eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Informatika Ingeniaritzako Gradua

Konputazioa

Gradu Amaierako Proiektua

Deep Learning eredu batean oinarritutako irudien analisi eta azterketaren prototipo orokorgarri baten garapena: NVIDIA Jetson TX1 plataforman inplementatuta

---

Egilea

Itsaso Rodríguez Moreno

informatika fakultatea
facultad de informática

2018

Laburpena

Proiektu honetan, konputagailu bidezko ikusmenean erabilitako metodoak landu dira irudien klasifikaziorako erabiltzeko helburuarekin. Horretarako, irudien aurreprozesaketa eta eredu desberdinen garapena gauzatu da. Zehazki, *machine learning* eta *deep learning* teknika desberdinak garatu eta probatu dira, irudi jakin batzuen klasifikazioan lortutako emaitzak aztertuz. NVIDIA-k eskainitako software bat, DIGITS, ere erabili da proba desberdinak egiteko.

Horrez gain, *deep learning*-erako erabiltzen den hardware desberdinarekin lan egin da, hala nola Jetson TX1, robot mugikorretan erabiltzen den teknologiarekin lehenengo kontaktu bat edukitzeko.

Resumen

En este proyecto se ha trabajado con diferentes métodos utilizados en visión por computador con el objetivo de usarlos para la clasificación de imágenes. Para ello, se ha llevado a cabo un pre-procesado de imágenes y se han desarrollado diferentes modelos de clasificación. Concretamente se han desarrollado y probado diferentes técnicas de *machine learning* y *deep learning*, analizando los resultados conseguidos en la clasificación de unas imágenes. También se ha utilizado un software que ofrece NVIDIA, DIGITS, para hacer diferentes pruebas.

Aparte de eso, se ha trabajado con diferente hardware utilizado para el *deep learning*, como Jetson TX1, con el fin de tener un primer contacto con las tecnologías utilizadas en los robots móviles.

Abstract

In this project we have worked with different methods used in computer vision with the objective of using them for the classification of images. For that, a pre-processing of images has been carried out and different classification models have been developed. Specifically, different techniques of machine learning and deep learning have been developed and tested, analyzing the results obtained in the classification of some images. It has also been used a software offered by NVIDIA, DIGITS, to do some more tests.

Apart from that, we have worked with different hardware used for deep learning, such as Jetson TX1, in order to have a first contact with the technologies used in mobile robots.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

1. KAPITULUA

Sarrera

Dokumentu hau 2018. urtean Itsaso Rodríguez Morenok UPV/EHU unibertsitateko Informatika Fakultatean egindako Gradu Amaierako Proiektuari dagokio. Proiektuko zuzendariak Basilio Sierra Araujo, Informatika Fakultateko irakaslea, eta José María Martínez-Otzeta, Informatika Fakultateko RSAIT taldeko ikertzailea, dira.

1.1 Helburuak

Proiektuaren helburu nagusia ikasketa sakonarekin (*deep learning*) trebatzea da. Konputagailu bidezko ikusmenerako *deep learning* teknikak landu nahi dira, etorkizunean robot mugikor batean erabiltzeko xedearekin. Hortaz, konputagailu bidezko ikusmenean erabili ohi diren neurona-sareak eta ikasketa sakonerako erabiltzen diren liburutegi desberdinak landuko dira. Horrez gain, NVIDIA-k eskaintzen dituen teknologiak ere erabiliko dira, software programak zein hardware gailuak.

1.2 Motibazioa

Gaur egun *deep learning* eta sare neuronalen erabilera ugaria da ataza desberdinak burutzeko, hala nola, objektuak sailkatzeko, testua sortzeko, bideojokoak automatikoki jokatzeko, etab. Kasu honetan, aplikazio jakin bat aztertuko dugu, irudien prozesamendua, makinek automatikoki ikusten dutena sailkatzea eta identifikatzea asko lantzen ari den

gaitasuna da eta.

Beraz, unean erabiltzen ari den teknika denez, *deep learning*-ak irudien prozesamendurako ematen dituen aukerak aztertzea erabaki da. Zehazki, *deep learning* sare neuronalekin erabiliko da proiektu osoan zehar, hau da, sare neuronal sakonak (*deep neural networks*) erabiliko dira.

1.3 Artearen egoera

Azkeneko urteetan, ordenagailu bidezko ikusmena guztiz menderatu du *deep learning*-ak, oso emaitza onak lortuz hainbat atazetan eta burututako txapelketetan. Txapelketa garrantzitsuena eta ezagunena **ImageNet** da eta honi esker, irudien klasifikazioan egondako aurrerapenak ikusi ahal izan dira. *Deep learning*-aren aurrerapen garrantzitsuenak 2011-2012 urteetatik aurrera gertatu dira, ia urtero aurrerapauso handiak emanez.

2012. urtean, Torontoko Unibertsitatean artikulu bat argitaratu zen, *ImageNet Classification with Deep Convolutional Networks* [9], **ImageNet** txapelketaren errorea $\sim$%50 txikitzea lortu zuena (%26.2-tik %15.3-ra). Artikuluan, *AlexNet* izena hartu zuen sare konboluzionala (*convolutional neural network*) proposatzen zen irudien prozesamendurako. Gaur egun erabiltzen diren sareekin konparatuz, nahiko sinplea da baina, hala ere, erreferentzia puntu bat izan da konboluzio sareak ordenagailu bidezko ikusmenean erabiltzerako orduan.

2014. urtean, VGGNet artikulua argitaratu zen, *Very Deep Convolutional Neural Networks for Large-Scale Image Recognition* [14]. Bertan azaldutako ideia zera da: ez da beharrezkoa inongo trikimailurik emaitza egokiak lortzeko, sakonera handiko sareak erabiltzearekin nahikoa da.

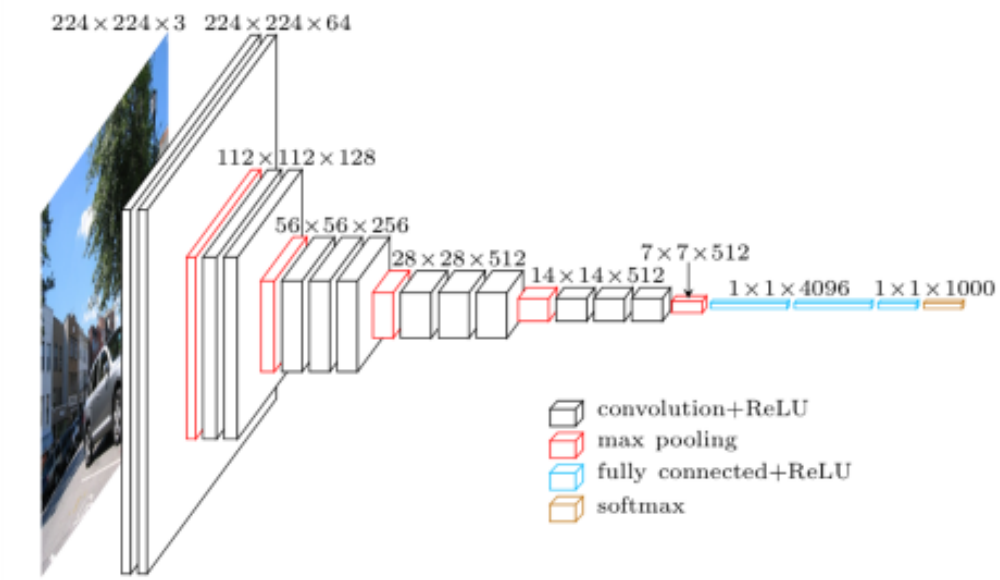
GoogLeNet arkitektura, baliabide konputazionalen arazoa tratatzen lehena izan zen, *Going Deeper with Convolutions* [15] artikuluan. Sailkapen sareak geroz eta sakonagoak egiten zirenez, memoria gehiegi erabiltzen ari zen. GoogLeNet-en aurkeztu zen *inception* moduluaren bitartez, kalkuluak eta memoria kontsumoa murriztu ziren.

ResNets sareek, 2015. urtean *Deep Residual Learning for Image Recognition* [5] artikulua argitaratu zenetik, ordenagailu bidezko ikusmenaren ataza askotan aurrerapen handiak

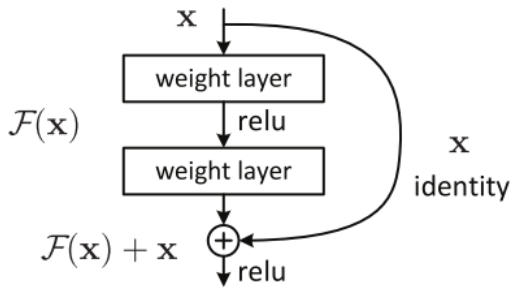
ekarri dituzte. ResNet arkitektura **ImageNet** txapelketan gizakien maila gainditzen lehena izan zen, eta beraien ikaskuntza mota gaur egungo sareetan oso erabilia da.

Hauek izan dira irudien sailkapenean aurrerapenak ekarri dituzten arkitekturak. Esan bezala, aurrerakuntza handiak egon dira eta problema erreal asko ebatzi daitezke ikasketa sakonari esker.

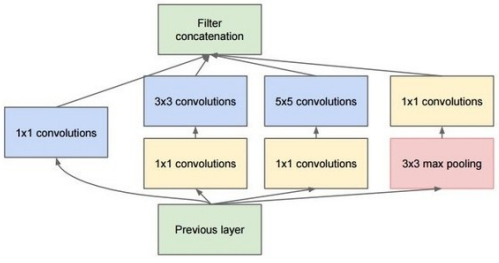
1.1 irudian, aipatutako arkitektura batzuen adibideak ikusi daitezke.



(a) *VGGNet* arkitektura



(b) *ResNet* sareetan erabilitako *residual block*



(c) *GoogLeNet* sareko *Inception* modulua

1.1 Irudia: Arkitektura desberdinen adibideak

2. KAPITULUA

Proiektuaren Helburuen Dokumentua

Kapitulu honetan, proiektuaren kudeaketarekin erlazionatutako atalak aztertuko dira, hala nola, proiektuaren irismena, plangintza, lan metodologia, emangarriak, arriskuen kudeaketa, baliabideen kudeaketa eta interesatuak.

2.1 Irismena

Proiektuaren irismena *deep learning* eredu bat garatzea da, irudien prozesamendurako erabiltzeko. Zehazki, proiektua garatu ondoren edozein irudirekin entrenatzeko ahalmena duen eredu orokorgarri bat edukitzea da helburua, datu-base desberdinetarako erabili ahal izateko.

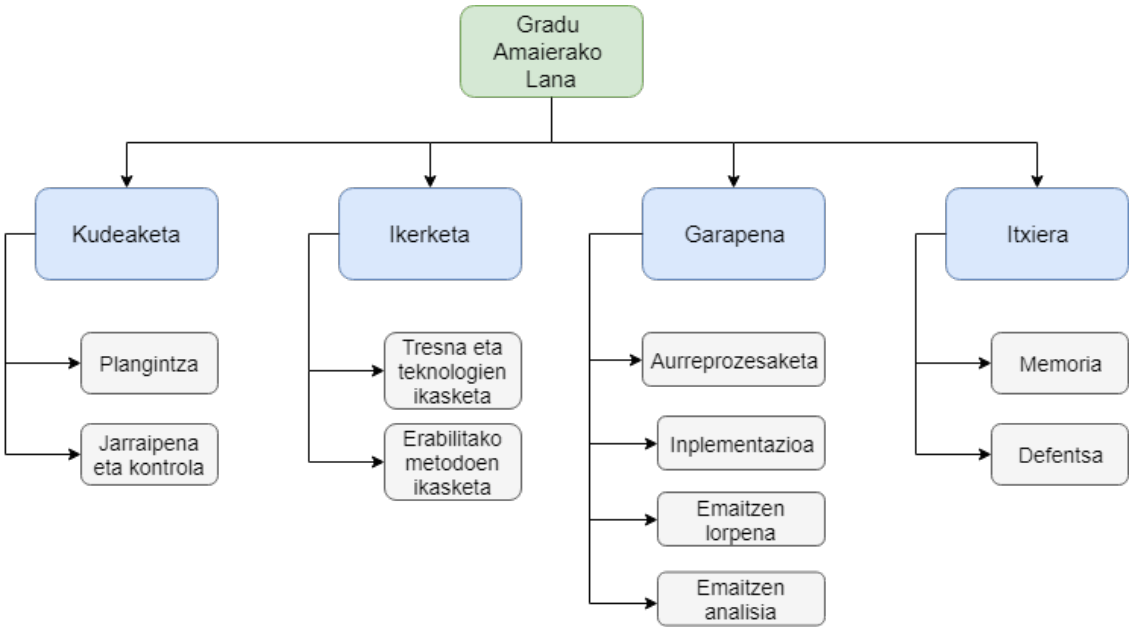
2.2 Plangintza

Atal honetan, proiektuaren plangintza aztertuko da. Alde batetik, garatu beharreko ataza desberdinak azalduko dira eta ondoren, ataza bakoitzari eskaini beharreko denboraren aurreikuspena egingo da.

2.2.1 Atazen identifikazioa

Proiektua lau atal desberdinetan banatzea erabaki da. 2.1 irudian ataza hauek biltzen di-

tuen Lanaren Deskonposaketa Egitura, LDE diagrama hain zuzen, azaltzen da.



2.1 Irudia: LDE diagrama

Jarraian, ataza hauek azalduko dira banan banan:

- Kudeaketa: ataza honen barruan, **plangintza** eta **jarraipena eta kontrola** sartzen dira. **Plangintza** proiektuaren nondik norakoak erabakitzeko balio du, bete beharreko mugak zehaztuko dira garatu beharreko ataza desberdinetarako. **Jarraipen eta kontrola**ren bidez, aldiz, zehaztutako helburuak betetzen diren kontrolatuko da eta horrez gain, adibidez, egon daitezkeen arazoen jarraipena eramango da.

- Ikerketa: atal honetan proiektuaren garapenerako beharrezkoak diren oinarri teorikoak landuko dira. Bi ataza desberdinetan banatu da: **tresna eta teknologien ikasketa** eta **erabilitako metodoen ikasketa**. Lehenengoan, erabiliko den hardware eta softwareari buruzko ezagutzak lortuko dira. Bestetik, bigarren atala, landuko diren *machine learning* eta *deep learning* metodoak aztertu eta ulertzeko pentsatu da.

- Garapena: izenak adierazten duen bezala, ataza hau proiektuaren garapenari dagokio. Honen barruan lau ataza desberdin sartzea erabaki da. Alde batetik, **aurreprozesaketa** dago. Honetan, erabiliko diren irudiak aukeratu eta hauei egin beharreko

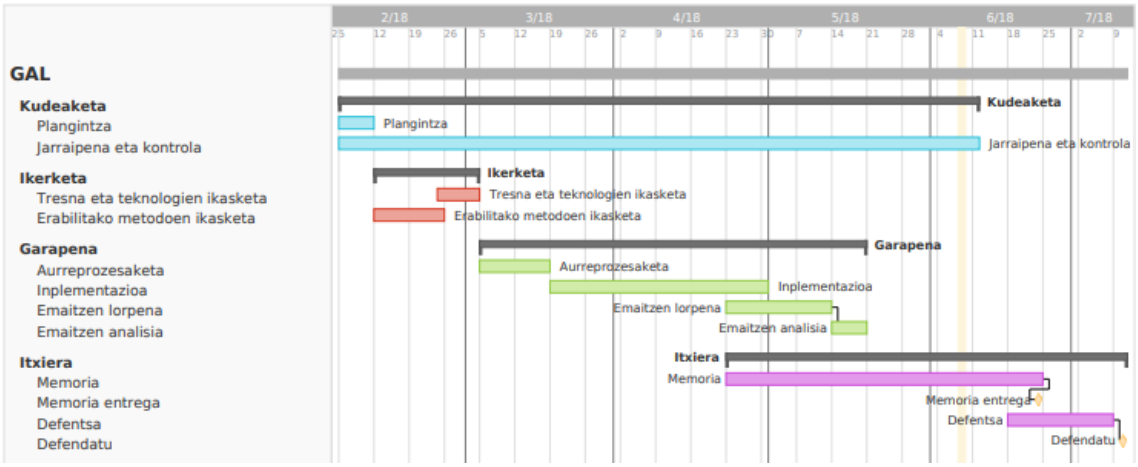
transformazio eta aldaketa guztiak egingo dira. Beste alde batetik, **inplementazioa**n sortu beharreko programak eta ereduak garatuko dira. Horrez gain, **emaitzen lorpena** atazaren barruan egin beharreko proba guztiak egin eta emaitzak gordeko dira. Bukatzeko, **emaitzen analisia**n lortutako emaitzak aztertu eta proiektuaren ondorioak aterako dira.

- Itxiera: azkeneko atal hau, proeiktuaren entrega eta aurkezpenerako bideratuta dago. Beraz, honen barruan idatzi beharreko **memoria** eta egin beharreko **defentsa**ren prestakuntza sartzen dira.

2.2.2 Planifikatutako egutegia (Gantt diagrama)

Proiektuak hartuko duen denboraren ideia egokiagoa izateko, 2.2 irudiko Gantt diagrama garatu da. Bertan, ataza bakoitzak egutegian hartuko duen denbora agertzen da.

Ikusi daitekeenez, kudeaketa proiektu osoan zehar landuko da eta gainontzekoek, ordena kronologiko bat jarraituko dute. Lehenengo landuko dena ikerketa izango da, ondoren garapena eta azkenik, itxiera.



