Шинжлэх Ухаан Технологийн Их Сургууль
Мэдээлэл, Холбооны Технологийн Сургууль

Төрбатын Төгөлдөр

Андройд хортой кодыг илрүүлэх

Бакалаврын төгсөлтийн ажил

Улаанбаатар хот

Шинжлэх Ухаан Технологийн Их Сургууль
Мэдээлэл, Холбооны Технологийн Сургууль

Мэдээллийн сүлжээ, аюулгүй байдлын салбар

Андройд хортой кодыг илрүүлэх

Мэргэжлийн индекс: D061904
Мэргэжил: Системийн аюулгүй байдал

Удирдагч: Док. (PhD) Б. Мөнхбаяр
Зөвлөгч: Док.(Ph.D) Ч. Эрдэнэбат
Магистр Ц. Энхтөр
Гүйцэтгэгч: Т. Төгөлдөр

Улаанбаатар хот
2018 он 01 сар

Батлав. Мэдээллийн сүлжээ, аюулгүй байдлын салбарын эрхлэгч:

..... /Док Проф. (PhD)Я. Дашдорж/

Удирдагч:

..... /Док. (PhD) Б. Мөнхбаяр/

ТӨСЛИЙН АЖЛЫН ТӨЛӨВЛӨГӨӨ

Сэдэв: "Андройд хортой кодыг илрүүлэх"

№	Ажлын бүлэг, хэсгийн нэр	Эзлэх хувь	Дуусах хугацаа
1	Удиртгал	5%	2018-10-05
2	Онолын хэсэг	25%	2018-10-30
3	Судалгааны хэсэг	30%	2018-11-15
4	Төслийн хэсэг	30%	2018-12-20
5	Дүгнэлт	10%	2018-12-25

Төлөвлөгөөг боловсруулсан оюутан: ..... /Т. Төгөлдөр/

ТӨГСӨЛТИЙН АЖЛЫН ЯВЦ

№	Хийж гүйцэтгэсэн ажил	Биелсэн хугацаа	Удирдагчийн гарын үсэг
1	Удиртгал	2018-10-05	
2	Онолын хэсэг	2018-10-30	
3	Судалгааны хэсэг	2018-11-15	
4	Төслийн хэсэг	2018-12-20	
5	Дүгнэлт	2018-12-25	

Ажлын товч дүгнэлт

.....

.....

.....

.....

.....

.....

.....

Удирдагч: ..... /Док. (PhD) Б. Мөнхбаяр/

ЗӨВШӨӨРӨЛ

Оюутан Т. Төгөлдөр-ийн бичсэн төгсөлтийн ажлыг УШК-д хамгаалуулахаар тодорхойлов.

Салбарын эрхлэгч: ..... /Док Проф. (PhD)Я. Дашдорж/

Шинжлэх Ухаан Технологийн Их Сургууль
Мэдээлэл, Холбооны Технологийн Сургууль

ШҮҮМЖИЙН ХУУДАС

Мэдээллийн сүлжээ, аюулгүй байдлын салбарын төгсөх курсийн оюутан Т. Төгөлдөр-н "Андройд хортой кодыг илрүүлэх" сэдэвт төгсөлтийн ажлын шүүмж.

1. Төслөөр дэвшүүлсэн асуудал, үүнтэй холбоотой онолын материал унших судалсан байдал. Энэ талаар хүмүүсийн хийсэн судалгаа, түүний үр дүнг уншиж тусгасан эсэх. **(0-6 оноо)**

   .....
   .....
   .....
   .....
   .....
   .....
   .....

2. Төслийн ерөнхий агуулга. Шийдсэн зүйлүүд, хүрсэн үр дүн. Өөрийн санааг гарган, харьцуулалт хийн, дүгнэж байгаа чадвар. **(0-6 оноо)**

   .....
   .....
   .....
   .....
   .....
   .....
   .....

3. Инженерийн тооцоо хийсэн, схем техник угсарч туршсан, симуляци хийсэн эсвэл програм зохиосон эсэх. **(0-6 оноо)**

   .....
   .....

. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .

4. Диплом бичих стандарт шаардлагуудыг биелүүлсэн эсэх. Төслийн бичлэгийн алдаанууд, зөв бичгийн болон өгүүлбэр зүйн гэх мэт (хуудас, зураг хүснэгтийн дугаар болон тайлбар, шрифт хольсон, хувилсан зүйл ихээр оруулсан г.м) **(0-6 оноо)**

. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .

5. Төслөөр орхигдуулсан болон алдаатай хийсэн зүйлүүд. Цаашид анхаарах хэрэгтэй зүйлүүд. **(0-6 оноо)**

. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .

6. Төслийн талаар онцолж тэмдэглэх зүйлүүд.**(0-6 оноо)**

. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .

7. Ерөнхий оноо. **(0-30 оноо)**

. . . . . .

Шүүмж бичсэн: . . . . . /Док PhD. Л. Одончимэг/

Ажлын газар: ШУТИС-МХТС МСАБ салбар

Хаяг (Утас): Баянзүрх дүүрэг 22-р хороо ШУТИС-ийн 6-р байр. 70161333

Цаг зав гарган дипломтой бүрэн танилцаж үнэтэй шүүмж гаргасан танд баярлалаа. Сургууль дээр мөрдөж байгаа тарифын дагуу шүүмжилсний хөлсийг олгох болно.

Мэдээллийн сүлжээ, аюулгүй
байдлын салбарын эрхлэгч . . . . . /Док проф. PhD. Я.Дашдорж/

2018.12.25

Зохиогч эрхийн хамгаалал

Миний бие Т. Төгөлдөр, "Андройд хортой кодыг илрүүлэх" сэдэвт энэ ажил нь минийх бөгөөд дараахыг нотолж байна. Үүнд:

- Горилогч энэ ажлыг тус сургуулиас боловсролын зэрэг авахаар бүхэлд нь буюу голлон хийсэн болно.
- Энэ ажлын аль нэг хэсгийг тус сургуульд эсвэл өөр байгууллагад боловсролын зэрэг, мэргэшил авахаар өмнө нь илгээсэн бол түүнийгээ тодорхой заасан болно.
- Бусад хүмүүсийн хэвлүүлсэн ажлаас зөвлөгөө авсан бол түүнийгээ үндэслэсэн болно.
- Бусад хүмүүсийн ажлаас ишлэл хийхдээ гол эх үүсвэрийг нь заасан болно.
- Миний ажилд тусалсан голлох бүх эх үүсвэрт талархаж байна.
- Ажлыг бусадтай хамтарсан бол алийг нь бусад хүмүүс хийсэн болохыг тодорхой заасан болно.

Гарын үсэг: _____

Огноо: 2019.01.07

Шинжлэх Ухаан Технологийн Их Сургууль
Мэдээлэл, Холбооны Технологийн Сургууль

Хураангуй

Андройд хортой кодыг илрүүлэх

Т. Төгөлдөр
B140970165@sict.edu.mn

Андройд үйлдлийн систем нь утасны үйлдлийн системүүд дотроо хамгийн өргөн хэрэглэгддэг үйлдлийн систем юм. Тийм ч учраас үүнд харъяалагдах хортой програм, вируснууд хэрэглэгдэж буй төхөөрөмж буюу утаснуудын тоогоор өсөн нэмэгдэж байна. Энэ нь орчин үед томоохон асуудлуудын нэг болоод байгаа бөгөөд үүний эсрэг төрөл бүрээр арга хэмжээ авсаар байна. Хортой програмыг илрүүлэх нь ерөнхийдөө тухайн эх програм дээр шинжилгээ хийж цуглуулсан мэдээлэл дээр суурилан хортой болон хоргүй гэж ангилах зарчим юм. Энэхүү төгсөлтийн судалгааны ажлаараа би Гүний мэдрэлийн эсийн сүлжээг/Deep Neural Network/ хортой програм илрүүлэхэд ашиглаж, түүний үр дүнг гаргахын тулд Андройд апп-уудын цуглуулгыг хортой болон хоргүй гэж ялгах ажиллагаан дээр төвлөрч ажиллалаа.

Гүний мэдрэлийн эсийн сүлжээ нь төвөгтэй болон уян хатан шинж чанарууд дээр үндэслэн суралцаж хортой програмыг цаг алдалгүй, үр дүнтэйгээр илрүүлэх боломжтой хиймэл оюун дээр суурилсан технологи юм.

Апп-уудын цуглуулгыг ялгах болон хортой програмыг илрүүлэх туршилтуудыг гүйцэтгэхдээ Конволюшнал эсийн сүлжээ буюу CNN-г загварчилсан бөгөөд CNN давхаргуудаа харьцуулахдаа Лонг-Шорт терм Мемори буюу Гекуррент эсийн сүлжээ болон н-грам дээр суурилсан аргуудыг ашиглан харьцуулалт хийсэн болно.

CNN болон LSTM хоёр нь хоёулаа n-gram арга дээр суурилан илүү нарийн бүтэцтэй, зөв зохион байгуулалттай ажилладаг алгоритм буюу шийдэл ашигладаг нэгэн төрлийн технологи юм. Миний тооцоолон загварчилж буй CNN давхарга нь өгөгдлийн цуглуулга буюу манай тохиолдолд апп-уудын цуглуулгад зориулж энгийн бүтэцтэйгээр ангилал хийх чадвар бүхий LSTM давхаргаас илүү буюу давуу талтай ажиллагаатай юм.

Зорилго

Энэхүү төгсөлтийн төслийн ажлын зорилго нь Андройд програмыг Реверс инженеринг буюу APK файлыг задлан програмын эх кодон дээр Гүний судалгааны арга ашиглан Андройд програмыг хортой болон хоргүй гэж ангилах, машин суралцах процесс буюу мэдрэлийн эсийн сүлжээн дэх конволюшнал давхаргыг зөв зохистойгоор/Tune-Up/ зохион байгуулах, эцсийн үр дүнг үзүүлж чадах програмыг бүтээхийг зорьсон болно.

Зорилт

Андройд хортой програмыг Deep Neural Network ашиглан илрүүлэн, ангилах ажлын хүрээнд хийгдэх зорилтууд нь:

- Өгөгдлийг бэлдэх
  Хэрэгжүүлтэнд шаардлагатай өгөгдлүүдийг цуглуулах буюу олон тооны APK файл цуглуулж бэлдэх
- Бэлдсэн өгөгдлүүдийг задлаж шинжлэхэд бэлдэх
  Олон тооны APK файлуудыг нийлүүлж жагсаалт буюу Sequence үүсгэж Python дээр хөгжүүлэгдсэн Androguard tool-г ашиглаж задлах
- Задласан өгөгдөл дээр шинжилгээ хийж ангилахад бэлдэх
  APK файлаас задласан .dex өргөтгөлтэй файлын бүтцийг X болон Y хэмжээстэй матрицад хөрвүүлэн Neural Network сүлжээгээр дамжуулан ангилах үйл явцад бэлдэх
- Конволюшнал эсийн сүлжээг зохион байгуулж програмчлах
  CNN1, CNN2, CNN3 гэсэн гурван давхарга бүхий машин сургах үйл явцыг бүрдүүлэх
- Наейв бэйс ангилагч
  Дээрх гурван давхаргаар дамжаад гаралтын давхаргаас гарсан өгөгдлийг Gaussian Naive Bayes ангилах алгоритм ашиглан хортой эсвэл хоргүйгээр нь ангилж эцсийн үр дүнг харна.

Талархал

Энэхүү дипломын ажлыг хийж гүйцэтгэхэд мэргэжлийн зүгээс арга арга-чилгааг зааж зөвлөн, тусалж дэмжсэн удирдагч багш Док.(PhD) Б.Мөнхбаяр, зөвлөх док. (Ph.D) Ч. Эрдэнэбат, магистр Ц. Энхтөр багш нартаа чин сэтгэлийн талархал илэрхийлэн, цаашдын сурган хүмүүжүүлэх үйлсэд нь амжилт хүсье.

Мөн бүх зүйлд намайг дэмжиж, миний төлөө сэтгэл зүрхээ зориулдаг аав Төрбат, ээж Цэнгэлмаа нартаа чин сэтгэлийн талархал илэрхийлж байна.

Товчилсон үгс

CNN	**C**onvolutional **N**eural **N**etwork
DNN	**D**eep **N**eural **N**etwork
LSTM	**L**ong **S**hort **T**erm **M**emory
API	**A**pplication **P**rogramming **I**nterface
APK	**A**ndroid **P**ackage
SVM	**S**upport **V**ector **M**achine
KNN	**K** - **N**earers **N**eighbor

Гарчиг

Зургийн жагсаалт

Хүснэгтийн жагсаалт

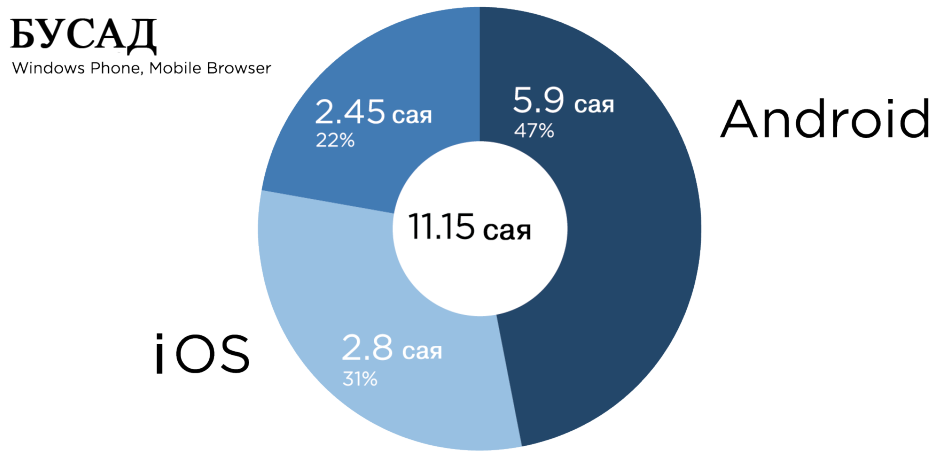
БҮЛЭГ 1

Онолын хэсэг

Орчин үед мэдээллийн технологийн салбарт ч гэлтгүй үүнийг хэрэглэж буй хүн бүхэн хиймэл оюун, их хэмжээний өгөгдөл/Big Data/, өгөгдлийн шинжлэх ухаан/Data Science/ гэх мэт цоо шинэ технологиудын тухай ярих болсон билээ. Гэвч хиймэл оюун цаашлаад Deep Learning нь хэрхэн ажиллаж байна вэ ? Гол цаад зарчим нь юун дээр үндэслэж байгаа эсэх дээр мэдлэг, ухагдахуун дутмаг байгаа юм. Энэхүү судалгааны ажлын эхний хэсэг буюу онолын хэсэг нь дээрх бүх асуултанд хариу өгч, тодорхой хэмжээний мэдлэгийг өгөх зорилгоор мэдээллийн технологийн анхан шатны мэдлэггүй хүнд ч ойлгогдохоор тайлбарлахыг хичээсэн болно.

1.1 Ерөнхий ойлголт

2017 оны байдлаар 1.53 тэрбум ухаалаг утас хэрэглэгчид байна. Судалгаагаар ухаалаг утасны хэрэглээ нь жилд 2.8 хувиар өсөж байгаа учир 2020 он гэхэд ухаалаг утас хэрэглэгчдийн тоо 1.68 тэрбум болно гэсэн таамаглал байна. Бүх ухаалаг утасны 86 хувь нь Андройд үйлдлийн систем бүхий утас юм. 2018 оны судалгаагаар гар утсан дээр суурилсан Google Playstore дахь хортой програм хангамжын тоо 2015 оноос 2017 оны хооронд 388 хувиар өссөн тоо баримт байдаг. Зураг 1.1-д дэлхий дээрх нийт хүн амын хэрэглэж буй гар утаснуудын тоо, хоорондын харьцааг/ширхэгийн хувьд/ харуулж байна.



Зураг 1.1: Ухаалаг утасны үйлдлийн системүүд ба зах зээлд эзлэх хувь

Ухаалаг утасны зах зээлд эзлэх хувийг жагсаалт байдлаар харуулвал:

- Андройд 86 %
- IOS 12.6 %
- Microsoft 1.6 %
- Бусад 0.8 %

Андройд бол мэдрэгчтэй дэлгэц бүхий зарим ухаалаг гар утас ба таблет зэрэг хөдөлгөөнт хэрэгслүүдэд зориулан бүтээгдсэн, линукс буюу нээлттэй эхийн үйлдлийн системд тулгуурласан үйлдлийн систем юм. Андройд үйлдлийн систем нь нээлттэй эх бөгөөд Жава програмчлалын хэлд үндэслэгдэн хийгдсэн. Анхны андройд ашигласан утас 2008 оны 10-р сард зарагдсан болно.

Андройдыг Inc хөгжүүлдэг ба Google 2005 онд худалдан авч түүнээс хойш хөгжүүлсээр байна. Андройд нь 2007 он хүртэл олон нийтэд ил гараагүй байсан ба нээлттэй гэрээний ачаар харилцаа холбооны компаниуд гар утсанд үйлдлийн систем болгон ашиглаж эхлэсэн.

Андройд үйлдлийн систем дээр ажиллах програмыг аппликейшн гэж нэрлэдэг бөгөөд Google Play Store гэсэн дэлгүүртэй. Андройд програмын өргөтгөл нь *.apk юм. Эдгээрийг C, C++, Java хэл дээр хийдэг боловч хэрэглэгчийн интерфейсийг жава дээр хийнэ. Одоогоор 900000 гаруй аппууд google дэлгүүр дээр бэлэн байна.

2018 оны байдлаар андройд үйлдлийн систем хэрэглэгчид нь 1.61 тэрбум болтлоо өссөн судалгаа байдаг.

Андройд програмын хөгжүүлэлтийн орчин

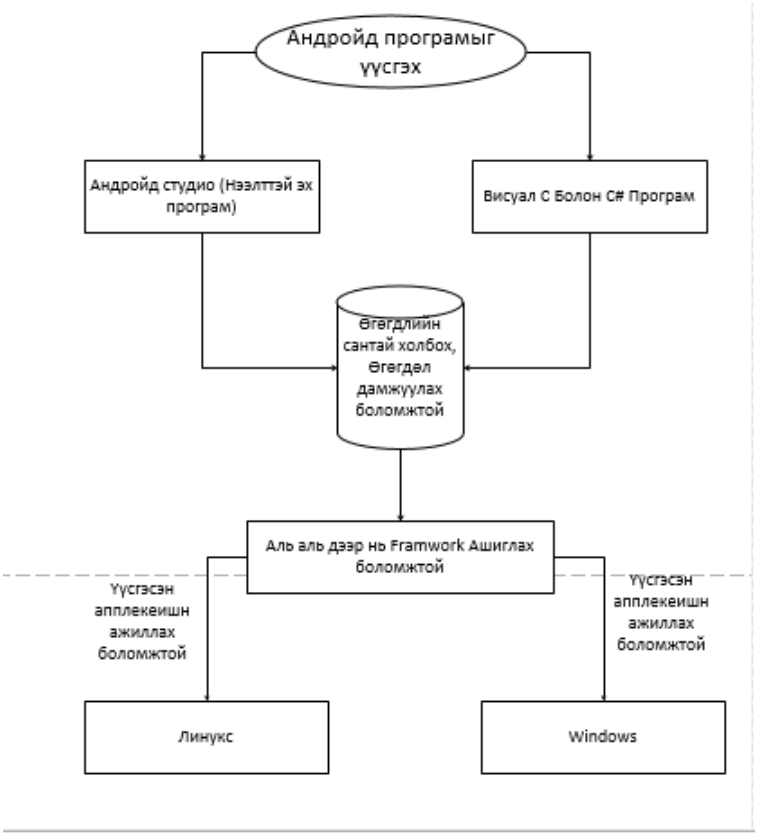
Андройд үйлдлийн систем бүхий хөдөлгөөнт төхөөрөмж дээр install хийж ажилладаг *.apk файл гэж байдаг. *.apk файл нь C, Java хэлнүүд дээр бичихэд хамгийн тохиромжтой юм. Дээд түвшиндээ Linux, Windows гэх мэт үйлдлийн систем дээр хөгжүүлэлт явагддаг.

Андройд хөгжүүлэлтийн орчины бүтээмжийг дээшлүүлэх илүү сайн програмыг ашиглахын тулд Андройд-г дэмждэг дараах түүлүүдээс хэрэглэж болно. Эдгээр түүлүүд нь програм хөгжүүлэлтийн үед хугацааг хэмнэх болон, илүү дэвшилтэт програмыг бүтээхэд тус болно. Зураг 1.2-т андройд програмыг хөгжүүлэх шатлалыг маш энгийнээр схемчилж харуулсан болно. Үүнээс харахад андройд програмыг виндовс болон линукс гэсэн үйлдлийн системүүд дээр хөгжүүлэх боломжтой байна.

