Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek

Računarstvo usluga i analiza podataka

SEMINARSKI RAD

„Klasifikacija slika korištenjem Azure platforme i preddefiniranog DesneNet modela“

Ivor Plander

Osijek, 2025.

Sadržaj

1. Uvod 1
2. Opis problema 1
2.1. Korišteni podaci 1
2.2. Korišteni postupci strojnog učenja 1
3. Opis programskog rješenja 1
3.1. Model strojnog učenja 1
3.2. Način korištenja API-ja 1
3.3. Klijentska aplikacija 1
3.4. Dodatno 1
4. Zaključak 1
5. Poveznice i literatura 1

1. Uvod

Cilj projektnog zadatka je korištenjem Azure platforme (Azure ML Studio-a) realizirati model neuronske mreže za prepoznavanje tj. klasifikaciju slika. Budući da je autor ovog projektnog zadatka student na smjeru "DRA" - Računalno inženjerstvo, skup podataka za treniranje neuronske mreže je skup ručno crtanih simbola jednostavnih shematskih elektroničkih simbola kao što su simbol otpora, induktiviteta, kapaciteta, diode i slično. Skup podataka preuzet je sa "Kaggle" (www.kaggle.com) platforme.

Razvijeno rješenje u trenutnom opsegu nastalo je iz potrebe za rješavanjem projektnog zadataka iz kolegija "Računarstvo usluga i analiza podataka" i kao takvo za autora ima najveću korist u svrhu učenja, stjecanja novih vještina i potvrđivanja teoretskih koncepata prezentiranih na predavanjima i laboratorijskim vježbama. Dakle trenutno rješenje pogodno je samo za prepoznavanje osnovnih simbola čije klase/kategorije će biti navedene i opisane u daljnjim paragrafima, međutim potencijal i motivacija ostaju isti, odnosno sustav je proširiv i nadogradiv.

2. Opis problema

Kao i u svakoj branši koja se bavi rješavanjem problema tehničke prirode pa tako i u elektronici, ljudsko znanje je ograničeno, a vrlo često i usko specijalizirano za specifične potrebe industrije i znanosti. Obzirom na rečeno u kontekstu elektronike, dizajna i analize elektroničkih krugova inženjeri, tehničari i hobisti se prilikom rješavanja problema susreću s nepoznatim elektroničkim simbolima što može biti rezultat njihove uske specijalizacije, neznanja ili čak nemogućnosti pronalaska odgovarajuće dokumentacije pojedinih komponenata.

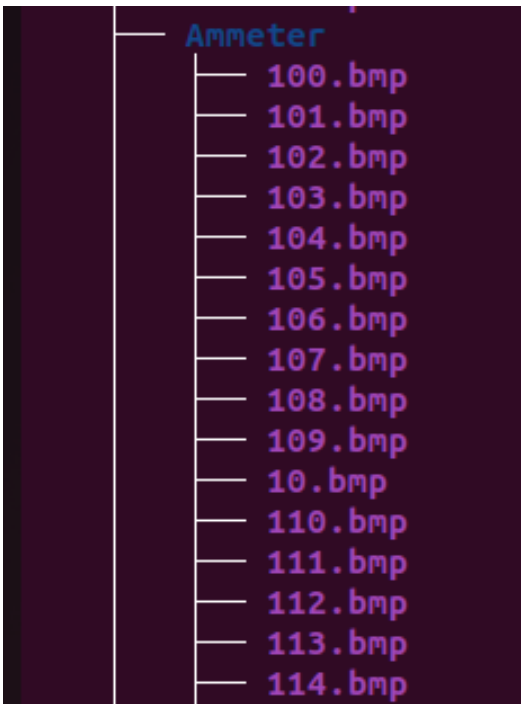
Slični pristupi za rješavanje ovakvog konkretnog problema ne postoje obzirom da su elektronika i sve njezine podgrane ustanovljene početkom 20. stoljeća kada su se ljudi oslanjali na znanja stručnjaka ,a informacije uglavnom postojale i bile zapisane u analognim medijima.

