UNIVERSITATEA DE VEST DIN TIMIŞOARA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
PROGRAMUL DE STUDII DE LICENŢĂ:
Informatică Aplicată

Recunoasterea semnelor de circulatie

COORDONATOR:
Conf. Dr. Cosmin BONCHIȘ

STUDENTI:
Chera Denis Marian
Constantinescu Marius Alexandru

TIMIŞOARA
2022

Cuprins

1 Descrierea proiectului **3**

2 Prezentare generala **4**

3 Algoritmi **5**
3.1 Convolutional Neural Network (CNN) 5

4 Testare **6**
4.1 Setul de date 6
4.2 Pregatirea datelor 6

5 Rezultate **10**

Capitolul 1

Descrierea proiectului

Scopul esential al acestui proiect este utilizarea tehnicilor de procesare a imaginilor pentru a ajuta in recunoasterea semnelor de circulatie distincte. Acest lucru va contribui la simplificarea vietii oamenilor si la imbunatatirea calitatii acesteia, reducand considerabil dependenta lor de ceilalti si, prin urmare, asigurandu-le siguranta in timpul conducerii. Abordarea actuala are mai multe defecte si preia mai putine caracteristici din fotografii. Metoda noastra sugerata utilizeaza o abordare a retelei neuronale convolutionale (CNN), care imbunatateste acuratetea recunoasterii si permite extragerea de informatii suplimentare.

Detectarea semnelor de circulatie este instruita in aceasta lucrare folosind o abordare de baza a procesarii imaginilor, care reduce timpul de procesare si imbunatateste acuratetea.

Capitolul 2

Prezentare generala

Este bine cunoscut faptul ca numarul de vehicule rutiere a crescut enorm datorita realizarilor tehnologice din industria auto si, foarte precis, disponibilitatii tarifelor mici. Odata cu aceasta crestere remarcabila, numarul accidentelor este de asemenea intr-o crestere accentuata de la an la an, din cauze diferite, in care necunoasterea semnelor de circulatie este considerata drept o cauza majora a acestor accidente.

Dezvoltarea sistemelor autonome de recunoastere a semnelor de circulatie ajuta la asistarea soferului in diferite moduri pentru a garanta siguranta acestuia, ceea ce pastreaza, de asemenea, si siguranta celorlalti soferi si pietoni.

Semnele de circulatie sunt utilizate ca metoda de avertizare si indrumare a soferilor, ajutand la reglarea fluxului de trafic printre vehicule, pietoni, motociclete, biciclete si altii care calatoresc pe strazi, autostrazi si alte drumuri. Semnele de circulatie sunt unul dintre echipamentele rutiere, sub forma de simboluri, litere, numere, propozitii si combinatii precum avertismente, interdictii, ordine sau indicatii pentru utilizatorii drumurilor. In incercarea de a se concentra asupra drumului in timpul conducerii, soferii rateaza adesea indicatoarele aflate pe marginea drumului, lucru care ar putea fi periculos pentru ei si pentru oamenii din jurul lor.

Capitolul 3

Algoritmi

3.1 Convolutional Neural Network (CNN)

Algoritmul utilizat pentru acest proiect este reprezentat de CNN (Convolutional Neural Network). Retelele neuronale au capacitati mai mari de recunoastere a modelelor de imagine si sunt utilizate pe scara larga in algoritmii de computer Vision, iar CNN este o clasa de retele neuronale care se aplica cel mai frecvent la analiza imaginilor vizuale. Clasificarea si detectarea semnelor de circulatie reprezinta sarcini majore in conducerea autonoma, deoarece ofera infomatii despre semnele intalnite si contribuie in luarea deciziilor.

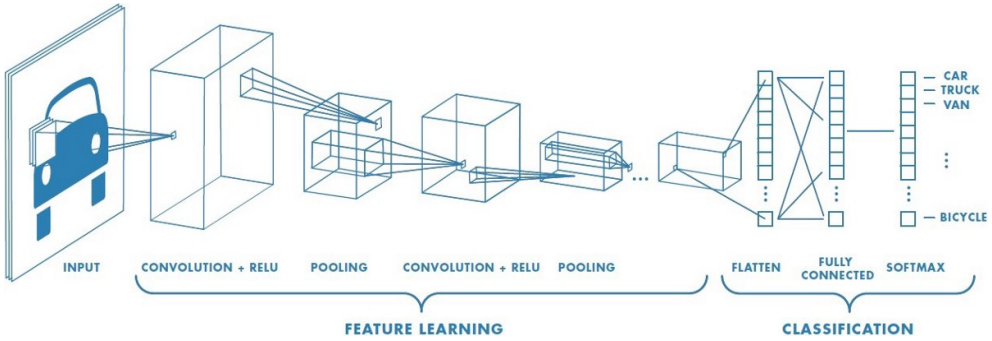

Figura 3.1: Convolutional Neural Network

Imaginea de mai sus arata cum functioneaza un algoritm CNN cu anumite date. Mai intai extrage caracteristicile necesare de date si apoi le clasifica pe baza acelor informatii.

