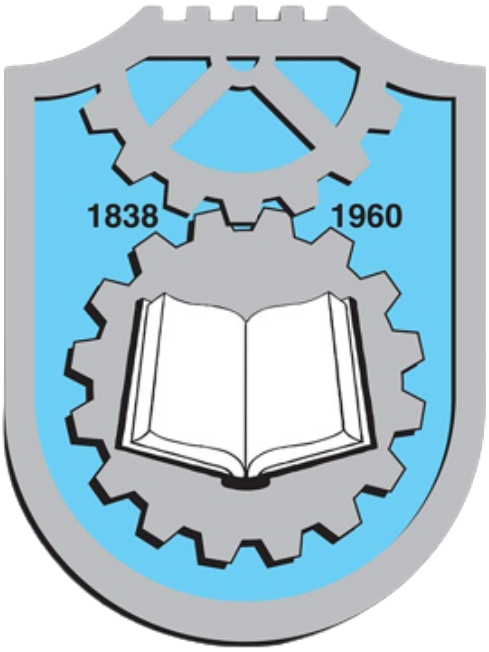
Универзитет у Крагујевцу
Факултет инжењерских наука



Неуронске мреже

Документација

Други домаћи задатак
Класификација саобраћајних знакова

Професор:
Проф. др Весна Ранковић

Студент:
Каришић Ђорђе 393/2023

__.__.2023.

Садржај

1 Увод

Рачунарски вид, као савремено поље истраживања чији је спектар примене неограничен и не дефинише се индустријом или делатношћу, може наћи своју примену у оквиру интелигентних система чији је циљ надзор окружења и адекватан одговор на промене у истом.
Пример таквог окружења може бити пут на којем се креће аутомобил, док један од задатака интелигентног система може бити посматрање непосредне околине аутомобила, конкретно саобраћајних знакова, стања пута или осталих аутомобила, и слање сигнала за обављање одређене радње главном систему у зависности од промене окружења.
Систем који је потребно развити мора бити временски ефикасан - како би систем могао да ради у реалном времену, као и веома прецизан - како би систем могао да генерише и проследи тачан сигнал ка главном систему. У конкретном примеру са аутомобилом, важно је да аутомобил, као главни систем, у сваком тренутку апсолутно познаје своје окружење и адекватно реагује на њега, како би могао да направи добру одлуку (уколико се налази ауто у стационарном стању испред њега, послати сигнал за кочење, уколико се испред њега налази знак стоп, потребно је да се заустави испред истог). Наведени услови могу бити испуњени употребом метода дефинисаних у оквиру гране дубоког учења, попут неуронских мрежа са задатком класификације објеката из окружења.
Овај рад се бави реализацијом једног таквог система, који на основу прослеђене слике саобраћајног знака, може прецизно и временски ефикасно да на излазу да сигнал који указује на то који саобраћајни знак се налази са слици. Потребно је обратити пажњу на разноврсност саобраћајних знакова, као и на физичко стање и дистанцу знака са слике, који представљају реалне факторе који могу донети потенцијалне проблеме у оваквом систему.
Приступ развоју интелигентног система за дати задатак препоручен овим радом јесте развој конволуционе неуронске мреже са задатком екстракције и обраде карактеристика са слика саобраћајних знакова - конкретно, развој цевовода који ће за дату улазну слику вршити трансформације над њом, проследити адекватном моделу или ансамблу модела конволуционих неуронских мрежа који ће такву трансформисану слику детаљно обрадити и на излазу дати инфромацију о којем се саобраћајном знаку ради.
Претходно развоју система, потребно је познавати податке са којима систем ради, тј. скуп слика саобраћајних знакова над којима ће се модел конволуционе неуронске мреже система обучавати, како би се приметили потенцијални проблеми, шаблони и обрасци, или карактеристике у оквиру тог скупа података слика.

2 Реализација система

Систем реализован овим задатком је подељен у више делова:

- Претпроцесирање података
- Анализа података и резултата
- Дефинисање модела
- Имплементација система класификације

2.1 Препроцесирање података

Формат слика из скупа података није подржан од стране библиотека потребних за сврхе система. Исечком кода 1 дата је функционалност конверзије слика.

```
from PIL import Image
import os
root_dir = "../Dataset"
train_dir, test_dir = [os.path.join(os.path.dirname(__file__), root_dir, x) for x in ['Training', 'Testing']]
def convert_and_save_images_in_folder(folder_path):
    for subdir, dirs, files in os.walk(folder_path):
        for file in files:
            if file.endswith(".ppm"):
                ppm_path = os.path.join(subdir, file)
                image = Image.open(ppm_path)
                png_file = os.path.splitext(file)[0] + ".png"
                png_path = os.path.join(subdir, png_file)
                image.save(png_path)
convert_and_save_images_in_folder(train_dir)
convert_and_save_images_in_folder(test_dir)
print("PPM to PNG conversion complete.")
```

Исечак кода 1: Конверзија слика из .ppm у .png формат

2.1.1 Класа Collector

Класа Collector садржи статичке методе које омогућавају прикупљање, груписање и читање слика. Ова класа зависи од библиотека дефинисаних у исечку 2.

```
import os
import numpy as np
from PIL import Image
import tensorflow as tf
```

Исечак кода 2: Библиотеке од којих Collector зависи

Исечак 3 садржи функцију за агрегацију и форматирање података како би се дошло до јасно дефинисаних скупова за обучавање и тестирање мреже.

```
@staticmethod
    def aggregate(root: str = None, target_size: tuple[int, int] = (32, 32)) -> tuple[tf.data.Dataset, tf.data.Dataset]:
        train_dir, test_dir = [os.path.join(os.path.dirname(__file__), root, x) for x in ['Training', 'Testing']]
        train_dataset = tf.keras.utils.image_dataset_from_directory(
            train_dir,
            image_size=target_size,
            batch_size=None,
            label_mode='categorical',
            seed=1337,
            crop_to_aspect_ratio=True)
        test_dataset = tf.keras.utils.image_dataset_from_directory(
            test_dir,
            image_size=target_size,
            shuffle=False,
            batch_size=None,
            label_mode='categorical',
            seed=1337,
            crop_to_aspect_ratio=True)
        train_dataset = train_dataset.map(lambda x, y: (x / 255.0, y))
        test_dataset = test_dataset.map(lambda x, y: (x / 255.0, y))
        return train_dataset, test_dataset
```

Исечак кода 3: Функција за агрегацију скупа података

Исечак 4 садржи функцију која ради исто што и 3, само задржава редослед тренинг података, и враћа их у другом формату, како би се лакше испитали.

```
    @staticmethod
    def aggregate_orig(root:str=None) -> tuple[np.ndarray, np.ndarray]:
        trainroot, testroot = (os.path.join(os.path.dirname(__file__), root, x) for x in ['Training','Testing'])
        train, test = [],[]
        train, _ = Collector.read_img(trainroot)
        test, _ = Collector.read_img(testroot)
        return train,test
```

