

МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ  
МЭДЭЭЛЛИЙН ТЕХНОЛОГИ, ЭЛЕКТРОНИКИЙН СУРГУУЛЬ  
МЭДЭЭЛЭЛ, КОМПЬЮТЕРЫН УХААНЫ ТЭНХИМ

Гансүхийн Дэлгэрмаа

**Монгол бичиг танилт**  
**(Mongolian script recognition)**

Компьютерын ухаан (D061301)  
Бакалаврын судалгааны ажил

Улаанбаатар

2024 оны 12 сар

МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ  
МЭДЭЭЛЛИЙН ТЕХНОЛОГИ, ЭЛЕКТРОНИКИЙН СУРГУУЛЬ  
МЭДЭЭЛЭЛ, КОМПЬЮТЕРЫН УХААНЫ ТЭНХИМ

Монгол бичиг танилт  
(Mongolian script recognition)

Компьютерын ухаан (D061301)  
Бакалаврын судалгааны ажил

Удирдагч: \_\_\_\_\_ Др. Б.Сувдаа  
Гүйцэтгэсэн: \_\_\_\_\_ Г.Дэлгэрмаа (20B1NUM2595)

Улаанбаатар  
2024 оны 12 сар

# Зохиогчийн баталгаа

Миний бие Гансүхийн Дэлгэрмаа ”Монгол бичиг танилт” 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

УДИРТГАЛ.....	1
1. СЭДВИЙН СУДАЛГАА.....	2
1.1 Монгол бичгийг дэмждэг OCR судалгааны ажлууд .....	2
1.2 Ашиглах технологийн судалгаа .....	4
2. ОНОЛЫН СУДАЛГАА .....	7
2.1 Хиймэл оюун ухаан (AI) гэж юу вэ? .....	7
2.2 Байгалийн Хэлний Боловсруулалт (NLP) гэж юу вэ?.....	9
2.3 Машин сургалт ба гүн сургалт .....	10
2.4 OCR гэж юу вэ?.....	11
2.5 Монгол бичгийн судалгаа .....	12
3. ӨГӨГДЛИЙН ШИНЖИЛГЭЭ .....	14
3.1 Өгөгдлийн багцын танилцуулга .....	14
3.2 Өгөгдлийн шинжилгээ .....	17
3.3 Өгөгдлийн багцын бэлтгэл .....	19
4. АРГА БА АЛГОРИТМ .....	23
4.1 Моделийн Оновчтой Архитектур .....	23
4.2 CNN, BiLSTM, CTC-ийн Үүрэг .....	26
5. ХЭРЭГЖҮҮЛЭЛТ .....	29
5.1 Сургалтын орчин бэлтгэсэн нь.....	29
5.2 Хэрэгжүүлэлт №1 .....	30
5.3 Хэрэгжүүлэлт №2 .....	33
5.4 Үр дүн.....	35
6. КОД .....	37
ДҮГНЭЛТ .....	58
НОМ ЗҮЙ .....	60

## ЗУРГИЙН ЖАГСААЛТ

1.1	Results Table .....	2
2.1	AI .....	7
2.2	NLP.....	9
2.3	ML vs DL .....	10
2.4	OCR .....	11
2.5	Монгол бичгийн бүтэц .....	12
3.1	Урьдчилан боловсруулсан Юникод өгөгдөл .....	15
3.2	Латин үгнүүд болон Графем кодын уялдаа .....	16
3.3	Графемийн кодын тодорхойлолт .....	16
3.4	Хамгийн их давтагдаж буй шошгуудын гистограмм .....	17
3.5	Хамгийн бага давтагдаж буй шошгуудын гистограмм.....	17
3.6	Нийт шошгуудын уртын хамаарлын гистограмм .....	18
3.7	Хамгийн богино 10 шошго ба тэдгээрийн давтамжийн гистограмм ....	18
3.8	Хамгийн урт 10 шошго ба тэдгээрийн давтамжийн гистограмм.....	19
3.9	Координатыг холбож зураг үүсгэх жишээ .....	20
3.10	Render хийгдсэн зургуудын жишээ .....	20
3.11	Render хийгдсэн зургийн жишээ .....	21
3.12	Шошгожуулсан өгөгдлийн багц.....	21
3.13	Боловсруулж дууссан өгөгдлийг хуваан бэлтгэсэн нь .....	22
4.1	Model Architecture .....	23
4.2	CNN + BiLSTM + CTC.....	26
4.3	CNN .....	27
4.4	BiLSTM.....	28
5.1	Идэвхижүүлсэн API-ууд .....	29
5.2	CNN модель.....	30

5.3	CER (Character Error Rate) .....	31
5.4	WER (Word Error Rate) .....	31
5.5	Сургалт болон баталгаажуулалтын алдагдал .....	32
5.6	Тэмдэгтийн алдааны үнэлгээ .....	32
5.7	Үгийн алдааны үнэлгээ .....	33
5.8	Сургалтын Алдагдал .....	35
5.9	Туршилтын үр дүн .....	36
6.1	Шүүмж .....	59

# Кодын жагсаалт

5.1	Cloud Storage-тай холбож өгөгдөл татах нь . . . . .	30
5.2	Зургийг боловсруулах функц . . . . .	34
6.1	Зураг рендер хийх процесс . . . . .	37
6.2	CRNN (Convolutional Recurrent Neural Network) моделийг сургах скрипт . . .	40
6.3	Сургасан моделийг үнэлэх скрипт . . . . .	50
6.4	CRNN моделийн архитектурын тодорхойлолт . . . . .	54

## УДИРТГАЛ

Монгол бичиг нь Монгол үндэстний өв соёл, түүх, бичгийн уламжлалыг хадгалан ирсэн чухал бичиг үсэг юм. Сүүлийн жилүүдэд OCR (Optical Character Recognition) буюу зургаас текст таних технологиуд хурдацтай хөгжиж байгаа боловч Монгол бичгийн хувьд тэдгээрийг шууд ашиглах боломж хязгаарлагдмал байгаа билээ. Монгол бичгийн онцлог нь босоо чиглэл, тасралтгүй холбогдсон тэмдэгтүүдтэй байх ба үгийн сан харьцангуй их учир энэ нь зургаас текст хөрвүүлэх OCR моделиудыг сургахад томоохон сорилт үүсгэдэг шинжүүд юм.

### **Зорилго:**

Энэхүү судалгааны ажлын гол зорилго нь Монгол бичгийн гар бичмэл зургаас текст хөрвүүлэх OCR системд тохиромжтой өгөгдлийн багцыг боловсруулж бэлтгэх, тэрхүү бэлтгэсэн өгөгдлийн багцыг ашиглан анхан шатны OCR моделийг сургаж, турших юм.

### **Зорилт:**

1. OCR технологийн үндэс, холбогдол бүхий онолын болон технологийн судалгааг хийж гүйцэтгэх. Монгол бичигт тохируулах боломжуудыг судлах.
2. Ижил төст судалгааны ажлуудтай танилцах.
3. Монгол бичгийн гар бичмэл зурган өгөгдлийн санг бэлтгэж, боловсруулан , шинжлэх.
4. Монгол бичгийг зургаас танихад тохирсон алгоритм, моделийн архитектурыг сонгон хөгжүүлж анхан шатны интеграци хийж, туршиж, үнэлэх.
5. Судалгааны үр дүнг нэгтгэн тайлан боловсруулж, дүгнэлт гаргах.



# 1. СЭДВИЙН СУДАЛГАА

## 1.1 Монгол бичгийг дэмждэг OCR судалгааны ажлууд

### 1.1.1 *Online Mongolian Handwriting Recognition Based on Encoder–Decoder Structure with Language Model:*

Хэд хэдэн төслөөр гар бичмэл монгол бичгийг таних нейрон сүлжээнд суурилсан моделийг бий болгосон. Эдгээр системүүд нь гараар бичсэн монгол хэлний нарийвчлалыг онцгойлон сайжруулахын тулд хэлний моделиудыг нэгтгэсэн энкодер-декодерийн моделийг ихэвчлэн ашигладаг. Онлайн болон офлайн гараар бичсэн өгөгдлийг зохицуулж, тэмдэгт болон дарааллыг таних чадварыг сайжруулснаараа ач холбогдолтой. Энэхүү судалгаа нь таних нарийвчлалыг нэмэгдүүлэхийн тулд урьдчилан бэлтгэгдсэн хэлний моделийг дараалалаас дараалал (Seq2Seq) + анхаарлын механизм (AM) загварт нэгтгэх замаар урьд өмнө нь байгаагүй арга замыг санал болгосон. Мөн former, latter, complete гэсэн 3 fusion моделийг танилцуулж байгаа нь үндсэн моделиос ихээхэн ахиц гаргаж чадсан байна. [<https://www.mdpi.com/2079-9292/12/20/4194>].

Table 5. Comparison with other studies.				
Model	train_ACEC	train_WER	test_ACEC	test_WER
LSTM-CTC [7]	0.347	21.432%	0.528	30.14%
Seq2Seq + AM [4]	0.234	13.52%	0.473	24.28%
<b>Ours</b>	<b>0.202</b>	<b>10.89%</b>	<b>0.428</b>	<b>21.05%</b>

Bold font indicates optimal results.

Зураг 1.1: Results Table

### 1.1.2 A new dataset for mongolian online handwritten recognition:

Энэхүү нийтлэл нь MOLHW хэмээх монгол хэлний үгийн түвшний онлайн гар бичмэлийн шинэ өгөгдлийн багцыг танилцуулсан байна. Өгөгдлийн багц нь гар бичмэл монгол үгсээс бүрдэх бөгөөд үүнд 200 бичээчийн бичсэн 164,631 түүвэр, 40,605 монгол үг багтана. Үгийн координатын цэгүүдийг цуглуулж, гар утсанд зориулсан тусгай программ дээр харгалзах үгийг бичжээ. Монгол хэлний латин галиг үсгээр үг бүрийн координатыг тэмдэглэсэн. Үүний зэрэгцээ өгөгдлийн багцад бичээчийн ID, гар утасны дэлгэцийн мэдээллийг бүртгэсэн. Энэхүү өгөгдлийн багцыг ашиглан үндсэн үнэлгээний загвар болгон хоёр чиглэлтэй гүн гүнзгий давтагдах нэгж, анхаарлын механизм бүхий кодлогч-декодрийн монгол гар бичмэлийг онлайнаар таних загварыг санал болгосон. Энэ загварын дагуу туршилтын багц дээрх үгийн алдааны түвшин (WER) оновчтой гүйцэтгэл нь 24.281 хувь байсан. Цаашлаад монгол гар бичмэлийг онлайнаар таних янз бүрийн загваруудын туршилтын үр дүнг танилцуулсан. Туршилтын үр дүнгээс харахад Transformer дээр суурилсан загвар нь бусад загваруудтай харьцуулахад өгөгдлийн багцын координатын өгөгдлөөс харгалзах тэмдэгтүүдийн дарааллыг илүү үр дүнтэй сурч, туршилтын багц дээр 16.969 хувь WER-тэй болохыг харуулж байна. Өгөгдлийн багцыг одоо дэлхийн өнцөг булан бүрээс судлаачид чөлөөтэй ашиглах боломжтой. Өгөгдлийн багцыг гараар бичсэн текстийг таних, гараар бичсэн текст үүсгэх, гар бичмэлийг таних, гарын үсгийг таних зэрэгт ашиглаж болно.

**Дүгнэлт :** Хиймэл оюун ухаан болон машин сургалт дээр суурилсан судалгаанууд сүүлийн жилүүдэд хурдацтай хөгжиж байна. Эдгээр системүүд нь ихэвчлэн эрдэм шинжилгээний болон судалгааны ажил бөгөөд энэ нь хяналттай орчинд туршиж, баталгаажуулагдсан гэсэн үг юм. Гэсэн хэдий ч эдгээр нь Латин хэл дээр суурилсан Google Tesseract гэх мэт системүүд шиг өргөн тархсан биш байгаа юм.

## 1.2 Ашиглах технологийн судалгаа

Монгол бичгийн OCR системийг хөгжүүлэхэд ашиглах технологиудыг сонгохдоо төслийн шаардлага, өгөгдөл боловсруулалтын хэрэгцээ, танилт хийх нарийвчлал зэрэг хүчин зүйлсийг анхаарах хэрэгтэй. Технологиудыг доорх үндсэн бүлгүүдэд хуваан авч үзвэл:

### 1.2.1 Програмчлалын хэл

Монгол бичгийн OCR систем хөгжүүлэхэд програмчлалын хэл сонгох нь маш чухал. Тухайн хэлний нээлттэй эхийн сангууд болон машин сургалтын дэмжлэгийг харгалзан сонгох хэрэгтэй.

- Python: Машин сургалт, гүн сургалтанд хамгийн өргөн хэрэглэгддэг хэл. ‘TensorFlow’, ‘Keras’, ‘PyTorch’, ‘Tesseract’ зэрэг сангуудыг ашиглах боломжтой.

### 1.2.2 Машин сургалтын сангууд

OCR системийг илүү нарийвчлалтай болгохын тулд машин сургалт болон гүн сургалт ашиглах нь чухал. Энэхүү судалгааны ажилд доорх сангуудыг ашигласан:

- PyTorch: Фейсбүүкийн хөгжүүлсэн гүн сургалтын платформ. PyTorch нь уян хатан байдал болон өөрийн хэрэгцээнд тааруулж тусгайлсан өөрчлөлт хийх боломжтой тул Seq2Seq, Анхаарлын механизм зэрэг тусгай моделиудыг бүтээхэд тохиромжтой.
- Torchvision: PyTorch-ийн зураг боловсруулахад зориулсан нэмэлт сан. Энэ нь зургийг хөрвүүлэх, урьдчилан боловсруулах, өгөгдлийг нэмэгдүүлэхэд хэрэглэгддэг.
- Google Cloud Storage Python Client: Google Cloud дээрх өгөгдөлтэй ажиллахад ашиглагддаг Python сан. Үүнийг ашиглан өгөгдлийг хадгалах, унших үйлдлүүдийг гүйцэтгэсэн.
- Pandas: Өгөгдлийг удирдах, боловсруулахад өргөн ашиглагддаг сан. CSV өргөтгөл бүхий файлуудыг унших, хуваах, хадгалах зэрэгт ашигласан.

- scikit-learn: Өгөгдлийг хуваах (Сургалт, Баталгаажуулалт, Туршилт) болон үнэлгээ хийх зэрэгт ашигласан машин сургалтын сан.
- Editdistance: CER, WER-ийг тооцоход ашигласан.
- CUDA: GPU дээр сургалт хийхэд ашигласан.

Эдгээр сангууд нь OCR системийн хөгжүүлэлтийн явцад моделийг нарийн тохируулах, өгөгдөл боловсруулах, дүрс таних, сургах үйл явцад туслалцаа үзүүлсэн.

