ივანე ჯავახიშვილის სახელობის
თბილისის სახელმწიფო უნივერსიტეტი
ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტი
კომპიუტერული მეცნიერების დეპარტამენტი

დავით იაკობიძე

ღრმა სწავლება სახის ამოცნობისთვის

სამაგისტრო პროგრამა: კომპიუტერული მეცნიერება

სამაგისტრო ნაშრომი შესრულებულია კომპიუტერული
მეცნიერების მაგისტრის აკადემიური ხარისხის მოსაპოვებლად

ხელმძღვანელი: კომპიუტერული მეცნიერების
ასისტენტ-პროფესორი გელა ბესიაშვილი

თბილისი
2022

სარჩევი

ანოტაცია

ნაშრომში განხილული იქნება კვლევა ღრმა სწავლებით სახის ამოცნობის შესახებ. აღწერილია ამოცანის გადაწყვეტა, ალგორითმებისა და ტექნლოგიების განხილვა, სახის ამომცნობი სისტემა, მოწყობილობის შექმნა და სისტემის რეალიზაცია, როგორც მოწყობილობაზე, ასევე ამომცნობ სერვერზე.

ასევე განხილულია სახის ამოცნობის განვითარების ისტორია, გამოყენების მაგალითები და მისი მნიშვნელობა ადამიანთა ცხოვრებაში. ნაჩვენებია კონკრეტული მაგალითები თუ რა ეტაპებს გადის ადამიანის სახის გამოსახულება მისი გადაღებიდან საბოლოო, პასუხის მიღების, ეტაპამდე.

აღწერილია სისტემაში გამოყენებული ნეირონული ქსელი და მის უპირატესობებს სხვა ნეირონულ ქსელებთან შედარებით. წარმოვადგენთ ღრმა სწავლების მიერ მიღებულ შედეგებს, აღვწერთ სახის ამოცნობის დატრენინგებას, ამომცნობი მოდელის შექმნასა და მის როლს უკეთესი შედეგის მიღებაში.

შევაჯამებთ შედეგებს, განვიხილავთ პროექტის გამოყენების მაგალითებს. სისტემისა და მოწყობილობის სამომავლო ფუნქციებსა და განვითარების გზებს.

Annotation

This paper describes research of deep learning with face recognition. There are described solutions of the problem, review of the algorithms and technologies, face recognition system, creation of the device and realization of the system for the device and the recognition server.

The history of the development of face recognition, examples of use and its importance in human life are also discussed. Specific examples are shown of what stages the image of a human face goes through from its capture to the final stage of receiving a response.

The neural network used in the system and its advantages compared to other neural networks are described. We present the results obtained by deep learning, describe the training of face recognition, the creation of the recognition model and its role in obtaining better results.

We will summarize the results, and consider examples of how to use the system. Also future functions and possibility development of the system and device.

1. შესავალი

ტექნოლოგიების განვითარებასთან ერთად საჭირო გახდა ხელოვნური ინტელექტის შექმნა, რომელიც ადამიანის ცხოვრებას ამატივებს. ხელოვნური ინტელექტის ერთ-ერთ მნიშვნელოვან მიმართულებას წარმოადგენს Computer vision (კომპიუტერული ხედვა) [1], ის წარმოადგენს სისტემას, რომელიც იღებს: სურათს, ვიდეოს ან სხვა ვიზუალურ გამოსახულებას და გვაძლევს რაიმე შედეგს ან მითითებას მიღებული ინფორმაციიდან გამომდინარე. თუ ხელოვნური ინტელექტი არის სისტემა, რომელსაც შეუძლია ფიქრი, მაშინ computer vision ამ სისტემას აძლევს საშუალებას დაინახოს და ამასთანავე იფიქროს როგორც ადამიანმა.

Computer vision თანამედროვე ცხოვრების განუყოფელი ნაწილია, ის ფართოდ გამოიყენება: მედიცინაში, უსაფრთხოებაში, თვითმართვად ავტომობილებში, სოფლის მეურნეობაში, მრეწველობაში და ა.შ.

სახის ამოცნობა ერთ-ერთი მაგალითია computer vision-ის გამოყენების, სახის ამოცნობა არის ადამიანის იდენტიფიკაციის ერთ-ერთი გზა, ესაა ტექნოლოგია, რომელიც მოწყობილობებს აძლევს საშუალებას ამოიცნოს ადამიანი გრაფიკულ ინფორმაციაზე დაყრდნობით წინასწარ გამზადებული სახის მონაცემთა ბაზიდან.

სახის ამოცნობის განვითარება უსაფრთხოების დონის ამაღლებითაა განპირობებული, სოწორედ აქ ვხვდებით მისი გამოყენების მაგალითებს, როგორებიცაა: კრიმინალური შემთხვევების გამოძიება, აეოპორტების უსაფრთხოება, მობილური ტელეფონების ვალიდაცია, საბანკო სფერო და ა.შ. ასევე მისი საშუალებით შესაძლებელია სხვადასხვა დაწესებულებებში დასწრების აღრიცხვა, მარკეტინგის მიზნით ინფორმაციის შეგროვება, ადამიანების ემოციების ამოცნობა და ა.შ.

თემის აქტუალურობიდან გამომდინარე ნაშრომში განხილული იქნება სახის ამოცნობის ტექნოლოგია Deep Learning-ის (ღრმა სწავლება) გამოყენებით, მისი მუშაობის პრინციპი, უშუალოდ გამოყენება და იმპლემენტაცია მოწყობილობაზე.

1.1 ამოცანის დასმა

ჩამოვაყალიბოთ მოცემული ამოცანა სამე ეტაპად: პირველ ეტაპზე საჭიროა განვსაზღვოთ გრაფიკული გამოსახულება, შემდეგ გამოსახულებიდან ვიპონოთ ადგილი სადაც ჩანს ადამიანის სახე. სახის გამოსახულებიდან ამოვიღოთ ინფორმაცია და წინასწარ შენახული პიროვნებების ბაზაში ამოვიცნოთ კონკრეტული პიროვნება.

მეორე ეტაპს წარმოადგენს მოწყობილობის შექმნა, რომელსაც შეეძლება გრაფიკული ინფორმაციის: სურათის ან ვიდეოს გადაღება, მისი დამუშავება და გაგზავნა ამომცნობი სერვისის სერვერზე საიდანაც ჩვენ მივიღებთ საჭირო ინფორმაციას პიროვნების შესახებ.

მესამე, შუამავალ ეტაპს წარმოადგენს დატრენინგება, ესაა ეტაპი როდესაც ხელოვნური ინტელექტი სწავლობს ადამიანის სახის ნაკვთებს და მის ამოცნობას, შესაბამისად არ გვიწევს მონაცემთა ბაზაში ყველა ჩანაწერის შემოწმება, რაც ბევრად ასწრაფებს ამოცნობის პროცესს.

იმის გამო, რომ სახის ამოცნობა გულისმობს ადამიანის სახის ნაკვთების გაზმოვას სისტემა მიეკუთვნება ბიომეტრიას. სხვა ბიომეტრიული სისტემებისაგან განსხვავებით, როგორიცაა თითის ანაბეჭდის ან თვალის ირისის ამოცნობა, სიზუსტე უფრო დაბალია, თუმცა სახის ამოცნობის უპირატესობა მისი უკონტაქტოდ გამოყენებაა.

ყველაზე ხშიად სახის ამოცნობას მობილურ ტელეფონებში ვიყენებთ, ის გვიმარტივებს მოწყობილობის გამოყენებას და უსაფრთხოს ხდის ჩვენს პესონალურ მონაცემებს, მისი საშუალებით შესაძლებელია მობილურში ვალიდაციის გავლა, საბანკო ოპერაციების შესრულება ერთი თითის დაჭერით, ფოტო ალბომებიდან მეგობრების ამოცნობა და ა.შ.

ადამიანის გონებას სახის ამოცნობა ზედმეტი ძალისხმევის საშუალებით შეუძლია, თუმცა სისტემის შექმნა, რომელიც ადამიანივით ამოიცნობს სახეს რთლი ამოცანაა. ადამიანის სახე წარმოადგენს სამ განზომილებიან კვლევის ობიექტს, რომლის ვიზუალურ გამოსახულებას, ორგანზომილებიან სურათს, იღებს. პრობლემა მეტად კომპლექსური ხდება როცა საქმე გვაქვს სხვადასხვა სახის სურათთან.

შეიძლება გვქონდეს ერთი და იმავე ადამიანის სახის გამოსახულება, მაგრამ სისტემისთვის მათი მსგავსების პოვნა რთულია. ადამიანის სახე იცვლის გარეგნობას, მიმიკებს, დამატებით ფაქტორია სურათის ხარისხი ან თუნდაც არა-თანაბარი განათება და სახის ჩრდილები. აქედან გამომდინარე ვიზუალური გამოსახულებებიდან უნდა გავათანაბროთ ეს გარემოებები და ისე შევადაროთ სურათები ერთმანეთს.
სახის ამოცნობის ამოცანის შესასრულებლად საჭიროა შესრულდეს ოთხ ეტაპიანი პროცესი. პირველი ეტაპია სურათზე სახის სეგმენტირება, ანუ ვიპოვნოთ ადგილი სადაც ადამიანის სახეა. მეორე ეტაპია გავასწოროთ: სეგმენტ, სახის სეგმენტის სიმეტრიულობა, მისი ზომა, განათგანათებაებისა და ნაცრისფერი ტონები. მესამე ეტაპზე დამუშავებული სახის სურათიდან ვიღებთ სახის ნაკვთებს, როგორებისაა: თვალები, ცხვირ, პირი, ნიკაპი. ამ მონაცემების საშუალებით კი ვადგენთ სახის ვექტორს. ბოლო მეოთხე ეტაპზე მიღებულ ვექტორს ვადარებთ სხვა სახის ვექტორების ბაზას და ვეძებთ მსგავს სახეს.

სახის ამოცნობის მონაცემების სხვადასხვა ტიპი არსებობს, გამოიყოფა მთავარი ოთხი ტიპი, ტრადიციული, Human identification at a distance (HID) [2], სამ-განზომილებიანი და თერმული. ტრადიციული მეთოდი გულისხმობს სურათიდებს, HID არის დაბალი გარჩევადობის მქონე სურათიდან ამოცნობა, სამ-განზომილებიანი - ადამიანის სახის სამ-განზომილებიანი მოდელიდან ამოცნობა, ხოლო თერმული - ადამიანის სახიდან მიღებული ტემპერატურის შედეგად მიღებული მონაცემები. ნაშრომში განხილული იქნება ტრადიციული მეთოდი.

1.2 კვლევის ისტორია

პირველად მსგავს სისტემაზე მუშაობა, კომპიუტერის საშუალებით ადამიანის სახის ამოცნობა, 1960-იან წლებიდან დაიწყეს ვუდი ბლედსომ, ჰელენ ჩან ვოლფმა და ჩარლზ ბისონმა. მათ პროექტს უწოდეს “ადამიანი მანქანა”, რადგან სისტემას არ შეეძლო სახის ნაკვთების ავტომატურად გაზმოვა, ისინი სპეციალური ფირის გამოყენებით, რომელზეც ადამიანის სახის გამოსახულება იყო, ნიშნავდნენ 20-მდე სახის თვისებსბს, როგორიცაა: გუგების ცენტრები, თვალების შიდა და გარე კუთხეები, თვალის ჭრილის ზომა, პირი და ა.შ. ესეთი ხერხით ერთ საათში შესაძლებელი იყო დაახლოებით 40-მდე სურათის დამუშავება. მონაცემების შეგროვების შემდეგ კომპიუტერი ავტომატურად

ითვლიდა ადამიანების სახეების მსგავსებას, თუმცა სისტემა ვერ იყო ზუსტი და ამასთანავე მისთვის მონაცემების მოგროვება დიდ დროს მოითხოვდა.