Исечак кода 4: Функција за агрегацију слика скупа података у другом формату

Како се са исечка 4 може видети да функција дефинисана у оквиру тог исечка зависи од друге функције, потребно ју је дефинисати. Исечак 5 приказује функционалност система за читање података и креирање скупова података. Ова функција може радити са .ppm фајловима.

```python
    @staticmethod
    def read_img(subroot:str=None)->tuple[np.ndarray,np.ndarray]:
        print(f"Reading data from {subroot}")
        data = []
        labels = []
        for class_name in os.listdir(subroot):
            print(f"Current directory: {class_name}")
            if not os.path.isdir(os.path.join(subroot, class_name)):
                continue
            class_dir = os.path.join(subroot, class_name)
            for filename in os.listdir(class_dir):
                if not filename.endswith('.ppm'):
                    continue
                filepath = os.path.join(class_dir,filename)
                label = class_name.lstrip('0')
                data.append((np.array(Image.open(filepath)).astype('float32')))
                labels.append(label.zfill(1))
        return data, labels
```

Исечак кода 5: Функција за читање слика и креирање скупова података од њих

Попут функције у оквиру 4, постоји и функција за искључиво издвајање информација о класама којим подаци припадају. Дефинисана је исечком 6.

```python
    @staticmethod
    def aggregate_labels(root:str=None) -> tuple[np.ndarray, np.ndarray]:
        trainroot, testroot = (os.path.join(os.path.dirname(__file__), root, x) for x in ['Training','Testing'])
        _, train_labels = Collector.read_img(trainroot)
        _, test_labels = Collector.read_img(testroot)
        return train_labels,test_labels
```

Исечак кода 6: Функција за издвајање класа којима припадају слике из скупа

Класом Collector омогућава се читање *raw* слика и њихових класа, форматирање и енкапсулација у објекат *tf.Dataset* за обучавање и тестирање мреже, као и претпроцесирање скупа података са задржавањем редоследа и формата ради њихове анализе.
Класа Collector, као статичка класа, не садржи ниједно поље, већ само статичке методе дефинисане претходним исечцима кода, што омогућава лаку и широку употребу ове класе у оквиру система.

2.1.2 Класа Analyzer

Класа Analyzer се састоји из метода чија је сврха анализа и графичко приказивање веза и образаца између слика у оквиру скупа података. Овим приступом могу се приметити и адекватно решити потенцијални проблеми који се јављају у оквиру скупа, као и разумети сложеност индивидуалних података или целокупног скупа, која директно указује на захтеве сложености архитектуре мреже.

Исечком 7 дате су библиотеке од којих ова класа зависи и конструктор класе. Исечком 8 дата је функција за графички приказ бројева примерака који припадају свакој од класа.

```
import random
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from matplotlib.gridspec import GridSpec
import tensorflow as tf
from collector import Collector
class Analyzer():
    def __init__(self) -> None:
            self.cmap = plt.get_cmap('twilight')
            self.root = "../Dataset"
            self.train, self.test = Collector.aggregate_orig(self.root)
            self.train_labels,self.test_labels = Collector.aggregate_labels(self.root)
```

Исечак кода 7: Библиотеке од којих класа Analyzer зависи и конструктор

```
def class_distribution_plot(self, data_array: np.ndarray) -> None:
        class_counts = {}
        for labels in list(map(int, data_array)):
            if labels not in class_counts:
                class_counts[labels] = 1
            class_counts[labels] += 1
        sorted_class_counts = sorted(class_counts.items(), key=lambda item: item[0])
        classes, counts = zip(*sorted_class_counts)
        plt.figure(figsize=(15, 5))
        plt.bar(classes, counts, color=self.cmap(300))
        plt.xlabel('Класа')
        plt.ylabel('Број слика')
        plt.title('Расподела слика по класама')
        plt.xticks(ticks=classes, labels=classes, fontsize=7)
        plt.savefig('plots/class_distribution_plot.png') #plt.savefig('plots/class_distribution_plot_test.png')
        plt.close()
```

Исечак кода 8: Функција за графички приказ расподеле података по класама

Исечак 9 садржи функцију за графички приказ расподеле величине сваког примерка слике, груписаних по више категорија величине слика. Категорије су направљене на основу промењиве *base*.

```
def size_distribution_plot(self) -> None:
        base = 25000
        data_array, _ = Collector.aggregate_orig(self.root)
        size_classes = [[] for _ in range(10)]
        def group_images_by_size(data_array):
            for i, image in enumerate(data_array):
                size = image.shape[0] * image.shape[1]
                for j in range(10):
                    if size < (j + 1) * base:
                        size_classes[j].append(size)
                        break
                else:
                    size_classes[9].append(size)
        group_images_by_size(data_array)
        plt.figure(figsize=(10, 5))
        for class_num in range(10):
            plt.subplot(2, 5, class_num + 1)
            sizes_in_class = size_classes[class_num]
            plt.hist(sizes_in_class, bins=20, edgecolor='k', color=self.cmap(300))
            plt.xticks(fontsize=7)
            plt.xlabel('Број пиксела слике')
            plt.ylabel('Број слика')
            if class_num != 9:
                plt.title(f"Од {class_num * base} до {(class_num + 1) * base} пиксела", fontsize=7)
            else:
                plt.title(f"Веће од {class_num * base} пиксела", fontsize=7)
        plt.tight_layout()
        plt.savefig('plots/size_distribution_plot.png')
        plt.close()
```

Исечак кода 9: Функција за графички приказ расподеле података по величини

Исечак 10 садржи функцију за графички приказ слика више различитих класа, као и функцију за креирање топлотних мапа на основу канала боја слика.

```python
def image_show(self, data_array: np.ndarray, seed: int = 3, num: int = 5) -> None:
    random.seed(seed)
    num_total = len(data_array)
    indices = random.sample(range(num_total), num)
    plt.figure(figsize=(12, 4))
    for i, idx in enumerate(indices):
        image_data = tf.image.resize(data_array.element_spec[0].shape, (32, 32))
        label = int(data_array.element_spec[1].shape)
        plt.subplot(1, num, i + 1)
        plt.imshow(image_data.numpy().astype(np.uint8))
        plt.title(f'Класа: {label}')
    plt.tight_layout()
    plt.savefig('plots/image_show.png')
def pixel_heatmap(self,data_array:np.ndarray, seed:int=3)->None:
    random.seed(seed)
    idx = random.randint(0,len(self.train)-1)
    image_data = self.train[idx]
    label = self.train_labels[idx]
    ch_r,ch_g,ch_b = image_data[:,:,0],image_data[:,:,1],image_data[:,:,2]
    plt.figure(figsize=(16, 4))
    gs = GridSpec(1, 3, width_ratios=[1, 1, 1], wspace=0.1)
    plt.subplot(gs[0])
    sns.heatmap(ch_r, cmap='Reds', square=True, annot=False)
    plt.title(f"Матрица корелације за црвену боју",fontsize=10)
    plt.subplot(gs[1])
    sns.heatmap(ch_g, cmap='Greens', square=True, annot=False)
    plt.title(f"Матрица корелације за зелену боју",fontsize=10)
    plt.subplot(gs[2])
    sns.heatmap(ch_b, cmap='Blues', square=True, annot=False)
    plt.title(f"Матрица корелације за плаву боју",fontsize=10)
    plt.suptitle(f"Матрица корелације за примерак из класе {label}")
    plt.savefig('plots/pixel_heatmap.png')
    plt.close()
```

Исечак кода 10: Функција за графички приказ слика различитих класа и њихових топлотних мапа

