Kristina Radivojevic 22/0457 Nera Mihajlović 22/0211

DRUGI ZADATAK

Naš zadatak je projektovanje **potpuno povezane konvolucione neuronske mreže** koja vrši **klasifikaciju slika lica prema emocijama** u unapred definisane klase.

Korišćen je **FER-2013** skup podataka, preuzet sa platforme *Kaggle* (https://www.kaggle.com/datasets/msambare/fer2013). Ovaj skup sadrži slike ljudskih lica koje pripadaju jednoj od **7 emocionalnih klasa**:

- **angry** (ljutnja)
- **disgust** (gađenje)
- **fear** (strah)
- **happy** (sreća)
- **neutral** (neutralno stanje)
- **sad** (tuga)
- **surprise** (iznenađenje)

Sve slike su u **nivoima sive (grayscale)** i imaju dimenzije **48×48 piksela**, čime dataset predstavlja kompaktan, ali izazovan skup za prepoznavanje emocija.

Podaci su podeljeni u dve grupe:

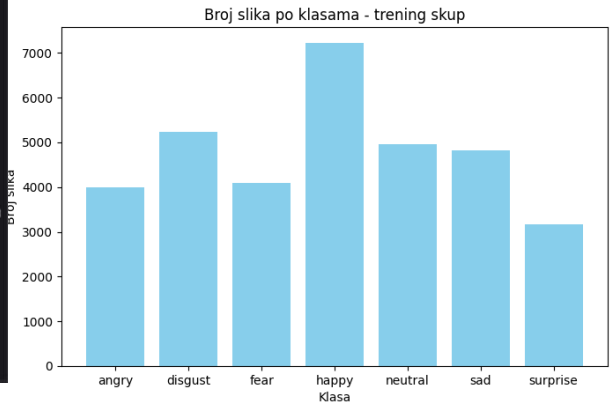
- **Trening skup:** `"C:/Users/PC/Desktop/archive/train"`
- **Test skup:** `"C:/Users/PC/Desktop/archive/test"`

Učitavanje podataka vrši se pomoću funkcije `image_dataset_from_directory` iz biblioteke **TensorFlow**, gde se za svaku sliku automatski dodeljuje oznaka klase na osnovu strukture direktorijuma. Veličina slike postavljena je na **(48, 48)**, dok je veličina mini-batcheva tokom obuke iznosila **32**.

Nakon učitavanja, potvrđene su sledeće klase:

Klase: ['angry', 'disgust', 'fear','happy', 'neutral', 'sad', 'surprise']

```
test = "C:/Users/PC/Desktop/archive/test"
train = "C:/Users/PC/Desktop/archive/train"

img_size = (48, 48)
color_mode = "grayscale"
batch_size = 32  #mozemo menjati

testData = image_dataset_from_directory(test,
                                        color_mode="grayscale",
                                        labels="inferred",
                                        label_mode="int",
                                        image_size=img_size,
                                        batch_size=batch_size,
                                        shuffle=False)

fullTrain = image_dataset_from_directory(train,
                                        color_mode="grayscale",
                                        labels="inferred",
                                        label_mode="int",
                                        image_size=img_size,
                                        batch_size=batch_size,
                                        shuffle=True)


classes = fullTrain.class_names
num_classes = len(classes)
print("Klase:", classes)

val_size = 0.2
total_batches = tf.data.experimental.cardinality(fullTrain).numpy()
train_batches = int((1 - val_size) * total_batches)
trainData = fullTrain.take(train_batches).map(lambda x, y: (x/255., tf.one_hot(y, num_classes)))
valData = fullTrain.skip(train_batches).take(total_batches - train_batches).map(lambda x, y: (x/255., tf.one_hot(
testData = testData.map(lambda x, y: (x/255., tf.one_hot(y, num_classes)))


klase = os.listdir(train)
counts = [len(os.listdir(os.path.join(train, c))) for c in klase]

plt.figure(figsize=(8,5))
plt.bar(klase, counts, color='skyblue')
plt.title("Broj slika po klasama - trening skup")
plt.xlabel("Klasa")
plt.ylabel("Broj slika")
plt.show()
```



Pre treniranja modela slike se **normalizuju** tako da svi pikseli budu u opsegu 0–1, tako što se vrednosti podeljuju sa 255. Ovaj postupak ubrzava i stabilizuje učenje neuronske mreže jer sve ulazne vrednosti imaju sličan rang.