1.1.1 Андройд *.Apk файлын бүтэц, ажиллагаа

*.APK файл гэдэг нь Android package kit гэсэн үгний товчлол бөгөөд андройд апп-ыг түгээх суулгахад ашигладаг файлын формат юм. Нэг үгээр бол таны windows үйлдлийн систем дээрх exe файл гэсэн үг юм. *.Apk файлыг ашиглан хөдөлгөөнт төхөөрөмж дээр апплекейшн суулгахыг side loading гэж нэрлэдэг.

***.Apk файлын ажиллагаа**

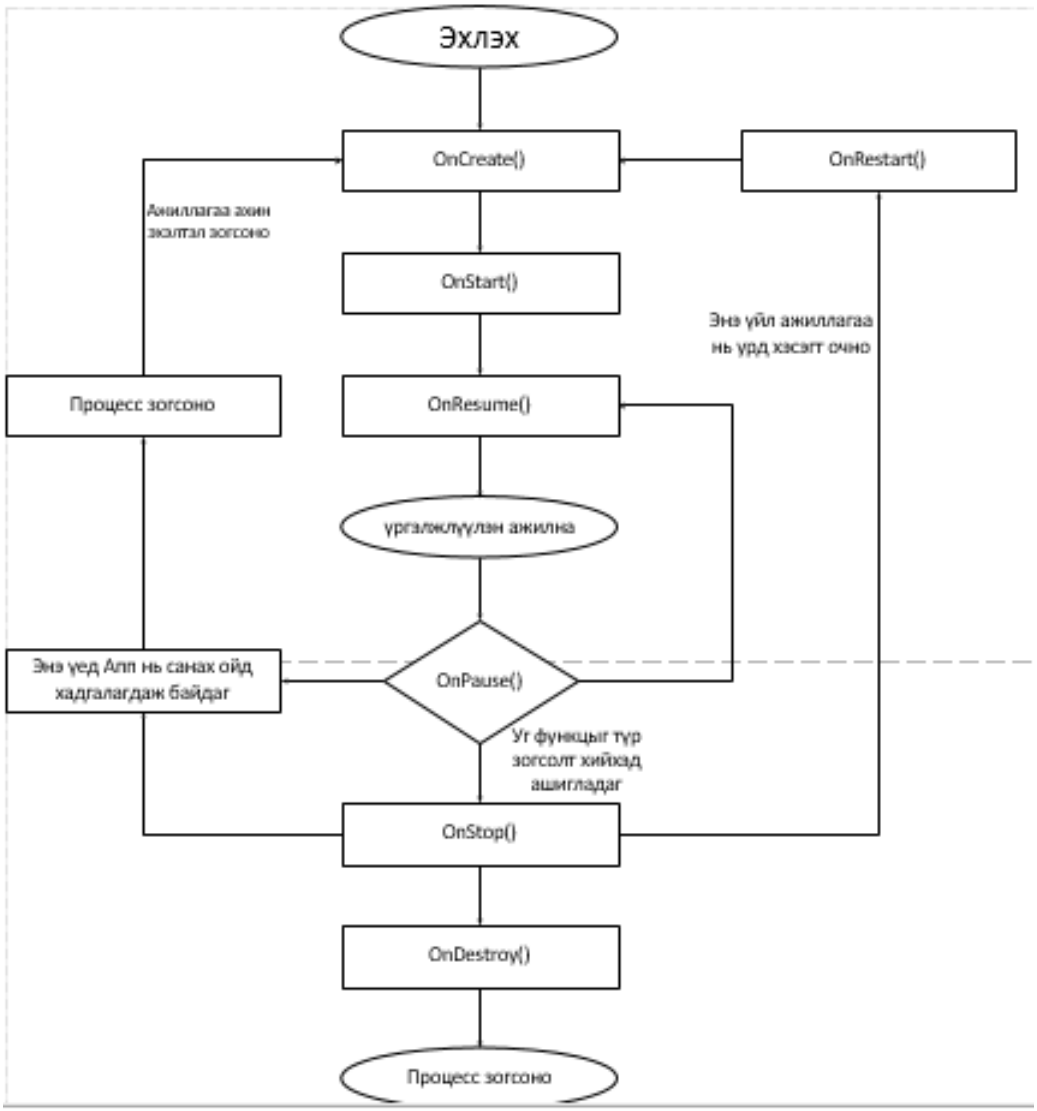


Зураг 1.2: Андройд програм хөгжүүлэлтийн явц

*.Apk файл нь ажиллахдаа эхлэх, зогсох, түр зогсох буюу pause Функцуудтай байдаг. Уг үндсэн функцуудын тусламжтайгаар application нь хэрэглэгчийн хүссэнээр ажиллаж, түр зогсолт хийж үргэлжилж байдаг. Одоогийн нөхцөлд андройд *.Apk файл нь улам бүр хөгжиж байгаа бөгөөд ажиллагаа нь илүү дэвшилтэд, хэрэглээнд нийцтэй болсоор байгаа билээ. *.Apk файл нь зөвхөн Андройд үйлдлийн систем бүхий утас дээр ажилладаг. Зураг 1.3-т андройдын гол чухал функцуудыг хэрхэн ашигладаг болон хэрэглээний жишээг схемчилж харуулсан болно. Заавал уг схемийн дагуу ажиллах шаардлагагүй бөгөөд тухайн програмын бүтэж ажиллагаанаас хамаарч өөрчлөгдөх нь мэдээж юм.

- OnCreate() - Энэ нь хамгийн эхэнд дуудагдах функц бөгөөд app-ийг анх үүсэх үед нь дууддаг.
- OnStart() - Энэ үйлдлийг хэрэглэгчид анх харагдах үед дууддаг.
- Resume() - Хэрэглэгч програмтай харилцаж эхлэх үед дууддаг функц юм.
- onPause()-Хүлээгдэж буй үйлдэл нь хэрэглэгчийн оролтыг хүлээн авдаггүй бөгөөд ямар ч кодыг ажиллуулж чадахгүй.Одоогийн үйл ажиллагааг түр зогсоож, өмнөх үйл ажиллагаа сэргэж эхэлдэг.

- onStop ()-Энэ үйлдэл нь цаашид хэрэглэгчид харагдахаа болиход дуудагдана.
- onDestroy ()-Энэ үйлдлийг системээр устгахаас өмнө дууддаг.
- onRestart()-Энэ үйлдлийг процессөр зогссоны дараа үйл ажиллагаа дахин эхлэхэд дууддаг.



Зураг 1.3: APK файлын ажиллагаа

***.Apk файлын бүтэц**

*.Apk файл нь андройд программд шаардлагатай мөн ажиллуулах бүх файлуудыг өөр дээрээ агуулж байдаг энгийн шахагдсан файл юм. *.Apk файлын бүтэцээс дурдвал :

- META-INF - Манифест файл, гарын үсэг, архив дахь нөөцийн жагсаалт агуулдаг. Үүнд *.Apk болон бусад холбогдох файлуудын гэрчилгээ , гарын үсэг нь байж болно. Андройд үйлдийн систем нь уг апплекеишныг

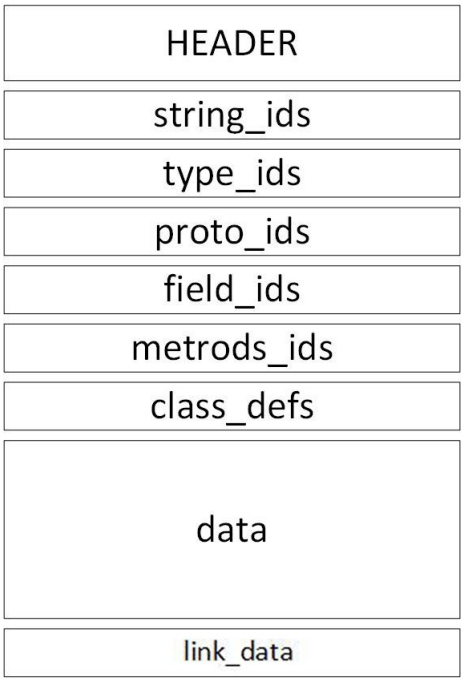
зөв гарыг үсэггүйгээр суухыг зөвшөөрдөггүй. *.Apk доторх ямар нэгэн файлд бага зэрэг өөрчлөлт орсон бол гарын үсэг өөрчлөгдөж анхны файлын хувийн түлхүүрийг ашиглахгүй бол хэнч буцааж үндсэн файлыг сэргээж чадахгүй болно. Хэрвээ бүх *.Apk файл итгэмжлэгдсэн сертипкаттай болж чадвал хорт програмыг шинжилэхгүйгээр хортой эсхийг ялгаж болно.

- LIB - Тодорхой төхөөрөмжүүдийн архитектур дээр ажилладаг төрөлжсөн сангууд (armeabi-v7a, x86, гэх мэт).
- RES - Нөөцүүд байрладаг. Жишээлвэл зураг гэх мэт хөрвүүлэгдээгүй файлууд байдаг.
- AndroidManifest.xml - *.Apk файл -ийн нэр, Version, агуулга зэргийг тодорхойлно. *.Xml файл мөн уг хавтаст хадгалагдаж байдаг.
- classes.dex - Төхөөрөмжүүд дээр ажиллах хөрвүүлэгдсэн жава class, exe файл гэж ойлгож болно. *.Dex файл нь нэг *.Apk дотор хэд хэд байх боломжтой. Үүнд бүх bytecode багтадаг.
- resources.arsc - Application -д хэрэглэдэг зарим файлууд
- Asset - asset гэдэг нь *.Apk файл-ийн нэг хэсэг бөгөөд таны хэрэгцээтэй байгаа зураг текст файл зэргийг хадгалж байдаг. Зураг 1.4 нь андройдын APK багц доторхи файлуудыг харуулсан байна.

Зураг 1.4: APK файлын бүтэц

***.Dex файл (classes.dex файл)**

*.Dex нь Dalvik Executable Format гэсэн үгний товчлол бөгөөд үндсэндээ программын логик юм. Өгөгдсөн программын кодыг Java дээр бичээд дараа нь class файлруу хөрвүүлээд VM format-д хөрвүүлдэг. Уг файлыг устгаж болохгүйгээс гадна таны хөдөлгөөнт төхөөрөмж дээр ажилладаг үндсэн файл юм. Зураг 1.5-т .dex файлын үндсэн дотоор бүтцийг харуулж байна. Энэхүү хэсэг нь андройд програмын гол хөдөлгүүр хэсэг гэж ойлгож болно.



Зураг 1.5: *.Dex файл -ийн бүтэц

***.Dex файлын шинж чанарууд**

- Санах ойн сул хадгалалтын багасгах
  - LEB 128 кодчлол
  - Харьцангуй индексжүүлэлт
  - Бүх class уудын дунд нэг файл үүсгэх
  - Давхардсан тэмдэгт мөр байхгүй
- Modified UTF-8 String кодчилол
- Зориулалтын хатуу шаардлага тавигддаг
- Ажиллах хугацааг илүү хатуу баталгаажуулдаг

Доорх хүснэгтэд бүтэцийн тодорхойлолтууд болон тэдгээрийн холбоотой нэмэлт өгөгдлүүдийн багцыг агуулж буй *.Dex файлын байршил, агуулгыг тодорхойлсон болно. *.Dex файлын өгөгдлийн хэсэгт data section нь header хэсэгт код section нь нуугдсан байдаг. Толгойн хэсгийн метход хэсэгт класс дефф нь кодын хэсгийг тодорхойлсон байдаг.

Нэр	Формат	Тайлбар
header	header-item	Толгойн хэсгүүд байрлана
string ids	string id item[]	Мөрийг тодорхойлох жагсаалт. Эдгээр нь дотоод файл (ж.нь төрөл, тодорхойлогч) эсвэл кодоор илэрхийлэгдэх байнгын объектуудад зориулсан энэ файлд ашигладаг бүх тэмдэгт мөрүүд багтана. Энэ жагсаалт нь UTF-16 кодын утгыг ашиглан string байдлаар эрэмблэдэг.Ямар ч давхардсан бичлэг агуулаагүй байх ёстой.
type ids	type id item[]	Төрлийг тодорхойлох жагсаалт. Эдгээр нь файлд тодорхойлсон эсэхээс үл хамааран бүх төрлийг (ангилал, массив, эсвэл хувьсагчийн төрлийн хувьд) тодорхойлогчид юм. Энэ жагсаалтыг string id индексээр эрэмбэлэх ёстой бөгөөд ямар ч давхардсан бичлэг оруулаагүй байх ёстой.
proto ids	proto id item[]	Энэ жагсаалтыг return-type (type-id index) том эрэмбийн дагуу эрэмбэлэх ба дараа нь аргументуудаар (бас type id индексээр) эрэмбэлдэг. Эдгээр нь файлын нэрлэсэн бүх протоколын төрлүүдэд зориулсан тодорхойлогчид юм.
field ids	field id item[]	Талбар тодорхойлогч жагсаалт. Эдгээр нь файлд тодорхойлсон эсэхээс үл хамааран энэ файлд хамаарах бүх талбарт зориулсан талбар юм. Энэ жагсаалтыг эрэмбэлэхдээ type id, string id index, type id зэрэгээр эрэмбэлнэ.
method ids	method id item[]	Энэ нь файлд зориулан гаргасан Method -ууд тодорхойлох жагсаалт юм. type id, string id, proto id зэрэг багануудад эрэмбэлдэг.
class defs	class def item[]	class тодорхойлох жагсаалт. Өгөгдсөн классууд нь өөрөөсөө өмнө үүсгэгдсэн гэх мэт олон классуудын өмнө гарч ирэх шаардлага гардаг тул тус талбарыг тодорхойлдог.
data	ubyte[]	өгөгдлийн талбар, дээр жагсаагдсан хүснэгтэд зориулсан бүх өгөгдлийг агуулсан.
link data	ubyte[]	Бусад холбоотой файлд ашигласан өгөгдөл үүд

Хүснэгт 1.1: *.Dex файл-ийн ерөнхий байршил

***.Apk файлын хэмжээ**

Application-ийн хандалт, хүмүүсийн үнэлэх үнэлгээ нь янз бүрийн хүчин зүйлээс хамаарч байдаг. Эдгээрээс хамгийн чухал байж болохуйц нь application-ийн хэмжээ юм. Энгийн хэрэглэгчид их хэмжээтэй программыг татаж авахаас төвөгшөөх, мөн утасны багтаамж зэрэг үүнд нөлөөлнө. Иймээс applicaion хөгжүүлэгчид *.Apk файлыг аль болох бага хэмжээтэй байлгахыг эрмэлзсээр ирсэн. Одоогын байдлаар Google play дээр тавигдаж байгаа app-ийн дээд хэмжээ нь 100MB байна. Хэрэв үүнээс дээш хэмжээтэй файл хадгалхыг хүсвэл өргөтгөлийн файлууд ашиглаж болно. Өргөтгөлийн файл нь 2GB хүртэлх хэмжээтэй байж болдог.

- Хамгийн их хэмжээ нь 100MB - андройд хувилбар нь 2.3 буюу түүнээс дээш (API -ийн төвшин нь 9-10 ба 14+ байна.)
- Хамгийн их хэмжээ нь 50MB - андройд хувилбар нь 2.2 буюу түүнээс доош (API -ийн төвшин нь 8 буюу түүнээс доош)

```
1 MAX_APK_SIZE_MB = 50;
2
3 Public static void accountForProgress()
4 {
5   MAX_APK_SIZE_MB = 100;
6 }
```

Та *.Apk файлынхаа хэмжээг томосгох боломжтой боловч энэ нь төдийлөн шаардлаггүй зүйл юм. Доорх хүчин зүйлүүдээс хамаарна.

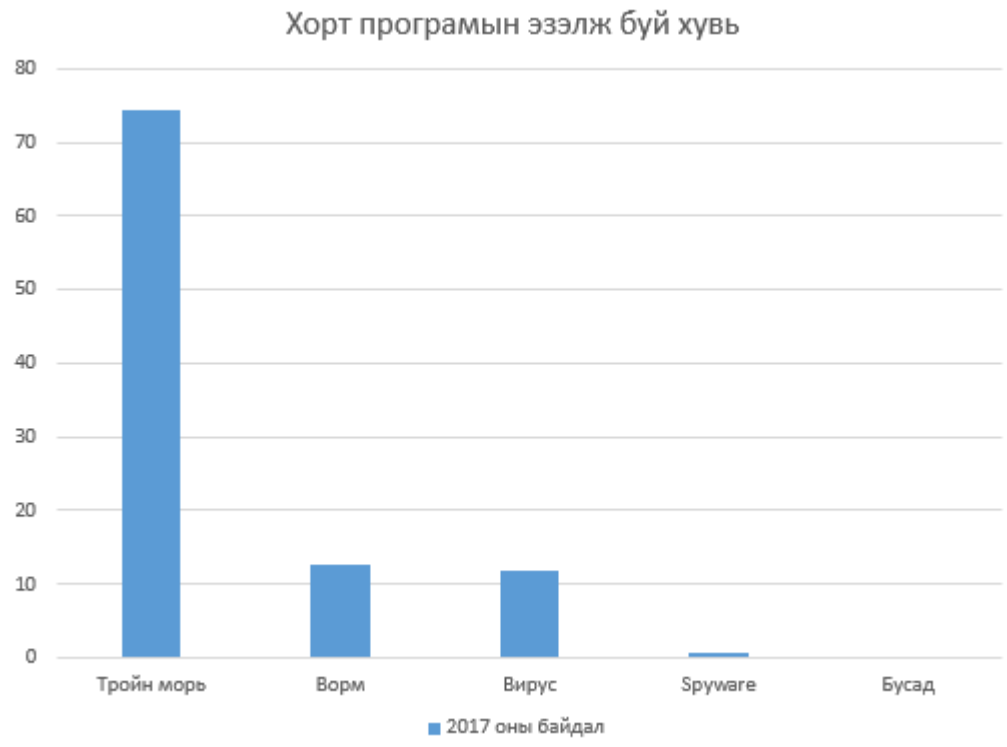
- Мобайл дата холболт - Дэлхий даяар хэрэглэгчид хөдөлгөөнт өгөгдлийн холболтын хурдыг янз бүрээр өөрчилдөг. Ялангуяа хөгжиж буй орнуудад АНУ, Япон зэрэг орнуудын хэрэглэгчдийнхээ тоотой харьцуулахад олон хүн онлайнаар холбогдож байдаг. Ийм холболт дээр байгаа хэрэглэгчид *.Apk файлаа татаж авахын тулд удаан хугацаа шаардагдаж апп эсвэл тоглоомыг суулгах магадлал бага болно.
- Гар утасны төлбөрүүд - Дэлхий даяар олон тооны гар утасны сүлжээнүүд хэрэглэгчдэд хязгаарлагдмал тооны MB-г өгдөг бөгөөд тэд сар бүр татаж авах боломжтой нэмэлт төлбөргүй хэмжээг заадаг. Ийм үед хэрэглэгчид нь их хэмжээний файлуудыг татаж авахаас болгоомжилдог.
- Апп-ийн гүйцэтгэл - Хөдөлгөөнт төхөөрөмжүүд хязгаарлагдмал RAM болон хадгалах зайтай байдаг. Ийм учраас их хэмжээтэй апп ууд нь гацаах болон Hard дискийг дүүргэх аюултай. Install time - Таны

application их хэмжээтэй байснаар удаан хугацаанд install хийгдэх болно. Энэ нь хэрэглэгчдийн залхаах аюултай.

Хэрэв энгийн app-ууд их хэмжээтэй байх юм бол хорт програм байх боломжтой юм.

1.1.2 Андройд хортой програм

Хорт програм нь програм хангамжийн хувьд богино бичиглэлтэй, компьютерийн үйл ажиллагааг таслан зогсоож чухал мэдээлэл цуглуулан, хувийн компьютерийн системд хандах боломжийг олж авах, эсвэл хүсээгүй зар сурталчилгааг ашиглан хандалтын эрх олж авдаг програм хангамж юм. Компьютерийн вирус бол нэг төрлийн malware юм. Вирусээр халдсан файлыг нээж хэрэглэхэд, өөрийгөө хувилж хувилсан хувилбаруудаа компьютерийн бусад програм, дата файл, хатуу дискийн бүүт секторт тархаж халддаг юм. Өөрөөр хэлвэл компьютерт зөвшөөрөлгүйгээр нэвтрэх чадвартай, тодорхой үүрэг, зорилготой програмыг вирус буюу хорт програм гэнэ. Үүний зонхилох хувийг Тройн морь , Worm зэрэг эзэлдэг.