Исечак 11 садржи функцију за графички приказ слике са најмањим, и слике са највећим димензијама, као и како те слике изгледају након промене величине.

```
def min_max_size(self,target_size:tuple[int,int])->None:
        images = Collector.aggregate_orig(self.root)[0]
        sizes = np.array([data.shape[0] * data.shape[1] for data in images])
        i_min, i_max = np.argmin(sizes), np.argmax(sizes)
        smallest_image = images[i_min] / 255.0
        largest_image = images[i_max] / 255.0
        plt.figure(figsize=(12, 4))
        plt.subplot(121)
        plt.imshow(smallest_image)
        plt.title("Најмања слика")
        plt.xlabel(f"{smallest_image.shape[1]}")
        plt.ylabel(f"{smallest_image.shape[0]}")
        plt.subplot(122)
        plt.imshow(largest_image)
        plt.title("Највећа слика")
        plt.xlabel(f"{largest_image.shape[1]}")
        plt.ylabel(f"{largest_image.shape[0]}")
        plt.savefig('plots/size_comparison.png')
        smallest_image_resized = tf.image.resize(smallest_image, target_size)
        largest_image_resized = tf.image.resize(largest_image, target_size)
        plt.figure(figsize=(12, 4))
        plt.subplot(121)
        plt.imshow(smallest_image_resized)
        plt.title("Најмања слика са промењеним димензијама")
        plt.xlabel(f"{smallest_image_resized.shape[1]}")
        plt.ylabel(f"{smallest_image_resized.shape[0]}")
        plt.subplot(122)
        plt.imshow(largest_image_resized)
        plt.title("Највећа слика са промењеним димензијама")
        plt.xlabel(f"{largest_image_resized.shape[1]}")
        plt.ylabel(f"{largest_image_resized.shape[0]}")
        plt.savefig('plots/resized_comparison.png')
        plt.close()
```

Исечак кода 11: Функција за графички приказ слика најмање и највеће величине у оригиналном формату и у *resize* формату

2.1.3 Класа Model

Класа Model представља *wrapper* класу, која вештачки имплементира већину метода класе *tensorflow.keras.models.Sequential*, и тиме постиже ефекат креирања специфичног, специјализованог модела мреже са додатним методама.
Ова класа зависи од библиотека и класа дефинисаних исечком 12.

```
from matplotlib import pyplot as plt
import numpy as np
from sklearn.metrics import ConfusionMatrixDisplay, classification_report, confusion_matrix
import tensorflow as tf
from tensorflow.keras.layers import Conv2D, BatchNormalization, Activation, MaxPooling2D, Dense,\
                                    Flatten, Dropout, Input
from tensorflow.keras.models import Sequential
from tensorflow.keras import optimizers
from tensorflow.keras.regularizers import l1, l2
from PIL import ImageFont
import visualkeras
from tensorflow.keras.utils import plot_model
from tensorflow.keras.callbacks import EarlyStopping
import seaborn as sns
from typing import Self
from backend.collector import Collector
```

Исечак кода 12: Библиотеке и класе од којих класа Model зависи

Конструктор ове класе, као и поља класе дати су исечком 13.

```
    def __init__(self,target_size:tuple[int,int]=(32,32)) -> None:
        self.root = "../Dataset"
        self.train, self.test_dataset = Collector.aggregate(self.root, target_size=target_size)
        num_samples = sum(1 for _ in self.train)
        train_size = int(0.8 * num_samples)
        self.train_dataset = self.train.take(train_size)
        self.validation_dataset = self.train.skip(train_size)
        self.train_dataset = self.train_dataset.batch(32)
        self.validation_dataset = self.validation_dataset.batch(32)
        self.test_dataset = self.test_dataset.batch(32)
        self.name=f"model_{target_size[0]}x{target_size[1]}"
        self.model = Sequential(name=self.name)
        self.model.add(Input(shape=(target_size[0],target_size[1],3), name="input_layer"))
```

Исечак кода 13: Библиотеке и класе од којих класа Model зависи

Исечком 14 дате су функције за очитавање постојећег модела и за графичко исцртавање модела, које се врши на два начина.

```
    def load_model(self)->Self:
        try:
            self.model = tf.keras.models.load_model(f"models/{self.name}.keras")
        except:
            print("Unexistant model.")
        return self
    def plot(self):
        plot_model(self.model, to_file=f"plots/{self.model.name}_g.png", show_shapes=True,\
                                                    show_layer_names=True)
        font = ImageFont.truetype("arial.ttf", 12)
        visualkeras.layered_view(self.model, legend=True, font=font,to_file=f"plots/{self.model.name}_a.png")
```

Исечак кода 14: Функције за очитавање постојећег модела и графичко представљање

Функције за дефинисање основних конволуционих блокова, тј. архитектуре дела мреже за екстракцију карактеристика дате су исечком 15

```
    def inputnet_conv(self,filters:int=32,kernel_size:int=7, padding:str='valid',pool_size:int=2,\
                      dropout_rate:float=0.25, l1_reg:float=0.01, l2_reg:float=0.01)->Self:
        self.model.add(Conv2D(filters, kernel_size=(kernel_size, kernel_size), padding=padding,\
        activation='relu', kernel_regularizer=l2(l2_reg), bias_regularizer=l1(l1_reg)))
        self.model.add(BatchNormalization())
        self.model.add(MaxPooling2D(pool_size=pool_size))
        self.model.add(Dropout(dropout_rate))
        return self
    def midnet_conv(self,filters:int=32,kernel_size:int=3, padding:str='valid',pool_size:int=2,\
                    dropout_rate:float=0.25, l1_reg:float=0.01, l2_reg:float=0.01)->Self:
        self.model.add(Conv2D(filters, kernel_size=(kernel_size, kernel_size), padding=padding,\
        activation='relu', kernel_regularizer=l2(l2_reg), bias_regularizer=l1(l1_reg)))
        self.model.add(Conv2D(filters, kernel_size=(kernel_size, kernel_size), padding=padding,\
        activation='relu', kernel_regularizer=l2(l2_reg), bias_regularizer=l1(l1_reg)))
        self.model.add(BatchNormalization())
        self.model.add(MaxPooling2D(pool_size=pool_size))
        self.model.add(Dropout(dropout_rate))
        return self
```

Исечак кода 15: Функције за дефинисање конволуционих блокова

Функције за изравњавање мапе карактеристика и за дефинисање основних потпуно повезаних блокова, тј. архитектуре дела мреже за обраду карактеристика дате су исечком 16.

```
    def midnet_flatten(self)->Self:
        self.model.add(Flatten())
        return self

    def fc_block(self, units:int,activation:str='relu',dropout_rate:float=0.3,\
                 l1_reg:float=0.01, l2_reg:float=0.01)->Self:
        self.model.add(Dense(units, activation=activation, kernel_regularizer=l2(l2_reg), bias_regularizer=l1(l1_reg)))
        self.model.add(BatchNormalization())
        self.model.add(Activation(activation))
        self.model.add(Dropout(dropout_rate))
        return self

    def outnet_block(self)->Self:
        self.model.add(Dense(62,activation='softmax'))
        return self
```

Исечак кода 16: Функције за изравњавање мапа и дефинисање потпуно повезаних блокова

Функција за компајлирање модела мреже дата је исечком 17.

```
def compile(self, optimizer:str = 'adam', loss:str = 'categorical_crossentropy',\
                metrics: list[str] = ['accuracy','categorical_crossentropy'], learning_rate:float = 0.001)->Self:
        optimizer = optimizers.get(optimizer)
        optimizer.learning_rate.assign(learning_rate)
        self.model.compile(optimizer=optimizer, loss=loss, metrics=metrics)
        return self
```