### 1.2.3 Дүрс боловсруулалт

- Pillow (PIL): Python-д өргөн хэрэглэгддэг зураг боловсруулах сан. Энэ санг ашиглан зурган өгөгдлийг унших, хэмжээг өөрчлөх, форматлах, хөрвүүлэх болон зурган өгөгдлийг урьдчилан боловсруулахад ашигласан. Мөн зурагнаас координат, зурмал замуудыг дүрс хэлбэрээр хадгалахдаа PIL-ийг ашигласан.
- NumPy: Зургийн координатыг боловсруулж, хэвийнжүүлэхэд NumPy санг ашигласан. Координат боловсруулахад шаардлагатай математик тооцоолол, өгөгдөл удирдахад NumPy чухал үүрэг гүйцэтгэсэн.

Эдгээрийг ашиглан зурагнуудыг урьдчилан боловсруулж, OCR системийн сургалтад зориулсан чанартай өгөгдөл бэлтгэсэн.

### 1.2.4 Хөгжүүлэлтийн Орчин ба Хэрэгслүүд

- Vertex AI Workbench: Энэ нь Python болон гүн сургалтын моделуудыг турших, хөгжүүлэх, өгөгдөл боловсруулахад тохиромжтой орчин юм. Vertex AI Workbench-д шинэ instance асаасны дараа JupyterLab орчинд ажиллах боломжтой бөгөөд энэ нь өгөгдөл боловсруулалт, модель сургалтын процессыг төвлөрсөн байдлаар удирдах, боловсруулахад тусалдаг.

- Google Cloud Storage: Зургийн өгөгдөл болон моделийн файлуудыг хадгалах, татаж авах үйл ажиллагаанд Google Cloud Storage ашигласан. Ингэснээр өгөгдлийг аюулгүй хадгалж, бусад төхөөрөмжөөс хялбар хандах боломжийг олгосон.
- VS Code: Локал хөгжүүлэлт хийх үед Visual Studio Code ашигласан. Энэ нь хялбар, олон төрлийн програмчлалын хэл болон платформуудтай ажиллах боломжтой, нэмэлт өргөтгөлүүдээр хөгжүүлэлтийг дэмждэг.
- Kaggle: Kaggle платформ дээр өгөгдөл боловсруулж, жижиг туршилтуудыг хийсэн. Kaggle нь үүлэн орчинд өгөгдөл боловсруулах, туршилтын үр дүнг харах боломж олгодог тул судалгааны ажилд үр ашигтайгаар ашигласан.

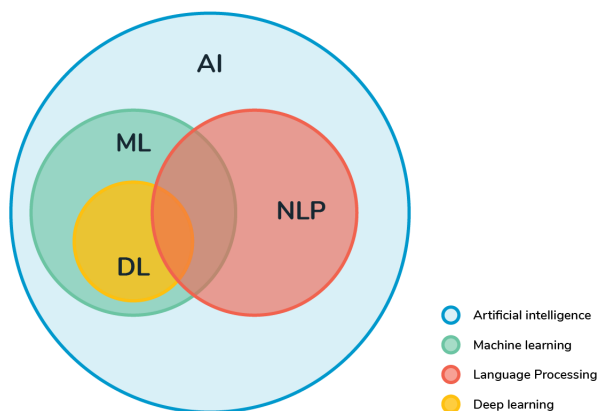
Эдгээр орчин, хэрэгслүүд нь судалгааны ажлын үр дүнг сайжруулж, хөгжүүлэлтийг хялбарчилж, өгөгдөл боловсруулах үйл явцыг илүү хурдан болгосон.

**Дүгнэлт:** Монгол бичгийн OCR системийг хөгжүүлэхэд Python хэл болон PyTorch, Torchvision зэрэг машин сургалтыг сангуудыг ашиглах нь үр дүнтэй гэж үзлээ. Дүрс боловсруулалт, машин сургалтын процессуудыг хялбаршуулахад Pillow зэрэг сангуудыг хэрэглэж, Google Cloud Storage ашиглан өгөгдөл хадгалж, боловсруулсан. Хөгжүүлэх болон туршихад Google Colab платформыг ашиглаж, GPU-тэй интеграци хийснээр, модель сургалтын үр ашгийг нэмэгдүүлэх боломжтой юм.

## 2. ОНОЛЫН СУДАЛГАА

### 2.1 Хиймэл оюун ухаан (AI) гэж юу вэ?

Хиймэл оюун ухаан буюу AI (Artificial Intelligence) нь компьютер болон бусад ухаалаг төхөөрөмжүүдийг хүн шиг сэтгэж, шийдвэр гаргах, асуудал шийдвэрлэх, өөрөө өөрийгөө хөгжүүлэх чадвартай болгох зорилготой шинжлэх ухааны салбар юм. AI нь машин сургалт, дүрс танилт, яриа боловсруулалт, өгөгдлийн шинжилгээ зэрэг технологиудыг багтааж, хүний оюун ухааны зарим шинжийг дуурайн ажиллахыг зорьдог.



Зураг 2.1: AI

AI нь анх 1950-иад онд хөгжиж эхэлсэн ба анхны AI системүүд нь бодлого, тоглоом зэргийг шийдвэрлэхэд чиглэж байв. 1980-аад оноос AI нь машин сургалт, нейрон сүлжээний тусламжтай илүү нарийн асуудлыг шийдвэрлэх чадвартай болж, орчин үед гүн сургалт (deep learning)-ын технологиор хүний чадварт ойртсон, нарийвчлал сайтай болон хөгжиж, олон чиглэлд өргөн хэрэглэгдэж байна.

## 2.1. ХИЙМЭЛ ОЮУН УХААН (AI) ГЭЖ ЮУ ВЭ? БҮЛЭГ 2. ОНОЛЫН СУДАЛГАА

### 2.1.1 AI-ийн төрөл ба түвшин

AI нь дараах үндсэн түвшинд ангилагддаг:

- *Хязгаарлагдмал AI (Weak AI)*: Тодорхой даалгавар гүйцэтгэхэд зориулагдсан, өнөөдрийн AI-ийн ихэнх нь энэ түвшинд хамаарагдана (жишээ нь, чатбот, дүрс таних систем).
- *Ерөнхий AI (Strong AI)*: Хүнтэй адил түвшний олон төрлийн даалгаврыг бие даан гүйцэтгэх чадвартай AI бөгөөд одоогоор бодит байдалд хүрээгүй.
- *Супер AI (Super AI)*: Хүний оюун ухаанаас давж гарах, илүү ухаалаг шийдвэр гаргах чадвартай AI. Энэ нь зөвхөн онолын түвшинд судлагдаж байна.

### 2.1.2 AI-ийн үндсэн технологиуд

AI нь дараах технологиудыг ашиглан үйл ажиллагаа явуулдаг:

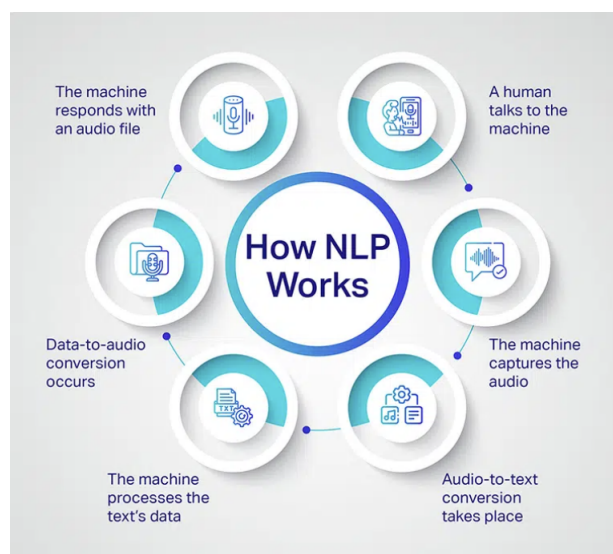
- *Машин сургалт (Machine Learning)*: Машиныг (компьютерийг) өгөгдлөөс суралцуулах зорилготой технологи бөгөөд статистик, магадлал дээр суурилсан аргаар өгөгдлийн дотоод хамаарлыг таних боломжтой.
- *Байгалийн хэлний боловсруулалт (NLP)*: NLP нь хүний хэлийг тайлбарлах, удирдах, ойлгох чадварыг компьютерт олгодог машин сургалтын технологи юм.
- *Компьютерын хараа (Computer Vision)*: Дүрс, видеог шинжлэх технологи бөгөөд энэ нь объект илрүүлэлт, дүрс танилт зэрэгт ашиглагддаг.

### 2.1.3 AI-ийн хэрэглээ ба ирээдүй

AI нь эрүүл мэнд, санхүү, боловсрол, үйлдвэрлэл зэрэг олон салбарт өргөн нэвтэрч, ажлыг автоматжуулж, үр дүнг сайжруулж байна. Эрүүл мэндийн салбарт оношлогоо хийх, санхүүгийн салбарт зээлийн эрсдлийг тооцох, аж үйлдвэрлэлд чанарын хяналт хийхэд ашиглагдаж байгаа нь үүний жишээ юм. Ирээдүйд AI нь технологийн дэвшлийг хурдасгах боловч ёс зүй, нууцлалын асуудал, ажлын байрыг халах эрсдэл зэрэг сорилтуудыг даван туулах шаардлагатай юм.

## 2.2 Байгалийн Хэлний Боловсруулалт (NLP) гэж юу вэ?

Байгалийн Хэлний Боловсруулалт буюу NLP (Natural Language Processing) нь компьютерыг ашиглан хүний бичсэн болон ярьсан хэл зүйн өгөгдлийг ойлгож, боловсруулж, дүн шинжилгээ хийх зорилготой хиймэл оюун ухааны нэг салбар юм. NLP нь хэл зүйн болон утга зүйн мэдээллийг таньж, текст ангилах, орчуулга хийх, мэдрэмж илрүүлэх зэрэг олон талт хэрэглээнд ашиглагддаг.



Зураг 2.2: NLP

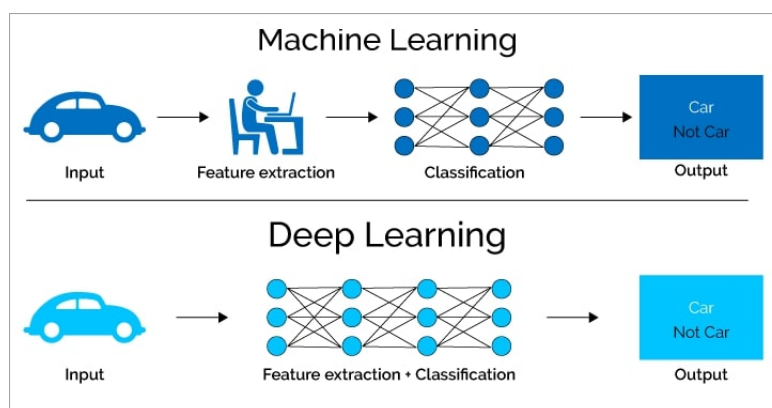
### 2.2.1 NLP-ийн арга зүй ба орчин үеийн хэрэглээ

NLP нь дүрэмд суурилсан арга (тодорхой хэл зүйн дүрмүүдэд тулгуурлан), статистикт суурилсан арга (үгийн давтамж, хэл зүйн хамаарал ашиглан), гүн сургалт (RNN, Transformer гэх мэт) гэсэн аргуудаар хөгжиж ирсэн. NLP нь өнөөдөр орчуулгын систем, чатбот, текст ангилах, мэдрэмжийн шинжилгээ зэрэг олон салбарт өргөн хэрэглэгддэг.



## 2.3 Машин сургалт ба гүн сургалт

Машин сургалт (ML) нь өгөгдлөөс суралцан, хараахан боловсруулаагүй өгөгдөлд дүгнэлт хийж, тодорхой зааварчилгаагүйгээр даалгаврыг гүйцэтгэх чадвартай статистик алгоритмуудыг боловсруулах, судлахтай холбоотой хиймэл оюун ухааны салбар юм. Гүн сургалтын салбарт гарсан ахиц нь нейрон сүлжээнүүдийн гүйцэтгэлийг сайжруулж, өмнөх олон аргуудыг гүйцэтгэлийн хувьд давж гарах боломжийг олгосон.



Зураг 2.3: ML vs DL

### 2.3.1 Машин сургалтын үндсэн төрлүүд

Машин сургалт нь дараах гурван төрлөөр хөгжиж байна:

- *Хяналттай сургалт (Supervised Learning):* Урьдчилан шогшожуулсан оролт, гаралт дээр тулгуурлан моделийг сургаж, ангилал, таамаглал хийх.
- *Хяналтгүй сургалт (Unsupervised Learning):* Урьдчилан шошгожуулаагүй өгөгдлөөс бүлэг бүлгээр ангилах, өгөгдлийн хэв маягийг таних.
- *Бэхжүүлэлт сургалт (Reinforcement Learning):* Орчны өөрчлөлтөд тохируулан шийдвэр гаргаж, модель өөрийгөө сайжруулах чадварыг эзэмших.

### 2.3.2 Гүн сургалт ба үндсэн архитектурыуд

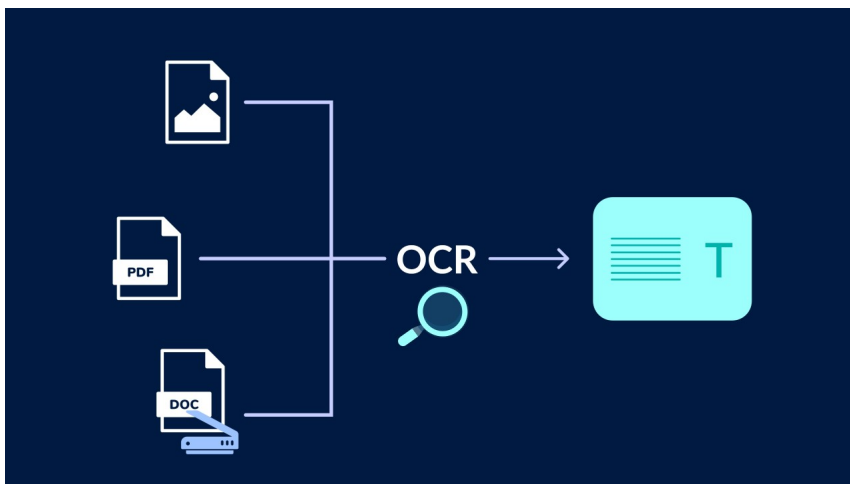
Гүн сургалт (Deep Learning) нь олон давхарга бүхий нейрон сүлжээн дээр суурилсан бөгөөд том хэмжээний өгөгдлөөс нарийн төвөгтэй мэдээллийг таних, шинжлэх чадвартай. Энэ нь CNN (дүрс танихад), RNN (дараалсан өгөгдөл боловсруулахад), Transformer (урт текстийг ойлгоход) зэрэг архитектурыг ашигладаг.

### 2.3.3 Машин сургалт ба гүн сургалтын хэрэглээ

Эдгээр технологиуд нь анагаах ухаан, санхүү, боловсрол, маркетинг зэрэг олон салбарт өргөн хэрэглэгдэж байна. Жишээ нь, дүрс боловсруулалтаар онош тодорхойлох, санхүүгийн таамаглал хийх гэх мэт.

## 2.4 OCR гэж юу вэ?

OCR (Optical Character Recognition) буюу дүрсээс текст таних технологи нь компьютерыг ашиглан хэвлэмэл болон гар бичмэл текстийг текст файл болгон хувиргадаг. Энэ нь ном, сонин зэрэг материалыг сканнердан, дижитал хэлбэрт оруулж, засварлах, хайлт хийх боломжийг олгодог.