Зураг 1.6: Хорт програмын эзлэх хувь

Андройд хортой програм гэдэг нь андройд үйлдлийн систем бүхий гар утас эсвэл утасгүй хөдөлгөөнт төхөөрөмжийг чиглэсэн хорт програм юм. Энэ

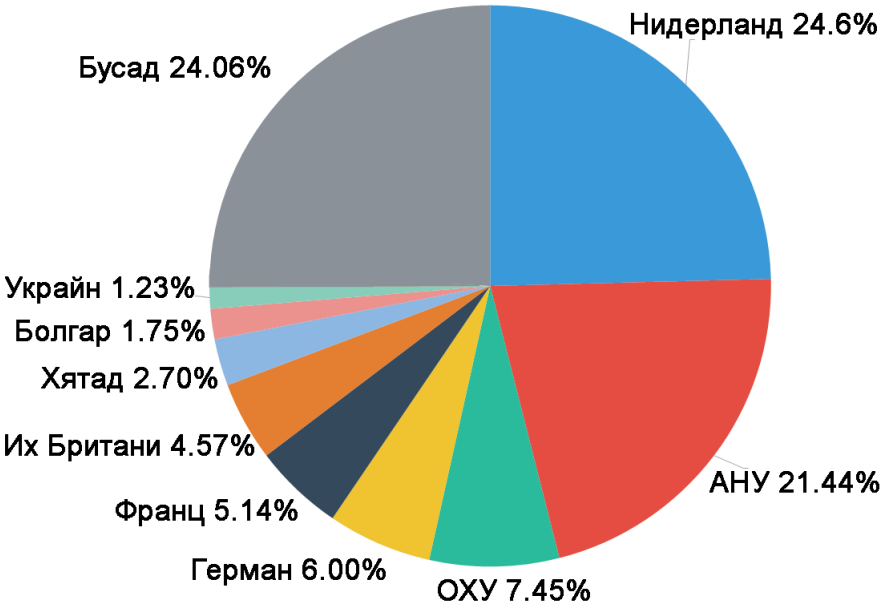
нь системийг гацаах, зогсоох болон нууц мэдээллийн алдах, төхөөрөмж чагнах зэрэг маш олон эрсдэлийг хэрэглэгчид учируулж болзошгүй.

Андройд хортой програмыг анх Бразил улсын програм хангамжийн инженер Маркос Веласко танилцуулсан бөгөөд аюул заналаас сэргийлэхийн тулд хэн нэгэн ашиглаж болох (Нээлттэй эх) хортой кодыг хийж байжээ. Хамгийн анхны гар утасны хортой код Timofonica нь Спайн улсаас үүссэн гэж үздэг ба Spain хэл дээрх гар утасны мессижийг уншиж серверлүү дамжуулдаг байсан. 2000 оны 6-р сард ОХУ, Финландад уг вирусыг вирусны эсрэг лабораториудаар илрүүлсэн байна.

Түгээмэл андройд хортой кодууд

- Өт буюу Worm - Энэ хортой код нь бие даасан байдалтай байдаг ба гол зорилго нь төгсгөлгүй байдлаар өөрийгөө үржүүлж, бусад төхөөрөмжүүдэд тараах явдал юм. Хор хөнөөлийн хувьд янз бүр байж болох ба гол шинж тэмдэг нь уг байдал юм.
- Трожан буюу Трояны морь - Worm -оос ялгаатай нь хэрэглэгчийн харилцан үйлчлэлийг идэвхжүүлэхийг үргэлж шаарддаг. Энэ төрлийн вирусын гол шинж тэмдэг нь хэрэглэгчийн татаж авсан үнэгүй эсвэл хэрэглээний зөрчилтэй програмд давхар байршисан байдаг. Өгөгдөл цуглуулж алсын серверд илгээх, бусад төхөөрөмжинд халдварлаж тарах замаар ноцтой хохирол үзүүлж болдог.
- Спаявер - Уг хортойд код нь ихэвчлэн system monitors, trojans, adware, tracking cookie гэсэн 4н хэсэгт хуваагддаг. Таны компьютерийг тагнуулддаг програм юм. Гол үүрэг нь төхөөрөмж дээрх заагдсан мэдээллийг цуглуулж үүсгэгчрүүгээ илгээдэг.
- Rootkit - Үйлдлийн системийн ажиллагааг өөрчлөх замаар администраторуудаас хортой програмыг нууцласан байх хооронд нь компьютерт давуу эрх олгох боломжийг олгодог програм хангамж. Нэг үгээр бол root эрхийг авна гэсэн үг юм.
- Backdoor - Хамгийн хүчирхэг трояны морь нь систем рүү алсаас хандах боломжийг олгодог бөгөөд ердийн нэвтрэлтийг оруулалгүйгээр өөртөө суулгасан програм шиг хувирдаг. Мөн нэвтрэлт хийж байгаа зарим оролтуудыг нуухыг оролддог.

2017 оны байдлаар андройд хортой код илрүүлэлтийн 31 хувь нь Europe тивээс илэрсэн байдаг. Харин улс бүрээр ангилж үзвэл дээрх 10н улс тэргүүлж байна. Kaspersky Lab мэдээлснээр [4]:



Зураг 1.7: Олон улсын хорт програмын халдлагын эзлэх хувь

Манай улсын хувьд хамгийн их илэрдэг андройд вирусууд нь :

- Trojan-SMS.AndroidOS.OpFake.bo. Энэ нь хамгийн нарийн бүтэц үйл ажиллагаа бүхий SMS трояуудын нэг юм. Ялгарах гол онцлогт нь сайн зохион байгуулалттай интерфейс болон, хөгжүүлэгчдийнх нь ашиг олох сонирхол юм. Ажиллагааныхаа явцад гар утас хэрэглэгчийн данснаас 9 доллар-с эхлээд дансан дахь нийт мөнгийг хүртэл хулгайлах чадамжтай. Мөн түүнчлэн хэрэглэгчийн утсанд хадгалагдсан утасны дугаарууд нэвтэрч, SMS илгээх аюултай.Голчлон ОХУ буюу хуучин Зөвлөлт Холбоот улсын бүрэлдхүүнд байсан орны хэрэглэгчдэд халддаг.
- AdWare.AndroidOS.Ganlet.a. Бусад аплекейшнүүдийг суулгахад нууцар хавсран суудаг процес бүхий зар суртчилгааны модул.
- Trojan-SMS.AndroidOS.FakeInst.a. Уг хортой код нь 2 жилийн өмнө энгийн SMS Троянаас ботоор удирдагдан олон төрлийн сүлжээнд ажиллах чадвартай болтлоо хөгжсөн байна. Энэ троян нь хэрэглэгчээс мөнгө хулгайлах болон хохирогчийнхоо утсанд хадгалагдсан дугаарууд руу хуурамч SMS илгээх чадвартай.

Уг илэрцийг гаргахдаа Касперскийн програмууд болон Касперскийн Аюулгүйн Сүлжээний үүлэн технологийг ашиглан гаргасан болно.

-	Нэр	Халдлагад эзлэх хувь
1	DangerousObject.Multi.Generic	40.42%
2	Trojan-SMS.AndroidOS.Op.Fake.bo	21.77%
3	AdWare.AndroidOS.Ganlet.a	12.40%
4	Trojan-SMS.AndroidOS.Fakelnst.a	10.37%
5	RiskTool.AndroidOS.SMSreg.cw	8.80%
6	Trojan-SMS.AndroidOS.Agent.u	8.03%
7	Trojan-SMS.AndroidOS.OpFake.a	5.49%
8	Trojan-AndroidOS.Plangton.a	5.37%
9	Trojan.AndroidOS.MTK.a	4.25%
10	AdWare.AndroidOs.Hamob.a	3.39%

Хүснэгт 1.2: Андройд хорт програмын эзлэх хувь

1.2 Машин сургалтын аргад математикийн ойлголт ашиглагдах нь

Машин сургалтын тухай суурь ойлголт болон түүнийг хэрэгжүүлэхэд бүхэлдээ математикийн тооцоолол хийж байдаг. Үүнд хамгийн нийтлэг ашиглагддаг нь магадлалын онол ба статистик, шугаман алгебр, олон хувьсагчыг тооцоолон бодох аргачлал гэх мэт ойлголтууд хамаарагддаг бөгөөд манай судалгааны ажилд энгийн ойлголт боловч хамгийн их ач холбогдолтой нь шугаман алгебрийн ойлголт юм.

1.2.1 Шугаман алгебр

Ямарваа зүйлийн зүй тогтлыг олох, магадлалыг олох зэрэг инженерийн болон шинжлэх ухааны салбарт ашиглагдах тооцооллыг хийхэд шаардагдах суурь ойлголтуудын нэг бол шугаман алгебр юм. Хэдий энгийн боловч Machine сургах үйл явцад хамгийн ихээр хэрэглэгддэг тул компьютерийн ухааны салбарт шугаман алгебрийн талаар дискрет/тасалдалтай/ математик зэргээс илүү их мэдлэг туршлагатай байх хэрэгтэй.

Хэрэв та шугаман алгебрийн талаар хангалттай мэдлэг туршлагатай бол энэхүү бүлгийг алгасаж болох бөгөөд шаардлагатай гэж үзвэл энэхүү бүлэг нь зөвхөн Machine сургалт болоод Deep Learning чиглэлээр судалгаа хийхэд зориулсан мэдлэг туршлагыг олгох болно. Шугаман алгебрийн судалгаанд хэд хэдэн обьектууд хамаардаг. Үүнд:

Скалярын тухай Зөвхөн тоогоор илэрхийлэгдэх хэмжигдэхүүнийг скаляр хэмжигдэхүүн гэдэг. Тухайлбал талбай, эзэлхүүн, ажил, масс, температур зэрэг нь скаляр хэмжигдэхүүн юм. Хүч, хурд хурдатгал гэх зэрэг тооноос

гадна чиглэлээр илэрхийлэгдэх хэмжигдэхүүн байдаг. Скалярыг ихэнхдээ жижиг үсгээр тэмдэглэдэг бөгөөд

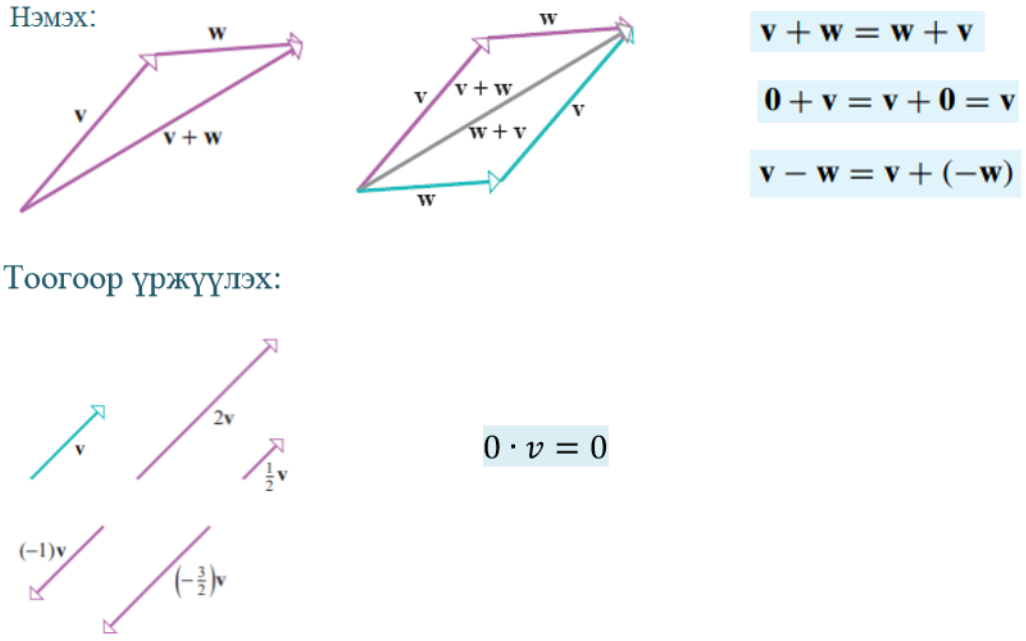
$$n \in N$$

гэсэн хамаарлын дагуу тодорхойлогддог.

Векторын тухай Чиглэлтэй хэрчмийг вектор гэнэ. Эхлэл нь А төгсгөл нь В байх векторыг $\overrightarrow{AB}$ гэж тэмдэглэнэ. Векторыг мөн ν, ω, a, b гэх мэт үсгээр тэмдэглэнэ. Эхлэл төгсгөл нь давхцсан векторыг тэг вектор, векторын уртыг модуль гэнэ. Ижил чиглэлтэй,тэнцүү урттай векторуудыг тэнцүү векторууд, эсрэг чиглэлтэй тэнцүү урттай векторуудыг эсрэг векторууд гэнэ. Параллель шулуун дээр орших векторыг коллинеар векторууд

$$(\nu||\omega)$$

гэнэ. Зураг 1.8-т векторуудын хооронд хийх үйлдлүүдийг жишээгээр харуулсан болно. Эдгээр үйлдлүүдийг (нэмэх, тоогоор үржүүлэх) шугаман үйлдэл



Зураг 1.8: Шугаман үйлдлүүд

гэдэг.

Векторыг тэмдэглэхдээ хоёр талдаа дөрвөлжин хаалтанд элементүүдийг 2 хэмжээстээр тэмдэглэнэ.

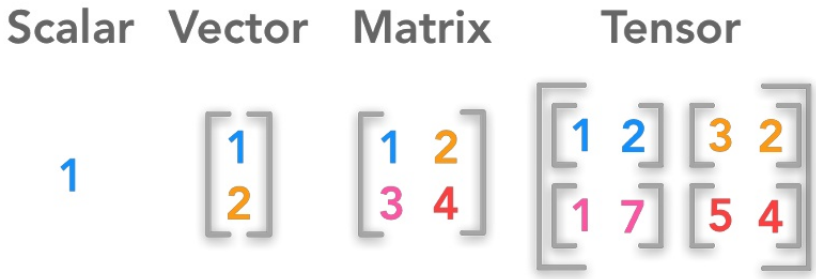
$$\overrightarrow{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \end{bmatrix}$$

Матрицын тухай n, m-үүд нэгээс их бүхэл тоонууд байг.

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & x_{m3} & \dots & x_{mn} \end{bmatrix}$$

хэлбэрийн бодит тоон хүснэгтийг **матриц** гэж нэрлэдэг. Матрицыг ихэнхидээ том үсгээр тэмдэглэдэг бөгөөд ямар ч матриц элементүүдээс бүрдэх бүюу m өндөртэй, n өргөнтэй байдаг. Матрицыг төрөл бүрээр ашиглан талбайн болон огторгуйн геометрт өргөнөөр хэрэглэх боловч компьютерийн ухаанд ихэвчлэн векторын чиглэл ба хэмжээг заах, тодорхойлох зорилгоор ашигладаг. Ийм учраас манай судалгааны ажилд матриц болон векторын тухай ойлголт хамгийн чухал суурь ойлголт болж үргэлжилнэ.

Тенсор-н тухай Энэхүү ойлголтыг энгийнээр авч үзвэл маш жижиг деталиар хэмжсэн векторуудыг нийлүүлж тодорхойлоход гарах үр дүн юм. Өөрөөр хэлвэл бие биенээсээ үргэлжилсэн хоорондоо адилгүй векторуудыг нийлүүлж нэг вектор болгон хувиргавал тахир, муруй нэгэн төрлийн зураас гарах болно. Эхний чиглэлээс эцсийн цэг хүртэл үргэлжилсэн векторуудын замыг хадгалахад тенсор үүснэ. Зураг 1.9-с шугаман алгебрын обьектуудын ялгаа, харилцан холбоог харж болно.



Зураг 1.9: Шугаман алгебрын обьектуудын хоорондын харилцан холбоо

Магадлалын хууль Санамсаргүй үзэгдлүүдийн тухай математик дахь судалгаа 16, 17-р зууны үеэс эхтэй ч эхэндээ дискрет үзэгдлүүдийн тухай, голчлон комбинаторикийн аргаар судалж байв. Сүүлд математик анализын

аргаар тасралтгүй тархалтын тухай ойлголтыг оруулж ирэн судалж эхэлсэн байна.

Орчин үеийн магадлалын онолын эцэг нь Андрей Колмогоров юм. Тэрээр Ричард фон Майзесын оруулж ирсэн түүврийн огторгуй, мөн хэмжээсийн онолыг ашиглан Колмогоровын аксиомууд гэж нэрлэгдэх болсон аксиомуудын системийг магадлалын онолд оруулж ирсэн нь 1933 он байв.

Магадлалын онолын хувьд ямар ч төгс алгоритм ашигласан хиймэл оюун 100% гүйцэтгэлтэй үр дүнг үзүүлж чадахгүй. Учир нь магадлалын хууль ёсоор таамаглал дэвшүүлж түүн дээрээ суралцан дахин таамаглах байдлаар давтдаг.

Бид магадлалын онолыг хиймэл оюунд ашиглахдаа хоёр арга замаар ашигладаг.

1. Магадлалын хууль нь хиймэл оюун хэрхэн бий болж бүтээгдэж байгааг тайлбарлаж байдаг учраас магадлалын онол дээр суурилсан илэрхиллүүдийг тооцоолохын тулд тохирсон алгоритмыг бид өөрсдөө зохион байгуулдаг.
2. Хиймэл оюуны системийг анализ хийхийн тулд магадлалын онол, статистикийг ашиглаж болно.

Хиймэл оюун дахь магадлалын онол нь их хэмжээний өгөгдлүүдийн тархалт, өгөгдөл хоорондын харилцан адилгүй зүй тогтлыг олох гэх мэт зүйлс дээр хэрэгжүүлдэг байна. **Магадлалын тархалт** Хиймэл оюунд нийтлэг хэрэглэгддэг энгийн магадлалын тархалтуудын хэд хэдэн төрөл байдаг. **Бернулийн тархалт** Магадлалын онол болон статистикт, Бернуллийн тархалт (англ. Bernoulli distribution) гэдэг нь үр дүн нь "амжилттай", эсвэл "амжилтгүй"гэсэн хоёр боломжоос санамсаргүйгээр сонгогддог туршилтыг хэлнэ.

- Зоос сүлд-ээрээ буусан уу?
- Шинэ төрсөн хүүхэд охин уу?
- Тэр хүн ногоон нүдтэй байсан уу?
- Хэрэглэгч бүтээгдэхүүнийг худалдан авахаар шийдсэн үү?
- Иргэн тодорхой нэр дэвшигчийн төлөө саналаа өгөхөөр шийдсэн үү? зэрэг...

Гауссын тархалт Комплекс бус бүхэл тоон дээр хамгийн нийтлэг ашиглагддаг тархалт бол энгийн тархалт буюу Гауссын тархалт гэж нэрлэдэг. Хэвийн тархалтын графикийг дор харууллаа. Эдгээр нь хэр тархаж байрласан байгаагаар бие биенээсээ ялгарч байна. Гэхдээ тэдгээрийн талбайн хэмжээ

$$\mathcal{N}(x;\mu,\sigma^2) = \sqrt{\frac{1}{2\pi\sigma^2}}\exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right).$$

Зураг 1.10: Гауссын тархалтын функц

ижилхэн байна. Хэвийн тархалтын эдгээр ялгаатай хэлбэр нь 2 параметрээс шалтгаална: дундаж(μ) (mean)ба квадрат дундаж хэлбэлзэл(σ)(standart deviation).

Мэдээллийн онол Мэдээллийн онол нь мэдээллийн квантчлалын талаар тайлбарладаг хэрэглээний математик, цахилгааны инженерчлэл, компьютерийн шинжлэх ухааны салбар.