1970 წელს იაპონელმა კომპიუტერულმა მეცნიერმა ტაკეო კანადემ წარადგინა სახის ამომცნობი სისტემა, რომელის ადამიანის ჩარევის გარეშე იღებდა სურათიდან სახის ნაკვთებს და ითვლიდა სხვა სახის მონაცემების ნაკვთებს შორის მანძილს. მოგვიანებით ცდებმა აჩვენა, რომ სისტემა ყოველთვის ვერ აღიქვამდა ნაკვთებს, თუმცა მისმა კვლევამ გარდატეხა მოახდინა სახის ამოცნობის განვითარებაში და 1977 წელს გამოაქვეყნა წიგნი, სადაც დეტალურად იყო აღწერილი სახის ამოცნობის ტექნოლოგია.

1980-იანი წლების ბოლოს და 1990-იანი წლების დასაწყისამდე მნიშვნელოვნად განვითარდა სახის ამოცნობა წრფივი ალგებრის გამოყენებით. ამ მიდგომას ეწოდება Eigenface [3] მეთოდი, მისი საშუალებით შესაძლებელი იყო დაბალგანზომილებიანი სურათებიდან სახის ამოცნობა, კვლევამ აჩვენა, რომ დაბალგანზომილებიანი სურათიდან სახის ნაწილების ანალიზიდან შესაძლებელი იყო სრულყოფილი სურათის წარმოდგენა. სწორედ Eigenface მეთოდია ღრმა სწავლების ალგორითმების საფუძველი.

1993 წელს აშშ-ს თავდაცვის მოწინავე კვლევების პროექტების სააგენტომ (DARPA) და არმიის კვლევის ლაბორატორიამ (ARL) შექმნეს სახის ამოცნობის ტექნოლოგია FERET [4], რათა განევითარებინათ სახის ამოცნობის შესაძლებლობები და უსაფრთხოების მიზნით მისი გამოყენება რეალურ გარემოში შესაძლებელი უნდა ყოფილიყო. კვლევების შედაგად დადგინდა, რომ ეს ტექნოლოგია კარგად მუშაობდა რეალურ გარემოში მკაფიო სურათებდან მიმართებით, ამის შემდეგ პროექტი კომერციული გახდა და რამდენიმე კომპანიამ კიდევაც შეისყიდა.

2000-იან წლებში აშშ-ს სტანდარტებისა და ტექნოლოგიების ეროვნულმა ინსტიტუტმა FERET გამოყენებით დაიწყო კვლევა The Face Recognition Vendor Tests (FRVT) [5], მათი მიზანი იყო სახის ამოცნობის ფართოდ კომერციალიზაცია და ადამიანებისთვის ამ ტექნოლოგიის გაცნობა.
2002 წელს ჩატარდა გამოწვევა სახელად The Face Recognition Grand Challenge (FRGC) [6], მისი მიზანი იყო FRVT-ზე უკეთესი სისტემის შემუშავება და ამოცნობის შეცდომის მინიმუმამდე დაყვანა. გამოწვევისთვის შეიმუშავეს სხვადასხვა ალგორითმები

სურათიდან, სამ-განზომილებიანი გამოსახულებიდან და თვალის ირისიდან ადამიანის სახის ამოცნობაზე. შედეგად მიიღეს 10-ჯერ უფრო ზუსტი ალგორითმი ვიდრე FRGC. ხოლო ზოგიერთმა ალგორითმა ადამიანებსაც კი აჯობა სახის ამოცნობაში და შეძლო ცალსახად გაერჩია იდენტური ტყუპები ერთმანეთისგან.

2010-იანი წლებიდან დაიწყო სახის ამოცნობის თანამედროვე ერა, როცა კომპიუტერები საკმარისად მძლავრები გახდნენ შესაძლებელი გახდა ნეირონული ქსელების გაწვრთნა და ამით მოდელების შექმნა.

2. სახის მოძებნა სურათზე

პირველი ნაბიჯი სახის ამოცნობისთვის არის გაგება თუ სად მდებარეობს სახე სურათზე, ზოგიერთ მოწყობილობას, კომპიუტერების ან სმარტფონების ზოგიერთ აპლიკაციას ჩაშენებული აქვს ეს ფუნქცია და როცა ჩვენ ვიღებთ სურათს ის წინასწა გვინიშნავს კვადრატით სახეებს. სწორედ ესაა პირველი ნაბიჯი რათა პროგრამას მივცეთ მხოლოდ სახის გამოსახულება და არა მთლიანი სურათი.

სახის მოსაძებნად გამოვიყენებთ მეთოდს “ორიენტირებული გრადიენტების ჰისტოგრამა” (Histogram of Oriented Gradients (HOG)) [7], ეს მეთოდი 2005 წელს შეიმუშავეს ნავნეტ დალალმა და ბილ ტრიგსმა. მეთოდის შინაარსი მარტივია: ვიღებთ სურათს, ვაქცევთ მას შავ-თეთრად, რადგან არ გვჭირდება ფერებით გადატვირთვა, თითოეულ ფიქსელზე ვაგებთ გრადიენტს, რომელიც მიუთითებს მის მეზობელ ყველაზე მუქი ფიქსელიკენ.

2.1 როგორ მუშაობს HOG

პირველ რიგში სურათზე უნდა ავაგოთ გრადიენტები. ამისათვის ვიღებთ 3x3 განზომილების მქონე ფიქსელების ბლოკს და თითოეული ფიქსელისთვის ვითვლით ვერტიკალურ და ჰორიზონტალურ გრადიენტებს G_x G_y.

$G_x(r, c) = I(r, c + 1) - I(r, c - 1)$

$G_y(r,c) = I(r-1,\ c) - I(r+1,c)$

სადაც r მიუთითებს სტრიქონზე, ხოლო c სვეტზე.
$I(r,c)$ არის შედარებითი სიკაშკაშე და RGB ფერის მქონე ფიქსელისთვის გამოითვლება შემდეგნაირად $0.2126 * R + 0.7152 * G + 0.0722 * B$

გრადიენტების გაგების შემდეგ გვჭირდება გამოვითვალოთ გრადიენტის მაგნიტუდა და მიმართულება. გრადიენტის მაგნიტუდა აღნიშნავს თუ რამდენად მკვეთრია ცვლილება პიქსელებს შორის, რაც უფრო დიდია მაგნიტუდა შესაბამისად პიქსელებზე სიკაშკაშის სხვაობა დიდია. ხოლო მიმართულება კი აღნიშნავს ღია ფიქსელიდან მუქისკენ მიმართულებას. ისინი გამოითვლებიან შემდეგი ფორმულებით:

$Magnitude(\mu) = \sqrt{G_x^2 + G_y^2}$

$Angle(\theta) = arctan\frac{G_y}{G_x}$

შემდეგი ეტაპია მიღებული მაგნიტუდის და მიმართულების მატრიცის საშუალებით ავაგოთ ჰისტოგრამა. ამისათვის საჭიროა დავყოთ სურათი 8x8 ფიქსელის ზომის ნაწილებად და თითოეული ნაწილისთვის ავაგოთ ჰისტოგრამა.

[0. 89. 89. 89. 89. 89., 89. 0.]	[0. 22. 96. 10. 32. 158. 80. 0.]
[180. 179. 8. 171. 165., 70. 97. 0.]	[478. 448. 380. 286. 162., 117. 78. 12.]
[180. 175. 176. 157. 139. 172. 89. 180.]	[514. 512. 468. 401. 303. 72. 78. 2.]
[180. 168. 164. 141. 123. 117. 114. 0.]	[288. 280. 266. 327. 393. 186. 57. 52.]
[180. 120. 109. 111. 105. 103. 90. 0]	[110. 131. 78. 174. 386. 293. 8. 32.]
[0. 86. 101. 87. 89., 95. 122. 180.]	[34. 129. 78. 142. 351. 309. 60. 22.]
[0. 23. 121. 89. 80. 83. 73. 0.]	[124. 42. 93. 211. 334. 278. 68. 24.]
[0. 89. 90. 90. 90. 90. 90. 0.]	[0. 68. 78. 264. 328. 258. 70. 0.]
გრადიენტის მიმართლება	გრადიენტის მაგნიტუდა

მიღებული მატრიცებით უნდა ავაგოთ ჰისტოგრამა, ჰისტოგრამა წარმოადგენს 0-დან 160 გრადუსამდე 20 გრადუსის ინტერვალით 9 მნიშვნელობას და თითოეული მნიშვნელობა აღნიშნავს გრადუსის ინტერვალში ინტენსიობას

მაგნიტუდა H									
B	0	20	40	60	80	100	120	140	160
j	0	1	2	3	4	5	6	7	8

ჰისტოგრამის აგების პრინციპი შემდეგია, თითოეული გრადიენტის მიმართულების მიხედვით ვჯამავთ გრადიენტის მაგნიტუდებს. მაგალითად (2;4) გრადნენტის მიმართულება არის 120, შესაბამისად (2;4) მაგნიტუდის მატრიციდან მნიშვნელობა 131 ჩავარდება ცხრილის იმ უჯრაში რომელიც მიუთითებს 120 გრადუსზე.

მაგნიტუდა H							131		
B	0	20	40	60	80	100	120	140	160

თუ მიმართულების მნიშვნელობა ზუსტად არ ემთხვევა ჰისტოგრამის მნიშვნელობას მაშინ მას ვანაწილებთ ორ უჯრაში შემდეგნაირად.

$H_{j+1} = (a - B_j)/\Delta\theta * m$

$H_j = (B_{j+1} - a)/\Delta\theta * m$

სადაც $\Delta\theta = 20$; m − მაგნიტუდა; a − მიმართულება

ხოლო j არის იმ პირველი უჯრის ინდექსი რომლის გრადუსიც ნაკლებია a − ზე.

თუ $a > 160$ მაშინ განვიხილავთ 160-180 შუალედს რაც ექვივალენტია 0 გრადუსიანი უჯრის.

მაგალითად: ავიღოთ (5;5) რომლის მნიშვნელობებია: მიმართულება a=105 და მაგნიტუდა m=386

j=5

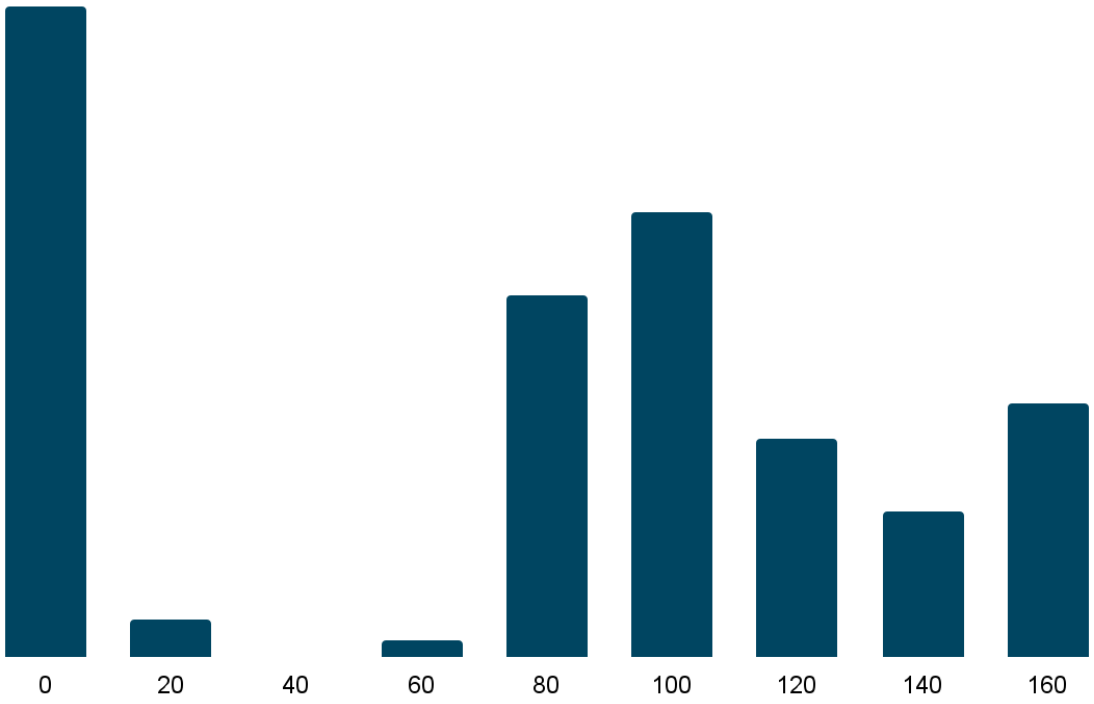
$H_5 = (120 - 105)/20 * 386 = 289.5$

$H_6 = (105 - 100)/20 * 386 = 96.5$

მაგნიტუდა H						96.5	289.5		
B	0	20	40	60	80	100	120	140	160

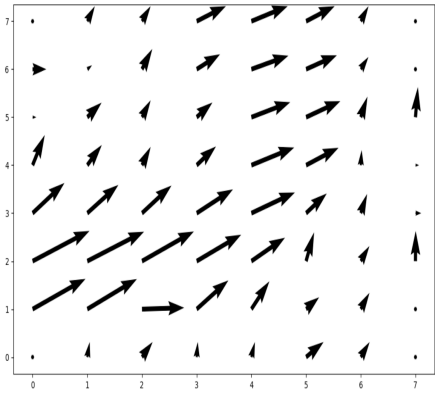
მთლიანი მონაცემებისთვის გვაქვს შედეგი:

მაგნიტუდა H	3232	187	6	82	1801	2207	1084	728	1260
B	0	20	40	60	80	100	120	140	160



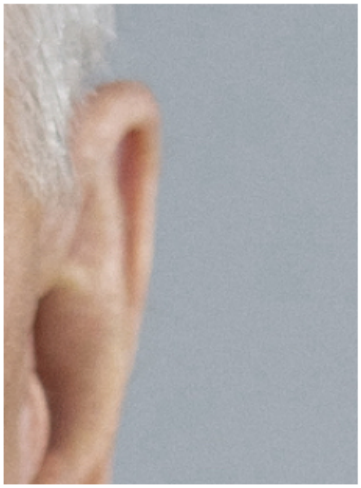
სურათი 1

სურათი 2

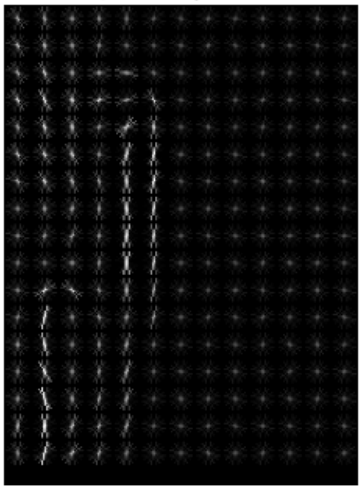


სურათი 3

სურათი

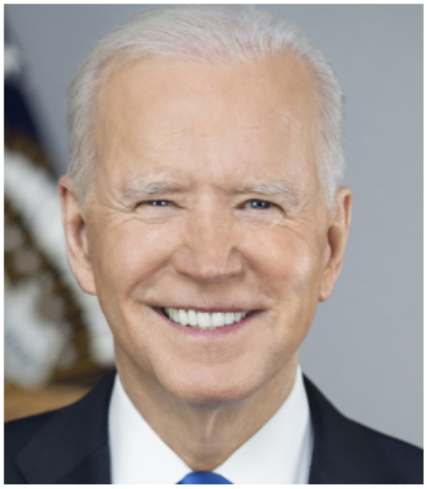


ორიენტირებული გრადიენტის ჰისტოგრამა

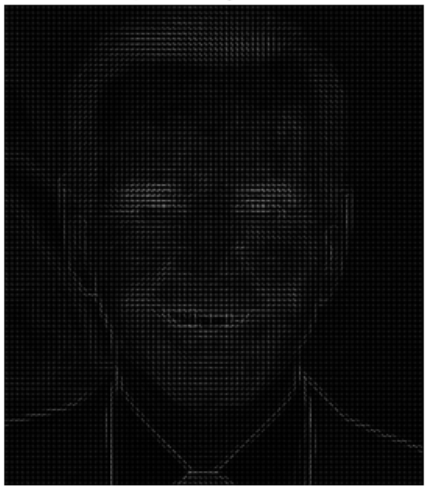


სურათი 4

სურათი



ორიენტირებული გრადიენტის ჰისტოგრამა



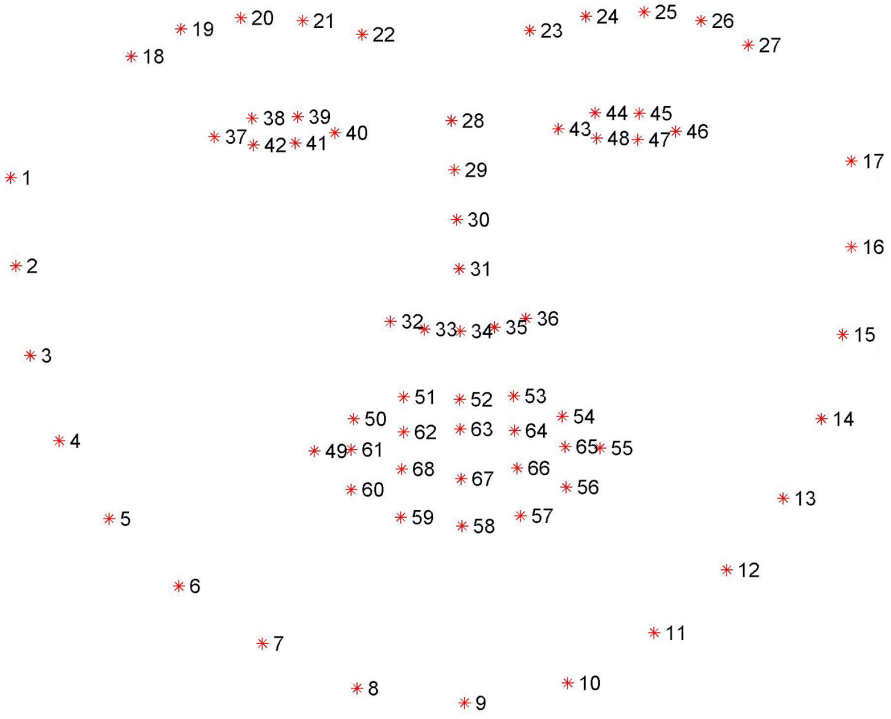
სურათი 5

აგებული ორინეიტრბული გრადიენტების ჰისტოგრამის საშუალებით უკვე გაწვრთნილი სახეებიდან ხვდება თუ სადაა სახე სურათზე, შესაბამისად ის გვიბრუნებს კოორდინატებს თუ სად მდებარეობს სახე.

2.2 სახის პოზიციის გასწორება და ნაკვთების მონიშვნა

სურათზე ადამიანის სახეს არ აქვს ყველთვის ერთი ორიენტაცია, შესაძლოა სახე გადახრილი იყოს ან თუნდაც სურათის რაკურსი სახის მიმათ არ იყოს პირდაპირი. იმისათვის რომ სახის ამოცნობა ყველა სურათისთვის თანაბარ პირობებში წარიმართოს საჭიროა სახის პოზიციის გასწორება, შედეგად მიღებული სურათი იქნება გასწორებული და შედარდება სხვა უკვე გასწორებულ სურათებს.

სურათის გასწორების შემდეგ საჭიროა სახის ნაკვთების მონიშვნა, როგორიცაა: პირი, ცხვირი, ნიკაპი, თვალები, წარბები და ა.შ. ამისათვის ჩვენ გამოვიყენებთ ბიბლიოთეკას Dlib [8], ესაა თანამედროვე C++-ზე დაწერილი ბიბლიოთეკა, რომელიც შეიცავს მანქანური სწავლების ალგორითმებს და გამოიყენება რეალური ამოცანების გადაწყვეტისთვის. საერთო ჯამში საჭიროა მოვნიშნოთ 68 წერტილი, ქვემოთ მოცემულ სურათზე გმაოსახულია Dlib-ის წერტილების მოდელი, სადაც ნაჩვენებია წერტილები 1-დან 68-ის ჩაჩთვლით.



სურათი 6

Dlib-ს უკვე აქვს წინასწარ შექმნილი მოდელი რომელიც იცნობს სახეს, მოვიყვანოთ კოდის ფრაგმენტი რომელიც გვაძლებს 68 წერტილს კოორდინატებით (x,y):

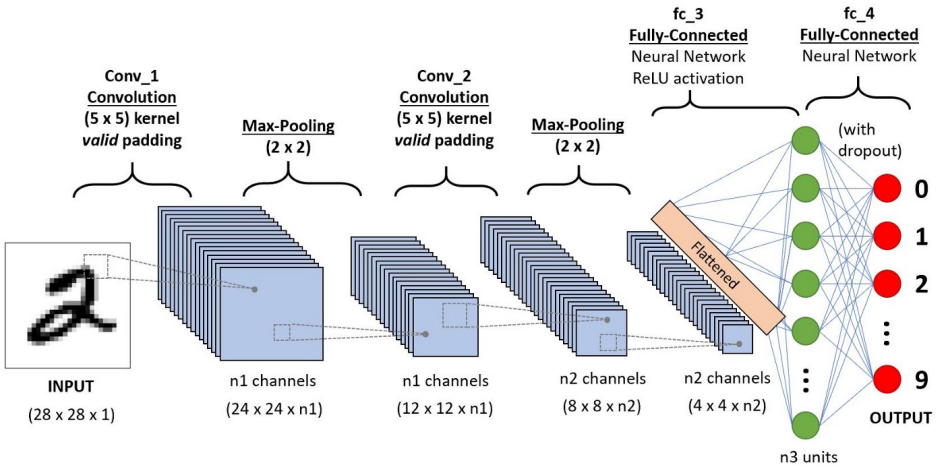
```
# წინასწარ დატრენინგებული მოდელი 68 წერტილის საპოვნელად
model_path = "shape_predictor_68_face_landmarks.dat"
# სახის ამოსაცნობი მეთოდი
frontal_face_detector = dlib.get_frontal_face_detector()
# სახის წერტილების მეთოდი
face_landmark_detector = dlib.shape_predictor(model_path)
# სურათის წაკითხვა
img = cv2.imread(“image.png”)
image_RGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# სურათზე სახეების პოვნა
allFaces = frontal_face_detector(image_RGB, 0)
# სახის კოორდინატების პოვნა სურათზე
face_rectangle_dlib = dlib.rectangle(int(allFaces[k].left()), int(allFaces[k].top()),
                                     int(allFaces[k].right()), int(allFaces[k].bottom()))
# წერტილების ამოღება
landmarks = face_landmark_detector(image_RGB, face_rectangle_dlib)
```

(254,637),(262,733),(273,827),(291,919),(329,1007),(391,1086),(461,1153),(540,1213),(637,1226),(731,1204),(807,1138),(870,1062),(925,980),(959,890),(974,799),(975,708),(971,616),(341,576),(386,540),(448,535),(512,542),(577,553),(667,563),(725,544),(787,532),(849,534),(894,572),(623,611),(626,678),(628,745),(631,816),(540,836),(582,854),(630,868),(675,854),(715,838),(415,617),(452,603),(485,603),(521,620),(485,617),(452,616),(719,624),(757,606),(789,604),(825,620),(790,622),(757,621),(441,927),(509,923),(576,923),(626,933),(673,926),(732,929),(788,938),(731,997),(672,1028),(622,1033),(571,1025),(504,991),(457,932),(577,939),(626,947),(673,942),(768,944),(671,996),(622,1002),(573,994)

სურათი 7

მიღებული მონაცემებით უკვე შეგვიძლია სახის გასწორება, ჩვენ უკვე ვიცით სადაა პირი და თვალები, შესაბამისად შეგვიძლია სახე მოვაბრუნოთ და გამოსახულება გავადიდოთ. მიღებულ მონაცემებს კი შემდეგი ეტაპისთვის გამოვიყენებთ.