Исечак кода 17: Функција за компајлирање модела мреже

Функција за чување модела мреже је дата исечком 18

```
def save(self)->None:
        self.model.save(f"models/{self.model.name}.keras")
```

Исечак кода 18: Функција за чување модела мреже

Функција за тренирање мреже над скупом података и генерисање графичком и текстуалног извештаја о резултатима обучавања дата је исечком 19.

```
def fit(self, batch_size: int = 32, epochs: int = 100) -> any:
    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
    reduce_lr_callback = ReduceLROnPlateau(monitor='val_loss',factor=0.5,patience=3,min_lr=1e-6)
    history = self.model.fit(self.train_dataset, validation_data=self.validation_dataset,epochs=epochs,
    batch_size=batch_size, callbacks=[early_stopping, reduce_lr_callback])
    train_accuracy, val_accuracy = history.history['accuracy'], history.history['val_accuracy']
    learning_rate = history.history['lr']
    fig, ax = plt.subplots(nrows=2, ncols=1, figsize=(12, 10))
    ax[0].set_title('Тачност по епохама')
    ax[0].plot(train_accuracy, 'o-', label='Тренинг - тачност')
    ax[0].plot(val_accuracy, 'o-', label='Валидација - тачност')
    ax[0].set_xlabel('Епоха')
    ax[0].set_ylabel('Тачност')
    ax[0].legend(loc='best')
    ax[1].set_title('Стопа учења по епохама')
    ax[1].plot(learning_rate, 'o-', label='Стопа учења')
    ax[1].set_xlabel('Епоха')
    ax[1].set_ylabel('Губитак')
    ax[1].legend(loc='best')
    plt.tight_layout()
    plt.savefig(f'plots/{self.name}_report.png')
    return history
```

Исечак кода 19: Функција за чување модела мреже

Исечак кода 20 приказује функцију за графичко представљање учинка мреже над тестним скупом.

```
def evaluate(self)->np.ndarray:
    test_loss, test_accuracy, _ = self.model.evaluate(self.test_dataset)
    plt.figure()
    labels, values = ['Тест - губитак', 'Тест - тачност'], [test_loss, test_accuracy]
    plt.bar(labels, values, color=['red','green'])
    plt.title('Тест - губитак и тачност')
    plt.savefig(f"plots/{self.name}_testacc.png")
```

Исечак кода 20: Функција за графичко представљање учинка мреже

Функција за предвиђање класа на основу слика из тестног скупа и генерисање извештаја и матрице конфузије на основу учинка је дефинисана у оквиру исечка 21.

```
    def predict(self)->np.ndarray:
        predictions = self.model.predict(self.test_dataset)
        self.test_dataset = self.test_dataset.unbatch()
        images, labels = tuple(zip(*self.test_dataset))
        decoded_labels = tf.argmax(labels, axis=-1)
        decoded_pred = tf.argmax(predictions, axis=-1)
        orig = np.array(decoded_labels)
        confusion = confusion_matrix(decoded_labels, decoded_pred,labels=range(0,62))
        self.actual_out = orig
        plt.figure(figsize=(8, 6))
        sns.set(font_scale=0.6)
        sns.heatmap(confusion, annot=True, fmt='d', cmap='twilight', cbar=True)
        plt.xlabel("Претпостављено")
        plt.ylabel("Стварно")
        plt.savefig(f'plots/confm_{self.name}.png')
        report = classification_report(decoded_labels, decoded_pred)
        with open(f'plots/classification_report_{self.name}.txt', 'w') as report_file:
            report_file.write(report)
        return np.array(decoded_pred)
```

Исечак кода 21: Функција за предвиђање класа слика из тестног скупа

2.2 Имплементација и интерфејс

Идеја иза овог рада јесте креирање ансамбл више модела са различитим улазним димензијама како би се побољшао учинак над тестним скупом. Направљена су пет мдоела, са различитим димензијама улаза, и њихови излази биће подвргнути тзв. *гласању*, помоћу којег ће се одабрати најистакнутији излаз за сваки улазни податак. Исечак 22 приказује библиотеке од којих зависи имплементација.

```
from collections import Counter
from matplotlib import pyplot as plt
import seaborn as sns
import tensorflow as tf
from backend.model import Model
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

Исечак кода 22: Библиотеке неопходне за дефинисање интерфејса

Иницијализација, обучавање и евалуација модела, уз помоћ Model класе која је креирана тако да имплементира модификовани Builder образац, се може извршити као на исечку кода 23.

```
def create_model():
    wrapper = Model(target_size=(128,128))
    wrapper\
    .midnet_conv()\
    .midnet_conv(filters=64)\
    .midnet_flatten()\
    .fc_block(512)\
    .outnet_block()\
    .compile()\
    .plot()
    wrapper.fit(epochs=100)
    wrapper.save()
    wrapper.evaluate()
    pred = wrapper.predict()

    accuracy = accuracy_score(wrapper.actual_out,pred)
    report = classification_report(wrapper.actual_out,pred)
    print(accuracy, report)
```

Исечак кода 23: Иницијализација, тренинг и евалуација модела уз помоћ Model класе

Иницијализација претходно сачуваног модела се може извршити као на исечку 24, уз помоћ променљиве која указује на димензије улазних података, што једнозначно указује на јединствен модел (како се сваки модел разликује искључиво по димензији улазних података које обрађује).

```
def load_model():
    mod = Model(target_size=(32,32)).load_model()
    mod.evaluate()
    pred = mod.predict()
    accuracy = accuracy_score(mod.actual_out,pred)
    report = classification_report(mod.actual_out,pred)
    print(accuracy, report)
```

Исечак кода 24: Иницијализација претходно сачуваног модела са димензијама 32×32

Комплетно решење, које подразумева ансамбл више модела и његову имплементацију, дато је исечком кода 25. У оквиру овог кода, генерише се текстуални извештај евалуације мреже, као и графички приказ матрица конфузија, како би се резултати додатно протумачили. Пролази се кроз излазе свих модела и креира се листа класа предложених од стране сваког модела за сваки улазни податак, након чега се изолује листа тих предложених класа за сваки улаз, и бира најучесталија класа међу предложеним за дати улаз. Изабрана вредност се уписује у нову листу предложених класа, и пореди са стварним класама.