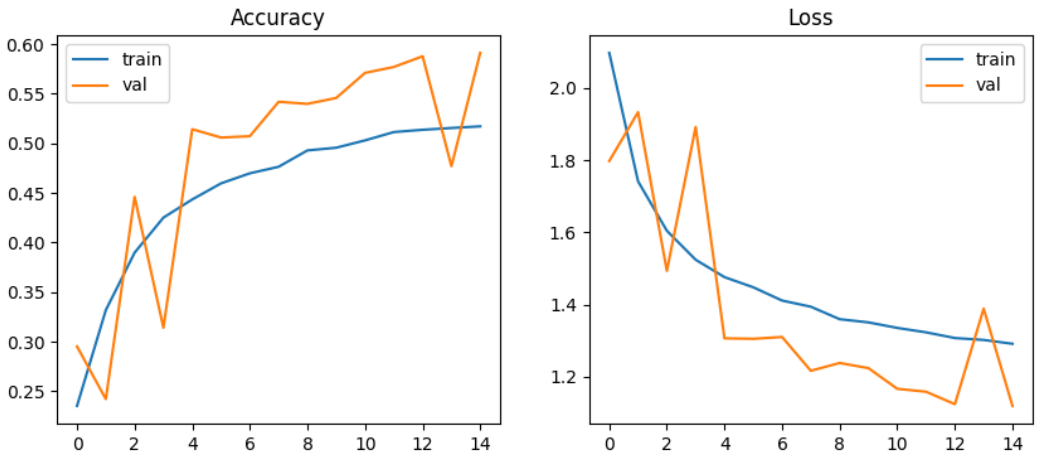
Podaci su podeljeni na trening, validacioni i test skup. Validacioni skup zauzima 20% ukupnih podataka, dok se preostali podaci koriste za trening. Takođe, etikete (klase) se kodiraju u one-hot format, što znači da je svaka klasa predstavljena vektorom gde je pozicija odgovarajuće klase jednaka 1, a sve ostale 0.

Ova priprema omogućava modelu da efikasno uči i precizno predviđa klase na izlazu.

Prikaz po jedne slike iz svake od 7 klasa


```
classes = trainData.class_names
one_per_class = {}
plt.figure(figsize=(14,3))

for images, labels in trainData:
    for img, lab in zip(images, labels):
        lab_int = int(lab.numpy())
        if lab_int not in one_per_class:
            one_per_class[lab_int] = img
    if len(one_per_class) == len(classes):
        break

for i, (lab_int, img) in enumerate(one_per_class.items()):
    plt.subplot(1, len(classes), i+1)
    plt.imshow(img.numpy().astype('uint8'))
    plt.title(classes[lab_int])
    plt.axis('off')

plt.show()
```

Prikaz performansi modela nad podacima trening i test skupa




```
early_stop = EarlyStopping(monitor='val_loss', patience=8, restore_best_weights=True)


y_train_classes = np.concatenate([y.numpy() for x, y in trainData], axis=0)
y_train_classes = np.argmax(y_train_classes, axis=1)


class_weights = {i: 1.0 for i in range(num_classes)}


history = model.fit(
    trainData,
    epochs=15,
    validation_data=valData,
    callbacks=[early_stop],
    verbose=2,
    class_weight=class_weights
)

plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='val')
plt.title('Accuracy')
plt.legend()

plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='val')
plt.title('Loss')
plt.legend()
plt.show()
```

Model je konvolucioni neuronski sa tri konvoluciona bloka i potpuno povezanim slojem na izlazu.

