

**МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
ХЭРЭГЛЭЭНИЙ ШИНЖЛЭХ УХААН, ИНЖЕНЕРЧЛЭЛИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРИЙН УХААНЫ ТЭНХИМ**

Цэрэнбямбаагийн Хасбилэгт

**ГАР БИЧМЭЛ ТАНИЛТАНД ЗОРИУЛСАН
МУИС-Н НЭЭЛТТЭЙ ӨГӨГДӨЛ БЭЛТГЭХ НЬ
(Preparation of open NUM handwritten character dataset)**

Мэдээллийн технологи (D061304)
Бакалаврын судалгааны ажил

Улаанбаатар

2020 он

МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
ХЭРЭГЛЭЭНИЙ ШИНЖЛЭХ УХААН, ИНЖЕНЕРЧЛЭЛИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРИЙН УХААНЫ ТЭНХИМ

ГАР БИЧМЭЛ ТАНИЛТАНД ЗОРИУЛСАН МУИС-Н
НЭЭЛТТЭЙ ӨГӨГДӨЛ БЭЛТГЭХ НЬ
(Preparation of open NUM handwritten character dataset)

Мэдээллийн технологи (D061304)
Бакалаврын судалгааны ажил

Удирдагч: _____ Др. Б.Сувдаа

Гүйцэтгэсэн: _____ Ц.Хасбилэгт (15B1SEAS0980)

Улаанбаатар

2020 он

Зохиогчийн баталгаа

Миний бие Цэрэнбямбаагийн Хасбилэгт "ТАР БИЧМЭЛ ТАНИЛТАНД ЗОРИУЛСАН МУИС-Н НЭЭЛТТЭЙ ӨГӨГДӨЛ БЭЛТГЭХ НЬ" сэдэвтэй судалгааны ажлыг гүйцэтгэсэн болохыг зарлаж дараах зүйлсийг баталж байна:

- Ажил нь бүхэлдээ эсвэл ихэнхдээ Монгол Улсын Их Сургуулийн зэрэг горилохоор дэвшүүлсэн болно.
- Энэ ажлын аль нэг хэсгийг эсвэл бүхлээр нь ямар нэг их, дээд сургуулийн зэрэг горилохоор оруулж байгаагүй.
- Бусдын хийсэн ажлаас хуулбарлаагүй, ашигласан бол ишлэл, зүүлт хийсэн.
- Ажлыг би өөрөө (хамтарч) хийсэн ба миний хийсэн ажил, үзүүлсэн дэмжлэгийг дипломын ажилд тодорхой тусгасан.
- Ажилд тусалсан бүх эх сурвалжид талархаж байна.

Гарын үсэг: _____

Огноо: _____

Гарчиг

ЗУРГИЙН ЖАГСААЛТ	iv
КОДЫН ЖАГСААЛТ	vi
УДИРТГАЛ	1
Зорилго	1
Зорилт	1
БҮЛГҮҮД	2
1. СУДАЛГАА	2
1.1 Нийтлэг ашиглагддаг сангууд	2
1.2 Судалгаанаас хийсэн дүгнэлт	6
2. АРГЫН ТОДОРХОЙЛОЛТ	8
2.1 Оролт	8
2.2 Боловсруулалт	11
2.3 Гаралт	12
3. ХӨГЖҮҮЛЭЛТ	13
3.1 Оролтын зураг боловсруулалт	13
3.2 Тэмдэгтүүдийг хүснэгтээс салгах	18
3.3 Гаралт	19
4. ПРОГРАМ ХАНГАМЖ	23
4.1 Sanitizer	23
4.2 NUMiner	24
5. ҮР ДҮН	28
6. ДҮГНЭЛТ	29
НОМ ЗҮЙ	30
ХАВСРАЛТ	32
А. ИРМЭГ ИЛРҮҮЛЭГЧ	32

B. ХҮСНЭГТ ИЛРҮҮРЭГЧ	33
C. ТЭМДЭГТҮҮДИЙГ САЛГАХ	35
D. ТЭМДЭГТ БОЛОВСРУУЛАХ	36
E. NUMINER — ТОХИРГООГ УНШИХ	38

Зургийн жагсаалт

Зураг	Хуудас
1.1 NIST SD19 сангийн гар бичмэл тэмдэгтүүд цуглуулах зорилготой- гоор бэлдсэн маягт. Гараар бөглөх 33 хэсгүүдтэй ба нэр, огноо оруу- лах нүднээс бусад хэсгүүд нь боловсруулагдсан.	3
1.2 MNIST сангийн жишээ тэмдэгтүүд [3]	4
1.3 EMNIST санг үүсгэхэд NIST сангийн зургуудыг боловсруулахад ашигласан хувиргалтын дараалал [2]	5
2.1 Өгөгдөл цуглуулахдаа ашиглахаар бэлтгэсэн гар бичмэлийн маяг- тын хувилбарууд	9
2.2 Маягтын зургуудыг зураг боловсруулах аргууд ашиглан үндсэн бич- вэрийг хадгалж буй хүснэгтийг илрүүлэн, хүснэгтийн нүд бүрийг салган авахад үүссэн үр дүн.....	10
2.3 Гар бичмэлийн маягтын хамгийн эцсийн хувилбар	11
3.1 Hysteresis Thresholding - https://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html -аас зургийг авав.....	14
3.2 (a). Маягтын зураг дээр доод заагийг 100, дээд заагийг 200 утгаар Canny -н ирмэг илрүүлэх аргыг ашигласан үр дүн, (b). Илрүүлсэн ирмэгүүдээс контор цэгүүдийг олж, хамгийн урт хүрээний урттай контор цэгүүдийг багтаасан хамгийн жижиг тэгш өнцөгт олсон бай- дал, (c). Хамгийн том контор цэгүүдээс захын дөрвөн цэгийн утгуу- дыг олж харах өнцөгийн хувиргалтыг арилгасан байдал	17
3.3 Оролтын зураг боловсруулалтын үр дүнгээс гараар бичигдсэн тэм- дэгтүүдийг нэг нэгээр нь ялган авсан байдал	18

3.4	(a). Хүснэгтээс салган авсан A үсэг, (b). Зургаас контор цэгүүдийг олж бүгдийг нь багтаах хамгийн жижиг тэгш өнцөгт олж харьцааг нь алдагдуулалгүйгээр талуудыг тэнцүүлсэн байдал, (c). Талыг нь тэнцүүлсэн зургийг 28x28 болгож багасгасан байдал, (d). Зураг-ны пикселүүдийг Otsu -н зааглалтын аргыг ашиглан хоёртын зураг болгон хувиргасан ба гаралтанд бэлэн болсон зураг	20
4.1	Sanitizer -г ашиглан нийт 97 тэмдэгт бүхий нийт 255576 зургаас давхардсанг ялгах үеийн оролт болон гаралт.....	23
4.2	Sanitizer програмын жишээ үр дүн	25
5.1	Цугларсан тэмдэгт бүрт харгалзах өгөгдөл/зургийн тоо	28

Кодын жагсаалт

3.1	Хамгийн урт контор цэгүүд буюу энэ тохиолдолд бичвэр хадгалж буй хүснэгтийн ирмэг дээрх цэгүүд	15
3.2	Тоймлосон цэгүүдээс хүснэгтийн булангийн цэгүүдийг олох	15
3.3	Харах өнцгийн хувиргалтыг арилгах	17
3.4	Хамгийн эхний маягтын хувилбарын нүднүүд, тэдгээрт харгалзах тэмдэгтүүдийн дараалал	19
3.5	Тэмдэгтээс контор цэгүүдийг олж бүгдийг нь багтаасан хамгийн жижиг тэгш өнцөгтөөр тасдан авах	20
4.1	Параметраар өгөгдсөн файлын нэрээр зургийг уншин тэмдэгтийн түлхүүр (label), зургийн нэр, hash утгыг агуулсан tuple буцаах функц	23
4.2	NUMiner — нэг маягтыг боловсруулах команд	25
4.3	NUMiner — олон маягтуудыг нэг дор боловсруулах команд буюу маягтуудыг агуулж буй хавтсын замыг зааж өгөх	26
4.4	NUMiner — config хавтас доторх labels.json нэртэй файлаас тэмдэгтийн түлхүүрүүдийг ямар дугаартай харгалзуулан хадгалахыг уншиж, data/INPUT доторх маягтуудыг боловсруулан үр дүнг data/OUTPUT дотор хадгал	26
4.5	NUMiner — labels.json	27
4.6	NUMiner convert — <src> доторх тэмдэгтийн төрөл бүрээс <size> ширхэг зургийг авах ба нийт өгөгдлөөс <ratio> хувийг нь (хэрэв ratio бүхэл тоо бол нийт өгөгдлөөс салган авах тестийн өгөгдлийн хувь, train бол бүхэлдээ сургалтын, test бол бүхэлдээ тестийн өгөгдөл гэж тус тус үзнэ) тестийн өгөгдөлд зориулан боловсруулж <dst> -д хадгална.	27

УДИРТГАЛ

Өнөөдөр хиймэл оюун ухаан болон машин сургалтын салбаруудын технологи, судалгаанууд хөгжихийн хэрээр, өмнө нь хүний төсөөлж байгаагүй олон шинэ боломжууд, зөвхөн технологийн салбаруудаар хязгаарлагдалгүй ар араасаа нээгдэж байгаагаас гадна өнөөдөр нийтэд ашиглагдаж буй технологиудын нарийвчлал, үр дүн, чанар ч мөн үүнийг дагаад сайжирч байгаа билээ. Энэ дундаас гар бичмэл танилттай холбоотой судалгаанууд олноор хөгжүүлэгдэх нь бидний мэдэх хүн компьютерийн харилцааг эергээр өөрчилж байна.

Зорилго

Энэ чиглэлийн судалгаанууд, түүн дээр суурилсан програм хангамж, үйлчилгээ, бүтээгдэхүүн хөгжүүлэгдэхэд гар бичмэлийг таних, машин сургалтанд ашиглах өгөгдөл зайлшгүй хэрэгтэй, мөн Монгол хэл дээр энэ төрлийн нээлттэй, нэгдсэн өгөгдлийн сан хомс учир энэхүү судалгааны ажлаар өөрийн эх хэл дээрх нээлттэй гар бичмэлийн санг бий болгох, санг үүсгэхэд ашиглах арга, хэрэгслийг тодорхойлохыг зорьсон юм.

Зорилт

Монгол хэл дээрх гар бичмэлийн нээлттэй санг үүсгэхдээ дараах үе шатын дагуу ажиллана.

1. Үндсэн шаардлагуудыг, аргуудыг тодорхойлох
2. Тэмдэгтүүд бичүүлж авах маягтын загварыг гаргах
3. Бөглөгдсөн маягт дээр боловсруулалт хийх, тэмдэгтүүдийг ялгаж авах
4. Их хэмжээний оролтын үед боловсруулалт хийх боломжтой байдлаар програмыг хөгжүүлэх
5. Програмын нээлттэй авч ашиглах боломжийг хангахын тулд эх код, програмыг GitHub, PyPi дээр байршуулах

1. СУДАЛГАА

Машин сургалт тэр дундаа гар бичмэл танилттай холбоотой машин сургалтын өгөгдлийг хэрхэн бэлтгэх, зохион байгуулах тал дээр бодитой, албан ёсны судалгааны ажлууд хомс байсан тул өгөгдлийн хэлбэр, хэмжээ, санг хэрхэн зохиомжлох зэргийг энэ чиглэлийн өөр бусад хэл дээрх түгээмэл ашиглагддаг нээлттэй өгөгдлүүдийг судалсаны үндсэн дээр тэдгээрийн жишигт тааруулан гүйцэтгэхээр шийдсэн.