```python
def main():
    models = []
    names=['model_32x32','model_48x48','model_64x64','model_72x72','model_128x128']
    sizes=[32,48,64,72,128]
    for i,val in enumerate(names):
        print(f"Initializing {val} for dataset w/ size ({sizes[i]},{sizes[i]})")
        models.append(Model(target_size=(sizes[i],sizes[i])).load_model())
    all_predictions = [model.predict() for model in models]
    ensemble_predictions = []
    for i in range(len(models[0].actual_out)):
        input_predictions = [predictions[i] for predictions in all_predictions]
        most_common_prediction = Counter(input_predictions).most_common(1)[0][0]
        ensemble_predictions.append(most_common_prediction)
    accuracy = accuracy_score(models[0].actual_out,ensemble_predictions)
    report = classification_report(models[0].actual_out,ensemble_predictions)
    confusion = confusion_matrix(models[0].actual_out,ensemble_predictions,labels=range(0,62))
    plt.figure(figsize=(8, 6))
    sns.set(font_scale=0.6)
    sns.heatmap(confusion, annot=True, fmt='d', cmap='twilight', cbar=True)
    plt.xlabel("Претпостављено")
    plt.ylabel("Стварно")
    plt.savefig(f'plots/confm_ensemble.png')
    with open(f'plots/classification_report_ensemble.txt', 'w') as report_file:
        report_file.write(report)
```

Исечак кода 25: Имплементација ансамбл решења задатка класификације

2.3 Зависности пројекта и виртуелно окружење

Сви фактори од којих овај пројекат зависи, попут библиотека, су запаковани и сачувани у оквиру виртуелног окружења, са верзијама које су употребљене у оквиру овог пројекта.
Верзије библиотека се могу пронаћи у оквиру **requirements.txt** текстуалног фајла у **root** директоријуму пројекта.

3 Резултати

Резултати, представљени графички или текстуално подељени су у два сегмента:

- Резултати анализе података
- Резултати имплементације система

3.1 Резултати анализе података

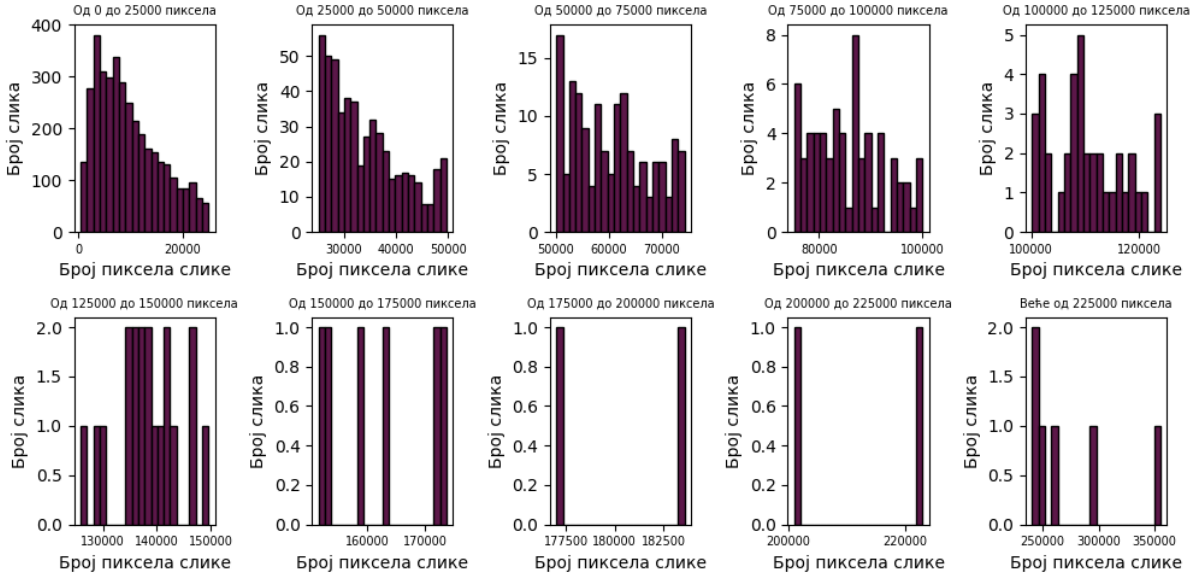
Слике 1 и 2 приказују резултате примене кода 8.

Слика 1: Расподела класа у тренинг скупу

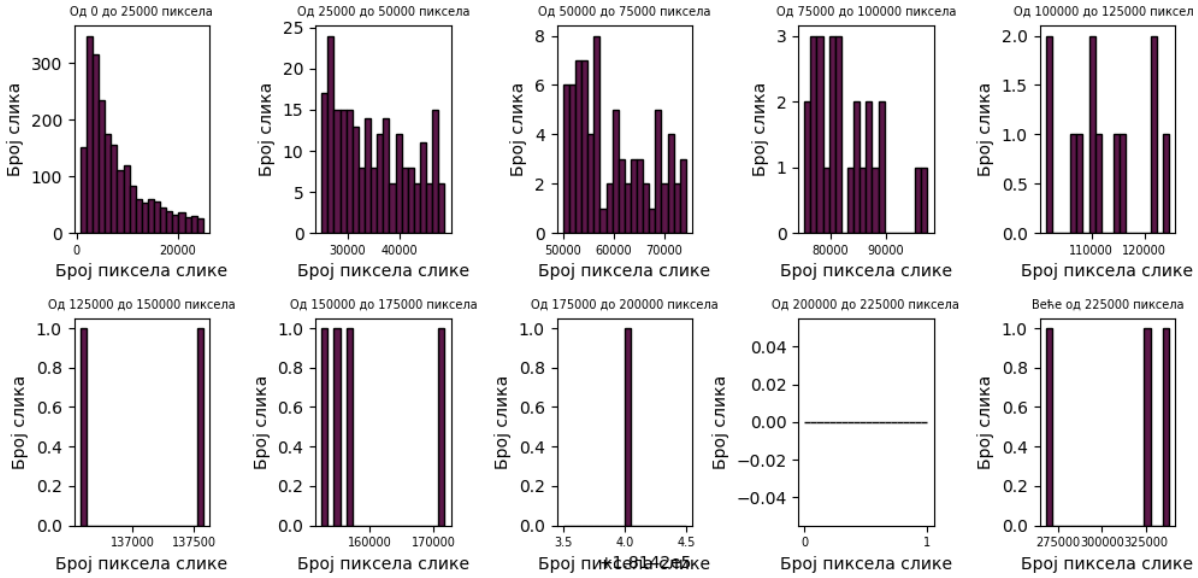
Слика 2: Расподела класа у тестном скупу

