ԵՐԵՎԱՆԻ ՊԵՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

ԻՆՖՈՐՄԱՏԻԿԱՅԻ ԵՎ ԿԻՐԱՌԱԿԱՆ ՄԱԹԵՄԱՏԻԿԱՅԻ ՖԱԿՈՒԼՏԵՏ

ԾՐԱԳՐԱՎՈՐՄԱՆ ԵՎ ԻՆՖՈՐՄԱՑԻՈՆ ՏԵԽՆՈԼՈԳԻԱՆԵՐԻ ԱՄԲԻՈՆ

ԻՆՖՈՐՄԱՏԻԿԱՅԻ ԵՎ ԿԻՐԱՌԱԿԱՆ ՄԱԹԵՄԱՏԻԿԱՅԻ ԿՐԹԱԿԱՆ ԾՐԱԳԻՐ

ՄԵՀՐԱՔՅԱՆ ՄԱՐԻՆԵ ԳՐԻԳՈՐԻ

ԱՎԱՐՏԱԿԱՆ ԱՇԽԱՏԱՆՔ

ՓԱՍՏԱԹՂԹԵՐԻ ԿՆԻՔՆԵՐԻ և ՍՏՈՐԱԳՐՈՒԹՅՈՒՆՆԵՐԻ ՓՈՐՁԱՔՆՆՈՒԹՅԱՆ ՀԱՄԱԿԱՐԳԻ ՄՇԱԿՈՒՄ

«Ինֆորմատիկա. համակարգչային գիտություն» մասնագիտությամբ և Ինֆորմատիկայի բակալավրի որակավորման աստիճանի հայցման համար

ԵՐԵՎԱՆ 2024

Ուսանող՝_____

ստորագրություն

Մեհրաբյան Մարինե

_____

ազգանուն, անուն

Ղեկավար՝_____

ստորագրություն

Ֆ.մ.գ.թ., դոցենտ , Գասպարյան Հ. Վ.

_____

գիտ. աստիճան, կոչում, ազգանուն, անուն

«Թույլատրել պաշտպանության»

Ամբիոնի վարիչ՝ _____

ստորագրություն

Ֆ.մ.գ.թ., դոցենտ , Սարգսյան Ս. Գ.

_____

գիտ. աստիճան, կոչում, ազգանուն, անուն

«_____»_____2024թ.

Փաստաթղթերի կնիքների և ստորագրությունների փորձաքննության համակարգի մշակում

Разработка системы проверки печатей и подписей документов

Development of a document stamp and signature examination system

ՀԱՄԱՌՈՏԱԳԻՐ

Դիպլոմային այս աշխատանքը նախատեսված է օգնելու բազմաթիվ առաջադրանքներին: Այն առաջարկում է օգտակար գործիքների լայն շրջանակ՝ փաստաթղթերի հետ աշխատելն ավելի հեշտ և հուսալի դարձնելու համար:

Այս համակարգի առանցքը մեքենայական ուսուցման բարդ մոդելներն են, որոնք նախագծված են տարբերակելու իրական և կեղծված կնիքներն ու ստորագրությունները:

Աշխատանքը կենտրոնանում է նաև փաստաթղթերի պատկերներում կնիքների և ստորագրությունների հայտնաբերման վրա՝ օգտագործելով ալգորիթմներ, որոնք տալիս են գերազանց արդյունք՝ չնայած դասավորության տատանումներին, ֆոնի բարդությանը կամ աղմուկին:

Աշխատանքում անդրադառնում եմ նաև համընկնող (Overlap) կնիքների և ստորագրությունների պատկերներին՝ առաջարկելով սեգմենտավորման տեխնիկա՝ բաղադրիչներն առանձնացնելու համար:

Աշխատանքում նկարագրված է համակարգի ընդհանուր նախագծումը, ստեղծման ուղին, այս կամ այն խնդիրների իրականացման մոտեցումները, գործիքների կիրառման ընտրությունը և նկարագրությունը:

ԲՈՎԱՆԴԱԿՈՒԹՅՈՒՆ

ՆԵՐԱԾՈՒԹՅՈՒՆ

Ժամանակակից թվային աշխարհում, որտեղ գործարքներն ու պայմանագրերը գնալով ավելի շատ իրականացվում են էլեկտրոնային եղանակով, փաստաթղթերի իսկությունն ու ամբողջականությունը առաջնային նշանակություն ունեն: Այս փաստաթղթերում ստորագրությունները և կնիքները ծառայում են որպես իսկության հիմք՝ մարմնավորելով ներգրավված կողմերի համաձայնությունն ու ինքնությունը: Այս դիպլոմային աշխատանքը փորձում է նպաստել փաստաթղթերի իսկությունը հաստատելու մեթոդների առաջխաղացմանը՝ բազմակողմ հնարավորություններով փորձագիտական համակարգի մշակման միջոցով:

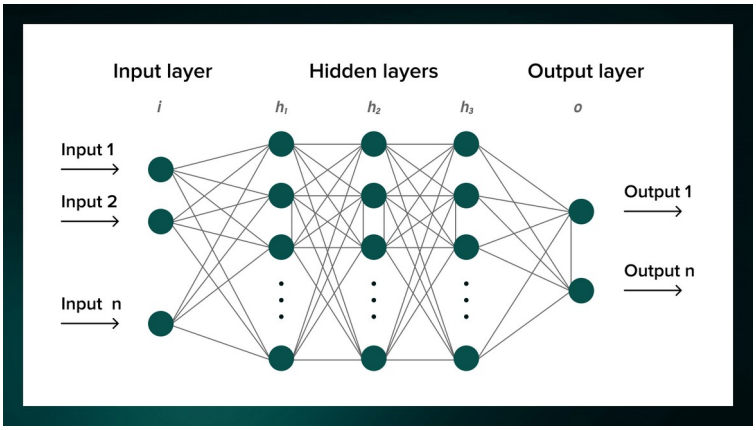
Փաստաթղթերի պատկերներից ստորագրությունների և կնիքների ավտոմատ հայտնաբերումը էական նշանակություն և արդիականություն ունի տարբեր ոլորտներում՝ ներառյալ իրավական, վարչական և ֆինանսական: Հաշվի առնելով դա, համակարգի նպատակներից է նախագծել ալգորիթմներ, որոնք փաստաթղթերի պատկերներից կարող են հայտնաբերել ստորագրությունները և կնիքները, ինչպես նաև overlap(ներդրված) պատկերների դեպքում կատարել առանձնացում: Այս ալգորիթմական լուծումները իրենցից ներկայացնում են արագ և պարզ մոտեցումներ, միաժամանակ ապահովելով բավարար արդյունք:

Թվային փաստաթղթերի տարածման և խարդախ գործողությունների աճման հետ մեկտեղ, աճում է փորձագիտական համակարգի անհրաժեշտությունը: Փաստաթղթերի իսկությունը ստուգելու ավանդական մեթոդները հաճախ հիմնվում են ձեռքով ստուգման վրա, սակայն նույնիսկ ամենահմուտ մասնագետները կարող են անտեսել նուրբ կեղծիքները՝ վտանգելով փաստաթղթերի անվտանգությունը: Այդ պատճառով համակարգը կենտրոնանում է նաև կնիքների և ստորագրությունների վավերացուման վրա: Այն տրամադրում է գործիքներ, որոնք կարող են ստուգել կնիքների և ստորագրության իսկությունը: Ստուգումն իրականացվում է մեքենայական ուսուցման միջոցով:

Մեքենայական ուսուցումը և խորը ուսուցումը արհեստական բանականության ենթաճյուղեր են, որոնց հիմքում ընկած ալգորիթմները հնարավորություն են տալիս համակարգիչներին սովորել տվյալներից և կանխատեսումներ կամ որոշումներ կայացնել առանց հստակ ծրագրավորման: Մեքենայական ուսուցման ալգորիթմները, ներառյալ նեյրոնային ցանցերը *(Neural Metwork)*, Decision Trees(որոշումների ծառեր) և *Support Vector Machines* (SVM, օժանդակ վեկտորային մեքենաներ), ի թիվս այլոց, կատարում են կանխատեսումներ կամ դասակարգումներ՝ բացահայտելու համար տվյալների օրինաչափություններն ու հարաբերությունները: Մեքենայական ուսուցումը և խորը ուսուցումը ներառում են բազմաթիվ շերտերով նեյրոնային ցանցեր (հետևաբար՝ «խորը» տերմինը), ինչը նրանց հնարավորություն է տալիս սովորել բարդ առանձնահատկություններ չմշակված տվյալներից, ինչպիսիք են պատկերները, տեքստը և ձայնը: Ցանկացած նեյրոնային ցանց՝ պարզ պերցեպտրոններից մինչև հսկայական AI-համակարգեր, բաղկացած է հանգույցներից, որոնք նմանակում են մարդու ուղեղի նեյրոններին: Այս բջիջները սերտորեն փոխկապակցված են:

Յուրաքանչյուր հանգույց կապված է նախորդ և հաջորդ շերտի որոշ հանգույցների հետ: Հանգույցը տեղեկատվություն է ստանում իր տակ գտնվող շերտից, ինչ-որ բան անում դրա հետ և տեղեկատվություն է ուղարկում հաջորդ շերտին:

Ստորև նշված նկարում պատկերված է նեյրոնային ցանցերի կառուցվածքը:



Նկար 1.1 Նեյրոնային ցանց

Այնուամենայնիվ, կարևոր է ընդունել ML(Machine Learning) և DL(Deep Learning) մոտեցումների հետ կապված մարտահրավերները: Այս մոդելները մեծապես կախված են ուսուցման համար մեծ, պիտակավորված տվյալների հավաքածուներից, որոնց ձեռք բերումը կարող է բարդ ինել:

Այս նախագծում առաջնորդվում ենք մի մոտեցմամբ, որը միավորում է ML/DL տեխնիկայի և computer vision ալգորիթմական մեթոդների ուժեղ կողմերը: Այս մոտեցումն ապահովում է արդյունավետություն, ճշգրտություն և մասշտաբայնություն:

ԽՆԴՐԻ ՆՊԱՏԱԿԸ

Խնդրի նպատակն է մշակել էքսպերտիզացիայի համակարգ, որը կարող է կատարել հետևյալ գործողությունները.

- Նախագծել և կիրառել ալգորիթմներ փաստաթղթերի պատկերներում կնիքները և ստորագրությունները հայտնաբերելու համար: Արդյունքը մեկ ելքային պատկեր է, որտեղ պահպանվում են միայն հայտնաբերված ստորագրությունները և կնիքները:
- Մշակել մեթոդ՝ կնիքների և ստորագրությունների համընկնող պատկերները բաժանելու համար: Այսինքն այն սցենարներում, որտեղ ստորագրությունները և կնիքները միմյանց վրա են, ալգորիթմը պետք է առանձնացնի այդ տարրերը՝ որպես ելք ունենալով երկու տարբեր պատկերներ՝ մեկը պարունակում է միայն ստորագրությունը, իսկ մյուսը՝ միայն կնիքը:
- Իրականացնել մեքենայական ուսուցման մոդելներ՝ կնիքների և ստորագրությունների իսկությունը ստուգելու համար: Ստորագրության ստուգումը պետք է հիմնվի այսպիսի հատկանիշների վրա ինչպիսիք են. գրիչի ճնշման տատանումները, թանաքի լայնությունը և հետևողականությունը, ելակետային հետևողականությունը, ստորագրության ընդհանուր հոսքը և ռիթմը: Կնիքի վավերացումը պետք է ստուգել հիմնվելով անոմալիաներ հայտնաբերելու տեխնիկայի վրա:
- Մշակել օգտվողի համար հարմար գրաֆիկական ինտերֆեյս (GUI)՝ համակարգից ավելի հեշտ օգտվելու համար:

ԽՆԴՐԻ ԴՐՎԱԾՔԸ

Դիտարկվող համակարգի մշակման համար անհրաժեշտ է հետևել ստորև նշված քայլերին.