Англи, орос, хятад, япон, солонгос хэл дээрх хамгийн нийтлэг ашиглагддаг сангуудаас анх өгөгдлийг хэрхэн цуглуулсан болон яаж боловсруулалт хийгдсэн талаарх мэдээлэл хэр хангалттай байсныг харгалзан үзээд дараах англи хэл дээрх дөрвөн сангуудын хэрхэн боловсруулагдсан байдлыг дэлгэрэнгүй судлалаа.

1.1 Нийтлэг ашиглагддаг сангууд

1.1.1 *NIST 19 special database (SD19)*

1995 оны 3 дугаар сард NIST¹ газраас танилцуулагдсан энэ сан нь нийт 3699 хуудас бүхий хоёртын гараар бөглөгдсөн маягт (Зураг 1.1), тэдгээрээс ялгаж авсан 814255 тоо, үгсийн цуглуулгаас бүрдэж байсан. Зөвхөн тоо ([0-9]), үсэг ([a-z][A-Z]) оролцсон учир нийт 62 төрлийн тэмдэгтүүд, тэмдэгт бүр нь 128x128 харьцаатайгаар зохион байгуулагдсан. [1]

Цуглуусан өгөгдлөө `by_write`, `by_field`, `by_class`, `by_merge` хэмээх дөрвөн өөр төрлөөр ангилсан бөгөөд өгөгдлүүдээ хадгалахдаа дараах өөрсдийн зохиосон өгөгдлийн форматуудыг ашигалсан. Үүнд:

- `.mis`² — ялган авсан тэмдэгтийн зургуудыг хадгалах файлын төрөл
- `.pct` — бүтэн хуудас маягтын зургийг хадгалах файлын төрөл
- `.cls` — 62 ялгаатай төрлийн тэмдэгтүүдийн холбогдох зургийн мэдээллийг хамт хад-

¹National Institute of Standards and Technology

²Multiple Image Set

HANDWRITING SAMPLE FORM

NAME [REDACTED] DATE 8-3-89 CITY MINDEN CITY STATE MI ZIP 48456

This sample of handwriting is being collected for use in testing computer recognition of hand printed numbers and letters. Please print the following characters in the boxes that appear below.

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

0123456789 0123456789 0123456789

87 701 3752 80759 960941

87 701 3752 80759 960941

158 4586 32123 832656 82

158 4586 32123 832656 82

7481 80539 419219 67 904

7481 80539 419219 67 904

61738 729658 75 390 5716

61738 729658 75 390 5716

109334 40 625 4234 46002

109334 40 625 4234 46002

gyxlakpdsbtziumwfgjenhocv

gyxlakpdsbtziumwfgjenhocv

ZXSBNCECMYWQTKFLUOHPIRVDJA

ZXSBNCECMYWQTKFLUOHPIRVDJA

Please print the following text in the box below:

We, the People of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common Defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America.

We, the People of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common Defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America.

Зураг 1.1: NIST SD19 сангийн гар бичмэл тэмдэгтүүд цуглуулах зорилготойгоор бэлдсэн маягт. Гараар бөглөх 33 хэсгүүдтэй ба нэр, огноо оруулах нүднээс бусад хэсгүүд нь боловсруулагдсан.

галж буй файлын төрөл

- .mit — маягт бөглөсөн хүний мэдээллийг холбогдох тэмдэгтүүдийн мэдээлэлтэй хамт хадгалж буй файлын төрөл

NIST SD19 сангийн тийм ч өргөн ашиглагдаагүй шалтгаан нь өөрсдийн зохиосон форматаар өгөгдлийг хадгалж байсан ба энэ нь авч ашиглахад хүндрэлтэй байсан учир түүнээс хойш 2016 оны 9 сард санг хадгалах файлын төрлийг PNG форматтай болгон шинэчилж дахин шинэ хувилбар гаргасан. Өгөгдлийн форматыг өөрчилсөн хэдий ч өмнөх хувилбаруудтайгаа нийцгүй мөн үндсэн өгөгдөл нь өөрчлөгдсөн байсан нь сөргөөр нөлөөлсөн.

1.1.2 MNIST [3]

Анх 1998 онд дээр дурдсан NIST -н Special Database 1 болон 3 дахь хувилбаруудын 0 -с 9 хүртэл 10 төрлийн тэмдэгтүүдтэйгээр танилцуулагдсан бөгөөд өнөөдрийг хүртэл оюун ухаан, хиймэл оюуны салбарт хамгийн түгээмэл ашиглагддаг сан яах аргагүй мөн билээ.

NIST нь анхлан SD-3 санг сургалтанд, SD-1 санг тестэд зориулан зохион байгуулсан ба учир нь сургалтанд зориулсан өгөгдлүүд нь нөгөөхөөсөө хамаагүй цэвэрхэн, сайн боловсруулагдсан, амар танигдахаар байсан бол SD-1 нь ахлах сургуулийн сурагчдаар бөглүүлсэн маягтаас боловсруулагдсан учир танихад харьцангуй хэцүү байсан юм. Тийм ч учраас эдгээрх сангуудыг тэнцүү хуваан, дахин зохион байгуулж сургалтын 60000, тестийн 10000 зурагтай MNIST³ -г үүсгэсэн байна.[4]



Зураг 1.2: MNIST сангийн жишээ тэмдэгтүүд [3]

NIST -н оролтын хоёртын зургуудыг эхлээд 20x20 хэмжээтэйгээр багасган боловсруулж үндсэн гурван хэлбэрээр туршсан. Эхнийх нь хэмжээг нь багасгасан зургуудынхаа төвийг олон, харьцааг алдагдуулахгүйгээр 28x28 дотор багтаасан. Хоёр дахь хувил-

³Modified NIST

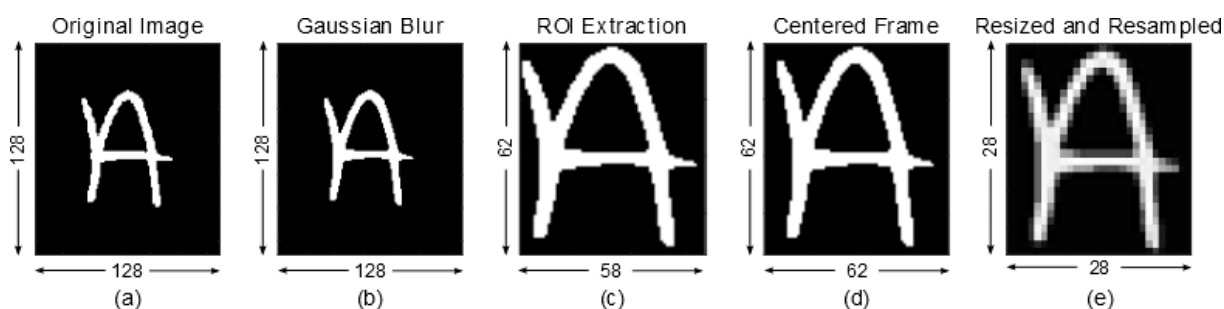
барт оролтын зургуудын налалтыг арилган тэгшлээд 20x20 хэмжээтэй болгон багасгаж *deslanted* буюу налууг арилгасан санд ашигласан. Харин сүүийн хэлбэр нь зургуудын хэмжээг нь 16x16 болгон багасгаж ашигласан.

Өнөөдөр MNIST сан нь дээрх төрлүүдийн хамгийн эхний хэлбэрээр вектор, олон хэмжээст матриц хадгалахад амар *IDX* гэх файлын төрөлтэйгөөр ашиглагдаж байгаа билээ.

1.1.3 EMNIST [2]

Extended Modified NIST буюу EMNIST нь аль хэдийн MNIST санг ашиглаж байгаа алгоритм, системүүд ямар ч асуудалгүйгээр авч ашиглах боломжтой байх илүү өргөтгөсөн нээлттэй санг үүсгэх зорилготойгоор танилцуулагдсан юм.

Оролтын зургийг боловсруулах дараалал нь ерөнхийдөө MNIST[3] -г үүсгэх зорилгоор NIST[1] -н өгөгдлүүдийг хэрхэн боловсруулсан аргатай ойролцоо ч зургийг жижигрүүлэхэд ашигласан алгоритм, арга зэрэг нь өөр байгаа юм. Зурган дээр хувиргалт хийхдээ эхлээд зургийн урт, өргөнөөс хамгийн их хэмжээгээр хамгийн жижиг багтаасан тэгш өнцөгтийг тооцоолж хоосон үлдсэн хэсэгт дүүргэлт хийснээр урт, өргөнийг тэнцүүлэн дөрвөлжин зураг үүсгэсэн. Зургийн хэсгүүдийг захын ирмэгүүдэд наалдуулахгүйн тулд мөн 2 пикселийн хүрээг нэмжээ. Эндээс зургаа *bi-cubic interpolation* алгоритм ашиглан 28x28 хэмжээтэй болгон жижигрүүлсэн байна. Зураг 1.3.



Зураг 1.3: EMNIST санг үүсгэхэд NIST сангийн зургуудыг боловсруулахад ашигласан хувиргалтын дараалал [2]

Машин сургалтанд зориулагдсан сангууд сургалтын болон тестийн гэх хоёр хэсэгт хуваагддаг бөгөөд EMNIST -н хувьд NIST -н өгөгдлийг боловсруулахдаа MNIST -н баримталсан аргыг буюу Census Bureau -н ажилчид, ахлах сургуулийн сурагчдаас цуглуулсан

тэмдэгтүүдийг сургалт болон тестийн өгөгдөлдөө тэнцүү хуваан, тестийн өгөгдөл үүсгэхдээ тэдгээрээс санамсаргүй аргаар сонгон авч тус тус үүсгэжээ.

1.1.4 IAM [6]

Англи хэл дээрх хамгийн түгээмэл ашиглагддаг сангуудын нэг нь IAM⁴ сан бөгөөд дээр дурдсан сангуудаас ялгаатай нь үндсэн өгөгдөл нь дан гараар бичигдсэн бүтэн өгүүлбэрээс цуглуулагдсан буюу LOB corpus [5]⁵ дээр суурилж энэхүү сан боловсруулагдсан. LOB corpus -н анхны хувилбар 1970 онд танилцуулагдсан бөгөөд 400 гаруй хүний 1066 гараар бөглөгдсөн маягтаас авсан 10841 ялгаатай үгсийг агуулдаг.

