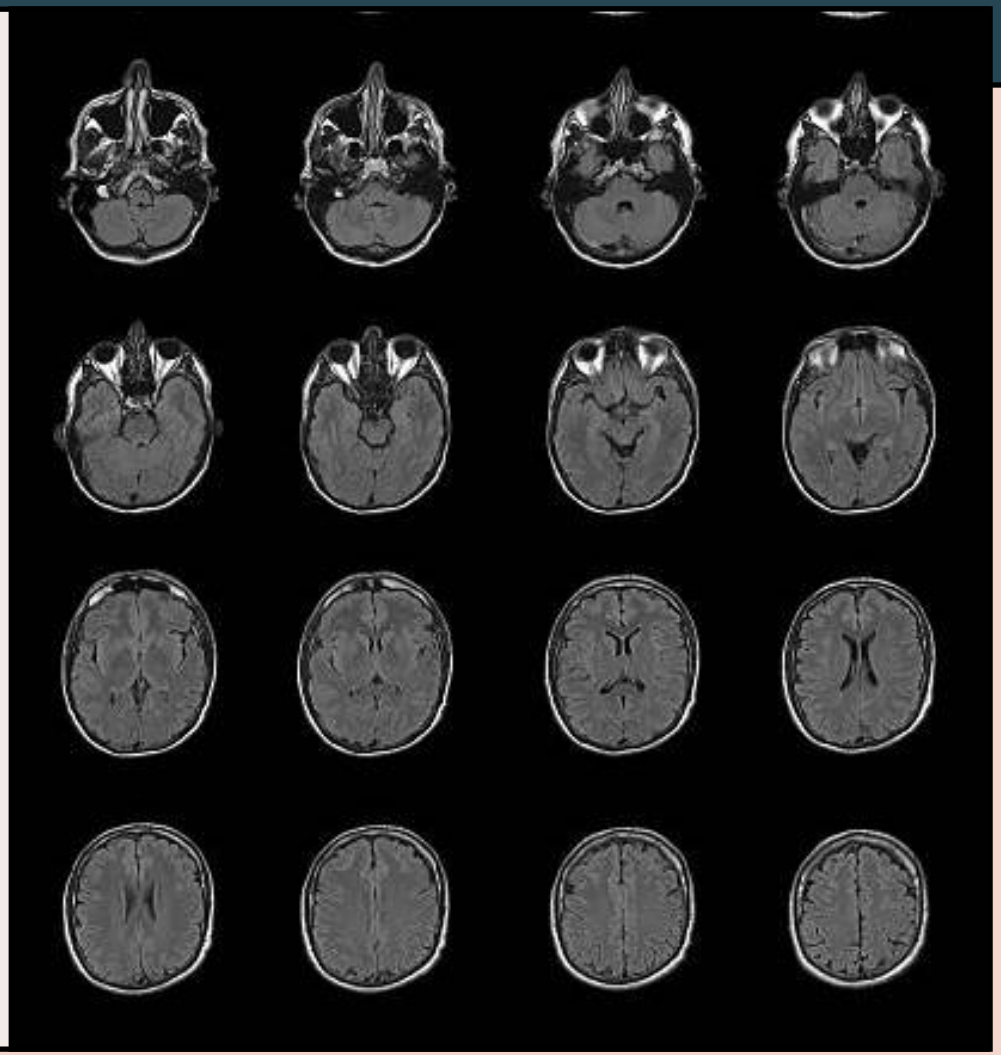
Detekcija Alzheimerove bolesti na MRI slikama mozga