2.2 Irudia: Gantt diagrama

Horrez gain, proiekturako 300 ordu daudela kontuan izanik, ordu hauek adierazitako atazei esleitu zaie 2.1 taulan agertzen den modura. Betiere esleipen hau proeiktua aurrera doan heinean aldatu daitekeela kontuan izanik, gertatu daitezkeen desbideraketak direla medio.

ATAZA	*ORDU KOPURUA*
Kudeaketa	**40 ordu**
Plangintza	20 ordu
Jarraipen eta kontrola	20 ordu
Ikerketa	**50 ordu**
Tresna eta teknologien ikasketa	20 ordu
Erabilitako metodoen ikasketa	30 ordu
Garapena	**135 ordu**
Aurreprozesaketa	35 ordu
Inplementazioa	60 ordu
Emaitzen lorpena	25 ordu
Emaitzen analisia	15 ordu
Itxiera	**75 ordu**
Memoria	60 ordu
Defentsa	15 ordu
GUZTIRA	*300 ordu*

2.1 Taula: Ataza bakoitzari esleitutako ordu kopurua

2.3 Lan metodologia

Proeiktuaren arrakasta ziurtatzeko lan metodologia bat zehaztu da. Proiektua bakarrik garatuko denez, hau da, ez denez talde baten menpe egongo, nahiko erraza izango da honen jarraipena eramatea. Hala ere, zuzendariekin bilerak maiz egingo dira, garatutakoari buruz eta jarraian egin beharreko atazei buruz hitz egiteko.

Horrez gain, egunero fakultateko RSAIT taldeko bulegora lan egitera joatea erabaki da, horrela egunero lau orduz proeiktuan lan egiten dela ziurtatuko da. Egun bakoitzean zer egiten den jakiteko, eguneroko bat eramango da *Google Drive* plataforman.

Memoria *Overleaf*[1]-en idatziko da, LaTeX online zerbitzua eskaintzen duena. Tresna honek edozein ordenagailutik idaztea eta aldaketak modu argian ikustea ahalbidetuko du.

[1] https://www.overleaf.com/

2.4 Emangarriak

Emangarriei dagokienez, proiektuan hainbat sortuko dira eta 2.2.2 ataleko Gantt diagramaren arabera bukaera data desberdinak izango dituzte. Hauek dira sortuko diren emangarriak:

- Kodea: garatuko diren programa desberdinak biltzen dituen kode bilduma. Bukaera data apirilaren 30ean ezarri da.
- Memoria: proiektuaren nondik norakoak, helburuak, garapena, emaitza eta ondorioak biltzen dituen txosten idatzia. Memoriaren entrega ekainak 24rako ezarrita dagoenez, ekainak 15erako bukatuta egotea espero da, denbora-tarte hori gertatu daitekeen edozein ezustetarako utziz.
- Aurkezpena: defentsaren egunean erabiliko den aurkezpena. Defentsa uztailak 10-13 egunetan izango denez, aurkezpena uztailaren 8rako bukatu nahi da.

2.5 Arriskuen kudeaketa

Proeiktuan zehar gertatu daitezkeen arriskuak aztertzea ezinbestekoa da hauek saihestu ahal izateko eta gertatzekotan nola jokatuko den erabakitzeko. Jarraian, gertatu daitezkeen arazoak eta hauen aurrean hartuko diren neurriak azalduko dira:

- Informazio galera: sistema informatikoan gertatu daitezkeen arazoak direla eta, informazio galerak egon daitezke. Hauek ekiditeko, garatutako kodea mahaigaineko ordenagailuan gordetzeaz gain, *GitHub* plataformara igotzea erabaki da. Gainera, memoria garatzerako orduan sortuko diren dokumentu guztiak *Overleaf*-en gordetzeaz gain, *Google Drive* zein ordenagailuan lokalean gordeko dira.
- Arazo teknikoak: erabiliko diren software eta hardware tresnetan gertatu daitezkeen arazoak dira. Hauek ekiditeko, tresna guztiak modu egokian erabili eta gordeko dira. Arazoren bat gertatuz gero, konponbidea bilatzen saiatuko da eta bestela, RSAIT taldeko kideei laguntza eskatuko zaie.
- Komunikazio falta: proiektua modu egokian garatzeko, zuzendariekin komunikazioa ezinbestekoa da. Hortaz, bilerak maiz hitzartuko dira aurrerapen eta arazo guztiei buruz hitz egiteko.

- Plangintzarekin bateraezintasuna: garatutako plangintza jarraitzea ezinezkoa denean gertatzen diren atzerapen arazoak dira hauek. Plangintzan ataza bakoitzari zehaztutako denbora eskasa bada, ataza garrantzitsuenei lehentasuna emango zaie. Hala ere, desbiderapenak saihesteko, planifikatutakoa betetzen saiatuko da.
- Osasun arazoak: egon daitezkeen gaixotasunek sortutako arazoak dira. Ekidin ezin diren arazoak direnez, planifikatutakoa modu egokian bete behar da, gaixotasunen bat egonez gero, ahalik eta denbora gutxien galtzeko.

2.6 Baliabideen kudeaketa

Proiektuaren garapenean hainbat baliabide desberdin erabiliko dira. Memoria idazterako orduan LaTeX erabiliko da, *Overleaf* tresnarekin batera. Gainera, egindakoaren egunerokoa idazteko eta memoriaren bertsioak zein proiektuaren emaitzen taulak gordetzeko *Google Drive* erabiliko da. Aldiz, kodea *GitHub* plataforman gordeko da.

Proiektuaren garapenean erabiliko diren baliabide nagusiak honakoak dira: *Keras, TensorFlow, Caffe, DIGITS, Python, scikit-learn, numpy* eta *OpenCV* softwareari dagokionez eta *JetsonTX1* hardwareari dagokionez. Bukatzeko, erabiliko den sistema eragilea **Ubuntu 16.04 LTS** izango da. Beharrezko material guztia RSAIT ikerketa taldearen bulegotik lortuko da.

2.7 Interesatuak

Proiektuaren interesatuetako bat proiektuaren egilea bera da, proiektua arrakastaz bukatzeko ardura naguesiena du eta.

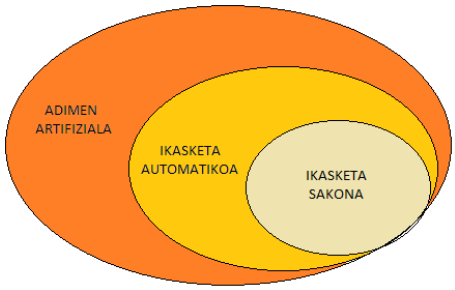
Bestetik, proiektuaren zuzendariak, Basilio Sierra eta José María Martínez-Otzeta, daude. Hauek ikasleari languntza eskaini eta proiektuan zehar gidatu egingo diote.

Bukatzeko, proiektuaren epaimahaia egongo da. Epaimahaiako kideek proiektuaren aurkezpena ikusi eta egindakoaren arabera proiektuaren ebaluazioa egingo dute.

3. KAPITULUA

Oinarri teorikoak

Hasteko, *deep learning*-a edo ikasketa sakona zer den ondo ulertu ahal izateko, *artificial intelligence, machine learning* eta *deep learning* kontzeptuen azalpena beharrezkoa da, irudikapen grafikoa 3.1 irudian.



3.1 Irudia: Oinarrizko kontzeptuen irudikapen grafikoa

- Adimen artifiziala (*artificial intelligence*): gizakiek ebatz ditzaketen problema konplexuak konputagailu baten bidez ebaztean datza adimen artifizialaren oinarrizko ideia. Hau da, gizakien prozesu arrazional eta deduktiboaren imitazioan oinarritzen den zientzia da. Orotariko problemak daude.

- Ikasketa automatikoa (*machine learning*): esperientziatik ikasten duen adimen artifizial modu bat da. Ikasketa automatikoko sistemek datu mordo batekin lan egiten dute, patroi jakin batzuk identifikatzen dituzte eta horiei esker, etorkizuneko portaerak iragarri ditzakete. Ikasketa automatikoa, algoritmoen bitartez gauzatzen da. Bere helburua, ataza jakin bat betetzeko eredu bat sortzea da eta ondoren, eredua

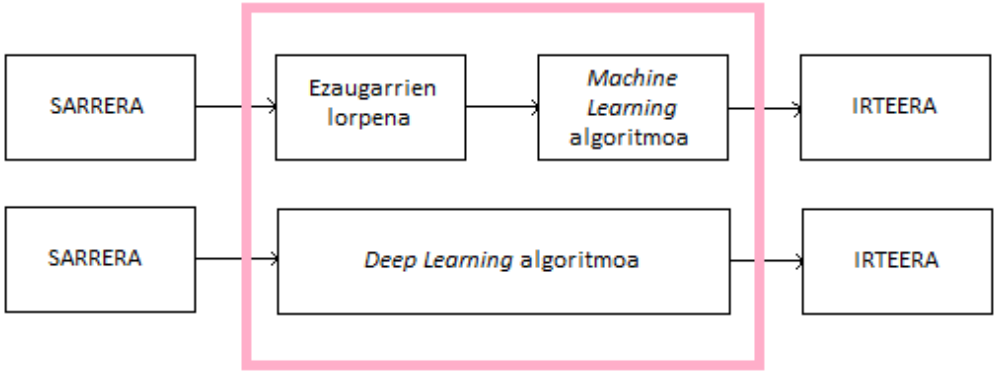
entrenatu egiten da datu asko erabiliz. Beraz, ereduak datu hauen bidez ikasi eta etorkizunean datu berri bat iristerakoan, iragarpen bat emateko gaitasuna izango du. Algoritmoaren aukeraketa ebatzi beharreko atazaren araberakoa izango da eta lortutako eredua algoritmo horren menpekoa izango da. Aurrerago, algoritmo hauen pare bat adibide azalduko dira.

- Ikasketa sakona (*deep learning*): ikasketa sakona, ikasketa automatikoko metodo bat da, modu automatikoan eta datuetatik abiatuz ikasteko diseinatutakoa. Ikasketa automatiko eta sakonaren artean dagoen desberdintasunik handiena honakoa da:

  Bata zein bestean bi fase nagusi daude, entrenamendua edo ikasketa fasea eta iragarpen fasea. *Machine learning* erabiliz egindako klasifikazioetan, entrenatzeko atalean bi fase garrantzitsu daude:

  1. Ezaugarrien lorpena: eredua entrenatzeko baliagarriak izango diren ezaugarrien lorpena.
  2. Ereduaren entrenamendua: lortutako ezaugarriekin eta kasu bakoitzaren klasearen bidez eredua entrenatzen da.

  Beraz, *machine learning* algoritmoetan, ezaugarriak eskuz lortu behar dira. Aldiz, *deep learning* algoritmoak erabiltzerakoan, ezaugarrien lorpena automatikoki egiten du algoritmoak. 3.2 irudian, desberdintasun hau azaltzen duen eskema agertzen da. Ezaugarriak automatikoki lortzea abantaila handia da, ezaugarrien lorpena eta aukeraketa prozesu luze eta zaila da eta.



3.2 Irudia: *Machine learning* VS *deep learning*

Jarraian, proiektuan landu diren kontzeptuak zehatzago azalduko dira.

3.1 Machine learning

Esan bezala, esperientziatik ikasten duten algoritmoak *machine learning*-aren barruan sartzen dira. Ikasketa automatikoa bi modukoa izan daiteke: gainbegiratuta edo gainbegiragabea. Ikasketa gainbegiratuan, entrenatzeko erabiltzen diren kasuak etiketatuak daude, hau da, kasu bakoitza bere klasearekin lotuta dago. Aldiz, ikasketa gainbegiragabean, ez dago inongo klaserik lotuta kasuekin. Proiektu honetan, ikasketa gainbegiratuarekin lan egin da, erabilitako irudi bakoitzak bere etiketa du eta.

Sailkatzaile batek, klaserik gabeko aldagai berri bat datorkigunean, aldagai horrentzat klase ezagun baten iragarpena egiteko, hau da, aldagai berri horri ezaguna den klase bat esleitzeko balio du. Sailkatzaile hau lortzeko hainbat algoritmo desberdin daude. Algoritmo hauek, aurretik ezagututako datuen bidez lortutako informazioa erabiltzen dute sailkapena egiteko. Ezaguna den informazioa, salkatzaileak ikasteko erabiliko duena hain zuzen, entrenamendurako datu-basea izango da. Beraz, algoritmo hauek, informazio ezagunetik abiatuta, kasu berriak klasifikatzeko erregela bat definitzen saiatzen dira.

Sailkatzaile mota desberdinak daude, hala nola, *K-NN*, sailkapen-zuhaitzak (*decision trees*), *support vector machines (SVM)* eta sare neuronalak (*neural networks*). Proiektuan *K-NN* eta *decision tree* sailkatzaileekin lan egin denez, hauek azalduko dira jarraian:

3.1.1 K-NN

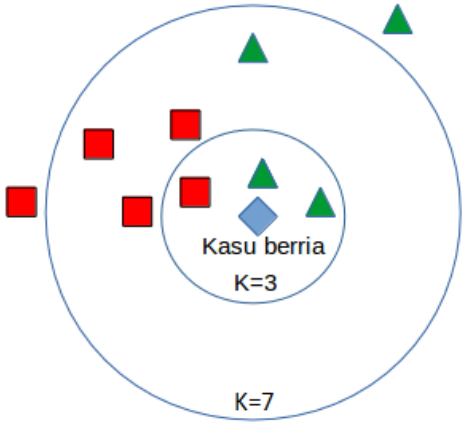
K-NN algoritmoa (*K-Nearest Neighbors*), datorkigun kasu berriarekiko **K** auzokide hurbilenak hartzean datza. Lortutako auzokide hurbil horien artean gehien errepikatzen den klasea da kasu berriari esleituko zaiona. Proiektu honetan, gordetako kasuen eta kasu berriaren arteko distantzia kalkulatzeko, distantzia euklidestarra erabiltzen da, nahiz eta distantziak kalkulatzeko beste aukera asko egon:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{p} (x_{ri} - x_{rj})^2}$$

p indizea instantzia baten irudikapena da, bere **p** ezaugarriek adierazitako **p**-dimentsioko espazioan kokatuta.

1-NN-ren kasuan auzokide hurbilenaren klasea iragarriko da, 3-NN-k aldiz 3 auzokide

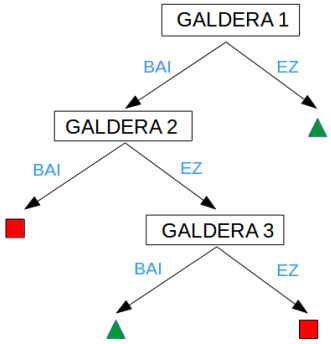
hurbilenak hartuko ditu kontuan, etab. K-NN sailkatzailea lortzeko, aurrerago azaltzen den ***scikit-learn*** liburutegia erabili da. Algoritmoa hobeto ulertzeko, 3.3 irudian honen adibide grafikoa ikusi daiteke. $K = 3$ kasuan, ▲ klasea iragarriko da eta aldiz, $K = 7$ denean, ■ iragarriko da.



3.3 Irudia: *K-NN* algoritmoaren adibide grafikoa

3.1.2 Decision Tree

Sailkapen-zuhaitza (*decision tree*) parametrorik gabeko ikasketa metodo gainbegiratua da, klasifikaziorako eta erregresiorako erabilia. Helburua, sarrerako datuetatik ateratako ezaugarriekin erabaki erregela batzuk sortzea da, kasu berri bat iristerakoan erregela horien bidez etiketatu ahal izateko. 3.4 irudian sailkapen-zuhaitza baten adibidea ikusi daiteke.



3.4 Irudia: Sailkapen-zuhaitza baten adibide grafikoa

Sailkapen-zuhaitza sortzeko, oraingoan ere ***scikit-learn*** liburutegia erabili da. Honek,

CART ***(Classification and regression trees)*** [2] algoritmoaren hobekuntza bat erabiltzen du zuhaitza sortzeko.

Hobekuntzarik lortu ezin daitekeela ikusten denean edo amaiera erregelaren bat zehazten denean bukatzen da zuhaitzaren eraketa prozesua. Hala ere, aldizka, egin daitezkeen banaketa guztiak egin, eta ondoren zuhaitza inausten da.

Zuhaitzaren adar bakoitza muturreko nodo batean bukatzen da. Behaketa bakoitza muturreko nodo batera eta bakarrik batera iristen da. Gainera, muturreko nodo bakoitza, hau da, hosto bakoitza, erregela multzo paregabe baten arabera zehaztu da.

3.2 Deep learning

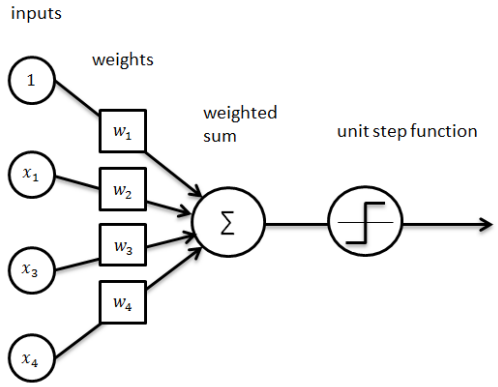
Deep learning, ordenagailuek esperientziatik ikastea eta mundua modu hierarkikoan ikustea ahalbidetzen duen ikasketa automatiko mota bat da. Adimena esperientziatik lortzen denez, ez da behar inongo gizakirik ordenagailuak behar duen ezagutza guztia zehazteko. Hierarkia honek, kontzeptu sinpleetatik kontzeptu konplikatuagoak ikastea ahalbidetzen du; hierarkia hauen grafo bat hainbat geruzetako sakonera izan dezake, hortik ikasketa sakona (*deep learning*) izena.