1.2 Судалгаанаас хийсэн дүгнэлт

Өнөөдөр машин сургалт, гар бичмэл танилтын судалгаануудад хамгийн нийтлэг ашиглагддаг англи хэлний дөрвөн сангуудад судлагаа хийж үзээд дараах дүгнэлтүүдийг гаргалаа. Үүнд:

- Сан нь хэрхэн зохион байгуулагдсан, өгөгдлүүд ямар форматтайгаар хадгалагдсан, авч ашиглах, хувиргах боломжууд зэрэг нь энэ чиглэлээр судалгаа хийж буй хүмүүсийн дунд тархах, түгээмэл ашиглагдахтай шууд холбоотой. Иймээс санг авч ашиглах, өгөгдлүүдийг хувиргахад амар байх
- MNIST болон EMNIST нь энэ чиглэлийн сургалт, судалгаа, шинэ төслүүдэд хамгийн түгээмэл ашиглагдаж буй сан мөн бөгөөд үүнтэй холбоотойгоор эдгээр сангуудыг ашигладаг олон систем, програм хангамжууд байгаа учир өөрийн үүсгэж буй сангаа эдгээр сангуудтай нийцтэй ажиллах боломжтой байдлаар зохиомжлох
- Цаашид өөрийн сангаас хүмүүст авч ашиглахад амар байлгах үүднээс боломжит дэд сангууд (subset) нэмэлтээр үүсгэх

⁴Institut für Informatik und angewandte Mathematik (= Компьютерийн ухаан, хэрэглээний математикийн тэнхим), Берны их сургууль, Берн, Швейцарь

⁵Lancaster-Oslo болон Bergen сургуулиудын хамтран боловсруулсан сан <http://korpus.uib.no/ibase/manuals/LOB/INDEX.HTM>

- Сангийн сургалтын болон тестийн өгөгдлүүдийг хэрхэн ялгаж, зохион байгуулах нь чухал бөгөөд гаргацтай зургууд, танихад хэцүү зургуудыг аль аль талд тэнцүү байдлаар хуваан зохион байгуулах

2. АРГЫН ТОДОРХОЙЛОЛТ

Энэхүү ажлын мөн чанар нь үндсэн судалгаа бөгөөд гэсэн хэдий ч үр дүн, түүнтэй хамт ашиглагдах програм болон санг үүсгэхдээ өөрийн тодорхойлсон дараах шаардлагуудад нийцүүлэн хөгжүүлэхээр зорьсон. Үүнд:

1. Өгөгдөл нь цэвэр гараар бичигдсэн ба маш сайн боловсруулагдсан байх
2. Тэмдэгтийн зургууд дахин давтагдаагүй, мөн тухайн тэмдэгтийн бичигдэж болох аль болох олон хувилбаруудыг агуулсан байх
3. Тэмдэгтүүд дээр шинэ зургууд нэмж өгөгдлийн хэмжээг томруулах, зөвхөн шаардлагатай тэмдэгтүүдээр өөр шинэ датасет үүсгэх боломжтой байх
4. Өгөгдлийг аль болох олон төрлөөр авч ашиглах, шаардлагатай статистик мэдээллийг авах боломжтой байх
5. Өгөгдөл нь өөрөө болон, өгөгдөл үүсгэх программ нь хэнд ч авч ашиглахад нээлттэй байх

2.1 Оролт

Энэхүү судалгааны ажлын оролтын хувьд хүний гараар бичигдсэн крилл үсэг, тоо болон тусгай тэмдэгтүүд байх бөгөөд *Шаардлага №1* -т тэмдэглэсний дагуу өгөгдөл нь цэвэр гараар бичигдсэн ба боловсруулалт сайн хийгдсэн байх ёстой. Тийм ч учраас тоо, том жижиг крилл үсгүүд болон хэд хэдэн тусгай тэмдэгтүүдээс бүрдсэн маягт гаргасан бөгөөд боловсруулалтын үр дүнг хялбаршуулах, өгөгдлийн чанарыг сайжруулах зорилготойгоор хэд хэдэн удаа шинэчилсэн. Зураг 2.1. Гэвч тайлангийн энэ хэсэгт ерөнхийдөө маягтын загварууд хэрхэн өөрчлөгдсөн дээр анхаарах бөгөөд яг боловсруулалт, ашигласан арга зэргийг энэ бүлгийн 2.2 хэсэгт дэлгэрэнгүй авч үзнэ.

Гэр бичмэлийн маягтыг зураг боловсруулах үед үндсэн бичвэрийг агуулах хүснэгтийг олоход амар байлгах үүднээс зузаан, тод хүрээ оруулсан, харин хүснэгтийн нүднүүдийн

Дугаар № _____

0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
№	т								
№	т								
А	Б	В	Г	Д	Е	Ё	Ж	З	И
А	Б	В	Г	Д	Е	Ё	Ж	З	И
Й	К	Л	М	Н	О	Ө	П	Р	С
Й	К	Л	М	Н	О	Ө	П	Р	С
Т	У	Ү	Ф	Х	Ц	Ч	Ш	Щ	Э
Т	У	Ү	Ф	Х	Ц	Ч	Ш	Щ	Э
Ю	Я								
Ю	Я								
а	б	в	г	д	е	ё	ж	з	и
а	б	в	г	д	е	ё	ж	з	и
й	к	л	м	н	о	ө	п	р	с
й	к	л	м	н	о	ө	п	р	с
т	у	ү	ф	х	ц	ч	ш	щ	э
т	у	ү	ф	х	ц	ч	ш	щ	э
ю	я								
ю	я								
ы	ь	э	ю	я					
ы	ь	э	ю	я					

(а) Хамгийн эхний хувилбар

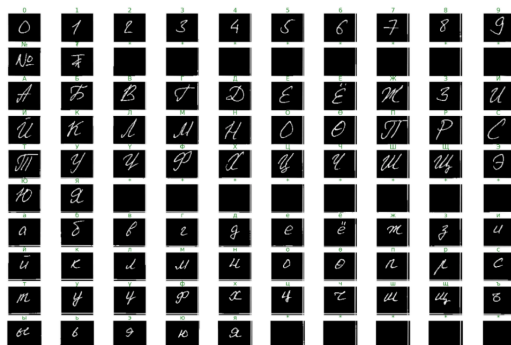
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
№	т								
№	т								
А	Б	В	Г	Д	Е	Ё	Ж	З	И
А	Б	В	Г	Д	Е	Ё	Ж	З	И
Й	К	Л	М	Н	О	Ө	П	Р	С
Й	К	Л	М	Н	О	Ө	П	Р	С
Т	У	Ү	Ф	Х	Ц	Ч	Ш	Щ	Э
Т	У	Ү	Ф	Х	Ц	Ч	Ш	Щ	Э
Ю	Я								
Ю	Я								
а	б	в	г	д	е	ё	ж	з	и
а	б	в	г	д	е	ё	ж	з	и
й	к	л	м	н	о	ө	п	р	с
й	к	л	м	н	о	ө	п	р	с
т	у	ү	ф	х	ц	ч	ш	щ	э
т	у	ү	ф	х	ц	ч	ш	щ	э
ю	я								
ю	я								
ы	ь	э	ю	я					
ы	ь	э	ю	я					

(b) Удаах хувилбар

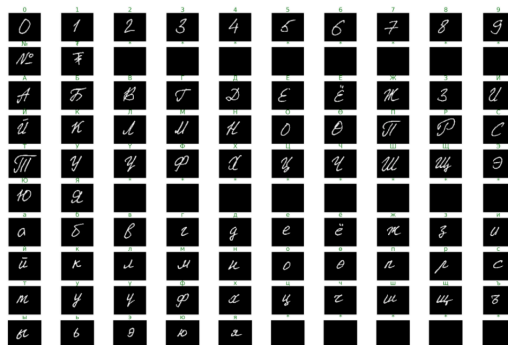
Зураг 2.1: Өгөгдөл цуглуулахдаа ашиглахаар бэлтгэсэн гар бичмэлийн маягтын хувилбарууд

хүрээг гадна талын хүрээнээс харьцангуй бүдэг байдлаар бэлтгэсэн хэдий ч боловсруулалтын үед бичвэр агуулж буй хүснэгтээс нүд болгоныг салган авахад Зураг 2.1a дээрх хамгийн гадна талын нүднүүдийн хүрээ хэтэрхий тод байсан нь цааш тэмдэгт бүр дээр боловсруулалт хийхэд саад болж байсныг Зураг 2.2a-ээс харж болно. Учир нь зургийг ямар өнцөгөөс дарсан, гэрэл сүүдэр хэрхэн бууснаас хамааран ялангуяа хүснэгтийн захын нүднүүдийн харгалзах талд маш тод зураасууд боловсруулалт хийгдсэний дараа ч үлдсэн байсан.

Энэ асуудлыг боловсруулалт хийх аргуудаа өөрчлөх, сайжруулах байдлаар шийдэж болох байсан хэдий ч нүдэнд тэмдэгтүүд хэрхэн бөглөгдсөн, зураг ямархуу байдлаар дарагдсан зэргээс хамааран тийм ч тохиромжтой шийдэл болохгүй гэж үзээд үндсэн маягтын загварыг Зураг 2.1b -т харагдаж буй болгон өөрчилсөн ба боловсруулалтын үр дүн хэрхэн сайжирсаныг Зураг 2.2b -аас харж болно.



(a) Хамгийн эхний хувилбарын боловсруулалт




(b) Удаах хувилбарын боловсруулалт

Зураг 2.2: Маягтын зургуудыг зураг боловсруулах аргууд ашиглан үндсэн бичвэрийг хадгалж буй хүснэгтийг илрүүлэн, хүснэгтийн нүд бүрийг салган авахад үүссэн үр дүн

Үүнээс гадна *Шаардлага №3* -т тодорхойлогдсоны дагуу шинээр тэмдэгтүүд нэмэх, өгөгдлийг өргөжүүлэх боломжтой байх ёстой ба үүний тулд оролтын маягтыг зөвхөн крилл үсгүүд, тоо болон хэдхэн тусгай тэмдэгтүүдээр хязгаарлаж болохгүй. Маягтад бөглөгдөх тэмдэгтүүдийг өөрчлөн (жишээ нь: өөр хэл дээр гар бичмэлийн сан үүсгэх) ашиглах боломжтой байлгах үүднээс маягтын загвар үүсгэхдээ дараах зүйлсийг анхаарсан. Үүнд:

1. Хүснэгтийн нүд бүрийн хэмжээ адилхан байх — Учир нь хүснэгтээс тэмдэгтүүдийг ялган авахдаа анх танисан хүснэгтийг маягтын оролтын зурагны файлын нэр дээр тодорхойлогдсон мөр, баганы тоогоор тэнцүү хуваан авч боловсруулж байгаа юм.
2. Мөр, баганы тоог маягтын файлын нэрэнд хавсаргасан байх — Програм маягтыг уншаад боловсруулалт хийн тэмдэгтүүдийг олохдоо хүснэгтээс файлын нэрэнд хавсаргасан мөр, баганыг тоог ашиглаж байгаа.
3. Хүснэгтэд байгаа тэмдэгтүүдийн дараалал зүүнээс баруун, дээрээс доош чиглэлд байх — Боловсруулалт амжилттай хийгдэж тэмдэгт бүрийг олсон хэдий ч аль нүд яг ямар тэмдэгтийг төлөөлж байгаа гэдгийг мэдэхийн тулд маягт бүр дээр энэ дараалал тодорхой байх ёстой.

Дээрх шаардлагуудад нийцүүлэн санг үүсгэхдээ ашиглах эцсийн хувилбар, гар бичмэлийн маягтын жишээг Зураг 2.3-аас харж болно.