Projekt iz predmeta
ANALIZA SLIKA U BIOMEDICINI



Maja Jurić, Klara Ilčić, Borna Nikolić, Ana Ujević, Magda Radić

Medicina i umjetna inteligencija

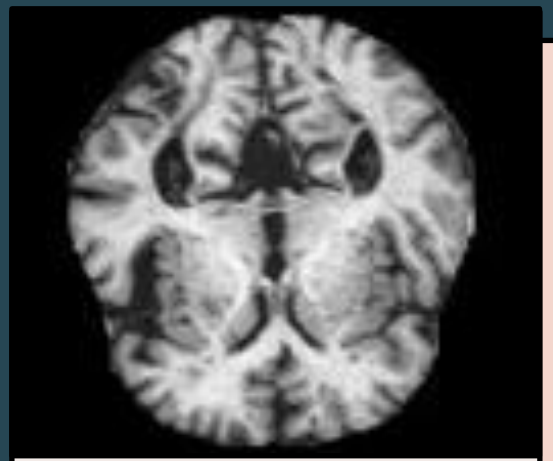
- preciznost i učinkovitost dijagnostike
- personalizirano liječenje
- ubrzanje dijagnostike i liječenje

MOTIVACIJA

- 55 milijuna ljudi u svijetu boluje od Alzheimerove bolesti (podatci iz 2020.)
- rani stadiji Alzheimerove bolesti često ostaju neprepoznati
- detekcija promjena u mozgu u vrlo ranoj fazi bolesti

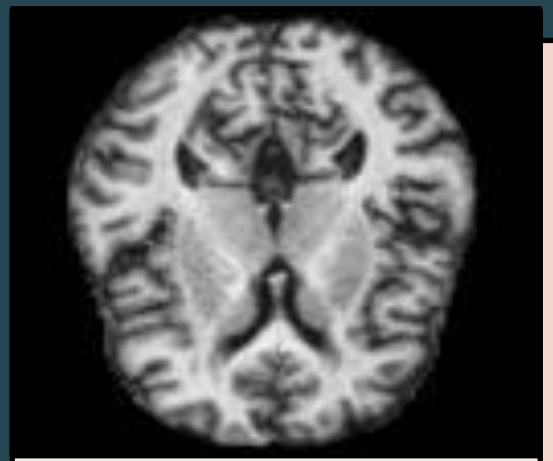
Skup podataka

- preuzeto s platforme *Kaggle*
- preprocesirane slike, reduciranih dimenzija 128 x 128 piksela
- treniranje:testiranje - 75:25