Hasteko, aipatu beharra dago *deep leaning*-ak normalean sare neuronalak erabiltzen dituela. Hauek, historia luzea dute atzetik eta hau pixka bat ulertzeko, laburki *perceptron* eta *multilayer perceptron (MLP)* sare neuronalak azalduko dira.

- *Perceptron* [11]: sarrerako geruza bat eta irteerako neurona bat dituen sare neuronal sinpleena da (3.5 irudia). Instantzia bakoitza hiperplano baten alde batean edo bestean kokatzen den arabera klasifikazio bitarrak egiteko erabiltzen da. Beraz, sare neuronal hau nahikoa mugatua da, bakarrik linealki banangarriak diren funtzioentzako balio duelako. Pisuen eguneraketa honako formularen bidez gauzatzen da:

$$w(j)' = w(j) + \alpha(\beta - y)x(j),$$

  non **w(j)** pisuen bektoreko **j** posizioko balioa den, $\boldsymbol{\alpha}$ $0<\alpha<1$ balio arteko konstantea den, $\boldsymbol{\beta}$ espero den irteera den, **y** neuronaren irteera den eta **x(j)** sarrera bektoreko **j** posizioko balioa den.



3.5 Irudia: *Perceptron* neurona-sarea

- *Multilayer perceptron:* gutxienez ezkutuko geruza bat duen eta aktibazio funtzio ezlinealak erabiltzen dituen sare neuronala da. Modu bitarrean banandu ezin daitezkeen datuak bereizi ditzake. Honek jada *backpropagation* [13] metodoa erabiltzen du ikasketa prozesuan. Metodo hau gradientearen jaitsieraren optimizazio algoritmoek erabiltzen dute normalean, neuronen pisuak eguneratzeko galera funtzioaren gradientea kalkulatuz. Hau da, irteeran errorea kalkulatu eta atzera bidaltzen da sarearen ezkutuko geruzen neuronetan zehar. Hala ere, ezkutuko geruzaren neuronek errore seinaleko zati bat baino ez dute jasotzen, neurona bakoitzak jatorrizko irteeran izandako ekarpen erlatiboaren araberakoa dena. Prozesu hau errepikatuz doa, geruzaz geruza, sareko neurona guztiek errore totalaren bere ekarpen erlatiboaren araberako errore seinalea jaso arte.

Gaur egun, algoritmoen hobekuntzek zein kalkulu-ahalmen eta datuen bilketa masiboen hobekuntzek, paradigma aldatzea ahalbidetu dute. Jarraian, proiektuan erabilitako sare neuronal mota azalduko da.

Proiektuan, konputagailu bidezko ikusmenerako, *convolutional neural network* neurona-sare konboluzionalak erabili dira. Neurona-sarea entzuterakoan, neurozientzia edota biologiarekin erlazionatu daiteke. Kortex bisual biologikoaren inspirazioa hartu dute sare neuronal konboluzionalek. Kortex bisualak, ikusmen eremuko eskualde zehatzetara sentikorrak diren zelula eremu txikiak ditu. Burmuineko zelula neuronal bakar batzuk orientazio jakin bateko ertzetara erantzuten dutela frogatu zuten **Hubel** eta **Wiesel**-ek 1962an [7]. Adibidez, neurona batzuk ertz horizontalak antzematerakoan aktibatzen ziren eta beste batzuk aldiz, bertikalak edo diagonalak antzematerakoan. **Hubel** eta **Wisel**-ek ikusi zu-

ten neurona guzti hauek zutabe arkitektura moduan antolatuta zeudela eta guztiek batera, pertzeptzio bisuala lortu zezaketela. Sistema baten barruan ataza jakin bat betetzen espezializatutako osgaiak edukitzea (adibidez, kortex-bisualeko neuronak ezaugarri zehatzak bilatzen) makinek jarraitzen duten ideia bat da, eta sare konboluzionalen oinarrizko ideia da.

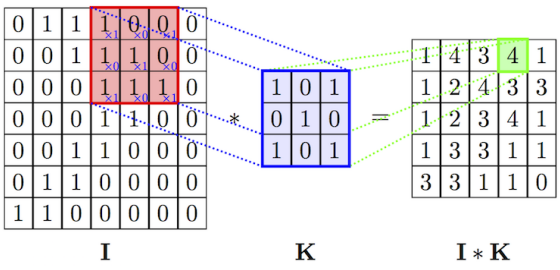
Sare konboluzionalak, irudien azterketa eta klasifikaziorako egokiak suertatu diren neurona-sare artifizialak dira. Bere aplikazioa bi dimentsioetako matrizeetan burutzen denez, konputagailu bidezko ikusmenean oso eraginkorrak dira.

3.2.1 Geruza motak

Convolutional neural network sareetan, sarrera eta irteera geruzez gain, hainbat ezkutuko geruza daude. Ezkutuko geruza hauek normalean *convolutional, pooling* eta *fully connected* geruzak dira. Jarraian, geruza mota hauek banan-banan aztertzen dira, sarea nola funtzionatzen duen hobeto ulertzeko.

- *Convolutional layer*:
  Geruza konboluzional bat sarrerari konboluzio bat aplikatzean datza, emaitza hurrengo geruzara pasaz.

  3.6 irudia adibide moduan hartuz, konboluzio prozesua azalduko da jarraian:



3.6 Irudia: Konboluzio baten adibide grafikoa

  K matrize txikia, **I** sarrera matrizean zehar mugitzen joaten da eta posizio bakoitzerako biderketa gauzatzen da. Ez da matrizeen arteko biderketa, **K**-ko posizio bakoitza dagokion **I**-ren posizioko balioarekin biderkatzen da. Ondoren, $\mathbf{I} * \mathbf{K}$ irteeraren posizio bakoitzeko elementua lortzeko, posizio horretarako matrize txikiarekin lortutako biderketen batura egiten da. Irudiko adibidean: $1 \cdot 1 + 0 \cdot 0 + 0 \cdot 1 + 1 \cdot 0 + 1 \cdot$

$1+0\cdot 0+1\cdot 1+1\cdot 0+1\cdot 1=4$. **K** irudiko matrize txikiak *filter* edo *kernel* izena hartzen du eta sarrera matrizearen eta *filter*-aren arteko konboluzioaren bitartez lortzen den irteera matrizeak, aldiz, *Feature Map* izena hartzen du.

Filtroak sarrera originaletik ezaugarri desberdinak detektatzeko erabiltzen dira. Filtro matrizearen balioak aldatuz, *Feature Map* desberdinak lortzen dira sarrera berdinarentzako.

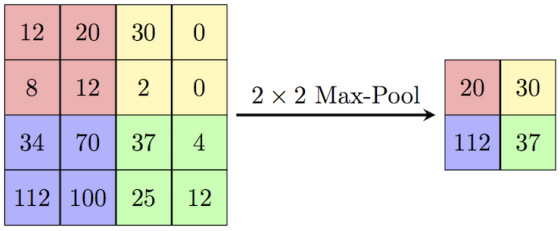
Geruza konboluzionaletan, ezaugarri asko zehaztu daitezke, adibidez:

- *Depth*: konboluzio eragiketan erabiliko den filtro kopurua.
- *Stride*: filtro matrizea sarreratik irristatzen den pixel kopurua.

- *Pooling layer*:
  Geruza hauek, neurona talde baten irteerak neurona bakar batean batzen dituzte.

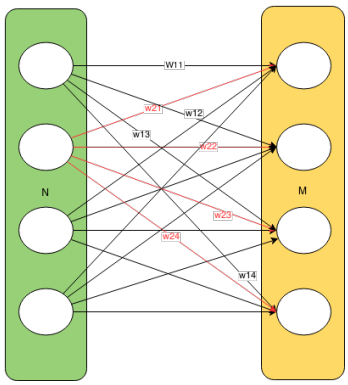
  Ohikoa da, aldizka, *pooling* geruza bat jartzea *convolutional* geruzen artean sare konboluzional batean. Bere funtzioa errepresentazioaren tamaina pixkanaka txikiagotzea da, parametro kopurua eta sarearen karga konputazionala murrizteko. Horrez gain, ondoren azalduko den *overfitting*-a murrizteko ere balio du.

  Pooling geruza mota desberdinak daude, baina proiektuan bakarrik *max-pooling* geruzak erabili dira. Hauek, neurona talde bakoitzetik balio maximoa hartzen dute, 3.7 irudian ikusi daitekeen moduan.



3.7 Irudia: *Max-pooling* baten adibide grafikoa

- *Fully connected/dense layer*:
  Hauek, geruza bateko neurona guztiak hurrengo geruzako neurona guztiekin lotzen dituzte, *multilayer perceptron* izena hartzen duen sare neuronalaren parekoa dela esan daiteke. 3.8 irudian, geruza honen adibidea ikusi daiteke.



3.8 Irudia: *Dense* geruza baten adibide grafikoa

3.2.2 Optimizatzaileak

Sare neuronalen beste atal garrantzitsu bat optimizazio algoritmoak dira. Hauek, errore funtzio bat minimizatzen (edo maximizatzen) dute. Kasu honetan, algoritmo honek sarearen galera funtzioa (*loss function*) minimizatuko du. Optimizazio algoritmoak, irteera kalkulatzeko erabiltzen diren barne-parametroak ikasten ditu eta hauek iteratiboki eguneratzen ditu entrenamenduko datuak erabiliz, soluzio optimoaren norabidean.

Optimizazio algoritmo ugari egon arren, pare bat erabili dira. Hauek izan dira proiektuan zehar erabili diren optimizazio algoritmoak:

- Stochastic Gradient Descent (SGD):
  Optimizatzaile mota bat *gradient descent* izenekoa da [12], gradientearen jaitsiera hain zuzen. Honek, parametroak eguneratzen ditu funtzioaren gradientearen kontrako aldera. Aldagai anitzeko funtzio batean, bere deribatu partzialek osaturiko bektorea da gradientea. Honek puntu jakin batean funtzioaren hazkunde handieneko norabidea azaltzen du. Beraz, minimora gerturatzen joango da. Gradientearen jaitsieraren hiru aldaera daude, funtzioaren gradientea kalkulatzeko erabiliko den datu kopuruan desberdintzen direnak.

  1. *Batch gradient descent:* entrenamendurako datu-base osoa erabiltzen du gradientea kalkulatzerakoan eta hortaz, parametroak eguneratzerakoan.

     $\theta = \theta - \eta \cdot \bigtriangledown_\theta J(\theta)$

     non θ eguneratuko diren parametroak, η ikaskuntza-tasa (*learning rate*), $J(\theta)$ optimizatuko den funtzioa eta $\bigtriangledown J(\theta)$ funtzioaren gradientea diren.

2. *Stochastic gradient descent:* entrenamenduko kasu bakoitzeko ($x(i)$, $y(i)$ klasearekin), gradientea kalkulatu eta parametroak eguneratzen ditu.

   $$\theta = \theta - \eta \cdot \bigtriangledown_\theta J(\theta; x^{(i)}; y^{(i)})$$

3. *Mini-batch gradient descent:* n entrenamenduko kasu erabiltzen ditu parametroen eguneraketa egiteko.

   $$\theta = \theta - \eta \cdot \bigtriangledown_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$$

Proiektuan erabilitako metodoa *stochastic gradient descent* izan da. Normalean, besteak baino azkarragoa da, kasu bakoitzeko eguneraketa egiten duenez erredundantea ez delako. Honek, aldaketa nabarmenak egiten ditu konbergitzeko arriskuak egonez, baina, hala ere, ikaskuntza-tasa gutxinaka jaitsiz (3.2.3 atala) besteen konbergentzia ahalmen bera lortu dezakeela ikusi da.

- Adam:
  Torontoko unibertsitatean 2015. urtean aurkeztu zen algoritmoa da [8]. Adam izena *adaptive moment estimation*-etik dator. *Stochastic gradient descent* metodoaren bi hedapen desberdinen abantailak konbinatzen ditu. Zehazki:

  1. *Adaptive Gradient Algorithm (AdaGrad)* [3]: parametro bidez ikaskuntza-tasa bat mantentzen du, gradiente sakabanatuko problemen (adibidez, konputagailu bidezko ikusmeneko problemak) etekina hobetzen duena. AdaGrad algoritmoak bere eguneraketa erregelan ikaskuntza-tasa orokorra aldatzen du t denbora tarte bakoitzean θ_i parametro bakoitzeko, θ_i-rentzako kalkulatu diren gradienteen arabera. $g_{t,i}$ erabiliko da $\bigtriangledown_\theta J(\theta_{t,i})$ adierazpena laburtzeko.

     $$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}} \cdot g_{t,i}$$

     G_t matrize diagonal bat da, non diagonaleko i,i elementu bakoitza gradienteen karratuen batura den. ε aldiz, zerorekin zatitzea saihesteko erabiliko da. θ parametro guztien aurreko gradienteen karratuen batura G_t matrizearen diagonalean gordeta daudenez, bektoreekin lan egin daiteke:

     $$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} \odot g_t$$

2. *Root Mean Square Propagation (RMSProp)*: Geoff Hinton-ek proposatutako argitaratu gabeko metodoa da. Parametro bidezko ikaskuntza-tasak mantentzen ditu ere, gradienteen azkeneko magnitudeen batezbestekoan oinarrituz moldatzen direnak.

   $$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$

   $E[g^2]_t$ gradienteen batezbestekoa izanik, horrela gauzatuko da parametroen eguneraketa:

   $$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}} g_t$$

   Hinton-ek γ-ri 0.9 eta η-ri 0.001 balioak ezartzea gomendatzen du.

Ikaskuntza-tasa banaketaren lehenengo momentuan (batezbestekoa) oinarrituz aldatu beharrean (RMSProp moduan), Adam optimizatzaileak ere gradienteen bigarren momentua erabiltzen du (bariantza). Zehazki, algoritmoak gradientearen eta gradiente karratuko batezbesteko mugikor esponentzial bat kalkulatzen du.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

m_t eta v_t lehenengo momentuko eta bigarren momentuko gradienteen balioespenak dira hurrenez hurren. Biak zeroz osatutako bektore moduan hasieratzen direnez, lehenengo pausuetan zerorantz joko dute, batez ere jaitsiera-tasak txikiak direnean (β_1 eta β_2 bat baliotik gertu daudenean). Joera hori ekiditeko gradienteen momentuak horrela kalkulatuko dira:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

Jarraian, hauek erabiliko dira parametroen eguneraketa gauzatzeko, RMSProp metodoan ikusi den bezala:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t$$

Egileek balio hauek proposatu zituzten: $\beta_1 = 0.9, \beta_2 = 0.999$ eta $\varepsilon = 10^{-8}$.

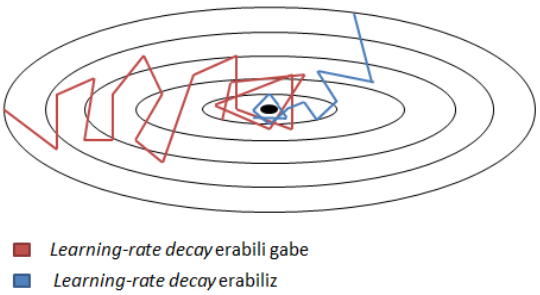
3.2.3 Hiperparametroak

Sarearen egitura eta sarea nola entrenatzen den erabakitzen duten aldagaiak zehazten dituzten parametroak dira hauek. Hiperparametroak entrenamendua baino lehen zehaztu behar dira.

Jarraian, hainbat hiperparametro azalduko dira, sarearen funtzionamendua hobeto ulertu ahal izateko:

- *Learning-rate*:
  Ikaskuntza-tasak (*learning-rate*) sareak bere parametroak zer azkartasunarekin eguneratzen dituen zehazten du. Ikaskuntza-tasa baxu batekin, ikasteko prozesua moteltzen da baina leunki konbergitzen du. Aldiz, ikaskuntza-tasa altu batekin, ikaste prozesua arintzen da baina baliteke inoiz ez konbergitzea. Bien arteko balio egoki bat aurkitu beharra dago.

  Normalean, ikaskuntza-tasaren beherakada (*learning-rate decay*) erabiltzea komeni da. 3.9 irudian, honen adibide grafikoa ikusi daiteke.



3.9 Irudia: *Learning-rate decay*

  Kasu honetan, ikaskuntza-tasaren balioa txikitzen joaten da denboran zehar. Pausuak ematen doan heinean, *learning-rate*-a txikitzen doa *decay* erabiltzen badugu. Modu horretan, puntu optimotik gertuago gelditzea lortuko du, besteak baino pausu txikiagoak ematen dituelako.

- *Activation functions*:
  Nodo baten aktibazio funtzioa (*activation function*), sarrera bat emanez, nodo ho-

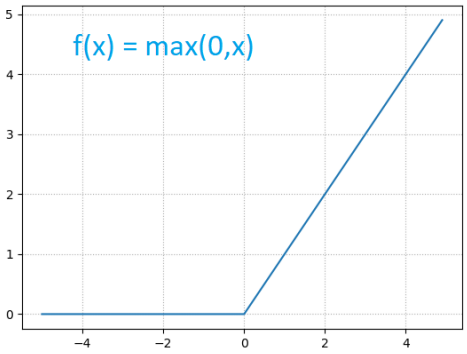
rren irteera zehazten du. Modu sinplean azalduz, neurona bat aktibatzen den edo ez erabakitzen du. Aktibazio funtzio linealak eta ez-linealak daude. Linealetan, sarrera irteerara pasatzen da inongo aldaketarik jasan gabe. Aldiz, funtzio ez-linealetan ez da horrela eta linealki bereizi ezin daitezkeen datuak banatzeko erabiltzen dira.