Хэрэглээний шинжлэх ухаан, Инженерчлэлийн сургууль

NUMiner гар бичмэлийн маягт

0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
А	Б	В	Г	Д	Е	Ё	Ж	З	И
А	Б	В	Г	Д	Е	Ё	Ж	З	И
Й	К	Л	М	Н	О	Ө	П	Р	С
Й	К	Л	М	Н	О	Ө	П	Р	С
Т	У	Ү	Ф	Х	Ц	Ч	Ш	Щ	Э
Т	У	Ү	Ф	Х	Ц	Ч	Ш	Щ	Э
Ю	Я	а	б	в	г	д	е	ё	ж
Ю	Я	а	б	в	г	д	е	ё	ж
з	и	й	к	л	м	н	о	ө	р
з	и	й	к	л	м	н	о	ө	р
р	с	т	у	ү	ф	х	ц	ч	ш
р	с	т	у	ү	ф	х	ц	ч	ш
ш	ъ	ь	э	ю	я	№	@	#	\$
ш	ъ	ь	э	ю	я	№	@	#	\$
%	&	*	()	;	:	?	>	<
%	&	*	()	;	:	?	>	<
!	?	+	-	\	/	>	<	~	=
!	?	+	-	\	/	>	<	~	=

<https://github.com/khasbileg/numiner>

Зураг 2.3: Гар бичмэлийн маягтын хамгийн эцсийн хувилбар

2.2 Боловсруулалт

Гар бичмэлийн маягтыг бөглүүлэн авсны дараа тухайн маягтын зураг нь оролтын зураг болно. Оролтын зураг дээр тоон зураг боловсруулалт¹ -ын аргууд ашиглан тэмдэгт бүрт харгалзах гараар бичигдсэн хэсгүүдийг олох, бичвэр тус бүрт мөн тодорхой боловсруулалт дахин хийх ба ерөнхий байдлаар дараах алхамуудыг гүйцэтгэнэ. Үүнд:

- Маягтын зургийг хүн бөглөсөн даруйд сканнердах боломжгүй учир тоон зураг болгохын тулд маягтын зургийг нь дарж оруулж байгаа. Иймээс оролтын зураг тодорхой хэмжээнд шуугиантай, гэрлийн нөлөөлөлд автсан байх магадлалтай хэдий ч

¹Компьютерийн шинжлэх ухаанд, зураг боловсруулалт гэдэг нь тоон хэлбэрийн зургийг, цахим тооцоолуурын тусламжтай тодорхой алгоритмын хүрээнд боловсруулахыг хэлнэ. [8]

бэлдсэн маягтаас, тодорхой шаардлагын дагуу маягтын зургийг аль болох шуугиан багатай, гэрлийн нөлөөгүй авч байгаа. Гэсэн ч энэ нь үр дүн ямар байхад сөргөөр нөлөөлөх учир оролтын зурагнаас шуугиан арилгах, гэрлийн нөлөөг багасгах

- Маягт доторх мэдээллүүдээс бидэнд хамгийн хэрэгтэй хэсэг нь тэмдэгтүүд болон бичвэрүүдийг агуулж буй хүснэгт бөгөөд энэхүү хүснэгтийг оролтын зурагны аль хэсэгт байгааг олж мэдэх зорилгоор дөрвөн булангуудыг оролтын зураг дээрээс олох
- Маягтыг тоон болгон хөрвүүлэхдээ дийлэнхи тохиолдолд гар утас камерыг ашиглана гэж үзэж байгаа учир өмнөх шатанд илрүүлсэн бичвэр агуулсан хүснэгт нь зураг дээр эгц, тэгш буух нь эргэлзээтэй. Тийм ч учраас оролтын зургаас харах өнцгийн хувиргалтыг арилган, тэгшлэх
- Өмнөх шатны үр дүнгээс тэмдэгт болон тухайн тэмдэгтийн гараар бичигдсэн хэсгийг хүснэгтийн мөр, баганы тооноос ялган авах, нүд бүрээс ангилалтай (= label) хэсгийг салгах
- Гараар бичигдсэн тэмдэгт тус бүрд гаралтад тодорхойлогдсон шаардлагуудын дагуу дахин боловсруулалт хийх

2.3 Гаралт

Энэхүү ажлын 1.2 хэсгийг үндэслэн гараар бөглөгдсөн маягтаас ялгасан тэмдэгтүүдийг EMNIST сангийн жишгээр боловсруулахаар шийдсэн.

- Зураг нь хоёртын буюу binary зураг байх
- Үндсэн тэмдэгтийг багтаасан хамгийн жижиг тэгш өнцөгт авч, 2 пикселийн хүрээ нэмсэн 28x28 пиксел хэмжээтэй байх

3. ХӨГЖҮҮЛЭЛТ

Өөрийн тодорхойлсон шаардлага, аргын дагуу хөгжүүлэлтийг хийхдээ дараах технологиудыг авч ашигласан.

- OpenCV [9] — Анх 2000 онд танилцуулагдсан энэхүү нээлттэй эхийн сан нь өнөөдөр компьютерийн хараа¹ -тай холбоотой ажил, судалгаануудад хамгийн өргөн ашиглагддаг бөгөөд зураг боловсруулалтанд энэхүү сангийн санал болгосон функц, аргуудыг ашиглана.
- Python [10] — OpenCV санг Python болон C++ хэл дээр ашиглах боломжтой байдаг бөгөөд зураг боловсруулах програм нь хөгжүүлэлтийн болон нийтэд нээлттэй тавих боломжийг үндэслэн Python хэлийг сонгон ашиглана.
- PyPi² — Зураг боловсруулж буй программаа энэхүү платформ дээр байршуулах ба хүссэн хэн нь ч Python багц удирдагч ашиглан амархан татан авах, ашиглах боломжтой болно.

3.1 Оролтын зураг боловсруулалт

Оролтын зургаас үсэгт тэмдэгтүүдийг ялгахын тулд эхлээд зургаас шуугиан арилгах, гэрлийн нөлөөг багасгах, зургийг хоёртын болгон хувиргах гэх мэт урьдчилсан боловсруулалт хийсны үр дүнд хар цагаан зургаас ирмэг илрүүлэх боломжтой болдог.

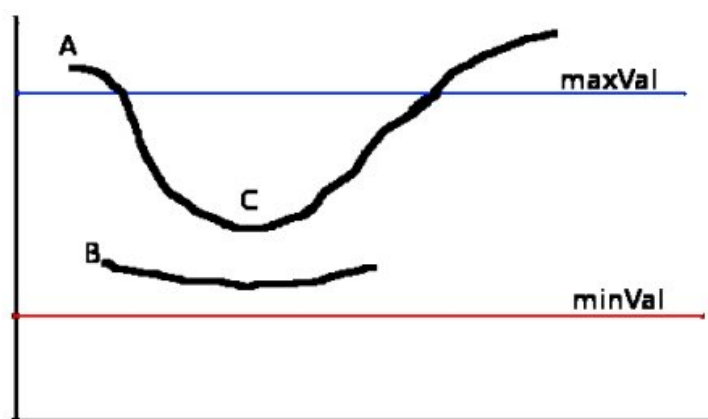
John F. Canny -н ирмэг танилтын арга нь өөрөө зургаас шуугиан арилгах, гэрлийн нөлөөллийг багасгах, хар цагаан зураг болгон хувиргах зэрэг урьдчилсан боловсруулалтын аргуудыг гүйцэтгэдэг бөгөөд дараах үе шатуудтай. Үүнд:

1. Шуугиан арилгах — Оролтын зургийн шуугиан нь ирмэг илрүүлэхэд шууд нөлөөлөх учир илрүүлэлтийн хамгийн эхний хэсэгт 5x5 хэмжээтэй Гаусс -н шүүлтүүрийг ашигладаг.

¹Компьютерийн хараа (= Computer Vision)

²The Python Package Index (PyPI) — <https://pypi.org/>

2. Зургийн градиент олох — Зургийн градиент гэдэг нь зургийн өнгө, пикселүүдийн идэвхжил нь чиглэлийн хувьд хэрхэн өөрчлөгдөж байгааг хэлдэг ба энэ хэсэгт Sobel kernel³ ашиглан босоо, хэвтээ хоёр тэнхлэгийн градиентыг тооцоолдог.
3. Non-maximum Suppression — Өмнөх шатны гаралт дотроос ирмэг гэж үзэж буй пикселүүдийг дараагийн шатны боловсруулалтанд оруулах ба бусдыг нь 0 гэж тооцно. Цэгийн ирмэг эсэхийг тодорхойлохдоо зургийн градиент болон дотоод өндрийг (local maximum) тооцон мөн бол ирмэг, үгүй бол ирмэг гэж үзэхгүй.
4. Hysteresis Thresholding — Энэ хэсэг нь өмнөх шатны ирмэг гэж үзсэн цэгүүд дотор дахин хоёр утгаар зааглалт хийдэг. Өнгөний өөрчлөлттэй ирмэгийн цэгүүдийн утга дээд заагийн утгыг давж байвал баталгаатай ирмэг, доод заагийг давж чадахгүй цэгүүд ирмэг биш. Харин заагийн голд орших цэгүүд нь ямар нэг дээд зааг давсан цэгтэй холбоотой бол ирмэг, үгүй бол мөн ирмэг гэж тооцогдохгүй. Зураг 3.1.



Зураг 3.1: Hysteresis Thresholding -

https://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html -аас зургийг авав.

Өөрөө хоёртын зураг болгож хувиргах, шуугиан арилгах, пикселүүдийг зааглах дарааллаар хар цагаан, зөвхөн цааш боловсруулалтанд чухал хэрэгтэй хэсгүүдийг оролтын зургаас ялгах нь John F. Canny -н ирмэг илрүүлэлтийн аргаас дутмаг болж байсан учир Canny -н ирмэг илрүүлэх аргыг сонгосон. Зураг 3.2a.

³https://en.wikipedia.org/wiki/Sobel_operator

Үүний дараа ирмэг илрүүлсэн зургаас маягтын хамгийн чухал хэсэг болох бичвэрүүдийг хадгалж буй хүснэгтийг олох ёстой ба маягтын зургийг эгц дээрээс авч чадаагүй байх магадлалтай учир харах өнцгийн хувиргалтыг арилган, хүснэгтийг маягтын зурагнаас зүсэж авсан. Зураг 3.2.