Non demented

3200



Very mild demented

2240

Mild demented

896

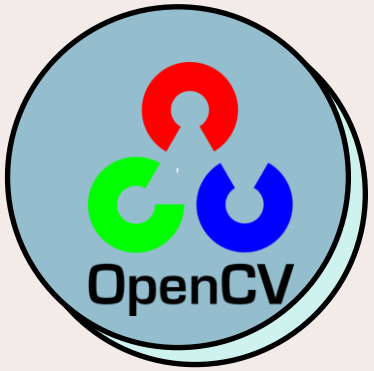
Moderate demented

64

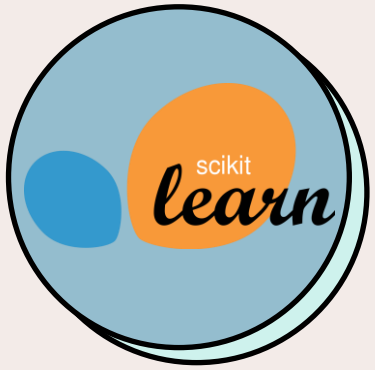
Tehnologije

- metode strojnog i dubokog učenja
- Jupyter notebook

Python



OpenCV



scikit-learn

TensorFlow

PyTorch

Metode

SVM

Efficient Net

GoogLe Net

ResNet

CNN

SVM

- klasifikacijski regresijski algoritam
- pronalazi najbolju granicu za razdvajanje podataka u kl ase
- testirane: linearna, polinomijalna drugog stupnja i RBF jezgra
- sa i bez težina klasa


```
 X_train, X_test, Y_train, Y_test):
X_train, Y_train)
odel.predict(X_train)
ocnosti na skupu za treniranje:")
classification_report(Y_train, Y_pred, target_names = cla
odel.predict(X_test)
ocnosti na skupu za testiranje:")
classification_report(Y_test, Y_pred, target_names = clas
el


tke(image_size, asGrayscale=True)

0/3200 [00:01<00:00, 1825.63it/s]
/896 [00:00<00:00, 1821.24it/s]
64 [00:00<00:00, 1766.30it/s]
0/2240 [00:01<00:00, 1817.52it/s]


(classes[y])


hape(len(X), -1)


_train, Y_test = train_test_split(X_reshaped, Y_class, tr


t_class_weights(Y_train)
ts)


ss weights:")
 SVC(kernel='linear')
 svm(svm_model_linear, X_train, X_test, Y_train, Y_test)


s:
upu za treniranje:
```

Validacija SVM

	preciznost	odziv	f1	broj primjera
non demented	0.86	0.62	0.72	800
mild demented	0.74	0.81	0.77	224
moderate demented	0.83	0.94	0.88	16
very mild demented	0.57	0.77	0.65	560

EfficentNet

- ujednačeno skaliranje dubine, širine i rezolucije mreže
- zadovoljavajući rezultati za manji broj parametara
- 25 epoha
- veličina grupe 30
- dimenzije slika 224 x 224

```
 = 224
ate = 0.01
ate_delta = 0.001
 = 25
 = 30

itavanje podatka (+ mijenjanje dimenzija)

taj_podatke(image_size, asGrayscale=False) # input mora imati 3

| 3200/3200 [00:03<00:00, 815.75it/s]
| 896/896 [00:01<00:00, 842.05it/s]
| 64/64 [00:00<00:00, 938.35it/s]
| 2240/2240 [00:02<00:00, 820.18it/s]

[]
:
s.append(classes[y])
 = tf.keras.utils.to_categorical(Y_class)

_test, Y_train, Y_test = train_test_split(X, Y_class, train_size
                                          shuffle = True, random
np.array(X_train)
o.array(X_test)
np.array(Y_train)
o.array(Y_test)

nts = get_class_weights(Y_train)

tf.keras.utils.to_categorical(Y_train)
f.keras.utils.to_categorical(Y_test)

 jer se drugi predugo treniraju (ali bi vjerojatno onda imali bolje performanse).

etModel = tf.keras.applications.EfficientNetB0(weights='imagenet

ficientNetModel.output
.keras.layers.GlobalAveragePooling2D()(model)
.keras.layers.Dropout(0.5)(model)
.keras.layers.Dense(len(classes.keys()), activation='softmax')(m
.keras.models.Model(inputs=EfficientNetModel.input, outputs=mode
```

Validacija EfficentNet

	preciznost	odziv	f1	broj primjera
non demented	0.86	0.62	0.72	800
mild demented	0.74	0.81	0.77	224
moderate demented	0.83	0.94	0.88	16
very mild demented	0.57	0.77	0.65	560