Za kriterijumsku funkciju izabrana je **categorical crossentropy**, koja je pogodna za višeklasne probleme sa one-hot kodiranim etiketama i efikasno kažnjava veće greške, što pomaže modelu da preciznije uči.

Skriveni slojevi koriste **ReLU** aktivaciju, koja uvodi nelinearnost i sprečava problem nestajućeg gradijenta, omogućavajući modelu učenje kompleksnih obrazaca.

Izlazni sloj koristi **softmax**, koja pretvara rezultate u verovatnoće po klasama i omogućava pravilno klasifikovanje slike u klasu sa najvećom verovatnoćom.

Za optimizaciju je korišćen **Adam**, jer adaptivno menja learning rate na osnovu prosečnih vrednosti gradijenta i njegovog kvadrata, što ubrzava i stabilizuje učenje.

```
model = Sequential([
    # Data augmentation
    RandomFlip("horizontal", input_shape=(48, 48, 1)),
    RandomRotation(0.1),
    RandomZoom(0.1),

    # 1. blok
    Conv2D(32, (3, 3), activation='relu', padding='same'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Dropout(0.2),

    # 2. blok
    Conv2D(64, (3, 3), activation='relu', padding='same'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Dropout(0.25),

    # 3. blok
    Conv2D(128, (3, 3), activation='relu', padding='same'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Dropout(0.3),

    # Fully connected
    Flatten(),
    Dense(256, activation='relu'),
    BatchNormalization(),
    Dropout(0.4),
    Dense(num_classes, activation='softmax')
])

# Kompajliranje modela
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

Model: "sequential"

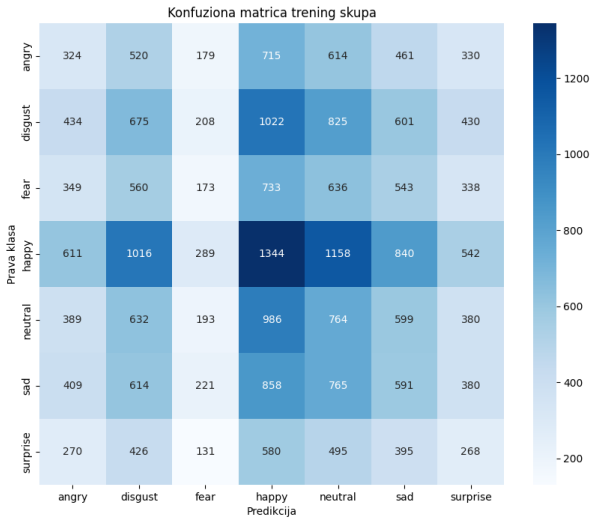
Layer (type)	Output Shape	Param #
random_flip (RandomFlip)	(None, 48, 48, 1)	0
random_rotation (RandomRotation)	(None, 48, 48, 1)	0
random_zoom (RandomZoom)	(None, 48, 48, 1)	0
conv2d (Conv2D)	(None, 48, 48, 32)	320
batch_normalization (BatchNormalization)	(None, 48, 48, 32)	128
max_pooling2d (MaxPooling2D)	(None, 24, 24, 32)	0
dropout (Dropout)	(None, 24, 24, 32)	0
conv2d_1 (Conv2D)	(None, 24, 24, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 24, 24, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 64)	0
dropout_1 (Dropout)	(None, 12, 12, 64)	0
conv2d_2 (Conv2D)	(None, 12, 12, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 12, 12, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 128)	0
dropout_2 (Dropout)	(None, 6, 6, 128)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 256)	1,179,904
batch_normalization_3 (BatchNormalization)	(None, 256)	1,024
dropout_3 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 7)	1,799

Total params: 1,276,295 (4.87 MB)
Trainable params: 1,275,335 (4.87 MB)
Non-trainable params: 960 (3.75 KB)

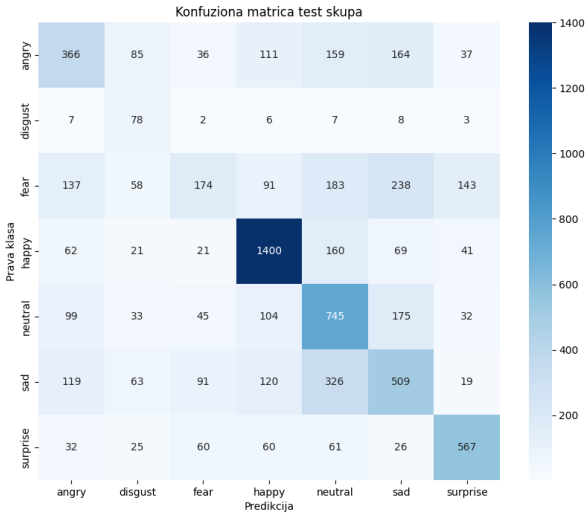
Konfuziona matrica trening skupa

Trening tačnost: 0.6060