Скуп података није балансиран, што предлаже настанак потешкоћа при класификацији података који припадају класама које се јављају у малом броју.

Слике 3 и 4 приказују резултате примене кода 9.



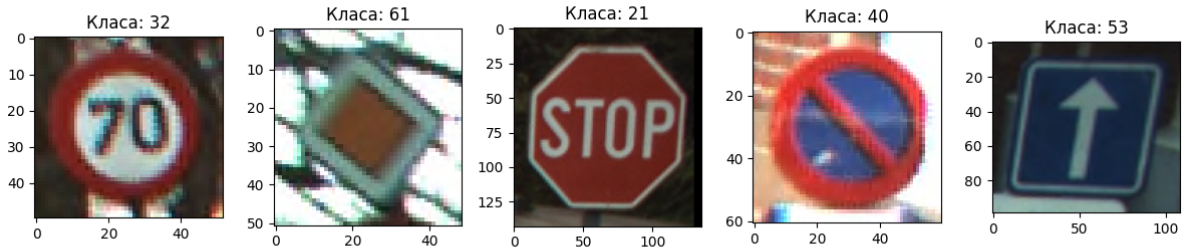
Слика 3: Расподела слика по величини у тренинг скупу



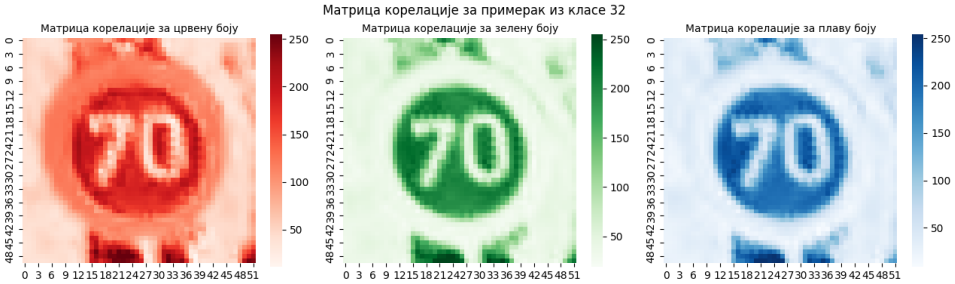
Слика 4: Расподела слика по величини у тестном скупу

Величине слика доста варирају у скупу података. Потребно им је променити величину и притом задржати однос ширине и висине, као што је урађено у сегменту кода 3. Овим приступом се врши унификација свих слика по величини а притом очување оригиналног односа ширине и висине.

Слике 5 и 6 приказују резултате примене исечка кода 10.

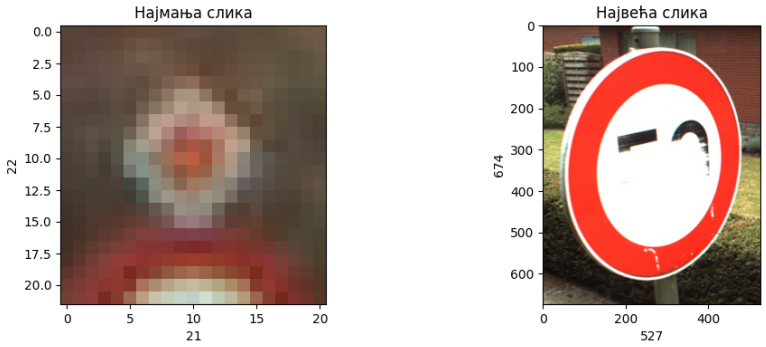


Слика 5: Приказ насумичних слика из скупа података

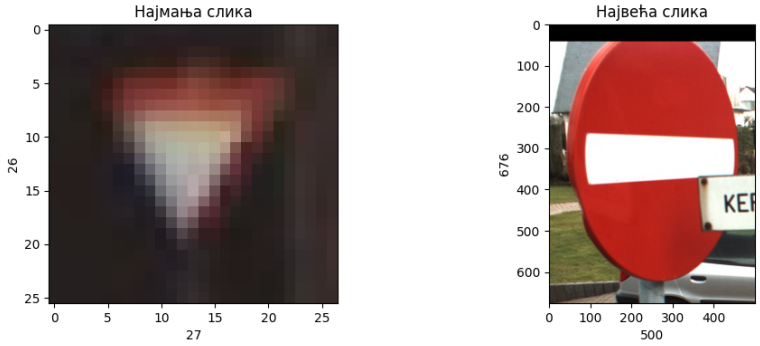


Слика 6: Приказ топлотних мапа насумичног примерка

На основу слике 6 примећује се да су боје које се налазе на сликама јако битне, и да их је потребно сачувати како би се обрадиле у мрежи.
Сликама 7 и 8 приказан је резултат функције дефинисан сегментом кода 11. Како су разлике у величини слика огромне, потребно је проверити како ће те слике изгледати након промене величине, што је приказано сликама 9 и 10.



Слика 7: Најмања и највећа слика у тренинг скупу



Слика 8: Највећа и најмања слика у тестном скупу

Слика 9: Најмања и највећа слика у тренинг скупу са промењеним димензијама

Слика 10: Највећа и најмања слика у тестном скупу са промењеним димензијама

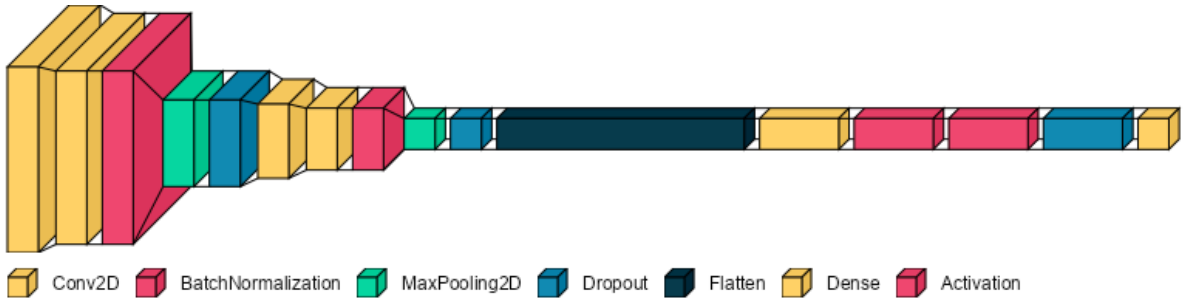
Може се закључити да су сликама из скупа података правилно промењене димензије, и да су важне информације сачуване.

3.2 Резултати имплементације система

Тумачење резултата, тј. учинка имплементације система дефинисаног у оквиру овог рада се може вршити у више корака, тумачењем индивидуалних резултата свих компоненти које га чине, а то су:

- Модел са улазом 32×32
- Модел са улазом 48×48
- Модел са улазом 64×64
- Модел са улазом 72×72
- Модел са улазом 128×128
- Ансамбл претходних модела

Сликом 11 приказана је генерална архитектура конволуционе неуронске мреже предложена у оквиру овог рада, која ће бити основа за сваки од претходно наведених модела.