GoogLeNet

- 27 slojeva
- zanemarivanje težina
- slike 224 x 224
- 25 epoha
- veličina grupe 30
- aktivacijske funkcije: ReLu, softmax (u zadnjem sloju)

```
 Conv2D(filters=f1, kernel_size = (1,1), padding = 'same', activ

educe  = Conv2D(filters = f2_conv1, kernel_size = (1,1), padding
 Conv2D(filters = f2_conv3, kernel_size = (3,3), padding = 'same

educe = Conv2D(filters = f3_conv1, kernel_size = (1,1), padding
 Conv2D(filters = f3_conv5, kernel_size = (5,5), padding = 'same

 MaxPooling2D((3,3), strides= (1,1), padding = 'same')(input_lay
roj = Conv2D(filters = f4, kernel_size = (1,1), padding = 'same

yer = Concatenate(axis = -1)([conv1x1, conv3x3, conv5x5, maxpool

tput_layer


(input_shape, num_classes):
er = Input(shape = input_shape)

tional layer: filters = 64, kernel_size = (7,7), strides = 2
D(filters = 64, kernel_size = (7,7), strides = 2, padding = 'sam

ling layer: pool_size = (3,3), strides = 2
oling2D(pool_size = (3,3), strides = 2)(X)

tional layer: filters = 64, strides = 1
D(filters = 64, kernel_size = (3,3), strides = 1, padding = 'sam

ling layer: pool_size = (3,3), strides = 2
oling2D(pool_size = (3,3), strides = 2)(X)

on block 3a
tion_block(X, f1 = 64, f2_conv1 = 96, f2_conv3 = 128, f3_conv1 =

on block 3b
tion_block(X, f1 = 128, f2_conv1 = 128, f2_conv3 = 192, f3_conv

ling layer: pool_size = (3,3), strides = 2
oling2D(pool_size= (3,3), strides = 2)(X)

on block 4a
tion_block(X, f1 = 192, f2_conv1 = 96, f2_conv3 = 208, f3_conv1

on block 4b
tion_block(X, f1 = 160, f2_conv1 = 112, f2_conv3 = 224, f3_conv

on block 4c
tion_block(X, f1 = 128, f2_conv1 = 128, f2_conv3 = 256, f3_conv

on block 4d
tion_block(X, f1 = 112, f2_conv1 = 144, f2_conv3 = 288, f3_conv

on block 4e
```

Validacija GoogLeNet

	preciznost	odziv	f1	broj primjera
non demented	0.91	0.98	0.94	800
mild demented	0.95	0.85	0.9	224
moderate demented	0.92	0.75	0.83	16
very mild demented	0.92	0.86	0.89	560

Razvijeni CNN

- razvijen CNN prema pronađenoj literaturi
- 15 slojeva
- 8 potpuno povezanih, ostali slojevi udruživanja i konvolucijski
- zanemarivanje težina
- dimenzije 224 x 224
- 25 epoha
- veličina grupa 30


```
est, Y_train, Y_test = train_test_split(X, Y_class, train_size =
                                        shuffle = True, random_s
.array(X_train)
array(X_test)
.array(Y_train)
array(Y_test)

.keras.utils.to_categorical(Y_train)
keras.utils.to_categorical(Y_test)

dela

ential()

nv2D(16, (3, 3), activation='relu', input_shape=(image_size, ima

xPooling2D(pool_size=(2, 2)))

nv2D(32, (3, 3), activation='relu'))

xPooling2D(pool_size=(2, 2)))

nv2D(32, (3, 3), activation='relu'))

xPooling2D(pool_size=(2, 2)))

atten())

nse(10, activation='relu'))
nse(5, activation='relu'))
nse(12, activation='relu'))
nse(30, activation='relu'))
nse(10, activation='relu'))
nse(100, activation='relu'))
nse(133, activation='relu'))
nse(4, activation='softmax'))

y()

ial_10"
                 Output Shape              Param #
=================================================================
nv2D)            (None, 148, 148, 16)      160

152 (MaxPool     (None, 74, 74, 16)        0

nv2D)            (None, 72, 72, 32)        4640
```

Validacija razvijenog CNN

	preciznost	odziv	f1	broj primjera
non demented	0.96	0.94	1.7	800
mild demented	0.88	0.88	0.88	224
moderate demented	1.00	0.75	0.86	16
very mild demented	0.87	0.91	0.89	560