Ингэхдээ хамгийн эхлээд оролтын зураг дээрээс `contour` буюу өнгө, идэвхжил ойролцоо хоорондоо холбоотой бүлэг муруйг `cv.findContours()` функцын тусламжтайгаар олж, олдсон контор цэгүүд дотроос хамгийн урт хүрээний урттай бүлэг буюу 3.1 -аас булангийн дөрвөн цэгүүдийн вектор утгуудыг олсон. Зураг 3.2b.

```
1 biggest_contour = sorted(  
2     contours,  
3     key=lambda contour: cv.arcLength(contour, True),  
4     reverse=True  
5 ) [0]
```

Код 3.1: Хамгийн урт контор цэгүүд буюу энэ тохиолдолд бичвэр хадгалж буй хүснэгтийн ирмэг дээрх цэгүүд

Код 3.1 -н үр дүнд хүснэгтийн ирмэгүүд дээрх цэгүүдийн байршлыг олж авах бөгөөд эдгээрээс хамгийн захын дөрвөн цэг буюу яг зураг дээрх хүснэгтийн булангийн дөрвөн цэгийн байршлыг олох хэрэгтэй. Үүний тулд эхлээд `cv.approxPolyDP()` функцыг ашиглан `biggest_contour` -н утгуудыг багасгасан. Энэхүү функц нь оролтын цэгүүд дотроос `epsilon` -д заасан утгаас бага зайнд орших цэгүүдийг тоймлон нэг шулуун болгодог функц юм. Хүснэгтийн булангийн дөрвөн цэгүүдийн олохын тулд `cv.approxPolyDP()` функцын үүсгэсэн цэгүүд бүрээс x , y -н утгуудын нийлбэр нь хамгийн их болон хамгийн бага утгуудыг олсон ба нийлбэр нь хамгийн бага байна гэдэг нь хүснэгтийн хамгийн зүүн дээд булангийн цэг, нийлбэр нь хамгийн их цэг нь хүснэгтийн хамгийн баруун доод цэг болно. Харин x - y буюу ялгавраар нь эрэмбэлсэн үед хамгийн их нь баруун дээд булангийн цэг, хамгийн бага нь зүүн доод булангийн цэг тус тус болно. Код 3.2.

```
1 points = tuple(  
2     (  

```

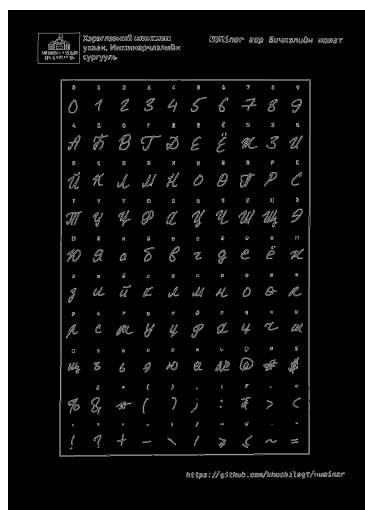
```

3         point[0][0] + point[0][1],
4         point[0][0] - point[0][1],
5         (point[0][0], point[0][1]),
6     )
7     for point in approx_curve
8 )
9 sum_sorted_points = sorted(
10     points, key=lambda point: point[0]
11 )
12 diff_sorted_points = sorted(
13     points, key=lambda point: point[1]
14 )
15
16 extreme_points = (
17     sum_sorted_points[0][-1],
18     diff_sorted_points[-1][-1],
19     diff_sorted_points[0][-1],
20     sum_sorted_points[-1][-1],
21 )

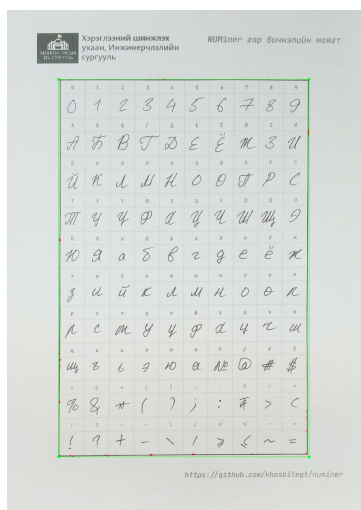
```

Код 3.2: Тоймлосон цэгүүдээс хүснэгтийн булангийн цэгүүдийг олох

Маягт боловсруулалтын сүүлийн хэсэгт бичвэр агуулж буй хүснэгтийн булангийн дөрвөн цэгийн утгуудыг ашиглан харах өнцгийн хувиргалтыг (deskew) арилгах ёстой. Хэрэв оролтын зургийн эгц дээрээс нь авсан тэгш хувилбарыг $M_{original}$ гэж үзвэл оролтын зургийг $M_{input} = A \cdot M_{original}$ гэж үзэж болно. Харин өнцгийн хувиргалтыг арилгахын тулд оролтын зураг болох M_{input} -г A^{-1} буюу хувиргалтын функцын урвуугаар үржүүлбэл эгц, тэгш зураг $M_{original} = A \cdot A^{-1} \cdot M_{input}$ олох боломжтой. Гэвч эндээс мэдэгдэж байгаа нь оролтын зураг дээрх хүснэгтийн цэгүүд болон эгц дээрээс нь авсан үед хаана байх ёстой цэгүүд бөгөөд OpenCV-н `cv.findHomography()` функц нь параметртаа авах хоёр цэгийн хувиргалтын матрицыг олдог. Үүнийг ашиглан оролтын зургийн харах өнцгийн



(a)



(b)



(c)

Зураг 3.2: (a). Маягтын зураг дээр доод заагийг 100, дээд заагийг 200 утгаар Canny -н ирмэг илрүүлэх аргыг ашигласан үр дүн, (b). Илрүүлсэн ирмэгүүдээс контор цэгүүдийг олж, хамгийн урт хүрээний урттай контор цэгүүдийг багтаасан хамгийн жижиг тэгш өнцөгт олсон байдал, (c). Хамгийн том контор цэгүүдээс захын дөрвөн цэгийн утгуудыг олж харах өнцөгийн хувиргалтыг арилгасан байдал

хувиргалтыг арилган (Код 3.3) дараагийн шатанд тэмдэгтүүдийг ялгахад бэлэн болгосон.

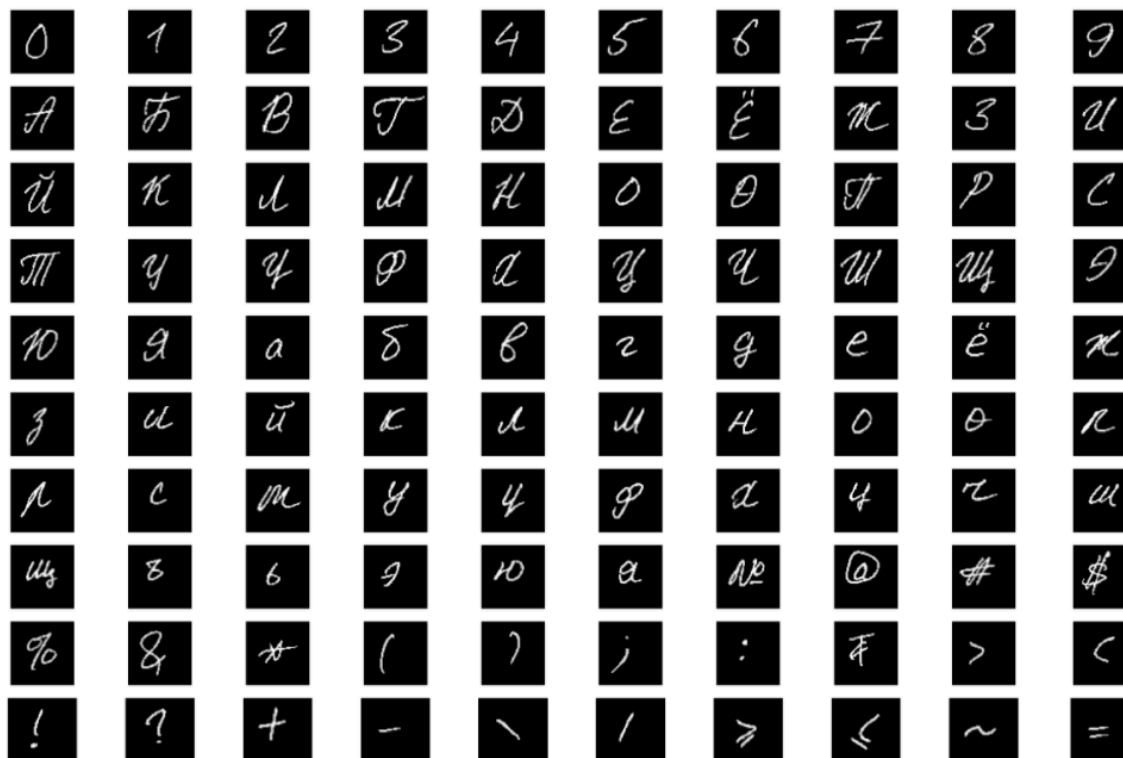
Зураг 3.2с.

```
1 homog, _ = cv.findHomography(
2     np.array(extreme_points), np.array(bounding_rect_points),
3 )
4 transformed = cv.warpPerspective(
5     source,
6     homog,
7     (int(source.shape[1]), int(source.shape[0])),
8 )
```

Код 3.3: Харах өнцгийн хувиргалтыг арилгах

3.2 Тэмдэгтүүдийг хүснэгтээс салгах

Оролтын зураг боловсруулагдаж дуусаад дан бичвэр агуулсан хүснэгтийг илрүүлсэний дараа хүснэгтийн мөр, баганы тоогоор нүд болгоныг ялган авсан ба ингэхдээ гараар бөглөх хэсгийн дээд талын тэмдэгтийн нэрүүд (label) агуулсан хэсгүүдийг оруулахгүй байхаар гүйцэтгэсэн. Зураг 3.3.



Зураг 3.3: Оролтын зураг боловсруулалтын үр дүнгээс гараар бичигдсэн тэмдэгтүүдийг нэг нэгээр нь ялган авсан байдал

Шаардлага №3 -т нийцүүлэх зорилгоор маягт доторх тэмдэгтүүд өөрчлөгдөх эсвэл маяг тэр чигтээ өөрчлөгдөх тохиолдолд хүснэгтийн бүх нүдийг дүүргэх шаардлагагүй ба оролтын маягтыг өгөхдөө тодорхойлох тэмдэгтүүдийн дараалалд `skip_char` байгаа эсэхийг шалгаж хэрэв байгаа бол тэр тэмдэгттэй харгалзах нүдийг боловсруулалтанд хэрэгсэхгүй. Үүний жишээг Зураг 2.1a -аас харж болох ба маягтын тэмдэгтүүдийн дараалал (`label_seq`) Код 3.4 болно. `skip_char = "*" тухайн тэмдэгттэй харгалзах нүдийг гаралтанд тооцохгүй.`

```

1 labelSeq = ('0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
2            '№', '№', '*', '*', '*', '*', '*', '*', '*', '*',
3            'А', 'В', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'З', 'И',
4            'Й', 'К', 'Л', 'М', 'Н', 'О', 'Ө', 'П', 'Р', 'С',
5            'Т', 'У', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Э',
6            'Ю', 'Я', '*', '*', '*', '*', '*', '*', '*', '*',
7            'а', 'б', 'в', 'г', 'д', 'е', 'ё', 'ж', 'з', 'и',
8            'й', 'к', 'л', 'м', 'н', 'о', 'ө', 'п', 'р', 'с',
9            'т', 'у', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ',
10           'ы', 'ь', 'э', 'ю', 'я', '*', '*', '*', '*', '*')

```

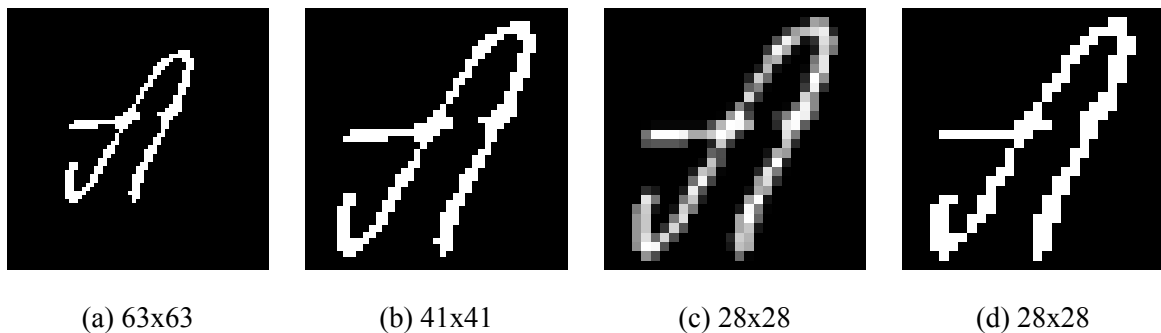
Код 3.4: Хамгийн эхний маягтын хувилбарын нүднүүд, тэдгээрт харгалзах тэмдэгтүүдийн дараалал

3.3 Гаралт

Хүснэгтээс ялган авсан зураг бүр дээр дахин боловсруулалт хийх ба үр дүнд хоёртын (binary), 28x28 хэмжээтэй зураг үүсэх ёстой. Энэ шатны оролтын зураг нь хүснэгтээс ялган авсан нэг тэмдэгтийн зураг байх бөгөөд тэдгээр зургууд гаралтын зургийн байх ёстой хэмжээнээс хамаагүй том, дээр нь харьцаанууд нь харилцан адилгүй байх магадлалтай.