```
train_loss, train_acc = model.evaluate(trainData, verbose=0)
print(f"Trening tačnost: {train_acc:.4f}")

y_true = np.concatenate([y.numpy() for x, y in trainData], axis=0)
y_pred = model.predict(trainData, verbose=0)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true_classes = np.argmax(y_true, axis=1)

cm = confusion_matrix(y_true_classes, y_pred_classes)
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=classes, yticklabels=classes)
plt.title('Konfuziona matrica trening skupa')
plt.xlabel('Predikcija')
plt.ylabel('Prava klasa')
plt.show()
```



	precision	recall	f1-score	support
angry	0.45	0.38	0.41	958
disgust	0.21	0.70	0.33	111
fear	0.41	0.17	0.24	1024
happy	0.74	0.79	0.76	1774
neutral	0.45	0.60	0.52	1233
sad	0.43	0.41	0.42	1247
surprise	0.67	0.68	0.68	831
accuracy			0.53	7178
macro avg	0.48	0.53	0.48	7178
weighted avg	0.53	0.53	0.52	7178



Konfuziona matrica test skupa

Test tačnost: 0.5348

```
x_test_np = np.concatenate([x.numpy() for x, y in testData], axis=0)
y_test_np = np.concatenate([y.numpy() for x, y in testData], axis=0)
y_true_classes = np.argmax(y_test_np, axis=1)

y_pred_classes = np.argmax(model.predict(x_test_np, verbose=0), axis=1)

test_loss, test_acc = model.evaluate(x_test_np, tf.one_hot(y_true_classes, num_classes), verbose=0)
print(f"Test tačnost: {test_acc:.4f}")
print(classification_report(y_true_classes, y_pred_classes, target_names=classes))


cm = confusion_matrix(y_true_classes, y_pred_classes)
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=classes, yticklabels=classes)
plt.title('Konfuziona matrica test skupa')
plt.xlabel('Predikcija')
plt.ylabel('Prava klasa')
plt.show()
```

Primeri dobro i loše klasifikovanih slika


```
correct = np.where(y_pred_classes == y_true_classes)[0]
incorrect = np.where(y_pred_classes != y_true_classes)[0]

num_samples = 5
correct_sample = np.random.choice(correct, size=min(num_samples, len(correct)), replace=False)
incorrect_sample = np.random.choice(incorrect, size=min(num_samples, len(incorrect)), replace=False)

plt.figure(figsize=(12, 6))

for i, idx in enumerate(correct_sample):
    plt.subplot(2, 5, i+1)
    plt.imshow((x_test_np[idx] * 255).astype('uint8'), cmap='gray')
    plt.title(f"Pred: {classes[y_pred_classes[idx]]}\nPrava: {classes[y_true_classes[idx]]}")
    plt.axis('off')

for i, idx in enumerate(incorrect_sample):
    plt.subplot(2, 5, i+6)
    plt.imshow((x_test_np[idx] * 255).astype('uint8'), cmap='gray')
    plt.title(f"Pred: {classes[y_pred_classes[idx]]}\nPrava: {classes[y_true_classes[idx]]}")
    plt.axis('off')

plt.tight_layout()
plt.show()
```