Intrarea unui strat convolutional este o imagine de dimensiunea m $\times$ m $\times$ r, unde m este inaltimea si latimea imaginii, iar r este numarul de canale (ex. imaginile RGB au 3 canale: rosu, verde si albastru).[2]

Asadar, acest domeniu permite masinilor sa vada lumea asa cum o fac oamenii, sa o perceapa intr-un mod similar si chiar sa foloseasca cunostintele pentru o multitudine de sarcini, cum ar fi recunoasterea imaginilor, analiza si clasificarea imaginilor, recrearea media, sistemele de recomandare, procesarea limbajului natural etc. Progresele in viziunea Computer Vision cu invatare profunda au fost construite si perfectionate cu timpul, in primul rand pe un anumit algoritm - o retea neuronala convolutionala .[1]

Capitolul 4

Testare

4.1 Setul de date

Setul de date pentru acest proiect este descarcat de pe site-ul Kaggle. Acesta contine aproximativ 40000 de imagini cu diferite semne de circulatie si este clasificat in 43 de clase diferite. Setul de date este destul de variat, unele dintre clase au mai multe imagini, in timp ce unele au mai putine imagini.

Figura de mai jos cuprinde cele 43 de clase sau imaginile care sunt prezente in setul nostru de date.

Figura 4.1: Imagini din setul de date

4.2 Pregatirea datelor

Pentru a face ca modelul sa fie usor de inteles si sa prezica un rezultat precis, mai intai am efectuat cateva operatii asupra setului nostru de date .

Astfel, am efectuat urmatorii pasi:

- Am importat setul de date impreuna cu bibliotecile necesare, iar apoi am creat un dictionar unde am salvat denumirile tuturor semnelor de circulatie aflate in setul nostru de date (figura 5.2). In continuare, am extras numarul de imagini pentru fiecare clasa in parte pentru a sti care este volumul de date cu care lucram pentru fiecare categorie (figura 5.3).

  Dupa colectarea datelor, am creat un array cu toate imaginile pe care le-am redimensionat(30x30, 'rgb') si i-am atribuit fiecareia eticheta corespunzatoare (Denumirea semnului de circulatie ilustrat de aceasta) (vezi Figura 5.4);

```
# Label Overview
classes = { 0:'Limita de viteza (20km/h)',
            1:'Limita de viteza (30km/h)',
            2:'Limita de viteza (50km/h)',
            3:'Limita de viteza (60km/h)',
            4:'Limita de viteza (70km/h)',
            5:'Limita de viteza (80km/h)',
            6:'Sfaritul zonei cu limita 80km/h',
            7:'Limita de viteza (100km/h)',
            8:'Limita de viteza (120km/h)',
            9:'Trecerea interzisa',
            10:' Accesul interzis vehiculelor cu peste 3.5 tone',
            11:'Prioritate',
            12:'Drum cu prioritate',
            13:'Cedeaza trecerea',
            14:'Stop',
            15:'Accesul interzis vehiculelor',
            16:'Depasirea interzisa a autovehiculelor cu peste 3.5 tone',
            17:'Intrarea interzisa',
            18:'Atentie sporita',
            19:'Curba periculoasa la stanga',
            20:'Curba periculoasa la dreapta',
            21:'Curba dubla',
            22:'Drum cu denivelari',
            23:'Drum alunecos',
            24:'Drumul se ingusteaza pe dreapta',
            25:'Drum in lucru',
            26:'Semafor',
            27:'Pietoni',
            28:'Copii care traversează',
            29:'Traversarea bicicletelor',
            30:'Atenție la gheață/zăpadă',
            31:'Atentie animale',
            32:'Sfarsit restrictii',
            33:'Virare la dreapta',
            34:'Virare la stanga',
            35:'Doar înainte',
            36:'In fata sau la dreapa',
            37:'In fata sau la stanga',
            38:'La dreapta',
            39:'La stanga',
            40:'Sens giratoriu',
            41:'Sfarsitul zonei unde depasirea e interzisa',
            42:'Sfarsitul interzicerii trecerii vehiculelor de peste 3.5 tone' }
```

Figura 4.2: Dictionarul cu semnele de circulatie

```
folders = os.listdir(train_path)

train_number = []
class_num = []

for folder in folders:
    train_files = os.listdir(train_path + '/' + folder)
    train_number.append(len(train_files))
    class_num.append(classes[int(folder)])

# Sorting the dataset on the basis of number of images in each class
zipped_lists = zip(train_number, class_num)
sorted_pairs = sorted(zipped_lists)
tuples = zip(*sorted_pairs)
train_number, class_num = [list(tuple) for tuple in tuples]

# Plotting the number of images in each class
plt.figure(figsize=(21, 10))
plt.bar(class_num, train_number)
plt.xticks(class_num, rotation='vertical')
plt.show()
```