Зураг 2.4: OCR

Анх 1920-иод онд дараалсан дүрс, тоонуудыг таних технологи хэлбэрээр үүссэн ба 1970-аад оноос текстийг таних чадвартай болж, орчин үед хиймэл оюун ухаан, гүн сургалтад суурилан илүү нарийвчлалтай болсон. Өнөөдөр OCR нь бичмэл текстийг электрон хэлбэрт хөрвүүлж, хайлт хийх, архивлах, засварлах зэрэгт өргөн ашиглагдаж байна.

### 2.4.1 OCR-ийн төрлүүд ба алгоритмууд

- *Хээ тааруулах (Template Matching)*: Урьдчилан бэлтгэсэн модельтой харьцуулан тэмдэгтүүдийг таних арга.
- *Онцлогт суурилсан OCR (Feature-based OCR)*: Тэмдэгтийг шугам, өнцөг, муруй хэлбэрээр задлан, тохирох тэмдэгтийг таних арга.
- *Гүн сургалт дээр суурилсан OCR*: CNN болон RNN зэрэг нейрон сүлжээг ашиглан нарийвчилсан танилт хийх арга бөгөөд орчин үед өргөн хэрэглэгдэж байна.

## 2.5 Монгол бичгийн судалгаа

Монгол бичиг нь босоо бичигддэг бөгөөд үсгүүд нь хоорондоо холбогдсон, үгийн байрлалаас хамаарч хэлбэр нь өөрчлөгддөг онцлогтой. Монгол бичгийн нэг үсэг нь тухайн үгийн эхэнд, дунд, төгсгөлд өөр хэлбэртэй байж болдог. Ирээдүйд хэвлэл мэдээлэл болон боловсролын салбарт Монгол бичгийн OCR технологи нь Монгол хэл бичгийн судалгаа, боловсрол, соёлын өвийг хамгаалах, түгээхэд ихээхэн ач холбогдолтой. Хэвлэмэл материал, гар бичмэл, түүхэн эх сурвалжуудыг цахим хэлбэрт хөрвүүлэх, хадгалах ажилд өргөн хэрэглэгдэх боломжтой.

	<i>t</i>	<i>d</i>	<i>ta/te</i> <i>da/de</i>	<i>ti</i> <i>di</i>	<i>to/tu</i> <i>do/du</i>	<i>ts/ts</i> <i>ds/ds</i>
initial						
non-initial						
final						

Зураг 2.5: Монгол бичгийн бүтэц

### 2.5.1 Ач холбогдол

Монгол бичгийн OCR технологийн судалгаа нь Монголын түүх, соёл, бичиг соёлын өвийг хамгаалах, түгээхэд асар их ач холбогдолтой. Монгол бичгийг цахим хэлбэрт хөрвүүлэх нь боловсролын салбарт олон талын хэрэглээтэй бөгөөд сурах бичиг, судалгааны материалууд, түүхэн эх сурвалжийг олон нийтэд илүү хүртээмжтэй болгох боломжийг бүрдүүлнэ. Нэмж хэлэхэд, монгол бичгийн хэвлэмэл болон гар бичмэлүүдийг цахимжуулснаар тэдгээрийг судлаачид болон уншигчид онлайн орчинд хялбархан ашиглах боломжтой болно. Ингэснээр үндэсний өв соёл, хэлний цаашдын судалгааг дэмжиж, залуу үедээ тэдгээр уламжлалыг дэлгэрүүлэхэд чухал нөлөө үзүүлнэ.

### 2.5.2 Ирээдүйн чиглэл

Ирээдүйд Монгол бичгийн OCR технологийг боловсронгуй болгож, олон хэл дээрх текстүүдтэй уялдуулах, боловсролын системд нэвтрүүлэх чиглэлд судалгаа, хөгжүүлэлтийг үргэлжлүүлэх шаардлагатай. Тухайлбал, гар бичмэл текстийг таних, хуучин эх сурвалжуудыг сэргээн цахимжуулах, бичвэрийн нарийвчлалыг нэмэгдүүлэх зэрэг олон төрлийн судалгааны чиглэлд хиймэл оюун ухаан болон машин сургалт, гүн сургалтын аргуудыг ашиглах нь өндөр үр дүн үзүүлэх боломжтой байна. Мөн боловсруулсан мэдээллүүдийг хадгалах, хамгаалах, нийтэд түгээх боломжийг бүрдүүлснээр Монгол бичгийн боловсрол, судалгааны хөгжлийг олон улсын түвшинд хүргэх боломжтой.

**Дүгнэлт:** Монгол бичгийн OCR системийг хөгжүүлэх нь босоо бичгийн бүтэц, үсгийн хэлбэрийн олон янз байдал, холболтын онцлог зэргээс шалтгаалан бусад OCR системүүдээс илүү төвөгтэй юм. Энэ судалгаанд Монгол бичгийн өвөрмөц шинж чанаруудыг харгалзан OCR технологийн онолын үндсийг судалж, хөгжүүлэлтийн арга замуудын эхлэлийг авч үзлээ. Цаашид Монгол бичгийн OCR технологийг сайжруулснаар түүх, соёлын өвийг хамгаалах, боловсролын чанарыг нэмэгдүүлэхэд үнэтэй хувь нэмэр оруулах боломжтой бөгөөд энэ нь Монгол хэл, бичгийн судалгааны хөгжлийг шинэ түвшинд гаргах чухал алхам юм.

## 3. ӨГӨГДЛИЙН ШИНЖИЛГЭЭ

### 3.1 Өгөгдлийн багцын танилцуулга

Өгөгдлийн багцыг Kaggle-аас сонгон авсан. 2021 онд "MOLHW" хэмээх монгол хэлний үгийн түвшний онлайн гар бичмэлийн шинэ өгөгдлийн багцыг нийтэлсэн байна. Өгөгдлийн багц нь гар бичмэл монгол үгсээс бүрдэх бөгөөд үүнд 200 бичээчийн бичсэн 164,631 түүвэр, 40,605 монгол үг багтана. Үгийн координатын цэгүүдийг цуглуулж, гар утсанд зориулсан тусгай программ дээр харгалзах үгийг бичжээ. Монгол хэлний латин галиг үсгээр үг бүрийн координатыг тэмдэглэсэн. Үүний зэрэгцээ өгөгдлийн багцад бичээчийн ID, гар утасны дэлгэцийн мэдээллийг бүртгэсэн. нийт нэг жил орчмын хугацаанд 200 хүн монгол бичгээр бичигдсэн үгсийг мэдрэгчтэй дэлгэц дээр хуруугаараа бичин гараар нягт нямбай баталгаажуулсан байна.

- Нийт үгсийн сан: 40605
- Нийт бичсэн хүний тоо: 200
- Нийт түүврийн тоо: 164631

Kaggle платформд нээлттэй эхээр байршуулсан "MOLHW" өгөгдлийн багц нь нийт 6.68GB бүхий 6 өөр текст файлуудаас бүрдэх ба тэдгээрээс судалгааны ажлынхаа зорилгод нийцүүлэн 3-ыг нь татаж ашигласан ба файлуудын дэлгэрэнгүйг тайлбарлав.

### 3.1. ӨГӨГДЛИЙН БАГЦЫН ТАНИЛЦУУЛ **В**ҮЛЭГ 3. ӨГӨГДЛИЙН ШИНЖИЛГЭЭ

- **Урьдчилан боловсруулсан юникод өгөгдөл бүхий файл**

Урьдчилан боловсруулсан өгөгдлийг агуулах файл нь гар бичмэл үг бүрт харгалзах шошго болон тухайн гар бичмэл үгийг дүрсэлсэн координатуудыг агуулна. Доторх мөр бүр нь нэг түүвэр буюу нэг үгийн жишээ юм. Жишээ бүр нь дараах багана бүхий өгөгдлүүдээс бүрдэнэ.

**Шошго:** Монгол үг латин галигаар

**Зохиогчийн ID:** Writer md5 шифрлэгдсэн ID

**Дэлгэцийн өргөн:** Гар утасны дэлгэцийн өргөн

**Дэлгэцийн өндөр:** Гар утасны дэлгэцийн өндөр

**Дэлгэцийн пикселийн нягтрал:** Гар утасны дэлгэцийн пикселийн нягтрал

**Замын координат:** Координатын өгөгдлийн нум нь  $[[x,y],[x,y],\dots,[x,y]]$ , нэг координатын формат нь "[x, y]", энд "x" Х тэнхлэгийн координатыг, "y" нь Y тэнхлэгийн координатыг илэрхийлнэ. Координат дахь "[-1,-1]" нь үзэг өргөхийг илэрхийлнэ.

DataFrame shape: (164631, 6)

	Label	Author_ID	Screen_Width	Screen_Height	\
0	ab	1df87f3d797035f6f919ef8faeaa7b1a	1080	2265	
1	ab	2284c343bb9739ad6c124c1022e38ad3	1080	2160	
2	ab	2284c343bb9739ad6c124c1022e38ad3	1080	2160	
3	ab	a0fc0439c4b0bee243cc9c8cb934fa8b	720	1440	
4	aba	37daadd776b8a6eff7f50548c44e7b56	1080	2163	

	Pixel_Density	Writing_Track
0	3.0	$[[[-1, -1], [0.7425679968102712, 0.0], [0.75426...$
1	3.0	$[[[-1, -1], [0.33827970840834104, 0.0], [0.3577...$
2	3.0	$[[[-1, -1], [0.4377576951907115, 0.0], [0.45196...$
3	2.0	$[[[-1, -1], [0.6360624108658747, 0.0], [0.65164...$
4	2.55	$[[[-1, -1], [0.8663801691206826, 0.0], [0.88342...$

Зураг 3.1: Урьдчилан боловсруулсан Юникод өгөгдөл

### 3.1. ӨГӨГДЛИЙН БАГЦЫН ТАНИЛЦУУЛ **В**ҮТЭГ 3. ӨГӨГДЛИЙН ШИНЖИЛГЭЭ

- **Шошгууд (Латинаар) болон Графем кодын уялдаа**

Урьдчилан боловсруулсан Юникод өгөгдлийн эхний баганад харгалзах шошгууд болон тэдгээрийн графем кодын хамаарлыг агуулсан файлыг мөн ашиглана.

```
a=a ax
a^ca=a c ax
ai=a az ix
ainstEn=a az iz iz az s d w ax
aiGgwri=a az iz iz ng gz o r ix
aibagar=a az iz iz bos bax gz az rx
aibwgwhan=a az iz iz bos box gz o az az az ax
aih_a=a az iz iz hx sp ex
aihaljahw=a az iz iz az az az lz jz az az az ox
aihwbtwr=a az iz iz az az o bos d o rx
aigag=a az iz iz gz az hx
aimag=a az iz iz mz az hx
aimaglan=a az iz iz mz az az az lz az ax
aimaglahw=a az iz iz mz az az az lz az az az ox
aimaglagsan=a az iz iz mz az az az lz az az az s az ax
```

Зураг 3.2: Латин үгнүүд болон Графем кодын уялдаа

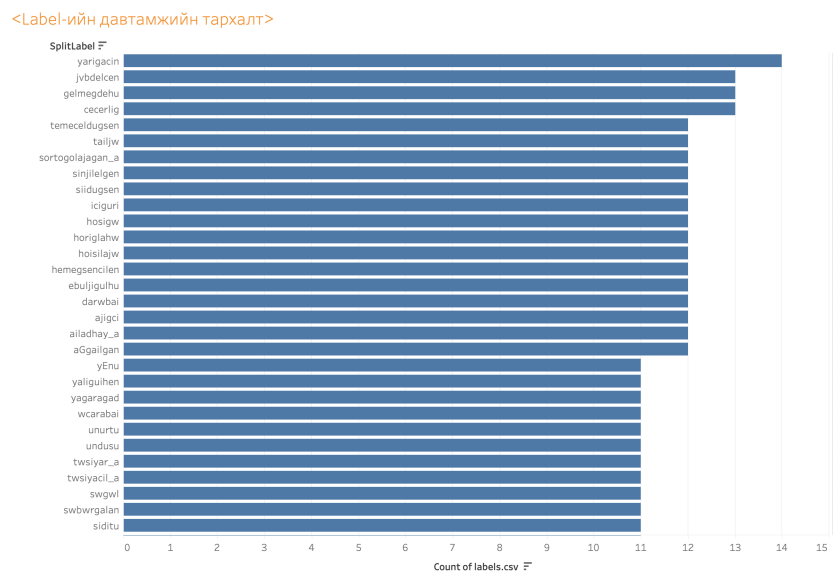
- **Графем код болон Монгол бичгийн Юникодын уялдаа** Графем код болон монгол юникод хоорондын уялдаа нь ASCII2Unicode.txt дээр тайлбарлагдсан болно. Жишээлбэл, ”abaci” шошгыг ASCII2Unicode.txt дагуу ”0x1820 0x182a 0x1834 0x1822” болгон хувиргаж болно. Монгол үг нь ”□□□□”.

Name	Shape	Name	Shape	Name	Shape
d	ᠳ	o	ᠣ	si	ᠰᠢ
n	ᠨ	bos	ᠪᠣᠰ	box	ᠪᠣᠭ
pos	ᠮᠣᠰ	hus	ᠬᠤᠰ	m	ᠮ
ex	ᠡᠬᠡ	l	ᠯ	s	ᠰ
ox	ᠣᠬᠡ	sh	ᠰᠢ	t	ᠲ
ax	ᠠᠬᠡ	ix	ᠶ	w	ᠠᠢ
c	ᠴ	j	ᠵ	y	ᠶ
r	ᠷ	fos	ᠮᠣᠰ	kos	ᠬᠣᠰ
ci	ᠴᠢ	z	ᠵ	a	ᠠ
hew	ᠬᠡᠠ	re	ᠷᠡ	bax	ᠪᠠᠬᠡ
bix	ᠪᠢᠬᠡ	hos	ᠬᠣᠰ	hes	ᠬᠡᠰ
g	ᠭ	az	ᠠᠵ	iz	ᠢᠵ
nz	ᠨᠵ	mz	ᠮᠵ	lz	ᠯᠵ
jz	ᠵᠵ	gz	ᠭᠵ	ng	ᠨᠭ
nx	ᠨᠬᠡ	hx	ᠬᠡᠬᠡ	gx	ᠭᠡᠬᠡ
mx	ᠮᠬᠡ	lx	ᠯᠬᠡ	rx	ᠷᠬᠡ
sx	ᠰᠬᠡ	shx	ᠰᠢᠬᠡ	sp	ᠰᠠᠭ

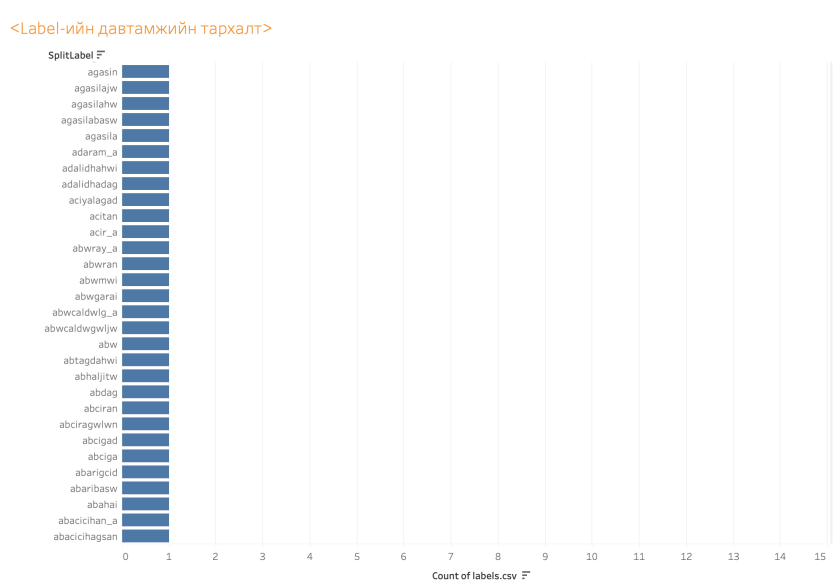
Зураг 3.3: Графемийн кодын тодорхойлолт

## 3.2 Өгөгдлийн шинжилгээ

- **Шошгуудын давтамжийн шинжилгээ:** Нийт 164631 түүврээс хамгийн их болон хамгийн бага давтагдаж буй шошгуудыг онцолж харууллаа.