2.3 კონვოლუციური ნეირონული ქსელი



სურათი 8

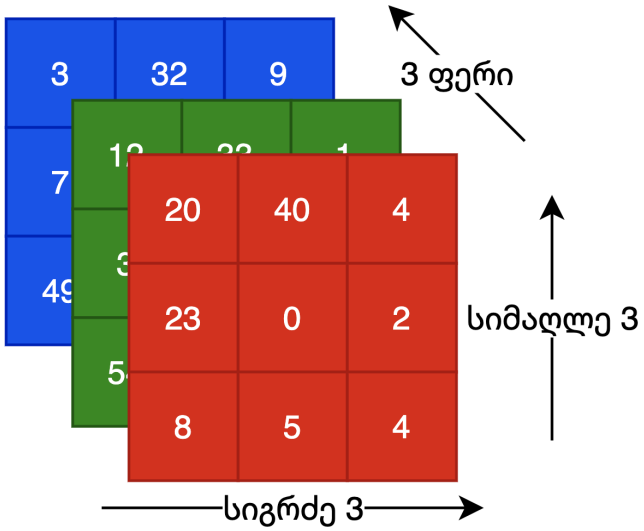
გამოსახულებების კლასიფიკაცია არის სხვადასხვა კატეგორიებად დაყოფის პროცესი მათი მახასიათებლების მიხედვით. მახასიათებლად შეიძლება ჩავთვალოთ პიქსელების ინტენსივობა, მათი მნიშვნელობა და ა.შ. სურათების კლასიფიკაციისთვის გამოიყენება ნეირონული ქსელები.

კომპიუტერული ხედვის განვითარება ღრმა სწავლების დამსახურებაა, მისი გამოყენებით ხელოვნური ინტელექტი დაიხვეწა დროთა განმავლობაში. ღრმა სწავლებაში უდიდესი წვლილი მიუძღვის კონვოლუციურ ნეირონულ ქსელს (Convolutional Neural Network - CNN) [9]. მოცემულ თავში განხილული იქნება სწორედ ეს ნეირონული ქსელი, რომელსაც იყენებს ჩვენი სისტემა ადამიანთა სახეების სწავლებაში.

კონვოლუციური ნეირონული ქსელი არის ღრმა სწავლების ალგორითმი, რომელიც იღებს სურათს, ანიჭებს მას მნიშვნელობას, წონას და შეუძლია განასხვაოს ერთი დანარჩენებისგან. CNN-ში წინასწარი დამუშავება გაცილებით მცირეა სხვა კლასიფიკაციის ალგორითმებთან შედარებით.

CNN-ის არქიტექტურა ადამიანის თავის ტვინიში ნეირონების კავშირის მსგავსია და სწორედ მისგანაა შთაგონებული. ცალკეული ნეირონები რეაგირებენ სტიმულებზე მხოლოდ ვიზუალური ველის შეზღუდულ ნაწილებში, რომელიც ცნობილია როგორც მიმღები ველი. ამ ველების ერთობლივობა კი მთელს ვიზუალურ არეალს ფარავს.

CNN-ს სხვადასხვა ფილტრების საშუალებით შეუძლია კარგად გაიგოს სურათში სივრცე-დროითი დამოკიდებულებები. მისი არქიტექტურა კარგად ერგება სურათების მონაცემთა ბაზას გამოსაყენებელი პარამეტრების შემცირებითა და წონების ხელახალი გამოყენებით. უკეთ რომ ვთქვათ, ქსელს შეუძლია იყოს კარგად გაწვრთნილი, რომ გაიგოს სურათების სპეციპიკა.

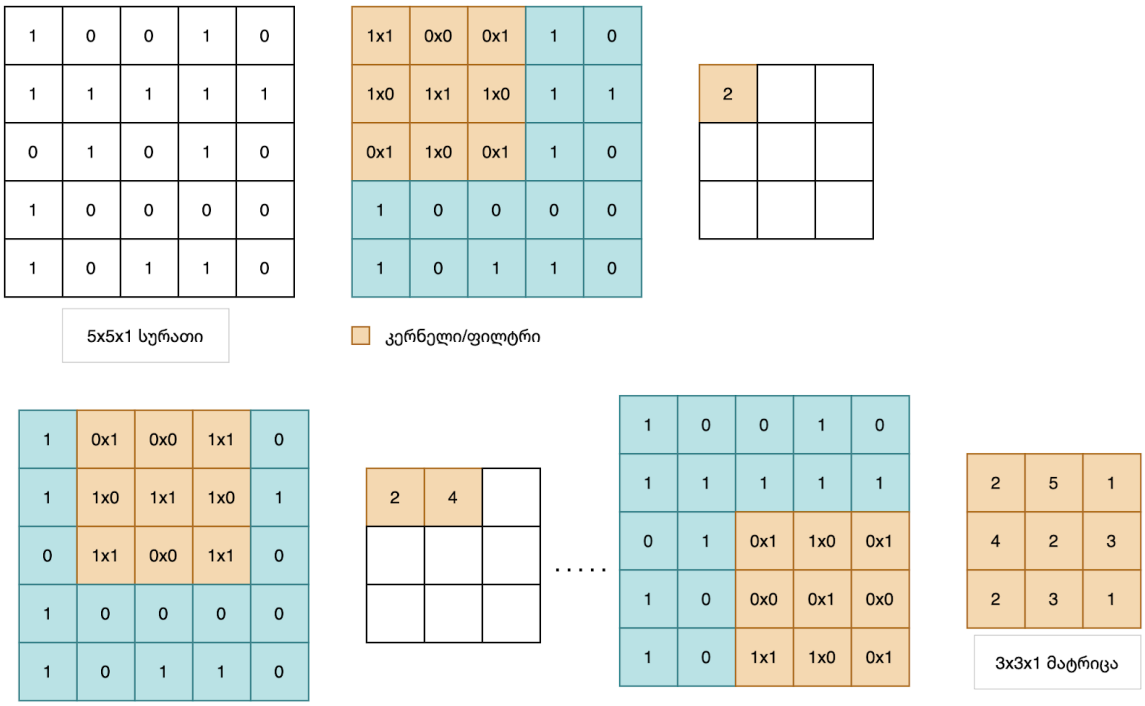


სურათი 9

მოცემულ სურათზე გამოსახულია RGB გამოსახულება, მას აქვს სამი ფერის შრე წითელი, მწვანე და ლურჯი. თუმცა არსებობს სხვა ფორმატები რომელშიც სურათი შეიძლება იყოს გამოსახული მაგალითად Grayscale, HSV, CMYK.

მოცემულ მაგალითზე სურათის ზომაა 3x3, თუმცა წარმოვიდგინოთ რა გამოთვლებია საჭირო როდესაც გვაქვს დიდი ზომის სურათი, მაგალითად 4K მისი ზომებია 3840 x 2160. კონვოლუციური ნეირონული ქსელის ფუნქციაა ამ მონაცემების შემცირება, ისე რომ გამოთვლები გამარტივდეს, ხოლო სურათიდან ისე ამოიღოს ინფორმაცია, რომ არ დაკარგოს მისი შინაარსი, რაც გადამწყვეტი იქნება საბოლოო შედეგის მისაღებად. ეს ფაქტი გადამწყვეტია იმ შემთხვევაში, როდესაც ვქმნით რაღაც არქიტექტურას, რომელიც ეფექტურია არა მხოლოდ სწავლაში, არამედ მდგრადია დიდი მონაცემების მქონე ამოცანებთან.

2.3.1 კერნელი

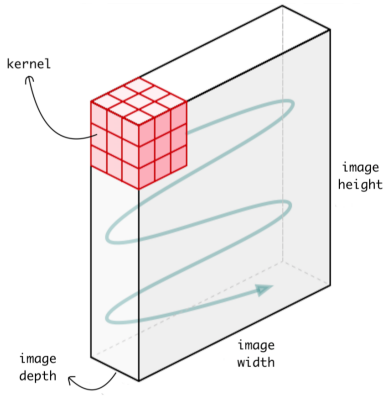


სურათი 10

განხილულ მაგალითზე მოცემულია 5x5x1 სურათი, ყვითელი ფერით აღნიშნულია ფილტრი ე.წ. კერნელი. კერნელი გადაადგილდება 9 ჯერ მატრიცაში, ყვითელი ფერით მონიშნული ზონა არის 3x3 მატრიცა P. ხოლო კერნელად გვაქვს აღებული შემდეგი მატრიცა:

$$K = \begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix}$$

ორი მატრიცის ერთმანეთზე გამრავლებით კი ვიღებთ საბოლოო პასუხს.



სურათი 11

კერნელის ფილტრი მოძრაობს მარჯვნივ, სანამ არ გაივლის მთელ სიგანეს, შემდეგ ის ისევ დაიწყებს მოძრაობას სურათის მარჯვნიდან თუმცა ჩამოვა ერთი ნაბიჯით ქვემოთ, პროცესი გაგრძელდება მანამ, სანამ არ გაივლის მთელ სურათს.

იმ შემთხვევაში როცა გვაქვს მრავალი არხის მქონე სურათი, მაგალითად RGB. ბირთვს აქვს იგივე სიღრმე რაც შეყვანის გამოსახულებას. მატრიცული გამრავლება ხდება K_n-სა და სტეკს შორის. თუ სტეკი არის I_n მაშნ გვაქვს ($[K_1, I_1]$; $[K_2, I_2]$, $[K_3, I_3]$) და მათი შეჯამებით მიიღება ერთ-განზომილებიანი გამოსახულება.

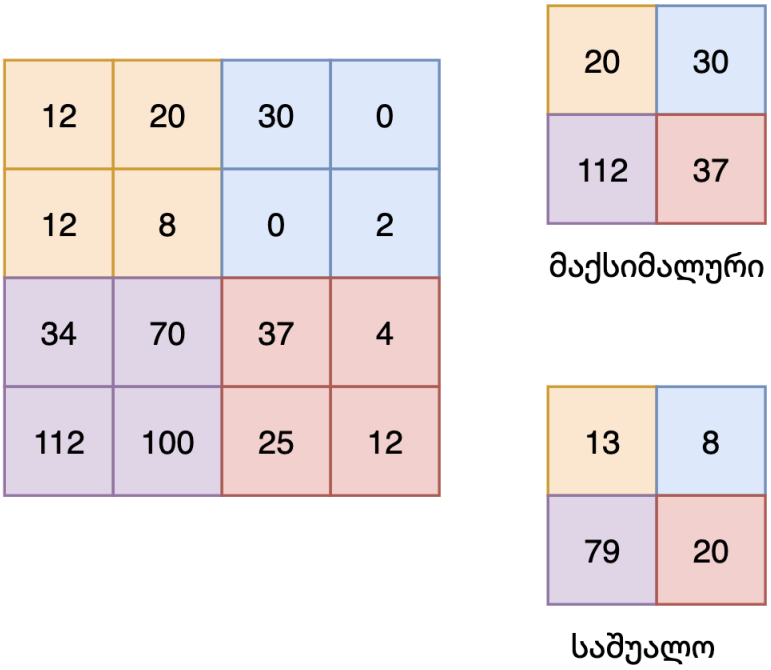
კონვოლუციური ოპერაციის მიზანია სურათიდან ამოიღოს მაღალი დონის მახასიათებლები, როგორებიცაა სილოეტი, CNN-ი სურათიდან არ ახასიათებს მხოლოდ ერთ შრეს, რადგან შესაძლოს სხვა ფენები პასუხისმგებლები იყვნენ სურათის ისეთ მახასიათებლებზე, რომლებიც სხვა ფენას არ გააჩნია. შედეგად ვიღებთ ისეთ გამოსახულებას, რომელსაც აქვს სრული ინფორმაცია.