Figura 4.3: Extragerea numarului de imagini

```
image_data = []
image_labels = []

for i in range(NUM_CATEGORIES):
    path = data_dir + '//Train//' + str(i)
    images = os.listdir(path)

    for img in images:
        try:
            image = cv2.imread(path + '//' + img)
            image_fromarray = Image.fromarray(image, 'RGB')
            resize_image = image_fromarray.resize((IMG_HEIGHT, IMG_WIDTH))
            image_data.append(np.array(resize_image))
            image_labels.append(i)
        except:
            print("Error in " + img)

# Changing the list to numpy array
image_data = np.array(image_data)
image_labels = np.array(image_labels)
```

Figura 4.4: Pregatirea datelor

- am amestecat datele care urmeaza sa fie antrenate si le-am impartit in 2 categorii: de antrenare si de validare (vezi Figura 5.7);

```
shuffle_indexes = np.arange(image_data.shape[0])
np.random.shuffle(shuffle_indexes)
image_data = image_data[shuffle_indexes]
image_labels = image_labels[shuffle_indexes]
```

```
X_train, X_val, y_train, y_val = train_test_split(image_data, image_labels, test_size=0.3, random_state=42, shuffle=True)

X_train = X_train/255
X_val = X_val/255

print("X_train.shape", X_train.shape)
print("X_valid.shape", X_val.shape)
print("y_train.shape", y_train.shape)
print("y_valid.shape", y_val.shape)
```

```
X_train.shape (27446, 30, 30, 3)
X_valid.shape (11763, 30, 30, 3)
y_train.shape (27446,)
y_valid.shape (11763,)
```

Figura 4.5: Amestecarea datelor si sortarea acestora

- am creat un model utilizat pentru antrenarea algoritmului de machine learning CNN, astfel incat sa obtinem un rezultat cat mai precis (o acuratete de 98 la suta) (vezi Figura 5.7).

  Ulterior, am folosit modelul respectiv pentru realizarea predictiilor (figura 5.8).

```
model = keras.models.Sequential([
    keras.layers.Conv2D(filters=16, kernel_size=(3,3), activation='relu', input_shape=(IMG_HEIGHT,IMG_WIDTH,channels)),
    keras.layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu'),
    keras.layers.MaxPool2D(pool_size=(2, 2)),
    keras.layers.BatchNormalization(axis=-1),

    keras.layers.Conv2D(filters=64, kernel_size=(3,3), activation='relu'),
    keras.layers.Conv2D(filters=128, kernel_size=(3,3), activation='relu'),
    keras.layers.MaxPool2D(pool_size=(2, 2)),
    keras.layers.BatchNormalization(axis=-1),

    keras.layers.Flatten(),
    keras.layers.Dense(512, activation='relu'),
    keras.layers.BatchNormalization(),
    keras.layers.Dropout(rate=0.5),

    keras.layers.Dense(43, activation='softmax')
])
```

Figura 4.6: Structurarea modelului

```
aug = ImageDataGenerator(
    rotation_range=10,
    zoom_range=0.15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.15,
    horizontal_flip=False,
    vertical_flip=False,
    fill_mode="nearest")

history = model.fit(aug.flow(X_train, y_train, batch_size=32), epochs=epochs, validation_data=(X_val, y_val))
```

Figura 4.7: Antrenarea setului de date

```
plt.figure(figsize = (25, 25))

start_index = 0
for i in range(10):
    plt.subplot(6, 2, i + 1)
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
    prediction = pred[start_index + i]
    actual = labels[start_index + i]
    col = 'g'
    if prediction != actual:
        col = 'r'
    plt.xlabel('Actual={} || Pred={}'.format(classes[actual], classes[prediction]), color = col)
    plt.imshow(X_test[start_index + i])
plt.show()
```