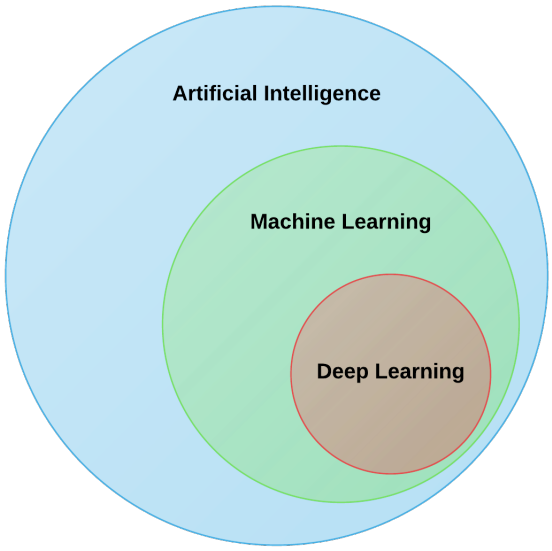
АНУ-ын эрдэмтэн Клод Шэннон 1948 онд мэдээлэл нь бие даасан шинжлэх ухаан мөн гэдгийг баталж, түүний онолыг боловсруулж, аливаа мэдээллийн хэмжээг мөрийн тоо, цаг хугацаагаар бус агуулгаар нь хэмжиж болно гэсэн дүгнэлт хийжээ. Тэрээр аль нь ч үнэн байж болох хоёр боломжийн нэгийг тодруулсан хариунд багтаж буй мэдээллийг нэг бит хэмжээтэй гэж үзээд, түүнийг хэмжих дараах томъёог боловсруулсан байна. Үүнд: $H = \log_2 n$. Энд H – бит хэмжигдэхүүн, n – боломж бүхий тохиолдлууд юм. Үүнийг орчин үеийн мэдээллийн шинжлэх ухаанд өргөн ашиглаж байгаагийн хамт зарим тохиолдолд сэтгүүл зүйн практик ч хэрэглэдэг байна.

Хиймэл оюуны салбарт мэдээллийн онолыг ашиглахдаа их хэмжээний өгөгдөл/өгөгдөл, мэдээлэл хоёр нь үндсэндээ утга ижил гэж үзэв/ дээр суурилан шинжилгээ хийж, түүн дээр мэдээлэл нэмэх хасах, дундаас нь мэдээлэл авах, задлах гэх мэт зүйлсийг хэрэгжүүлдэг. Маш их хэмжээний өгөгдөл дээр ажиллах тийм ч амар биш бөгөөд анхан шатнаас эхлээд төгсгөл хүртэл онолын дагуу гүйцэтгэдэг болно.

Мэдээллийн онолын нарийн төвөгтэй хуулиудыг энд ашиглахгүй ч компьютерийн ухааны салбарт үүнийг дурьдахгүй байж болохгүй юм.

1.3 Машин сургалтын тухай үндсэн ойлголт

Deep Learning хэмээх технологи нь Machine Learning/Машин сургалт/-ын тухайн нэгэн тохиолдол юм. Өөрөөр хэлвэл Machine Learning нь ерөнхий бүтэц агуулгатай дутагдалтай тал ихтэй бөгөөд давуу тал болсон зүйлсийг нь нийлүүлээд Deep Learning хэмээн нэрийдэж байгаа гэж хэлж болно. Зураг 1.11-ээс илүү дэлгэрэнгүйг харна уу. Ийм учраас Deep Learning-н тухай сайн ойлгохын тулд эхлээд Machine Learning/Машин сургалт/-ын алгоритм, ажиллагааны зарчмыг ойлгосон байвал зохистой юм.



Зураг 1.11: Хиймэл оюуны ерөнхий схем

Энэ бүлэг нь Machine Learning-н тухай бүхэлд нь агуулахгүй бөгөөд зөвхөн үүнээс Deep Learning-н технологид ашиглаж хөгжүүлсэн, шаардлагатай Machine Learning-н үндсэн арга техникуудыг танилцуулж, тайлбарлан товч ойлголт өгөх зорилготой юм. Үүнд:

- Өөрөө сурах алгоритм буюу Learning algorithm гэж юу вэ ? Жишээ нь: Linear Regression Algorithm...
- Өгөгдлийг сургалтын training буюу Neural сүлжээний загварыг гаргахад тулгардаг асуудал, тэдгээрийг хэрхэн шийдсэн тухай
- Ихэнхи Machine Learning алгоритмд байдаг "Hyperparameters"гэх тохиргооны тухай болоод үүнийг хэрхэн өөрөөр нь тохируулдаг тухай
- Machine Learning алгоритм нь дотроо 2 төрөл болон хуваагддаг. Тэдгээрийн тухай болон жишээ алгоритмуудын тухай гэх мэт

Эдгээр ойлголтууд нь цаашдаа "Deep Learning"алгоритмын тухай судлах, хэрэгжүүлэхэд хөтлөх болно. **Өөрөө сурах алгоритм буюу Learning Algorithm**

Machine Learning Algorithm гэдэг нь ерөнхийдөө өгөгдлөөс өөрөө суралцах боломжтой алгоритм юм. Гэхдээ суралцах гэдэг нь яг юу гэсэн үг вэ ?

"Компьютерийн програм нь хэд хэдэн даалгаврууд/Task - T/ болон гүйцэтгэлийн хувиас/Performance measure - P/ хамаарч цуглуулсан туршлага/Experience - E/ дээрээс үндэслэн суралцана. Даалгаврыг гүйцэтгэхдээ туршлагаас хамаарч гүйцэтгэлийн хувь өндөр эсвэл бага байна."

Үүнээс үзэхэд өөрөө суралцах алгоритмыг анх зохиохдоо хүний суралцдаг зарчмаар машинд зориулж бүтээсэн байгааг харж болно. Машин сургалтын талаар илүү их хөгжүүлж байгаа нь мөн өөрсдийнхөө суралцах зарчмыг хөгжүүлж буй гэсэн ойлголт байгаа нь илүү сонирхолтой санаа юм. **Даалгавар буюу Task** Даалгавар гэдэг нь өөрөө суралцах үйл явцыг хэлж буй явдал биш юм. Өөрөөр хэлвэл суралцах гэдэг нь даалгаврыг гүйцэтгэх чадварт хүрэх үйл явцыг хэлж байгаа юм. Хэрвээ бид роботод алхахыг зааж өгөх ёстой бол "алхах"гэдэг нь даалгавар бөгөөд бид түүнд хэрхэн алхахыг зааж өгнө эсвэл шууд алхах үйлдлийг програмчилж болно гэсэн үг.

Дээр дурьдсанчлан бид роботод алхах үйлдлийг зааж өгнө гэдэг нь машиныг суралцахдаа ихэвчлэн жишээн дээр суурилаж өөрөө суралцдаг. Тэрхүү жишээ нь “Feature” -уудын цуглуулга байдаг бөгөөд энэ нь машин сурахад зориулж ашиглагдах обьектууд байх юм. Feature-г vector-оор төлөөлүүлдэг бөгөөд векторын төгсгөл бүр дараагийн feature-г тодорхойлно. Одоогоор машин сургалтаар сурсан машинууд дараах даалгаврууд ыг өндөр бүтээмжтэй гүйцэтгээд байгаа юм.

- Classification буюу зураг, авиа, текс ангилах
- Зурагнаас дүрс таних
- Аль нэг хэлнээс нөгөө хэлрүү орчуулга хийх
- Газар зүйн байрлал бүртгэж авч түүн дээр үндэслэн гэмт хэрэг илрүүлэх

Мэдээж эдгээрээс бусад даалгаврууд ыг гүйцэтгэх бүрэн боломжтой бөгөөд дээрх жагсаалтан дахь даалгаврууд нь зөвхөн машин хийж чадах жишээн дээрээс суралцсан тохиолдлууд юм.

Гүйцэтгэл буюу Performance Measure Machine Learning Algorithm-н ажиллагааг сайн эсвэл муу хэмээн үнэлэхийн тулд бидэнд гүйцэтгэлийн хувиар илэрхийлэгдэх үнэлгээ хэрэгтэй болно.

БҮЛЭГ 2

Судалгааны хэсэг

Хүмүүс сэтгэн бодох чадвартай машин бүтээхийг эртний Грекийн үеэс л хүсэж эхэлжээ. Анхны компьютер бий болох үед ч хүмүүс хиймэл оюунтай эсэхийг нь сонирхож байв. Тэгвэл өнөөдөр тэр хүсэл биелэхэд тун ойртсон, AI нь ШУ -ны нэгэн салбар болж чадсан төдийгүй, бидний өдөр тутмын амьдралд ч ашиглагдаад эхэлжээ.

Нэгэн хэвээр хийгддэг ажлыг хөнгөвчлөх, үг яриа, зураг дүрсийг таних, өвчтөнд оношлох тавих зэрэгт "оюунлаг" програмчлалыг ашиглах хэрэгтэй болж байна шүү.

Эхэн үедээ AI хүнд хамгийн хүндрэлтэй байдаг хэдий ч компьютерын хувьд бол амархан (формалаар илэрхийлэх боломжтой, математик загварчлал хийх боломжтой) тооцооллуудыг хийж асуудал шиддэг байлаа. Тэгвэл одоо AI -н жинхэнэ шийдэх учиртай асуудал бол хүмүүсийн зөн совингоороо хийдэг, формал загварчлалд оруулахад хүндрэлтэй үйлдлүүд юм. Жишээ нь хүний ярьж байгааг ойлгох, зурган дээрх хүмүүсийг таних гэх мэт.

Үүнийг шийдэх арга бол компьютерт өмнөх туршлагаасаа суралцах боломжийг олгож, хорвоог энгийн ойлголтуудын шаталсан бүтэц байдлаар ойлгуулах юм.

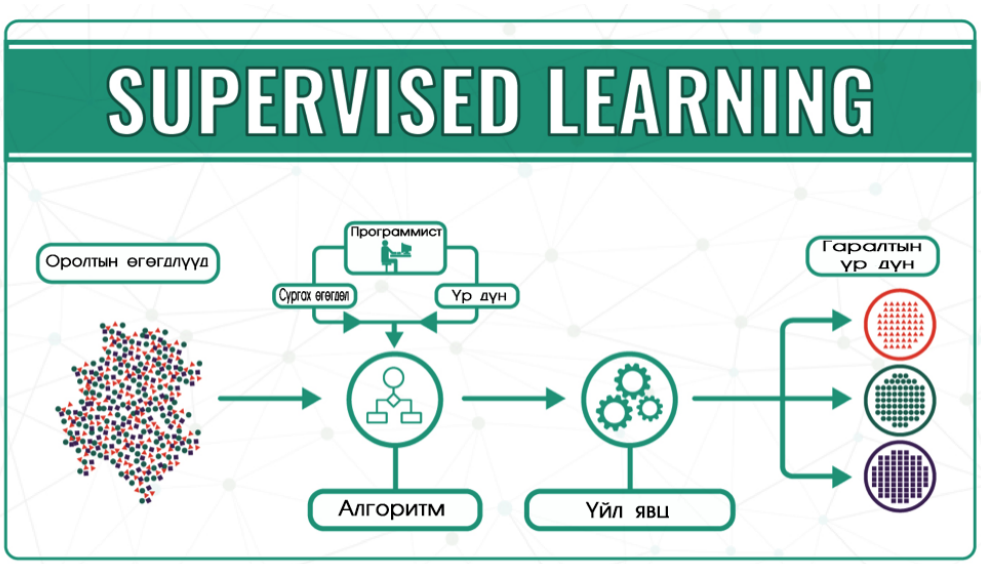
2.1 Машин сургалтын алгоритмууд

Машин сургалтын алгоритмыг дотор нь хэд хэд ангилдаг бөгөөд тэрхүү ангилсан алгоритмууд нь өөр хоорондоо адилгүй зорилгоор ашиглагдах нь бий. Тухайн сургасан машин Ямар зорилгоор ажиллах эсэхийг юуны түрүүнд тодорхойлох шаардлагатай юм.

1. **Супервайзэд сургалтын/Supervised Learning/ Алгоритм**
   Ихэнхи machine learning хэрэгжүүлэлтийг хийхийн тулд supervised сургалтын арга ашигладаг. Supervised сургалтын арга нь оролтын хувьсагч (X) болон гаралтын хувьсагчийг (Y) загварчлах функцыг боловсруулан суралцуулахын тулд алгоритм ашигладаг байна. Гаралтын хувьсагч (Y) болон оролтын хувьсагч (X) хоорондын харьцаа нь

   $$Y = f(X)$$

   ийм байна. Зураг 2.1-с ерөнхий ажиллагааг илүү ойлгомжтойгоор харна уу.
   Supervised сургалтын гол зорилго нь гаралтын хувьсагчаас (X) шинэ өгөгдөл оруулах үед гаралтын хувьсагчийг тооцоолон таан загварчлах функцыг маш сайн тоймлон хураангуйлахад оршино. Өөрөөр хэлвэл



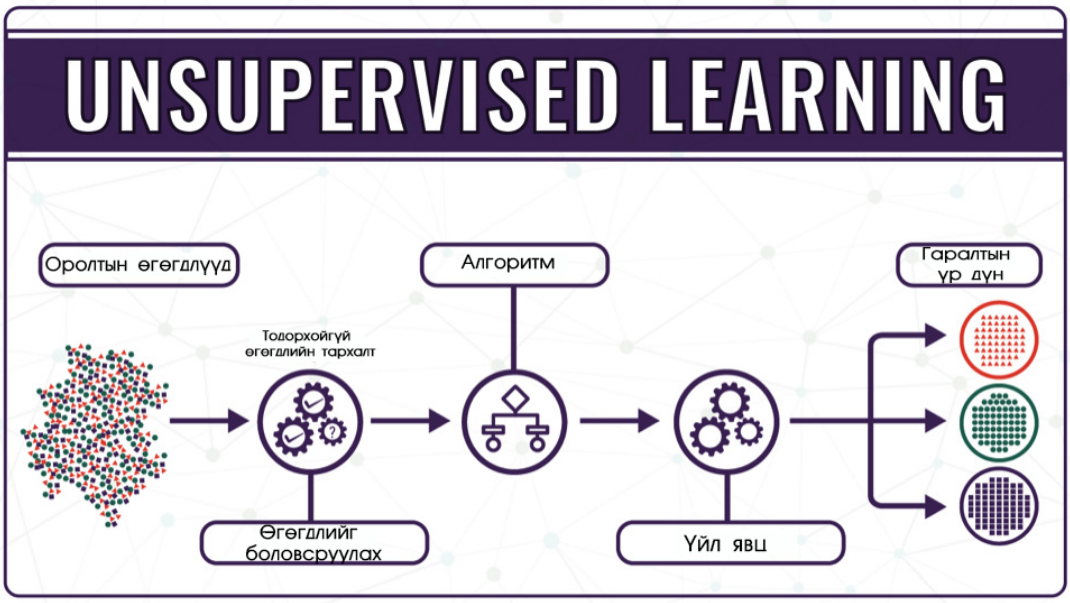
Зураг 2.1: Supervised сургалтын аргын ажиллагааны зарчим

багш зааж сурагч сурдаг шиг урьдчилж бэлдсэн өгөгдлийн тархалт дээрээс үндэслэн суралцдаг алгоритмын үйл явц байдаг учраас үүнийг **supervised learning** гэж нэрлэдэг. Үүн дээр машин болон хүн хоорондын харилцаа нь машин өгөгдөл дээр үндэслэн таамаглал дэвшүүлэхэд хүн түүнийг засаж зөв болгоход сургадаг гэхэд ойлгомжтой юм. Бүх өгөгдөл өөр өөрсдийн шинж чанартайгаар тархсан байх бөгөөд сургалтын алгоритм нь гаралтын үр дүнг оролтын өгөгдөл дээрээс үндэслэн таамаглах зарчимтай. Алгоритм нь зөвшөөрөгдсөн гүйцэтгэлийн түвшинд хүрэх үедээ зогсдог бөгөөд supervised сургалтын арга нь **regression** болон **classification** гэсэн ажлууд дээр илүү тохиромжтой байдаг. Энэ нь:

- **Classification:** Хэрвээ гаралтын хувьсагч нь категорит ялгах боломжтой байвал алгоритм үүнийг ангилж чадна. Жишээ нь: Хөх эсвэл улаан, өвчтэй эсвэл эрүүл гэх мэт
- **Regression:** Гаралтын хувьсагч нь жинхэнэ утга байх үед буюу жишээ нь төгрөг, хүний биеийн жин гэх мэт.

2. **Ансупервайзэд сургалтын/Unsupervised Learning/ Алгоритм** Unsupervised сургалтын арга нь зөвхөн оролтын хувьсагчийгг (X) авах ба түүнээс харгалзах гаралтын хувьсагч (Y) үл байна. Энэхүү аргын гол зорилго нь өгөгдөл дээр илүү суралцахын тулд эх өгөгдлийн үндсэн бүтцийг эсвэл түүний тархалтыг загварчлах юм. Supervised сургалтын арга шиг сургах хүн болон үр дүнд нь зөв хариулт гэж байхгүй учраас Үүнийг **Unsupervised Learning** гэж нэрлэдэг. Бүх өгөгдөл

шинж чанарын хувьд ач холбогдолгүйгээр тархсан байх бөгөөд сургалтын алгоритм нь оролтын өгөгдлөөс хамаарч түүний суурь бүтцийг сурах зарчимтай байдаг. Зураг 2.2-с ерөнхий ажиллагааг харна уу.



Зураг 2.2: Unsupervised сургалтын аргын ажиллагаа

Unsupervised сургалтын арга нь **clustering** болон **association** гэсэн ажлууд дээр илүү тохиромжтой байдаг. Энэ нь:

- **Clustering:** Өгөгдөл дэх нийтлэг хэрэглэгддэг багцуудыг хайх үйл ажиллагаа буюу жишээ нь хэрэглэгчдийг худалдаж авсан зүйлсээр нь ангилах гэх мэт.
- **Association:** Үндсэн өгөгдөл дээрээс үндэслэн дараагийн үүсэх өгөгдлийг таамаглах. Жишээ нь: Хүмүүсийн худалдаж авсан зүйлээс нь дараагийн худалдаж авахад хүрэх зүйлийг таамаглах гэх мэт.

3. **Нейв бэйс/Naive-Bayes/** Naive-Bayes нь энгийн боловч таамаглал тооцоолоход маш хүчирхэг алгоритм юм. Үүний машин сургахад хамгийн сонирхолтой нь өгөгдсөн өгөгдлөөс (d) хамгийн шилдэг таамаглалыг (h) сонгож чаддагт байгаа юм. Ангилахад эхний ээлжинд бидний таамаглал нь (h) өгөгдлийн жишээнд(d) зориулан ялгах давхарга байж болно. Ерөнхийдөө хамгийн амархан арга бол өгөгдөл дээрээс сурсан өмнөх мэдлэг дээрээсээ үндэслэн хамгийн өндөр магадлалтайг нь сонгож үр дүнд хүрэх юм. Өөрөөр хэлвэл Bayes-н теорем бол өмнөх сурсан

туршлагаасаа магадлал өндөртэй таамаглалыг тооцоолох боломжыг олгодог байна. Bayes-н теорем:

$$P(h|d) = \frac{P(d|h) \times P(h)}{P(d)}$$

- $P(h|d)$ нь өгөгдлөөс олсон таамаглалын магадлал юм. Үүнийг Постериор магадлал гэх нь ч бий.
- $P(d|h)$ нь таамаглал үнэн байсан үеийн өгөгдлийн магадлал.
- $P(h)$ нь таамаглал нь өгөгдлөөс үл хамааран үнэн байх магадлал. Үүнийг таамаглалын эхний магадлал гэдэг.
- $P(d)$ нь таамаглалаас үл хамаарсан өгөгдлийн магадлал.

Нейв бэйс ангилагч/Naive-Bayes Classifier/ Naive Bayes ангилагч нь аливаа оролтын утгыг хоёр эсвэл түүнээс их төрөлд ангилах зориулалттай юм. Хэрвээ оролтын утга нь хоёртын тооллын системд болон категорид хуваах боломжтой байвал naive-bayes ангилагчын техник буюу ажиллах зарчмыг ойлгоход төвөггүй. Тооцооллыг маш энгийн атлаа хүчирхэг байдлаар хийхийн тулд таамаглал бүрийг хялбаршуулсан байдаг учир үүнийг naive буюу гэнэн гэж нэрлэсэн байна. Өөрөөр idiot буюу маанаг гэх нь ч бий. Хэрвээ таамаглалын атрубутууд $P(d1, d2, d3|h)$ гэсэн байдалтай байвал $P(d1, d2, d3|h)$-ын атрибут бүрийн утгыг тооцоолох нь хэтэрхий ярвигтай болон тооцоолол их шаардана. Уг тооцооллыг өөрөөр тайлбарлавал цаашид $P(d1|h) \times P(d2|h)$ ингэж үргэлжлэх тул жинхэнэ бодит өгөгдөл дээр тооцоолон таамаглал дэвшүүлэхэд хэтэрхий магадлал багатай үр дүнг харах юм. Үүнээс бусад тохиолдолд бол өгөгдөл дээр маш сайн ажилладаг болно.