- Կնիքների և ստորագրությունների վերլուծություն
- Տվյալների բազայի ձեռքբերում և վերլուծություն
- Փաստաթղթերի պատկերներում ստորագրության և կնիքների հայտնաբերում
- Համընկնող ստորագրությունների և կնիքների առանձնացում
- Տվյալների նախնական մշակում
- Ստորագրության վավերացման մոդելի ընտրություն
- Կնիքի վավերացման մոդելի ընտրություն
- Գրաֆիկական ինտերֆեյսի մշակում

ՕԳՏԱԳՈՐԾՎԱԾ ՏԵԽՆՈԼՈԳԻԱՆԵՐ

Այս նախագիծը օգտագործեց մի քանի Python գրադարաններ պատկերների մշակման, մեքենայական ուսուցման և տվյալների վիզուալիզացիայի համար.

- ✓ **OpenCV (cv2)**: Պատկերների մշակման առաջադրանքների համար, ինչպիսիք են նկարները կարդալը, շահարկելը և վերլուծելը:
- ✓ **Matplotlib**: Օգտագործվում է գրաֆիկների արտացոլման և գծագրման համար:
- ✓ **Scikit-image (skimage)**: Պատկերների մշակման առաջադրանքների համար, ինչպիսիք են չափումը, մորֆոլոգիան և առանձնահատկությունների արդյունահանումը:
- ✓ **NumPy**: Թվային հաշվարկների և զանգվածների մանիպուլյացիայի համար:
- ✓ **XGBoost (xgb)**: Գրադարան մեքենայական ուսուցման մոդելների գրադիենտ խթանման համար:
- ✓ **Scikit-learn (sklearn)**: Օգտագործվում է մեքենայական ուսուցման առաջադրանքների համար, ինչպիսիք են դասակարգումը, ռեգրեսիան և հիպերպարամետրերի կարգավորումը:
- ✓ **imbalanced-learn (imblearn)**: Անհավասարակշռված տվյալների հավաքածուներ մշակելու համար, հատկապես SMOTE-ի հետ չափից դուրս նմուշառման համար:
- ✓ **PIL (Python Imaging Library)**: Պատկերների մշակման առաջադրանքների համար, ինչպիսիք են կարդալը, գրելը և մանիպուլյացիաները:
- ✓ **Tkinter:** Ինտերֆեյսի միջերեսային միջավայր՝ աշխատասեղանի հավելվածներ մշակելու համար:
- ✓ **joblib**: Մեքենայական ուսուցման մոդելների պահպանման և բեռնման համար:
- ✓ **glob**: դիրեկտորիաներում ֆայլերի մշակման և որոնման համար**:**

ԿՆԻՔՆԵՐ ԵՎ ՍՏՈՐԱԳՐՈՒԹՅՈՒՆՆԵՐ

Փաստաթղթերի ոլորտում երկու կարևոր տարրերի հետ ենք առնչվում՝ կնիքներ և ստորագրություններ:

Փաստաթղթերի կնիքները, որոնք նաև հայտնի են որպես դրոշմներ, փաստաթղթերի վրա կիրառվող խորհրդանշաններ են: Կնիքները օգտագործվում են վավերացնելու պաշտոնական փաստաթղթերը, ինչպիսիք են լիցենզիաները, պայմանագրերը, թույլտվությունները կամ դատական որոշումները: Փաստաթղթի կնիքը ցույց է տալիս փաստաթուղթը թողարկող պաշտոնական անձը, օրինակ՝ պետական մարմինը, դատարանը կամ նոտարը: Դրոշմված կնիքները կիրառվում են հատուկ դիզայնով թանաքոտված ռետինե դրոշմակնիքի միջոցով: Փաստաթղթում իսկական կնիքի առկայությունը նշանակում է, որ փաստաթուղթը ծագել է լիազորված մարմնից և չի կեղծվել:

Կնիքները կարող են կանխել կեղծիքի փորձերը, քանի որ պաշտոնական կնիքի կրկնօրինակումը կարող է բարդ և չարտոնված գործողություն լինել: Ցավոք, թվային տեխնոլոգիաների աճը նպաստեց կեղծ կնիքների ստեղծմանը:

Ստորագրությունը որևէ անձի անվան, մականվան, նշանների ձեռագիր պատկերումն է, որի միջոցով տվյալ անձը հաստատում է ինքնությունը: Ստորագրությունը նույնականացնում է փաստաթուղթը ստորագրող անձին՝ հաստատելով նրա մասնակցությունը և համաձայնությունը դրա բովանդակության հետ: Վավեր ստորագրությունը ցույց է տալիս, որ ստորագրողը չի կարող հետագայում հերքել իր համաձայնությունը փաստաթղթին: Ստորագրությունները ևս ենթակա են կեղծիքի:

Փաստաթղթերի կեղծումը լուրջ խնդիր է բոլոր ոլորտներում: Կեղծ փաստաթղթերն օգտագործվում են ֆինանսական խարդախության համար (օրինակ՝ ապօրինի միջոցների համար ստորագրություններ կեղծելու), ինքնության գողության (կեղծ անձը հաստատող փաստաթղթերի օգտագործումը չարտոնված օգուտների համար) և պայմանագրային խարդախության համար (փաստաթղթերը փոփոխելու համար պայմանները շահարկելու կամ իրավական

պարտավորություններից խուսափելու համար): Սա հանգեցնում է ֆինանսական կորուստների, վնասված հեղինակության և իրավական հետևանքների:

Ավանդաբար, փաստաթղթերի ստուգումը ներառում է կնիքների և ստորագրությունների ձեռքով ստուգում մասնագետների կողմից: Այս մոտեցումը հիմնված է փորձաքննության վրա՝ հայտնաբերելու անհամապատասխանությունները և հնարավոր կեղծիքները: Նշեմ մի քանի բացասական կողմեր:

- Փաստաթղթերի, հատկապես մեծ ծավալների ձեռքով ուսումնասիրությունը կարող է դանդաղ և աշխատատար գործընթաց լինել:

- Ստուգման ճշգրտությունը կարող է ենթարկվել մարդկային սխալի: Գործոնները, ինչպիսիք են հոգնածությունը, փորձի մակարդակը և փաստաթղթի որակը, կարող են ազդել արդյունքի վրա:

Այս սահմանափակումները ընդգծում են ավտոմատացված համակարգերի անհրաժեշտությունը:

Ստուգման ավտոմատացված համակարգերն օգտագործում են առաջադեմ տեխնոլոգիաներ, ինչպիսիք են մեքենայական ուսուցումը և համակարգչային տեսլականը՝ կնիքները և ստորագրությունները վերլուծելու համար: Այս համակարգերն ունեն մի քանի առավելություններ ավանդական մեթոդների համեմատ: Այդ առավելություններն են.

- **Արագություն և արդյունավետություն**. ավտոմատացված համակարգերը կարող են մեծ քանակությամբ փաստաթղթեր մշակել զգալիորեն ավելի արագ, քան ձեռքով ստուգումը:

- **Օբյեկտիվություն**. Մեքենայական ուսուցման ալգորիթմները, որոնք պատրաստված են տվյալների հսկայական հավաքածուների վրա, կարող են վերլուծել փաստաթղթերը հետևողական և անկողմնակալ չափանիշներով:

- **Մասշտաբայնություն**. ավտոմատացված համակարգերը կարող են հեշտությամբ ստուգել փաստաթղթերը մեծ ծավալի դեպքում:

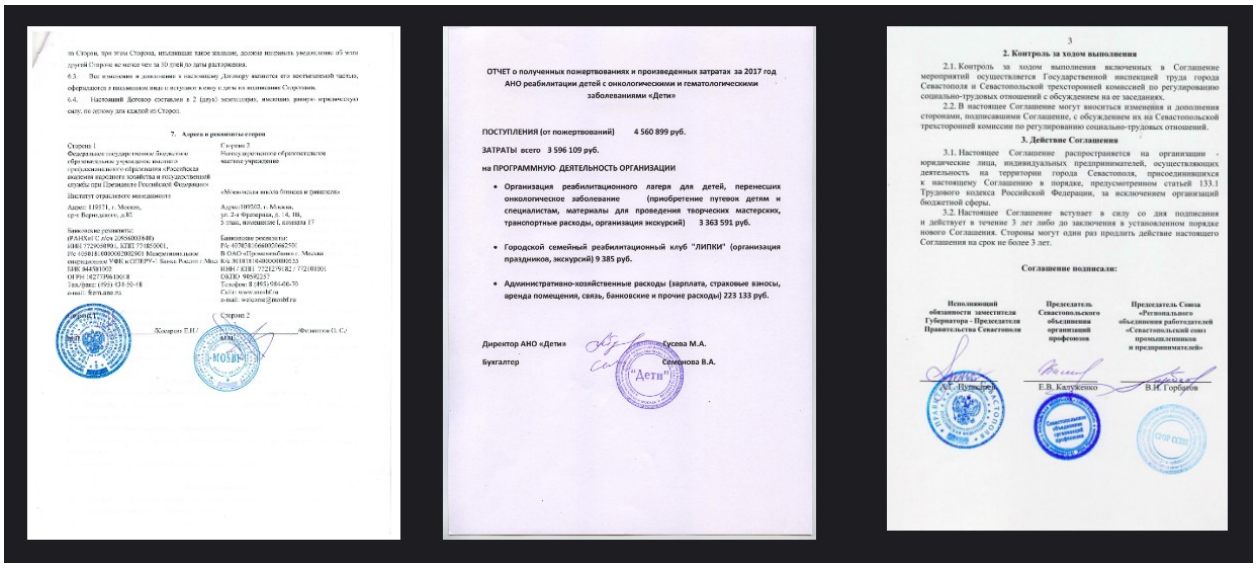
Այս աշխատանքում առաջարկվող համակարգը օգտագործում է ավտոմատացման այս առավելությունները:

ՏՎՅԱԼՆԵՐԻ ԲԱԶԱՅԻ ՁԵՌՔԲԵՐՈՒՄ ԵՎ ՏՎՅԱԼՆԵՐԻ ՎԵՐԼՈՒԾՈՒԹՅՈՒՆ

Այս աշխատանքն օգտագործում է մի քանի տվյալների բազաներ:

Առաջին տվյալների բազան (*Dataset*) փաստաթղթի պատկերի տվյալների հավաքածուն է: Այն պարունակում է բազմաբնույթ փաստաթղթերի պատկերներ: Այս տվյալների հավաքածուն ներառում է փաստաթղթերի մի շարք ձևաչափեր (պաշտոնական փաստաթղթեր, պայմանագրեր)՝ չափ, լուսավորություն, աղմուկ, ստորագրությունների և կնիքների ոչ ֆիքսված թվաքանակ: Այս ընտրությունը նպատակ ունի ապահովելու արդյունավետ կերպով փաստաթղթերի պատկերներից հայտնաբերել կնիքները և ստորագրությունները:

Նկար 2.1-ում պատկերված են այս դատասեթի պատկերների օրինակներ:



Նկար 2.1 Փաստաթղթի պատկերներ

Երկրորդ տվյալների բազան կնիքի և ստորագրության համընկնող տվյալների հավաքածու է: Այն ներառում է փաստաթղթի պատկերների հավաքածուի ենթաբազմություն, որը պարունակում է համընկնող կնիքներով և ստորագրություններով պատկերներ: Այս ենթաբազմությունը հնարավորություն է տալիս փորձարկել և գնահատել

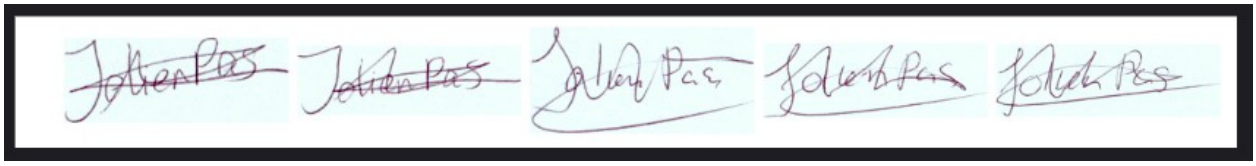
առանձին ալգորիթմը, որը նախատեսված է համընկնող (*overlap*) օբյեկտներն արդյունավետորեն առանձնացնելու համար:

Նկար 2.2-ում պատկերված են այս տվյալների բազայի պատկերների օրինակներ:

Նկար 2.2 համընկնող պատկերներ

Երրորդ տվյալների բազան ստորագրության ստուգման տվյալների հավաքածուն է: Ստորագրության ստուգման համար մեքենայական ուսուցման մոդելը վերապատրաստելու և գնահատելու համար կազմվել է հատուկ տվյալների բազա: Այս տվյալների բազան կազմված է առանձին ֆայլերից, որոնցից յուրաքանչյուրը ներկայացնում է որոշակի ստորագրողի: Ամեն ստորագրողի համար կան իրական և կեղծ ստորարգություններով ֆայլեր: Այս տվյալների հավաքածուն բազմազան է և կազմված է ընդհանուր 2000 պատկերներից: Նկար 2.3-ում պատկերված է մեկ ստորարգրողի կողմից իրական, իսկ 2.4-ում՝ կեղծ պատկերների օրինակներ:

Նկար 2.3 իրական ստորագրություն



Նկար 2.4 կեղծ ստորագրություն

Չորրորդ տվյալների բազան կնիքների ստուգման տվյալների հավաքածուն է: Այն կազմված է 2 ֆայլերից՝ իրական և կեղծ: Այս

տվյալների հավաքագրումը դժվար էր, այդ պատճառով կեղծ կնիքները ստեղծվել է ձեռքով՝ ավելացվել են անոմալիաներ։ Թեև այս մոտեցումը իդեալական չէ, այն թույլ է տալիս ուսուցանել մեքենայական ուսուցման մոդել՝ օգտագործելով անոմալիաների հայտնաբերման տեխնիկան։ Կնիքների ստուգման տվյալների հավաքածուն իր մեջ ներառում է 120 կեղծ և 200 իրական պատկերներ։ Նկար 2.5-ում պատկերված է մեկ իրական կնիքի օրինակ, իսկ 2.6-ում՝ կեղծված։

Նկար 2.5 իրական կնիք

Նկար 2.6 կեղծ կնիք

ՓԱՍՏԱԹՂԹԵՐԻ ՊԱՏԿԵՐՆԵՐՈՒՄ ՍՏՈՐԱԳՐՈՒԹՅՈՒՆՆԵՐԻ և ԿՆԻՔՆԵՐԻ ՀԱՅՏՆԱԲԵՐՈՒՄ

Ծրագրի այս հատվածում մշակվում է մեթոդ, որն ապահովում է փաստաթղթերի պատկերների մեջ ստորագրությունների և կնիքների ավտոմատ հայտնաբերում: Այս ալգորիթմն օգտագործում է երկքայլ մոտեցում՝ կենտրոնանալով պատկերի ալիքների ինտենսիվության մանիպուլյացիայի և ֆոնի մեկուսացման վրա:

Ալգորիթմի նկարագրություն.

- **Ալիքի բաժանում**: Թվային պատկերների մեծ մասը օգտագործում է գունային ձևաչափ, ինչպիսին է BGR (կապույտ, կանաչ, կարմիր)՝ գունային տեղեկատվությունը ներկայացնելու համար: Պատկերի յուրաքանչյուր պիքսել ունի երեք արժեք, որոնք համապատասխանում են այս երեք գունավոր ալիքների ինտենսիվությանը: *Split* ֆունկցիան վերցնում է BGR պատկերը որպես մուտք և այն բաժանում է իր երեք առանձին ալիքների՝ կապույտ (b), կանաչ (g) և կարմիր (r): Այդ ալիքներն այնուհետև պահվում են առանձին փոփոխականներում՝ հետագա մշակման համար:

```
b, g, r = cv2.split(image)
```

- **Ինտենսիվության մասշտաբավորում և կտրում.** Այս քայլում ալիքների յուրաքանչյուր պիքսելի արժեքը բազմապատկում է նախապես սահմանված արժեքով (այս դեպքում factor = 0.5): Այս քայլը նվազեցնում է պատկերի ներսում ալիքների ինտենսիվությունը: Սա ձեռք է բերվում ալիքի արժեքները 1-ից պակաս գործակցով բազմապատկելով: RGB գունային տարածության մեջ կապույտ օբյեկտները ունեն բարձր կապույտ ալիքի արժեք (սովորաբար մոտ 255): Նվազեցնելով կապույտ ալիքի ինտենսիվությունը, կապույտ ֆոնի կապույտ արժեքը նվազում է: Սա ստիպում է այն ավելի մուգ կամ ավելի քիչ հագեցած տեսք ունենալ՝ համեմատած առաջին պլանի տարրերի հետ: Այս տարրերը (տեքստ, ստորագրություններ, կնիքներ) սովորաբար ունեն ավելի ցածր կապույտ ալիքի արժեքներ սկզբում: Արդյունքում նրանք պահպանում են իրենց հարաբերական ինտենսիվությունը և տեսողականորեն ավելի տարբերվում են թուլացած կապույտ ֆոնից: Ինտենսիվության

գործակիցը սահմանել ավելի ցածր (օրինակ 0.1), կարող է հանգեցնել կապույտ բաղադրիչների մանրամասների զգալի կորստի, ինչը կարող է ազդել ստորագրության/կնիքների հայտնաբերման ճշգրտության վրա: Մյուս կողմից, ավելի բարձր արժեքը (օրինակ 0.8) կարող է չափազանց շատ ինտենսիվություն պահպանել, ինչը դժվարացնում է ստորագրությունները/կնիքները ֆոնից մեկուսացնելը: *np.clip()* ֆունկցիան օգտագործվում է ստացված արժեքները 0-ից 255-ի սահմաններում սահմանափակելու համար: Այն կանխում է պիքսելների արժեքների գերհոսքը կամ ներհոսքը՝ ապահովելով, որ դրանք վավեր են պատկերի ներկայացման համար: Իսկ հետո փոխակերպվում է տվյալների տեսակը , որը պատկերի պիքսելների արժեքները պահելու ստանդարտ ձևաչափն է:

```
b = np.clip(b * factor, a_min: 0, a_max: 255).astype(np.uint8)
g = np.clip(g * factor, a_min: 0, a_max: 255).astype(np.uint8)
r = np.clip(r * factor, a_min: 0, a_max: 255).astype(np.uint8)
```

- **Ալիքների միավորում.** գունային ալիքների վրա գործողություններ կատարելուց հետո, կարևոր է վերամիավորել այդ ալիքները մեկ պատկերի մեջ: merge([b, g, r]) ֆունկցիան կատարում է այդ գործողությունը:

```
image = cv2.merge([b, g, r])
```

- **Գույնի տարածության փոխարկում.** Փոխակերպում է գունային պատկերը BGR (Blue-Green-Red) գունային տարածությունից դեպի HSV (Hue-Saturation-Value) գունային տարածության: HSV-ի ներկայացումը ձեռնտու է գույնի մեկուսացման համար՝ հիմնվելով դրանց երանգի վրա:

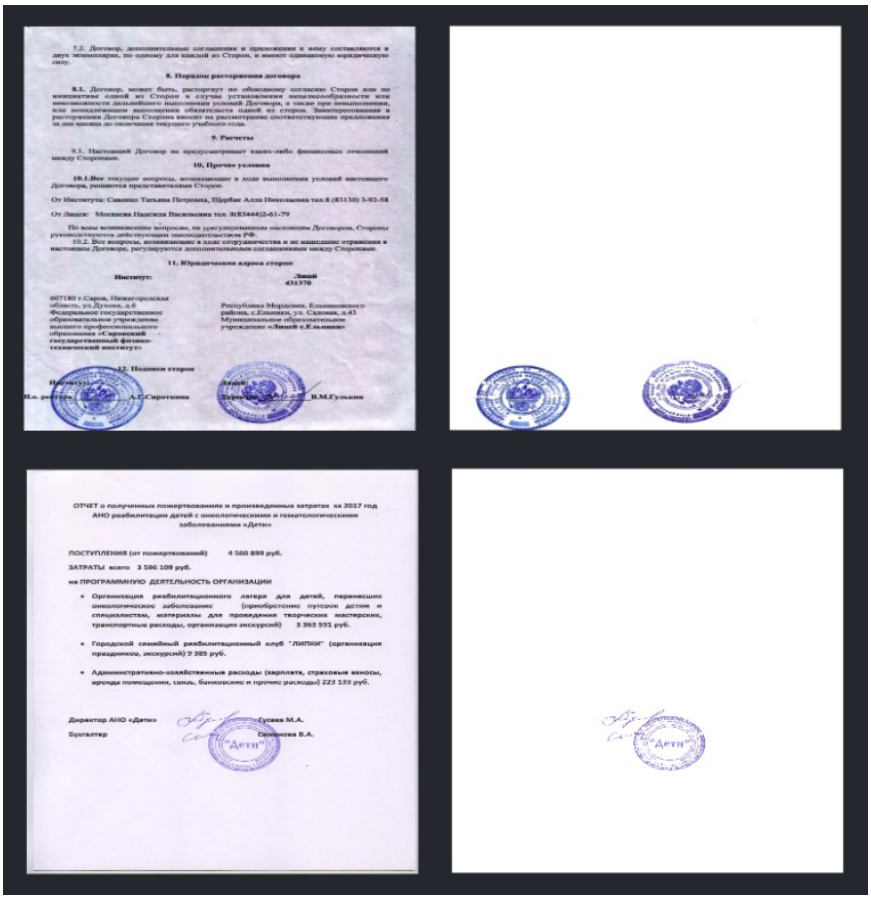
```
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
```

- **Ֆոնի մեկուսացման.** Սահմանելով կապույտ գույնին համապատասխանող երանգների տիրույթ (lower_blue_hue = 100 , upper_blue_hue = 140), մենք կարող ենք մեկուսացնել պատկերի այն շրջանները, որոնք պարունակում են կապույտ երանգներ: ստեղծվում է երկուական դիմակ (*blue_mask*), որը մեկուսացնում է կապույտ երանգներով պիքսելները նշված HSV գունային տիրույթում: Պիքսելները, որոնք