კონვოლუციურ ოპერაციას ორი სახის შედეგი შელიძება ჰქონდეს - პირველი, სადაც მახასიათებლები შემციებულია შეყვანილ ინფორმაციასთან შედარებით და მეორე, რომელშიც განზომილება იზრდება ან იგივე რჩება. პირველ მეთოდს ეწოდება Valid Padding ხოლო მეორეს Same Padding. როცა 5x5x1 გამოსახულებას ვზრდით 6x6x1 გამოსახულებად და შემდეგ ვიყენებთ 3x3x1 კერნელს მასზე, შედეგად კონვოლუციური მატრიცა აღმოჩნდება იგივე ზომის 5x5x1 - ესაა Same Padding. ხოლო თუ ოპერაციას ჩავატარებთ გამოსახულების გაზრდის გარეშე მივიღებთ 3x3x1 მატრიცას რაც ტოლია კერნელის ზომის, ამას ეწოდება - Valid Padding.

2.3.2 გაერთიანების ფენა

კონვოლუციური ფენის მსგავსად გაერთიანების ფენაც მიზნად ისახავს კონვოლუციური მახასიათებლების შემცირებას, ამ პროცესის მიზანია გამოთვლების შემცირება განზომილების შემცირების ხარჯზე. მისი საშუალებით შესაძლებელია ამოვიღოთ დომინანტური მახასიათებლები და მივიღოთ უფრო ეფექტური სწავლების პროცესი.

არსებობს გაერთიანების ორი მიდგომა: მაქსიმალური და საშუალო. მაქსიმალურია როდესაც ბრუნდება მაქსიმალური მნიშვნელობა გამოსახულების ნაწილიდან, რომელსაც კერნელი ფარავს, ხოლო საშუალოა როდესაც კერნელის მიერ დაფარული მნიშვნელობებიდან ბრუნდება ყველა მნიშვნელობის საშუალო. მაქსიმალური გაერთიანება კარგად აქრობს სურათებიდან ე.წ. ხმაურს. ის საერთოდ უგულებელყოფს ხმაურს სურათიდან და ასევე სრულებს განზომილების შემცირებას. ხოლო საშუალო გაერთიანება უბრალოდ სურათის განზომილებას ამცირებს, აქედან გამომდინარე მაქსიმალური გაერთიანება უფრო ეფექტურია.



სურათი 12

კონვოლუციური და გაერთიანების ფენები ერთად ქმნიან კონვოლუციური ნეირონული ქსელის i-ურ ფენას. სურათების სირთულიდან გამომდინარე, ასეთი ფენების რაოდენობა შეიძლება გაიზარდოს დაბალი დონის დეტალების ამოსაებად, რომელიც მოითხოვს სისტემის მეტ სიმძლავრეს. აღნიშნული პროცესების შესრულების შემდეგ ჩვენს მოდელს მივეცით საშუალება გაარკვიოს სურათის მახასიათებლები. ახლა საჭიროა გავაბრტყელოთ მიღებული მონაცემები და მივაწოდოთ ნეირონულ ქსელს კლასიფიკაციის შესასრულებლად.

2.3.3 კლასიფიკაცია - სრულად დაკავშირებული შრე

სრულად დაკავშირებული ფენის დამატება არის ადვილი გზა მაღალი დონის მახასიათებლების არაწრფივი კომბინაციების სასწავლებლად. სულად დაკავშირებული ფენა არის სივრცეში შესაძლო არაწრფივი ფუნქციების სწავლება. ახლა, როცა შემომავალი სურათ გადავაქციეთ მრავალდონიანი პერცეპტრონის შესაბამის ფორმად, ჩვენ გავაბრტყელებთ სურათს ვექტორად. პროცესის მიმდინარეობის დროს მოდელს შეუძლია განასხვაოს დომინანტური და დაბალი დონის მახასიათებლები სურათში და მოახდინოს მათი კლასიფიკაცია.

3. სახის კოდირება

სახის ამოსაცნობად უმარტივესი გზა არის ავიღოთ პიროვნების ორი სურათი, პირველ სურათზე დანამდვილებით უნდა ვიცოდეთ რომ სწორედ ეს პიროვნებაა გამოსახული და შევადაროთ მეორე სურათს. თუ სახეები ერთმანეთს ჰგავს მაშინ შეგვიძლია ვთქვათ, რომ სურათებზე ერთი და იგივე პიროვნებაა გამოსახული.

როდესაც საქმე გვაქვს დიდ მონაცემებთან და გვინდა პიროვნების ამოცნობა, მაშინ საჭიროა მისი სახე შევადაროთ ყველა დანარჩენ სახესთან, ეს კი დიდ დროს მოითხოვს. ამიტომ საჭიროა მოვძებნოთ უფრო ეფექტური საშუალება მის ამოსაცნობად. იმისათვის რომ სახეები ერთმანეთს შევადაროთ საჭიროა თითოეულ სახეზე ინფორმაციის ქონა. ეს ინფორმაცია უნდა შეიცავდეს სახის პარამეტრებს, მაგალითად: თვალის ცენტრებს შორის მანძილი, ცხვირის ზომა, წარბების ფორმა, ტუჩების ფორმა და ა.შ.

სახიდან ინფორმაციის მისაღებად საჭიროა დავატრენინგოთ Deep Convolutional Neural Network კერძოდ FaceNet-ის [10] მეთოდი, ის სწავლობს და შეუსაბამებს სახეებს ადგილს სახეების რუკაზე მრავალგანზომილებიან სივრცეში, სადაც წერტილებს შორის მანძილი პირდაპირ შეესაბამება სახეებს შორის მსგავსებას.

3.1. მოდელის გაწვრთნა

მეთოდი მდგომარეობს შემდეგში: ვიღებთ სამ სურათს, აქედან ორი სურათი არის ერთი და იმავე ადამიანის, ხოლო მესამე სხვა ადამიანის. დატრენინგების შემდეგ მოდელი ორ ერთნაირ სურათს განათავსებს სივრცეში ერთმანეთთან ახლოს, ხოლო მესამეს კი შორს. თუ ამ მეთოდს გავაკეთებთ დიდ მონაცემებზე სახეების სხვადასხვა კონფიგურაციებისთვის მივიღებთ მით უფრო ზუსტ მოდელს, სადაც ერთი და იმავე ადამიანების მონაცემები სივრცეში ერთად განლაგდება.

ამოცანის გადასაწყვეტად საჭიროა გავიაროთ ორი ეტაპი, ესენია:

- Embedding - პროცესი რომელის ფუნდამენტურია FaceNet-ის მუშაობისთვის, ეს პროცესი სწავლობს სახეების განთავსებას მრავალგანზომილებიან სივრცეში, სადაც მანძილები სახის მსგავსებას შეესაბამება.
- კლასიფიკაცია - რომელიც სახეებისგან მიღებულ ინფორმაციას იყენებს მრავალგანზომილებიან სივრცეში განსხვავებული სახეების გამოსაყოფად.

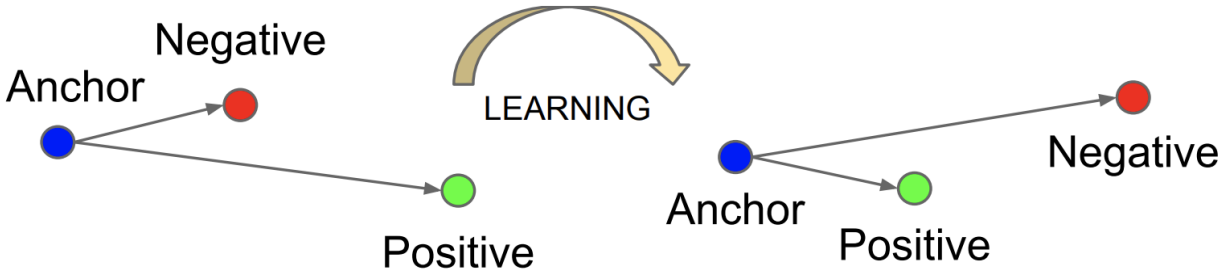
ჩვენ ასევე გვჭირდება დავნერგოთ დამატებითი ფუნქცია წონისთვის. ესაა პროცესი, რომელიც ტრენინგს ორ ნაწილად ჰყოფს. პირველი - ასწავლის მთელ მოდელს დიდ მონაცემებზე, ხოლო მეორე - ასწავლის ბოლო შრეს ახალ მონაცემზე პირველი ნაწილიდან მიღებული ცოდნის საფუძველზე. წონის ანაბეჭდი გვაძლევს საშუალებას სწავლება მოხდეს მოწყობილობაზე მხოლოდ რამდენიმე სურათის გამოყენებით. მას შეუძლია შეისწავლოს სახის თვისებები, რომლებიც აქამდე არ უნახავს და შეადაროს ეს ცოდნას იმას რაც უკვე მანამდე ჰქონდა. ქვემოთ დეტალურად აღვწეროთ მისი მუშაობის პრინციპი.

ვექტორი წარმოდგენილია როგორც, $f(x) \in \mathbb{R}^d$. ის გარდაქმნის x სურათს d განზომილებიან ევკლიდურ სივრცეში. ჩვენ გვინდა დავრწმუნდეთ, რომ ყოველი $x_i^a(anchor)$ პიროვნების სურათი ევკლიდურ სივრცეში ახლოსაა $x_i^p(positive)$ სურათებთან, რომელიც ეკუთვნის იმავე პიროვნებას, ხოლო შორს უნდა იყოს $x_i^n(negative)$ სურათებთან, რომელიც ეკუთვნის სხვა ადამიანებს.

აქედან გამომდინარე ჩვენ გვინდა: $||x_i^a - x_i^p||_2^2 + \alpha < ||x_i^a - x_i^n||_2^2, \forall (x_i^a, x_i^p, x_i^n) \in T$ (1)