Нейв бэйс сургалт

Энэхүү жишээ нь цаг агаарын байдлыг харгалзан теннис тоглох эсэхийг шийдэхийг naive bayes алгоритмоор таамаглуулна. Хүснэгт 2.3-с харахад эхний машин сургах процесс/трейнинг/ маш бага магадлалтай үр дүнтэй байх бөгөөд үргэлжлүүлэн энэ байдлаар сургасаар байх бөгөөд /гэхдээ зохимжтой сургалт буюу tune-up хийх/ маш энгийнээр таамаглан шийдэлд хүрэх трейнингийг жишээгээр харуулсан болно.

Өгөгдлөөс naive-bayes моделийг сургах тухай Naive-Bayes моделийг трейнинг өгөгдөл дээр сургах нь хурдан юм. Учир нь тархалт бүрийн магадлал нь тус бүр бүгд ялгаатай оролтын утгатай (x) байх бөгөөд үүнийг тооцоолох нь маш хурдан байхаар зохион байгуулагдаж байгаа гэж хэлж болно. Коффицентуудыг бүгдийг нь оптимацлах байдлаар үйлдэл бүрийн дараалал дээр үндэслэсэн үргэлжилсэн тооцооллыг

Хүснэгт 2.1: Алгоритмын ажиллах жишээ хамгийн энгийнээр тайлбарлавал

Цаг агаар	**Шийдэл**
Нартай	Үгүй
Бүрхэг	Тийм
Бороотой	Тийм
Нартай	Тийм
Нартай	Тийм
Бүрхэг	Тийм
Бороотой	Үгүй
Бороотой	Үгүй
Нартай	Тийм
Бороотой	Тийм
Нартай	Үгүй
Бүрхэг	Тийм
Бүрхэг	Тийм
Бороотой	Үгүй

Хүснэгт 2.2: Сургалтын процесс жишээгээр

Давтамж		
Цаг агаар	Үгүй	Тийм
Бүрхэг		4
Бороотой	3	2
Нартай	2	3
Нийт	5	9

Хүснэгт 2.3: Бүх цаг агаарын төлөвийн давтамжын тоо

Магадлал				
Цаг агаар	Үгүй	Тийм	Үйлдэл	Үр дүн
Бүрхэг		4	=4/14	0.29
Бороотой	3	2	=5/14	0.36
Нартай	2	3	=5/14	0.36
Нийт	5	9		
	=5/14	=9/14		
	0.36	0.64		

Хүснэгт 2.4: Өгөгдлөөс магадлалыг тооцоолох

хийх шаардлагагүй юм.

- **Бүлэг магадлал** Үүнийг өөрөөр хэлвэл өгөгдлийн тархалтыг нийлүүлж магадлах гэсэн үг. Трейнинг өгөгдлийн тархалт дээрх бүлэг бүр дээрх магадлал.
- **Нөхцөл дэх магадлал** Бүлгийн утгад өгсөн оролтын утга бүр дээрх нөхцөл шалгаж буй магадлал.

Бүлгийн магадлалыг тооцоолох нь Бүлгийн магадлал нь бүлэг тус

бүрийн хамаарах тохиолдлын давтамжийг нийт тохиолдлын тоонд хуваана. Жишээ нь хоёртын тооллын системийн ангилалд эхний ангиллын хамаарах магадлалыг дараах байдлаар тооцоолно:

$$P(class = 1) = \frac{count(class = 1)}{count(class = 0) + count(class = 1)}$$

- count = тооцоолох оролдлого
- class = бүлэг

4. **K-д хамгийн ойр хувьсагч/K-Nearest Neighbors/ тухай**
k-Nearset Neighbor(KNN) нь ерөнхийдөө бүхэлдээ трейнинг өгөгдлийн тархалт юм. Бусад аргуудтай харьцуулвал арай өөр бөгөөд энэ нь бүхэл өгөгдлийн тархалтыг агуулах зарчмаас өөр зарчмаар ажилладаг. Тиймээс сурах шаардлагагүйгээр идэвхтэй шийдлүүдийг урьдчилан таамаглах явцад шинэ загваруудыг тохируулахын тулд хоёртын мод гэх мэт цогц өгөгдлийн бүтцийг ашиглан өгөгдлийг хадгалдаг.
Өгөгдөл KNN-д зохицох нь

- Өгөгдлийг дахин төлөвлөх/Rescale Data/ - Бүх өгөгдлийн хэмжээ ижил хэмжээтэй байвал KNN илүү үр дүнтэй ажилладаг. Нарийвчилвал өгөгдлийг 0-ээс 1-ийн хооронд оруулж зохион байгуулах нь хамгийн оновчтой санаа байдаг. Гауссын тархалттай бол заавал 0-ээс 1-ийн хооронд бус энгийн байдлаар стандартчилвал зохистой юм.
- Хаяггүй өгөгдөл /Address missing data/ - Мэдээлэл байхгүй бол өгөгдлүүдийн хоорондох зайг тооцоолох боломжгүй. Эхний болон эцсийн хаягийн мэдээлэлгүй өгөгдөл бол тэрхүү хаягуудыг хасч тооцооллыг хийх боломжтой.
- Нягтрал багатай өгөгдөл - KNN нь бага хэмжээст өгөгдөлд илүү тохиромжтой байдаг. Зуу эсвэл мянгаар тоологдох оролтын утгатай өгөгдөл дээр бусад техникүүдтэй харьцуулвал KNN нь илүү үр дүн багатай ажилладаг нь онолын хувьд үнэн юм.

KNN алдаатай тооцоолол
Дээр дурьдсанчлан KNN нь бага оролтын хувьсагчуудтай маш сайн ажилладаг боловч оролтын тоотой их хэмжээгээр зөрчилддөг. Оролтын хувьсагч болгон р-хэмжээст оролтын зайны хэмжигдэхүүн гэж үзэж болно. Жишээлбэл, хэрэв та X1 болон X2 гэсэн оролтын хоёр хувьсагчтай бол оролтын зай 2 хэмжээст байна. Орон зайн тоо нь нэмэгдэхийн хэрээр оролтын зайны хэмжээ нь экспоненциал хурдаар нэмэгддэг. Өндөр хэмжээстэй үед ижил төстэй цэгүүд хоорондоо маш их

зайтай байж болно. Бүх цэгүүд бие биенээсээ хол зайтай байх бөгөөд энгийн 2, 3-хэмжээст зайнд эвдэрч буй зайг хайх болно. Яг энэ үйл явц тооцооллыг гажуудуулдаг хандлагатай байдаг.

5. **Суппорт вектор машины/Support Vector Machine/ тухай**
   Support Vector Machine/SVM/ нь машин сургалтын технологид хамгийн их ашиглагддаг арга бөгөөд үүнийг 1990-ээд онд анх хөгжиж байх үедээ хамгийн алдартай арга байсан. Өндөр түвшний буюу өндөр давтамжын алгоритм бүтээхэд ашиглан үргэлжлүүлэн хөгжүүлсэн байдаг.
   Maximal-Margin ангилагч Maximal-Margin гаралтын утга буюу эцсийн таамаглалыг ангилагч нь SVM хэрхэн хэрэглэгддэг тухай тайлбарлах хамгийн тод жишээ юм.
   Таны өгөгдлийн тоон оролтын хувьсагч (x) (багана) нь n-хэмжээст зайг үүсгэдэг. Жишээлбэл, хэрэв хоёр оролтын хувьсагчтай байсан бол энэ нь хоёр хэмжээст орон зай үүсгэх болно.**Hyperplane** гэдэг нь оролтын хувьсагчийн зайг хуваах шугам юм. SVM-д оролтын хувьсагчийн цэгүүдийг хамгийн сайн хуваахын тулд hyperplane-ийг сонгохдоо 0, эсвэл 1-р ангиудын аль нэгээр нь ангилна. Хоёр хэмжээсээр энэ нь үүнийг мөр болгон дүрслэн харуулах боломжтой бөгөөд бидний бүх оролтын цэгүүд нь энэ мөрөөс бүрэн тусгаарлагдсан байна.

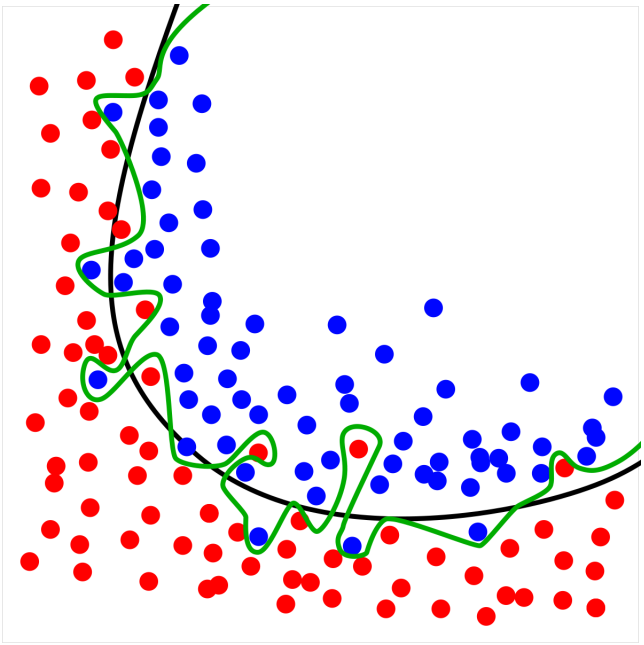
$$B0 + (B1 * X1) + (B2 * X2) = 0$$

   Коэффициент (B1 ба B2) нь мөрийн хөндлөн огтлол ба хөндлөн огтлолыг (B0) тодорхойлж байгаа тохиолдолд learning алгоритмаас олдох ба X1 ба X2 нь оролтын хоёр хувьсагч байна. Та энэ мөрийг ашиглан ангиллыг хийж болно. Оролтын утгыг шугам тэгшитгэлд оруулах замаар шинэ цэг нь шугамнаас дээш эсвэл түүнээс доогуур байгаа эсэхийг тооцоолох боломжтой.

   - Шугамны дээр тэгшитгэл нь 0-ээс их утгыг буцаана, утга бүрийг тодорхойлж буй цэг нь эхний ангилалд хамаарна (class 0).
   - Шугамны доорх тэгшитгэл нь 0-өөс бага утгыг буцаана,утга бүрийг тодорхойлж цэг нь хоёрдугаар ангилалд (class 1) хамаарна.
   - Шугаманд ойролцоо байрласан цэгийг 0/тэг/ утгатай буцааж өгч болох бөгөөд тэрхүү цэгийг ангилахад хүндрэлтэй байдаг.
   - Хэрэв гарсан утгууд нь их байвал таамаглал тооцоолоход илүү хялбар бөгөөд найдвартай байдаг.

   (Зураг 2.3)-аас харахад тархсан өгөгдлийг ангилсан байгаа бөгөөд хар зураас нь эхний сургалтаар тооцоолсон үр дүн, ногоон өнгөтэй зураас

эцсийн сургалтын үр дүн юм.



Зураг 2.3: SVM ангилал хийж буй жишээ зураг

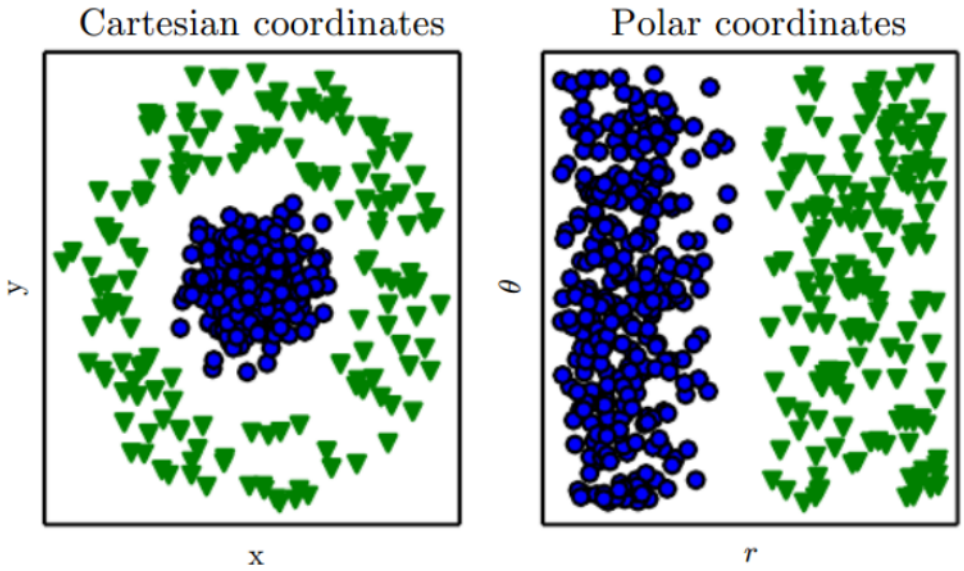
Өгөгдөл SVM-д зохицох нь

- **Тоон өгөгдөл**: Энэ нь оролтын бүх өгөгдлийг тоон өгөгдөл гэж үздэг. Хэрвээ оролтын өгөгдөл нь ангиллын өгөгдөл/Categorical input/ байвал үүнийг хоёртын хувьсагч болгон хөрвүүлэх шаардлагатай юм/ангилал бүрт нэг хувьсагч байхаар тооцоолох нь чухал/.
- **Хоёртын/binary/ ангилал**: Энэхүү арга нь SVM-д хамгийн зохимжтой хэрэглэгддэг арга юм. Регресс болон ангиллын олон тооцооллолд ашиглагддаг.

2.2 Гүний сургалт

Machine Learning алгоритмууд өгөгдлийн дүрслэл дээрээ маш их ач холбогдол өгдөг. Жишээ нь хийсвэр хийх эсэхийг зөвлөдөг алгоритм нь шууд өвчтөнийг оношлодог биш харин эмч нь оношинд хэрэгтэй, хөл хүнд эхийн талаарх мэдээллүүдийг түүж оруулдаг байв. Ийм нэг мэдээллийг feature/3.1.1 хэсэгт дурьдсан/ гэнэ. Feature -үүдийн тусламжтайгаар ямар дүгнэлт гаргахаа компьютер сурж эхэлдэг (Ө.Х эхэд ямар шинж тэмдэг байгаа нь ямар үр дүнтэй хамаарах вэ гэдгийг сурдаг). Гэтэл өвчтний MRI -н зургийг өглөө гэхэд дүгнэлт гаргаж чадахгүй.

Ингэж өгөгдлийн дүрслэлээс хамааралтай байх нь өдөр тутмын амьдралд ч, компьютерын ухаанд ч тогтмол тохиолддог зүйл. Жишээ нь компьютерын хайлт хийхэд өгөгдөл нь зөв бүтэцтэй, индекслэгдсэн байвал хэд дахин хурдан байна. Хүмүүс энгийн тооцооллыг араб тоон дээр сайн хийдэг ч Ром тоон дээр тийм ч амар чадахгүй.



Зураг 2.4: Өгөгдлийн тархалтын ялгаа

Зураг 2.4-с харахад гурвалжин, дугуй 2 төрлийн өгөгдлүүдийг зааглаасан шугам татах гээд үзвэл Cartesion координатыг ашигласан үед бараг л боломжгүй юм. Харин нөгөөх нь шууд харахад л ойлгомжтой байна.

AI -р шийдүүлэх асуудлаа зөв feature -үүдэд хуваагаад machine learning алгоритмд оруулж чадах юм ер нь асуудалгүй байхнээ гэсэн үг.

Гэтэл ихэнх асуудлуудын хувьд ингэж feature -үүдийг ялгаж авна гэдэг амаргүй. Жишээ нь зураг дээр машин байгаа эсэхийг таньдаг програм бүтээхийн тулд яах вэ? Машин бүр дугуйтай байдаг гэж үзээд дугуйгаар нь таних гэж үзвэл бас л ярвигтай. Дугуй яг ямар цэгүүдээс бүрддэг вэ гэдгийг тайлбарлах хэрэгтэй болж ирнэ. Уг геометрийн дүрс нь их энгийн боловч обуд дээр нь нарны гэрэл гялбах, өөр юмны сүүдэр дээр нь тусахад л хэрэггүй болчихож байгаа юм.

Энд factors of variation (өөрчлөлтийн коэффициент) гэдэг ойлголт гарч ирнэ. Энэ нь хүний өөрийн мэдэлгүй хийдэг тооцооллуудтай төстэй юм. Тухайн нэг feature -д нөлөөлж болзошгүй гаднын нөлөөллийг тооцоолно гэсэн үг.

Хүний аялгыг тодорхойлно гэдэг бараг л жинхэнэ хүний түвшний оюун ухаанаар шийдэгдэх түвэгтэй тооцооллыг шаардана. Хэл ярианы хувьд өгөгдлийн дүрслэлээ олж авахад тун ярвигтай гэсэн үг (өөр өөр акцентуудыг яаж дүрслэх вэ хө?) Тэгвэл deep learning -н тусламжтайгаар энгийн ойлголтуудаас тэрхүү хүнд ойлголтыг гаргаж авч чадна гэж үздэг.

Шийдэх арга нь өгөгдлийг бүтцэд оруулж өгөх биш компьютер өөрөө бүтцийг нь сурч авах арга юм. Үүнийг representation learning гэнэ. Хүний оролцоогүйгээр өөрөө feature -үүдээ олж авна гэсэн үг. Энгийн асуудлын хувьд хэдэн цагийн дотор, хүнд асуудлын хувьд хэдэн сарын дотор суралцаж чадна.

2.2.1 Гүний мэдрэлийн эсийн сүлжээ

Neural Network буюу мэдрэлийн эсийн сүлжээг машин сургахад зориулж анх тооцоолон бүтээхдээ олон layer буюу давхаргуудаас бүрдсэн /Зураг 2.6/ байдаг. Энд эхний давхарга бол input layer буюу оролтын давхарга үүнийг ашиглаж бид даалгаврыг сүлжээнд оруулна. Цаашлаад дунд байгаа hidden layer буюу нуугдмал давхаргууд нь их хэмжээний өгөгдөл дээрээс үндэслэн суралцсан байх бөгөөд үүний тусламжтай машинаар өгөгдсөн даалгаврыг гүйцэтгүүлэх боломж бүрдэх юм. Эцсийн давхарга нь гаралтын давхарга/output layer/-аар үр дүнгээ хүлээн авна.

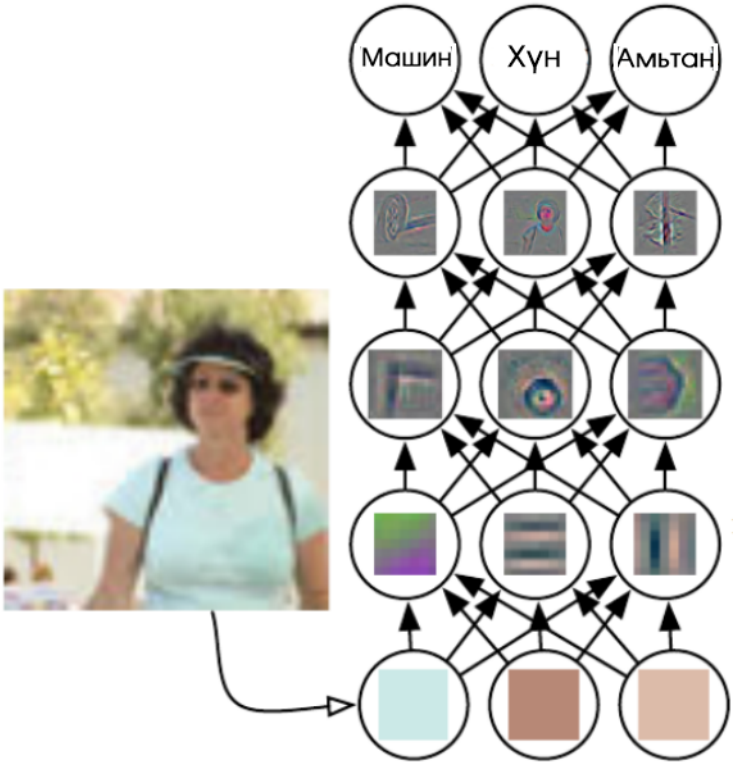
Зураг 2.5-с харахад хамгийн доод талд байгаа шиг цэгүүдийн утгууд дээр дүгнэлт хийхэд тун хэцүү. Дээрх зураг дээр үзүүлсэн deep learning загварыг тайлбарлавал: Нуугдмал давхаргууд нь зургаас машинд abstract feature -үүдийг салгаж авч байна. Анхны зурган дээр утга нь өгөгдөөгүй учраас нуугдмал гэж нэрлэж байна.