Slični digitalni sustavi koji olakšavaju posao inženjerima postoje u vidu simulatora i CAD alata. Npr. LTSpice, Altium Designer.

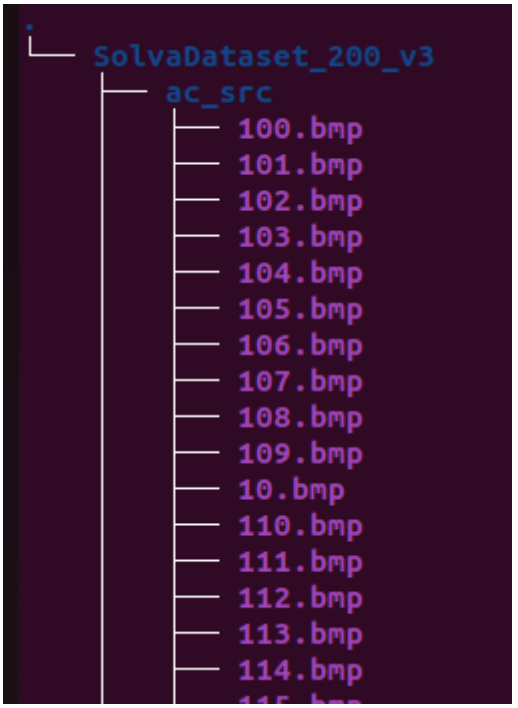
Problemi koji se rješavaju sličnim pristupima postoje u velikom broju, naime prepoznavanje objekata sa slika vrlo je koristan alat u medicini npr. za prepoznavanje srčanih bolesti i automobilskom računarstvu npr. za prepoznavanje pješaka i vozila.

2.1.Korišteni podaci

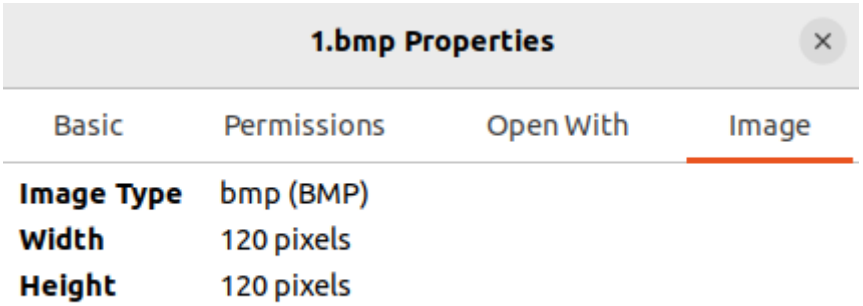
Korišteni podatci su kao što je već navedeno u predhodnim paragrafima slike ručno crtanih osnovnih elektroničkih simbola. Elektronički simboli nadalje će biti zvani klase ili kategorije, a skup podataka će se u daljnjem tekstu zvati dataset. Dakle dataset korišten u projektu sastoji se od 14 direktorija gdje se u svakom direktoriju nalazi otprilike 200 slika. Svaki direktorij nosi naziv klase slika koje se u njemu nalaze. Slika 1. i slika 2. prikazuju strukturu dataseta. Sve slike su 120 x 120 pixela, kao što je vidljivo na slici 3. Format zapisa slike je .bmp.



Slika 1.



Slika 2.