ResNet

- koncept preskočenih veza - omogućuju bržu propagaciju gradijenta i kombinaciju informacije iz ranijih slojeva
- rezidualni blokovi od više konvolucijskih slojeva
- zanemarivanje težina
- dimenzije 224 x 224
- 25 epoha
- veličina grupa 30


```
nsor = torch.tensor(X_train).unsqueeze(1).float()  # Add channe
or = torch.tensor(X_val).unsqueeze(1).float() # Add channel dim
sor = torch.tensor(X_test).unsqueeze(1).float()    # Add channe

nsor = torch.tensor(Y_train).long()
or = torch.tensor(Y_val).long()
sor = torch.tensor(Y_test).long()
```

```
set = TensorDataset(x_train_tensor, y_train_tensor)
t = TensorDataset(x_val_tensor, y_val_tensor)
et = TensorDataset(x_test_tensor, y_test_tensor)

er = DataLoader(train_dataset, batch_size=batch_size, shuffle=T
 = DataLoader(val_dataset, batch_size=batch_size, shuffle=False
r = DataLoader(test_dataset, batch_size=batch_size, shuffle=Fal
```

odela

```
models.ResNet18_Weights.IMAGENET1K_V1
 models.resnet18(weights=weights)

 resnet18.float()

onv1 = nn.Conv2d(1, 64, kernel_size=7, stride=2, padding=3, bia
in resnet18.parameters():
requires_grad = False
in resnet18.layer4.parameters():
requires_grad = True
in resnet18.fc.parameters():
requires_grad = True

 resnet18.fc.in_features
c = nn.Linear(num_ftrs, len(classes))
```

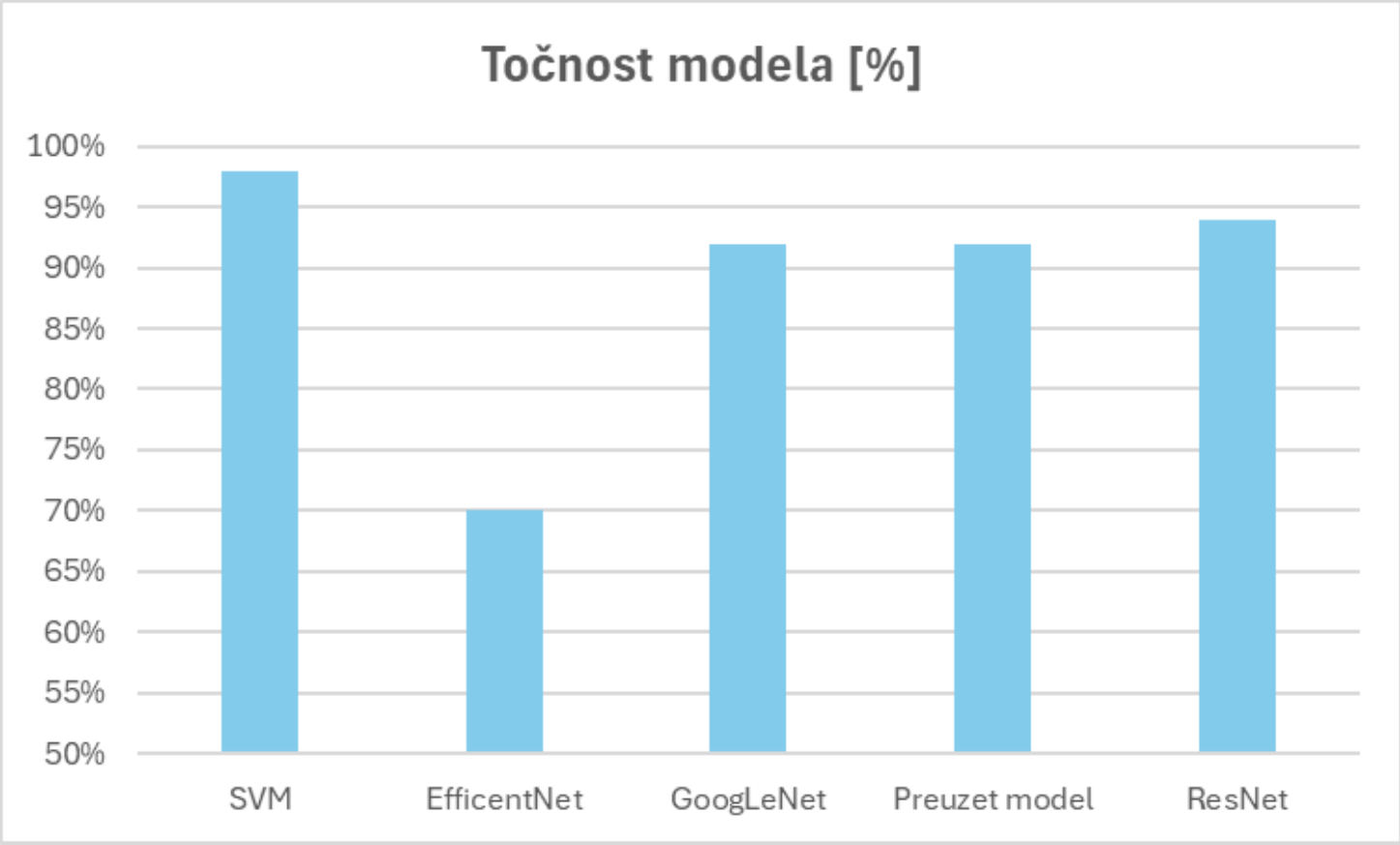
```
= nn.CrossEntropyLoss()
= optim.Adam([ # ili staviti SGD u skladu s paperom
ms': resnet18.layer4.parameters()},
ms': resnet18.fc.parameters()}
ning_rate)
```

```
es = []
 = [], []
racies = []
```