Hortaz, funtzio ez-linealak erabiliko dira, funtzio konplexuak inplementatu ditzaketelako. Gainera, ez-linealtasunik gabe, sare-neuronaleko hainbat geruza edukitzea bakar bat edukitzearen parekoa izango litzateke. Kasu honetan, optimizatzaileen antzera, funtzio ugari daude aukeratzeko. Proiektuan honako hauek erabiltzea erabaki da:

- ReLU [4]:
  Horrela definitzen da:

  $$f(x) = \max(0,x), \text{ non } \mathbf{x} \text{ neuronaren sarrera den.}$$

  2000. urtean deskribatu zen lehen aldiz, bere oinarri matematiko eta biologikoak azalduz. Neurona-sare konboluzionaletan erabiltzen da, funtzio logistikoekin baino errore txikiagoa lortuz. 3.10 irudian funtzioaren grafikoa ikusi daiteke.



3.10 Irudia: *ReLU* aktibazio funtzioa

- Softmax [1]:
  Azkeneko geruzan erabiltzen da multi-klase iragarpenetan. Unitate bakoitzaren irteera 0 eta 1 arteko balio batera mugatzen du. Gainera, irteera guztien batura 1 izatera behartzen du. Hortaz, klase bakoitzarentzako hori izateko probabilitatea ematen du *softmax* funtzioak.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

non **z** irteera geruzako sarrera bektorea den eta **K** klase kopurua.

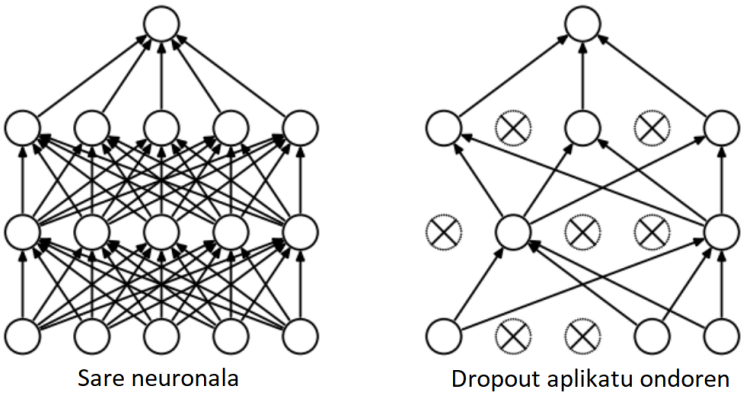
- *Number of epochs*:
  Epoka kopuruak entrenamenduko data osoa sareari zenbat aldiz erakusten zaion zehazten du. Normalean, baliozkotzeko datuekin lortutako emaitzak jaisten hasterakoan gelditu behar da entrenamendua, nahiz eta entrenamenduko datuekin emaitzak hobetzen jarraitu (*overfitting*).

3.2.4 Regularization

Sare neuronal batekin lan egiterakoan, entrenamenduko datuetara gehiegi moldatzen bada, orokortzeko gaitasuna galdu egiten du sareak. Hau da, kasu berri bat pasatzerakoan ez du predikzio egokia emango entrenamenduko datuetara gehiegi egokitu delako eta hortaz, kasu berrietara orokortzeko gaitasuna galdu duelako. *Overfitting* izeneko arazo hau saihesteko, hainbat metodo desberdin daude.

- *Early stopping:*
  Sortutako ereduak hobetzen ez duela ikusterakoan, entrenamendua gelditzean datza. Hau da, baliteke uneren batean emaitzak gehiago ez hobetzea. Momentu horretan entrenamendua bertan behera utziz denbora aurreztuko litzateke. Gainera, agian, emaitzak okertzen doaz baliozkotzeko datuekin (3.2.3 atalean aipatutako moduan) eta hobetzen jarraitu entrenamenduko datuekin. Kasu horretan, entrenamenduko datuetara gehiegi egokitzen ari da eta entrenamenduko prozesua goiz gelditzen *overfitting* arazoa saihestuko da.

- *Dropout:*
  Beste erregularizazio metodo ezagun bat *dropout* izenekoa da [6]. Kasu honetan, zehaztutako probabilitate batekin ausaz aukeratutako neurona batzuk baliogabetzen dira entrenamenduko fasean, ikusgaiak zein ikusezinak. 3.11 irudian, metodo honen adibide bat azaltzen da.



3.11 Irudia: *Dropout* metodoa

Parametroak eguneratzen diren bakoitzean, neuronen azpimultzo berri bat baliogabetuko da. Sare desberdinekin entrenatzearen parekoa izango da eta beraz, sare bakoitza modu desberdinean egokituko da datuetara. Hortaz, orokortzeko gaitasuna lortzea espero da metodo honen bidez.

4. KAPITULUA

Erabilitako tresnak

Proiektua garatzeko hainbat tresna eta teknologia erabili dira. Jarraian, erabilitako tresna bakoitzaren deskribapena egin, proiektuan zertarako erabili den argitu eta instalazio prozesua azalduko da.

4.1 Hardware

Exekutatu behar izan diren prozesuak denbora luzea hartzen dutenez, **CPU**-aren *(Central Processing Unit)* lana arintzeko **GPU** *(Graphics Processing Unit)* bat erabili da. Honako hauek dira erabilitako CPU eta GPU-aren ezaugarriak:

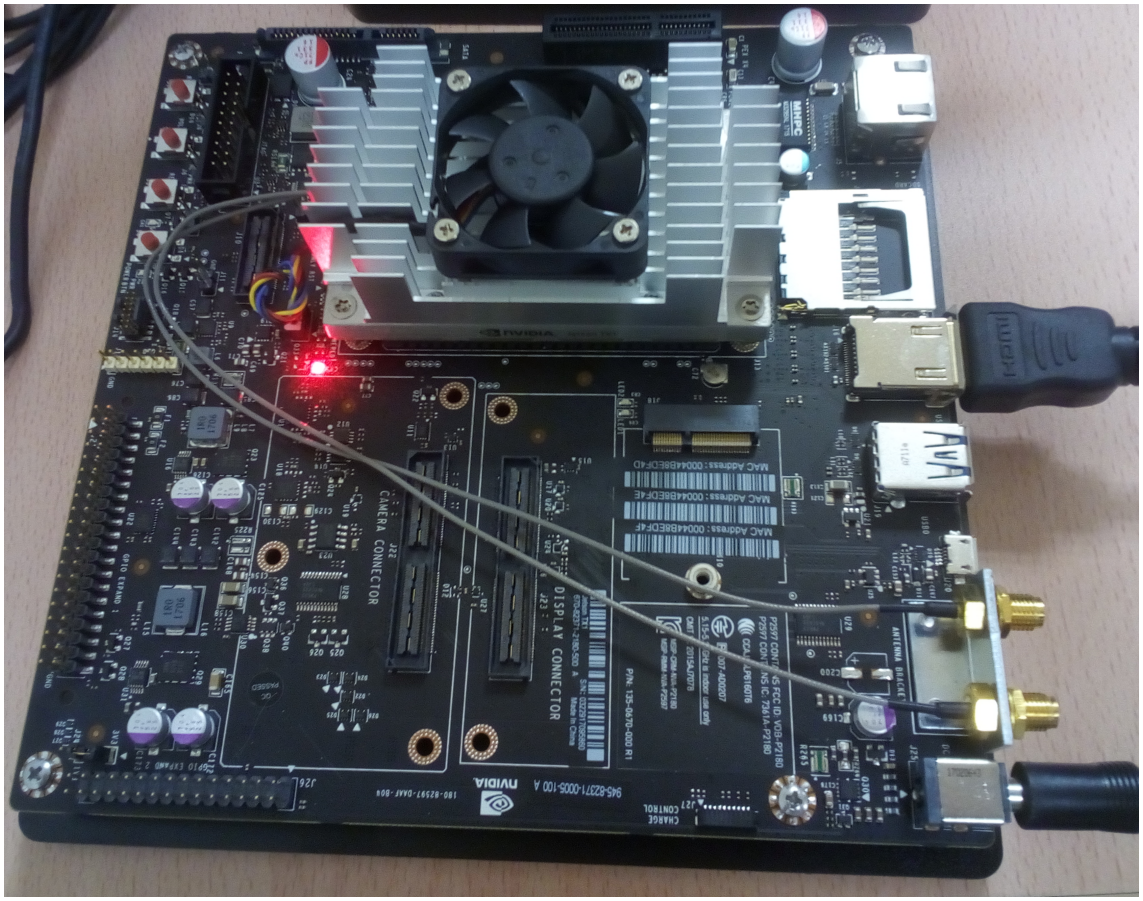
- **CPU:** Intel Core i7-3770 CPU @ 3.4GHz x 8
- **GPU:** GeForce GTX TITAN X 12GB

Mahaigaineko ordenagailuaz gain, sistema mugikorretan erabili ahal izateko plataforma bat erabili da, Jetson TX1 hain zuzen.

4.1.1 Jetson TX1

NVIDIA Jetson GPU-prozesadore paralelo azkarreko konputazio plataforma nagusia da sistema mugikorretan, 4.1 irudia. Bere errendimendu altuko eta potentzia baxuko konputazioek, ikasketa sakoneko eta konputagailu bidezko ikusmeneko kalkulu-kopuru handiko

proiektuetarako plataforma ezin hobea bihurtzen dute.



4.1 Irudia: *Jetson TX1* plataforma

Jetson TX1 NVIDIA-k eskaintzen dituen garapen kit-etariko bat da. Kit hauek adimen artifizialeko proiektuen garapenerako plataformak dira, beharrezko ezaugarri guztiak biltzen dituztenak. Ahalmen gutxiko inguruneetan garatutako konputazio errendimendu handia eskatzen duten aplikazioetarako egokiak dira eta azkar funtziona dezaten diseinatu dira. Gainera, Linux ingurunea instalatuta daukate, API komun askotarako euskarria barneratzen dute eta NVIDIA-ren garapen tresnen kate osoarekin bateragarriak dira. Jetson-en garatzaileentzako kitak plaka batera konektatutako Jetson modulo bat barneratzen dute, energia-iturri, kable eta hasteko laguntza moduan aurreinstalatutako softwarearekin batera.

Beraz, Jetson TX1 munduko lehen superordenagailua da modulu batean, irtenbide indartsu eta oso eraginkorra, adimen artifizialerako gaitutako produktuak sortzeko aproposa. NVIDIA Maxwell GPU familian garatuta dago eta 4.2 irudian azaldutako ezaugarriak biltzen ditu.

	Jetson TX2	Jetson TX1
GPU	NVIDIA Pascal™, 256 CUDA cores	NVIDIA Maxwell ™, 256 CUDA cores
CPU	HMP Dual Denver 2/2 MB L2 + Quad ARM® A57/2 MB L2	Quad ARM® A57/2 MB L2
Video	4K x 2K 60 Hz Encode (HEVC) 4K x 2K 60 Hz Decode (12-Bit Support)	4K x 2K 30 Hz Encode (HEVC) 4K x 2K 60 Hz Decode (10-Bit Support)
Memory	8 GB 128 bit LPDDR4 59.7 GB/s	4 GB 64 bit LPDDR4 25.6 GB/s
Display	2x DSI, 2x DP 1.2 / HDMI 2.0 / eDP 1.4	2x DSI, 1x eDP 1.4 / DP 1.2 / HDMI
CSI	Up to 6 Cameras (2 Lane) CSI2 D-PHY 1.2 (2.5 Gbps/Lane)	Up to 6 Cameras (2 Lane) CSI2 D-PHY 1.1 (1.5 Gbps/Lane)
PCIE	Gen 2 | 1x4 + 1x1 OR 2x1 + 1x2	Gen 2 | 1x4 + 1x1
Data Storage	32 GB eMMC, SDIO, SATA	16 GB eMMC, SDIO, SATA
Other	CAN, UART, SPI, I2C, I2S, GPIOs	UART, SPI, I2C, I2S, GPIOs
USB	USB 3.0 + USB 2.0	
Connectivity	1 Gigabit Ethernet, 802.11ac WLAN, Bluetooth	
Mechanical	50 mm x 87 mm (400-Pin Compatible Board-to-Board Connector)	

4.2 Irudia: *Jetson TX1* plataformaren ezaugarriak

Ikusi daitekeenez, 4GB-eko memoria, 256 nukleoko CUDA NVIDIA Maxwell GPU-a eta Quad ARM® A57/2 MB L2 CPU-a biltzen ditu adibidez.

4.2 Software

Lehenik eta behin, erabilitako sistema eragilea **Ubuntu** izan da, zehazki **Ubuntu 16.04 LTS** *(Long Term Support)*.

Horrez gain, ikasketa sakoneko ereduak garatu eta entrenatu ahal izateko, hainbat software tresna desberdin erabili dira, bai *deep learning* proiektuetan erabiltzeko liburutegiak bai NVIDIA-k honetarako eskainitako software jakin bat.

4.2.1 TensorFlow

TensorFlow kode irekiko software liburutegia da[1], errendimendu handiko zenbakizko kalkuluetarako erabilia. Bere arkitektura malguak inplementazio erraz bat ahalbidetzen du

[1] https://www.tensorflow.org/

plataforma informatiko ugarietan (CPU, GPU). Jatorrian, *Google*-eko adimen artifizial sail barruko *Google Brain* taldeko ingeniari eta ikertzaileek garatu zuten. Ikasketa automatikorako eta sakonerako euskarri handia ematen du eta zenbakizko konputazio malguaren muina beste arlo zientifiko askotarako erabiltzen da.

TensorFlow-en bidez garatutako programak normalean askoz azkarrago exekutatzen dira GPU batean CPU batean baino. Horregatik, GPU erabiliz exekutatuko dira garatutako programa guztiak, bai mahaigaineko ordenagailuan bai Jetson TX1 moduluan.

Instalazioa burutzeko, ondorengo komandoa exekutatu beharko da komando-lerroan *root* moduan:

```
pip install tensorflow-gpu #Python 2.7
pip3 install tensorflow-gpu #Python 3.n
```

Aurrebaldintza: Pip edo Pip3 >= 8.1 gomendatzen da bi kasuetan, Python 2.7 edo Python 3.4+. Horrez gain, GPU-arekin lan egiteko, NVIDIA software berezia instalatu behar da[2].

4.2.2 Keras

Keras, esperimentazio azkarra ahalbidetzeko ideiarekin garatu zen goi-mailako sare neuronalen APIa da[3], Python-en idatzia eta TensorFlow, CNTK edo Theano-ren gainean exekutatzeko ahalmena duena. Kasu honetan, TensorFlow erabili da oinarri moduan. Keras honako ezaugarriak betetzen dituen *deep learning* liburutegia da:

- Prototipoen sorrera erraza eta azkarra ahalbidetzen du.
- Sare konboluzionalak eta errepikariak onartzen ditu.
- CPU eta GPU onartzen ditu.
- Python 2.7-3.6-rekin bateragarria da.

Keras, gizakientzako diseinatua izan den API-a da, ez makinentzako. Hortaz, sintaxi ulergarria eskaintzen du. Gainera, API errazak eta trinkoak eskaintzen ditu, ohiko erabilpenkasuetarako erabiltzaileen ekintza kopurua txikiagotzen du eta erabiltzaileen erroreen aurrean komentario argiak ematen ditu.

[2] https://www.tensorflow.org/install/install_linux
[3] https://keras.io/

Keras-en garatutako ereduak independenteak eta guztiz konfiguragarriak diren moduluz ostutako sekuentzia edo grafo moduan uler daitezke. Hau da, geruza neuronalak, optimizatzaileak, aktibazio funtzioak, etab. modulu independenteak dira eta hauek konbinatuz eredu berriak sortzen dira. Horrez gain, modulu berriak gehitu daitezke modu errazean, adierazmen totala ahalbidetuz eta Keras ikerketa aurreratuetarako egokia bihurtuz.

Esan bezala, ereduak Python kodean idatzita daude, trinkotasuna, arazteko erraztasuna eta luzapenerako erraztasuna ematen duena.

Proiektuan zehar garatu behar izan diren neurona-sareak Keras erabiliz sortu dira, betiere TensorFlow oinarri moduan edukiz. Keras erabiltzea aukeratu da sintaxiaren erraztasun eta ulergarritasunarengatik.

Keras instalatzeko, ondorengoa exekutatu behar da *root* moduan:

```
pip install keras
```

Aurrebaldintza: Python 2.7-3.6, Pip eta oinarri moduan lan egingo duen tresna, kasu honetan, TensorFlow instalatu behar dira.

4.2.3 Caffe

Caffe *deep learning*-erako lan-ingurunea (*framework*) da[4], azkartasunean eta modulartasunean pentsatzen sortu zena. **Berkeley AI Research (BAIR)** taldeak garatu zuen.

Caffe-ren arkitektura adierazgarriak berrikuntza sustatzen du. Ereduak eta optimizazioa konfigurazio bidez definitzen dira kodifikazio zurrunik gabe. CPU eta GPU artean aldatu daiteke aldagai bakar bat konfiguratuz GPU makina batean trebatzeko.

Kode hedagarriak garapen aktiboa sustatzen du. Caffe bere lehen urtean, 1.000 sustatzailek baino gehiagok sortu dute eta aldaketa esanguratsuak izan ditu hauengatik. Laguntzaileei esker, laneko ingurunea egokia da bai kodean baita ereduetan ere.