- 1-р давхарга - Цэгүүд, тэдгээрийн зэргэлдээ цэгүүдийн цайралтыг харьцуулснаар ирмэгүүдийг тодорхойлно.
- 2-р давхарга - Ирмэгүүдийн мэдээлэл дээр үндэслээд өнцгүүд, ерөнхий дүрсүүдийг олж авна.
- 3-р давхарга - Обьектын бүрдэл хэсгүүдийг олж эхэлнэ.

2.2.2 Конволюшнал эсийн сүлжээ

Машин сургалтын нэг хувилбар болох Конволюшнал эсийн сүлжээ

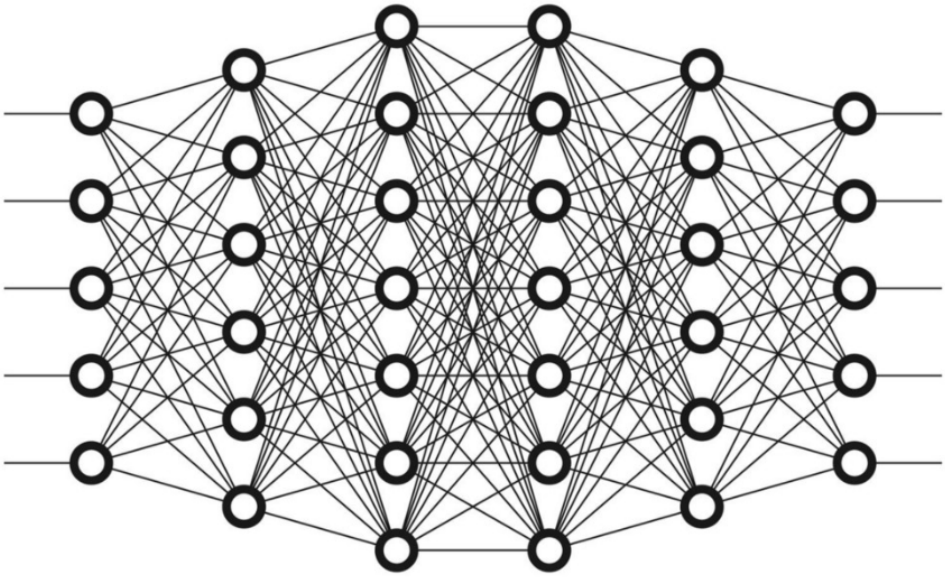
/Convolutional Neural Network/ нь анх 2012 онд Алекс Кризевский Image Net дэх уралдаанд/Компьютерийн харааны тэмцээн/ зураг ангилалтын алдааг 25%-с 15% хүртэл багасган рекорд тогтоон түрүүлснээр хөгжил дэвшил



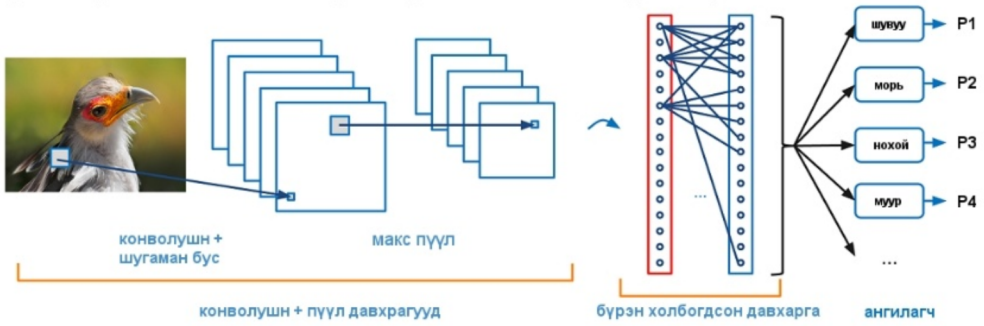
Зураг 2.5: Машиныг сургах мэдрэлийн эсийн жишээ

нь эхлэсэн байна. Хүн дүрсийг хүн ялгаж танихдаа, жишээ нь нохойг түүний сарвуу, нүд, арьс үс зэрэг онцлог шинжүүдээр нь ялгаж сурдаг бол компьютер ч мөн түүнтэй адил дүрс, биетийг доод түвшний мууй, хэрчим бүхий шинж чанаруудаас тогтсон convolution давхаргуудыг байгуулж ялгаж таньдаг аргачлал нь CNN юм. Зураг 2.7-т ерөнхий бүтцийг дүрслэн харуулав.

CNN нь convolutional, шугаман бус, пүүл, бүрэн холбогдсон давхарга болон гаралт гэсэн үндсэн хэсгүүдээс бүрдэнэ. Гаралт нь дан ангилал эсвэл тухайн дүрсийг хамгийн сайн тодорхойлж буй ангилалын магадлал юм. CNN-ын хамгийн эхний давхарга нь конволушнал байх бөгөөд жишээ нь, уг давхаргын оролт нь 32 x 32 x 3 хэмжээст цэгүүд бүхий матриц (тухайн зураг) байг. Уг давхаргыг ойлгомжтой, энгийнээр тайлбарлавал, тухайн зургийн зүүн дээд хэсгээс жижиг гэрлээр тусган гүйлгэн харж байгаа хэмээн төсөөлж болно. Уг жижиг гэрэл маань 5 x 5 хэмжээстэй тусгалтай байг. Машин сургалтын хэллэгт уг жижиг гэрлийг шүүлтүүр (заримдаа мэдрэлийн эс эсвэл цөм) гэж нэрлэдэг ба уг гэрэл тусч буйг хүлээн авах талбар гэдэг. Уг шүүлтүүр нь тоон массиваас (5 x 5 x 3) тогтох бөгөөд эдгээрийг жин эсвэл параметр гэдэг. Шүүлтүүрийг зураг дээгүүр гүйлгэхийг шүүлтүүрдэх гэх ба, тухайн өгөгдсөн зургийн цэгүүдийг шүүлтүүрийн цэгүүдээр харгалзан



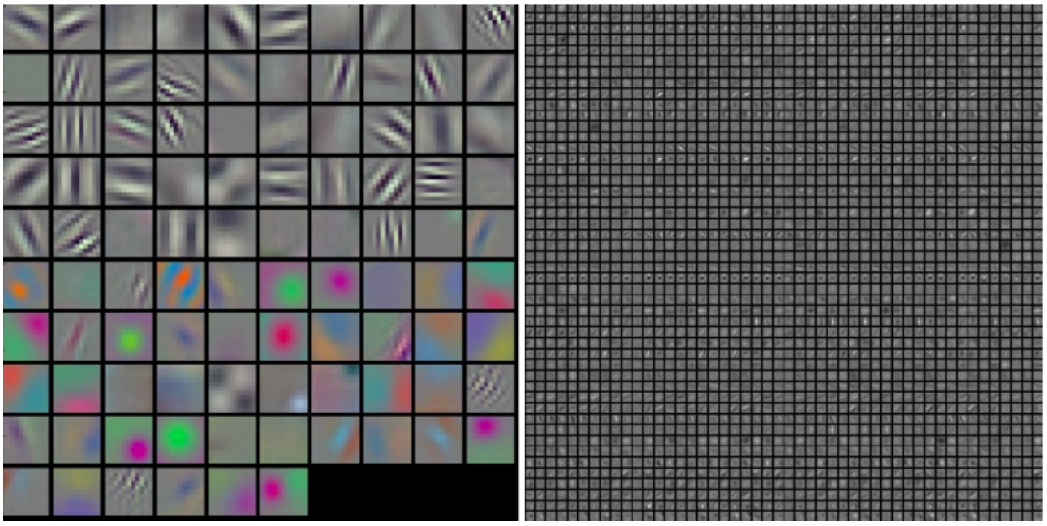
Зураг 2.6: Neural Сүлжээний загвар



Зураг 2.7: CNN-ий ерөнхий бүтэц

үржүүлнэ. Уг үйлдлийг шүүлтүүрийг дахин 1 цэгээр хажуу тийш шилжүүлэн давтах зэргээр тухайн зургийг дуустал давтсаны үр дүнд 28 x 28 x 1 хэмжээст үржвэрүүд бүхий матриц үүсэх ба үүнийг идэвхжилтийн зураглал эсвэл шинж чанарын зураглал гэж нэрлэнэ Convolutional Neural Network нь ерөнхийдөө дор хаяж нэг давхарга дээр өгөгдлөөс гарган авсан үндсэн матрицыг түүний тодорхойлогчоор үржүүлэх үйлдлийг ашиглаж байдаг.

Шүүлтүүрийг хүрээ, өнгө, муруй гэх мэт шинж чанарын ялгагч гэж ойлгож болно. Өөрөөр хэлбэл, бүхий л зураг дүрс бүрт байдаг хамгийн энгийн нийтлэг, шинж чанарууд байна. Эхний давхарга нь доод түвшний шинж чанарууд буюу муруй, хүрээ зэргийг танина. Гэхдээ тухайн эх зургийг яг юу вэ гэдгийг нь сайн ялгахын тулд гар, чих, нүд гэх мэт онцлог шинжүүдийг таних дээд түвшний шүүлтүүрүүд хэрэгтэй болно. 2-р давхарга дээр, жишээ нь 28 x 28 x 3 хэмжээстэй оролт дээр 5 x 5 x 3 хэмжээст шүүлтүүр



Зураг 2.8: Доод түвшний шүүлтүүрийн дүрслэл

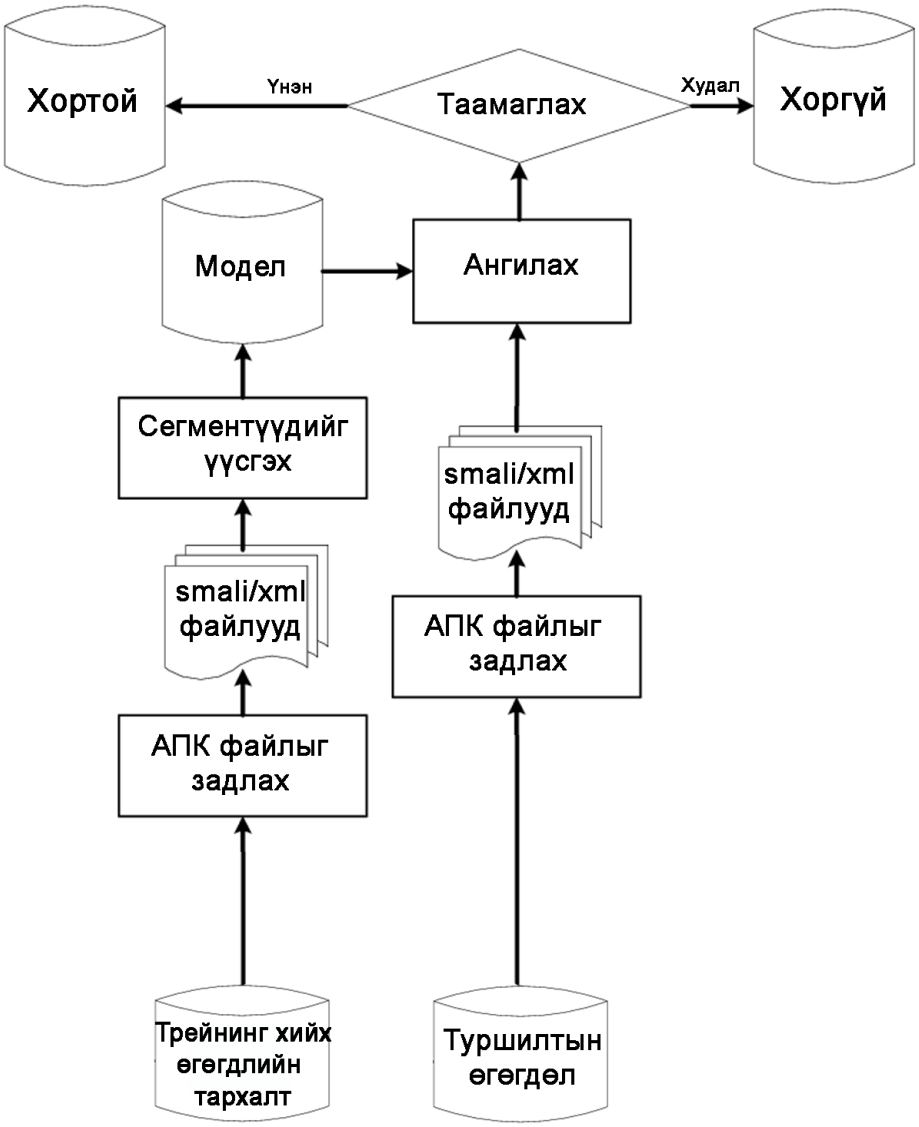
ашиглая. Уг давхаргын гаралт буюу шинж чанарын зураглал нь тал тойрог (муруй болон хүрээний хослол) эсвэл дөрвөлжин (хэд хэдэн хүрээнүүдийн хослол) зэрэг дээд түвшний шинж чанаруудын байршилууд гарна. Ингээд дараа дараачийн давхарга руу гүн орох тусам шинж чанарын зураглалууд илүү төвөгтэй, нарийн хэлбэрүүдийг дүрслэнэ. Сүлжээний төгсгөлд, аль нэгэн объект буюу дүрс бүхий шүүлтүүр идэвхижиж тодорсон байх болно

БҮЛЭГ 3

Төслийн хэсэг

Malware буюу хортой програмыг таних нь ерөнхийдөө програм хангамжийг хортой болон хоргүй гэж ангилах асуудал юм. Програмыг шинжлэх болон ангилах нь хоёр үндсэн үйл явцаас бүрдэнэ. Мэдээж програмыг ямар ч хорт код таних систем гаднаас нь буюу задлахгүйгээр танин шинжих боломжгүй. Тийм учраас эхлээд програмын эцсийн үр дүн буюу APK файлыг/EXE, DOC, DAT гэх мэт/ задлан тухайн програмын эх кодыг гарган шинжлэхэд бэлэн болгоно. Тэгээд дараагийн алхам нь задласан эх кодон дээр хортой програмыг танин, ангилах систем ажиллаж Malware болон Benign буюу хортой, хоргүй гэж ялгах ажиллагааг гүйцэтгэх юм. Олон тооны програмууд дээр боловсруулалт хийх хэрэгтэй учраас шинжлэх болон ангилах үйл явцыг хүний гараар үйлдэх оролцоогүйгээр зохион байгуулах шаардлагатай буюу автоматжуулсан байх шаардлагатай юм.

Зураг 3.1-д програмын ерөнхий бүтэц, үйл ажиллагааг схемчилж харуулсан байна. Үүнээс харахад програм нь трейнинг хийх өгөгдлийн тархалт болон туршилтын өгөгдөл гэсэн апк файлууд дээр ажиллах бөгөөд манай конволюшнал давхаргууд нь өгөгдлийн тархалт буюу олон тооны өгөгдлүүд дээрээс үндэслэн суралцаж моделийг үүсгэх юм. Тэрхүү модел дээрээс харьцуулалт хийж ангилагч нь туршилтын өгөгдлийг хортой болон хоргүй гэж ангилна.



Зураг 3.1: Програмын ерөнхий бүтэц

3.1 Андройд Апп-ууд дээрх дүн шинжилгээ

1. **Андройд апп** - Андройд аппликейшнд байх бүх шаардлагатай файлууд нь APK файл дотор багцлагдсан байдалтай байдаг. APK Файл бүр res хавтас, asset хавтас, classes.dex файл, AndroidManifest.xml файл, lib хавтас болон META-INF хавтас гэсэн файлуудыг агуулж байдаг. Бидний хэрэгжүүлэлтэнд хамааралтай файлууд бол classes.dex болон AndroidManifest.xml хоёр юм. APK файлын хэрэглэгдэж байгаа сангууд нь lib хавтас дотор байрладаг мөн шаардлагатай зураг, дуу гэх мэт файлуудыг res болон asset хавтаснуудад хадгалдаг байна.
   classes.dex файл нь аппликейшний байткодыг агуулж байдаг бөгөөд энэ нь "exe"файлтай адил хөрвүүлж ажиллуулах зорилготой юм. Хэдийгээр дээр дурьдсан class.dex-ээс бусад файлуудад хортой код агуулагдах болсон ч уламжлалт аргаар буюу зөвхөн class.dex файлыг задлан шинжилгээ хийх энэхүү судалгааны зорилго юм. Үүний тулд python хэл дээр нээлттэй эх байдлаар хөгжүүлэгдсэн **Androguard** санг ашиглан classes.dex файл дотроос smali кодыг гарган авна.
   Androguard сангаас classes.dex файлыг задлахад шаардлагатай функцуудыг дуудаж байна.

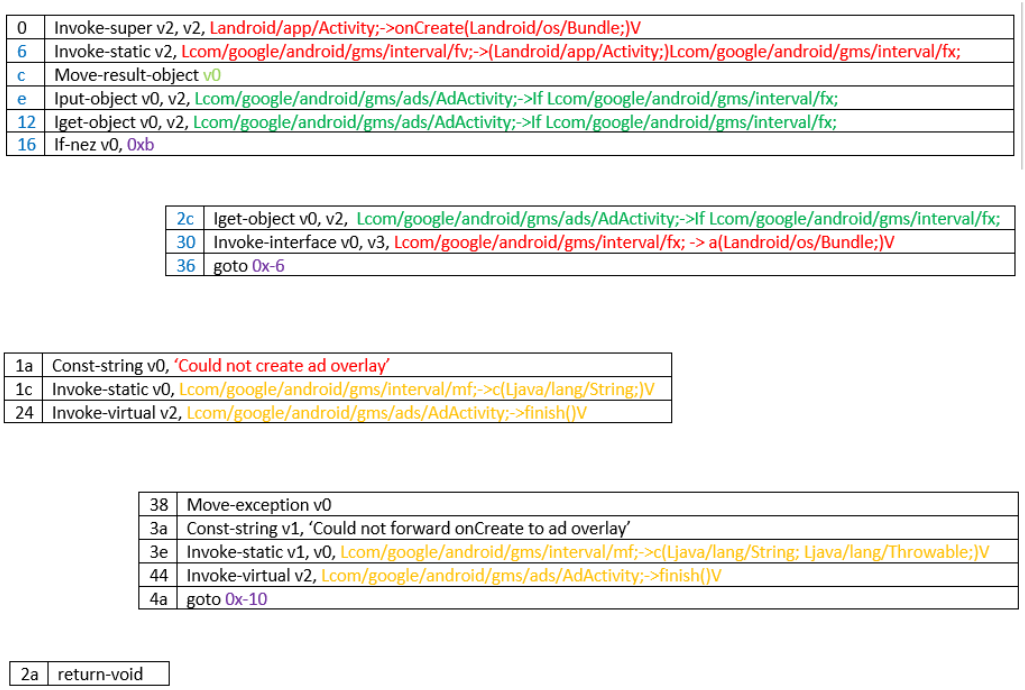
```
from androguard.core.bytecodes import apk, dvm
from androguard.core.analysis import analysis
```

   Зааж өгсөн хавтсанд байгаа apk өргөтгөлтэй файлуудыг задлах хэсэг байна.

```
if path.endswith('.apk'):
    app = apk.APK(path)
    app_dex = dvm.DalvikVMFormat(app.get_dex())
else:
    app_dex = dvm.DalvikVMFormat(open(path, "rb").read())

app_x = analysis.Analysis(app_dex)
```

2. **Псевдо-динамик код шинжилгээ** - APK файлыг задлан classes.dex файлыг гарган авсаны дараа түүний эх кодон дээр шинжилгээ хийнэ. Манай өгөгдлийн тархалт дээр байрлаж буй энгийн/benign/ apk файлуудыг задлан нийтлэг ашиглагддаг функцуудээр утга буцаах/return/ хүртэл дамжих байдлаар бүх функцын зүй тогтлыг суралцаж авна. Ингэснээр APK файл бүрээс санамсаргүйгээр зүй тогтлуудыг цуглуулж файлд хадгалснаар Deep Neural Network модел-г бүтээж байгаа болно.