Figura 4.8: Predictii

Capitolul 5

Rezultate

```
Epoch 19/30
858/858 [==============================] - 46s 54ms/step - loss: 0.0205 - accuracy: 0.9938 - val_loss: 0.0040 - val_accuracy:
0.9990
Epoch 20/30
858/858 [==============================] - 46s 54ms/step - loss: 0.0158 - accuracy: 0.9953 - val_loss: 0.0034 - val_accuracy:
0.9990
Epoch 21/30
858/858 [==============================] - 45s 52ms/step - loss: 0.0165 - accuracy: 0.9949 - val_loss: 0.0033 - val_accuracy:
0.9990
Epoch 22/30
858/858 [==============================] - 45s 52ms/step - loss: 0.0136 - accuracy: 0.9957 - val_loss: 0.0061 - val_accuracy:
0.9980
Epoch 23/30
858/858 [==============================] - 46s 53ms/step - loss: 0.0122 - accuracy: 0.9961 - val_loss: 0.0049 - val_accuracy:
0.9988
Epoch 24/30
858/858 [==============================] - 44s 51ms/step - loss: 0.0107 - accuracy: 0.9965 - val_loss: 0.0051 - val_accuracy:
0.9985
Epoch 25/30
858/858 [==============================] - 45s 52ms/step - loss: 0.0110 - accuracy: 0.9970 - val_loss: 0.0032 - val_accuracy:
0.9992
Epoch 26/30
858/858 [==============================] - 45s 52ms/step - loss: 0.0125 - accuracy: 0.9962 - val_loss: 0.0030 - val_accuracy:
0.9992
Epoch 27/30
858/858 [==============================] - 45s 52ms/step - loss: 0.0125 - accuracy: 0.9959 - val_loss: 0.0053 - val_accuracy:
0.9986
Epoch 28/30
858/858 [==============================] - 46s 53ms/step - loss: 0.0110 - accuracy: 0.9962 - val_loss: 0.0028 - val_accuracy:
0.9993
Epoch 29/30
858/858 [==============================] - 45s 53ms/step - loss: 0.0097 - accuracy: 0.9968 - val_loss: 0.0052 - val_accuracy:
0.9983
Epoch 30/30
858/858 [==============================] - 43s 50ms/step - loss: 0.0088 - accuracy: 0.9972 - val_loss: 0.0035 - val_accuracy:
0.9987
```

Figura 5.1: Rezultatele din urma antrenarii modelului

Pentru a verifica eficienta programului creat, am facut o serie de teste folosindu-ne de setul de date utilizat.

Mai intai am testat pe un volum mai mic de imagini, mai exact 10, iar rezultatele sunt prezentate in figura de mai jos.

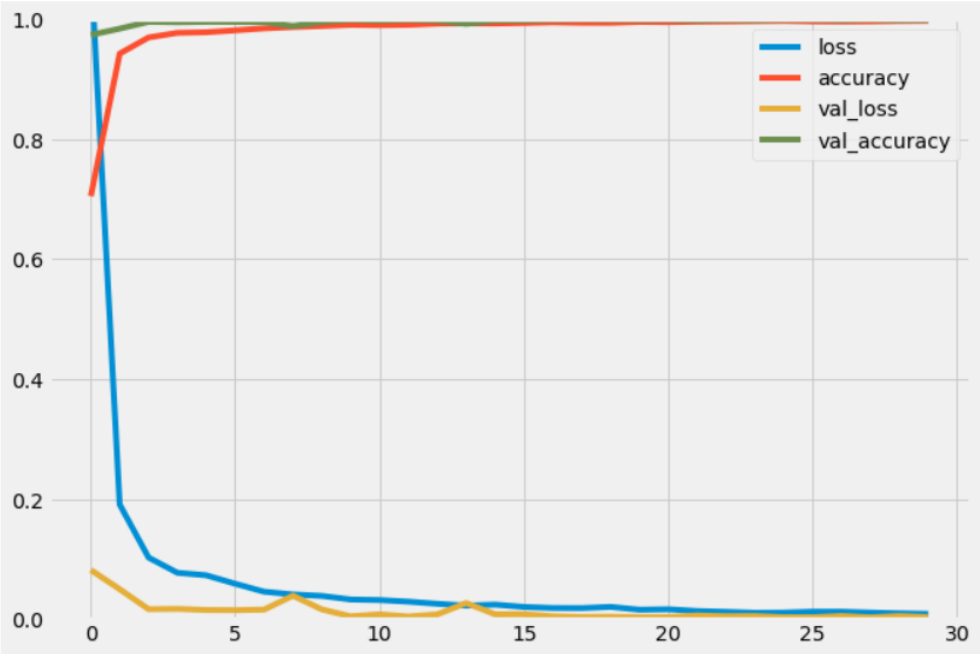

Figura 5.2: Graficul de precizie al modelului antrenat in raport cu nr. de epoci dat (30)

Figura 5.3: Rezultate pentru un set de 10 imagini

Figura 5.4: Rezultate pentru un set de 25 imagini

Figura 5.5: Rezultate pentru un set de 50 imagini

Figura 5.6: Rezultate pentru un set de 100 imagini

Figura 5.7: Rezultate pentru un set de 100 imagini

Putem observa cu usurinta privind figurile, ca exista o precizie aproape perfecta a algoritmului de detectare a semnelor de circulatie, existand o singura predictie gresita aparuta dupa multe teste cu rezultate impecabile.

Bibliografie

[1] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

[2] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.