Bere abiadurak, Caffe ikerketa-esperimentuetarako eta industria-inplementazioetarako ezin

[4] http://caffe.berkeleyvision.org/

hobea izatea egiten du. Caffe eguneko 60 milioi irudi baino gehiago prozesatu ditzake GPU NVIDIA bakar batean. Gainera, liburutegiaren bertsio berriak eta hardwarea oraindik azkarragoak dira. Caffe-k sare konboluzionalen inplementaziorik azkarrenetakoa eskaintzen du. Caffe-k dagoeneko ikerketa proiektu akademikoak sustatzen ditu eta baita eskala handiko industria aplikazioak ere, ikusmen, hizkera eta multimedia arloetan.

Proiektuan Caffe-ren beharra ikusi da NVIDIA-k eskaintzen dituen tresna batzuen erabilerarako, hala nola, jarraian azalduko den DIGITS edota aurretik azaldutako Jetson TX1.

Instalazioari dagokionez, honako pausuak jarraitu behar dira Caffe erabili ahal izateko:

1. Mendekotasunak: Caffe instalatu baino lehen, Protobuf 3[5] beharrezkoa da. Horrez gain, hainbat mendekotasun instalatu behar dira:

   ```
   sudo apt-get install --no-install-recommends build-essential cmake git
   gfortran libatlas-base-dev libboost-filesystem-dev libboost-python-dev
   libboost-system-dev libboost-thread-dev libgflags-dev libgoogle-glog-dev
   libhdf5-serial-dev libleveldb-dev liblmdb-dev libopencv-dev libsnappy-dev
   python-all-dev python-dev python-h5py python-matplotlib python-numpy
   python-opencv python-pil python-pip python-pydot python-scipy
   python-skimage python-sklearn
   ```

2. Iturburua deskargatu:

   ```
   export CAFFE_ROOT=~/caffe
   git clone https://github.com/NVIDIA/caffe.git $CAFFE_ROOT -b 'caffe-0.15'
   ```

3. Python pakete batzuk instalatu:

   ```
   sudo pip install -r $CAFFE_ROOT/python/requirements.txt
   ```

4. Caffe konfiguratu:

   ```
   cd $CAFFE_ROOT
   mkdir build
   cd build
   ```

[5] Instalazioa pausoz-pauso: https://github.com/NVIDIA/DIGITS/blob/master/docs/BuildProtobuf.md

```
cmake ..
make -j"$(nproc)"
make install
```

4.2.4 DIGITS

DIGITS NVIDIA Deep Learning GPU entrenamendu-sistema da[6], kode irekikoa. Neurona-sare sakonak modu azkarrean entrenatzeko balio du irudien sailkapenerako, segmentaziorako edota objektuen detekziorako.

DIGITS-ek ikasketa sakoneko ohiko atazak sinplifikatzen ditu, hala nola, datuen kudeaketa, GPU anitzeko sistemetan neurona-sareen diseinua eta entrenamendua, denbora errealean bistaratze aurreratuen bidezko jardueren monitorizazioa edota errendimendu oneneko ereduaren aukeraketa. Gainera, DIGITS guztiz elkarreragilea da, hortaz, erabiltzaileek sareen diseinuan eta entrenamenduan arreta jarri dezakete, programazioan edo arazketan kontzentratu beharrean.

Laburtuz, hauek dira DIGITS-ek eskaintzen dituen atazak:

- Neurona-sare sakonak diseinatu, entrenatu eta bistaratu irudien sailkapenerako, segmentazioarako edo objektuen detekziorako Caffe, Torch eta TensorFlow erabiliz.
- Aurretik entrenatutako ereduak deskargatu, hala nola, *AlexNet* eta *GoogLeNet*.
- Hiperparametroen balioak aldatu, adibidez ikaskuntza-tasaren balioa aldatu emaitza hobeak lortzekotan.
- Neurona-sareen entrenamendua programatu, gainbegiratu eta kudeatu.
- Emaitzak denbora errealean aztertu.
- Irudi formatu eta iturburu ugari inportatu.

DIGITS erabiliz, hainbat sare desberdin entrenatu dira proiektuan, ondoren garatutako sarearekin lortutako emaitzekin konparatzeko helburuarekin. Gainera, DIGITS erabiliz entrenatutako sareak izan dira ondoren Jetson TX1 gailuan erabili direnak.

DIGITS **Ubuntu 16.04** makina batean instalatzeko, honako pausuak jarraitu behar dira:

[6] https://developer.nvidia.com/digits

1. Aurrebaldintzak: NVIDIA driver bat behar da.

```
CUDA_REPO_PKG=http://developer.download.nvidia.com/compute/cuda/repos/
ubuntu1604/x86_64/cuda-repo-ubuntu1604_8.0.61-1_amd64.deb

ML_REPO_PKG=http://developer.download.nvidia.com/compute/machine-
learning/repos/ubuntu1604/x86_64/nvidia-machine-learning-repo-
ubuntu1604_1.0.0-1_amd64.deb

# Install repo packages
wget "$CUDA_REPO_PKG" -O /tmp/cuda-repo.deb && sudo dpkg -i
/tmp/cuda-repo.deb && rm -f /tmp/cuda-repo.deb

wget "$ML_REPO_PKG" -O /tmp/ml-repo.deb && sudo dpkg -i
/tmp/ml-repo.deb && rm -f /tmp/ml-repo.deb

# Download new list of packages
sudo apt-get update
```

2. Mendekotasunak instalatu:

```
sudo apt-get install --no-install-recommends git graphviz python-dev
python-flask python-flaskext.wtf python-gevent python-h5py python-numpy
python-pil python-pip python-scipy python-tk
```

   Caffe ere instalatuta egon behar da. DIGITS Caffe 0.15 bertsioarekin bateragarria denez, hau instalatu da.

3. Iturburua deskargatu:

```
DIGITS_ROOT=~/digits
git clone https://github.com/NVIDIA/DIGITS.git $DIGITS_ROOT
```

4. Beharrezko Python paketeak instalatu:

```
sudo pip install -r $DIGITS_ROOT/requirements.txt
```

5. Bukatzeko, zerbitzaria martxan jarri:

   ```
   ./digits-devserver
   ```

   Zerbitzaria hemen jarriko da martxan: http://localhost:5000/

4.2.5 Bestelakoak

Aurretik azaldutakoez gain, beste liburutegi eta tresna garrantzitsu batzuk erabili dira.

- **Python**[7]**:** aipatu beharra dago erabilitako lengoaia Python izan dela. Programatzerako orduan bere sintaxiak erraztasuna eskaintzen du. Baina garrantzitsuena honako hau da: Python erabilitako tresna guztiekin bateragarria da, hau da, erabilitako liburutegi guztiak Python lengoaian eskuragarri daude. Proiektuan erabiltzen diren *machine learning* eta *deep learning* liburutegiak Python-en erraz kudeatu daitezke. Aldiz, beste programazio lengoai batzuetan hauek erabiltzeko denbora gehiago beharrezkoa da liburutegi guztiak modu egokian funtzionatzea lortu arte. Python 3.5.2 bertsioa erabili da.

- **NumPy**[8]**:** Python erabiliz informatika zientifikoan lan egiteko funtsezko paketea da. N-dimentsioko matrizeekin lan egiteko eraginkortasuna, funtzio sofistikatuak, algebra lineal erabilgarria, etab. eskaintzen ditu. Kasu honetan, NumPy 1.14.2 erabili da irudiak kargatzeko zein hauen pixelekin lan egiteko.

  Instalatzeko ondorengo komandoa exekutatu behar da *root* moduan:

  ```
  pip install numpy
  ```

  Aurrebaldintza: Pip eta Python >=2.7,!=3.0.*,!=3.1.*,!=3.2.*,!=3.3.*.

- **Scikit-learn**[9]**:** ikasketa automatikoko tekniken aplikaziorako Scikit-learn liburutegia erabili da, zehazki 0.19.1 bertsioa. Liburutegi honek, aurretik aipatutako sailkatzaileen implementazioa eskaintzen du. Kode irekikoa da.

  Instalatzeko ondorengo komandoa exekutatu behar da *root* moduan:

[7] https://www.python.org/
[8] http://www.numpy.org/
[9] http://scikit-learn.org/stable/

```
pip install -U scikit-learn
```

Aurrbaldintza: Pip, NumPy >= 1.8.2, Scipy >= 0.13.3 eta Python (>= 2.7 edo >= 3.3).

- ***OpenCV***[10] *(Open Source Computer Vision Library):* konputagailu bidezko ikusmenerako eta ikasketa automatikorako iturburu irekiko software liburutegia da. Gehienbat denbora errealeko ikusmen aplikazioetara bideratzen da OpenCV eta berez C++ lengoaian idatzita dago. C++, Python, Java eta MATLAB interfazeak ditu eta Windows, Linux, Android eta Mac OS onartzen ditu. Kasu honetan, Linux eta Pythonekin lan egin da.

  Proiektuaren garapenean, web-cam batetik bideoa lortzeko eta irudietan banatzeko erabili da, garatutako ereduaren bitartez objektuen klasifikazioa denbora errealean gauzatzeko. Erabilitako bertsioa 3.4.0 izan da.

  Instalazioari dagokionez, honako komandoak exekutatu behar dira komando-lerroan:

  ```
  [konpiladorea] sudo apt-get install build-essential
  [beharrezkoa] sudo apt-get install cmake git libgtk2.0-dev pkg-config
  libavcodec-dev libavformat-dev libswscale-dev
  [aukerakoa] sudo apt-get install python-dev python-numpy libtbb2
  libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev
  libdc1394-22-dev
  ```

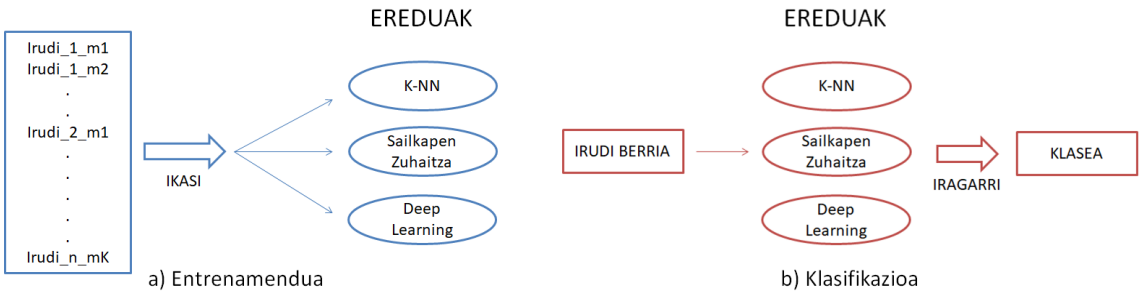
  Aurrebaldintza: Python >= 2.6, Numpy >=1.5 eta ffmpeg edo libav garapen paketeak.

[10] https://opencv.org/

5. KAPITULUA

Garapena

Atal honetan, proiektuan garatutakoa azalduko da pausoz pauso. Laburtuz, datu-base jakin bateko irudiak erabiliz hainbat eredu desberdin entrenatu dira. Horrez gain, web-cam baten bidez irudiak lortu eta unean klasifikatu dira. 5.1 irudian, egindakoaren eskema grafiko bat ikusi daiteke. Bukatzeko, irudi berri batzuk lortu eta aurretik entrenatutako sarea erabiliko da entrenatzen jarraitzeko, emaitzak hobetzen diren aztertzeko. Jarraian, zehatzago azalduko da prozesu guztia.



5.1 Irudia: Garapenaren eskema grafikoa

5.1 Aurrekariak

Erabiliko diren irudiak, ILSVRC12 *(ImageNet Large Scale Visual Recognition Challenge 2012)* datu-basekoak izango dira, **ImageNet** txapelketan erabilitakoak. Bertan, 1000 klase[1] desberdinetan banandutako miloi bat irudi daude eskuragarri.

[1] https://github.com/dusty-nv/jetson-inference/blob/master/data/networks/ilsvrc12_synset_words.txt

5.1.1 Irudien aukeraketa

Hainbeste irudi daudenez eta irudi kantitate erraldoi horrekin egin beharreko probak gehiegi luzatuko zirenez, 1000 klase horietatik bakarrik 14 hartzea erabaki da. Aukeratutako klaseak fakultatean eta bulegoetan aurkitu daitezkeen gauzak izan dira, robot batek fakultatetik ibiltzerakoan ikusi ditzakeen gauzak hain zuzen. Hortaz, hauek izan dira proiektuan erabili diren klaseak (5.2 irudia):

1. Motxila $\longrightarrow$ 917 irudi
2. Zakarrontzia $\longrightarrow$ 863 irudi
3. Erlojua $\longrightarrow$ 653 irudi
4. Ordenagailua $\longrightarrow$ 778 irudi
5. Idazmahaia $\longrightarrow$ 858 irudi
6. Apalategia $\longrightarrow$ 805 irudi
7. Teklatua $\longrightarrow$ 1314 irudi
8. Ordenagailu eramangarria $\longrightarrow$ 1021 irudi
9. Telefonoa $\longrightarrow$ 845 irudi
10. Imprimatzailea $\longrightarrow$ 1053 irudi
11. Pantaila $\longrightarrow$ 1627 irudi
12. Bozgoragailuak $\longrightarrow$ 780 irudi
13. Saltzeko makina $\longrightarrow$ 1406 irudi
14. Leihoa $\longrightarrow$ 709 irudi

Guztira, 13629 irudi erabili dira. Horietatik %10a baliozkotzerako erabiliko da eta gainontzekoa entrenamendurako, hau da, *val* $\sim$ 1363 irudi eta *train* $\sim$ 12266 irudi. Gainera, bi multzoetan klaseen banaketa antzekoa da, hau da, klase bakoitzeko kasu kopurua berdintsua da. Baliozkotzerako kasuak, ereduak inoiz ikusi ez dituen kasuak dira eta ereduaren benetazko portaera aztertzeko erabiltzen dira.

5.2 Irudia: Aukeratutako klaseen adibideak

5.1.2 Irudien aurreprozesaketa

Behin irudiak aukeratuta, sareak datu-base desberdinekin landu ahal izateko, 4 filtro desberdin erabili dira. Horrez gain, kolore-eredu desberdin batekin probatzea ere erabaki da.

- Filtroak:
  Filtro desberdineko irudiak lortzeko, *convert* komandoa erabili da. Komando honek irudietan orotariko aldaketak egitea ahalbidetzen du, formato aldaketa, tamaina aldaketa, filtroak...
  Proiektu honetan, jarraian azalduko diren 4 filtroak erabiliko dira. 5.3 irudian filtroen adibide bana ikusi daiteke.

  – BLUR: irudiaren zarata eta xehetasun maila murrizten ditu. Irudia konbolu-zionatzen du Gauss edo banaketa normala erabiliz.

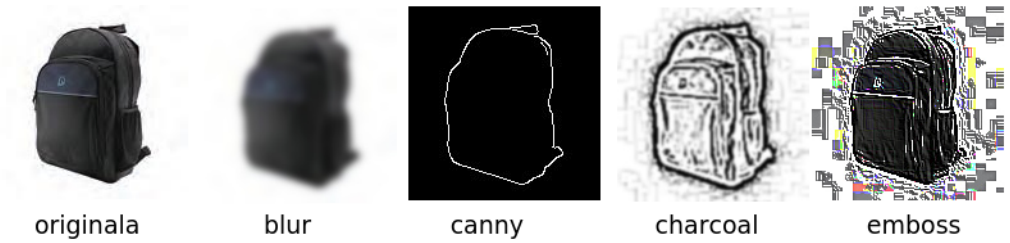
  $$G(u,v) = \frac{1}{2\pi\sigma^2} e^{-(u^2+v^2)/(2\sigma^2)}$$

  $\boldsymbol{\sigma}$ balioak lausotasun maila zehazten du.

  – CANNY: ertzak detektatzeko algoritmo aurreratu bat da. Pixel bakarreko lerro sendoak (bitarrak) sortzen ditu ertz zorrotz guztietan zarata interferentzia oso txikiarekin. *edge* aukera baino emaitza zorrotzagoak lortzen ditu baina ertz detektagailu guztien antzera, arazoak ematen ditu atzealde kargatuta duten irudiekin.

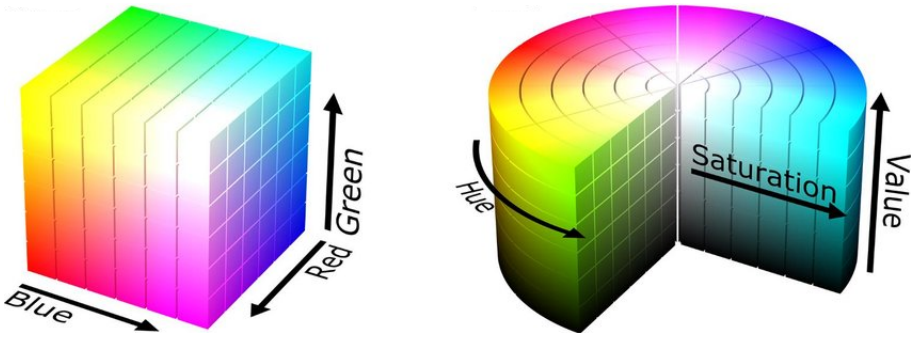
  – CHARCOAL: ikatz-marrazki bat simulatzen du. Aukera hau hainbat alderdietan ertzen detekziorako erabilitako transformazioen antzekoa da. Irudiko ertz nagusiak eta objektuen ertzak arkatz eta ikatz itzaletara bihurtzen ditu.