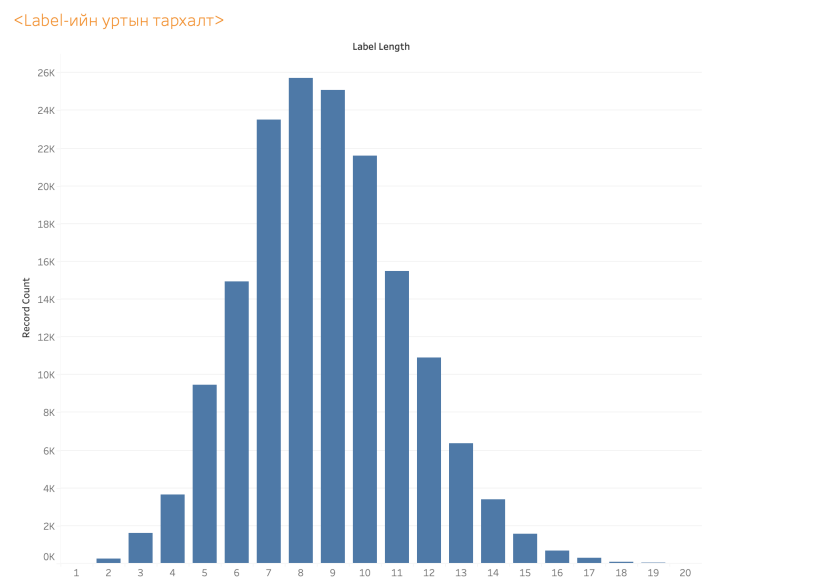
Зураг 3.4: Хамгийн их давтагдаж буй шошгуудын гистограмм



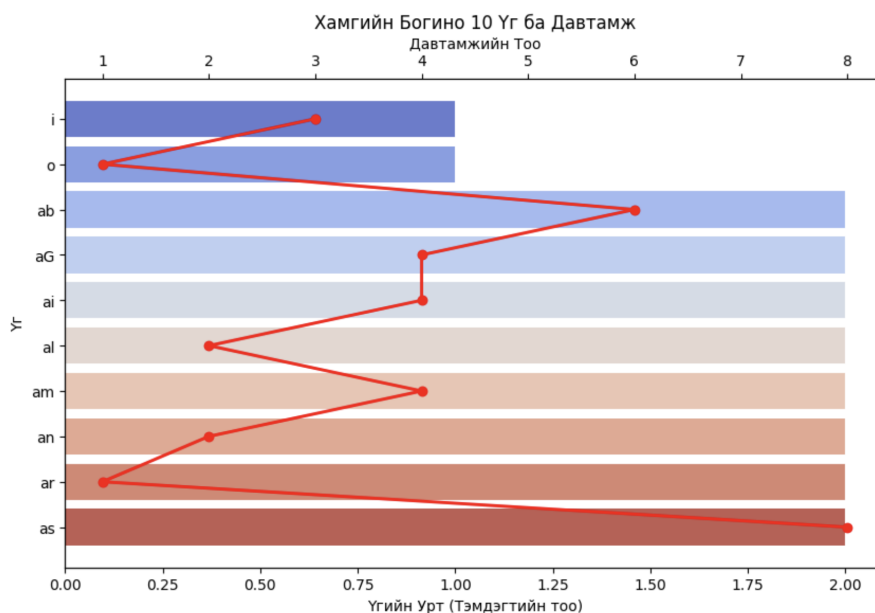
Зураг 3.5: Хамгийн бага давтагдаж буй шошгуудын гистограмм



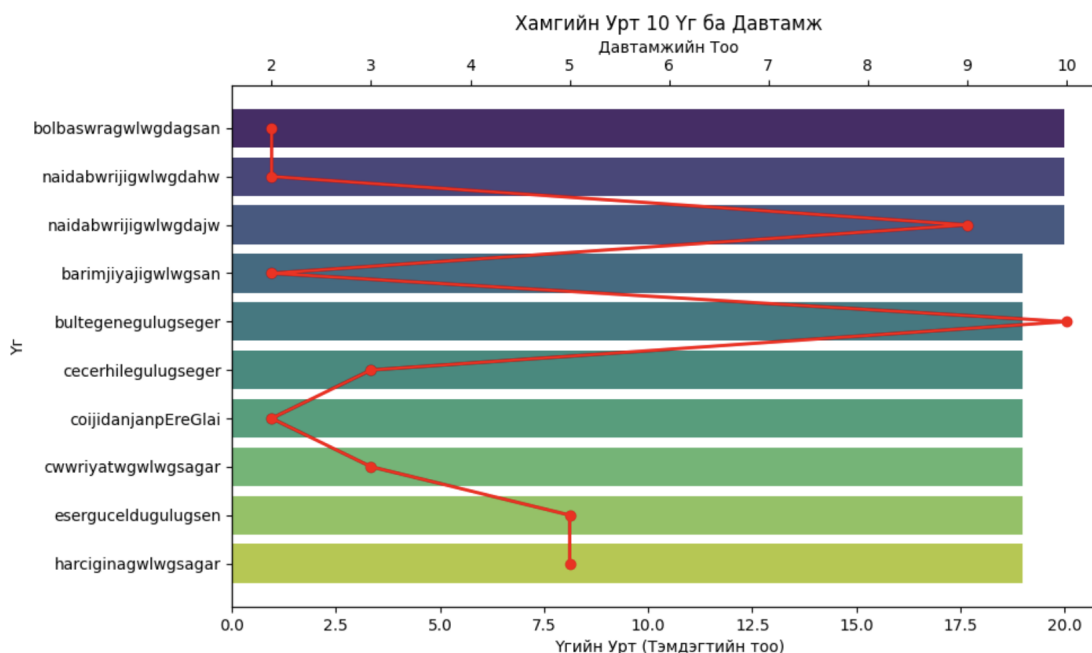
- **Шошгуудын уртын тархалтын шинжилгээ:** Нийт 164631 түүврээс хамгийн урт шошгууд болон тэдгээрийн давтамжийг хамтад нь харууглах зорилгоор хамгийн богино болон урт шошгуудыг онцолж харууллаа.



Зураг 3.6: Нийт шошгуудын уртын хамаарлын гистограмм



Зураг 3.7: Хамгийн богино 10 шошго ба тэдгээрийн давтамжийн гистограмм



Зураг 3.8: Хамгийн урт 10 шошго ба тэдгээрийн давтамжийн гистограмм

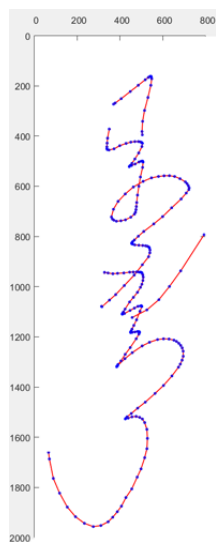
### 3.3 Өгөгдлийн багцын бэлтгэл

Kaggle дээрээс өгөгдлийн файлыг татан авсны дараагаар, боловсруулах шаардлагатай. Монгол бичгийн гар бичмэлийн координатуудыг холбон зурснаар зурган форматруу хөрвүүлж, тэдгээр зургуудыг шошгожуулан бэлтгэсний дараа сургалт, баталгаажуулалт, туршилтын зориулалтаар ангилан хувааж бэлтгэх ажлуудыг хийж гүйцэтгэсэн.

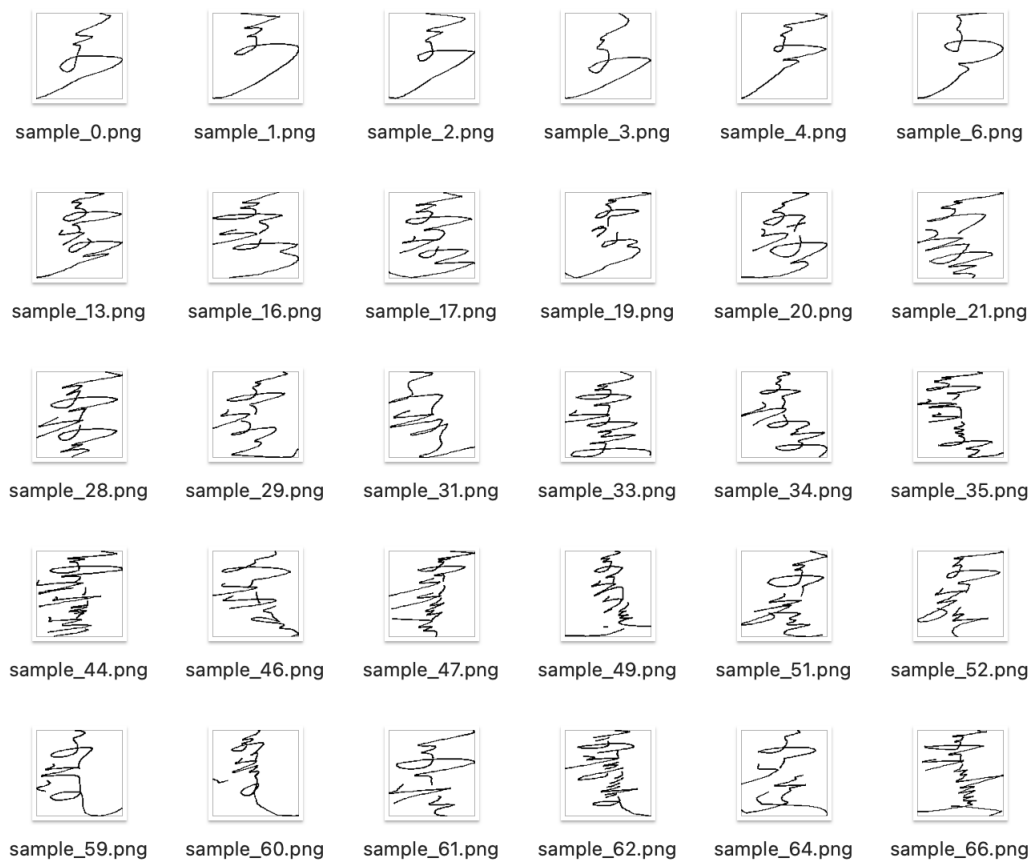
#### 3.3.1 Өгөгдлийн боловсруулалт

**Координатын дагуу зургуудыг үүсгэх :**

- Координатыг уншиж тэдгээрийг зураг болгон дүрсэлж, үзэг өргөсөн цэгүүдэд зургийг тасалж байгаа нь гар бичмэлийн дүрслэлийг зөв буулгахад чухал алхам юм. Координатыг зөв хэмжээнд нь масштаблаг, зурган форматыг жигд болгох үйлдэл мөн хийгдэх бөгөөд энэ нь render хийх үед дүрслэл жигд харагдахад тустай.



Зураг 3.9: Координатыг холбож зураг үүсгэх жишээ



Зураг 3.10: Render хийгдсэн зургуудын жишээ



Зураг 3.11: Render хийгдсэн зургийн жишээ

**Нийт үүсгэсэн зургийн түүврүүдийг шошгожуулах :**

- Үүсгэсэн зургуудыг тодорхой давтагдахгүй нэрээр хадгалж, зураг бүрт харгалзах Unicode шошгын хамт 'labels.csv' файлд хадгалсан. Үүний үр дүнд render хийгдсэн зургийн багц болон тэдгээрийн шошго бүхий 2 багана мэдээллээс бүрдэх файл үүсэх бөгөөд энэ нь сургалтанд ашиглахад бэлэн болснийг илтгэнэ.
- Kaggle дээрх өгөгдлийг ашиглан нийт 164631 зургийг бэлтгэн шошгожуулж хадгалсан байдал:

labels	
filename	label
sample_0.png	ab
sample_1.png	ab
sample_2.png	ab
sample_3.png	ab
sample_4.png	aba
sample_5.png	aba
sample_6.png	aba
sample_7.png	aba
sample_8.png	ab
sample_9.png	abaci
sample_10.png	abaci

Зураг 3.12: Шошгожуулсан өгөгдлийн багц

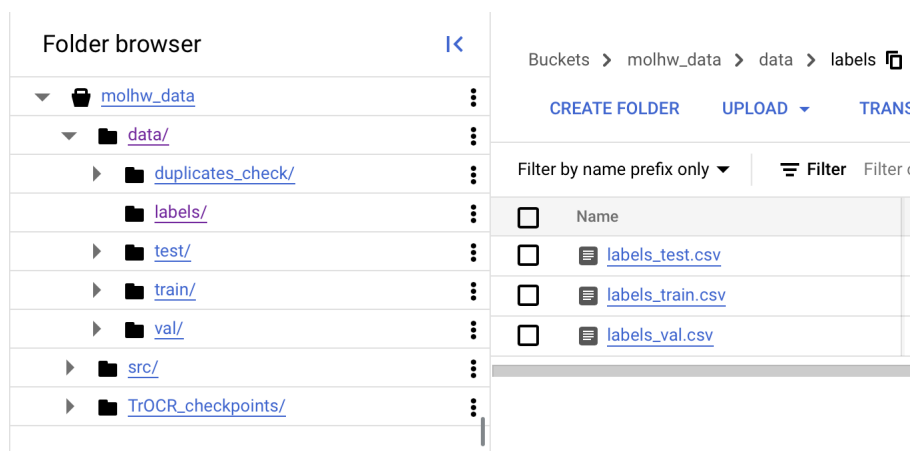
### Charset.txt файлыг бэлтгэх :

Монгол бичгийн үсэг, тэмдэгт бүрийг зөв танихын тулд тухайн бичгийн системийн бүх тэмдэгтийг charset буюу тэмдэгтийн багцад бүрэн багтаах шаардлагатай. Тиймээс charset.txt файл нь моделийн сургалтад болон дүрснээс текст таних үйл явцад чухал үүрэг гүйцэтгэнэ. Тэмдэгт бүрийг ангилж, шаардлагатай тэмдэгтүүдийг харгалзах файлд бичиж хадгална. Мөн тусгай тэмдэгтүүд болох <blank> болон <unk> зэргийг нэмдэг бөгөөд эдгээр нь OCR моделийн сургалт болон танилтын үед тусгай үүрэгтэй.