Эхний алхамд гар бичмэлийг бөглөсөн хүн тэмдэгтүүдийг тухайн нүдний аль нэг хэсэгт маш жижиг хэмжээтэйгээр эсвэл хангалттай дүүргэж ямар ч байдлаар бичиж болох учраас оролтын зургийг шууд гаралтын зургийн хэмжээнд тааруулан хэмжээг нь багасгаж болохгүй. Тийм учраас оролтын зураг дээр дахин тоон дүрс боловсруулалт хийх ёстой. Оролтын зурагт тэмдэгттэй ямар ч холбоогүй ямар нэг пиксел, шуугиан байх магадлалтай учир 5x5 хэмжээтэй `cv.GaussianBlur()` ашигласан ба `cv.adaptiveThreshold()` -аар пикселүүдийг зааглан хоёртын зургийг гарган авсан. 3.4a Оролтын зургийн пикселийн утгууд бүдэг, тод ямар ч байх боломжтой учир заагийн утга нь оролтоос хамааран динамик байдлаар тодорхойлогддог байх ёстой.

Урьдчилсан боловсруулалт хийгдсэн зургаас онцлог шинжүүдийг задлан авах (Feature



Зураг 3.4: (a). Хүснэгтээс салган авсан А үсэг, (b). Зургаас контор цэгүүдийг олж бүгдийг нь багтаах хамгийн жижиг тэгш өнцөгт олж харьцааг нь алдагдуулалгүйгээр талуудыг тэнцүүлсэн байдал, (c). Талыг нь тэнцүүлсэн зургийг 28x28 болгож багасгасан байдал, (d). Зурагны пикселүүдийг Otsu -н зааглалтын аргыг ашиглан хоёртын зураг болгон хувиргасан ба гаралтанд бэлэн болсон зураг

extraction) буюу энэ тохиолдолд хүний гараар бөглөсөн тэмдэгтийн хэсгийг оролтын зургаас илрүүлэх хэрэгтэй. Үүний тулд мөн `cv.findContours()` функцийг ашигласан ба хүснэгтийн контор цэгүүд дээр боловсруулалт хийж байснаас ялгаатай нь тэмдэгтээс олдсон бүх контор цэгүүдийг багтаасан хамгийн жижиг тэгш өнцөгт (bounding rectangle) -г олж тооцоолж зөвхөн тэр хэсгийг зургаас тасдан авна. Код 3.5.

```

1 contours, _ = cv.findContours(
2     binary, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE
3 )
4 sorted_contours = sorted(
5     contours,
6     key=lambda ctr: cv.contourArea(ctr),
7     reverse=True,
8 )
9
10 point_x = []
11 point_y = []
12
13 for x, y, w, h in (
```

```

14     cv.boundingRect(ctr) for ctr in sorted_contours
15 ):
16     point_x.append(x)
17     point_x.append(x + w)
18     point_y.append(y + h)
19     point_y.append(y)
20
21 points = list(zip(sorted(point_x), sorted(point_y)))
22
23 max_col = points[-1][0]
24 max_row = points[-1][1]
25 min_col = points[0][0]
26 min_row = points[0][1]
27
28 w_a = max_col - min_col
29 h_a = max_row - min_row
30
31 if w_a > h_a:
32     diff_x = 0
33     diff_y = (w_a - h_a) // 2
34 elif w_a < h_a:
35     diff_x = (h_a - w_a) // 2
36     diff_y = 0
37 else:
38     diff_x = 0
39     diff_y = 0
40
41 diff_x += cls._BORDER_THICKNESS # _BORDER_THICKNESS: 2
42 diff_y += cls._BORDER_THICKNESS # _BORDER_THICKNESS: 2
43

```

```

44 cropped = binary[min_row:max_row, min_col:max_col]
45 cropped = cv.copyMakeBorder(
46     cropped,
47     diff_y,
48     diff_y,
49     diff_x,
50     diff_x,
51     borderType=cv.BORDER_CONSTANT,
52     value=[0, 0, 0],
53 )

```

Код 3.5: Тэмдэгтээс контор цэгүүдийг олж бүгдийг нь багтаасан хамгийн жижиг тэгш
өнцөгтөөр тасдан авах

Үүссэн тэгш өнцөгтийн хэмжээ нь гараар бичигдсэн тэмдэгтээс хамааран ямар ч байж болох учир зургийг гаралтын хэмжээтэй болгож жижигрүүлэхээс өмнө талуудыг адил буюу, тэгш дөрвөлжин зураг болгон хувиргах шаардлагатай. Үүнээс гадна EMNIST -н зураг боловсруулалтын үйл явцад үндсэн тэмдэгтийн зургийн талуудыг харьцааг нь алдагдуулалгүйгээр адилхан болгосны дараа тэмдэгтийг ирмэгт наалдуулахгүйн тулд 2 пиксел хүрээ нэмж өгч байсныг (Зураг 1.3) давхар гүйцэтгэснийг Код 3.5 -оос мөн харж болно. Зураг 3.4b.

Ингээд талуудыг нь адилтгасан мөн 2 пиксел хүрээ нэмсэн зургаа `cv.resize()` функц ашиглан жижигрүүлж үүссэн зурагт дахин `thresholding` хийснээр гаралтын зураг (Зураг 3.4d) боловсруулагдаж дуусах ба нэг тэмдэгтийн зургийн боловсруулалт хэрхэн хийгдсэнийг Хавсралт D -ээс үзнэ үү.

4. ПРОГРАМ ХАНГАМЖ

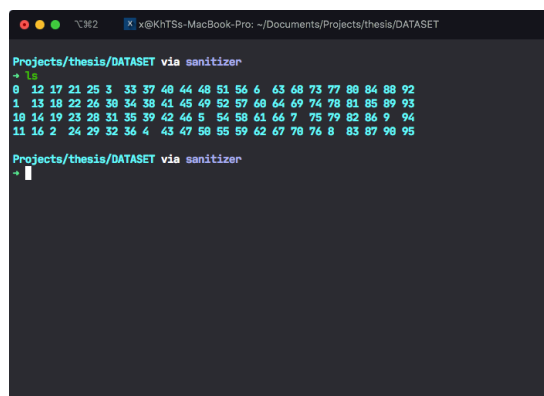
Энэ бүлэгт өөрийн ажлын судалгаан дээр үндэслэн өгөгдөл боловсруулах, сан үүсгэхээр хөгжүүлсэн програмууд болон тэдгээрийг хэрхэн ашиглах талаар авч үзнэ. Энэхүү ажил нь програмтай судалгаа буюу судалгаа голлосон, шаардлагтай байдлаар туслах програмуудыг хөгжүүлсэн учир удирдагч багшийн зөвлөснөөр шинжилгээ, зохиомж шаардлагагүйгээр CLI буюу команд мөр интерфэйстэйгээр хийж гүйцэтгэсэн.

4.1 Sanitizer

Энэхүү жижиг програм нь маягтаас боловсруулалт хийн салган авсан тэмдэгтүүд давхардсан эсэхийг зурган файлын hash -г тооцоолон (Код 4.1) цэвэрлэдэг бөгөөд маш олон тооны зургууд дээр боловсруулалт хийж байгаа учир тооцооллын хугацааг бага байлгахын тулд Python 3 [10] дээр нэмэгдсэн асинхрон тооцооллын `concurrent.futures` санг ашигласан.



(a) Оролт



(b) Гаралт

Зураг 4.1: Sanitizer -г ашиглан нийт 97 тэмдэгт бүхий нийт 255576 зургаас давхардсанг ялгах үеийн оролт болон гаралт

```
1 def fingerprint(cls, filename):  
2     try:  
3         return (
```

```

4         cls.LABELS[
5             str(filename.parent.stem).split("/")[-1]
6         ][ "id" ],
7         filename,
8         hashlib.md5(open(filename, "rb").read()).hexdigest(),
9     )
10 except KeyError:
11     print(f"Ignored: {str(filename.parent.stem).split('/')[-1]}")

```

Код 4.1: Параметраар өгөгдсөн файлын нэрээр зургийг уншин тэмдэгтийн түлхүүр (label), зургийн нэр, hash утгыг агуулсан tuple буцаах функц