Зураг 3.2: Андройд аппликеишны энгийн функцын жишээ

3.1.1 Апп ангилахад Neural сүлжээг бэлдэх

```
def extract_all_features():
    print "loading␣dict..."
    external_api_dict = cPickle.load(open("15
        IT201_15IT217_M1_common_dict_300.save", "rb"))
    print "done!"

    # X = []if __name__ if __name__ == '__main__':== '__main__':
    # Y = []

    # path_list = ["Dataset/benign","Dataset/all_drebin"]
    path_list = ["Dataset/all_drebin"]
    index = 0
    for i in range(2):

        count = 0
        for path in os.listdir(path_list[i])[::-1]:

            count += 1

            if (count == 34):
                break

            index += 1
```

```
23
24          print count, os.path.join(path_list[i], path)
25
26          try:
27              # X.append(get_feature_vector(os.path.join(path_list
                    [i],path), external_api_dict))
28              # Y.append(i)
29              x = get_compressed_feature_vector(os.path.join(
                    path_list[i], path), external_api_dict)
30              data_point = {}
31              data_point['x'] = x
32              data_point['y'] = 1
33              # fp = open(os.path.join('features',str(index) + '.
                    save'), 'wb')
34              # fp = open(os.path.join('all_compressed_features',
                    str(path) + '.save'), 'wb')
35              fp = open(os.path.join('acf2', str(path) + '.save'),
                     'wb')
36              cPickle.dump(data_point, fp, protocol=cPickle.
                    HIGHEST_PROTOCOL)
37              fp.close()
38
39          except Exception as e:
40
41              print "exception␣occured"
42              print e
43      return
```

Энэхүү функцын тусламжтайгаар хортой болон хоргүй өгөгдлийн тархалтаас суралцан авсан ялгаатай зүй тогтлуудыг "all_features_compressed"болон "acf2"гэсэн хоёр хавтсанд ".save"өргөтгөлтэйгөөр ялган хадгалж байна.

Тэгэхээр бид Convolutional Neural Сүлжээгээр аливааг зүйлийг ангилахын тулд эхлээд оролтын утгуудын хэмжээг ижил байхаар зохицуулах шаардлагатай. Бүх оролтын дараалал бүгд ижил байх ёстой бөгөөд апп-уудаас гарган авсан файлуудыг 20 килобайт байхаар шийдвэрлэсэн болно.

One-Hot Vector Апп бүрээс гарган авсан ижил хэмжээтэй файлуудыг one-hot vector аргаар кодчилно. Үүний тулд аппликэйшн бүрээс гарган авсан ижил хэмжээтэй файлуудыг нэг хавтсанд хадгалж өөрөөр хэлвэл багцалсан байгаа бөгөөд үүнийг A гэвэл түүний хэмжээ m = |A| байх ёстой. A-ын нэг i-р элементийн one-hot вектор нь vi=1 байх бөгөөд бусад бүх элементүүд 0 байх ёстой. Энэ нь чиглэлийг зааж байгаа учраас. One-hot аргын дагуу A-ын сегменд бүр NxM харьцаатай матрицад хөрвүүлэгдэн кодчилогдох учиртай юм.

```
1 def one_2d_to_3d(a,new_dim):
```

```
    b = np.zeros(( a.shape[0], a.shape[1], new_dim))
    layer_idx = np.arange(a.shape[0]).reshape(a.shape[0], 1)

    component_idx = np.tile(np.arange(a.shape[1]), (a.shape[0], 1))

    b[layer_idx, component_idx, a] = 1

    b = b.transpose(1,0,2)
    # voila!
    # print(b)

    # print b.shape

    return b


def uncompress(a,num_dim):

    ans = []
    for x in a:

        oh = one_2d_to_3d(np.squeeze(x),num_dim)
        # print 'oh',oh.shape
        ans.append(oh)

    ans2 = np.vstack(ans)

    ans2 = np.expand_dims(ans2,3)

    return ans2
```

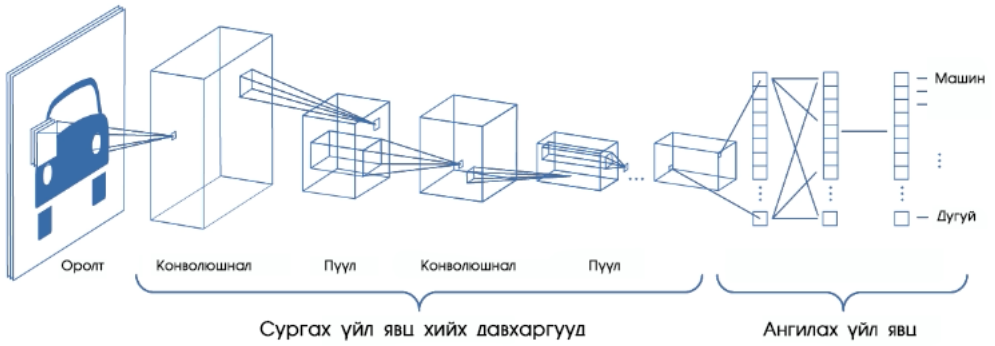
Convolutional Neural сүлжээ болон ангилагчыг байгуулах нь

Сегмент бүрт ангилал хийхэд CNN-г сургаж эхлэнэ. Өгөгдлийн тархалтнаас зүй тогтлоороо ялгарсан сегментүүдээс APK файлтай зүй тогтлоор нь харьцуулан ангилал хийнэ. Зураг 3.3-с харахад манай архитектур нь NxM харьцаатай матрицыг оролтын давхаргаруу оруулснаар апп-ын зүй тогтлыг хадгалж буй сегментээс суралцаж ангилахад бэлэн болох эхний шат дуусна. Pooling давхарга нь дараагийн convolutional давхаргатай холбогдох болно.

```
import tensorflow as tf
from data_reader import load_data
import numpy as np
from uncompress import *
import os

slim = tf.contrib.slim

```



Зураг 3.3: Манай convolutional сүлжээний архитектур жишээгээр

```
9  def lrelu(alpha):
10     def op(inputs):
11         return tf.maximum(alpha * inputs, inputs, name='leaky_relu'
               )
12     return op
13
14 def conv_net(input):
15
16     with slim.arg_scope([slim.conv2d, slim.fully_connected], #using
           scope to avoid mentioning the paramters repeatdely
17                                     activation_fn=lrelu(0.005),
18                                     weights_initializer=tf.
                                           truncated_normal_initializer
                                           (0.0, 0.01),
19                                     weights_regularizer=slim.
                                           l2_regularizer(0.0005)):
20   #net = slim.max_pool2d(input,(1,4),(1,4), padding='VALID', scope=
         'pool_0')
21
22
23         net = slim.conv2d(input, 512, (5,86796), 1, padding='SAME',
               scope='conv_1')
24
25         net = slim.max_pool2d(net, (4,1),4, padding='VALID', scope=
               'pool_2')
26
27         net = slim.conv2d(net, 512, (5,1), 1, scope='conv_3')
28
29         net = slim.max_pool2d(net, (4,1),4, padding='VALID', scope=
               'pool_4')
30
```

```
31   net = slim.conv2d(net, 1024, (5,1), 1, scope='conv_4')
32
33         net = slim.flatten(net, scope='flatten_5')
34
35   '''net␣=␣slim.fully_connected(net,␣1024,␣scope='fc_6',
       activation_fn=tf.nn.softmax)
36
37 ␣␣net␣=␣slim.fully_connected(net,␣256,␣scope='fc_7',activation_fn=
     tf.nn.softmax)
38
39 ␣␣␣␣␣␣␣␣net␣=␣slim.fully_connected(net,␣2,␣scope='fc_8',
     activation_fn=tf.nn.softmax)'''
40
41   net = slim.fully_connected(net, 4096, scope='fc5')
42   net = slim.dropout(net, 0.5, scope='dropout6')
43   net = slim.fully_connected(net, 4096, scope='fc7')
44   net = slim.dropout(net, 0.5, scope='dropout8')
45   net = slim.fully_connected(net,2, activation_fn=None, scope='fc9'
       )
46
47
48     return net
49
50 def one_hot(batch_size,Y):
51
52     B = np.zeros((batch_size,2))
53
54     B[np.arange(batch_size),Y] = 1
55
56     return B
57
58 if __name__=='__main__':
59
60     #os.environ['CUDA_VISIBLE_DEVICES'] = ''
61     # print one_hot(3,np.array((1,0,1)))
62     # exit(0)
63     # Training Parameters
64     learning_rate = 0.00001
65     num_epoch = 5
66     batch_size = 2
67     display_step = 1
68     input_size = 50
69     num_classes = 2
70
71
72
73     X = tf.placeholder(tf.float32, [None, input_size,86796,1])
```

```
74    Y = tf.placeholder(tf.float32, [None, num_classes])
75    #logits = conv_net(X)
76    #prediction = tf.nn.softmax(logits)
77    prediction = conv_net(X)
78
79    # Define loss and optimizer
80    '''loss_op␣=␣tf.reduce_mean(tf.nn.
          softmax_cross_entropy_with_logits(
81  ␣␣logits=logits,␣labels=Y))'''
82    loss_op = slim.losses.softmax_cross_entropy(prediction, Y)
83
84    tf.summary.scalar('loss',loss_op)
85    optimizer = tf.train.AdagradOptimizer(learning_rate =
          learning_rate)
86    train_op = optimizer.minimize(loss_op)
87
88
89    # Evaluate model
90    correct_pred = tf.equal(tf.argmax(prediction, 1), tf.argmax(Y,
          1))
91    accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))
92    tf.summary.scalar('accuracy',accuracy)
93
94    # Initialize the variables (i.e. assign their default value)
95    init = tf.global_variables_initializer()
96
97    data_X,data_Y = load_data()
98
99    indices = np.random.permutation(np.arange(data_X.shape[0]))
100
101   data_X = data_X[indices,:,:]
102
103   data_Y = data_Y[indices]
104
105
106
107   merged = tf.summary.merge_all()
108   saver = tf.train.Saver()
109
110   with tf.Session() as sess:
111
112 train_writer = tf.summary.FileWriter("cnn_logs_8/", sess.graph)
113
114 # Run the initializer
115 sess.run(init)
116
117 '''
```

```
118
119 ␣␣print␣'restoring session'
120 ␣␣saver.restore(sess,␣"logs3/epoch0i180.ckpt")
121 ␣␣print␣'done loading'
122 ␣␣#␣exit(0)␣'''
123
124  i = 0
125  print 'started␣training'
126  for epoch in range(num_epoch):
127      for step in range(data_X.shape[0]/batch_size):
128          batch_x, batch_y = data_X[step*batch_size:(step+1)*
                batch_size],\
129          data_Y[step*batch_size:(step+1)*batch_size]
130
131
132
133          batch_x = uncompress(batch_x,86796)
134
135          # print batch_y
136          batch_y = one_hot(batch_size,batch_y)
137
138          batch_y = np.repeat(batch_y,50,axis=0)
139
140          # print batch_y
141
142          assert(batch_x.shape[0]==batch_y.shape[0])
143
144          # print batch_x.shape
145          # print batch_y.shape
146
147          # exit(0)
148
149          # Run optimization op (backprop)
150          _,summary = sess.run([train_op,merged], feed_dict={X:
                batch_x, Y: batch_y})
151          train_writer.add_summary(summary, i)
152          if step % display_step == 0:
153              # Calculate batch loss and accuracy
154              loss, acc,summary = sess.run([loss_op, accuracy,merged
                    ], feed_dict={X: batch_x,
155               Y: batch_y})
156              print("LR␣:␣"+str(learning_rate)+"␣Epoch␣:␣" + str(
                    epoch) + "␣Step␣" + str(step) + ",␣Minibatch␣Loss=
                    ␣" + \
157                   "{:.4f}".format(loss) + ",␣Training␣Accuracy=␣" +
                        \
158                   "{:.3f}".format(acc))
```

```
159
160             # train_writer.add_summary(summary, step)
161
162         if i%20 == 0:
163
164             print 'saving checkpoint'
165             save_path = saver.save(sess, os.path.join('cnn_logs_8'
                    ,'epoch'+str(epoch)+\
166             'i'+str(i)+'.ckpt'))
167             print("Model saved in path: %s" % save_path)
168
169         i+=1
170
171     # print("Optimization Finished!")
```

3.1.2 LSTM/Long-Short Term Memory/ ангиллын туршилт

Манай ангилал нь Апп-уудын зүй тогтол буюу түүний сегментүүд дээр суурилан ажилладаг. Long-Short Term Memory нь дарааллыг шийдвэрлэхэд зориулсан neural network-н архитектур юм. Тиймээс үүн дээр суурилсан ангиллыг өөрийн өгөгдлийн тархалт дээр хийж үзсэн болно. 256 давхарга дээр туршилт хийсэн бөгөөд алхам бүрт one-hot вектор өгөгдлийн тархалт дээрээс үндэслэн магадлал нь багаас их рүү өссөөр гаралт нь хортой болон хоргүй гэсэн магадлалыг өгнө.

```
1   with tf.Session() as sess:
2
3       train_writer = tf.summary.FileWriter("logs4/",
4                                     sess.graph)
5
6       # Run the initializer
7       sess.run(init)
8
9
10      # print 'restoring session'
11      # saver.restore(sess, "logs3/epcoh0i0.ckpt")
12      # print 'done loading'
13      # exit(0)
14
15      i = 0
16      print 'started training'
17      for epoch in range(num_epoch):
18          for step in range(data_X.shape[0]/batch_size):
19              batch_x, batch_y = data_X[step*batch_size:(step+1)*
                    batch_size],\
```

```
20                data_Y[step*batch_size:(step+1)*batch_size]
21
22
23
24                batch_x = uncompress(batch_x,86796)
25
26                # print batch_y
27                batch_y = one_hot(batch_size,batch_y)
28
29                batch_y = np.repeat(batch_y,50,axis=0)
30
31                # print batch_y
32
33                assert(batch_x.shape[0]==batch_y.shape[0])
34
35                # print batch_x.shape
36                # print batch_y.shape
37
38                # exit(0)
39
40                # Run optimization op (backprop)
41                _,summary = sess.run([train_op,merged], feed_dict={X
                      : batch_x[:,:,:,0], Y: batch_y})
42                train_writer.add_summary(summary, i)
43                if step % display_step == 0:
44                    # Calculate batch loss and accuracy
45                    loss, acc,summary = sess.run([loss_op, accuracy,
                          merged], feed_dict={X: batch_x[:,:,:,0],
46                    Y: batch_y})
47                    print("Epoch␣:␣" + str(epoch) + "␣Step␣" + str(
                          step) + ",␣Minibatch␣Loss=␣" + \
48                     "{:.4f}".format(loss) + ",␣Training␣Accuracy=
                            ␣" + \
49                     "{:.3f}".format(acc))
50
51                    # train_writer.add_summary(summary, step)
52
53                if i%20 == 0:
54
55                    print 'saving␣checkpoint'
56                    save_path = saver.save(sess, os.path.join('logs4
                          ','epcoh'+str(epoch)+\
57                    'i'+str(i)+'.ckpt'))
58                    print("Model␣saved␣in␣path:␣%s" % save_path)
```

3.1.3 Naive-Bayes Ангилагч

Convolutional давхаргаас гарган авсан модел дээрээс үндэслэн хортой эсвэл хоргүй эсэхийг шалгах АПК файлыг "Testcase"хавтсанд байрлуулан Naive-Bayes ангилагчаар илрүүлнэ.

```

from data_reader import load_data
import numpy as np
from uncompress import *
import os
from sklearn.naive_bayes import GaussianNB
import cPickle



if __name__=='__main__':

    batch_size = 1

    data_X,data_Y = load_data()

    indices = np.random.permutation(np.arange(data_X.shape[0]))

    data_X = data_X[indices,:,:]
    data_Y = data_Y[indices]

    X = []
    Y = []

    gnb = GaussianNB()

    fp = open(os.path.join('nb_logs','nb_object' + '.save'), 'wb')
    cPickle.dump(gnb, fp, protocol = cPickle.HIGHEST_PROTOCOL)
    fp.close()


    s = 0.0

    i = 0

    for step in range(data_X.shape[0]/batch_size):

        print "Step:",step

        batch_x, batch_y = data_X[step*batch_size:(step+1)*
           batch_size],data_Y[step*batch_size:(step+1)*batch_size]
```

```
41
42          batch_x = uncompress(batch_x,86796)
43          # print batch_x.shape
44
45
46
47          batch_x = np.sum(batch_x,axis=1)
48          # print batch_x.shape
49          batch_x = np.squeeze(batch_x)
50          # print batch_x.shape
51
52          # print 'y'
53          # print batch_y.shape
54          batch_y = np.repeat(batch_y,50,axis=0)
55          # print batch_y.shape
56
57          gnb.partial_fit(batch_x,batch_y,classes=[0,1])
58
59
60          # X.append(batch_x)
61          # Y.append(batch_y)
62          # break
63
64      # X = np.vstack(X)
65      # Y = np.squeeze(np.asarray(Y))
66
67      # print X.shape,Y.shape
68      for step in range(data_X.shape[0]/batch_size):
69
70          print "Step:",step
71
72          batch_x, batch_y = data_X[step*batch_size:(step+1)*
                batch_size],data_Y[step*batch_size:(step+1)*batch_size]
73
74          batch_x = uncompress(batch_x,86796)
75          # print batch_x.shape
76
77
78
79          batch_x = np.sum(batch_x,axis=1)
80          # print batch_x.shape
81          batch_x = np.squeeze(batch_x)
82          # print batch_x.shape
83
84          # print 'y'
85          # print batch_y.shape
86          batch_y = np.repeat(batch_y,50,axis=0)
```

```
87          # print batch_y.shape
88
89          # gnb.partial_fit(batch_x,batch_y,classes=[0,1])
90
91          x = gnb.score(batch_x,batch_y)
92
93          print x
94
95          s += x
96          i +=1
97
98          print 'average␣:␣', s/i
99
100     # gnb.fit(X,Y)
101     #
102     print s/i
103
104     fp = open(os.path.join('nb_logs','nb_object' + '.save'), 'wb')
105     cPickle.dump(gnb, fp, protocol = cPickle.HIGHEST_PROTOCOL)
106     fp.close()
```

3.2 Туршилтын үр дүн

1. **Өгөгдлийн тархалт:** CNN-г туршихад шаардлагатай хоёр төрлийн Андройд APK файл хэрэглэсэн. Энгийн/benign/ аппууд болон Хортой/Malware/ аппууд. Хортой аппуудыг Contagio Mobile репозиторигоос зөвхөн туршилтанд ашиглах зорилгоор багцалж авсан болно. Хортой 8 төрлийн нийт 116 аппууд дээр суурилан ажилласан.

Хортой кодны төрөл	**Тоо ширхэг**
Godless	8
Overlay Locker	19
Crisis Android	53
Android XBot	11
Sberbbanker	5
Marcher	6
Hummingbad	27
Xiny	7

Хүснэгт 3.1: Туршилтанд хэрэглэгдсэн хортой кодны тоо, төрөл

Үүнээс гадна мөн адил хоргүй энгийн APK файлуудыг Google дэлгүүрээс татаж авч цуглуулсан бөгөөд нийт 484 энгийн апп-ыг туршилтанд ашигласан бөгөөд файлуудыг хооронд нь давхцуулахгүйн тулд MD5 хашласан утгуудыг нь хооронд нь харьцуулж нийцсэн тохиолдлуудыг хассан болно.

Төрөл	**Тоо ширхэг**	**Төрөл**	**Тоо ширхэг**
System tools	114	Shopping	19
Communications	59	Food/Drink	61
Books/References	18	Puzzle games	37
Healt/Fitness	7	Arcade games	42
Photos	11	Board games	3
Themes/Wallpapers	61		

Хүснэгт 3.2: Туршилтанд хэрэглэгдсэн хоргүй кодны тоо, төрөл

2. **Хортой програмыг ангилсан үр дүн**
   Програмын хортой болон хоргүй гэсэн магадлалыг тогтоохдоо дундаж магадлалыг 50% гэж үзэж байгаа бөгөөд 50-аас их хувиар таамаглаж байвал тэрхүү програм нь хортой бөгөөд эсрэгээрээ 50-аас бага хувь байвал хоргүй програм байна хэмээн тооцоолж байгаа болно.

Конволюшнал давхарга нь нийт 3 ширхэгээс бүрдэж байгаа бөгөөд эхний давхаргаас суралцсан модел дээрээс үндэслэн дараагийн давхаргаар суралцана. Мөн хоёр дахь давхаргаас үүссэн модел дээрээс дараагийн давхаргаар суралцуулан эцсийн модел үүсэх зарчмаар трейнинг хийж байгаа. Миний бие эхний давхаргаас суралцсан модел дээрээс ангилалыг хийхэд 50%-д маш ойрхон буюу арван мянганы/10'000/ нарийвчлалтай магадлал таамаглаж байсан нь хортой болон хоргүй хэмээн ангилахад хангалттай бус мэдээлэл юм. Өөрөөр хэлвэл эхний трейнинг хийх процесс нь модел хангалттай суралцаагүй байгааг илтгэх бөгөөд бидний эцсийн үр дүнг харуулж чадахгүй гэсэн үг юм.