### 3.3.2 Өгөгдлийн хуваалт

#### Өгөгдлийг 70, 15, 15 хувийн харьцаагаар хуваах:

- Сургалтын өгөгдөл нь моделийг сургах үндсэн өгөгдөл болж, баталгаажуулалтын өгөгдөл нь сургалтын явцад уг модель хэрхэн сурч байгааг үнэлэх, туршилтын өгөгдөл нь сургалтын дараа моделийн хүчин чадлыг шалгах боломжийг олгоно. - Хуваагдсан өгөгдлийг тус бүрийн .csv файлд хадгалснаар, моделийн сургалт, баталгаажуулалт, туршилтын үр дүнг тус тусад нь шалгах, хөгжүүлэх боломжийг олгоно. Өгөгдлийн багцыг сургалт, баталгаажуулалт, туршилтын багцуудад хуваан дараагийн сургалтанд бэлэн болгоод, Google Cloud Storage-д хадгалсан байдлыг харууллаа.

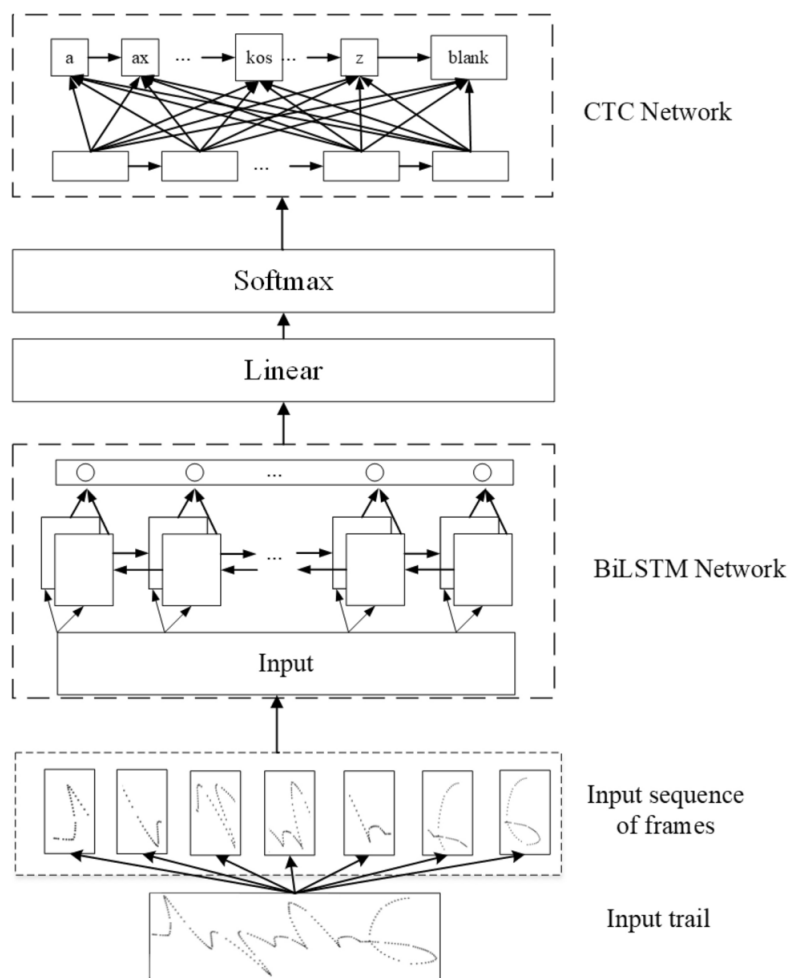


Зураг 3.13: Боловсруулж дууссан өгөгдлийг хуваан бэлтгэсэн нь

## 4. АРГА БА АЛГОРИТМ

### 4.1 Моделийн Оновчтой Архитектур

Монгол бичгийн гар бичмэл үгнүүдийн зургаас (нэг зураг = нэг үг) текст болгон хөрвүүлэх OCR системд олон төрлийн архитектуруудыг ашиглах боломжтой. Зөв тохиромжтой арга, алгоритмыг сонгох нь хамгийн эхний чухал алхам юм. Тохиромжтой байж болохуйц алгоритм, архитектуруудыг судалж, ижил төстэй ажлуудын судалгаанд мөн үндэслэн CNN + BiLSTM + CTC хосолсон моделийг сонгосон болно.



Зураг 4.1: Model Architecture

#### 4.1.1 CNN + BiLSTM + CTC хосолсон архитектурын давуу талууд

Энэхүү хосолсон архитектурыг сонгоход дараах шалгуур үзүүлэлтүүд болон, давуу талуудыг харгалзан үзсэн болно :

- **Бичгийн хэв маягийн ялгаа**

- Хүмүүсийн бичгийн хэв маяг, үсгийн хэлбэр, холбоос нь харилцан адилгүй. Тиймээс бусад урьдчилан сургасан моделиуд нь Монгол бичгийн гар бичмэлийн өвөрмөц онцлогуудыг зөв танихгүй байж болох ба өгөгдөлд дасан зохицоход хэцүү байж болно.
- Монгол бичгийн гар бичмэл ижилхэн үгийн уртууд ч харилцан адилгүй байж болох тул, CTC нь энэ асуудлыг шийдэж өгнө.
- BiLSTM нь үсгийн дарааллын алдааг бууруулах ба үгийн үсгүүдийг зөв дарааллаар таних магадлалыг нэмж өгнө.
- Нэг зураг = нэг үг тул, CNN + BiLSTM нь ийм төрлийн өгөгдөлд сайн тохирох боломжтой.

- **Цаг хугацаа, техник хангамжийн нөөц боломж**

- Гар бичмэлийн зургийг үсэг тус бүрээр embedding хийх нь их цаг хугацаа, нөөц шаардана. Бусад архитектурууд (жишээ нь, Анхааралын механизмд суурилсан модельууд) нь ихэвчлэн нарийн тохиргоо шаарддаг. CTC (Connectionist Temporal Classification) ашигласнаар зөвхөн үгийн түвшинд шошгожуулсан өгөгдөлтэй ажиллах боломжийг олгож өгнө.
- CNN + BiLSTM нь Трансформерд суурилсан модулиудаас бага нөөц шаарддаг харьцангуй хөнгөн архитектур. Бага хүчин чадалтай компьютер, GPU дээр ч сургалт, туршилтыг хурдан хийх боломжтой.

- **Харьцангуй хялбар архитектурын шийдэл**

- Сүлжээний давхаргууд, нейроны тоо болон Сургалтын Параметрууд буюу сургалтын хурд, багцын хэмжээ, алдагдлын функц зэргийг тухайн нөхцөлд тохируулан өөрчлөх боломжтой. Ингэснээр илүү хурдан, үр дүнтэй сургалт явуулах, алдааг хурдан засах боломжтой болно.

- **Бэлтгэсэн өгөгдлийн багц дээр эхнээс нь сургах боломж**

- Модель зөвхөн бэлтгэсэн өгөгдөл дээр эхнээс нь тохируулж сургаж буй тул , илүү өндөр нарийвчлалтай үр дүнд хүрэх боломжтой гэж үзсэн.
- Бусад урьдчилан сургасан моделиуд нь ихэвчлэн кирилл гар бичмэл өгөгдөл дээр суурилсан байдаг тул, уламжлалт монгол бичгийн өгөгдөлд сайн тохирохгүй хүндрэлтэй байж болно.

#### 4.1.2 Бусад Архитектуруудтай Харьцуулсан Давуу Талууд

##### **Анхаарлын Механизм Суурилсан Кодлогч-Декодлогч Моделиудтай Харьцуулсан Давуу Тал**

*Өгөгдлийн Шаардлага:* Анхаарлын Механизмд суурилсан моделиуд нь их хэмжээний өгөгдөл, нарийн шошгожуулалт шаарддаг. *Давуу Тал:* Өгөгдлийн багц хязгаарлагдмал бол, CNN + BiLSTM + CTC архитектур нь илүү үр дүнтэй.

##### **Трансформерүүдтэй Харьцуулсан Давуу Тал**

*Нөөцийн Шаардлага:* Трансформерд суурилсан моделиуд нь их хэмжээний тооцоолох нөөц, цаг хугацаа шаарддаг. *Давуу Тал:* CNN + BiLSTM архитектур нь бага нөөцөөр ажиллах боломжтой. Трансформерд суурилсан моделиуд нь урт дарааллыг боловсруулахад сайн боловч, богино дараалал (нэг үг) дээр CNN + BiLSTM нь хангалттай.

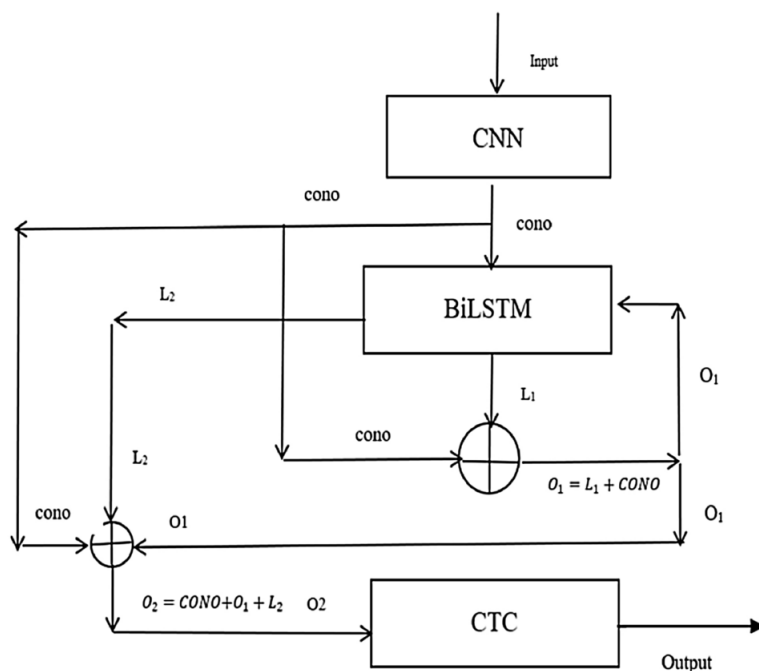
##### **Зөвхөн CNN Ашигласан Моделиудаас Давуу Тал**

*Дарааллын Мэдээлэл:* Зөвхөн CNN ашигласан моделиуд нь дарааллын мэдээллийг бүрэн дүүрэн ойлгож чадахгүй байх магадлалтай. *Давуу Тал:* BiLSTM нь үсгийн дарааллын хамаарлыг ойлгож, алдаа гаргах магадлалыг бууруулна. Гар бичмэлийн холбоос, дарааллыг зөв танихад тусална.



## 4.2 CNN, BiLSTM, CTC-ийн Үүрэг

Монгол гар бичмэл үгнүүдийг зурагнаас текст болгон хөрвүүлэх OCR системд ашиглаж буй Convolutional Neural Network (CNN), Bidirectional Long Short-Term Memory (BiLSTM), Connectionist Temporal Classification (CTC) гэсэн гурван үндсэн бүрэлдэхүүн хэсгүүдийг тус бүрт нь авч үзье. Эдгээр бүрэлдэхүүн хэсэг тус бүр өөрийн үүрэгтэй бөгөөд тодорхой давуу талуудтай.



Зураг 4.2: CNN + BiLSTM + CTC

### 4.2.1 Convolutional Neural Network (CNN)

#### Үүрэг:

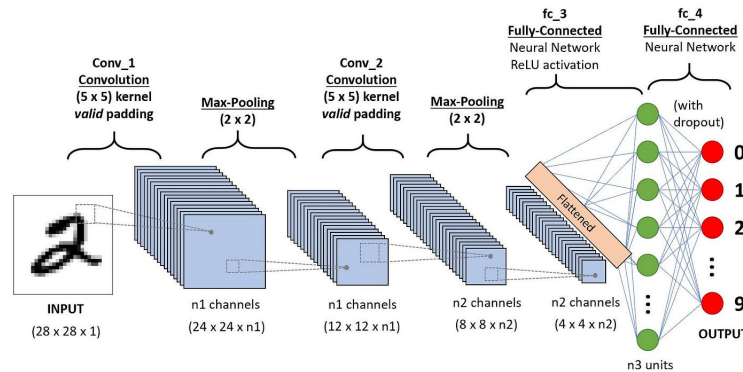
- Онцлог Илрүүлэлт (Feature Extraction): CNN нь оролтын зургаас орон зайн онцлогуудыг илрүүлэхэд ашиглагдана. Энэ нь гар бичмэлийн үсгийн мөр, нум, цэг зэрэг жижиг хэсгүүдийг олж илрүүлнэ.

- Хэмжээсийн Багасгалт (Dimensionality Reduction): Pooling давхаргуудыг ашиглан зурган дахь мэдээллийг нягтруулж, тооцооллын үр ашигтай байдлыг хангана.

**Давуу Тал:**

- Орон Зайн Онцлогийг Илрүүлэх Чадвар: CNN нь дүрсийн жижиг хэсгүүдийг илрүүлэхэд маш сайн тул гар бичмэлийн үсгүүдийн ялгааг зөв тодорхойлно.

- Хувиргалтад Тэсвэртэй Байдал: Конволюци болон pooling давхаргууд нь жижиг шилжилт, гажилт, эргэлтэд тэсвэртэй байх боломжийг олгоно.



Зураг 4.3: CNN

**4.2.2 Bidirectional Long Short-Term Memory (BiLSTM)****Үүрэг:**

- Дарааллын Мэдээллийг Боловсруулах (Sequence Modeling): BiLSTM нь CNN-ийн гаралтын онцлогуудыг өргөний дагуу дараалал болгон авч үзэж, үсгийн дарааллын хамаарлыг олж авна.

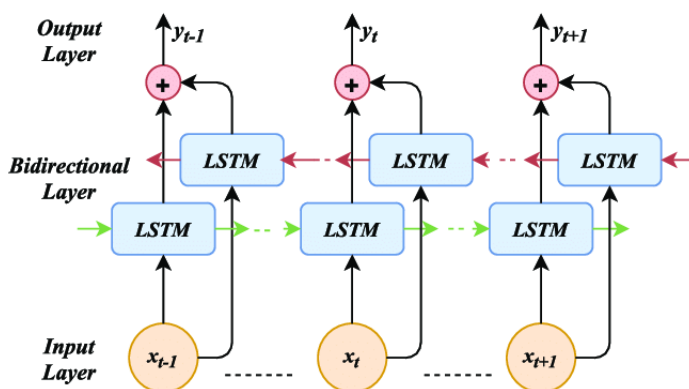
- Контекстийн Ойлголт (Contextual Understanding): Урд болон хойд чиглэлд мэдээллийг боловсруулснаар үсгийн өмнөх, дараах контекстийг харгалзан үзнэ.