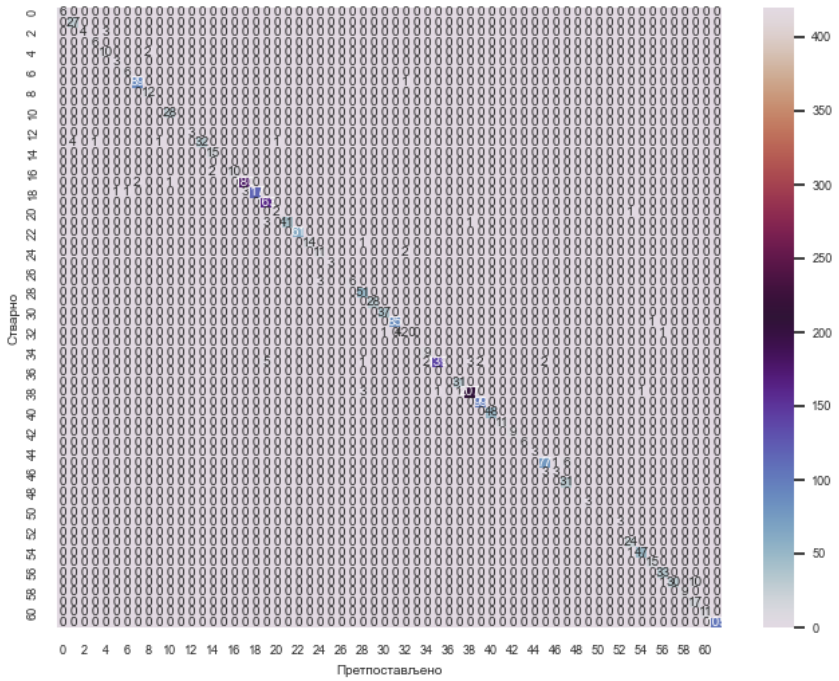


Слика 11: Архитектура модела

Сваки модел ће бити објекат класе Model која је дефинисана овим радом, и једина разлика између њих ће бити димензија улазног слоја, тј. димензије улазних података које они треба да анализирају.
Ансамбл ће представљати скуп пет претходно наведених модела, који ће на излазу дати листу предвиђених класа на основу предвиђања сваког модела, тако што ће се изабрати најчешће решење.
Евалуација учинка модела ће се вршити испитивањем прецизности, сензитивности и F1 мере за сваку од класа, као и посматрањем њихових средњих вредности.

3.3 Модел са улазом 32×32

Матрица конфузије је дата сликом 12, док је извештај о класификацији дат табелом 1.



Слика 12: Матрица конфузије примене модела са улазом 32×32

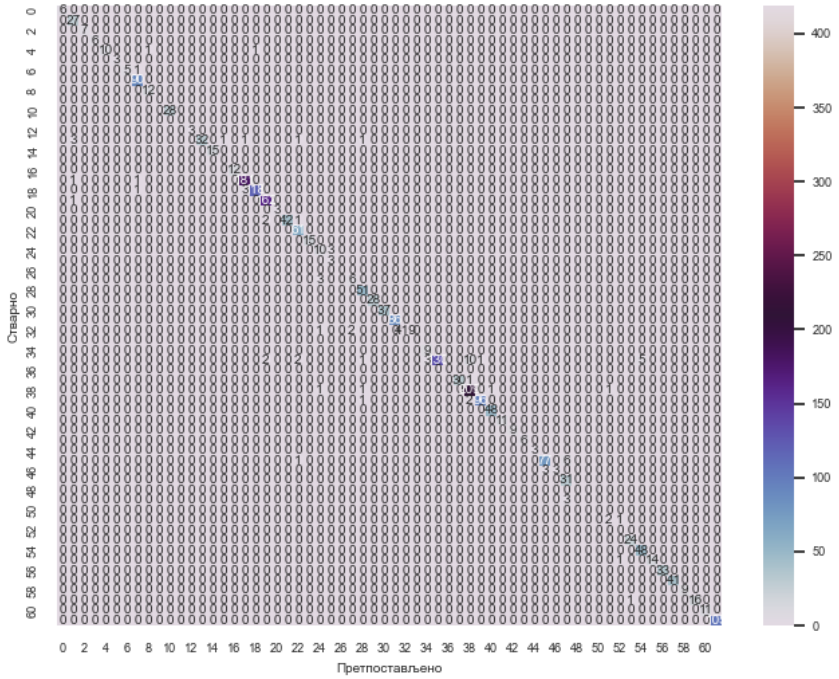
Табела 1: Извештај о класификацији

Класа	Прецизност	Сенз.	F1 учинак
0	1.00	1.00	1.00
1	0.87	1.00	0.93
2	1.00	0.57	0.73
3	0.86	1.00	0.92
	...		
58	1.00	1.00	1.00
59	0.63	1.00	0.77
60	1.00	1.00	1.00
61	1.00	1.00	1.00
$\overline{avg}$	0.97	0.97	0.97

Овај модел мреже постиже јако добре резултате, на основу извештаја и матрице конфузије.

3.4 Модел са улазом 48×48

Матрица конфузије је дата сликом 13, док је извештај о класификацији дат 2.



Слика 13: Матрица конфузије примене модела са улазом 48×48

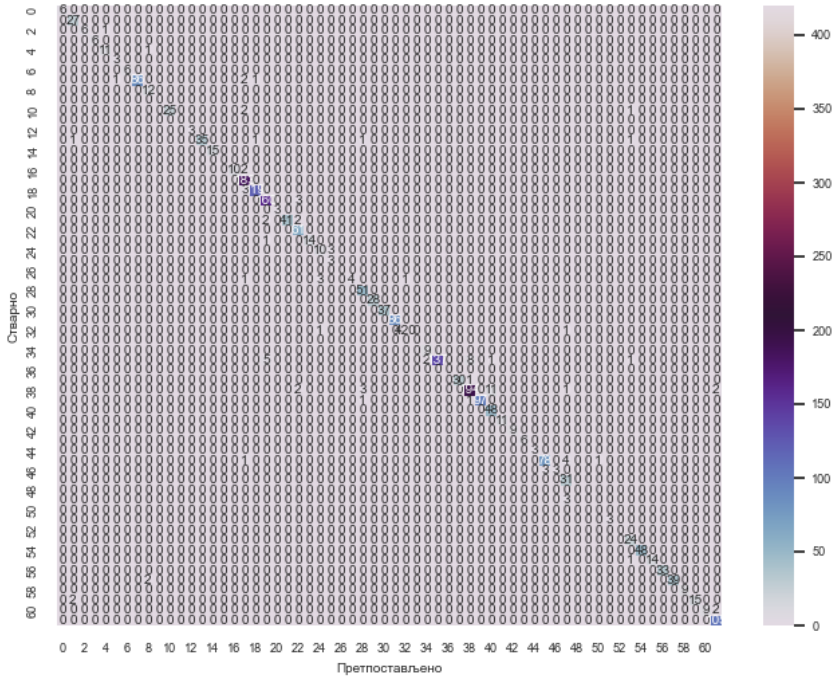
Табела 2: Извештај о класификацији

Класа	Прецизност	Сенз.	F1 учинак
0	1.00	1.00	1.00
1	0.84	1.00	0.92
2	1.00	1.00	1.00
3	1.00	1.00	1.00
	...		
58	1.00	1.00	1.00
59	1.00	0.94	0.97
60	1.00	1.00	1.00
61	1.00	1.00	1.00
$\overline{avg}$	0.97	0.97	0.97

Овај модел мреже такође остварује одличне резултате, на основу извештаја и матрице конфузије.

3.5 Модел са улазом 64×64

Матрица конфузије је дата сликом 14, док је извештај о класификацији дат 4.



Слика 14: Матрица конфузије примене модела са улазом 64×64

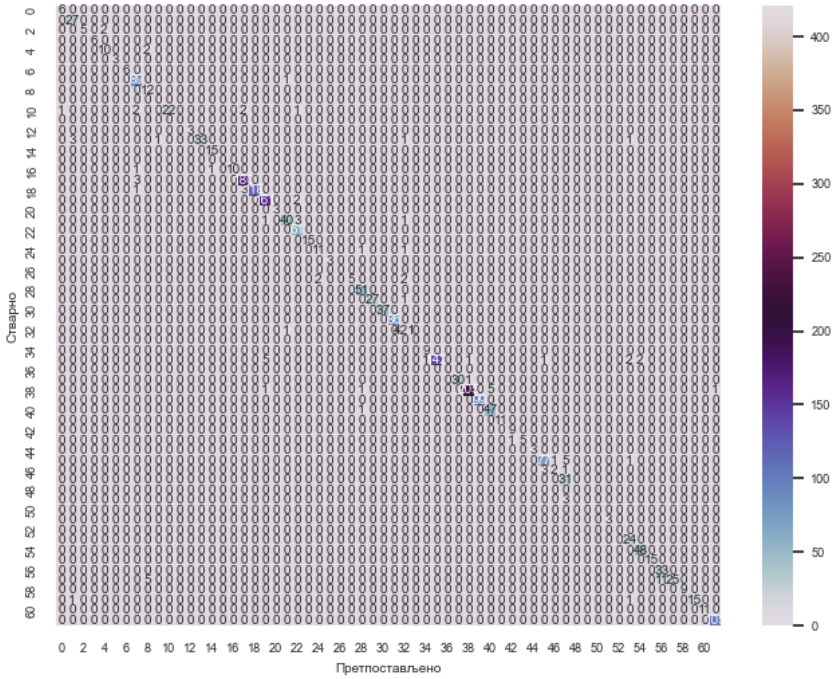
Табела 3: Извештај о класификацији

Класа	Прецизност	Сенз.	F1 учинак
0	1.00	1.00	1.00
1	0.90	1.00	0.95
2	1.00	0.86	0.92
3	1.00	1.00	1.00
	...		
58	1.00	1.00	1.00
59	1.00	0.88	0.94
60	1.00	0.82	0.90
61	0.96	1.00	0.98
$\overline{avg}$	0.97	0.96	0.96