համապատասխանում են կապույտ երանգի չափանիշներին, սահմանվում են մեկ գույնի (սպիտակ), իսկ մյուսները՝ մեկ այլ գույնի (սև)։ Որից հետո ապահովում ենք, որ դիմակը կարող է կիրառվել ամբողջ պատկերով։ հաջորդ քայլում նոր պատկեր է ստեղծվում նույն չափերով, ինչ բնօրինակ պատկերը՝ ամբողջությամբ լցված սպիտակ պիքսելներով։ Այն կծառայի որպես ֆոն, որի վրա կտեղադրվեն առանձնացված կապույտ շրջանները։ Կապույտ դիմակն օգտագործելով՝ սկզբնական պատկերից պիքսելները (որտեղ դիմակը ցույց է տալիս կապույտ շրջանները) պահպանվում են, մինչդեռ սպիտակ ֆոնի պատկերից պիքսելները (որտեղ դիմակը չի նշում կապույտ շրջանները) օգտագործվում են որպես ֆոն։

```
blue_mask = cv2.inRange(hsv_image, lowerb: (lower_blue_hue, 50, 50), upperb: (upper_blue_hue, 255, 255))
blue_mask = cv2.merge([blue_mask, blue_mask, blue_mask])
white_image = np.ones_like(image) * 255
result_image = np.where(blue_mask > 0, image_copy, white_image)
```

Նկար 3.1-ում ներկայացված են այս ալգորիթմի արդյունքների օրինակներ։



Նկար 3.1 Կնիքների և ստորագրությունների առանձնացում

ՀԱՄԸՆԿՆՈՂ ՍՏՈՐԱԳՐՈՒԹՅՈՒՆՆԵՐԻՒ ԿՆԻՔՆԵՐԻ ԱՌԱՆՁՆԱՑՈՒՄ

Այս ալգորիթմը նպատակ ունի առանձնացնել overlap (համընկնող) օբյեկտները (ստորագրություններ և կնիքներ)՝ օգտագործելով գույնի վրա հիմնված հատվածավորման տեխնիկա: Ահա, թե ինչպես է աշխատում ալգորիթմը.

Ալգորիթմի նկարագրություն.

- **Գույնի ձևաչափի փոխարկում.** Պատկերը որն ունի BGR ձևաչափ փոխակերպվում է HSV (*Hue-Saturation-Value*) գունային տարածության: Այս փոխակերպումը բաժանում է գունային տեղեկատվությունը երանգի(*Hue*), հագեցվածության(*Saturation*) և արժեքի(*value*) բաղադրիչների, որոնք ավելի հարմար են գունային վերլուծության համար: Ալգորիթմը հաշվարկում է երանգի միջին արժեքը (*hue_mean*) ամբողջ պատկերի վրա: Այս միջին երանգը ծառայում է որպես պատկերում առկա գերակշռող գունային երանգի ցուցիչ:

```
hsv_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2HSV)
hue_channel = hsv_image[:, :, 0]
```

- **Ալիքի շեմի որոշում.** Հաշվարկված hue_mean-ի հիման վրա ալգորիթմը հատկացնում է որոշակի շեմային արժեք (channel)՝ տարբեր գունային երանգները տարբերելու համար: Այս շեմերը օգտագործվում են երանգի ինտենսիվության հիման վրա ստորագրությունը կնիքից առանձնացնելու համար: Այս արժեքները որոշվել են մի շարք վերլուծություններից հետո:

```
if channel_value == 0:
    hist = np.histogram(hue_channel, bins=180, range=[0, 180])[0][1:]
    hist = hist[hist != 0]
    hue_mean = np.mean(hist)
    channel_values = {
        (0, 48): 115,
        (48, 73): 108,
        (73, 80): 100,
        (80, 140): 105,
        (140, 150): 115,
        (150, 180): 117,
        (180, 400): 120,
        (400, 10000): 125,
    }

    for (min_value, max_value), channel in channel_values.items():
        if min_value <= hue_mean < max_value:
            channel_value = channel
            break
    else:
        channel_value = 120
```

- **Երկուական դիմակի ստեղծում.** կոդը ստեղծում է երկուական դիմակ՝ համեմատելով պիքսելային արժեքները որոշված շեմի հետ: Այս շեմը գերազանցող պիքսելները նշվում են որպես առաջին պլան (1 կամ True), մինչդեռ մյուսները համարվում են ֆոնային (0 կամ False):

```
mask = hsv_image[:, :, 0] > channel_value
```

- **Միացված բաղադրիչների պիտակավորում.** Կապակցված առաջին պլանի պիքսելների յուրաքանչյուր առանձին խմբի հատկացվում է յուրահատուկ պիտակ:

```
blobs_labels = measure.label(mask, background=0)
if blobs_labels.max() == 0:
    return None, None
```

- **Խոշոր բաղադրիչի հայտնաբերում.** Ալգորիթմը վերլուծում է հայտնաբերված բաղադրիչները՝ օգտագործելով measure.regionprops: Ալգորիթմը անցնում է պիտակավորված շրջանների միջոցով (*blobs_labels*)՝ բացահայտելու ամենամեծ միացված բաղադրիչը (*largest_component*)՝ ելնելով իր տարածքից (*regions*): Ենթադրվում է, որ այս բաղադրիչը կամ ստորագրությունն է կամ կնիքը:

```
regions = measure.regionprops(blobs_labels)
largest_component = max(regions, key=lambda prop: prop.area)
biggest_component_coords = largest_component.coords
```

- **Հայտնաբերված բաղադրիչի դիմակի ստեղծում.** Ստեղծվում է նոր երկուական դիմակ(*component_mask*), որտեղ միայն ամենամեծ բաղադրիչին պատկանող պիքսելներն են սահմանվում են 255 (սպիտակ): Արդյունքում ստեղծվում է երկու վերջնական պատկեր (*sign_final_image* և *stamp_final_image*):

```
component_mask = np.zeros_like(mask, dtype=np.uint8)
component_mask[biggest_component_coords[:, 0], biggest_component_coords[:, 1]] = 255
sign_final_image = np.where(component_mask[:, :, None] == 255, original_image,
                            np.ones_like(original_image) * 255)
stamp_final_image = np.where(255 - component_mask[:, :, None] == 255, original_image,
                             np.ones_like(original_image) * 255)

return sign_final_image, stamp_final_image
```

Նկար 4.1-ում ներկայացված են այս ալգորիթմի արդյունքների օրինակներ:

Նկար 4.1 Overlap պատկերների առանձնացում

ՏՎՅԱԼՆԵՐԻ ՆԱԽՆԱԿԱՆ ՄՇԱԿՈՒՄ

Նախքան մեքենայական ուսուցման մոդելներին անցնելը կատարվում է պատկերների նախնական մշակում (pre-processing)՝ մոդելի ուսուցումնը հեշտացնելու համար: Այն կատարվում է թ՛ե սորտագրության և թ՛ե կնիքի վավերացման մոդել կառուցելուց առաջ: Ստորև ներկայացված է յուրաքանչյուր քայլը և իր հիմքում ընկած հիմնավորումը.

- **Պատկերի չափի կարգավորում.** Պատկերները կարող են շատ տարբեր լինել չափերով: Բոլոր պատկերների որոշակի չափի ստանդարտացումը ապահովում է հետևողականություն և հեշտացնում է մոդելի մշակումը: Այն նաև նվազեցնում է հաշվողական ծախսերը, քանի որ մոդելը կարիք չունի մշակելու տարբեր չափերի պատկերներ:
- **Գորշ գույնի փոխարկում**. պատկերները մոխրագույնի վերածելը վերացնում է գունային տատանումները, որոնք կարող են չհամապատասխանել ստորագրության ստուգմանը: Սա պարզեցնում է տվյալները՝ պահպանելով հիմնական հատկանիշները, ինչպիսիք են հարվածի լայնությունը և ձևը: Մոխրագույնի փոխակերպումը պարզեցնում է տվյալները՝ կենտրոնացնելով մոդելը պատկերի ձևերի և նախշերի վրա:
- **Պիտակների(Label) ստեղծում.** Պիտակներն ապահովում են տեղեկատվություն, այս դեպքում՝ պարզելով պատկերը ներկայացնում է իրական, թե կեղծ ստորագրություն (0 իրական, 1՝ կեղծված): Մոդելը սովորում է պատկերի առանձնահատկությունների և համապատասխան պիտակների միջև կապը՝ չտեսնված տվյալների վերաբերյալ կանխատեսումներ անելու համար:
- **Տվյալների միացում և վերաձևավորում.** Մշակված պատկերների և դրանց պիտակների միացումն առանձին զանգվածների մեջ ստեղծում է ձևաչափ (սովորաբար եռաչափ թենզոր(3D Tensor)՝ տողերի, սյունակների և ալիքների չափսերով), որը պահանջվում է մեքենայական ուսուցման մոդելների մեծ մասի կողմից:

- **Տվյալների բաժանում.** Տվյալների բաժանումը վերապատրաստման (*training*) և թեստավորման (*testing*) հավաքածուների էական նշանակություն ունի՝ գերհարմարեցումը կանխելու համար: Գերհարմարեցումը տեղի է ունենում, երբ մոդելը չափազանց լավ է անգիր անում տվյալները և չի կարողանում ճիշտ գուշակություններ անել չտեսնված տվյալներին: Վերապատրաստման (*training*) հավաքածուն օգտագործվում է մոդելները վարժեցնելու համար, իսկ փորձարկման (*testing*) հավաքածուն՝ գնահատելու դրանց կատարումը չտեսնված տվյալների վրա: Ընդհանուր բաժանման հարաբերակցությունը կազմում է 80% վերապատրաստման և 20% թեստավորման համար:

Ստորև ներկայացված է ստորագրությունների պատկերների նախնական մշակման համար կոդը: Կնիքների դեպքում կատարվում է նույն կերպ:

```
SIZE = 224
train_dir = "/home/marine/PycharmProjects/pythonProject/signature_data/data"
real_images, forged_images = [], []

for per in os.listdir(train_dir):
    for data in glob.glob(os.path.join(train_dir, per, '*.*')):
        img = Image.open(data).convert('L')  # Convert to grayscale
        img = np.array(img)
        img = cv2.resize(img, dsize: (SIZE, SIZE))
        if per[-1] == 'g':
            forged_images.append(img)
        else:
            real_images.append(img)

real_images = np.array(real_images)
forged_images = np.array(forged_images)
real_labels = np.zeros((real_images.shape[0], 1))
forged_labels = np.ones((forged_images.shape[0], 1))
images = np.concatenate((real_images, forged_images))
labels = np.concatenate((real_labels, forged_labels))
images = images.reshape(images.shape[0], -1)

train_data, test_data, train_labels, test_labels = (train_test_split(
                        *arrays: images, labels, test_size=0.2, random_state=42))

scaler = StandardScaler()
train_data = scaler.fit_transform(train_data)
test_data = scaler.transform(test_data)
```

ՍՏՈՐԱԳՐՈՒԹՅԱՆ ՎԱՎԵՐԱՑՄԱՆ ՄՈԴԵԼԻ ԸՆՏՐՈՒԹՅՈՒՆ

Ծրագրի այս հատվածի հիմնական նպատակն է մշակել և գնահատել մեքենայական ուսուցման մոդելներ՝ ստորագրության վավերացման համար: Սա ներառում է դասակարգիչների ուսուցում՝ իրական և կեղծ ստորագրությունները տարբերելու համար: Տվյալների հավաքածուն ձեռագիր ստորագրությունների հավաքածու է: Այն պարունակում է ինչպես իրական, այնպես էլ կեղծ ստորագրություններ:

Մանրամասն խոսենք մեքենայական ուսուցման մոդելի կառուցման մասին:

Եթե պատկերների նախնական մշակման փուլից անմիջապես հետո որևէ մեքենայական ուսուցման մոդել կիրառենք, այսինքն տարբերակումը կատարենք բացառապես ինտենսիվության տատանումների հիման վրա, ցուցանիշը 60%-ից բարձր չի լինի: Այս դեպքում մենք գործ ունենք ձեռագիր օբյեկների (ստորագրությունների) հետ, որպեսզի ստեղծենք արդյունավետ մոդել պետք է առանձնահատկություններ և օրինաչափություններ սահմանել, ըստ որի կկատարվի տարբերակումը:

Այս օրինաչափությունները ներառում են այնպիսի բաներ, ինչպիսիք են.