**Давуу Тал:**

- Урт Хугацааны Хамаарлыг Олж Авах Чадвар: LSTM нь урт хугацааны хамаарлыг хадгалж, өмнөх мэдээллийг мартахгүйгээр дараагийн мэдээлэлтэй холбох боломжтой.

- Хоёр Чиглэлд Боловсруулах: BiLSTM нь урд болон хойд чиглэлд мэдээллийг боловсруулснаар илүү нарийвчилсан ойлголттой болно.

- Хувьсах Урттай Дарааллыг Боловсруулах Чадвар: Дарааллын урт харилцан адилгүй байж болох тул үгний урт өөрчлөгдөхөд дасан зохицоно.



Зураг 4.4: BiLSTM

#### 4.2.3 Connectionist Temporal Classification (CTC) Алдагдал

##### Үүрэг:

- Тохиргоогүй Сургалт (Alignment-Free Training): CTC алдагдал нь оролт болон гаралтын дарааллын хоорондох тохиргоогүйгээр моделийг сургах боломжийг олгоно.

- Декодлох (Decoding): Сургалтын дараа CTC нь гаралтын дарааллыг дахин боловсруулж, давхардсан тэмдэгтүүдийг нэгтгэж, '<blank>' тэмдэгтийг хасна.

##### Давуу Тал:

- Өгөгдлийн Бэлтгэлийн Хялбаршилт: Тэмдэгт тус бүрээр шошгожуулаагүй өгөгдөлтэй ажиллах боломжтой тул өгөгдөл бэлтгэх ажлыг хөнгөвчилнө.

- Гар Бичмэлийн Өөрчлөлтөд Тэсвэртэй Байдал: Үсгийн хоорондын зай, хэмжээ өөрчлөгдөхөд моделийг зөв ажиллах боломжтой болгож, гар бичмэлийн онцлогт нийцүүлнэ.

Эдгээр бүрэлдэхүүн хэсгүүдийг хослуулснаар, Монгол бичгийн гар бичмэл үгнүүдийг үр дүнтэй таних OCR системийг бүтээх боломжтой гэж үзсэн. Тус бүрийн үүрэг, давуу талууд нь моделийн гүйцэтгэлийг сайжруулж, гар бичмэлийн онцлогт тохирсон шийдлийг өгнө.

## 5. ХЭРЭГЖҮҮЛЭЛТ

### 5.1 Сургалтын орчин бэлтгэсэн нь

Моделийн сургалт болон зураг боловсруулах явцад их хэмжээний өгөгдөл ашигласан тул нэмэлтээр Google Cloud Storage ашигласан.

- **Google Cloud Storage тохиргоо**

Судалгааны ажлын өгөгдлийг хадгалах, боловсруулахад Google Cloud Storage ашигласан.

Энэхүү системд хөгжүүлэлтийн орчныг тохируулахдаа дараах алхмуудыг хийсэн:

- Google Cloud Storage API-г идэвхжүүлж, credentials буюу нэвтрэх эрхүүдийг үүсгэх.
- Bucket үүсгэж, төслийн өгөгдлийг байршуулах.
- Google Cloud дээр bucket-той холбогдохын тулд credentials JSON файл ашиглах.

Name	↓ Requests	Errors (%)	Latency, median (ms)	Latency, 95% (ms)
<a href="#">Storage Insights API</a>	1	0	3,145	4,089
<a href="#">Cloud OS Login API</a>				
<a href="#">Cloud Storage API</a>				
<a href="#">Compute Engine API</a>				

Зураг 5.1: Идэвхижүүлсэн API-ууд

- **Хөгжүүлэлтийн орчин JupyterLab болон VSCode**

Энэхүү судалгааны ажлын интеграциуд нь JupyterLab болон VSCode ашиглан хийгдсэн.

Өгөгдөл боловсруулалт болон хуваалтын ажлыг VScode ашиглан хийж гүйцэтгээд Google Cloud Storage-ийн Project ID-гаа ашиглан байршуулсан.

Харин JupyterLab-д Google Cloud Storage-тай холбогдож өгөгдөл татахдаа дараах кодыг хэрэгжүүлсэн:

```

1  from google.colab import auth
2
3  auth.authenticateuser()
4
5  from google.cloud import storage
6
7  client = storage.Client()
8
9  bucket = client.bucket('mongolianscriptdata')

```

Код 5.1: Cloud Storage-тай холбож өгөгдөл татах нь

## 5.2 Хэрэгжүүлэлт №1

### 5.2.1 CNN + CTC Loss (Convolutional Neural Network with Connectionist Temporal Classification Loss)

- 1. CNN модель

```

src > cnn_model.py > MongolianCNN > forward
1  # src/cnn_model.py
2
3  import torch
4  import torch.nn as nn
5  import torch.nn.functional as F
6
7  class MongolianCNN(nn.Module):
8      def __init__(self, num_classes):
9          super(MongolianCNN, self).__init__()
10         self.conv1 = nn.Conv2d(1, 64, kernel_size=3, padding=1)
11         self.pool = nn.MaxPool2d(2, 2)
12         self.conv2 = nn.Conv2d(64, 128, kernel_size=3, padding=1)
13         self.conv3 = nn.Conv2d(128, 256, kernel_size=3, padding=1)
14         self.conv4 = nn.Conv2d(256, num_classes, kernel_size=3, padding=1)
15         self.num_classes = num_classes
16
17     def forward(self, x):
18         # x shape: [batch_size, 1, 128, 128]
19         x = self.pool(F.relu(self.conv1(x))) # [batch_size, 64, 64, 64]
20         x = self.pool(F.relu(self.conv2(x))) # [batch_size, 128, 32, 32]
21         x = F.relu(self.conv3(x))           # [batch_size, 256, 32, 32]
22         x = F.relu(self.conv4(x))           # [batch_size, num_classes, 32, 32]
23         x = x.permute(0, 2, 3, 1)           # [batch_size, 32, 32, num_classes]
24         x = x.reshape(x.size(0), -1, self.num_classes) # [batch_size, seq_length, num_classes]
25
26         return x
27

```

Зураг 5.2: CNN модель

- **2. Сургалтын параметрууд**

- batch size = 32
- num epochs = 10
- learning rate = 0.001
- num workers = 0
- pin memory = True

- **Моделийн үнэлгээ (Evaluation)**

Evaluate скрипт нь моделийг туршиж, үнэлэх үүрэгтэй.

Үүнд моделийн танилтын түвшинг үнэлэх хэмжигдэхүүнүүд болох CER (Character Error Rate) болон WER (Word Error Rate) ашиглан үнэлгээг хийж гүйцэтгэнэ.

- *Дундаж тэмдэгтийн алдаа - CER (Character Error Rate)*

**Character Error Rate (CER):**

$$CER = \frac{\text{Number of incorrect characters}}{\text{Total number of characters in the reference text}}$$

Зураг 5.3: CER (Character Error Rate)

- *Дундаж үгийн алдаа - WER (Word Error Rate)*

**Word Error Rate (WER):**

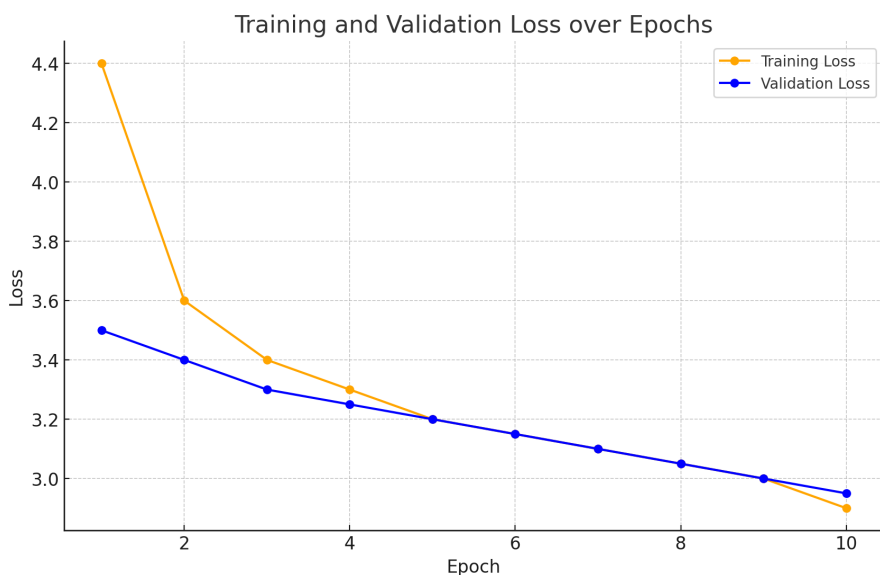
$$WER = \frac{\text{Number of incorrect words}}{\text{Total number of words in the reference text}}$$

Зураг 5.4: WER (Word Error Rate)

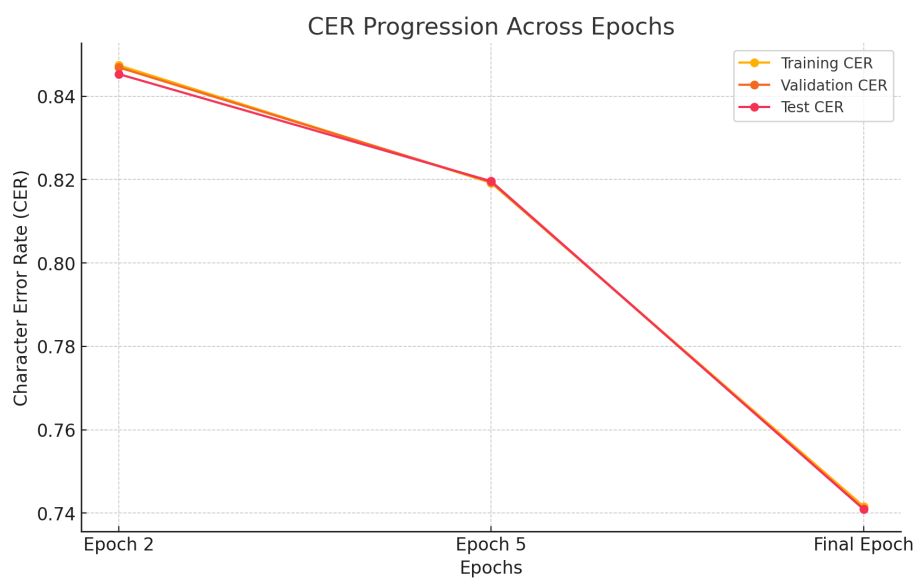
• 4. Үр дүн

– Дундаж Character Error Rate (CER): 0.7406

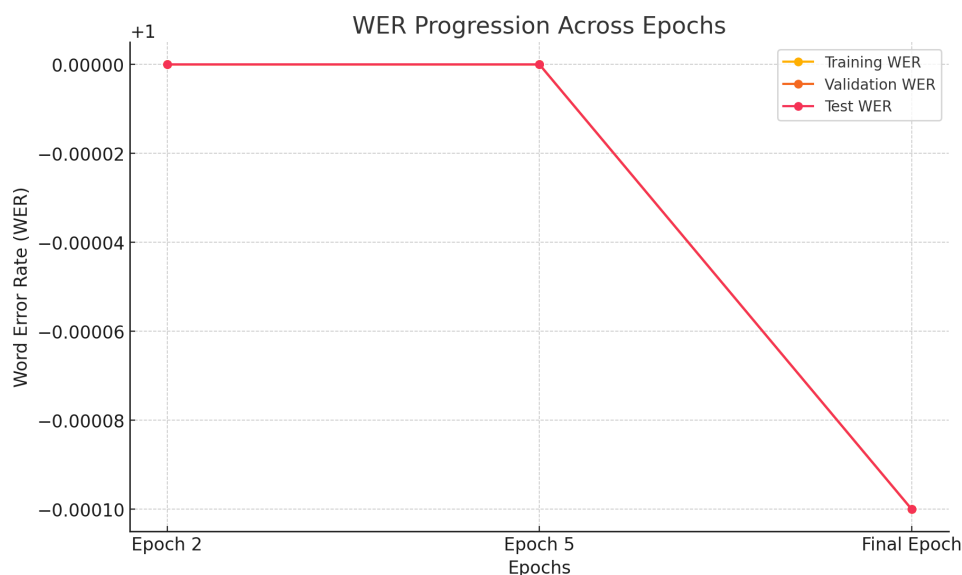
– Дундаж Word Error Rate (WER): 0.9901



Зураг 5.5: Сургалт болон баталгаажуулалтын алдагдал



Зураг 5.6: Тэмдэгтийн алдааны үнэлгээ



Зураг 5.7: Үгийн алдааны үнэлгээ

Үр дүнд Монгол бичгийн OCR модель нь шаардлагатай гүйцэтгэлийн түвшинд хүрээгүй тул уг моделийг үргэлжлүүлэн сайжруулж ажиллахаар зорьсон.

## 5.3 Хэрэгжүүлэлт №2

### 5.3.1 Сургалтын параметрууд

- batch size = 32
- num epochs = 20
- learning rate = 1e-4
- num workers (Өгөгдлийг ачааллах тооны хамгийн сайн тохиргоо)
- pin memory = True (График процессорт өгөгдлийг хурдан шилжүүлэхэд туслах параметр)



**5.3.2 CRNN моделийн сайжруулсан архитектур**

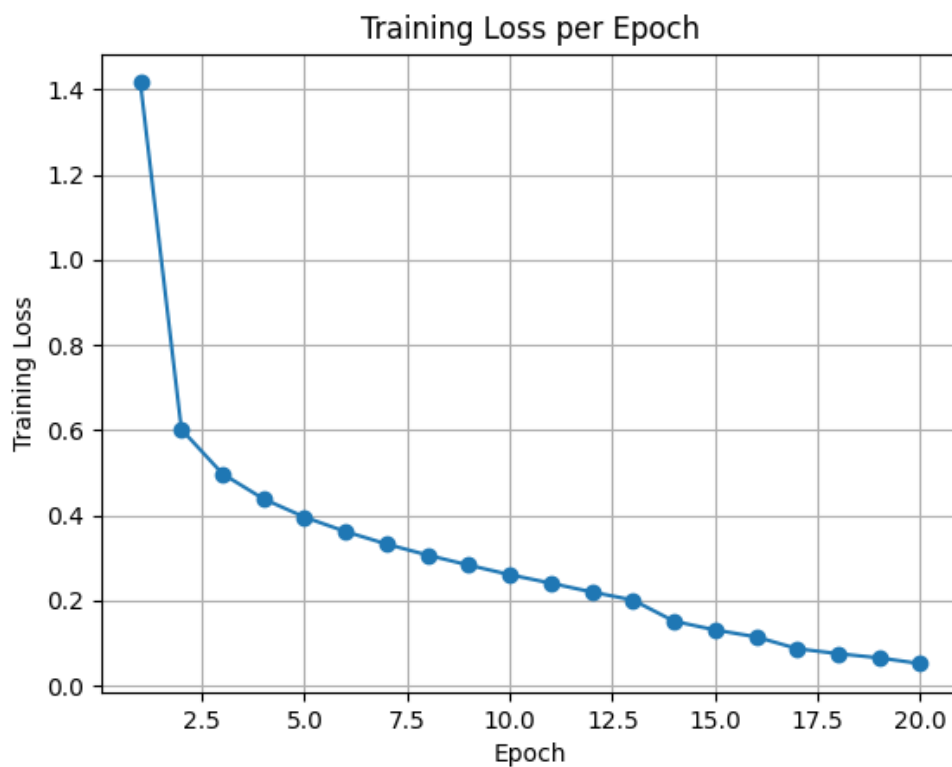
- CNN (Convolutional Neural Network) хэсэг нь зургийн онцлог шинж чанарыг авч үзэх бөгөөд 5 хэсгээс бүрдсэн:
  1. Conv2d: Текстын бүтэц болон дүрсийг танихад ашиглагддаг.
  2. BatchNorm2d: Нийтлэг дүрсийг стандартчилж, сургалтыг хурдасгахад туслана.
  3. MaxPool2d: Өгөгдлийн хэмжээг багасгаж, гол дүрсийн мэдээллийг хадгалж үлдэнэ.
- LSTM (Long Short Term Memory) нь цаг хугацааны дагуу мэдээлэл хадгалах чадвартай, тиймээс текстын үг, үсэг, хэлбэрийг танихад тохиромжтой. Fully connected layer хэсэг нь BiLSTM-ийн сүүлд гарсан гаралтыг анхдагч текстын кодоод хөрвүүлдэг.
- CTC (Connectionist Temporal Classification) Loss нь суралцах явцад үг, үсэг, эсвэл дүрсийн танилтыг хийнэ.
- Мөн өгөгдлийн багцын зургуудыг уншихдаа нэмэлтээр 90 градус хэвтүүлэх хувиргалтыг хийж өгсөн. Энэ нь Монгол бичгийн онцлогийг модельд тохируулах зорилготой сайжруулалт юм.

```
1      transform = transforms.Compose([
2          transforms.Lambda(lambda img: img.rotate(-90, expand=True))
3          ,
4          transforms.Resize((128, 128)),
5          transforms.ToTensor(),
6          transforms.Normalize((0.5,), (0.5,))
7      ])
```