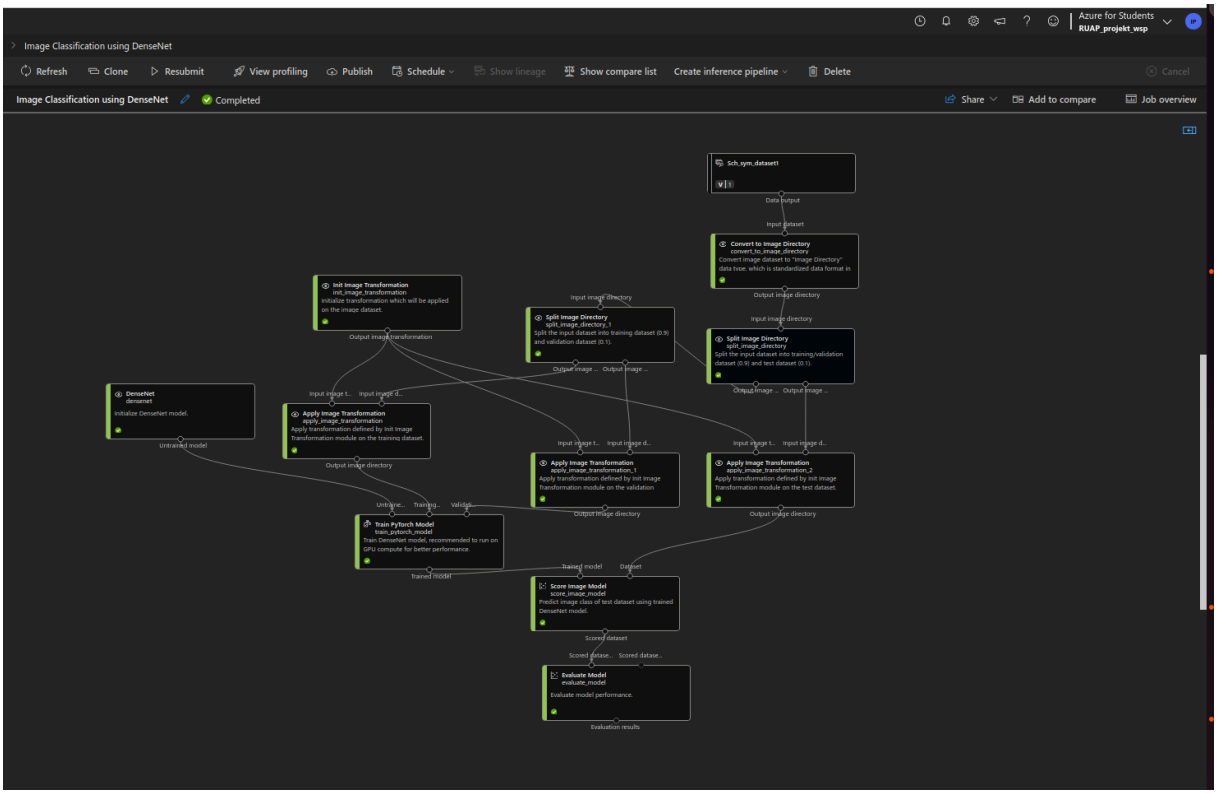
Slika 3.

2.2.Korišteni postupci strojnog učenja

Pustupak strojnog učenja korištenog u ovom projektu je treniranje neuronske mreže, specifično preddefiniranog DensNet modela unutar Azure ML designer-a.