– EMBOSS: irudia bozeltzeko filtroa. Irudiaren pixel bakoitza nabarmendu edo itzal batekin ordezkatzen da, irudi originalean ertza argia edo iluna den arabera. Horrela, irudi berriak, kokapen bakoitzerako irudi originalaren koloreen aldaketaren tasa irudikatuko du.



5.3 Irudia: Filtroen arteko desberdintasuna

- HSV kolore eredua:
  Erabilitako filtroez gain, irudiak beste kolore-eredu batera pasa dira hainbat proba egiteko eta hobekuntzak lortzen diren ikusteko. Bere izena *Hue, Saturation, Value (HSV)* hitzetatik dator, ñabardura, saturazioa eta distira hurrenez hurren.
  HSV kolore eredua zilindro-itxurako koordenatuen irudikapena da. Gainera, gizakiak koloreak eta hauen propietateak antzemateko erara gehiago gerturatzen da, koloreen tonalitateak multzokatzen direlako. Aldiz, RGB ereduan, koloreak ez dute zertan multzokatuak egon behar. 5.4 irudian, bi kolore ereduen arteko konparaketa ikusi daiteke.



5.4 Irudia: *RGB* eta *HSV* kolore ereduen konparaketa

HSV eredua, RGB ereduaren transforamzio ez-lineala da, koordenatu zilindrikoetan. Kolore bakoitza, aurretik aipatu dugun moduan, dimentsio hauek definitzen dute:

– Ñabardura: kolore ereduaren kolore atala da, 0-360ºtan adierazita. 5.1 taulan kolore bakoitzari dagokion gradu tartea agertzen da.

Kolorea	Gradua
Gorria	0-60
Horia	60-120
Berdea	120-180
Zian	180-240
Urdina	240-300
Magenta	300-360

5.1 Taula: *Hue* balioak

– Saturazioa: koloreko gris kantitatea adierazten du, %0-100 balioak hartuz. Kolorge efektu bat lortu daiteke saturazioa zerora gutxituz eta modu horretan gris gehiago sartuz. Batzuetan balio hau 0-1 tartean adierazten da, non 0 grisa den eta 1 oinarrizko kolore bat.

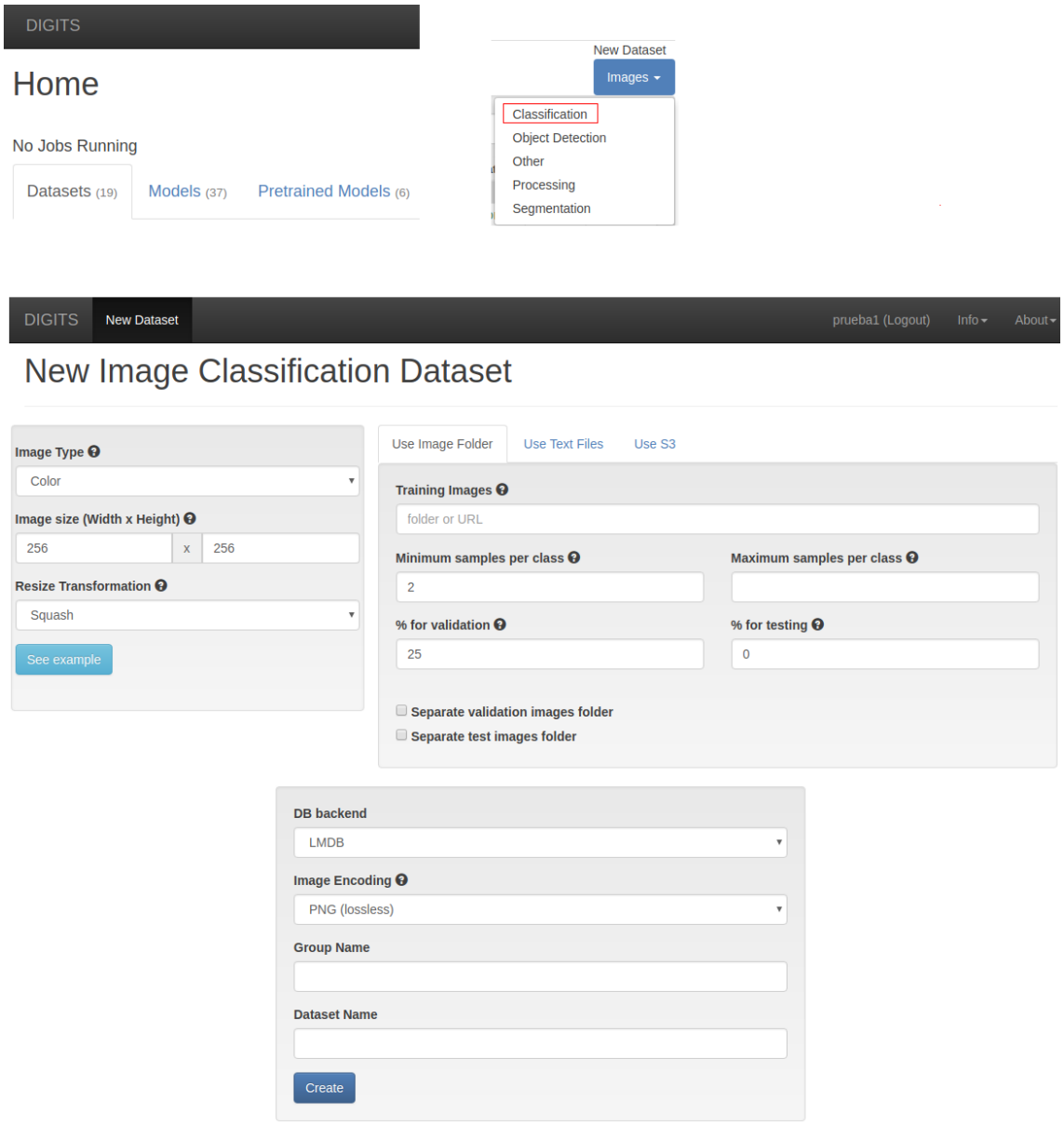
– Distira: saturazioarekin batera funtzionatzen du eta kolorearen intentsitatea eta distira adierazten du. %0-100 balioak hartzen ditu, non 0 guztiz beltza den eta 100 distiratsuena den.

Jarraian, erabilitako irudi baten adibidea ikusi daiteke (5.5 irudia), kolore-ereduen arteko desberdintasuna antzeman ahal izateko.

5.5 Irudia: *RGB* eta *HSV* kolore ereduen adibidea

5.1.3 Datu-baseen sorrera DIGITS-en

Entrenamendua DIGITS erabiliz egiteko, datu-base guztiak sortu behar ditugu bertan. Datu-baseak sortzeko aukera ematen du DIGITS sistemak 5.6 irudian ikusi daitekeen moduan. Modu horretan 6 datu-baseak sortu dira.



5.6 Irudia: Datu-baseak sortu *DIGITS* erabiliz

Bukatzeko, aipatu beharra dago sare konboluzionalekin egindako proba guztiak 64x64 tamainako irudiekin egin direla, ordenagailuan irudi guztiak prozesatu ahal izateko.

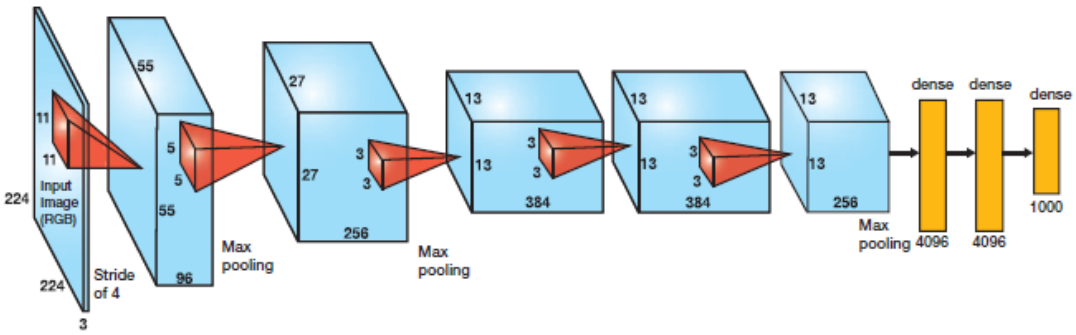
5.2 Erabilitako sareak

Irudien aurreprozesaketa bukatu ondoren, hainbat proba desberdin egin dira, neurona-sare konbouzional desberdinak erabiliz.

5.2.1 DIGITS

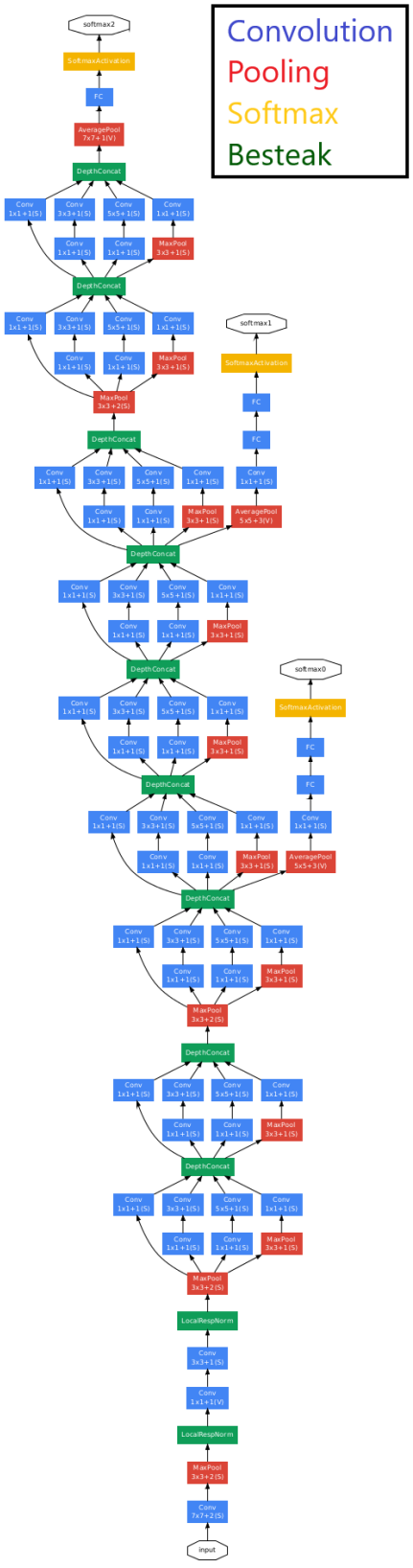
Alde batetik, DIGITS-ek eskainitako bi arkitektura desberdin erabili dira. Bi arkitektura hauek **ImageNet** txapelketako irabazleak izan dira urte desberdinetan.

- *AlexNet* [9]:
  Bost geruza konboluzional ditu, $11x11, 5x5$ eta $3x3$ tamainetakoak. Gainera, lehenengo, bigarren eta bosgarren geruza konboluzionalen atzetik *max-pooling* geruza bat dago. Bukatzeko, hiru *fully connected* geruza ditu. Geruza guztietan **ReLU** aktibazio funtzioa erabiltzen da, azkenekoan izan ezik bertan **softmax** erabiltzen da eta. Optimizatzaile moduan **SGD** erabiltzen du eta erregularizaziorako dropout erabiltzen du *fully connected* geruzetan. 5.7 irudian, azaldutako arkitektura ikusi daiteke.



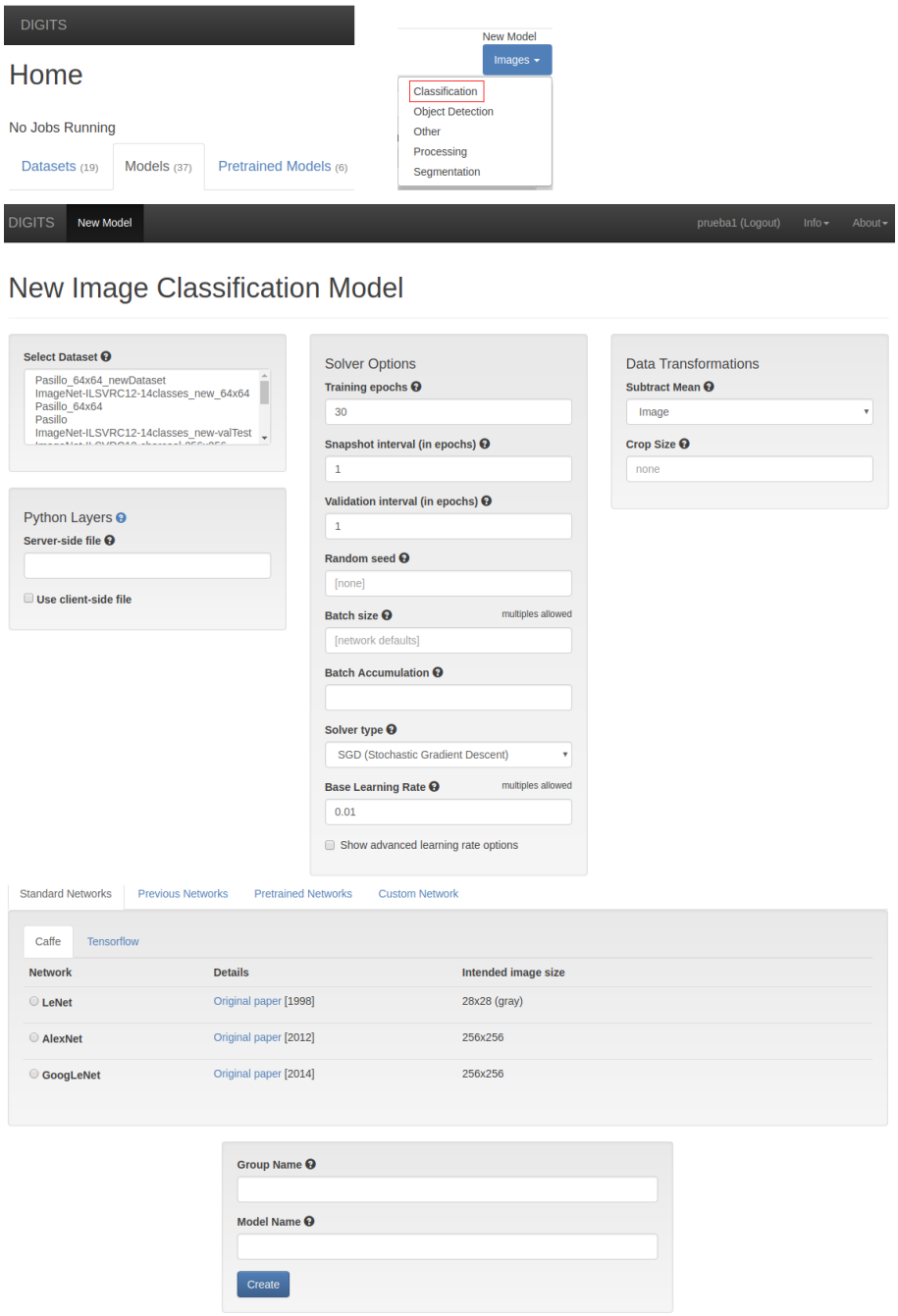
5.7 Irudia: *AlexNet* arkitektura

- *GoogLeNet* [15]:
  LeNet [10] sarean inspiratu zen baina *inception* izeneko modulu berria inplementatzen du. Modulu honek konboluzio txiki asko biltzen ditu, parametro kopurua ugari txikitzeko. 22 geruzez osatutako neurona-sare konboluzionala da baina, 60 milioi parametrotatik (AlexNet 5.7) 4 milioi izatera pasatzen da. *Inception* moduluak $1x1$, $3x3$ eta $5x5$ konboluzioak ditu paraleloan. Modu honetan, sareak erabakiko du ikasi eta erabili behar den informazioa. Optimizatzaileari dagokionez, kasu honetan **RMSProp** erabiltzen du. 5.8 irudian, azaldutako arkitektura ikusi daiteke.



5.8 Irudia: *GoogLeNet* arkitektura

Azaldutako bi neurona-sare hauek erabiliz, irudiekin osatutako 6 datu-baseekin (originala, charcoal, canny, emboss, blur, HSV) probak egin dira. Hurrengo atalean, lortutako emaitzak aztertuko dira. DIGITS-en bidez entrenamendua martxan jartzea oso erraza da, sortuta dauden datu-baseetatik bat, sarearen arkitektura eta hainbat parametroen balioak aukeratu behar dira, 5.9 irudian ikusi daiteke.