Код 5.2: Зургийг боловсруулах функц

## 5.4 Үр дүн

- Дундаж Character Error Rate (CER): 0.1088
- Дундаж Word Error Rate (WER): 0.4606



Зураг 5.8: Сургалтын Алдагдал

```
Sample 1:
Ground Truth: wlamjilagci
Prediction:  wlamjilagci
CER: 0.0000, WER: 0.0000

Sample 2:
Ground Truth: wdacihajai
Prediction:  wdacihajai
CER: 0.0000, WER: 0.0000

Sample 3:
Ground Truth: cvhvreG
Prediction:  cvhureG
CER: 0.1429, WER: 1.0000

Sample 4:
Ground Truth: hemelil_e
Prediction:  hemelil_e
CER: 0.0000, WER: 0.0000

Sample 5:
Ground Truth: haGhaljahw
Prediction:  hasalajahw
CER: 0.3000, WER: 1.0000

Sample 6:
Ground Truth: mvrgvlcen_e
Prediction:  mvrgvlcen_e
CER: 0.0000, WER: 0.0000

Sample 7:
Ground Truth: dair
Prediction:  tanira
CER: 0.7500, WER: 1.0000

Sample 8:
Ground Truth: janwn_a
Prediction:  jaman_a
CER: 0.2857, WER: 1.0000

Sample 9:
Ground Truth: obogalatal_a
Prediction:  obolwl_a
CER: 0.4167, WER: 1.0000

Sample 10:
Ground Truth: uliyen_e
Prediction:  uliyen_e
CER: 0.0000, WER: 0.0000

Average CER: 0.1088
Average WER: 0.4606
```

Зураг 5.9: Туршилтын үр дүн

## 6. КОД

```
1 # src/render_images.py
2 import os
3 import json
4 from PIL import Image, ImageDraw
5 import numpy as np
6 import pandas as pd
7 def render_image_pil(coords, image_size=256):
8     #
9     image = Image.new('L', (image_size, image_size), color=255)
10    draw = ImageDraw.Draw(image)
11    pen_down = False
12    last_point = None
13    #
14    for point in coords:
15        if point == [-1, -1]: #
16            pen_down = False
17            last_point = None
18        else:
19            x, y = point
20            #
21            x_norm = int(x * (image_size - 1))
22            y_norm = int(y * (image_size - 1))
23            if last_point is not None and pen_down:
24                #
```

```

25         draw.line([last_point, (x_norm, y_norm)], fill=0, width
26                     =2)
27         last_point = (x_norm, y_norm)
28         pen_down = True
29     return image #
30
31 def save_images_pil(data_file, output_dir, labels_csv_path, max_images=
32     None):
33     #
34     if not os.path.exists(output_dir):
35         os.makedirs(output_dir)
36     labels_list = []
37     #
38     with open(data_file, 'r', encoding='utf-8') as f:
39         for idx, line in enumerate(f):
40             #
41             if max_images and idx >= max_images:
42                 break
43             line = line.strip()
44             if not line:
45                 continue #
46             parts = line.strip().split(',', 5)
47             if len(parts) != 6:
48                 continue #
49             label, author_id, screen_w, screen_h, dpi, coords = parts
50             coords = json.loads(coords) # JSON
51             #
52             coords = normalize_coordinates(coords)
53             # PIL

```

```

51         image = render_image_pil(coords, image_size=128) #
52
53         #
54         print(idx)
55         image_filename = f'sample_{idx}.png'
56         image_filepath = os.path.join(output_dir, image_filename)
57         image.save(image_filepath, format='PNG')
58         labels_list.append({'filename': image_filename, 'label':
59                             label})
60
61     # CSV
62     labels_df = pd.DataFrame(labels_list)
63     labels_csv_dir = os.path.dirname(labels_csv_path)
64     if not os.path.exists(labels_csv_dir):
65         os.makedirs(labels_csv_dir) #
66     labels_df.to_csv(labels_csv_path, index=False) # CSV
67
68 def normalize_coordinates(coords):
69     valid_points = [point for point in coords if point != [-1, -1]]
70     if not valid_points:
71         return coords
72
73     #
74     points = np.array(valid_points)
75     min_vals = points.min(axis=0) #
76     max_vals = points.max(axis=0) #
77     scale = max(max_vals - min_vals) #
78     if scale == 0:
79         scale = 1 # 0-
80
81     #
82     normalized_coords = []

```

```

77     for point in coords:
78         if point == [-1, -1]:
79             normalized_coords.append(point)
80         else:
81             norm_point = (np.array(point) - min_vals) / scale
82             normalized_coords.append(norm_point.tolist())
83     return normalized_coords #
84 if __name__ == '__main__':
85     #
86     data_file = os.path.join('..', 'data', 'MOLHW_preprocess_unicode.
87         txt')
88     output_dir = os.path.join('..', 'data', 'images')
89     labels_csv_path = os.path.join('..', 'data', 'labels.csv')
90     #
91     max_images = 164621
92     save_images_pil(data_file, output_dir, labels_csv_path) #

```

Код 6.1: Зураг рендер хийх процесс

```

1 # src/train_cnn.py
2
3 import os
4 import re
5 import torch
6 import torch.optim as optim
7 import torch.nn as nn
8 from torch.utils.data import DataLoader
9 from torchvision import transforms

```

```

10 from image_dataset import MongolianImageDataset
11 from cnn_model import MongolianCRNN
12 from utils import load_charset, custom_collate_fn
13 import pandas as pd
14 from sklearn.model_selection import train_test_split
15 from torch.amp import autocast, GradScaler # Updated import
16
17 def train():
18     # Define paths
19     script_dir = os.path.dirname(os.path.abspath(__file__))
20     data_dir = os.path.join(script_dir, '..', 'data')
21     images_dir = os.path.join(data_dir, 'images')
22     labels_csv = os.path.join(data_dir, 'labels.csv')
23     charset_file = os.path.join(data_dir, 'charset.txt')
24
25     # Define parameters
26     batch_size = 32 # Adjusted based on GPU memory
27     num_epochs = 20
28     learning_rate = 1e-4
29     num_workers = os.cpu_count() // 2 # Adjust as needed
30     pin_memory = True # Set to True to speed up data transfer to GPU
31
32     # Load character set
33     char_to_idx, idx_to_char = load_charset(charset_file)
34     num_classes = len(char_to_idx)
35
36     transform = transforms.Compose([
37         transforms.Lambda(lambda img: img.rotate(-90, expand=True)),

```



```

38     transforms.Resize((128, 128)),
39     transforms.ToTensor(),
40     transforms.Normalize((0.5,), (0.5,))
41 ])
42
43 # Read labels from CSV
44 labels_df = pd.read_csv(labels_csv, encoding='utf-8')
45
46 # Split data
47 train_df, temp_df = train_test_split(labels_df, test_size=0.2,
48                                     random_state=42)
49 val_df, test_df = train_test_split(temp_df, test_size=0.5,
50                                   random_state=42)
51
52 # Print dataset sizes
53 print(f"Training samples: {len(train_df)}")
54 print(f"Validation samples: {len(val_df)}")
55 print(f"Test samples: {len(test_df)}")
56
57 # Create datasets
58 train_dataset = MongolianImageDataset(
59     train_df, images_dir, char_to_idx, transform=transform)
60 val_dataset = MongolianImageDataset(
61     val_df, images_dir, char_to_idx, transform=transform)
62 test_dataset = MongolianImageDataset(
63     test_df, images_dir, char_to_idx, transform=transform)
64
65 # Create DataLoaders

```

```

64 train_loader = DataLoader(
65     train_dataset, batch_size=batch_size, shuffle=True,
66     num_workers=num_workers, pin_memory=pin_memory, collate_fn=
        custom_collate_fn)
67 val_loader = DataLoader(
68     val_dataset, batch_size=batch_size, shuffle=False,
69     num_workers=num_workers, pin_memory=pin_memory, collate_fn=
        custom_collate_fn)
70 test_loader = DataLoader(
71     test_dataset, batch_size=batch_size, shuffle=False,
72     num_workers=num_workers, pin_memory=pin_memory, collate_fn=
        custom_collate_fn)
73
74 # Define model, loss function, optimizer
75 model = MongolianCRNN(num_classes, imgH=128)
76 criterion = nn.CTCLoss(blank=char_to_idx['<blank>'], zero_infinity=
    True)
77 optimizer = optim.Adam(model.parameters(), lr=learning_rate)
78 scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='
    min', factor=0.5, patience=2)
79
80 device = torch.device('cuda' if torch.cuda.is_available() else 'cpu
    ')
81 print(f"Using device: {device}")
82 model.to(device)
83
84 scaler = GradScaler("cuda") # Specify device type
85

```

```

86     start_epoch = 0
87     checkpoint_dir = os.path.join(script_dir, '..', 'checkpoints')
88     latest_checkpoint = None
89
90     # Load checkpoint if available
91     if os.path.exists(checkpoint_dir):
92         checkpoints = [f for f in os.listdir(checkpoint_dir) if f.
93                        endswith('.pth')]
94         if checkpoints:
95             pattern = re.compile(r'cnn_model_epoch_(\d+)\.pth')
96             checkpoint_epochs = []
97             for f in checkpoints:
98                 match = pattern.match(f)
99                 if match:
100                     epoch_num = int(match.group(1))
101                     checkpoint_epochs.append((epoch_num, f))
102             if checkpoint_epochs:
103                 latest_epoch, latest_checkpoint = max(checkpoint_epochs
104                                                         , key=lambda x: x[0])
105                 checkpoint_path = os.path.join(checkpoint_dir,
106                                                  latest_checkpoint)
107                 # Load the checkpoint
108                 checkpoint = torch.load(checkpoint_path, map_location=
109                                         device)
110                 model.load_state_dict(checkpoint['model_state_dict'])
111                 optimizer.load_state_dict(checkpoint['
112                                           optimizer_state_dict'])
113                 start_epoch = checkpoint['epoch']

```

```

109         print(f"Resumed training from epoch {start_epoch}")
110     else:
111         print("No valid checkpoints found. Starting training from scratch.")
112     else:
113         print("No checkpoints found. Starting training from scratch.")
114 else:
115     os.makedirs(checkpoint_dir, exist_ok=True)
116     print("Checkpoint directory created. Starting training from scratch.")
117
118 # Training loop
119 for epoch in range(start_epoch, num_epochs):
120     model.train()
121     total_loss = 0
122
123     for batch_idx, (images, labels) in enumerate(train_loader):
124         images = images.to(device, non_blocking=True)
125         labels = [label.to(device, non_blocking=True) for label in labels]
126
127         optimizer.zero_grad()
128         with autocast("cuda"):
129             outputs = model(images)
130             outputs = outputs.log_softmax(2) # [sequence_length, batch_size, num_classes]
131

```

```

132     # Compute input_lengths and target_lengths
133     sequence_length = outputs.size(0)
134     batch_size_actual = outputs.size(1) # Since outputs is [
        seq_len, batch_size, num_classes]
135     input_lengths = torch.full(size=(batch_size_actual,),
        fill_value=sequence_length, dtype=torch.long).to(device)
136     target_lengths = torch.tensor([len(label) for label in
        labels], dtype=torch.long).to(device)
137     labels_flat = torch.cat(labels).to(device)
138
139     # Disable autocast for loss computation
140     with autocast("cuda", enabled=False):
141         loss = criterion(outputs.float(), labels_flat,
            input_lengths, target_lengths)
142
143     scaler.scale(loss).backward()
144     scaler.step(optimizer)
145     scaler.update()
146
147     total_loss += loss.item()
148     if (batch_idx + 1) % 100 == 0:
149         print(f"Epoch_{epoch+1}/{num_epochs}, Batch_{
            batch_idx+1}/{len(train_loader)}, Loss: {loss.
            item():.4f}")
150
151     avg_loss = total_loss / len(train_loader)
152
153     # Save the model

```

```

154     checkpoint_path = os.path.join(checkpoint_dir, f'
        cnn_model_epoch_{epoch+1}.pth')
155     torch.save({
156         'epoch': epoch + 1,
157         'model_state_dict': model.state_dict(),
158         'optimizer_state_dict': optimizer.state_dict(),
159         'loss': avg_loss,
160     }, checkpoint_path)
161
162     print(f"Epoch_{epoch+1}/{num_epochs}, Average Training Loss
        :_{avg_loss:.4f}")
163
164     # Validation step
165     model.eval()
166     val_loss = 0
167     with torch.no_grad():
168         for images, labels in val_loader:
169             images = images.to(device, non_blocking=True)
170             labels = [label.to(device, non_blocking=True) for label
                in labels]
171
172             with autocast("cuda"):
173                 outputs = model(images)
174                 outputs = outputs.log_softmax(2)
175
176                 sequence_length = outputs.size(0)
177                 batch_size_actual = outputs.size(1)
178                 input_lengths = torch.full(size=(batch_size_actual,),