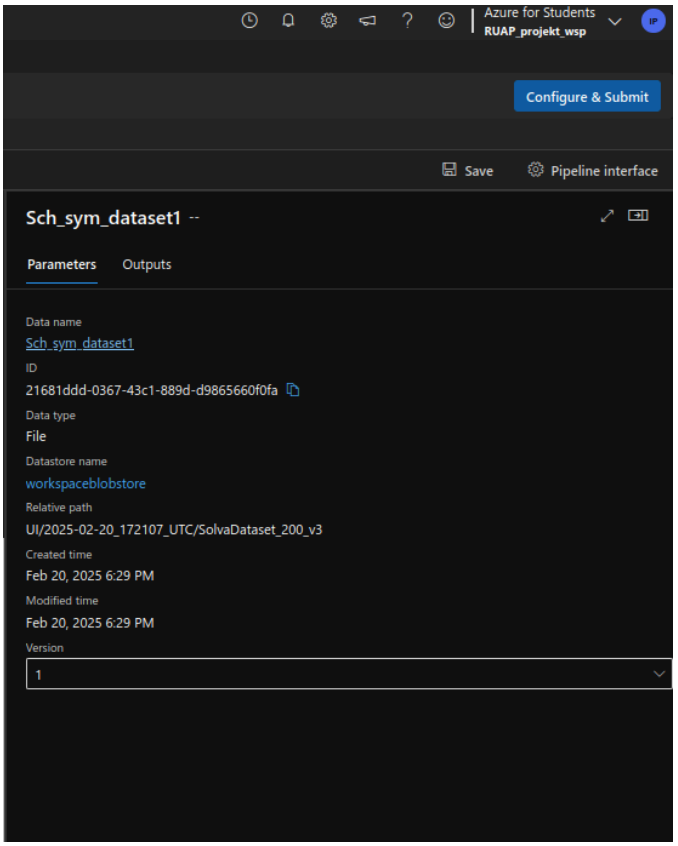
3. Opis programskog rješenja

Opis cjelokupnog programskog rješenja zajedno s načinom korištenja, screenshotovima i specifičnim/ključnim segmentima koda.



Slika 4.

Na slici 5. prikazan je dataset unutar Azure ML Studio-a.



Slika 5

Na slikama 6. i 7. nalazi se kod koji je pokrenut lokalno na računalu kako bi se ostvarila veza s enpointom i dobio response na request te dobiveni response ispisao čitljvije negu u raw json formatu.

```
        // Convert image to Base64
        byte[] imageBytes = File.ReadAllBytes(imgPath);
        string base64Image = Convert.ToBase64String(imageBytes);

        // Construct the JSON request body
        var requestBody = $@"
        {{
            ""Inputs"": {{
            ""WebServiceInput0"": [
            {{
            ""image"": ""{base64Image}"",
            ""id"": 6,""category"": ""cap""
            }}]}},
            ""GlobalParameters"": {{}}
        }}";

        // Replace with your API key
        const string apiKey = "hide API key for screenshot :)";
        if (string.IsNullOrEmpty(apiKey))
        {
            throw new Exception("A key should be provided to invoke the endpoint");
        }
        client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", apiKey);
        client.BaseAddress = new Uri("http://9677ccee-1b55-4b6c-ac9c-8325f1db841c.westeurope.azurecontainer.io/score");

        var content = new StringContent(requestBody, Encoding.UTF8, "application/json");

        HttpResponseMessage response = await client.PostAsync("", content);

        if (response.IsSuccessStatusCode)
        {
            string result = await response.Content.ReadAsStringAsync();

            Console.WriteLine("\nRaw JSON Response:");
            Console.WriteLine(result);  // Print the full response for debugging

            PrintProbabilities(result);
        }
```

Slika 6.

Metoda PrintProbabilities() uzima sirovi json string i ispisuje ga u terminal liniju po liniju radi lakše čitljivosti. Rezultat/response je vidljiv na slici 8.