Овај модел мреже такође остварује одличне резултате, на основу извештаја и матрице конфузије. Може се приметити да модели са улазима 64×64, 48×48 и 32×32 имају пар разлика, што се тиче метрика евалуација.
Ово је јако битно јер су за ансамбл потребни модели који се разликују, што доноси разноврсност у ансамбл.

3.6 Модел са улазом 72×72

Матрица конфузије је дата сликом 15, док је извештај о класификацији дат 4.



Слика 15: Матрица конфузије примене модела са улазом 72×72

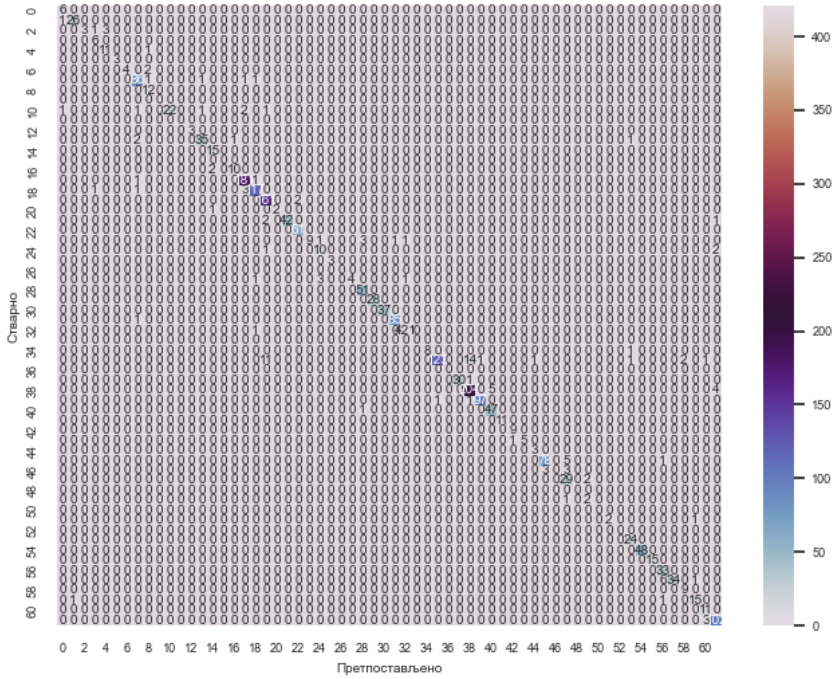
Табела 4: Извештај о класификацији

Класа	Прецизност	Сенз.	F1 учинак
0	0.86	1.00	0.92
1	0.87	1.00	0.93
2	1.00	0.71	0.83
3	1.00	1.00	1.00
	...		
58	1.00	1.00	1.00
59	1.00	0.88	0.94
60	1.00	1.00	1.00
61	0.99	1.00	1.00
$\overline{avg}$	0.96	0.96	0.96

Овај модел мреже остварује јако добре резултате, и такође се могу приметити исте разлике као и код претходних модела.

3.7 Модел са улазом 128×128

Матрица конфузије је дата сликом 16, док је извештај о класификацији дат 5.



Слика 16: Матрица конфузије примене модела са улазом 128×128

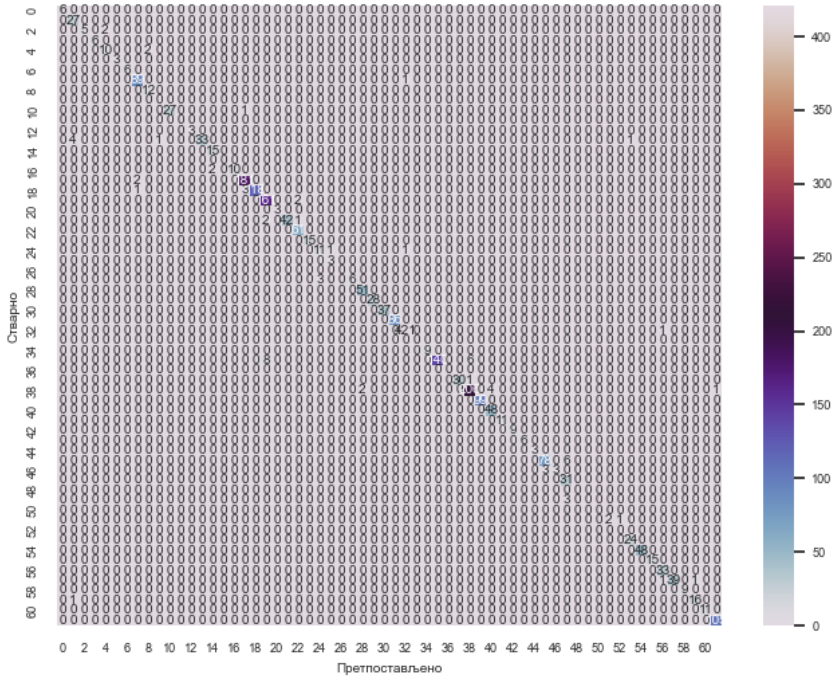
Табела 5: Извештај о класификацији

Класа	Прецизност	Сенз.	F1 учинак
0	0.75	1.00	0.86
1	0.96	0.96	0.96
2	1.00	0.43	0.60
3	0.75	1.00	0.86
	...		
58	0.82	1.00	0.90
59	0.88	0.88	0.88
60	0.73	1.00	0.85
61	0.94	0.97	0.95
$\overline{avg}$	0.95	0.95	0.95

Овај модел мреже остварује задовољавајуће резултате, али опет најгоре када се пореди са осталим моделима. Примећују се исте разлике као и код претходних модела.

3.8 Ансамбл претходно креираних модела

Матрица конфузије је дата сликом 17, док је извештај о класификацији дат 6.



Слика 17: Матрица конфузије добијена применом ансамбла модела

Табела 6: Извештај о класификацији

Класа	Прецизност	Сенз.	F1 учинак
0	1.00	1.00	1.00
1	0.84	1.00	0.92
2	1.00	0.71	0.83
3	1.00	1.00	1.00
	...		
58	1.00	1.00	1.00
59	0.94	0.94	0.94
60	1.00	1.00	1.00
61	0.99	1.00	1.00
$\overline{avg}$	0.98	0.98	0.98

На основу резултата са извештаја, може се приметити да је ансамбл, иако минимално, опет побољшао учинак класификације саобраћајних знакова. У оквиру резултата претходних модела, може се приметити да је модел са најбољим учинком био модел чије су димензије улаза најмање.