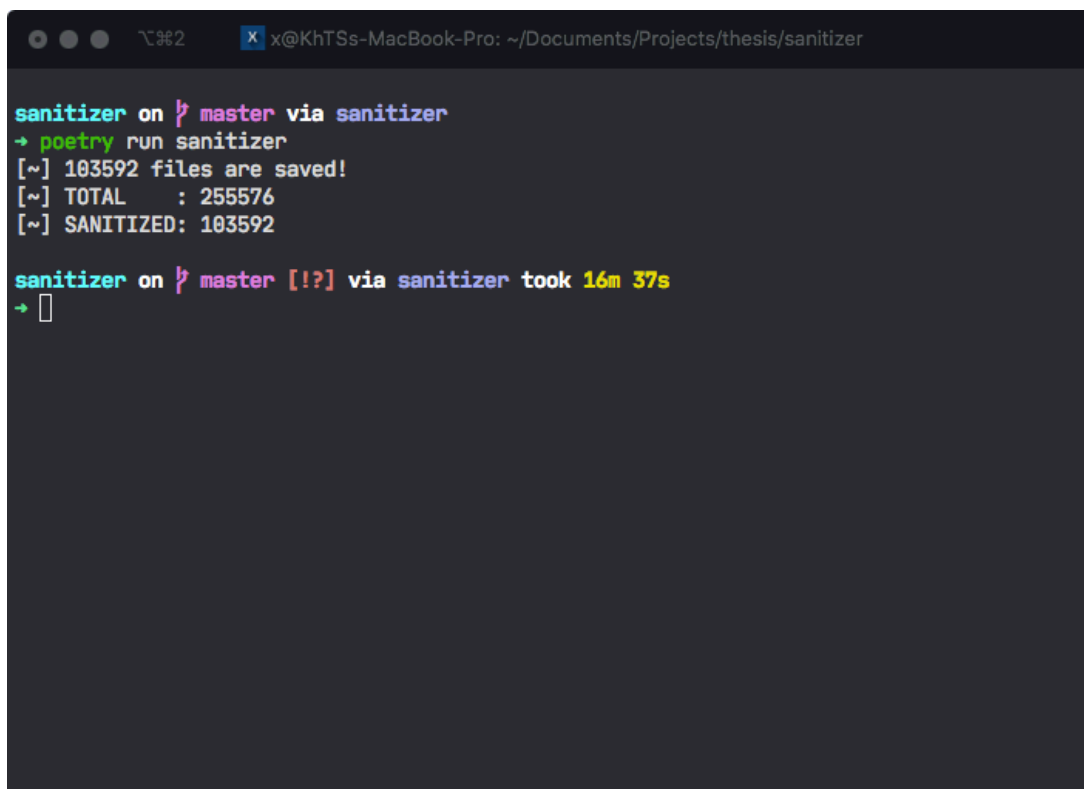
Удирдагч багшийн “Гинжин кодын аргаар Монгол хэлний гар бичмэл танилт” [11] судалгааны ажлын хүрээнд цуглуулсан хүмүүсийн гар бичмэлээс ялган ангилсан 97 ялгаатай тэмдэгтүүдээс бүрдэх нийт 255576 ширхэг зураг бүхий өгөгдлийг (Зураг 4.1a) боловсруулан ашиглахдаа эхлээд давхардсан зургуудыг арилгах шаардлагатай болсон ба үр дүнд давхардалгүй 103592 зургийг харгалзах тэмдэгтийн хавтас бүрт хадгалсан (Зураг 4.1b) байдлыг Зураг 4.2 -ээс харж болно. Үр дүнд үүссэн хавтаснууд (Зураг 4.1b) болон тэдгээрийн нэр нь `config.json`¹ болон `labels.json`² -оос хамаарах ба өгөгдөл нь тусгай тэмдэгтүүдийг ч мөн агуулж буй учир тэмдэгтийн түлхүүрээр хавтсыг нэрлэх боломжгүй. Тийм учраас тэмдэгтийн түлхүүрийг 0 -ээс эхэлсэн бүхэл тоогоор дугаарласан бөгөөд ямар дугаар ямар тэмдэгттэй харгалзаж буй мэдээллийг дээрх JSON файлуудад хадгалсан.

4.2 NUMiner

NUMiner нь гараар бөглөгдсөн маягтын зургийг боловсруулахаас тэмдэгтүүдийг ялган авч зураг болгоод MNIST -тэй адил IDX файл форматтай болгон хувиргах хүртэлх бүх боловсруулалтуудыг гүйцэтгэж байгаа бөгөөд *Шаардлага №5* -д тодорхойлсоны да-

¹<https://github.com/khasbilegt/sanitizer/blob/master/config.json>

²<https://github.com/khasbilegt/sanitizer/blob/master/labels.json>

A terminal window on a Mac with the title 'x@KHTSs-MacBook-Pro: ~/Documents/Projects/thesis/sanitizer'. The terminal shows the output of a 'sanitizer' command. It reports that 103592 files are saved, with a total of 255576 files and 103592 files sanitized. It also indicates that the sanitizer took 16 minutes and 37 seconds to complete. The prompt is a green arrow pointing to a space character.

```
sanitizer on  master via sanitizer
→ poetry run sanitizer
[~] 103592 files are saved!
[~] TOTAL    : 255576
[~] SANITIZED: 103592

sanitizer on  master [!?] via sanitizer took 16m 37s
→ 
```

Зураг 4.2: Sanitizer програмын жишээ үр дүн

гуу програмыг PyPi³ дээр байрлуулсан. Ингэснээр ямар ч хүн pip⁴, pipenv⁵, poetry⁶ гэх мэт Python хэлний ямар ч багц удирдлагын програмыг ашиглан амархан авч ашиглах боломжтой.

Энэхүү програм нь CLI буюу команд мөр интерфэйстэй учир системийн терминал эсвэл түүнтэй холбоотой програмыг ашиглан ажиллуулна. Нийтлэг ашиглах тохиолдол нь тодорхой нэг маягтыг оролтонд өгч боловсруулах эсвэл олон тооны маягтуудыг агуулж буй хавтасны замыг оролтонд өгч боловсруулалт хийх байх бөгөөд `-s`, `--sheet` командруудыг дараах байдлаар ашиглана.

```
$ numiner -s data/INPUT/SHEET_10_10_00001.jpg data/OUTPUT
```

Код 4.2: NUMiner — нэг маягтыг боловсруулах команд

³NUMiner — <https://pypi.org/project/numiner>

⁴Pip — <https://github.com/pypa/pip>

⁵Pipenv — <https://github.com/pypa/pipenv>

⁶Poetry — <https://python-poetry.org/>

```
$ numiner -s data/INPUT data/OUTPUT
```

Код 4.3: NUMiner — олон маягтуудыг нэг дор боловсруулах команд буюу маягтуудыг агуулж буй хавтсын замыг зааж өгөх

Код 4.3 -н үр дүнд data/INPUT хавтас дотроос PNG, JPG, JPEG зургуудыг олж, файлын нэр нь SHEET гэж эхэлсэн, хүснэгтийн мөр болон баганы тоог, мөн дугаар оруулсан байгаа эсэхийг шалган

```
SHEET_<row_count>_<col_count>_<sheet_id>.<extension>
```

цааш боловсруулалтыг гүйцэтгэнэ. Боловсруулалтын үр дүн нь data/OUTPUT хавтас дотор тэмдэгт бүр өөрийн харгалзах дугаартай хавтас дотор

```
<sheet_id>_<label_id>_<datetime>.<extension>
```

нэртэйгээр үүснэ.

Код 4.2 -н ажиллагаа Код 4.3 -тэй адилхан бөгөөд ганц ялгаа нь оролтын зам нь хавтасных бус яг нэг маягтын зургийг заасан зам учир гаралтанд зөвхөн тухайн маягтын тэмдэгтүүд л боловсруулагдсан байна.

Энэхүү бүлгийн эхний хэсэг 4.1 -т ашигласантай адилаар магт нь тусгай тэмдэгтүүд агуулж байгаа учир хавтсыг тэмдэгтийн түлхүүрээр хадгалах боломжгүй юм. Маягтын хүснэгтийн нүд бүрийг тодорхой дугаартай харгалзуулан гаралтын үр дүнд харгалзах дугаараар хавтсуудыг үүсгэх ба энэ програм энэ мэдээллийг --labels командыг ашиглан тэмдэгтийн түлхүүр болон харгалзах дугаарыг агуулах тохиргооны JSON файлаас дараах байдлаар унших боломжтой. Код 4.4. JSON файл доторх тохиргооны зохион байгуулалтыг Код 4.5 -аас харж болох бөгөөд NUMiner -н файлыг уншиж тохиргоог хийж байгаа кодыг Хавсралт E -с харж болно.

```
$ numiner --labels config/labels.json -s data/INPUT data/OUTPUT
```

Код 4.4: NUMiner — config хавтас доторх labels.json нэртэй файлаас тэмдэгтийн түлхүүрүүдийг ямар дугаартай харгалзуулан хадгалахыг уншиж, data/INPUT доторх маягтуудыг боловсруулан үр дүнг data/OUTPUT дотор хадгал

```

1 {
2   "labels": {
3     ...,
4     "91": "!",
5     "92": "?",
6     "93": "+",
7     "94": "-",
8     "95": "/",
9     ...
10  }
11 }

```

Код 4.5: NUMiner — labels.json

Боловсруулалт хийн тэмдэгтийн төрөл бүрт зургаар нь хадгалсны дараа *Шаардлага №4* -н дагуу авч ашиглахад хялбар байх зорилгоор MNIST [3] өгөгдлийн хэлбэртэй адил IDX форматтай болгон хувиргана. Ингэхдээ програмын `convert` командыг ашиглах бөгөөд командын араас оролтын болон гаралтын замууд, нэг классаас авах өгөгдлийн тоо болон сургалт, тестийн өгөгдлийн харьцаа гэх дөрвөн параметруудийг олгон ажиллуулах буюу дараах хэлбэртэй байна. Код 4.6.

```
$ numiner convert <src> <dst> <size> <ratio>
```

Код 4.6: NUMiner `convert` — `<src>` доторх тэмдэгтийн төрөл бүрээс `<size>` ширхэг зургийг авах ба нийт өгөгдлөөс `<ratio>` хувийг нь (хэрэв `ratio` бүхэл тоо бол нийт өгөгдлөөс салган авах тестийн өгөгдлийн хувь, `train` бол бүхэлдээ сургалтын, `test` бол бүхэлдээ тестийн өгөгдөл гэж тус тус үзнэ) тестийн өгөгдөлд зориулан боловсруулж `<dst>` -д хадгална.

Код 4.6 жишээнд буй `<size>` нь 0 байж болох ба энэ тохиолдолд тэмдэгтийн төрөл бүрээс бүх зургийг нь өгөгдөлд оролцуулна. Харин `<ratio>` буюу тестийн өгөгдлийн харьцаа нь бүхэл тооноос гадна `train`, `test` гэх үгнүүд байж болох ба ингэвэл тэмдэгтийн төрөлд зааж өгсөн өгөгдлийг бүхэлд нь сургалтын эсвэл тестийн болгон боловсруулна.

5. ҮР ДҮН

Машин сургалтанд ашиглах өгөгдлийн тоо хэмжээ аль болох их байх нь сургалтын үр дүн, үзүүлэлтэнд сайнаар нөлөөлдөг. Энэхүү судалгааны ажлыг гүйцэтгэх хугацаанд эхлээд ерөнхий судалгаа хийж ямархуу байдлаар санг үүсгэх, боловсруулалт хийх, боловсруулалт хийх програмыг хөгжүүлэх, засах болон сайжруулах ажлуудыг гүйцэтгэн үлдсэн хугацаанд машин сургалтанд ашиглахад бэлэн болсон эсвэл тодорхой үр дүн харуулах хэмжээний өгөгдөл цуглуулж амжаагүй учир одоогийн байдлаар ямар тэмдэгтүүд тус бүрт хэр их хэмжээний өгөгдөл/зураг цуглуулсан байгааг Зураг 5.1 -д харуулав.

