeva. Dopunite kod kako bi se svaki izdvojeni broj klasificirao s mrežom naučenom u prošlom zadatku. Evaluirajte rješenje na način da na bijelom papiru napišete flomasterom brojeve od 0 do 9 te ih snimajte s kamerom s odgovarajuće udaljenosti. Komentirajte dobivene rezultate.

Zadatak 3

Dopunite kod iz zadatka 2 tako da se na ekranu osim klase ispisuje i vjerojatnost pripadanja klasi.

Zadatak 4

Promijenite skripte 8.1. i 8.2. tako da se po želji korisnika koristi jedan od modela:
1) Logistička regresija
2) Metoda K najbližih susjeda
3) Neuronska mreža

V. Izvještaj s vježbe

Kao izvještaj s vježbe prihvaća se web link na repozitorij pod nazivom `rusu_lv_2019_20` koji sadrži rješenja unutar direktorija `rusu_lv_2019_20/LV8/solutions/`.

VI. Dodatak

Skripta 8.1.

```
import numpy as np
from sklearn.datasets import fetch_mldata
from sklearn.externals import joblib
import pickle


mnist = fetch_mldata('MNIST original')
X, y = mnist.data, mnist.target

#print('Got MNIST with %d training- and %d test samples' % (len(X), len(y_test)))
#print('Image size is:')

# rescale the data, train/test split
X = X / 255.
X_train, X_test = X[:60000], X[60000:]
y_train, y_test = y[:60000], y[60000:]


# TODO: build youw own neural network using sckitlearn MPLClassifier


# TODO: evaluate trained NN


# save NN to disk
filename = "NN_model.sav"
joblib.dump(mlp_mnist, filename)
```

Skripta 8.2.

```
import cv2
import numpy as np
from sklearn.externals import joblib
from sklearn.neural_network import MLPClassifier


# display
cv2.namedWindow("frame", cv2.WINDOW_NORMAL)
cv2.namedWindow("edges", cv2.WINDOW_NORMAL)
font = cv2.FONT_HERSHEY_SIMPLEX

# load neural network from disk
filename = "NN_model.sav"
mlp_mnist = joblib.load(filename)

# algorithm params
pad = 15
size_th = 32
mnist_size = 28

# video processing
cp = cv2.VideoCapture(0)
kernel1 = np.ones((7,7), np.uint8)
kernel2 = np.ones((5,5), np.uint8)

# some vars
label = "unkown"

while True:

    ret, frame = cp.read(0)

    # frame preprocessing - getting edges
```

```
gray_img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
gray_img = cv2.GaussianBlur(gray_img, (5, 5), 0)
v = np.median(gray_img)
lower = int(max(0, (1.0 - 0.33) * v))
upper = int(min(255, (1.0 + 0.33) * v))
edge_img = cv2.Canny(gray_img, lower, upper)
img_preprocessed = cv2.dilate(edge_img, kernel1, iterations=1)
img_preprocessed = cv2.erode(img_preprocessed, kernel2, iterations=1)

# get countours and bounding boxes (rects)
_, contours, _ = cv2.findContours(img_preprocessed.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

rects = [cv2.boundingRect(contour) for contour in contours]
rects = [rect for rect in rects if rect[2] >= 3 and rect[3] >= 8]


# loop over all rectangles (detections) and classify them
for rect in rects:

    x, y, w, h = rect

    #crop rectangle from image
    cropped_digit = img_preprocessed[y-pad:y+h+pad, x-pad:x+w+pad]
    cropped_digit = cropped_digit/255.0

    #filter small rectangles:
    if cropped_digit.shape[0] >= size_th and cropped_digit.shape[1] >= size_th:
        cropped_digit = cv2.resize(cropped_digit, (mnist_size, mnist_size))
        img_vector = cropped_digit.reshape(1,mnist_size*mnist_size)
    else:
        continue

    # start TODO: classify image (img_vector) with trained neural network and place correct class into variable label (as string)


    # end of TODO

    # show rectangle and label on frame
    cv2.rectangle(frame, (x - pad, y - pad), (x + pad + w, y + pad + h), color = (255, 255, 0))

    cv2.putText(frame, label, (rect[0], rect[1]), font,
        fontScale = 0.5,
        color = (255, 0, 0),
        thickness =  1,
        lineType =  cv2.LINE_AA)


# show results
cv2.imshow("frame", frame)
cv2.imshow("edges", img_preprocessed)

if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```