- Գրիչի ճնշման տատանումները
- Թանաքի լայնությունը և հետևողականությունը
- Ելակետային հետևողականություն (ինչպես է գրությունը նստած տողի վրա)
- Ստորագրության ընդհանուր հոսքը և ռիթմը

Նույնիսկ չտեսնված ստորագրությունների դեպքում մոդելը պետք է կարողանա վերլուծել այս հատկանիշները և համեմատել դրանք ուսումնական տվյալների կեղծիքներից սովորած օրինաչափությունների հետ: Եթե ստորագրությունը ցուցադրում է կեղծիքների մեջ հայտնաբերված հատկանիշներ (օրինակ՝ տարուբերվող, անհաստատ, շարժական, անկայուն գծեր, ճնշման անբնական փոփոխություն), մոդելը կարող է այն նշել որպես կասկածելի:

Այս արդյունքը ապահովում է կողմնորոշված գրադիենտի հիստոգրամը (HOG): Այս տեխնիկան պատկերների մեջ ֆիքսում է եզրերի հիմնական կողմնորոշումները և տեղական գրադիենտները՝ արժեքավոր պատկերացումներ տալով հարվածների ձևերի վերաբերյալ: HOG-ը կենտրոնանում է պատկերի ձևի և հյուսվածքի մասին տեղեկատվության գրավման վրա: Ահա թե ինչպես է HOG-ը աշխատում.

- **Պատկերի բաժանում.** Պատկերի բաժանում փոքր ուղղանկյունների, որոնք կոչվում են բջիջներ: Այս բջիջները նման են փոքր ցանցերի, որոնք դրված են ստորագրության պատկերի վրա:
- **Գրադիենտի հաշվարկ.** յուրաքանչյուր բջջի ներսում HOG-ը հաշվարկում է պատկերի գրադիենտը յուրաքանչյուր պիքսելում: Գրադիենտը հիմնականում ներկայացնում է պիքսելների արժեքների փոփոխության ուղղությունը և ինտենսիվությունը: Պատկերացրեք, որ փոքրիկ կուրսորը շարժվում է պատկերի միջով. գրադիենտը ցույց է տալիս, թե որքան արագ է փոխվում պայծառությունը, երբ դուք շարժում եք կուրսորը տարբեր ուղղություններով (վերև, վար, ձախ, աջ և այլն):
- **Գրադիենտների հիստոգրամ.** Յուրաքանչյուր բջջի համար HOG-ը ստեղծում է հիստոգրամ, որը իրենից ներկայացնում է այս գրադիենտների բաշխումը տարբեր ուղղությունների վրա: Այսինքն հիստոգրամը ցույց է տալիս, թե որքան հաճախ են որոշակի ուղղությունների գրադիենտներ (օրինակ՝ ուղղահայաց, հորիզոնական, անկյունագծային) հայտնվում այդ բջիջում:

Ըստ էության, HOG-ը պատկերը պիքսելների հավաքածուից վերածում է հիստոգրամների հավաքածուի, որոնք ներկայացնում են գրադիենտների տեղական բաշխումը: Այս հիստոգրամները դառնում են այն հատկանիշները, որոնք կարող են օգտագործել մեքենայական ուսուցման մոդելները՝ օրինաչափություններ սովորելու և իրական և կեղծ ստորագրությունները տարբերելու համար:

HOG-ը հատկապես զգայուն է պատկերի եզրերի նկատմամբ: Գրիչի հարվածների պատճառով ստորագրությունները լի են եզրերով, Վերլուծելով գրադիենտի ուղղությունները բջջի ներսում՝ այն կարող է որոշել եզրերի առկայությունն ու ուղղությունը այդ պատկերի այդ տարածքում: Բջջի գրադիենտների բաշխումը նաև որոշ տեղեկություններ է բացահայտում ստորագրության տեղական ձևի մասին: Օրինակ, ուղղահայաց գրադիենտների բարձր կոնցենտրացիան կարող է ցույց տալ ուղիղ գծի հատված, մինչդեռ ավելի ցրված բաշխումը կոր գիծ:

HOG-ը հզոր գործիք է պատկերներից տեղեկատվական առանձնահատկություններ հանելու համար, մասնավորապես, նրանք, որոնք պարունակում են եզրեր և հյուսվածքներ, ինչպիսիք են ձեռագիր ստորագրությունները: Այնուամենայնիվ, դա հաճախ ստորագրությունների ստուգման համակարգերում փազլի միայն մի մասն է:

Ծրագրի այդ հատվածում կատարվում են հետևյալ գործողությունները.

- compute_hog_features ֆունկցիայի սահմանում. այն հաշվարկում է նկարի առանձնահատկությունները՝ օգտագործելով hog ֆունկցիան, այդ ֆունկցիայի պարամետրերն են.
    1. *img* - մուտքագրվող պատկեր
    2. *orientations* - (*by default* 8) Նկարի գրադիենտների ուղղությունների քանակը
    3. *pixels_per_cell* - (*by default* 16x16) Բաժանված բջիջների չափ
    4. *cells_per_block* - (*by default* (1, 1)) Բջիջների թիվը, որոնք պետք է խմբավորվեն մեկ բլոկի մեջ
    5. *visualize* - (*by default* True) Եթե True է, այն նաև վերադարձնում է պատկեր, որը արտացոլում է HOG-ի առանձնահատկությունները (օգտակար է հասկանալու համար):
- HOG-ի առանձնահատկությունների հաշվում իրական և կեղծված պատկերների համար առանձին: Միավորել առանձնահատկությունները մեկ զանգվածի մեջ և ձևաչափի փոփոխում մեքենայական ուսուցման օգտագործման համար:

- HOG-ի առանձնահատկությունների և պիտակների բաժանում վերապատրաստման և թեստավորման խմբերի: Ստանդարտացնել վերապատրաստման և թեստավորման տվյալները:

Ստորև ներկայացված է ծրագրի այդ հատվածը:

```
def compute_hog_features(img):
    features, _ = hog(img, orientations=8, pixels_per_cell=(16, 16), cells_per_block=(1, 1), visualize=True)
    return features

hog_real_images = np.array([compute_hog_features(img) for img in real_images])
hog_forged_images = np.array([compute_hog_features(img) for img in forged_images])

hog_features = np.concatenate((hog_real_images, hog_forged_images))
hog_features = hog_features.reshape(hog_features.shape[0], -1)

hog_train_data, hog_test_data, hog_train_labels, hog_test_labels = train_test_split(
    *arrays: hog_features, labels, test_size=0.2, random_state=42
)

hog_scaler = StandardScaler()
hog_train_data = hog_scaler.fit_transform(hog_train_data)
hog_test_data = hog_scaler.transform(hog_test_data)
```

Grid Search-ը հիպերպարամետրային թյունինգի հզոր տեխնիկա է: Այն համակարգված կերպով գնահատում է հիպերպարամետրերի արժեքների տարբեր համակցություններ և ընտրում է այն համակցությունը, որը տալիս է վավերացման հավաքածուի լավագույն կատարումը: Սա ապահովում է, որ մոդելը օպտիմիզացված է կոնկրետ տվյալների և առաջադրանքի համար:

Մենք փորձարկել ենք մեքենայական ուսուցման դասակարգիչները՝ օգտագործելով Grid Search: Քննարկենք դասակարգիչների արդյունքները.

***Support Vector Machine with Sigmoid Kernel (SVM-Sigmoid)**:*

SVM դասակարգիչը սիգմոիդ միջուկով հզոր գործիք է երկուական դասակարգման առաջադրանքների համար: Սիգմոիդ միջուկը հատկապես օգտակար է, երբ տվյալները գծային բաժանելիություն չեն ցուցաբերում: Այսինքն իդեալում, մենք կարող եք ուղիղ գիծ գծել (*գծային բաժանում*) դրանք կատարելապես բաժանելու համար: Այնուամենայնիվ, երբեմն տվյալները ավելի ցրված են և չեն կարող հստակորեն բաժանվել

ուղիղ գծով: Սա կոչվում է *ոչ գծային բաժանելիություն*: Սիգմոիդ միջուկը հատուկ հնարք է, որն օգտագործվում է մեքենայական ուսուցման մեջ՝ այս իրավիճակը կարգավորելու համար: Այն հիմնականում փոխակերպում է տվյալները ավելի մեծ չափերի տարածության, որտեղ դրանք կարող են դառնալ գծային բաժանելի: Ավելի պարզ ասած, այն ստեղծում է տվյալների դիտարկման ավելի բարդ ձև, որը կարող է բացահայտել երկու խմբերի միջև ավելի հստակ տարանջատում: SVM դասակարգիչը սիգմոիդ միջուկով և HOG հատկանիշների հիման վրա կարողացել է ապահովել ***78% ճշգրտություն***:

```
hog_svm_classifier = svm.SVC(kernel='sigmoid')
hog_svm_classifier.fit(hog_train_data, hog_train_labels.ravel())
hog_y_pred = hog_svm_classifier.predict(hog_test_data)
```

Support Vector Machine with Polynomial Kernel (SVM-Poly, Tuned):

SVM դասակարգիչը բազմանդամ միջուկով SVM-ների մեկ այլ տարբերակ է, որը կարող է կարգավորել ոչ գծային որոշման սահմանները: Հիպերպարամետրերի թյունինգը, ինչպիսիք են բազմանդամի աստիճանը և կանոնավորացման պարամետրը (regularization parameter - C), կարևոր է դրա կատարողականությունը օպտիմալացնելու համար: Կատարում ենք նաև ցանցային որոնում (*grid search*)՝ գտնելու SVM բազմանդամների լավագույն հիպերպարամետրերը: SVM-Poly-ն օգտագործում է բազմանդամ միջուկի ֆունկցիա՝ մուտքագրված տվյալները վերափոխելու ավելի մեծ չափերի հատկանիշի տարածության, որտեղ գծային որոշման սահմանը կարող է արդյունավետորեն բաժանել դասերը: Բազմանդամի միջուկի աստիճանը որոշում է որոշման սահմանի բարդությունը: Ավելի բարձր աստիճանները թույլ են տալիս տվյալների մեջ ներառել բարդ օրինաչափություններ:

SVM-Poly-ն նպատակ ունի գտնել հիպերպլան, որը առավելագույնի է հասցնում դասերի միջև սահմանը, ինչը հանգեցնում է ընդհանրացման ավելի լավ կատարողականի՝ համեմատած դասակարգիչների հետ, որոնք կենտրոնացած են միայն դասակարգման սխալները նվազագույնի հասցնելու վրա: Բազմանդամի միջուկի հաշվողական բարդությունը մեծանում է բազմանդամի աստիճանի հետ, ինչը հանգեցնում է ուսուցման ավելի երկար ժամանակի: Այն ցուցաբերեց ***88% ճշգրտություն***:

```
svm_classifier = svm.SVC(kernel='poly')
param_grid = {'C': [0.1, 1, 10], 'degree': [2, 3, 4], 'coef0': [0.0, 0.1, 1.0]}
grid_search = GridSearchCV(svm_classifier, param_grid, cv=3)
grid_search.fit(hog_train_data, hog_train_labels.ravel())
best_params = grid_search.best_params_

best_svm_classifier = svm.SVC(kernel='poly', C=best_params['C'], degree=best_params['degree'],
                              coef0=best_params['coef0'])
best_svm_classifier.fit(hog_train_data, hog_train_labels.ravel())
hog_y_pred_poly = best_svm_classifier.predict(hog_test_data)
```

Random Forest:

Random Forest-ը անսամբլային ուսուցման մեթոդ է: Անսամբլային ուսուցումը մեքենայական ուսուցման մի քանի մոդելների համադրություն է մեկ խնդրի մեջ: Այս մոդելները հայտնի են որպես թույլ սովորողներ:

Երբ մի քանի թույլ սովորողների միավորում ենք, նրանք կարող են դառնալ ուժեղ սովորողներ: Սա կարող է շահավետ լինել, երբ գործ ունենք մեծ չափերի տվյալների հետ՝ բազմաթիվ անհամապատասխան հատկանիշներով: Random Forest-ը կառուցում է բազմաթիվ որոշումների ծառեր՝ օգտագործելով տվյալների և առանձնահատկությունների պատահական ենթաբազմություններ և միավորում է դրանց կանխատեսումները քվեարկության մեխանիզմի միջոցով՝ դասակարգման կամ ռեգրեսիայի առաջադրանքների ճշգրիտ կանխատեսումներ կատարելու համար:

Թեև Random Forests-ը ճշգրիտ կանխատեսումներ է տալիս, ծառերի համախումբը դժվարացնում է մոդելի որոշումների կայացման գործընթացը մեկնաբանելը: Հարմար չէ շատ անհավասարակշռված տվյալների համար. ***Ճշգրտությունը կազմում է 83% :***

```
from sklearn.ensemble import RandomForestClassifier

rf_classifier_hog = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier_hog.fit(hog_train_data, hog_train_labels.ravel())
rf_y_pred_hog = rf_classifier_hog.predict(hog_test_data)
```

Voting Classifier (KNN + SVM-Poly): Կառուցում ենք Voting Classifier (քվեարկության դասակարգիչ)՝ միավորելով K-մոտակա հարևանները

(KNN) և օպտիմիզացված SVM դասակարգիչը բազմանդամ միջուկով: Օգտագործելով տարբեր դասակարգիչների ուժեղ կողմերը, այն հաճախ ավելի լավ կատարողականություն է ձեռք բերում, քան ցանկացած առանձին դասակարգիչ: Այս մոդելը հասնում է ամենաբարձր **ճշգրտությանը՝ 91%**:

Երկուական դասակարգման դեպքում քվեարկության դասակարգիչը կարող է կիրառել փափուկ (Soft Voting) կամ կոշտ (Hard Voting) քվեարկության ռազմավարություն:

Hard Voting: Կոշտ քվեարկություն. վերջնական կանխատեսումը հիմնված է առանձին դասակարգիչների ձայների մեծամասնության վրա:

Soft Voting: Փափուկ քվեարկություն. Յուրաքանչյուր դասակարգիչ տրամադրում է հավանականության գնահատում յուրաքանչյուր դասի համար, և վերջնական կանխատեսումը հիմնված է բոլոր դասակարգիչների միջին հավանականության վրա:

Համատեղելով տարբեր ալգորիթմներ, ինչպիսիք են KNN-ը և SVM-Poly-ն, քվեարկության դասակարգիչը օգտագործում է յուրաքանչյուր մոդելի ուժեղ կողմերը՝ միաժամանակ մեղմելով նրանց թույլ կողմերը:

Բազմաթիվ մոդելներ համադրելով՝ Քվեարկության դասակարգիչը նվազեցնում է չափից ավելի հարմարվելու վտանգը և հակված է ավելի լավ ընդհանրացնել չտեսնված տվյալներին:

K-Nearest Neighbors (KNN) հայտնի է տեղական օրինաչափությունները տարբերելու իր ունակությամբ: Ավելին, KNN-ը կայուն է աղմուկի դեմ և ցույց է տալիս ճկունություն ոչ գծային ստորագրության տատանումների նկատմամբ: Այս որակը նրան հատկապես հմուտ է դարձնում տարբեր և բարդ ստորագրությունների տվյալների շտեմարանների մշակման գործում: Բազմանդամ միջուկով օժանդակ վեկտորային մեքենան (SVM-Poly) գործում է խոշորացույցով քննիչի նման: Այն խորանում է ավելի լայն համատեքստում՝ բացահայտելով բարդ հարաբերությունները տվյալների բազայում: SVM-Poly-ի ճկունությունը, որին նպաստում է բազմանդամ միջուկը, թույլ է տալիս նրան հարմարվել տարբեր սցենարների, ներառյալ այն սցենարներին, որոնք ներառում են ստորագրության առանձնահատկություններ:

KNN-ի և SVM-Poly-ի ինտեգրումը քվեարկության դասակարգիչում միավորում է նրանց համապատասխան ուժեղ կողմերը՝ ստեղծելով ստորագրության վավերացման համապարփակ և սիներգետիկ շրջանակ: Այս մոտեցումն առաջարկում է ամուր, հուսալի և հեշտությամբ մեկնաբանվող լուծում՝ ստորագրության նույնականացման համար:

```
knn_classifier = KNeighborsClassifier(n_neighbors=3)
poly_svm_classifier = svm.SVC(kernel='poly', C=10, degree=2, coef0=0.1, probability=True)

voting_classifier = VotingClassifier(
    estimators=[
        ('knn', knn_classifier),
        ('poly_svm', poly_svm_classifier),
    ],
    voting='soft'
)

voting_classifier.fit(hog_train_data_scaled, hog_train_labels.ravel())
hog_y_pred_voting = voting_classifier.predict(hog_scaler.transform(hog_test_data))
```

Կնիքի վավերացման մոդելի ընտրություն

Ծրագրի այս հատվածի նպատակն է մշակել և գնահատել մեքենայական ուսուցման մոդելներ՝ կնիքների վավերացման համար: Սա ներառում է դասակարգիչների ուսուցում՝ իրական և կեղծ կնիքները տարբերելու համար: Այս վերլուծության նպատակն է բացահայտել ստորագրության վավերացման ամենաարդյունավետ տեխնիկան՝ հիմնված տվյալների վրա:

Այս փուլում ևս փորձարկել ենք մեքենայական ուսուցման դասակարգիչները՝ օգտագործելով Grid Search:

Քննարկենք դասակարգիչների արդյունքները.

K-Nearest Neighbors (KNN) Այն հիմնվում է հյուսվածքի նկարագրիչների և ձևի բնութագրիչների վրա: Այն հարմար է դրոշմակնիքների վավերացման համար, քանի որ այն գրավում է տեղական նախշերը տարածության մեջ՝ օգնելով դասակարգել նամականիշները՝ հիմնված նմանատիպ հատկանիշների վրա: ***Ճշգրտությունը կազմում է 72%:***

```
param_grid_knn = {
    'n_neighbors': [3, 5, 7],
    'weights': ['uniform', 'distance'],
    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']
}
knn = KNeighborsClassifier()
grid_search_knn = GridSearchCV(knn, param_grid_knn, cv=5)
grid_search_knn.fit(train_data, train_labels.ravel())
best_knn_model = grid_search_knn.best_estimator_
predictions_knn = best_knn_model.predict(test_data)
```

Support Vector Machine (SVM) SVM-ները դրոշմակնիքի վավերացման խնդրում կօգտագործեն գույնի ինտենսիվության հիստոգրամներ, հյուսվածքի առանձնահատկությունները (օրինակ՝ հարթություն, կոպտություն), եզրերի և ձևերի նկարագրիչներ և պերֆորացիայի նախշեր. ***Ճշգրտությունը կազմում է 75%:***

```
param_grid = {
    'C': [0.1, 1],
    'kernel': ['linear'],
    'gamma': ['scale']
}
grid_search = GridSearchCV(SVC(class_weight='balanced'), param_grid, cv=2)
grid_search.fit(train_data, train_labels.ravel())
print("Best Parameters:", grid_search.best_params_)
best_estimator = grid_search.best_estimator_
predictions_svm = best_estimator.predict(test_data)
```

Voting Classifier (SVM and KNN): Ինչպես արդեն ասվել է քվեարկության դասակարգիչները միավորում են բազմաթիվ մոդելների ուժեղ կողմերը՝ բարձրացնելով կատարողականությունը և ամրությունը: ***Ճշգրտությունը կազմում է 74%:***

```
svm_classifier = SVC(class_weight='balanced')
knn_classifier = KNeighborsClassifier()
param_grid_svm = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf'],
    'gamma': ['scale', 'auto']
}
param_grid_knn = {
    'n_neighbors': [3, 5, 7],
    'weights': ['uniform', 'distance'],
    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']
}
grid_search_svm = GridSearchCV(svm_classifier, param_grid_svm, cv=5)
grid_search_svm.fit(train_data, train_labels.ravel())
grid_search_knn = GridSearchCV(knn_classifier, param_grid_knn, cv=5)
grid_search_knn.fit(train_data, train_labels.ravel())
best_svm_model = grid_search_svm.best_estimator_
best_knn_model = grid_search_knn.best_estimator_
estimators = [('svm', best_svm_model), ('knn', best_knn_model)]
voting_classifier = VotingClassifier(estimators, voting='hard')
voting_classifier.fit(train_data, train_labels.ravel())
predictions_voting = voting_classifier.predict(test_data)
```

XGBoost Classifier XGBoost-ը գրադիենտ խթանող հզոր ալգորիթմ է, որն ունակ է լուծել դասակարգման բարդ առաջադրանքները: Այն կարող է սովորել բարդ հարաբերություններ դրոշմակնիքի հատկանիշների միջև: Այն արդյունավետ է դրոշմակնիքների վավերացման համար՝ հաջորդաբար բարելավելով մոդելների աշխատանքը՝ ֆիքսելով բարդ

առանձնահատկությունների փոխազդեցությունները, որոնք բնորոշ են դրոշմակնիքների պատկերներին: ***Ճշգրտությունը կազմում է 75%:***

```
xgb_model = xgb.XGBClassifier(
    objective='binary:logistic',  # binary classification
    eval_metric='logloss',        # metric to be used
    use_label_encoder=False       # as we're providing labels directly
)

param_grid = {
    'max_depth': [3, 6],               # depth of trees
    'n_estimators': [50, 100, 150],    # number of trees
    'learning_rate': [0.01, 0.1, 0.2]  # step size shrinkage used to prevent overfitting
}
grid_search = GridSearchCV(xgb_model, param_grid, cv=3, verbose=1)  # Use 3-fold cross-validation
grid_search.fit(train_data_resampled, train_labels_resampled)
best_xgb_model = grid_search.best_estimator_
predictions = best_xgb_model.predict(test_data)
```