```

```

        fill_value=sequence_length, dtype=torch.long).to(
            device)

179     target_lengths = torch.tensor([len(label) for label in
        labels], dtype=torch.long).to(device)
180     labels_flat = torch.cat(labels).to(device)
181
182     # Disable autocast for loss computation
183     with autocast("cuda", enabled=False):
184         loss = criterion(outputs.float(), labels_flat,
            input_lengths, target_lengths)
185         val_loss += loss.item()
186
187     avg_val_loss = val_loss / len(val_loader)
188     print(f"Epoch_{epoch+1}/{num_epochs}, Validation Loss: {
        avg_val_loss:.4f}")
189
190     # Step the scheduler
191     scheduler.step(avg_val_loss)
192
193     # Test step
194     model.eval()
195     test_loss = 0
196     with torch.no_grad():
197         for images, labels in test_loader:
198             images = images.to(device, non_blocking=True)
199             labels = [label.to(device, non_blocking=True) for label in
                labels]
200

```

```

201     with autocast("cuda"):
202         outputs = model(images)
203         outputs = outputs.log_softmax(2)
204
205     sequence_length = outputs.size(0)
206     batch_size_actual = outputs.size(1)
207     input_lengths = torch.full(size=(batch_size_actual,),
208         fill_value=sequence_length, dtype=torch.long).to(device)
209     target_lengths = torch.tensor([len(label) for label in
210         labels], dtype=torch.long).to(device)
211     labels_flat = torch.cat(labels).to(device)
212
213     # Disable autocast for loss computation
214     with autocast("cuda", enabled=False):
215         loss = criterion(outputs.float(), labels_flat,
216             input_lengths, target_lengths)
217         test_loss += loss.item()
218
219     avg_test_loss = test_loss / len(test_loader)
220     print(f"Test Loss: {avg_test_loss:.4f}")
221
222     # Save the final model
223     final_checkpoint_path = os.path.join(checkpoint_dir, '
224         cnn_model_final.pth')
225     torch.save(model.state_dict(), final_checkpoint_path)
226
227 if __name__ == '__main__':
228     train()

```



Код 6.2: CRNN (Convolutional Recurrent Neural Network) моделийг сургах скрипт

```
1 # src/evaluate_model.py
2
3 import os
4 import torch
5 from torch.utils.data import DataLoader
6 from torchvision import transforms
7 from image_dataset import MongolianImageDataset
8 from cnn_model import MongolianCRNN
9 from utils import load_charset, custom_collate_fn
10 import pandas as pd
11 from sklearn.model_selection import train_test_split
12 import editdistance
13
14 def evaluate():
15     # Define paths
16     script_dir = os.path.dirname(os.path.abspath(__file__))
17     data_dir = os.path.join(script_dir, '..', 'data')
18     images_dir = os.path.join(data_dir, 'images')
19     labels_csv = os.path.join(data_dir, 'labels.csv')
20     charset_file = os.path.join(data_dir, 'charset.txt')
21     model_path = os.path.join(script_dir, '..', 'checkpoints', '
        cnn_model_final.pth')
22
23     # Load character set
24     char_to_idx, idx_to_char = load_charset(charset_file)
25     num_classes = len(char_to_idx)
```

```

26
27 # Image transformations (same as during training)
28 transform = transforms.Compose([
29     transforms.Resize((128, 256)),
30     transforms.ToTensor(),
31     transforms.Normalize((0.5,), (0.5,))
32 ])
33
34 # Read labels from CSV
35 labels_df = pd.read_csv(labels_csv, encoding='utf-8')
36
37 # Split data
38 _, temp_df = train_test_split(labels_df, test_size=0.2,
39                                random_state=42)
40
41 _, test_df = train_test_split(temp_df, test_size=0.5, random_state
42                                =42)
43
44 # Create dataset and DataLoader
45 test_dataset = MongolianImageDataset(
46     test_df, images_dir, char_to_idx, transform=transform)
47
48 test_loader = DataLoader(
49     test_dataset, batch_size=1, shuffle=False,
50     num_workers=0, collate_fn=custom_collate_fn)
51
52 # Load model
53 device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
54
55 model = MongolianCRNN(num_classes, imgH=128)

```

```

51     model.load_state_dict(torch.load(model_path, map_location=device))
52     model.to(device)
53     model.eval()
54
55     total_cer = 0
56     total_wer = 0
57     total_samples = 0
58
59     with torch.no_grad():
60         for idx, (images, labels) in enumerate(test_loader):
61             images = images.to(device)
62             labels = labels[0] # Since batch_size=1
63             label_text = ''.join([idx_to_char[idx.item()] for idx in
64                                     labels])
65
66             outputs = model(images)
67             outputs = outputs.log_softmax(2)
68             outputs = outputs.argmax(2)
69             outputs = outputs.permute(1, 0) # [batch_size, seq_len]
70
71             pred_indices = outputs[0].cpu().numpy()
72             # Remove consecutive duplicates and blanks
73             pred_text_chars = []
74             prev_idx = None
75             for idx_char in pred_indices:
76                 if idx_char != prev_idx and idx_char != char_to_idx['<
                    blank>']:
77                     pred_text_chars.append(idx_to_char[idx_char])

```

```

77         prev_idx = idx_char
78     pred_text = ''.join(pred_text_chars)
79
80     # Calculate CER and WER
81     cer = editdistance.eval(pred_text, label_text) / max(len(
82         label_text), 1)
83
84     wer = editdistance.eval(pred_text.split(), label_text.split
85         ()) / max(len(label_text.split()), 1)
86
87
88     total_cer += cer
89     total_wer += wer
90     total_samples += 1
91
92     # Print sample predictions
93     if idx < 10: # Print first 10 samples
94         print(f"Sample_{idx+1}:")
95         print(f"Ground_Truth:_{label_text}")
96         print(f"Prediction:_{pred_text}")
97         print(f"CER:_{cer:.4f},_WER:_{wer:.4f}\n")
98
99
100     avg_cer = total_cer / total_samples
101     avg_wer = total_wer / total_samples
102
103     print(f"Average_CER:_{avg_cer:.4f}")
104     print(f"Average_WER:_{avg_wer:.4f}")
105
106 if __name__ == '__main__':
107     evaluate()

```

Код 6.3: Сургасан моделийг үнэлэх скрипт

```
1 # src/cnn_model.py
2
3 import torch
4 import torch.nn as nn
5
6 class MongolianCRNN(nn.Module):
7     def __init__(self, num_classes, imgH=128):
8         super(MongolianCRNN, self).__init__()
9
10        # Ensure imgH is divisible by 32
11        assert imgH % 32 == 0, "imgH must be divisible by 32"
12
13        # Calculate H_out
14        H_out = imgH // 32 # Integer division
15
16        # CNN layers
17        self.cnn = nn.Sequential(
18            # First convolutional block
19            nn.Conv2d(3, 64, kernel_size=3, stride=1, padding=1), # [
20                batch_size, 64, H, W]
21            nn.BatchNorm2d(64),
22            nn.ReLU(inplace=True),
23            nn.MaxPool2d((2, 2)), # Halve height and width
24
25            # Second convolutional block
26            nn.Conv2d(64, 128, kernel_size=3, stride=1, padding=1), #
```

```

    [batch_size, 128, H/2, W/2]
26     nn.BatchNorm2d(128),
27     nn.ReLU(inplace=True),
28     nn.MaxPool2d((2, 2)), # Halve height and width
29
30     # Third convolutional block
31     nn.Conv2d(128, 256, kernel_size=3, stride=1, padding=1), #
        [batch_size, 256, H/4, W/4]
32     nn.BatchNorm2d(256),
33     nn.ReLU(inplace=True),
34     nn.MaxPool2d((2, 1)), # Halve height only
35
36     # Fourth convolutional block
37     nn.Conv2d(256, 256, kernel_size=3, stride=1, padding=1), #
        [batch_size, 256, H/8, W/4]
38     nn.BatchNorm2d(256),
39     nn.ReLU(inplace=True),
40     nn.MaxPool2d((2, 1)), # Halve height only
41
42     # Fifth convolutional block
43     nn.Conv2d(256, 512, kernel_size=3, stride=1, padding=1), #
        [batch_size, 512, H/16, W/4]
44     nn.BatchNorm2d(512),
45     nn.ReLU(inplace=True),
46     nn.MaxPool2d((2, 1)), # Halve height only
47 )
48
49 # BiLSTM layers

```

```

50     self.lstm = nn.LSTM(
51         input_size=512 * H_out,  # H_out is calculated based on
52         imgH
53         hidden_size=256,
54         num_layers=2,
55         bidirectional=True,
56         batch_first=True
57     )
58
59     # Fully connected layer
60
61     self.fc = nn.Linear(256 * 2, num_classes)
62
63 def forward(self, x):
64
65     # x: [batch_size, channels, height, width]
66     x = self.cnn(x)  # [batch_size, 512, H_out, W_out]
67
68     batch_size, channels, h, w = x.size()
69
70     # Reshape and permute to prepare for LSTM
71     x = x.permute(0, 3, 1, 2)  # [batch_size, W_out, channels,
72         H_out]
73     x = x.contiguous().view(batch_size, w, channels * h)  # [
74         batch_size, W_out, channels * H_out]
75
76     # Pass through LSTM
77     x, _ = self.lstm(x)  # [batch_size, W_out, hidden_size * 2]
78
79     # Pass through fully connected layer

```

```
75     x = self.fc(x) # [batch_size, W_out, num_classes]
76
77     # Transpose for CTC loss: [W_out, batch_size, num_classes]
78     x = x.permute(1, 0, 2) # [sequence_length, batch_size,
79                             num_classes]
80
81     return x
```

Код 6.4: CRNN моделийн архитектурын тодорхойлолт



## Дүгнэлт

Судалгааны ажлын хүрээнд Монгол бичгийн гар бичмэл үгнүүдийг зургаас текст хэлбэрт хөрвүүлэх OCR (optical character recognition) моделийг хөгжүүлэх ажлыг хийж гүйцэтгэх үндсийг судлан, өөрийн хэмжээнд хэрэгжүүлж туршлаа. Эхний шатанд уламжлалт Монгол бичгээр бичигдсэн гар бичмэл үгнүүдийг дүрслэх координатын цуваа өгөгдөлд урьдчилсан боловсруулалт хийж, үг тус бүрийг зураг болгон *render*-лэж, тэдгээрийг шошгожуулан нийт 164631 дахин давтагдахгүй зургуудаас бүрдэх өгөгдлийн багцыг OCR сургалтанд зориулан бэлтгэсэн.

Хэрэгжүүлэлтийн эхний үе шатанд CNN + CTC моделийн архитектурыг сонгон авч сургалт, туршилтыг хийж гүйцэтгэхэд дан CPU ашигласны улмаас сургалтын явц хэт удаан байсан ч Дундаж Character Error Rate (CER): 0.7406, Дундаж Word Error Rate (WER): 0.9901 гэсэн үр дүн үзүүлсэн тул нэмэлт судалгаа, сайжруулалт хийн моделийн архитектурыг CNN + BiLSTM + CTC болгон өөрчилж сургалт, туршилт хийсний үр дүнд Дундаж Character Error Rate (CER): 0.1088, Дундаж Word Error Rate (WER): 0.4606 болж сайжирлаа.

Монгол бичгийн гар бичмэл OCR моделийн үндсэн архитектур CNN + CTC-д BiLSTM давхаргыг нэмж өгсөн нь үр дүнг сайжруулах нэг талын хөшүүрэг болж өгсөн бөгөөд, ялангуяа Монгол бичиг шиг үргэлжилсэн, олон төрлийн үсгийн хэлбэр бүтэцтэй бичгийг зөв нарийвчлалтайгаар танихад туслах боломжтой гэж дүгнэж байна. Монгол бичиг нь нарийн төвөгтэй бүтэц бүхий, босоо чиглэлтэй өвөрмөц бичиг учир цаашид их хэмжээний өгөгдөл болон гүн сургалтад тохирсон дэвшилтэт техник хэрэгслүүд ашиглан судалгааг өргөжүүлснээр танилтын түвшинг илүү нарийвчлалтай, найдвартай түвшинд хүргэх боломжтой болох юм.

МУИС, МКУТ-ИЙН ОЮУТАН Г.ДЭЛГЭРМААГИЙН “МОНГОЛ БИЧИГ ТАНИЛТ”  
СЭДЭВТ БАКАЛАВРЫН СУДАЛГААНЫ АЖЛЫН ШҮҮМЖ


Уг ажил нь гүний сургалтын арга зүйг ашиглан гар бичмэлээс таних арга, загвар боловсруулан сургах, турших зорилгын хүрээнд дараах ажлуудыг хийж гүйцэтгэсэн байна. Үүнд:

- Загварыг сургах, турших өгөгдлийн багцыг Kaggle дахь сангаас бэлтгэсэн,
- Гар бичмэлийг таних арга, алгоритмуудыг судалсан,
- CNN, LSTM, CTC зэрэг аргуудыг нарийвчлан судалж, харьцуулсан үнэлгээ хийхийг эрмэлзсэн,
- Ижил төстэй ажлын судалгааг хийж, CNN+biLST+CNC архитектурыг санал болгож, хэрэгжүүлэлтийг гүйцэтгэсэн,
- Санал болгож буй архитектурыг бэлтгэсэн өгөгдөл дээр сургаж, турших, зүгшрүүлэх оролдлогыг гүйцэтгэсэн.

Энэ судалгааны ажил нь практик ач холбогдолтой, судалгаа, хөгжүүлэлт хосолсон, бакалаврын судалгааны ажлын түвшинд хүрсэн ажил хэмээн үнэлж, 90% үнэлгээ өгч байна.

Санал, зөвлөмж:

- Өмнө хийгдсэн ижил төстэй ажлуудаас санаа авч моделийн танилтын хувийг сайжруулах;
- Моделийн үнэлгээний хэсгийг дэлгэрүүлэх;
- Монгол хэлний найруулгыг сайжруулах, алдааг хянах;
- Системийг онлайнд (github) байршуулах.

Шүүмж бичсэн:  Др. П. Далайжаргал

# Bibliography

- [1] Онолын судалгаа, <https://www.itransition.com/computer-vision/ocr-algorithm>
- [2] Технологийн судалгаа, <https://www.hyland.com/en/resources/terminology/data-capture/what-is-optical-character-recognition-ocr>
- [3] CER болон WER, <https://medium.com/@tam.tamanna18/deciphering-accuracy-evaluation-metrics-in-nlp-and-ocr-a-comparison-of-character-e>
- [4] MOLHW Dataset, <https://www.kaggle.com/datasets/fandaoerji/molhw-ooo>
- [5] Судалгааны ажил, <https://www.nature.com/articles/s41598-022-27267-8#Sec3>
- [6] TrOCR, <https://huggingface.co/microsoft/trocr-base-handwritten>
- [7] Zero-Shot Learning, <https://www.deepchecks.com/question/how-does-zero-shot-learning-work/>
- [8] AI, <https://cloud.google.com/learn/what-is-artificial-intelligence?hl=en>
- [9] Machine Learning, <https://www.ibm.com/topics/machine-learning>
- [10] NLP, <https://aws.amazon.com/what-is/nlp/>
- [11] Deep Learning, <https://www.ibm.com/topics/deep-learning>
- [12] Deep Learning, <https://www.ibm.com/topics/deep-learning>
- [13] CNN + CTC Pipeline, <https://www.kaggle.com/code/mbmmurad/end-to-end-pipeline-cnn-rnn-model-with-ctc-loss>