- → 3	; → 13	: → 18	Ё → 124	ё → 299	й → 411	ь → 498	? → 618	Ө → 632	Ч → 679
Ю → 719	(→ 739	/ → 758	№ → 765	З → 766	% → 789	+ → 844	С → 852	К → 853	Ө → 865
< → 885	Н → 893	т → 904	Г → 908) → 915	> → 937	э → 941	ү → 943	Я → 962	1 → 974
л → 981	Т → 996	Э → 1008	г → 1011	М → 1019	& → 1028	Р → 1041	Х → 1049	д → 1051	Щ → 1064
Л → 1072	Ш → 1073	Ч → 1076	ц → 1079	Ц → 1093	х → 1099	р → 1104	п → 1116	б → 1116	ж → 1123
с → 1128	Б → 1138	э → 1146	Ү → 1152	я → 1152	У → 1163	7 → 1167	\$ → 1176	# → 1178	@ → 1181
в → 1189	Ж → 1208	м → 1208	а → 1211	Б → 1240	6 → 1267	2 → 1272	н → 1273	9 → 1314	8 → 1336
В → 1341	Ф → 1352	А → 1394	У → 1402	щ → 1410	4 → 1626	о → 1700	е → 1769	ө → 1803	* → 1812
И → 1822	ю → 1839	0 → 1897	З → 1947	у → 2197	и → 3477	Д → 3559	ш → 5481		

Зураг 5.1: Цугларсан тэмдэгт бүрт харгалзах өгөгдөл/зургийн тоо

Дээрх өгөгдлийн дийлэнхи хувь нь удирдагч багшийн “Гинжин кодын аргаар Монгол хэлний гар бичмэл танилт” [11] судалгааны ажлын хүрээнд цуглуулан ялгасан тэмдэгтийн зургуудийг өөрийн боловсруулалтын аргаар боловсруулан үүсгэсэн өгөгдлөөс бүрдсэн ба цаашид оролтын маягтыг аль болох олон хүнээр бөглүүлэн өгөгдлийн хэмжээг ихэсгэх шаардлагатай.

6. ДҮГНЭЛТ

Энэхүү судалгааны ажлаар бусад хэл дээрх гар бичмэл танилттай холбоотой судалгаануудын ажлууд дээр тулгуурлан өөрийн хэлний гар бичмэл танилтанд зориулсан нээлттэй сан, түүнд ашиглах програм хангамжийг хөгжүүлэхээр зорьлоо. Энэ судалгааны ажил нь цаашид гар бичмэл танилт болон энэ чиглэлийн судалгаа, програм хангамж, үйлчилгээ зэргийг хөгжүүлэхэд суурь болж өгнө гэдэгт итгэлтэй байна.

Нийт өгөгдлийн тоо хэмжээ, төрлүүдийг нэмэх, өргөжүүлэхийн тулд python хэл дээрх програм хангамжийн^{1 2} боломжуудыг тогтмол нэмэх, сайжруулах, бичвэр агуулах маягыг динамик байдлаар үүсгэх, ажиллагааг хурдасгах, нийтээс маягтын зургаа авч автоматаар боловсруулан сангаа шинэчлэх бие даасан вэбсайт, систем гэх мэт ажлууд нэмэгдэж хийх шаардлагатай.

¹Эх код: <https://github.com/khasbilegt/numiner>

²PyPi дээрх сан: <https://pypi.org/project/numiner/>

Ашигласан материал

- [1] NIST Special Database 19

<https://www.nist.gov/srd/nist-special-database-19>

- [2] Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017).

EMNIST: an extension of MNIST to handwritten letters.

<http://arxiv.org/abs/1702.05373>

- [3] MNIST — Wikipedia, The Free Encyclopedia

https://en.wikipedia.org/wiki/MNIST_database

- [4] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner *Gradient-based learning applied to document recognition* in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

- [5] S. Johansson, G.N. Leech, H. Goodluck: *Manual of information to accompany the Lancaster-Oslo/Bergen corpus of British English, for use with digital computers.* Department of English, University of Oslo, Oslo, (1978)

- [6] Marti, U., Bunke, H. *The IAM-database: an English sentence database for offline handwriting recognition.* IJDAR 5, 39–46 (2002). <https://doi.org/10.1007/s100320200071>

- [7] ETLCD — ETL Character Database

<http://etlcdb.db.aist.go.jp/obtaining-etl-character-database>

- [8] Digital image processing — Wikipedia, The Free Encyclopedia

https://en.wikipedia.org/wiki/Digital_image_processing

- [9] Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.
<https://opencv.org/>
- [10] Van Rossum, G., Drake Jr, F. L. (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace.
<http://www.python.org>
- [11] S.Nasan-Ochir, B.Mungunshagai, N.Bayarsaikhan, B.Suvdaa. *Mongolian handwritten recognition using Chain code*. Machine Intelligence Lab, SEAS, NUM.

А. ИРМЭГ ИЛРҮҮЛЭГЧ

```
1 @classmethod
2 def get_edges(cls, source):
3     # _THRESHOLD_MIN: 100, _THRESHOLD_MAX: 200
4     return cv.Canny(source, cls._THRESHOLD_MIN, cls._THRESHOLD_MAX)
```


В. ХҮСНЭГТ ИЛРҮҮРЭГЧ

```
1 @classmethod
2 def get_form_container(cls, source):
3     contours, _ = cv.findContours(
4         cls.get_edges(source),
5         cv.RETR_EXTERNAL,
6         cv.CHAIN_APPROX_SIMPLE,
7     )
8     biggest_contour = sorted(
9         contours,
10        key=lambda contour: cv.arcLength(contour, True),
11        reverse=True,
12    )[0]
13     x, y, w, h = cv.boundingRect(biggest_contour)
14     bounding_rect_points = rectangle2points(x, y, w, h)
15     approx_curve = cv.approxPolyDP(
16         biggest_contour,
17         0.01 * cv.arcLength(biggest_contour, True),
18         True,
19     )
20
21     if len(approx_curve) >= 4:
22         points = tuple(
23             (
24                 point[0][0] + point[0][1],
25                 point[0][0] - point[0][1],
26                 (point[0][0], point[0][1])),
27             for point in approx_curve
28         )
29         sum_sorted_points = sorted(
30             points, key=lambda point: point[0]
31         )
32         diff_sorted_points = sorted(
33             points, key=lambda point: point[1]
34         )
35
36         extreme_points = (
37             sum_sorted_points[0][-1],
38             diff_sorted_points[-1][-1],
39             diff_sorted_points[0][-1],
40             sum_sorted_points[-1][-1],
41         )
42
43         homog, _ = cv.findHomography(
44             np.array(extreme_points),
45             np.array(bounding_rect_points),
46         )
47         transformed = cv.warpPerspective(
48             source,
49             homog,
50             (int(source.shape[1]), int(source.shape[0])),
51         )
52
```

```
53         new_x, new_y, new_w, new_h = cls.compensate(  
54             x, y, w, h, offset=5  
55         )  
56         return transformed[  
57             new_y : new_y + new_h, new_x : new_x + new_w  
58         ]  
59     return None
```

С. ТЭМДЭГТҮҮДИЙГ САЛГАХ

```
1 @classmethod
2 def get_characters(cls, form, label_seq: str, skip_char: str):
3     h, w, _ = form.shape
4     height = (h + (10 - int(str(h)[-1]))) // 10
5     width = (w + (10 - int(str(w)[-1]))) // 10
6     label_height = height - width
7     character_seq = [
8         form[
9             row * height + label_height : row * height + height,
10            col * width : col * width + width,
11        ]
12        for row in range(10)
13        for col in range(10)
14    ]
15
16    return tuple(
17        (label, letter)
18        for label, letter in zip(label_seq, character_seq)
19        if label != skip_char
20    )
```

D. ТЭМДЭГТ БОЛОВСРУУЛАХ

```
1 @classmethod
2 def get_char(cls, label, source):
3     binary = cls.get_threshold(
4         cls.get_blurred_img(cls.get_gray_img(source)),
5         cv.THRESH_BINARY_INV,
6     )
7     contours, _ = cv.findContours(
8         binary, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE
9     )
10    sorted_contours = sorted(
11        contours,
12        key=lambda ctr: cv.contourArea(ctr),
13        reverse=True,
14    )
15    point_x = []
16    point_y = []
17    for x, y, w, h in (
18        cv.boundingRect(ctr) for ctr in sorted_contours
19    ):
20        point_x.append(x)
21        point_x.append(x + w)
22        point_y.append(y + h)
23        point_y.append(y)
24
25    points = list(zip(sorted(point_x), sorted(point_y)))
26
27    if not points:
28        return
29
30    max_col = points[-1][0]
31    max_row = points[-1][1]
32    min_col = points[0][0]
33    min_row = points[0][1]
34
35    w_a = max_col - min_col
36    h_a = max_row - min_row
37
38    if w_a > h_a:
39        diff_x = 0
40        diff_y = (w_a - h_a) // 2
41    elif w_a < h_a:
42        diff_x = (h_a - w_a) // 2
43        diff_y = 0
44    else:
45        diff_x = 0
46        diff_y = 0
47
48    diff_x += cls._BORDER_THICKNESS
49    diff_y += cls._BORDER_THICKNESS
50
51    cropped = binary[min_row:max_row, min_col:max_col]
52    cropped = cv.copyMakeBorder(
```

```

53         cropped,
54         diff_y,
55         diff_y,
56         diff_x,
57         diff_x,
58         borderType=cv.BORDER_CONSTANT,
59         value=[0, 0, 0],
60     )
61
62     if cropped.size != 0:
63         resized = cv.resize(
64             cropped, cls._SIZE, interpolation=cv.INTER_AREA
65         )
66         _, final = cls.get_threshold(
67             resized, cv.THRESH_BINARY + cv.THRESH_OTSU,
68             adaptive=False
69         )
70         return final
71     return

```

E. NUMINER — ТОХИРГООГ УНШИХ

```
1 def handle_config(arg):
2     import json
3     with open(arg) as config_file:
4         config = json.load(config_file)
5         Sheet._CHARACTER_SEQUENCE = tuple(config["labels"].items())
6
7     return Sheet._CHARACTER_SEQUENCE
```

Г ар бичмэл танилганд зориулсан МУИС-н нээлттэй өгөгдөл бэлтгэх нь

Preparation of NUM handwriting open dataset

сэдэвт бакалаврын судалгааны ажлын 7 хоногийн үечилсэн төлөвлөгөө

Хугацаа: 2020.02.10-аас 2020.05.08 хүртэл 13 долоо хоног

№	Хийх ажлг												12	13	Тайлбар
Долоо хоног	1	2	3	4	5	6	7	8	9	10	11	Урьдчилсан хамгаалалт			
	Онолын судалгаа														
1	Судалгаа														
2	Аргаа тодорхойлох	Судалсан аргуудаас давуу талуудыг тодорхойлох													
		Өгөгдөл цуглуулах аргаа тодорхойлох													
3	Өгөгдөл цуглуулах	Судалгааны аргыг турших													
		Жинхэнэ өгөгдөл цуглуулах													
4	Өгөгдлийг бэлдэх	Өгөгдлийг ялгах, цэвэрлэх													
		Өгөгдлийг сургалтанд ашиглахад бэлтгэх													
5	Өгөгдлийг турших	Цуглуулсан өгөгдлөө сургалтанд ашиглах, турших													
		Өгөгдлийн чанар, үр дүнг тодорхойлох, тооцоолох													
6	Тайлан	Явцын													
		Эцсийн													

Тайлбар: Тестийг хэрэгжүүлэх төлөвлөгөөг 7 хоногийн дотооджилаар хийж тоо харвар бүлэж тэмдэглэнэ. Хийх ажлыг дэд хэсэгтэй байвал үг ажилд зарцуулах хугацааг хувиан төлөвлөж болно. Ажлын эхлэх төгсгөл хугацааг хоорондоо давхцаж болно. Ажлын сүртингэлтийг үүсгэж тэмдэглэгээ хийх боломжтой байхад "хоног" басныг үүсгэнэ.