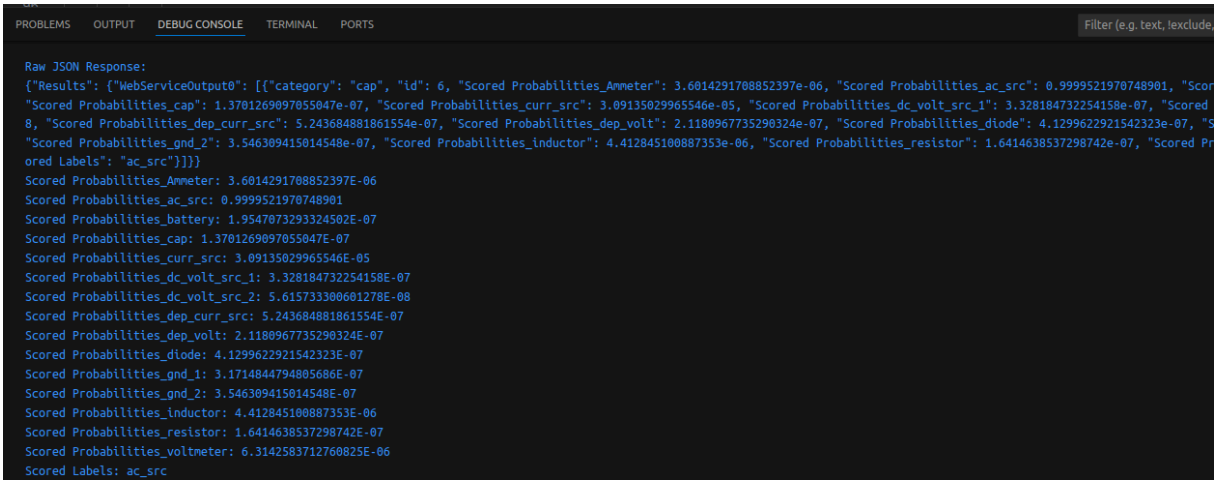
```
static void PrintProbabilities(string jsonResponse)
{
    try
    {
        JObject json = JObject.Parse(jsonResponse);

        // Navigate to "Results" -> "WebServiceOutput0"
        var predictions = json["Results"]?["WebServiceOutput0"] as JArray;
        if (predictions == null || predictions.Count == 0)
        {
            Console.WriteLine("Error: No prediction results found.");
            return;
        }

        JObject prediction = (JObject)predictions[0];

        // Print all scored probabilities
        foreach (var property in prediction.Properties())
        {
            string propName = property.Name;
            if (propName.StartsWith("Scored Probabilities_") || propName == "Scored Labels")
            {
                Console.WriteLine($"{propName}: {property.Value}");
            }
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error parsing response: " + ex.Message);
    }
}
```

Slika 7.



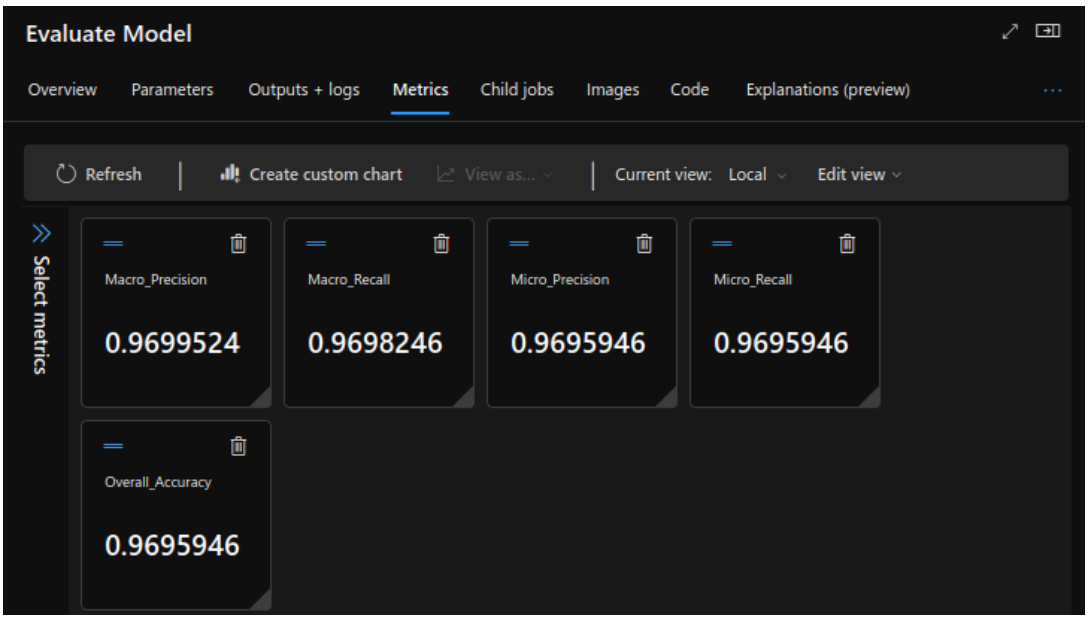
Slika 8.