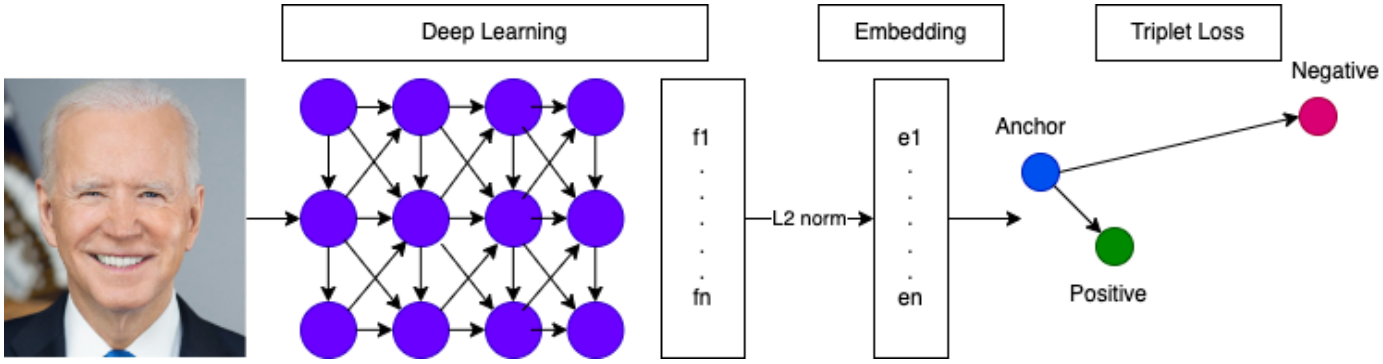
სადაც α არის მანძილი პოზიტივსა და ნეგატივს შორის, ხოლო T არის ყველა შესაძლო სამეული დასატრენინგებელ სიმრავლეში. აქედან სამეულის ცდომილება L გამოითვლება შემდეგნაირად:

$$L = \sum_i^N [||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha]_+$$



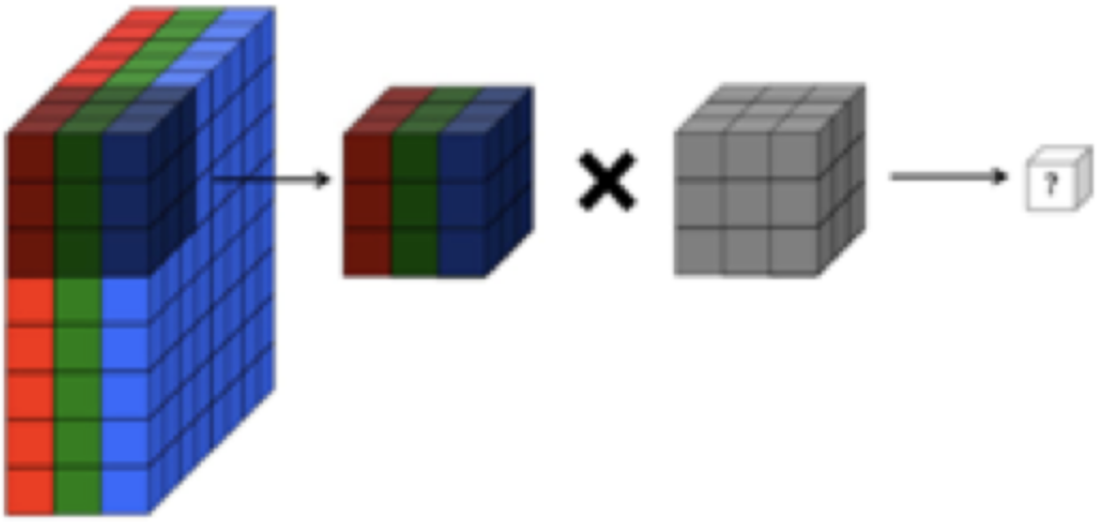
სურათი 13

ცდომილება გვიჩვენებს ორი სურათიდან, სადაც ერთი და იგივე ადამიანია, გამოსახული რამდენად ახლოს არიან ერთმანეთთან და ამასთანავე რამდენად შორს მესამე სურათთან, სადაც სხვა ადამიანის სახეა.



სურათ 14

ახლა განვიხილოთ რეალური მოდელირება, რომელიც იყენებს კონვოლუციურ ნეირონულ ქსელს (CNN), სურათზე ნაჩვენებია თუ რას აკეთებს პირველი კონვოლუციური ფენა მიწოდებულ სურათზე, რომელიც გამოსახულია როგორც RGB მატრიცა, ნაცრისფერი კუბი არის ფილტრი, რომცელიც ცვლის დასატრენინგებელ სურათს ფიქსელების წონების მიხედვით, მიღებულ შედეგს ეწოდება სახის ნაკვთების რუკა. შედეგად კონვოლუციური ნეირონული ქსელის სხვადასხვა სიღრმეებზე ვღებულობთ ახალ შრეებს.



სურათი 15

3.2 სახის ამოცნობა

შევაჯამოთ ზემოთ აღწერილი მეთოდი: დასატრენინგებელი სურათი მიეწოდება ამომცნობ ქსელს, რომელიც შემდეგ გადადის კონვოლუციურ ფენაში, ეს ფენა კიდევ სხვა ფენაში და ეს პროცესი მეორდება. რაც უფრო სირმეში ჩადის პროცესი მით უფრო რთულდება ფილტრები. ქსელის ბოლო ფენა დიდია და ის შედგება დაახლოებით 600-ზე მეტი ფილტრისაგან.
საბოლოოდ ეს ინფორმაცი მცირდება 128 განზომილებიან ვექტორამდე, რომელიც თითოეული სურათისთვის უნიკალურია. შედეგი FaceNet-ის მეთოდის საშუალებით განლაგდება ევკლიდურ სივრცეში ისე რომ ერთი და იგივე ადამიანის სურათები ერთმანეთთან ახლოს იქნება.

განვიხილოთ 128 განზომილებიანი ვექტორის ამოღება სურათ X-ის მაგალითზე, სადაც გამოვიყენებთ ბიბლიოთეკებს: OpenCV [11], numpy [12] და face_recognition [13], რომელიც იყენებს Dlib ბიბლიოთეკას:

```
import cv2 as cv
import face_recognition
import numpy as np
detector = cv.CascadeClassifier('haarcascade_frontalface_default.xml')
np_arr = np.fromstring(image_bytes, np.uint8)
frame = cv.imdecode(np_arr, 1)
gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
rgb = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
rects = detector.detectMultiScale(gray)
boxes = [(y, x + w, y + h, x) for (x, y, w, h) in rects]
encodings = face_recognition.face_encodings(rgb, boxes)
print(embeddings)
```

სურათი 7-თვის მოცემული კოდი დაგვიბეჭდავს: [-0.1068, 0.1813, 0.0421, -0.0027, -0.0718, 0.0404, -0.073, -0.0607, 0.0444, 0.0073, 0.2117, -0.0686, -0.3087, -0.0026, 0.0465, 0.1337, -0.1228, -0.0192, -0.2147, -0.0642, 0.0226, 0.0251, 0.0841, 0.0012, -0.1355, -0.2654, -0.0503, -0.1434, -0.0613, -0.1053, 0.0763, -0.0297, -0.1942, -0.0719, -0.0307, -0.0162, -0.0493, -0.0301, 0.1668, -0.0174, -0.1544, 0.0688, 0.0186, 0.2363, 0.2859, 0.035, 0.0042, -0.0657, 0.1308, -0.2078, 0.0413, 0.0802, 0.1949, 0.0886, 0.0785, -0.0606, 0.0349, 0.196, -0.1946, 0.0759, -0.0147, -0.0648, 0.0451, -0.0524, 0.1529, 0.0918, -0.0532, -0.1282, 0.1951, -0.1029, -0.0893, 0.1063, -0.1475, -0.1899, -0.3623, 0.002, 0.2724, 0.0667, -0.2615, -0.1055, -0.0285, -0.0553, -0.0318, 0.0669, -0.0337, -0.169, -0.0405, -0.0021, 0.2862, -0.1235, -0.0496, 0.2767, 0.034, -0.1116, 0.0106, 0.0467, -0.081, -0.038, -0.1212, -0.0335, -0.0222, -0.1351, -0.0488, 0.0993, -0.2104, 0.1246, -0.0165, -0.0131, -0.0467, -0.0001, -0.0686, -0.0234, 0.249, -0.1837, 0.2148, 0.2348, -0.0229, 0.0289, -0.0156, 0.0889, -0.0118, 0.0732, -0.1617, -0.1209, 0.0734, 0.0226, -0.0318, 0.0719]

სურათი 16

განვიხილოთ სურათი 7-ს შედარება სურათ 16-თან, რომლის სახის ვექტორიცაა შემდეგი:
y=(-0.0365, 0.1121, 0.0586, -0.0049, -0.1587, -0.0404, -0.033, -0.0918, 0.0923, -0.0338, 0.1802, -0.0772, -0.2155, -0.0671, 0.025, 0.1185, -0.0722, -0.0664, -0.1012, -0.1047, -0.0324, 0.0164, -0.0088, -0.0079, -0.0716, -0.2753, -0.0721, -0.0768, 0.0867, -0.0908, 0.0047, 0.0132, -0.2414, -0.0409, 0.0166, 0.0706, -0.0627, -0.079, 0.1518, -0.038, -0.1606, 0.0443, 0.0425, 0.2187, 0.1952, -0.0402, 0.0232, -0.0772, 0.1165, -0.2036, 0.0096, 0.1605, 0.1162, 0.0644, 0.0453, -0.0928, 0.049, 0.1934, -0.1855, 0.0685, 0.1281, -0.0657, -0.0951, -0.1007, 0.218, 0.0875, -0.1001, -0.1554, 0.1411, -0.1877, -0.0415, 0.0709, -0.0902, -0.1504, -0.3119, 0.101, 0.3992, 0.2018, -0.2186, 0.0024, -0.0907, 0.0302, 0.0351, 0.0561, -0.0762, -0.0462, -0.0955, 0.0255, 0.1986, -0.0489, 0.0569, 0.2071, 0.0226, 0.0091, -0.0632, -0.027, -0.1042, -0.0649, -0.0636, -0.0378, 0.0396, -0.1728, 0.0334, 0.1676, -0.2481, 0.1979, -0.0242, -0.0182, 0.037, 0.026, 0.0301, -0.005, 0.1671, -0.2478, 0.2243, 0.1677, 0.0124, 0.0986, 0.025, 0.1439, -0.0575, -0.0221, -0.1414, -0.1012, -0.0392, -0.0577, -0.0058, 0.0433)

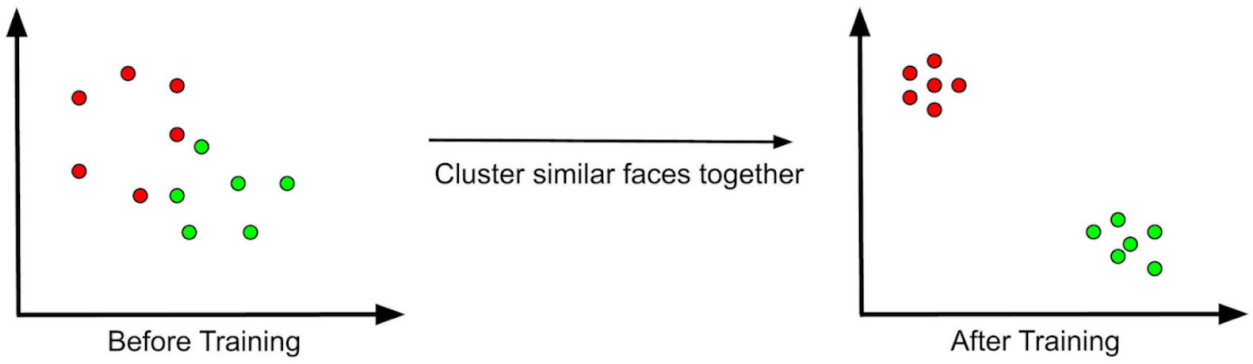
გამოვიყენოთ ორ წერტილს შორის მანძილის გამოსათვლელი ფორმულა,

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \approx 0.488$$

ხოლო თუ ჩვენ შევადარებთ x სურათზე გამოსახული ადამიანის სხვა სურათის ვექტორს:

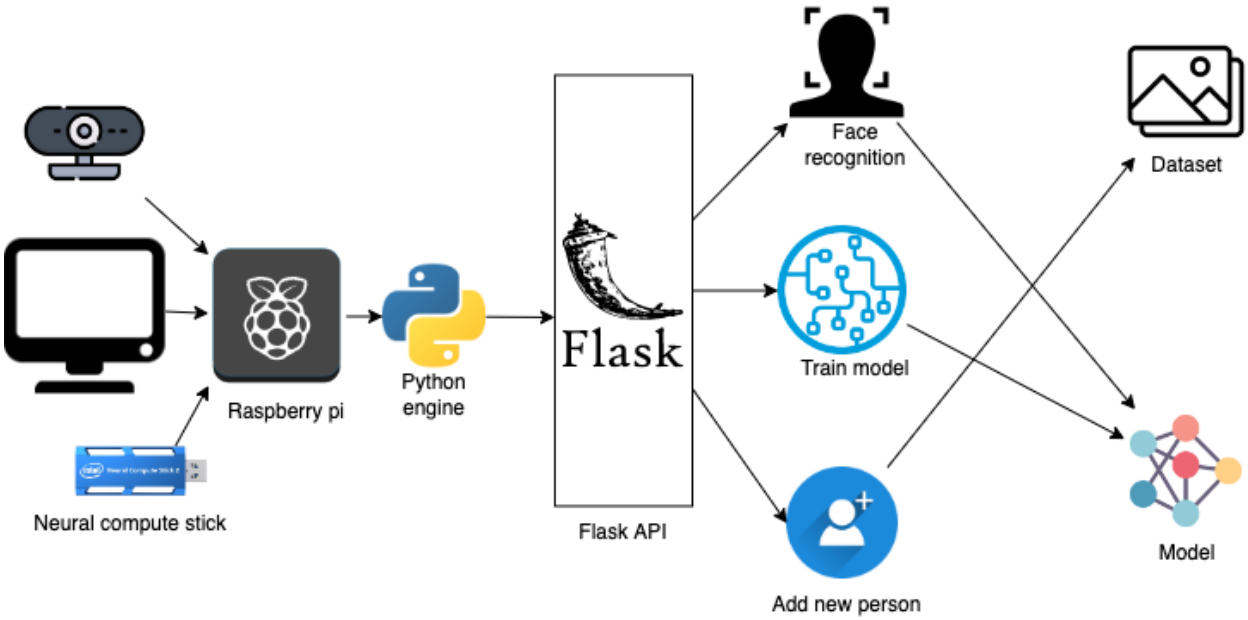
z=[-0.0365, 0.1121, 0.0586, -0.0049, -0.1587, -0.0404, -0.033, -0.0918, 0.0923, -0.0338, 0.1802, -0.0772, -0.2155, -0.0671, 0.025, 0.1185, -0.0722, -0.0664, -0.1012, -0.1047, -0.0324, 0.0164, -0.0088, -0.0079, -0.0716, -0.2753, -0.0721, -0.0768, 0.0867, -0.0908, 0.0047, 0.0132, -0.2414, -0.0409, 0.0166, 0.0706, -0.0627, -0.079, 0.1518, -0.038, -0.1606, 0.0443, 0.0425, 0.2187, 0.1952, -0.0402, 0.0232, -0.0772, 0.1165, -0.2036, 0.0096, 0.1605, 0.1162, 0.0644, 0.0453, -0.0928, 0.049, 0.1934, -0.1855, 0.0685, 0.1281, -0.0657, -0.0951, -0.1007, 0.218, 0.0875, -0.1001, -0.1554, 0.1411, -0.1877, -0.0415, 0.0709, -0.0902, -0.1504, -0.3119, 0.101, 0.3992, 0.2018, -0.2186, 0.0024, -0.0907, 0.0302, 0.0351, 0.0561, -0.0762, -0.0462, -0.0955, 0.0255, 0.1986, -0.0489, 0.0569, 0.2071, 0.0226, 0.0091, -0.0632, -0.027, -0.1042, -0.0649, -0.0636, -0.0378, 0.0396, -0.1728, 0.0334, 0.1676, -0.2481, 0.1979, -0.0242, -0.0182, 0.037, 0.026, 0.0301, -0.005, 0.1671, -0.2478, 0.2243, 0.1677, 0.0124, 0.0986, 0.025, 0.1439, -0.0575, -0.0221, -0.1414, -0.1012, -0.0392, -0.0577, -0.0058, 0.0433]

მაშინ $d(x, z) \approx 0.069$, თუ შევადარებთ ამ ორ რიცხვს ერთმანეთს თვალსაჩინოა თუ რამდენად ახლოსაა ერთი და იმავე პიროვნების სურათი ერთმანეთთან და ამასთანავე შორს სხვა პიროვნების სურათთან.



სურათი 17

4. პროექტის სტრუქტურა



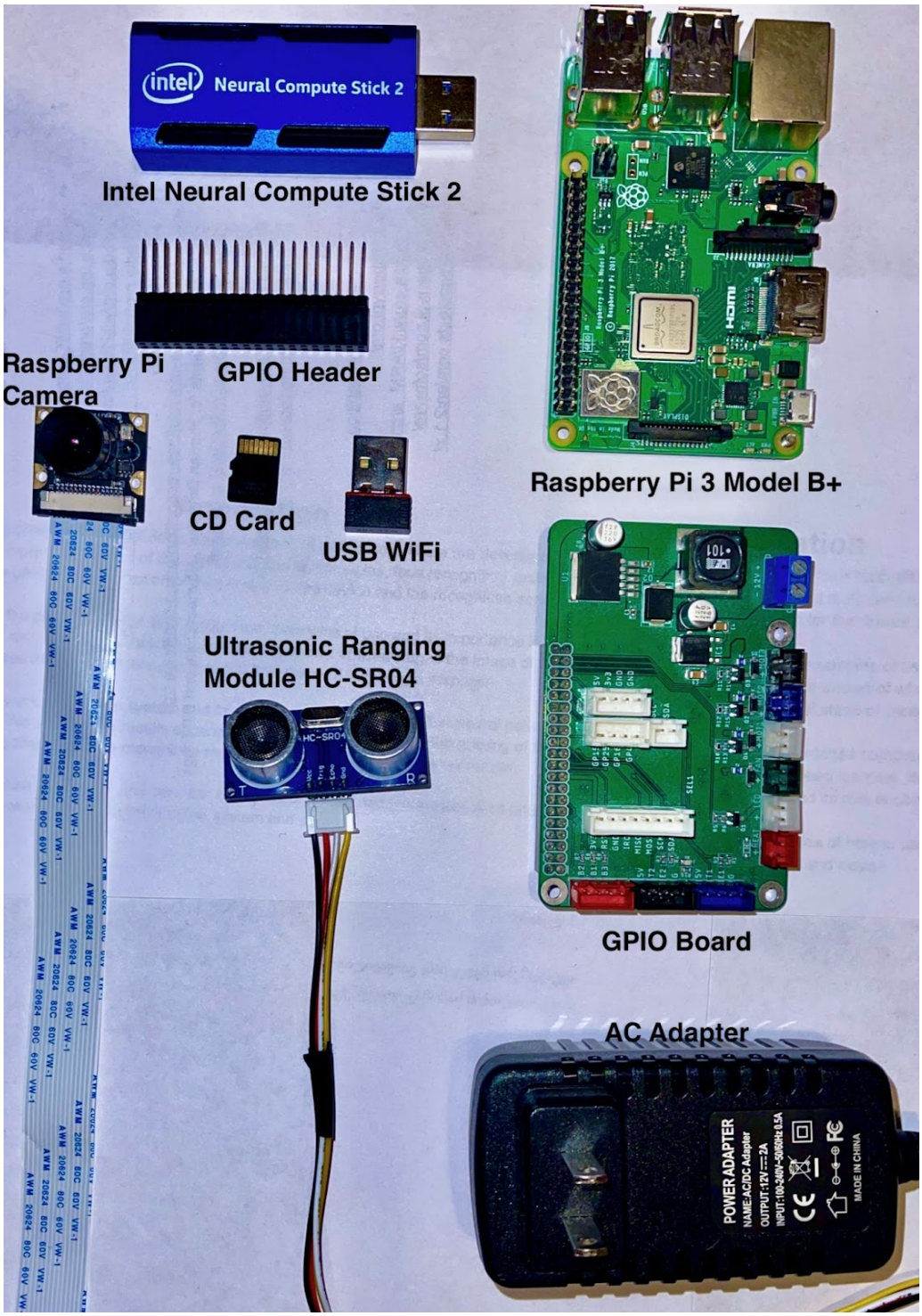
სურათი 18

პროექტი შეგვიძლია დავყოთ ორ ნაწილად: მოწყობილობის მხარე [14] და სახის ამოცნობის სერვისი [15]. მათი საშუალებით მივიღებთ პროდუქტს, რომელსაც შეუძლია ღრმა სწავლების გამოყენებით ადამიანთა იდენტიფიკაცია.

ორივე ნაწილი ერთმანეთისგან დამოუკიდებლად მუშაობს, ისინი ერთმანეთთან კავშირს ამყარებენ HTTP პროტოკოლის გამოყენებით. სერვერის არქიტექტურა ისეა მოწყობილი რომ, მას შეუძლია ერთზე მეტი მოწყობილობის ინფორმაციის დამუშავება, შესაბამისად შეგვიძლია გამოვიყენოთ ბევრი მოწყობილობა, დავაყენოთ სხვადასხვა ადგილას და მათი ინფორმაციის დამუშავებაზე პასუხისმგებელი იქნება მხოლოდ ერთი სერვერი.

სტრუქტურიდან გამომდინარე პროექტის კოდი დაყოფილია ორ ნაწილად, ორივე ნაწილში გამოყენებულია Python ენა და მისი ფრეიმვორკები. კოდის ორივე ნაწილი ღიაა ყველასთვის და ნებისმიერ მსურველს შეუძლია გამოიყენოს შესაბამისი მიზნებისთვის.

4.1 მოწყობილობა



სახის ამოცნობის რეალიზაციისთვის საჭიროა მოწყობილობა, რომელსაც შეუძლია გრაფიკული მონაცემების შეგროვება და მისი მიწოდება სახის ამომცნობ სერვისამდე, ასევე შედეგის სანახავად საჭიროა ეკრანი, სადაც მივიღებთშედეგს, აქედან გამომდინარე ეს მოწყობილობა უნდა იყოს კომპიუტერი, რომელსაც უნდა ჰქონდეს კამერა და საკმარისი სიმძლავრე მონაცემების დასამუშავებლად. მოწყობილობის პრაქტიკულობიდან გამომდინარე არჩევანი შეჩერდა მცირე ზომის ერთ-დაფიან კომპიუტერზე - Raspberry Pi [16], მას გააჩნია ARM [17] არქიტექტურის მქონე პროცესორი და აქვს სხვადასხვა პორტები მოწყობილობების დასაკავშირებლად, ჩვენს შემთხვევაში ერთ-ერთ პორტს გამოვიყენებთ გარე გამერისთვის. ქსელში ან ინტერნეტთან კავშირისთვის გააჩნია LAN პორტი, თუმცა უკაბელო კავშირისთვის ასევე აქვს WiFi მოდული.

Raspberry Pi-ს გააჩნია ARM არქიტექტურის მქონე პროცესორი, მასზე შესაძლებელია სხვადასხვა ოპერაციული სისტემის გამოყენება, მყარ მეხსიერებად ის იყენებს SD ბარათს, რაზეც შეგვიძლია დავაყენოთ სასურველი ოპერაციული სისტემა. ჩვენ შემთხვევაში ვიყენებთ Raspbian OS-ს [18] რომელიც წარმოადგენს Debian-ზე დაფუძნებულ Linux-ის კონტრიბუციას. ოპერაციულ სისტემაში გვჭირდება დავაყენოთ ყველა საჭირო ბიბლიოთეკა. რაც შეეხება მთავარ პროგრამას, რომელიც მოწყობილობის მხარესაა შექმნილია Python-ით, რომელიც სახის ამოცნობისთვის იყენებს ბიბლიოთეკას OpenCV-ის თავის მოდელთან ერთად.

Raspberry Pi არაა საკმარისად მძლავრი მოწყობილობა იმისათვის, რომ სწრაფად დაამუშაოს კამერიდან მიღებული მიმდინარე გამოსახულება. ამისათვის გამოვიყენებთ გარე მოწყობილობას სახელად Intel's Neural Compute Stick 2, რომელიც Raspberry Pi-სდაუკავშირდება USB პორტის საშუალებით. მასზე შესაძლებელია პროცესების გაშვება ბევრად სწრაფად ვიდრე Raspberry pi-ს პროცესორზე.