Validacija ResNet

	preciznost	odziv	f1	broj primjera
non demented	0.92	0.98	0.95	800
mild demented	0.92	0.92	0.92	224
moderate demented	1.00	0.75	0.86	16
very mild demented	0.96	0.88	0.92	560

Usporedba točnosti



Diskusija

SVM najbolji

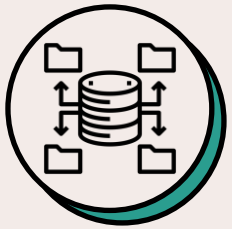
Suprotno literaturi i favoriziranju dubokog učenja.

EfficentNet najgori

224 od 1600 slika krivo.

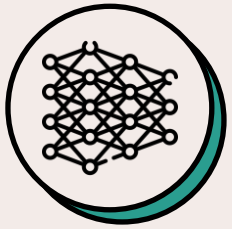
Razlog takvih rezultata?

Nešto krivo?



Skup podataka

Izjednačiti broj primjera u pojedinim klasama.



Složenije neuronske mreže

Isprobavanja kompliciranijih arhitektura neuronskih mreža.

Hvala na pozornosti

maja.juric@fer.hr

klara.ilicic@fer.hr

borna.nikolic@fer.hr

ana.ujevic@fer.hr

magda.radic@fer.hr

```
    true_labels = []

    print("Training...")
    for inputs, labels in tqdm(train_loader):
        optimizer.zero_grad()

        outputs = resnet18(inputs) # forward pass

        loss = criterion(outputs, labels)
        loss.backward() # backward pass
        optimizer.step() # korak optimizacije

        total_loss += loss.item()
        _, predicted = torch.max(outputs, 1) # vjerojatnosti u klase

        total_correct += (predicted == labels).sum().item()
        total += labels.size(0)

        predictions.extend(predicted.view(-1).tolist())
        true_labels.extend(labels.view(-1).tolist())

    train_accuracy = total_correct / total
    train_f1 = f1_score(true_labels, predictions, average='weighted'
    train_losses.append(total_loss / len(train_loader))
    train_accuracies.append(train_accuracy)
    train_f1_scores.append(train_f1)
    print("Train accuracy:", train_accuracy)

    # validacija
    resnet18.eval()  # evaluation mode
    total_val_loss = 0
    total_val_correct = 0
    total_val = 0
    val_predictions = []
    val_true_labels = []

    with torch.no_grad():+
```