3.1. Model strojnog učenja

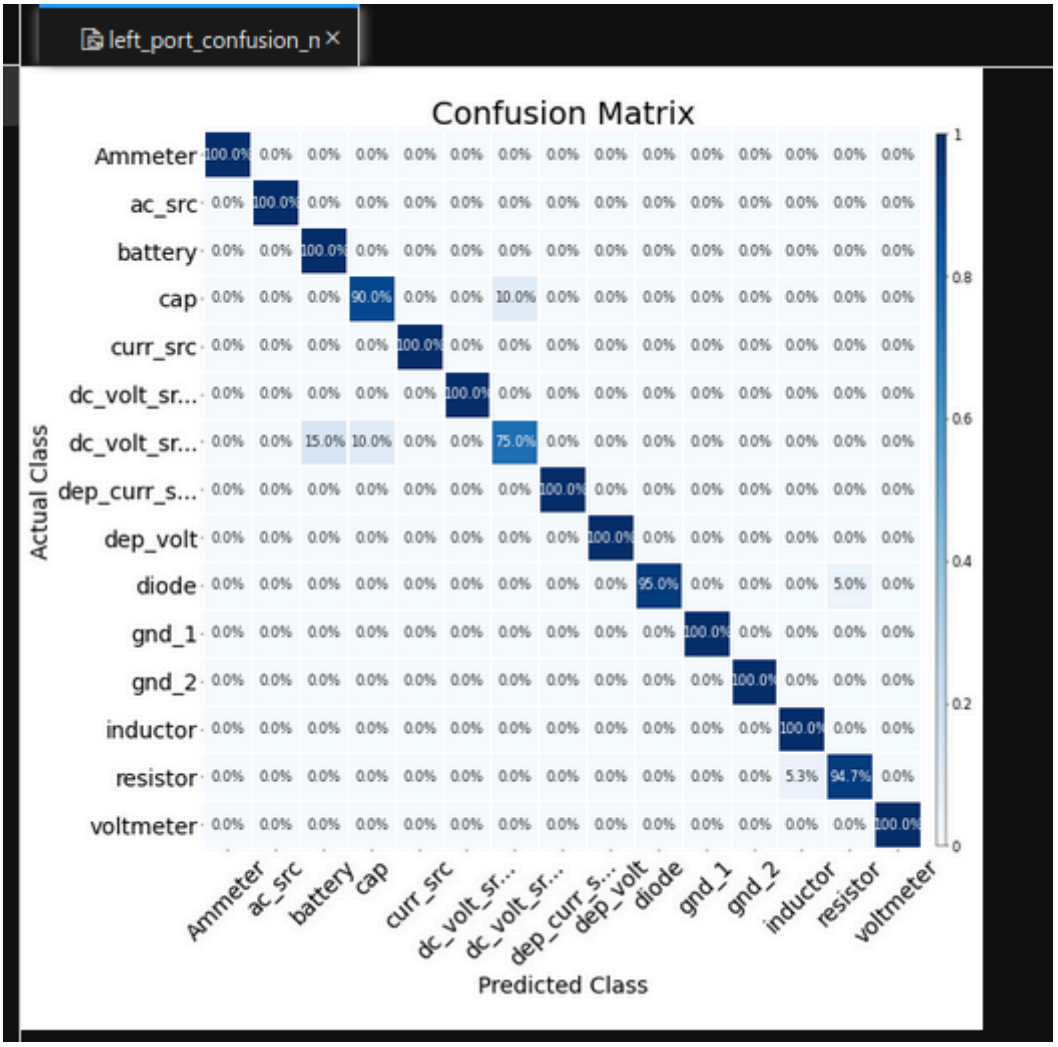
Prikaz modela koji je korišten, argumentacije zašto je odabran baš taj, provedene usporedbe između više njih kako bi se odabrao najbolji.