სახის ამოცნობის ოპტიმალური შედეგისთვის საჭიროა სურათის ახლოდან გადაება, რაც მოგვცემს კარგი გარჩევადობის სურათს, ამისათვის საჭიროა დავამატოთ მანძილის გამზომი მოწყობილობა, რომელიც მაღალი სიზუსტით გვიბრუნებს მანძილს სენსორიდან საგნამდე, ამ შემთხვევაში კი ადამიანამდე. მასთან მიახლოების დროს სისტემა აიღებს მანძილს, შეადარებს მინიმუმ მანძილს, რომელზეც საჭიროა სახის ამოცნობის გააქტიურება და თუ აკმაყოფილებს პირობას მაშინ მოწყობილობა გადაერთვება სახის ამოცნობის რეჟიმზე.

მოწყობილობის გამოყენების პრინციპი მდგომარეობს შემდეგში: პირველ რიგში გვჭირდება მონაცემთა შეგროვება და კლასიფიკაცია კონკრეტული პირის მიმართ, როდესაც მომხმარებელი დადგება კამერის პირისპირ მოწყობილობა გადაიღებს სურათებს და გააგზავნის სახის ამოცნობის სერვერზე შესაბამისი მომხმარებლის სახელით. ხოლო სახის ამოსაცნობად, როდესაც მომხმარებელი უკვე დამატებულია, ის დგება კამერის წინ, მისი სახის მონაცემები იგზავნება სახის ამომცნობ სერვერზე,

რომელიც გვიბრუნებს შესაბამის შედეგს. მნიშვნელოვანია ის ფაქტი რომ, რესურსების დაზოგვის მიზნით, სწორი სწავლებისა და ამოცნობისთვის საჭიროა მხოლოდ იმ სურათების გაგზავნა სადაც პროგრამამ სახე ამოიცნო, სწორედ ამიტომ ვიყენებთ OpenCV-ის ბიბლიოთეკას, რომელიც გვაძლევს საშუალებას ამოვიცნოთ სახე სურათიდან, ამოვჭრათ მხოლოდ სახე და გავაგზავნოთ ის სახის ამომცნობ სერვერზე.

მოწყობილობის გამოყენება შესაძლებელი იქნება ვერიფიკაციისთვის, ადამიანთა აღრიცხვისთვის ან უბრალოდ უსაფრთხოებისთვის. მოწყობილობის ზომებიდან გამომდინარე მისი დაყენება შესაძლოა ყველგან, მას მხოლოდ კვების წყარო ესაჭიროება. თუმცა ის მცირე ენერგიას მოიხმარს, ამოტომ შესაძლებელია მისი ბატარეაზე დაერთება. ხოლო რაც შეეხება ქსელში ჩართვას მას უკვე აქვს WiFi ადაპტერი, იმ შემთხვევაში როცა WiFi როუტერი შორსაა მოწყობილობიდან და მისი სიგნალი სუსტია შეგვიძლია დამატებით ადაპტერის შეერთება USB პორტში.

Raspberry Pi-ს მოწყობილობების დაკავშირება შესაძლებელია პორტებით, თუმცა ეს პორტები არაა სამარისი ან არათავსებადია ყველა მოწყობილობასთან. ამისათვის მას გააჩნია 40 ცალი პინი, არსებობს სპეციალური დაფა პორტებით, რომელიც ამ პინებზე ერთდება, შედეგად ვიღებთ კიდევ მეტ პორტს სხვა მოწყობილობების ან სენსორების დასაკავშირებლად. Raspberry Pi სიგნალებს აგზავნის პინებში და იღებს შესაბამის ინფორმაციას. პროექტის შემდგომი განვითარებისთვის ეს ფუნქცია დიდ შესაძლებლობას გვაძლევს დავამატოთ პროექტს სხვადასხვა მოწყობილობები და შესაბამისად ფუნქციები, მაგალითად სითბური ან მოძრაობის სენსორი.

პროექტის მომავალი განვითარებისთვის აუცილებელია ვერსიის კონტროლის გამოყენება მოწყობილობის მხარეს. რაცა გვინდა დავამატოთ ახალი ფუნქციონალი, გვიწევს ვერსიის განახლება მოწყობილობის კოდში. ამისათვის შეგვიძლია გამოვიყენოთ ვერსიის კონტროლი, ვამოწმოთ რაღაც დროის ინტერვალში მოწყობილობის ვერსიის შესაბამისობა და ცვლილების შემთხვევაში განვაახლოთ კოდი. იმ შემთხვევაში, როცა რამე პრობლემაა კონკრეტულ მოწყობილობაზე შეგვიძლია მას დავუკავშირდეთ SSH პროტოკოლით, ვნახოთ კონკრეტული მიზეზი ან თუნდაც ვმართოთ ის.

4.2 სახის ამოცნობის სერვერი

სახის ამოცნობის სერვერი წარმოადგენს Flask API-ს, [19] რომელას აქვს კომუნიკაცია სახის ამოცნობის სერვისთან. Flask არის Python-ის მიკრო ვებ ფრეიმვორკი, რადგან სახის ამოცნობის სერვისი წარმოადგენს Python აპლიკაციას, შესაბამისად საუკეთესო არჩევანი სწორედ Flask ფრეიმვორკია. მას გააჩნია სხვადასხვა როუთები: ახალი პერსონის შექმნის, მოდელის დატრენინგებისა და ამოცნობის. ის იღებს შესაბამის ბრძანებებს მოწყობილობიდან და მისი შინაარსიდან გამომდინარე აგზავნის მონაცემებს სახის ამოცნობის სერვისისკენ.

სახის ამოცნობის სერვისი, როგორც აღვნიშნეთ, არის Python სერვისი. მისი სტრუქტურა გამოიყურება შემდეგნაირად.

```
—dataset # დასატრენინგებელი სურათები პერსონების მიხედვით
        —person1
        —person2
—face_detection_model # სახის ამომცნობი მოდელები
            deploy.prototxt
            res10_300x300_ssd_iter_140000.caffemodel
—output # დატრენინგების შემდეგ მიღებული მოდელები
            embeddings.pickle
            le.pickle
            recognizer.pickle
add_person.py # პერსონის დამატება
recognise.py # ამოცნობა
train_model.py # მოდელის დატრენინგება
```

პერსონის დამატების დროს სურათები ინახება dataset დირექტორიაში შესაბამისი პიროვნების სახელით იქნება დირექტორია, სადაც ვინახავთ მხოლოდ ამ პიროვნების სურათებს. ამის შემდეგ კი ვუშვებთ *train_model-ს,* რომელიც ამუშავებს მონაცემებს და შესაბამის ინფორმაციას წერს output დირექტორიაში.

სახის ამომცნობი ნაწილი იღებს სურათს ან სურათებს, ამუშავებს მათ, იღებს სახის ვექტორს და დატრენინგებული მოდელის საშუალებით იგებს თუ ვინაა ეს პიროვნება. ეს პროცესი მეორდება ყველა მიღებული სურათისთვის და საბოლოოდ გპიბრუნდება იმ პიროვნების სახელი ვისაც ყველაზე მაღალი კოეფიციენტი აქვს.

აღსაღნიშნავია ის ფაქტი, რომ სახის ამოცნობის სერვერი დამოუკიდებლადაა მოწყობილობის ნაწილისგან, ამოტომ მისი გამოყენება მოწყობილობის გარეშეც შეიძლება. თუნდაც ვებ-კლიენტის მიბმით თავისუფლადაა შესაძლებელი ვებ აპლიკაციის შექმნა, რომლის front-end-დან შესაძლებელი იქნება იგივე მოქმედებების შესრულება რაც მოწყობილობიდანაა შესაძლებელი.

პროექტის განვითარებისთვის საჭიროა მისი მართვის გამარტივება, ამიტომ მომავალში იგეგმება სამ დონიანი ვებ აპლიკაციის შექმნა, რაც გულისხმობს სისტემის სამართავი გვერდის შექმნას. მისი გამოყენება შეეძლებათ მოწყობილობის მფლობელებს ან კომპანიებს, საიდანაც მოხდება მონაცემების ცვლილება, როგორიცაა: ახალი ადამიანის დამატება, სტატისტიკის ნახვა, უკვე დამატებული ადამიანის სატესტოდ ამოცნობა და ა.შ.

დასკვნა

კვლევის შედეგად გადაწყდა სახის ამოცნობის ამოცანა ღრმა სწავლების გამოყენებით და მოხდა მისი რეალიზება მოწყობილობაზე. ის უკვე წარმოადგენს პროდუქტს მოწყობილობის სახით და ასევე სერვისს, რომელიც ერგება ნებისმიერ გარემოს სახის ამოსაცნობად.

თეორიულად შევეხეთ ოპტიმალურ გზას იდენტიფიკაციის მაღალი მაჩვენებლის მისაღწევად, ხოლო პრაქტიკაში განვიხილეთ მაგალითები თუ როგორ მუშაობს აპლიკაცია. ასევე აღვწერეთ მოწყობილობა და სერვისთან ინფორმაციის მიმოცვლის გზები. ნაშრომის გაცნობის შემდეგ შესაძლებელია მსგავსი სისტემის დამოუკიდებლად აწყობა. აპლიკაციის ორივე ნაწილის კოდი ხელმისაწვდომია და მისი ინტეგრირება მარტივად შეიძლება. ის შეიძლება გამოყენებულ იყოს როგორ მხოლოდ beck-end-ი, ასევე მთლიანი პროდუქტი, საჭიროა მხოლოდ მოწყობილობისა და მისი ნაწილების მოძიება.

რადგან პროექტი რეალურ ამოცანასთანაა დაკავშირებული მისი გამოყენების ფართო შესაძლებლობაა. უსაფრთხოების, ვალიდაციის ან უბრალოდ აღწერისთვის საუკეთესო საშუალებაა. ის შეიძლება დაყენდეს სხვადასხვა დაწესებულებებში და გამოყენებულ იქნს კონკრეტული მიზნისთვის.

სამომავლო განვითარებისთვის შესაძლოა დაემატოს ახალი შესაძლებლობები: მოწყობილობის მხარეს დამატებით სენსორების მიერთება, ამოცანის შესაბამისი მოთხოვნებით, როგორიცაა: სითბური სენსორი, მოძრაობის დეტექტორი და ა.შ. ხოლო სერვისის მხარეს მომხმარებლის ინტერფეისის დამატება, რომელსაც ექნება სრული ვებ აპლიკაციის სახე და მისი საშუალებით შესაძლებელი იქნება აპლიკაციის მართვა.

გამოყენებული ლიტერატურა

1. Computer vision - Wikipedia https://en.wikipedia.org/wiki/Computer_vision
2. Human Identification at a Distance Using Body Shape Information https://iopscience.iop.org/article/10.1088/1757-899X/53/1/012058
3. Eigenface - Wikipedia https://en.wikipedia.org/wiki/Eigenface
4. FERET (facial recognition technology) - Wikipedia https://en.wikipedia.org/wiki/FERET_(facial_recognition_technology)
5. Repository for the Face Recognition Vendor Test (FRVT) https://github.com/usnistgov/frvt
6. Face Recognition Grand Challenge (FRGC) | NIST https://www.nist.gov/programs-projects/face-recognition-grand-challenge-frgc
7. Histograms of Oriented Gradients for Human Detection, Navneet Dalal and Bill Triggs http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf
8. http://dlib.net/
9. Convolutional neural network - Wikipedia https://en.wikipedia.org/wiki/Convolutional_neural_network
10. GitHub - davidsandberg/facenet: Face recognition using Tensorflow https://github.com/davidsandberg/facenet
11. https://opencv.org/
12. https://numpy.org/
13. face-recognition · PyPI https://pypi.org/project/face-recognition/
14. კოდი მოწყობილობის მხარისთვის https://github.com/davidiakobidze/face-recognition-rpi
15. კოდი სერვერი მხარისთვის https://github.com/davidiakobidze/face-recognition
16. https://www.raspberrypi.org/
17. ARM architecture family - Wikipedia https://en.wikipedia.org/wiki/ARM_architecture_family
18. Raspberry Pi OS https://www.raspberrypi.com/software/
19. https://flask.palletsprojects.com/en/2.1.x/