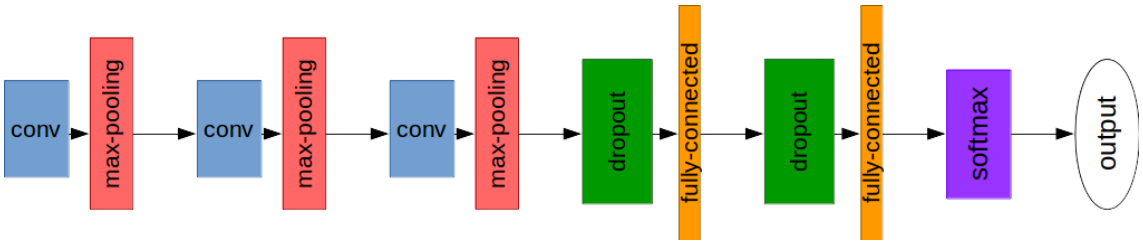


5.9 Irudia: Ereduak sortu *DIGITS* bidez

5.2.2 Bestelakoa

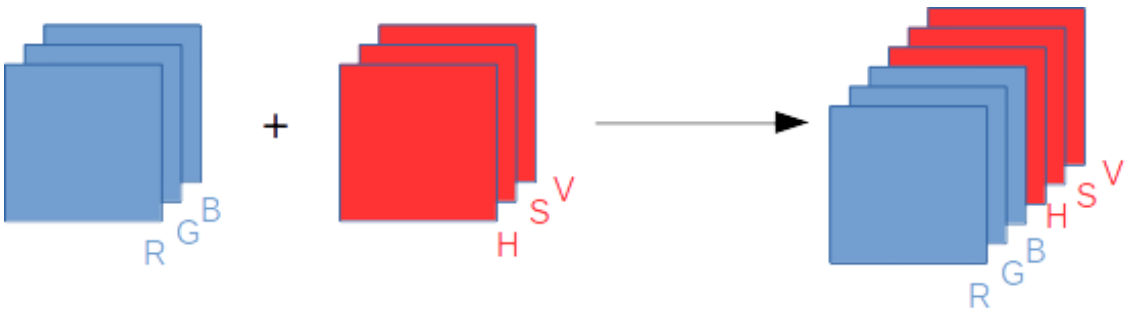
Arkitektura ezagun horiez gain, beste neurona-sare konboluzional desberdin bat garatu da, kasu honetan Keras erabiliz.

Sarea, 5.10 irudian ikusi daitekeen moduan, 3 geruza konboluzionalez osatuta dago, guztiak 3*x*3 tamainakoak eta bakoitzean 32 filtro erabiltzen dira. Geruza konboluzional bakoitzaren atzetik 2*x*2 tamainako *max-pooling* geruza bat dago. Ondoren, *dropout* eta *dense* geruzak datoz. Lehenengo *dropout* geruzan 0.25eko probabilitatearekin ezabatuko dira neuronak eta bigarrenean aldiz, 0.5eko probabilitatearekin. *Dense* geruzei dagokienez, lehenengoak 128 irteera ditu eta bigarrenak berriz, klase kopuru bezain irteera izango ditu. Azkeneko geruzan **softmax** aktibazio-funtzioa erabili arren, gainontzekoetan **ReLU** erabiltzen da. Horrez gain, aipatzekoa da geruza konboluzionalen irteerak normalizatzen direla.



5.10 Irudia: Garatutako sare konboluzionalaren arkitektura

Sare honen garapenerako ez da DIGITS entrenamendu-sistema erabili arrazoi jakin batengatik: sare honen bidez, irudien kanalen konbinaketa eginez hobekuntzak dauden ikusi nahi da. Hau da, kasu honetan ez dira datu-base guztiak banan-banan sarea entrenatzeko erabiliko. Aldiz, irudi originalak gainontzeko datu-baseekin konbinatuko dira eta hori izango da sarrera moduan sareari pasako zaiona. 5.11 irudian ikusi daiteke adibide bat. Beraz, orain sarrerak 3 kanal izan beharrean (sakonera 3), 6 kanal izango ditu. *Charcoal* eta *canny* filtroekin lortutako irudiak grisen eskalan daudenez, kanal bakarra izango dute eta irudi originalarekin konbinatzerakoan 4 kanal izango dituzte. Laburbilduz, bost datu-base desberdinekin egingo da lan sare konboluzional honekin: RGB+HSV, RGB+BLUR, RGB+EMBOSS (sakonera 6) eta RGB+CANNY, RGB+CHARCOAL (sakonera 4).



5.11 Irudia: Irudien kanalen konbinaketa

5.3 Machine Learning

Sare neuronalekin egindako probak onargarriak diren ikusteko, ikasketa automatikoko K-NN eta Decision Tree teknikak erabili dira. Ikasketa automatikoarekin lortutako emaitzak ikasketa sakonarekin lortutakoekin konparatuz, ikasketa sakona erabiltzea aukera ona den ikusi daiteke. Hala ere, proba hauek ez dira datu-base guztiekin egin, bakarrik DIGITS-eko bi sareekin emaitzarik onenak lortutako bi datu-baseekin. Ikasketa automatikoko proba hauek, sare konboluzionala garatu baino lehen gauzatu dira.

5.4 Web kamera

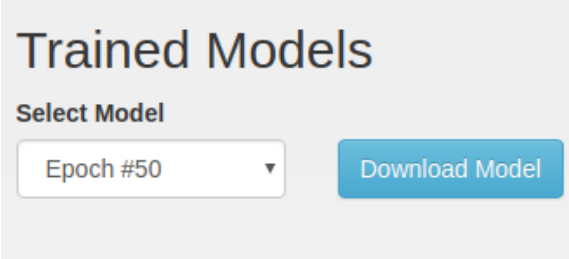
Neurona-sare konboluzionala garatu ondoren, denbora errealean iragarpenak egiteko, web kamera batetik irudiak hartzea eta momentuan klasifikatzea erabaki da.

Horretarako, eredua gorde egin da `model.save('model.h5')` bidez. Ondoren, *OpenCV* erabili da kameratik bideoa jasotzeko, `cap = cv2.VideoCapture(0)` eta `ret, frame = cap.read()` bideoa fotogramaka lortzeko. Lortutako irudia 64x64 tamainara transformatuko da (`image = resize_image(frame, 64)`), ereduak tamaina horretako sarrerak onartzen ditu eta. Bukatzeko, `load_model('model.h5')` bidez aurretik gordetako eredua kargatu eta irudiaren klasea ereduaren bitartez iragarriko da (`model.predict(image)`), komando-lerroan iragarritakoa inprimatuz.

Uneoro kamaratik lortzen dena ikusi ahal izateko, `cv2.imshow('frame',frame)` metodoa erabili da.

5.5 Jetson TX1

Aurreko azterketa guztiak egin ondoren, garatutako sarearekin lortutako eredua Jetson TX1 modulura pasako da. Horretarako DIGITS entrenamendu-sistemaren laguntza erabili da. Honek, entrenatutako ereduak jaisteko aukera ematen du (5.12 irudia) eta deskargatutakoa Jetson TX1 moduluan sartuz, eredua kargatua izango dugu bertan.



5.12 Irudia: Ereduak deskargatzeko aukera

Garatutako sarea DIGITS sisteman entrenatu ahal izateko, sarea Caffe ingurunera itzuli behar izan da, DIGITS-ek ez baitu Keras onartzen. TensorFlow erabili daitekeen arren, Caffe erabiltzea erabaki da ingurune berri bat lantzeko helburuarekin. Jarraian, itzulpenaren adibide txiki bat ikusi daiteke, *max-pooling* geruza baten itzulpena hain zuzen.

```
KERAS:
pool_1 = MaxPooling2D(pool_size=(2, 2))(conv_1)

CAFFE:
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 2
  }
}
```

Sarea itzuli ondoren, DIGITS erabiliz entrenatu eta deskargatu behar da. Eredua Jetson

TX1 moduluan kargatu, deskonprimitu eta jarraian honek eskaintzen dituen bi aukera erabili dira:

1. *imagenet-console:* sarrera moduan bi parametro jasotzen ditu, sarrera irudirako bidea eta irteera irudirako bidea. Sarrerako irudia kargatu, ereduaren bitartez klasifikazioa gauzatu eta emaitza irudian idatzi ondoren irteerako irudian gordetzen du. *aarch64/bin* karpetan aurkitzen da exekutagarria. Erabiliko den eredua *networks/nire_eredua* karpetan dagoela suposatuz, honako hau exekutatu beharko da:

```
$ NET=networks/nire_eredua
$./imagenet-console sarrera.png irteera.png \
> --prototxt=$NET/deploy.prototxt \
> --model=$NET/snapshot_iter_X.caffemodel \
> --labels=$NET/labels.txt \
> --input_blob=data \
> --output_blob=softmax
```

2. *imagenet-camera:* irudien azterketarako debora errealeko demoa da. *aarch64/bin* karpetan dago exekutagarria. Kasu honetan, ez du sarrerako parametrorik jasotzen, Jetson TX1-era USB bitartez konektatutako kameratik lortzen dira irudiak. Fotograma segunduko (FPS), bideotik klasifikatutako objektuaren izena eta klasifikatutako objektuaren konfiantza balioa *openGL* leihoan agertzen dira.

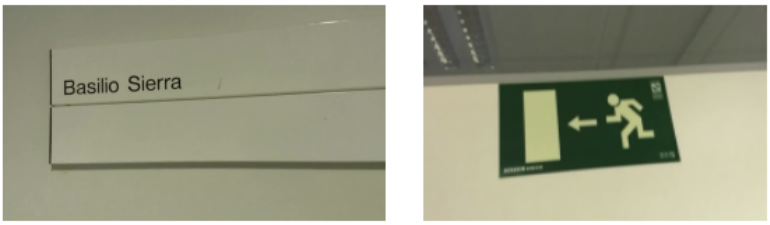
```
$ NET=networks/nire_eredua
$./imagenet-camera \
> --prototxt=$NET/deploy.prototxt \
> --model=$NET/snapshot_iter_X.caffemodel \
> --labels=$NET/labels.txt \
> --input_blob=data \
> --output_blob=softmax
```

Beraz, bi aukera desberdin horiek erabiliz, datu-baseko klase berdineko hainbat irudi eta objektu klasifikatu dira, entrenatutako ereduaren benetazko erabilera ikusteko.

5.6 Berrentrenamendua: korridoreko irudiak

Bukatzeko, aurretik entrenatutako sarea erabili da datu-base berri batekin entrenatzen jarraitzeko. Hau da, datu-base berri bat sortu eta sarea datu-base berri honekin hasieratik entrenatu beharrean, aurretik entrenatuta dagoen sarea erabili eta entrenatzen jarraituko da irudi berriekin.

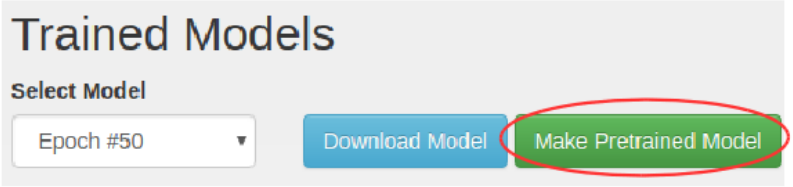
Datu-base berria Informatika Fakultateko korridore bateko irudiek osatuko dute. Irudi hauek, mugikorreko kamararen bidez egindako bideo batetik lortu dira. Kasu honetan, bi klase bakarrik desberdinduko dira: kartelik bai, kartelik ez. Bulegoko pertsonaren izena duen kartela identifikatu nahi da. Baiezko klasea bulegoen kartelen irudiek osatuko dute eta ezezkoa, aldiz, kartela ez den edozer. 5.13 irudian datu-base berri honen klaseen adibide bat agertzen da.



5.13 Irudia: Datu-base berriaren klaseen adibidea

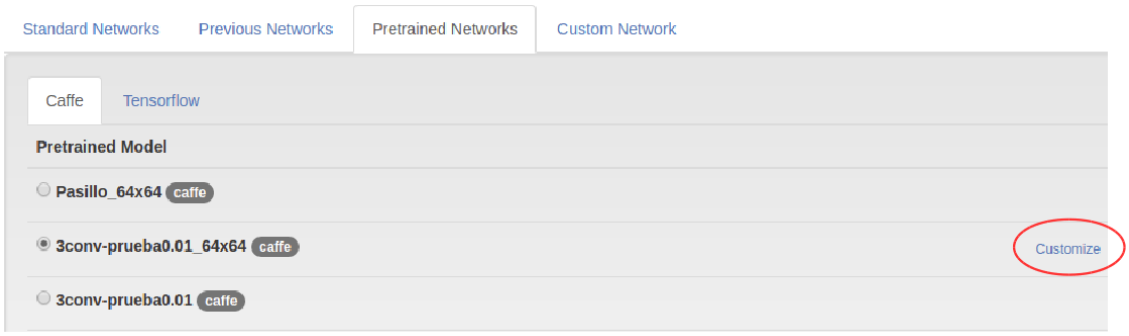
Guztira, 1931 irudi erabili dira, 1153 ezezko klaserako eta 778 baiezkorako. Horrez gain, 272 irudi gorde dira, 214 ezezko eta 58 baiezko, behin eredua entrenatuta, hau testeatzeko.

DIGITS-ek aukera ematen du aurretik entrenatutako ereduak gorde (5.14) eta etorkizunean erabiltzeko.



5.14 Irudia: Aurretik entrenatutako eredua gorde

Aurretik entrenatutako eredu bat erabiltzerakoan, aldaketa txiki bat egin behar da. Kasu honetan bi klase erabiliko direnez eta erabiliko den ereduan 14 klase zeudenez, azkeneko geruzaren irteera kopurua aldatu behar da. Hori egin beharrean, DIGITS-ek azkeneko geruzari klase kopuru bezain irteera kopuru esleitzen dizkionez, azkeneko geruzari izena aldatzearekin nahikoa da. Modu horretan, gainontzeko geruzek aurretik entrenatutakoaren balioak hartuko dituzte eta azkenekoak, izena aldatuta duenez, irteera kopuru egokia izango du, bi hain zuzen. 5.15 irudian ikusi daiteke *Pretrained Networks* atalean gordetako ereduak erabili daitezkeela eta *customize* gainean sakatuz, azaldutako aldaketak egin daitezke.



5.15 Irudia: Aurretik entrenatutako eredua aukeratu

Eredu berria entrenatu ondoren, testerako gordetako irudiekin probak egin dira, bai DIGITS bai Jetson TX1 erabiliz.

6. KAPITULUA

Lortutako emaitzak

Atal honetan, egindako azterketa guztien emaitzak azalduko dira. Banan-banan gauzatutako proba guztiak aztertu eta lortutako emaitzen arabera hainbat ondorio aterako dira.

6.1 DIGITS

Lehenik eta behin, DIGITS erabiliz lortutako emaitzak aztertuko dira. Aurretik esan bezala, honek jada inplementatutako hainbat sare eskaintzen ditu eta horietatik *AlexNet* eta *GoogLeNet* erabiliko dira, hamalau klaseko datu-baseekin. 6.1 taulan, datu-base desberdinekin lortutako emaitzak ikusi daitezke.

	ORIGINAL	CHARCOAL	CANNY	BLUR	EMBOSS	HSV
AlexNet	60%	61%	56%	57%	60%	56%
GoogLeNet	59%	58%	53%	52%	53%	52%

6.1 Taula: *DIGITS* erabiliz lortutako emaitzak

Taulari erreparatuz, *AlexNet* sareak emaitza hobeak lortzen dituela ikusi daiteke, nahiz eta asmatze-tasa %60koa besterik ez izan. Datu-base desberdinak konparatuz, filtroek hobekuntza handirik lortzen ez dutela antzematen da, irudi originalak erabiliz pareko emaitzak lortzen dira eta.

Bi sare hauek emaitza egokiak lortzen dituzte *ILSVRC* datu-base honekin. Baina, ikusten

denez, aukeratutako klaseekin ez dira hain egokiak izan. Klaseen eta irudien aukeraketa ereduaren emaitzetan guztiz eragiten dute.

6.2 Sailkatzaileak

Oraingoan, *deep learning* erabiltzeak abantailak ekartzen dituen ala ez ikusteko, *machine learning*-eko bi sailkatzaile desberdinekin lortutako asmatze-tasak aztertuko dira, berriz ere hamalau klaseko datu-baseekin.

Probak datu-base guztiekin egin beharrean, bakarrik emaitzarik onenak lortutako biekin egin dira, hau da, irudi originalak eta *charcoal* filtroarekin lortutakoak erabiliz. Datu-base guztiekin ez probatzearen arrazoia honako hau da: saiakuntza bakoitzak nahiko denbora behar du irudiak prozesatzeak denbora hartzen duelako eta lortu nahi den ondorioa ateratzeko nahikoa da bi datu-basekin lan egitea.

Alde batetik, K-NN algoritmoarekin lortutako asmatze-tasak agertzen dira 6.2 eta 6.3 tauletan, 64x64 tamaineko irudiak eta 128x128 tamaineko irudiak erabiliz hurrenez hurren.

64x64	ORIGINAL	CHARCOAL
k=1	18.52%	12.03%
k=3	17.05%	11.27%
k=5	17.64%	10.54%

6.2 Taula: *K-NN* sailkatzailearekin lortutako emaitzak, 64x64 tamainako irudiekin

128x128	ORIGINAL	CHARCOAL
k=1	20.66%	10.76%
k=3	16.64%	12.38%
k=5	19.04%	9.87%

6.3 Taula: *K-NN* sailkatzailearekin lortutako emaitzak, 128x128 tamainako irudiekin

Bestetik, 6.4 taulan Decision Tree sailkatzailearekin lortutako balioak agertzen dira.

Decision Tree	ORIGINAL	CHARCOAL
64x64	16.16%	13.63%
128x128	16.84%	14.22%

6.4 Taula: *Decision-tree* sailkatzailearekin lortutako emaitzak

Tamaina handiagoko irudiak erabiliz, informazio gehiago dago pixel kopurua handitzen delako. Hala ere, batzuetan, informazio horrek ez ditu emaitzak hobetzen sailkapenerako baliagarria suertatzen ez delako. Kasu honetan, batzuetan hobekuntzak ekartzen ditu eta beste batzuetan ez. Orokorrean, emaitzak berdintsu mantentzen dira.

Datu-basei dagokienez, sailkatzaileak erabiltzerakoan orokorrean irudi originalek emaitza hobeak lortzen dituzte, asmatze-tasarik hoberena 1-NN sailkatzaileak lortuz %20.66ko balioarekin. Horrez gain, K-NN sailkatzaileak sailkapen hobea egiten du nahiz eta desberdintasuna txikia izan.