Obzirom da je DenseNet već optimizirana arhietktura neuronske mreže za prepoznavanje slika i optimizirana za izvođenje na Azure-ovim compute instancama kao takva je odabrana kao rješenje. Za prepoznavanje slika mogla se koristiti i NAS (neural architecture search) metodologija međutim obzirom na ograničene resurse koje Azure nudi za studentske licence to je poprilično skup pristup obzirom na vrijeme i broj ljudi u timu.

Kako nebi riječ autora bila jedini argument za kvalitetu odabranog modela, na slikama 9. i 10. prikazuju se parametri eveluacije navedenog modela.



Slika 9.



Slika 10.

Iz matrice konfuzije vidljivo je da model poprilično dobro radi za ovaj dataset. Zanimljivo je da “inductor” i “resistor” te “dc_volt_source” i “battery” imaju znatne postotke konfuzije. Razlog tomu je što su i na oko simboli slični jedni drugima.

3.2.Način korištenja API-ja

API koji se koristi u projektu utilizira REST endpoint i autentifikaciju putem API ključa. Na slici 11. i 12. prikazan je kod (C#) korišten za komunikaciju s API-jem. Bitno je i naglasiti shemu prema kojoj se šalju podatci na API. Primjer request body prikazan je na slici 11. ,a json shema je prikazana na slici 12.

```
// Construct the JSON request body
var requestBody = $@"
{{  ""Inputs"":
    {{
        ""WebServiceInput0"":
        [
        {{
        ""image"": ""{base64Image}"",
        ""id"": 6,
        ""category"": ""cap""
        }}]
        }},
        ""GlobalParameters"": {{}}
}}";
```