Random Forest Classifier: SVM-ի նման, Random Forest-ը կարող է օգտագործել կնիքների պատկերներից արդյունահանված նույն հատկանիշները և տարածական բաշխման տեղեկատվությունը: Random Forest-ը կառուցում է բազմաթիվ որոշումների ծառեր և միացնում դրանց կանխատեսումները: Այն իդեալական է դրոշմակնիքների վավերացման համար՝ շնորհիվ տարբեր առանձնահատկություններ և բարդ առանձնահատկությունների փոխազդեցության ունակության, ինչը հանգեցնում է բարձր ճշգրտության: ***Ճշգրտությունը կազմում է 78%:***

```
param_grid_rf = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
rf_classifier = RandomForestClassifier(random_state=42)
grid_search_rf = GridSearchCV(rf_classifier, param_grid_rf, cv=5)
grid_search_rf.fit(train_data, train_labels.ravel())
best_rf_model = grid_search_rf.best_estimator_
predictions_rf = best_rf_model.predict(test_data)
```

Մանրակրկիտ փորձարկումներից և վերլուծությունից հետո Random Forest դասակարգիչը հայտնվում է որպես ստորագրության վավերացման

ամենաարդյունավետ մեթոդը, որն ապահովում է ամենաբարձր ճշգրտությունը՝ 78%:

Այնուամենայնիվ, կարևոր է հաշվի առնել հայտի հատուկ պահանջներն ու սահմանափակումները նախքան վավերացման մեթոդի ընտրությունը վերջնականացնելը:

Գրաֆիկական ինտերֆեյսի մշակում

Այս ինտերֆեյսը պատկերների մշակման հավելված է, որը հատուկ նախագծված է փաստաթղթերի կնիքի և ստորագրության համակարգի համար: Այն ծառայում է որպես բազմակողմանի գործիք փաստաթղթերի պատկերները վերլուծելու, մշակելու և վավերացնելու համար:

Նկար 4.1-ում նեկրկայացված է ինտերֆեյսի տեսքը և բաղադրիչները:

Մանրամասն քննարկենք ինտերֆեյսի յուրաքանչյուր բաղադրիչ:

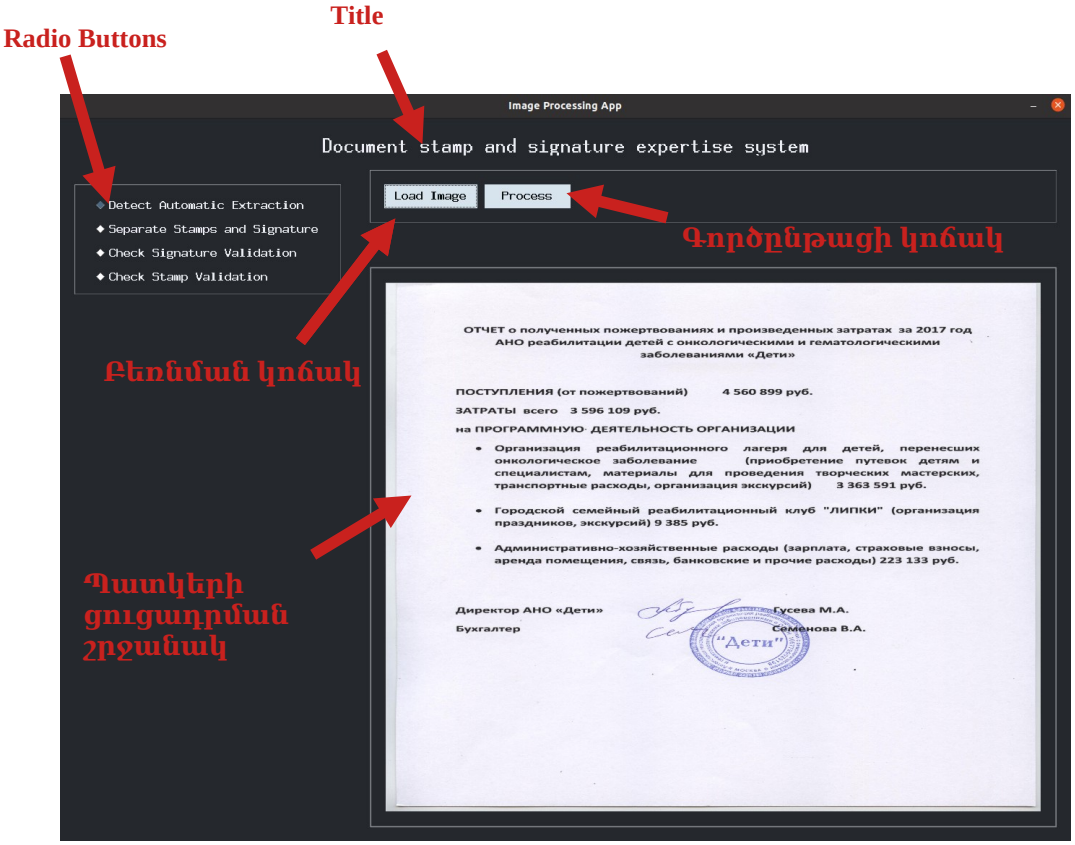
Վերնագիր և ռադիո կոճակներ (Title and Radio Buttons): Վերնագրի պիտակը հստակ նշում է հավելվածի նպատակը: Ռադիո կոճակները առաջարկում են տարբեր առաջադրանքների ընտրություն՝ ապահովելով օգտվողներին հեշտությամբ ընտրել այն գործողությունը, որը ցանկանում են կատարել: Յուրաքանչյուր ռադիոկոճակ կապված է որոշակի գործառույթի հետ, ինչը ինտուիտիվ է դարձնում օգտվողների համար տարբերակները հասկանալը:

Պատկերի բեռնման կոճակ (Load Image Button): Այս կոճակը օգտատերերի համար ծառայում է պատկերներ մուտքագրելու գործիք: Այն բացում է պատուհան, որը հնարավորություն է տալիս օգտվողներին իրենց ֆայլային համակարգից ընտրել նախընտրած պատկերը: Թույլ է տալիս ընտրել հետևյալ ձևաչափերից՝ *.png *.jpg *.jpeg :

Գործընթացի կոճակ (Process Button): «Process» կոճակը բեռնված պատկերի վրա ընտրված գործողությունը կատարելու գործիքն է: Դրա սկզբնական անջատված վիճակը թույլ չի տալիս օգտատերերին փորձել մշակել պատկերները նախքան դրանք բեռնելը: Պատկերը բեռնվելուց հետո կոճակը դառնում է ակտիվ՝ ազդարարելով օգտվողին, որ նրանք կարող են շարունակել մշակումը:

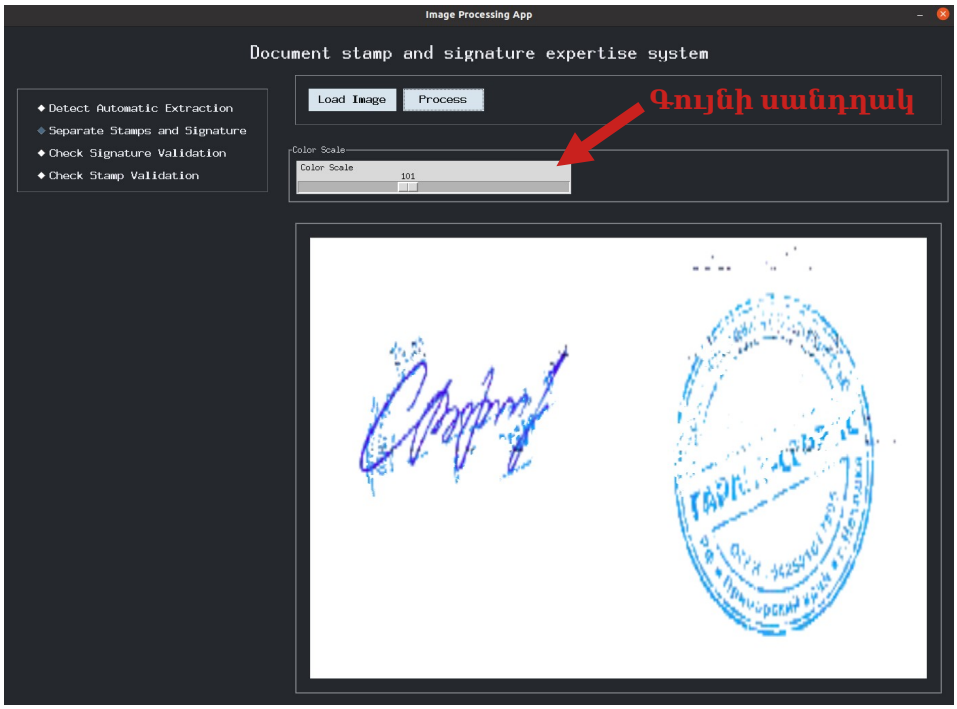
Պատկերի ցուցադրման շրջանակ (Image Display Frame): Այս տարածքը ծառայում է որպես տեսողական հետադարձ կապի մեխանիզմ՝ ցուցադրելով բեռնված պատկերը և մշակման գործողությունների արդյունքները: Շրջանակի չափերը ֆիքսված են՝ ապահովելով պատկերների հետևողական ներկայացում: Պատկերների չափերը փոխվում են՝ կադրում առկա տարածությանը համապատասխանելու

համար՝ օպտիմալացնելով տեսանելիությունը և ապահովելով պատշաճ հավասարեցում:



Նկար 4.1 Interface

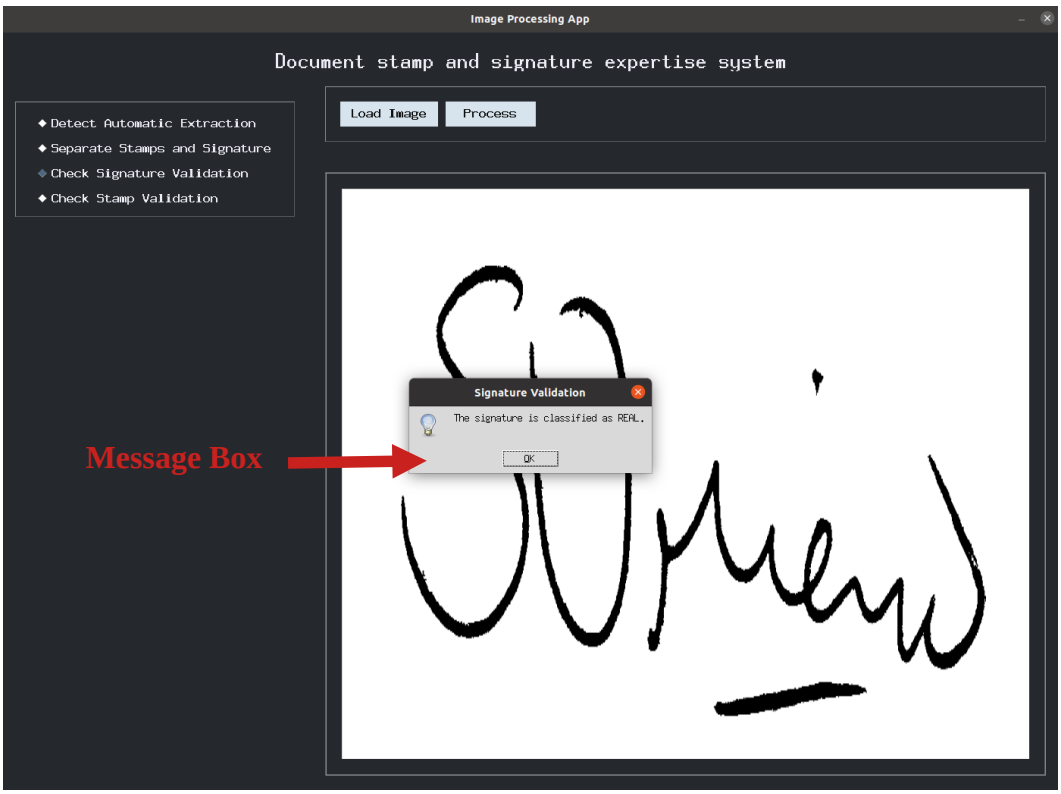
Գույնի սանդղակ (Color Scale): Գունային մասշտաբի սահիչն օգտատերերին հնարավորություն է տալիս շտկել գունային շեմը դրոշմակնիքների և ստորագրությունների բաժանման ժամանակ: Կարգավորելով գունային սանդղակը, օգտվողները կարող են հարմարեցնել տարանջատման գործընթացը: Այն ցուցադրված է նկար 4.2-ում