Тэрхүү моделийг үүсгэсний дараа дараагийн давхаргаар суралцуулах процесс явагдсан бөгөөд үүний үр дүнд манай моделийн хэмжээ эрс нэмэгдсэн бөгөөд дахин ангилагч функцээр дамжуулан үр дүнг харахад 49-51 хооронд буюу аравтын/10/ нарийвчлалтай үр дүнг харуулж байв. Хүснэгт 3.3-д туршилтаар оруулсан хортой програмыг давхарга давхаргуудаас үүссэн модел дээрээс үндсэслэн ангилал хийхэд гарсан магадлалыг харуулав.

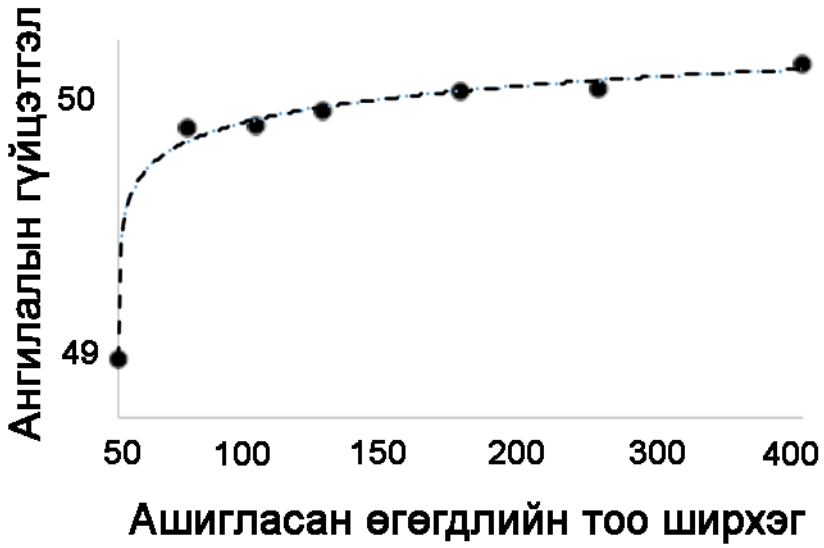
Модел	**Магадлал**	**Нарийвчлал**
CNN1	50%	50.0006%
CNN2	50.4%	50.4593%
CNN3	54.9%	54.9861%

Хүснэгт 3.3: Давхаргуудын гүйцэтгэл

Хүснэгт 3.3-д ашигласан хортой програм нь "Крайсис Андройд"буюу хамгийн их өргөн тархсан, нийтлэг хортой програм тул хортой байх магадлал бусад төрлийн хортой програмуудаасаа хамгийн их байгаа учир үүгээр жишээ авсан болно.

3. **Трейнинг өгөгдлийн хэмжээний нөлөө**

   Дээр дурьдсанчлан АПК бүрээс гарган авсан сегментүүдийг **One-Hot Vector**-н тусламжтайгаар нэгэн багц болгож байгаа. Тэрхүү багцын хэмжээнээс хамаарч магадлал 50-иас их эсвэл бага байгааг ажигласан юм. Өөрөөр хэлвэл өгөгдлийн тархалтын хэмжээ их байхын хэрээр програм үр дүндээ хүрэх магадлал өндөр байна гэсэн үг. Зураг 3.4-аас харахад трейнинг хийх өгөгдлийн тархалт нь их байх буюу модел нь их хэмжээний өгөгдлөөс суралцаж байвал үр дүнд хүрэх өндөр магадлалтай байгааг харж байна.



Зураг 3.4: Caption

3.3 Ерөнхий дүгнэлт

Энэхүү судалгааны ажлаараа би Гүний мэдрэлийн эсийн сүлжээ, ялангуяа Конволюшнал эсийн сүлжээний давуу талыг ашиглан програм хангамжийг ангилсан болно. Энэхүү аргыг ашиглах болсон шалтгаан нь конволюшнал эсийн сүлжээ нь текстийг ангилахад түлхүү ашиглагддаг учраас андройдын эх кодыг текст хэлбэр лүү хөрвүүлэн түүн дээрээ шинжилгээ хийж зүй тогтлыг суралцуулан өөрийн моделийг үүсгэх нь шинэлэг бөгөөд бусад аргуудаас үр дүнтэй байх боломжтой эсэхийг хэрэгжүүлэлттэйгээр туршиж үзсэн болно.

Хортой програмыг илрүүлэх болон ангилахад машин сургалтын нэгэн төрөл болох энэхүү аргыг хэрэгжүүлэх нь илүү зохимжтой болох нь судалгааны үр дүнд харагдав. Бусад n-gram арга дээр суурилсан *Support-Vector Machine*, *Long-Short term memory* гэх мэт аргуудтай харьцуулан туршилт хийж үзсэн тул манай судалгааны туршилтын үр дүн хортой програмыг илрүүлэх ажил дээр илүү магадлал өндөртэйгөөр ажиллаж байсан нь үйл ажиллагааны явцад батлагдсан.

Судалгааны үйл явцад машиныг сургах процессыг *tensorflow* санг ашиглан давхаргуудаа бүтээсэн бөгөөд цаашлаад ангилах явцыг *scikit learning* сангийн *Naive-Bayes* ангилах алгоритмыг ашиглан эцсийн үр дүнг харсан. Naive-Bayes нь текст ангилахад түлхүү ашиглагддаг ангилагч алгоритм юм. Үүний тулд андройд програмыг текстрүү хөрвүүлэх ажиллагаа нь энэхүү судалгааны ажлын хамгийн ярвигтай хэсэг байсан гэж хэлэх нь зүйтэй юм.

Улмаар өгөгдлийн тархалт буюу apk файлуудыг бүгдийг нь текстрүү хөрвүүлэн/Input Raw Data/ түүн дээрээсээ суралцуулан өөрсдийн моделийгг гарган авсан болно. Тэрхүү модел дээрээсээ үндэслэн *Naive-Bayes* ангилагч нь хортой болон хоргүй байх магадлалыг гарган таамаглах нь програмын эцсийн үр дүнд хүрч байгааг илэрхийлнэ.

БҮЛЭГ 4

Ном зүй

Bibliography

[1] `https://www.statista.com/`
[хандсан 23/10/2018]

[2] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu
"Large-scale malware classication using random projections and neural networks
[хандсан 26/10/2018]

[3] Y. Kim
Convolutional neural networks for sentence classication
[хандсан 26/10/2018]

[4] A. Desnos
"Androguard-reverse engineering, malware and goodware analysis of android applications
[хандсан 03/11/2018]

[5] T. Vidas and N. Christin
"Evading android runtime analysis via sandbox detection
[хандсан 08/11/2018]

[6] The Theano Development Team
Theano: A python framework for fast computation of mathematical expressions
[хандсан 21/11/2018]

[7] F. Scikit-learn Developlment TeamPedregosa, G. Varoquaux, A. Gramfort
"Scikit-learn: Machine learning in python
[хандсан 30/11/2018]

[8] SciKit- Learn
`https://scikit-learn.org/stable/modules/classes.html`
[хандсан 06/11/2018]

ХАВСРАЛТ А

Хавсралт

```
    from androguard.core.bytecodes import apk, dvm
    from androguard.core.analysis import analysis
```

```
    if path.endswith('.apk'):
        app = apk.APK(path)
        app_dex = dvm.DalvikVMFormat(app.get_dex())
    else:
        app_dex = dvm.DalvikVMFormat(open(path, "rb").read())

    app_x = analysis.Analysis(app_dex)
```

```
def extract_all_features():
    print "loading␣dict..."
    external_api_dict = cPickle.load(open("15
       IT201_15IT217_M1_common_dict_300.save", "rb"))
    print "done!"

    # X = []if __name__ if __name__ == '__main__':== '__main__':
    # Y = []

    # path_list = ["Dataset/benign","Dataset/all_drebin"]
    path_list = ["Dataset/all_drebin"]
    index = 0
    for i in range(2):

        count = 0
        for path in os.listdir(path_list[i])[::-1]:

            count += 1

            if (count == 34):
                break

            index += 1

            print count, os.path.join(path_list[i], path)

            try:
                # X.append(get_feature_vector(os.path.join(path_list
                   [i],path), external_api_dict))
                # Y.append(i)
                x = get_compressed_feature_vector(os.path.join(
                   path_list[i], path), external_api_dict)
                data_point = {}
                data_point['x'] = x
                data_point['y'] = 1
                # fp = open(os.path.join('features',str(index) + '.
                   save'), 'wb')
```

```
                # fp = open(os.path.join('all_compressed_features',
                    str(path) + '.save'), 'wb')
                fp = open(os.path.join('acf2', str(path) + '.save'),
                     'wb')
                cPickle.dump(data_point, fp, protocol=cPickle.
                    HIGHEST_PROTOCOL)
                fp.close()

            except Exception as e:

                print "exception occured"
                print e
        return
```

```
def one_2d_to_3d(a,new_dim):
    b = np.zeros(( a.shape[0], a.shape[1], new_dim))
    layer_idx = np.arange(a.shape[0]).reshape(a.shape[0], 1)

    component_idx = np.tile(np.arange(a.shape[1]), (a.shape[0], 1))

    b[layer_idx, component_idx, a] = 1

    b = b.transpose(1,0,2)
    # voila!
    # print(b)

    # print b.shape

    return b


def uncompress(a,num_dim):

    ans = []
    for x in a:

        oh = one_2d_to_3d(np.squeeze(x),num_dim)
        # print 'oh',oh.shape
        ans.append(oh)

    ans2 = np.vstack(ans)

    ans2 = np.expand_dims(ans2,3)

    return ans2
```

```
import tensorflow as tf
from data_reader import load_data
```

```
import numpy as np
from uncompress import *
import os

slim = tf.contrib.slim

def lrelu(alpha):
    def op(inputs):
        return tf.maximum(alpha * inputs, inputs, name='leaky_relu')
    return op

def conv_net(input):

    with slim.arg_scope([slim.conv2d, slim.fully_connected], #using scope to avoid mentioning the paramters repeatdely
                                    activation_fn=lrelu(0.005),
                                    weights_initializer=tf.truncated_normal_initializer(0.0, 0.01),
                                    weights_regularizer=slim.l2_regularizer(0.0005)):
  #net = slim.max_pool2d(input,(1,4),(1,4), padding='VALID', scope='pool_0')


        net = slim.conv2d(input, 512, (5,86796), 1, padding='SAME', scope='conv_1')

        net = slim.max_pool2d(net, (4,1),4, padding='VALID', scope='pool_2')

        net = slim.conv2d(net, 512, (5,1), 1, scope='conv_3')

        net = slim.max_pool2d(net, (4,1),4, padding='VALID', scope='pool_4')

  net = slim.conv2d(net, 1024, (5,1), 1, scope='conv_4')

        net = slim.flatten(net, scope='flatten_5')

  '''net = slim.fully_connected(net, 1024, scope='fc_6', activation_fn=tf.nn.softmax)

    net = slim.fully_connected(net, 256, scope='fc_7',activation_fn=tf.nn.softmax)

```

```
␣␣␣␣␣␣␣␣net␣=␣slim.fully_connected(net,␣2,␣scope='fc_8',
    activation_fn=tf.nn.softmax)'''

  net = slim.fully_connected(net, 4096, scope='fc5')
  net = slim.dropout(net, 0.5, scope='dropout6')
  net = slim.fully_connected(net, 4096, scope='fc7')
  net = slim.dropout(net, 0.5, scope='dropout8')
  net = slim.fully_connected(net,2, activation_fn=None, scope='fc9'
      )


    return net

def one_hot(batch_size,Y):

    B = np.zeros((batch_size,2))

    B[np.arange(batch_size),Y] = 1

    return B

if __name__=='__main__':

    #os.environ['CUDA_VISIBLE_DEVICES'] = ''
    # print one_hot(3,np.array((1,0,1)))
    # exit(0)
    # Training Parameters
    learning_rate = 0.00001
    num_epoch = 5
    batch_size = 2
    display_step = 1
    input_size = 50
    num_classes = 2



    X = tf.placeholder(tf.float32, [None, input_size,86796,1])
    Y = tf.placeholder(tf.float32, [None, num_classes])
    #logits = conv_net(X)
    #prediction = tf.nn.softmax(logits)
    prediction = conv_net(X)

    # Define loss and optimizer
    '''loss_op␣=␣tf.reduce_mean(tf.nn.
       softmax_cross_entropy_with_logits(
␣␣logits=logits,␣labels=Y))'''
    loss_op = slim.losses.softmax_cross_entropy(prediction, Y)
```

```
83
84    tf.summary.scalar('loss',loss_op)
85    optimizer = tf.train.AdagradOptimizer(learning_rate =
          learning_rate)
86    train_op = optimizer.minimize(loss_op)
87
88
89    # Evaluate model
90    correct_pred = tf.equal(tf.argmax(prediction, 1), tf.argmax(Y,
          1))
91    accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))
92    tf.summary.scalar('accuracy',accuracy)
93
94    # Initialize the variables (i.e. assign their default value)
95    init = tf.global_variables_initializer()
96
97    data_X,data_Y = load_data()
98
99    indices = np.random.permutation(np.arange(data_X.shape[0]))
100
101   data_X = data_X[indices,:,:]
102
103   data_Y = data_Y[indices]
104
105
106
107   merged = tf.summary.merge_all()
108   saver = tf.train.Saver()
109
110   with tf.Session() as sess:
111
112  train_writer = tf.summary.FileWriter("cnn_logs_8/", sess.graph)
113
114  # Run the initializer
115  sess.run(init)
116
117  '''
118
119 ␣␣print␣'restoring session'
120 ␣␣saver.restore(sess,␣"logs3/epoch0i180.ckpt")
121 ␣␣print␣'done loading'
122 ␣␣#␣exit(0)␣'''
123
124  i = 0
125  print 'started␣training'
126  for epoch in range(num_epoch):
127      for step in range(data_X.shape[0]/batch_size):
```

```
128         batch_x, batch_y = data_X[step*batch_size:(step+1)*
                batch_size],\
129         data_Y[step*batch_size:(step+1)*batch_size]
130
131
132
133         batch_x = uncompress(batch_x,86796)
134
135         # print batch_y
136         batch_y = one_hot(batch_size,batch_y)
137
138         batch_y = np.repeat(batch_y,50,axis=0)
139
140         # print batch_y
141
142         assert(batch_x.shape[0]==batch_y.shape[0])
143
144         # print batch_x.shape
145         # print batch_y.shape
146
147         # exit(0)
148
149         # Run optimization op (backprop)
150         _,summary = sess.run([train_op,merged], feed_dict={X:
                batch_x, Y: batch_y})
151         train_writer.add_summary(summary, i)
152         if step % display_step == 0:
153             # Calculate batch loss and accuracy
154             loss, acc,summary = sess.run([loss_op, accuracy,merged
                    ], feed_dict={X: batch_x,
155              Y: batch_y})
156             print("LR␣:␣"+str(learning_rate)+"␣Epoch␣:␣" + str(
                    epoch) + "␣Step␣" + str(step) + ",␣Minibatch␣Loss=
                    ␣" + \
157                   "{:.4f}".format(loss) + ",␣Training␣Accuracy=␣" +
                          \
158                   "{:.3f}".format(acc))
159
160             # train_writer.add_summary(summary, step)
161
162         if i%20 == 0:
163
164             print 'saving␣checkpoint'
165             save_path = saver.save(sess, os.path.join('cnn_logs_8'
                    ,'epoch'+str(epoch)+\
166             'i'+str(i)+'.ckpt'))
167             print("Model␣saved␣in␣path:␣%s" % save_path)
```

```
168
169         i+=1
170
171     # print("Optimization␣Finished!")
```

```
1   with tf.Session() as sess:
2
3       train_writer = tf.summary.FileWriter("logs4/",
4                                    sess.graph)
5
6       # Run the initializer
7       sess.run(init)
8
9
10      # print 'restoring␣session'
11      # saver.restore(sess, "logs3/epcoh0i0.ckpt")
12      # print 'done␣loading'
13      # exit(0)
14
15      i = 0
16      print 'started␣training'
17      for epoch in range(num_epoch):
18          for step in range(data_X.shape[0]/batch_size):
19              batch_x, batch_y = data_X[step*batch_size:(step+1)*
                    batch_size],\
20              data_Y[step*batch_size:(step+1)*batch_size]
21
22
23
24              batch_x = uncompress(batch_x,86796)
25
26              # print batch_y
27              batch_y = one_hot(batch_size,batch_y)
28
29              batch_y = np.repeat(batch_y,50,axis=0)
30
31              # print batch_y
32
33              assert(batch_x.shape[0]==batch_y.shape[0])
34
35              # print batch_x.shape
36              # print batch_y.shape
37
38              # exit(0)
39
40              # Run optimization op (backprop)
41              _,summary = sess.run([train_op,merged], feed_dict={X
                    : batch_x[:,:,:,0], Y: batch_y})
```

```
                train_writer.add_summary(summary, i)
                if step % display_step == 0:
                    # Calculate batch loss and accuracy
                    loss, acc,summary = sess.run([loss_op, accuracy,
                       merged], feed_dict={X: batch_x[:,:,:,0],
                    Y: batch_y})
                    print("Epoch␣:␣" + str(epoch) + "␣Step␣" + str(
                       step) + ",␣Minibatch␣Loss=␣" + \
                        "{:.4f}".format(loss) + ",␣Training␣Accuracy=
                           ␣" + \
                        "{:.3f}".format(acc))

                    # train_writer.add_summary(summary, step)

                if i%20 == 0:

                    print 'saving␣checkpoint'
                    save_path = saver.save(sess, os.path.join('logs4
                       ','epcoh'+str(epoch)+\
                    'i'+str(i)+'.ckpt'))
                    print("Model␣saved␣in␣path:␣%s" % save_path)
```

```

from data_reader import load_data
import numpy as np
from uncompress import *
import os
from sklearn.naive_bayes import GaussianNB
import cPickle



if __name__=='__main__':

    batch_size = 1

    data_X,data_Y = load_data()

    indices = np.random.permutation(np.arange(data_X.shape[0]))

    data_X = data_X[indices,:,:]
    data_Y = data_Y[indices]

    X = []
    Y = []

    gnb = GaussianNB()

```

```
27    fp = open(os.path.join('nb_logs','nb_object' + '.save'), 'wb')
28    cPickle.dump(gnb, fp, protocol = cPickle.HIGHEST_PROTOCOL)
29    fp.close()
30
31
32    s = 0.0
33
34    i = 0
35
36    for step in range(data_X.shape[0]/batch_size):
37
38        print "Step:",step
39
40        batch_x, batch_y = data_X[step*batch_size:(step+1)*
              batch_size],data_Y[step*batch_size:(step+1)*batch_size]
41
42        batch_x = uncompress(batch_x,86796)
43        # print batch_x.shape
44
45
46
47        batch_x = np.sum(batch_x,axis=1)
48        # print batch_x.shape
49        batch_x = np.squeeze(batch_x)
50        # print batch_x.shape
51
52        # print 'y'
53        # print batch_y.shape
54        batch_y = np.repeat(batch_y,50,axis=0)
55        # print batch_y.shape
56
57        gnb.partial_fit(batch_x,batch_y,classes=[0,1])
58
59
60        # X.append(batch_x)
61        # Y.append(batch_y)
62        # break
63
64    # X = np.vstack(X)
65    # Y = np.squeeze(np.asarray(Y))
66
67    # print X.shape,Y.shape
68    for step in range(data_X.shape[0]/batch_size):
69
70        print "Step:",step
71
```

```
72          batch_x, batch_y = data_X[step*batch_size:(step+1)*
               batch_size],data_Y[step*batch_size:(step+1)*batch_size]
73
74          batch_x = uncompress(batch_x,86796)
75          # print batch_x.shape
76
77
78
79          batch_x = np.sum(batch_x,axis=1)
80          # print batch_x.shape
81          batch_x = np.squeeze(batch_x)
82          # print batch_x.shape
83
84          # print 'y'
85          # print batch_y.shape
86          batch_y = np.repeat(batch_y,50,axis=0)
87          # print batch_y.shape
88
89          # gnb.partial_fit(batch_x,batch_y,classes=[0,1])
90
91          x = gnb.score(batch_x,batch_y)
92
93          print x
94
95          s += x
96          i +=1
97
98          print 'average : ', s/i
99
100     # gnb.fit(X,Y)
101     #
102     print s/i
103
104     fp = open(os.path.join('nb_logs','nb_object' + '.save'), 'wb')
105     cPickle.dump(gnb, fp, protocol = cPickle.HIGHEST_PROTOCOL)
106     fp.close()
```