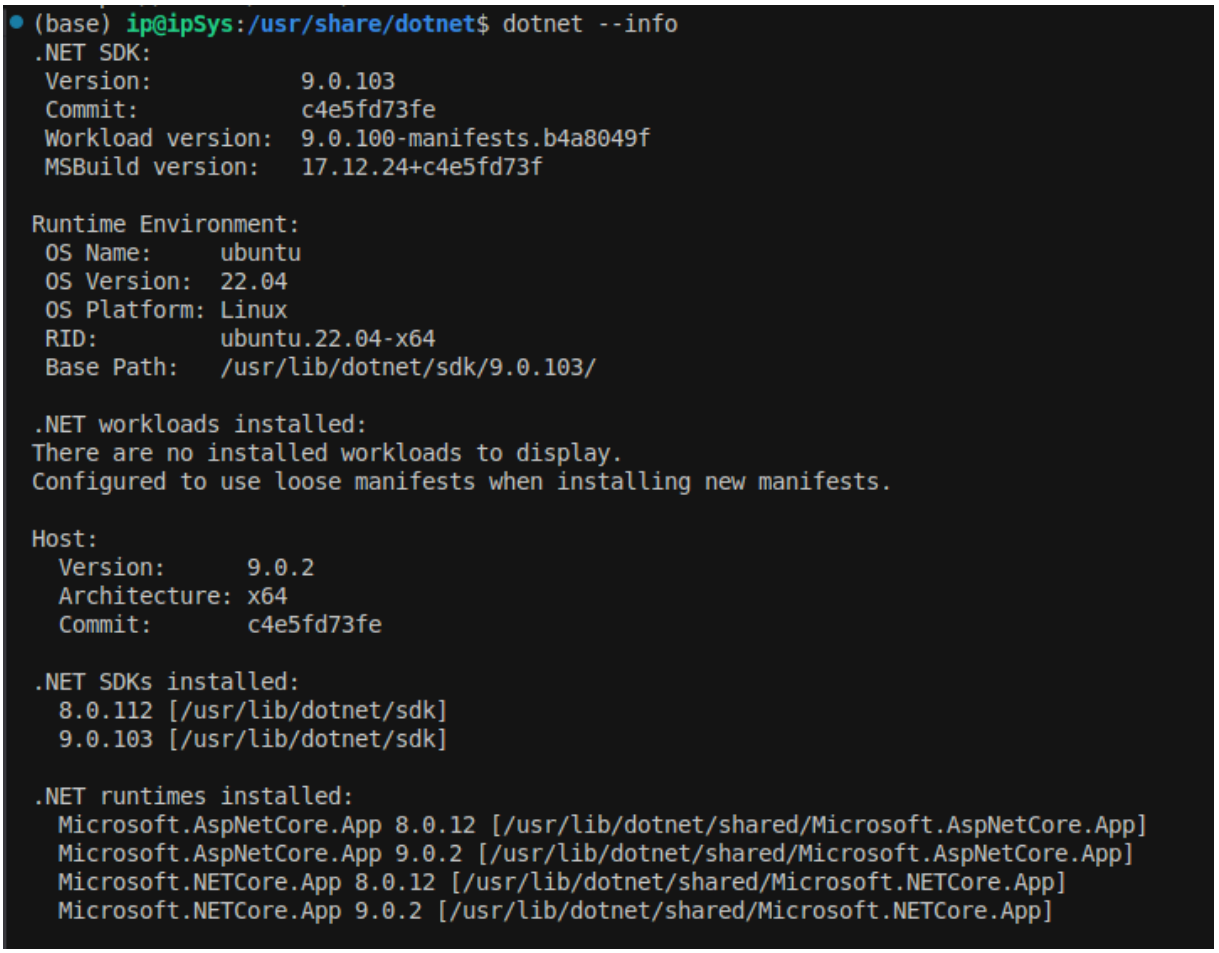
Slika 11.


```
{
  "columnAttributes": [
    {
      "name": "image",
      "type": "Bytes",
      "isFeature": true,
      "elementType": {
        "typeName": "bytes",
        "isNullable": false
      },
      "properties": {
        "mime_type": "image/png",
        "image_ref": "image_info"
      }
    },
    {
      "name": "id",
      "type": "Numeric",
      "isFeature": false,
      "elementType": {
        "typeName": "int64",
        "isNullable": false
      }
    },
    {
      "name": "category",
      "type": "String",
      "isFeature": false,
      "elementType": {
        "typeName": "str",
        "isNullable": false
      },
      "properties": {
        "annotation_type": "COCO_classification",
        "categories": [
          {
            "id": 0,
            "name": "Ammeter"
          },
          {
            "id": 1,
            "name": "ac_src"
          },
          {
            "id": 2,
            "name": "battery"
          },
          {
            "id": 3,
            "name": "cap"
          },
          {
```


Slika 12.

3.3.Klijentska aplikacija

Klijenstska aplikacija je C# aplikacija napisana za desktop okruženja. Klijensta aplikacija u projektu izvodi se u Linux okruženju s .NET 9.0 sdk. Slika 13. prikazuje detalje.



Slika 13.

3.4.Dodatno

Osim onih navedinih u predhodnim paragrafima nisu provedeni nikakvi dodatni napori.

4. Zaključak

Prepoznavanje objekata (klasifikacija) na slikama ključan je koncept u raznovrsnim djelatnostima, stoga poznavanje osnovnih koncepata strojnog učenja daje uvid u kompleksnost grane znanosti i inženjerstva koje se time bave ali i dobru podlogu za daljnji razvoj vještina i sposobnosti za rješavanje raznovrsnih problema ovim pristupom. Model strojnog učenja realiziran u ovom projektu dakako može biti poboljšan, a funkcionalnosti proširene. Za primjer prvi korak dalje bio bi proširiti dataset na više klasa kako bi sustav bio u mogućnosti prepoznati i složenije simbole.

Zatim klijentska aplikacija također može biti implementirana za Android ili IOs platforme kako bi se omogućila i mobilnost rješenja u smislu korištenja kamere na uređajima.

5. Poveznice i literatura

Programskom je rješenju moguće pristupiti preko:

Programsko rješenje na GitHubu
…