Նկար 4.2 Color scale

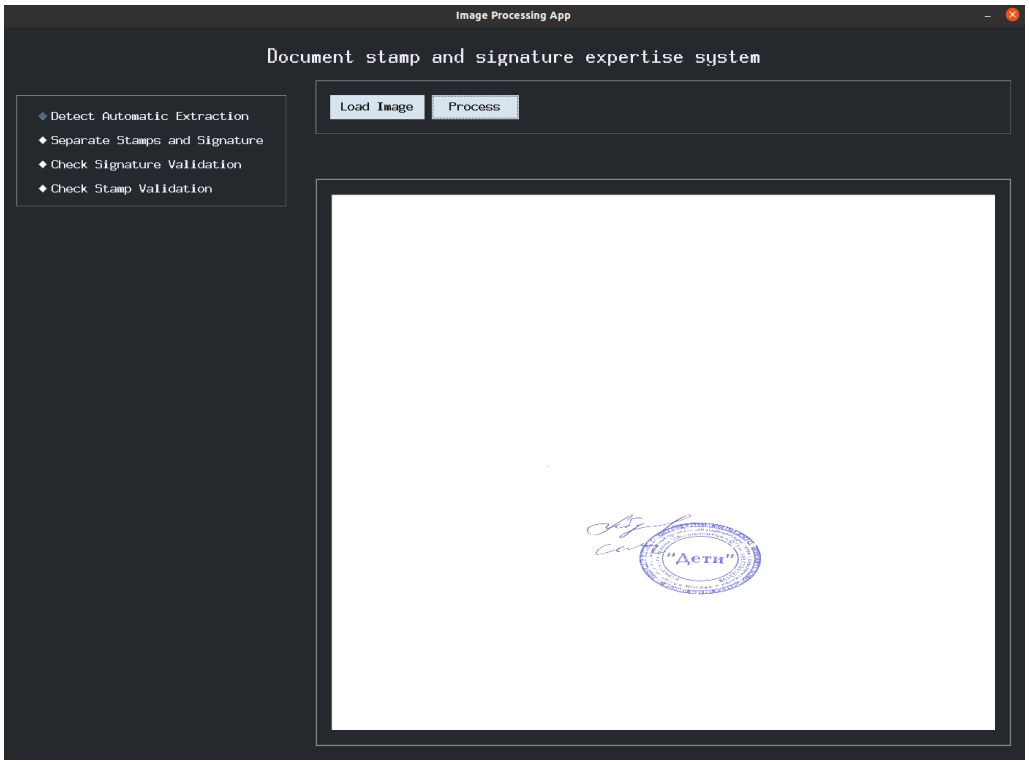
Հաղորդագրությունների տուփեր (Message Boxes): Հաղորդագրությունների տուփերը վճռորոշ դեր են խաղում արդյունքների և հետադարձ կապի համար: Նրանք տրամադրում են տեղեկատվական ծանուցումներ մշակման գործողությունների հաջողության կամ ձախողման մասին՝ օգնելով օգտատերերին հասկանալ իրենց առաջադրանքների կարգավիճակը: Սխալների հաղորդագրությունները ցուցադրվում են անվավեր մուտքերի կամ անսպասելի խնդիրների դեպքում: Այդպիսի օրինակ ներկայացված է նկար 4.3-ում:

Լրացուցիչ հատկանիշներ: Ինտերֆեյսը ներառում է առաջադեմ առանձնահատկություններ, ինչպիսիք են սխալների կառավարումը և դասավորության դինամիկ կարգավորումները՝ օգտագործելիությունն ու ամրությունը բարձրացնելու համար: Սխալների հետ աշխատելու մեխանիզմները թույլ չեն տալիս օգտվողներին հանդիպել անսպասելի վարքագծի կամ խափանումների՝ խթանելով օգտատիրոջ դրական փորձը:

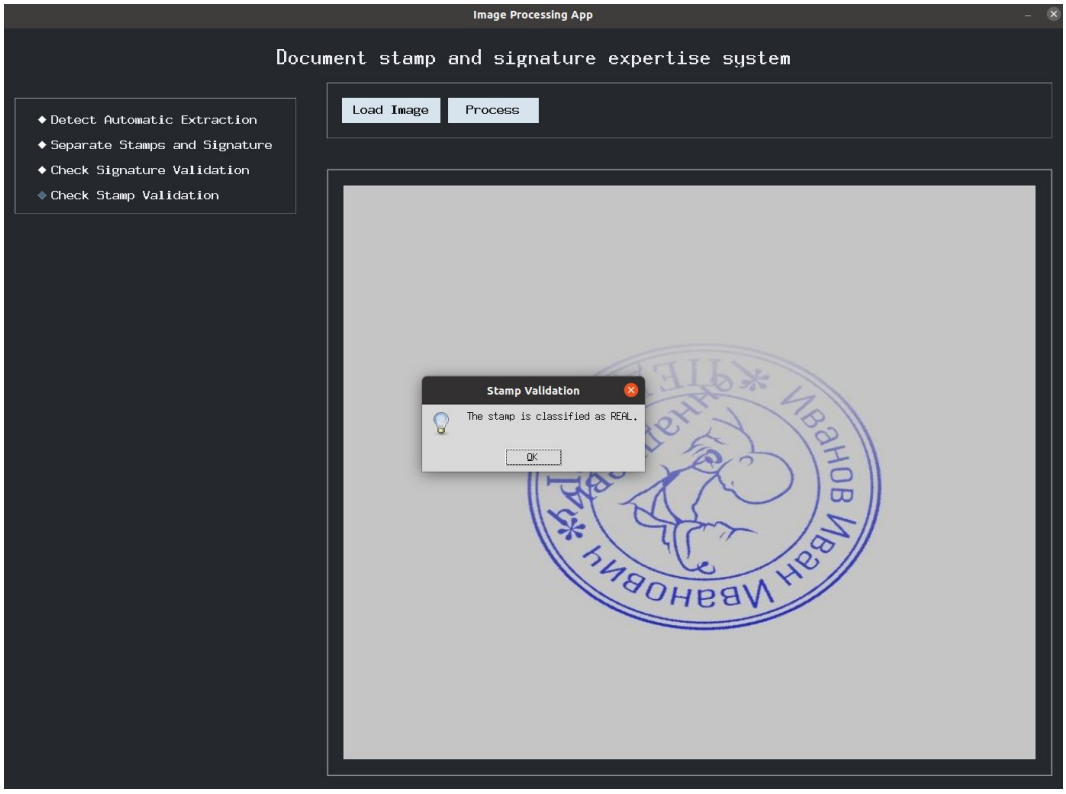


Նկար 4.3 Message Box

Նկար 4.4-ում և Նկար 4.5-ում ներկայացված են այլ օրինակներ



Նկար 4.3 Detect Automatic Extraction



Նկար 4.3 Check Stamp Validation

ԵԶՐԱԿԱՑՈՒԹՅՈՒՆ

Ուսումնասիրեցինք մեքենայական ուսուցման մոդելներ՝ կնիքների և ստորագրությունների իսկությունը ստուգելու համար: Ստորագրության ստուգումը կատարվեց այսպիսի հատկանիշների հիման վրա ինչպիսիք են. գրիչի ճնշման տատանումները, թանաքի լայնությունը և հետևողականությունը, ելակետային հետևողականությունը, ստորագրության ընդհանուր հոսքը և ռիթմը: Կնիքի վավերացումը կատարվեց հիմնվելով անոմալիաներ հայտնաբերելու տեխնիկայի վրա:

Մշակվեց ալգորիթմ փաստաթղթերի պատկերներում կնիքները և ստորագրությունները հայտնաբերելու համար, այս ալգորիթմում օգտագործվեց երկքայլ մոտեցում՝ կենտրոնանալով պատկերի ալիքների ինտենսիվության մանիպուլյացիայի և ֆոնի մեկուսացման վրա: Մշակվեց ալգորիթմ՝ overlap (համընկնող) օբյեկտները (ստորագրություններ և կնիքներ) առանձնացնելու համար՝ օգտագործելով գույնի վրա հիմնված հատվածավորման տեխնիկա:

Մշակեցինք օգտվողի համար հարմար գրաֆիկական ինտերֆեյս (GUI)՝ համակարգից ավելի հեշտ օգտվելու համար:

Ալգորիթմների առավելություններ:

Ներկայացված երկու ալգորիթմների պարզությունն ու արդյունավետությունը դարձնում են այն հարմար բազմաթիվ ծրագրերի համար: Այն արդյունավետ է անկախ պատկերի լուսավորությունից, աղմուկից, օբյեկների դասավորության քանակից և ձևից: Հայտնաբերուման արդյունքում՝ նկարի վայից հեռացվում են նաև կնիքի և ստորագրության վրա գտնվող փաստաթղթի այլ հատվածները, որոնք այլ մոտեցումների դեպքում լավ արդյունք չէին ապահովի:

Ալգորիթմների թերություններ:

Ալգորիթմների գաղափարը մեծապես հիմնված է կապույտ թանաքի առկայության վրա: Ոչ ստանդարտ թանաքի գույներով փաստաթղթերը կարող են հանգեցնել ոչ ճշգրիտ հայտնաբերման:

Մեքենայական ուսուցման մոդելների առավելություններ:

Մանրակրկիտ հետազոտության և հիպերպարամետրային թյունինգի միջոցով մենք հասանք բարձր ճշգրտության՝ օգտագործելով

տարբեր մեքենայական ուսուցման ալգորիթմներ: Կողմնորոշված գրադիենտների հիստոգրամի (HOG) առանձնահատկությունների օգտագործումը թույլ տվեց մանրամասն վերլուծել կառուցվածքային բնութագրերը և օրինաչափությունները ստորագրությունների մեջ՝ բարձրացնելով հայտնաբերման ճշգրտությունը:

Մեքենայական ուսուցման մոդելների թերություններ:

Կեղծ կնիքների դատասեթի բացակայության պատճառով, չունեցանք այն արդյունքը, որը ունենք ստորագրությունների դեպքում:

ԳՐԱԿԱՆՈՒԹՅՈՒՆ

- ✓ R. Gonzalez, R. E. Woods և S. L. Eddins, «Digital Image Processing Using MATLAB», 4th ed., Pearson Education, 2018:
- ✓ A. K. Jain, «Fundamentals of Digital Image Processing», Springer International Publishing, 2018:
- ✓ https://stackoverflow.com/
- ✓ https://docs.opencv.org/4.x/
- ✓ https://ml-cheatsheet.readthedocs.io/en/latest/index.html