Aurreko atalean (6.1) lortutako balioekin konparatuz, irudi hauen klasifikaziorako neurona-sare konboluzionalak erabiltzea aukera egokia dela ondorioztatu daiteke. Lehen aipatu den moduan *AlexNet* eta *GoogLeNet* sareen bitartez asmatze-tasa altuak lortu ez diren arren, *machine learning* teknikak erabiltzetik hobekuntza handia antzeman daiteke.

6.3 Neurona-sare konboluzionala

Jarraian garatutako sare konboluzionalaren bitartez lortutako emaitzak aztertuko dira, hamalau klaseko datu-baseak erabiliz kasu honetan ere. Sare honen helburu nagusia, *AlexNet* eta *GoogLeNet* sareek lortutako emaitzen mailara iristea da. Hau da, gutxienez antzeko asmatze-tasa lortu nahi da eta ahal izanez gero, emaitza hauek hobetu.

Hasteko, bakarrik irudi originalak erabiliko dira. Bi optimizatzaile desberdin erabiltzea erabaki da, **SGD** eta **adam**, ikaskuntza-tasak 0.01 balioa hartuko du eta 100 *epoch*-eko entrenamendua gauzatu da. Emaitzak ebaluatzeko hiru magnitude desberdin erabiliko dira, **ImageNet** bezalako txapelketetan kontuan hartzen direnak:

1. Top 1: benetazko klasea ereduak itzulitako probabilitate handieneko klasea da.
2. Top 3: benetazko klasea ereduak itzulitako probabilitate handieneko 3 klaseen artean aurkitzen da.

3. Top 5: benetazko klasea ereduak itzulitako probabilitate handieneko 5 klaseen artean aurkitzen da.

Entrenamenduko eta baliozkotzeko kasuekin lortutako balioak konparatuko dira jarraian, 6.5 eta 6.6 taulei erreparatuz. Entrenamenduko zein baliozkotzeko emaitzak tratatuko dira, biak kontuan hartuz *overfitting* gertatzen ari den ondorioztatu daitekeelako adibidez.

entrenamendua	top 1	top 3	top 5
SGD	89.34%	99.19%	99.86%
ADAM	53.80%	78.44%	88.22%

6.5 Taula: Sare konboluzionala erabiliz lortutako emaitzak, *train*

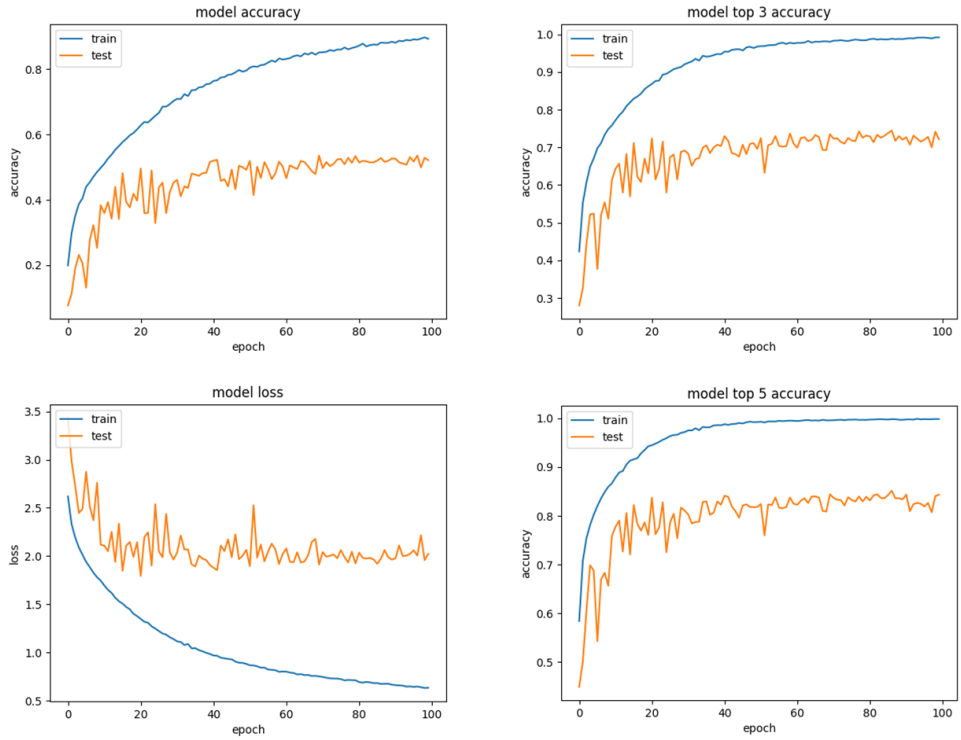
baliozkotzea	top 1	top 3	top 5
SGD	52.18%	72.18%	84.35%
ADAM	45.83%	69.59%	82.66%

6.6 Taula: Sare konboluzionala erabiliz lortutako emaitzak, *val*

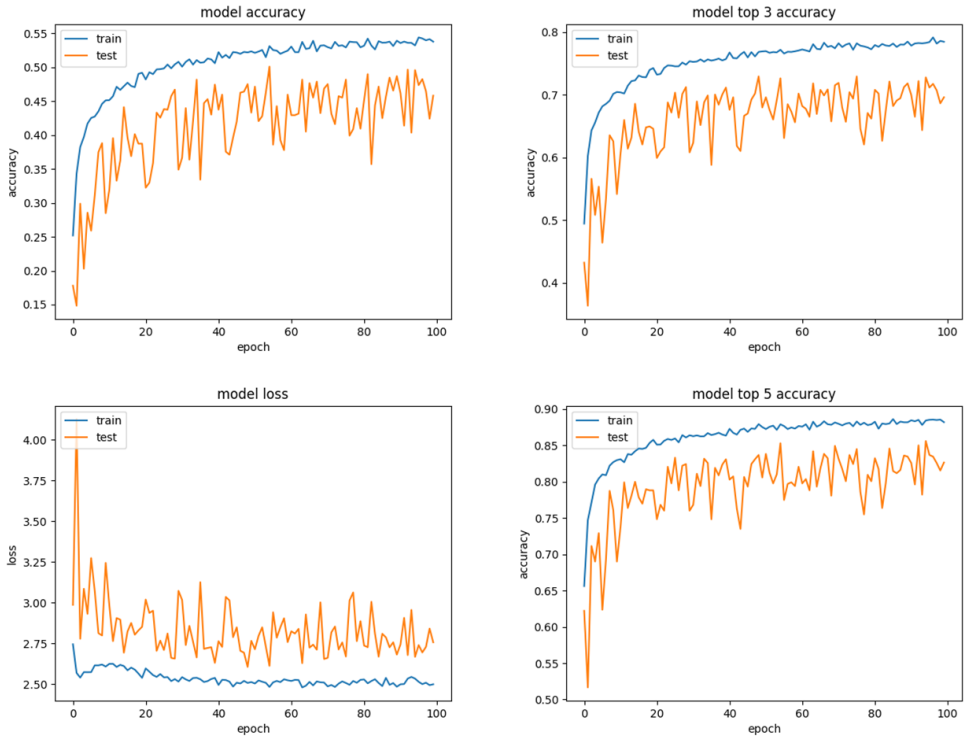
Lehenik eta behin, hiru neurketa desberdinak aztertuz, asmatze-tasa hobetzen doa esperotako moduan. Azken finean, hauen asmatze-tasek modu honetan jokatu behar dute derrigorrez: $top1 \leq top3 \leq top5$.

Horrez gain, entrenamenduko kasuekin emaitza hobeak lortzen dira. Azken finean, entrenatzeko erabiltzen diren kasuak izanik, asmatze-tasa altuagoa izatea normala da. Hala ere, entrenamenduko emaitzak hobetzen jarraitu eta baliozkotzekoak okertzen badoaz, *overfitting* gertatzearen seinalea da. Arrazoi horregatik, hainbat grafiko desberdin lortu dira (6.1 irudia) entrenamenduan zehar gertatutako asmatze-tasa aldaketak aztertzeko.

6.1 grafikoetan ikusi daitekeen moduan, ez dirudi *overfitting*-ik gertatu denik. Hau da, entrenamenduko emaitzak hobeak izan arren, baliozkotzekoak hobetzen joan dira eta azkeneko unean berdin mantendu dira, inoiz ez okertuz. Hala ere, **SGD** erabiliz, entrenamenduko eta baliozkotzeko emaitzen artean desberdintasuna handiagoa da **adam** erabiltzerakoan baino.



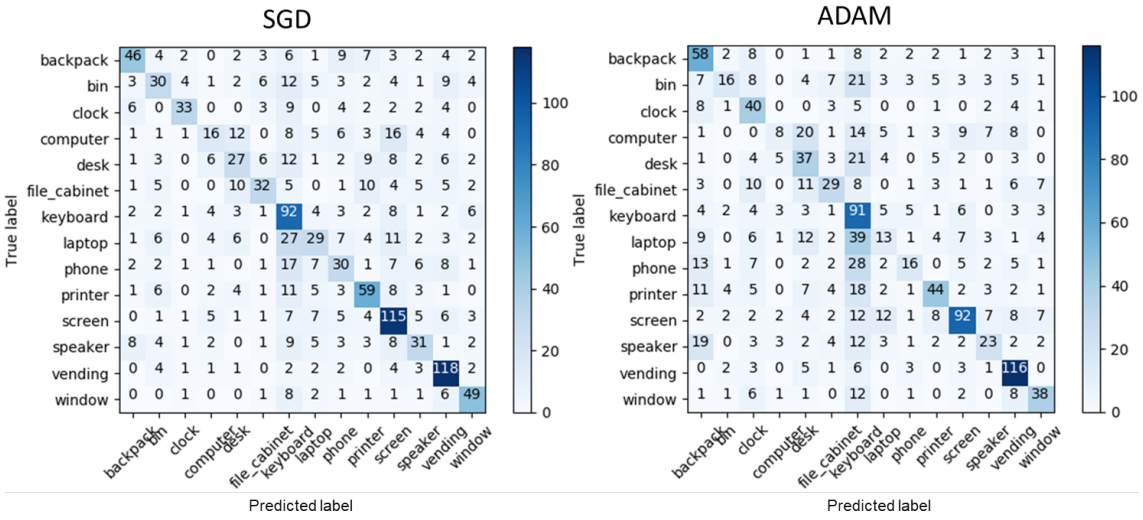
(a) SGD



(b) Adam

6.1 Irudia: Ereduaren entrenamendu prozesua, *SGD* eta *adam* optimizatzaileak erabiliz

Bukatzeko eta emaitzen balioak hobeto ulertu ahal izateko, nahasketa matrizea *(confusion matrix)* lortzea erabaki da, 6.2 irudia. Modu honetan, gehien iragartzen diren klaseak, errore gehien ematen dituzten klaseak, etab. azalduko dira.



6.2 Irudia: Bi optimizatzaileak erabiliz lortutako nahasketa-matrizeak

Modu okerrean gehien iragartzen den klasea teklatua, *keyboard*, da. Nahikoa zentzuzkoa da kontuan hartzen badugu portatilen irudietan teklatua agertzen dela edota idazmahaien irudi gehienetan ere teklatuak agertzen direla. Gainera, telefonoen teklekin ere nahastea erraza da. Bestetik, iragarpen errrore asko egon arren, matrizearen diagonala nabaritzen da, hortaz, klase gehienetan modu egokian iragarritako kasuak oker iragarritakoak baino gehiago dira.

AlexNet eta *GoogLeNet* arkitekturen emaitzak hobetu ez diren arren, antzeko asmatze-tasak lortu dira, %52.18ko balioa lortuz.

6.3.1 Kanalen konbinaketa

Irudi originalekin probak egin ondoren, aurreko atalean azaldutako kanalen konbinaketa-rekin lortutako emaitzak aztertuko dira. **SGD** optimizatzailearekin emaitza hobeak lortu direnez, hau erabiltzea erabaki da. Berriz ere, entrenamendua zein baliozkotzea aztertuko dira eta *top1, top2* eta *top3* neurketak. 6.7 eta 6.8 tauleetan komentatuko diren emaitzak ikusi daitezke.

entrenamendua	top 1	top 3	top 5
RGB + CHARCOAL	93.43%	99.60%	99.93%
RGB + CANNY	92.12%	99.49%	99.96%
RGB + EMBOSS	92.60%	99.60%	99.90%
RGB + BLUR	92.33%	99.56%	99.92%
RGB + HSV	91.02%	99.41%	99.93%

6.7 Taula: Kanalen konbinaketaren bitartez lortutako emaitzak, *train*

baliozkotzea	top 1	top 3	top 5
RGB + CHARCOAL	56.72%	79.54%	89.90%
RGB + CANNY	56.10%	76.73%	87.36%
RGB + EMBOSS	54.24%	74.96%	85.56%
RGB + BLUR	56.02%	76.47%	87.34%
RGB + HSV	54.26%	74.50%	86.13%

6.8 Taula: Kanalen konbinaketaren bitartez lortutako emaitzak, *val*

Emaitzetan ikusi daitekeenez, kanalen konbinaketaren bidez ez da hobekuntza handirik lortzen. Orokorrean, 6.5 eta 6.6 tauletan **SGD**-rekin lortutako asmatze-tasekin konparatuz, gehienez $\sim$%4ko hobekuntza lortu da. Hortaz, kanal gehiagorekin irudiaren informazio gehiago eduki arren, informazio gehigarri hori sailkapenerako oso baliagarria ez dela ondorioztatu daiteke.

Kasu honetan ere *charcoal* filtroarekin lortutako irudiekin lortzen dira emaitzarik hoberenak. Okerrenak, aldiz, *emboss* filtroak eta *HSV* kolore ereduak lortu dituzte. Hala ere, konbinaketa guztiek antzeko asmatze-tasak eman dituzte, %91-%94 tartean entrenamenduari dagokionez eta %54-%57 tartean baliozkotzeari dagokionez.

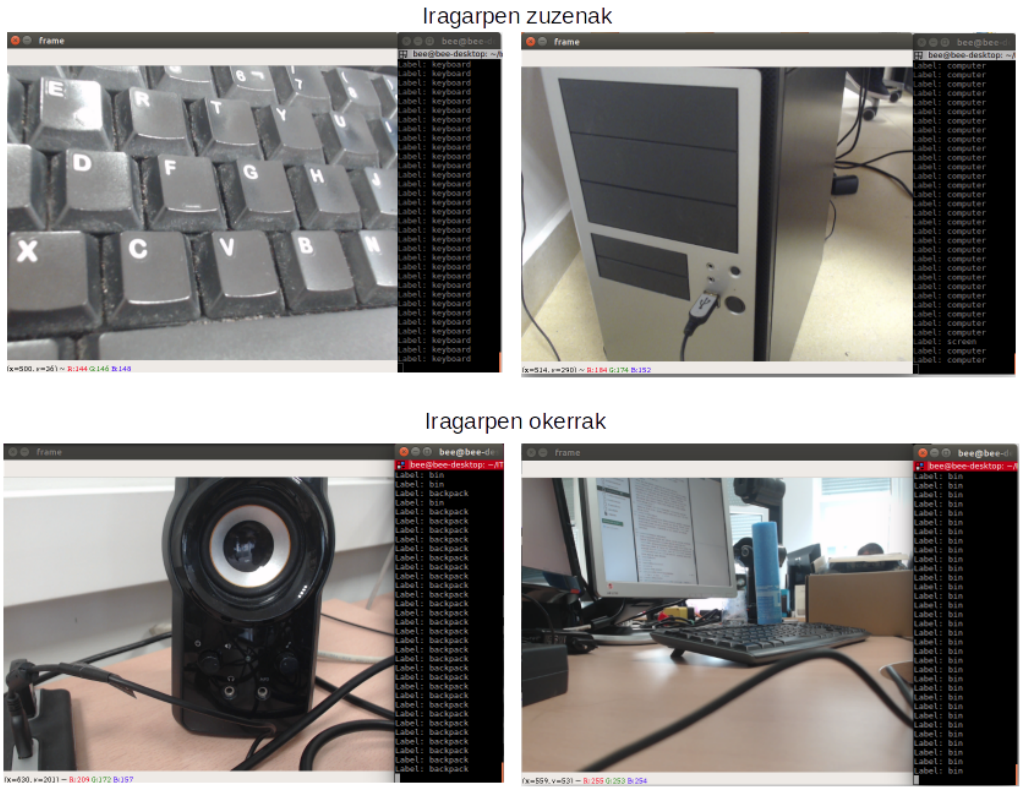
6.3.2 Web-kamera

Jarraian, web-kamera erabiliz denbora errealean egindako iragarpenen adibide batzuk azalduko dira, zuzenak zein okerrak. Kontuan izanik erabilitako eredua 6.3 atalean azaldutakoa dela, **SGD** optimizatzailearekin, iragarpenak ez dira zuzenak izan adibide guztietan, 6.3 irudian ikusten denez.

6.3 irudiko adibideetan, web-kameratik jasotako irudiaren etiketa komando-lerroan inprimatzen da uneoro. Hortaz, ordenagailuko iragarpena zuzena izan arren, momentu batean

pantaila (*screen*) iragatzen dela antzeman daiteke. Irudi okerrei dagokienez, bozgoragailua hasieran zakarrontzia (*bin*) eta gero motxila (*backpack*) dela iragatzen da eta idazmahaiak, zakarrontzia (*bin*) etiketa jasotzen du.

Bukatzeko, aipatzekoa da probak egiterakoan ikusi dela, objektuen iragarpena guztiz aldatu daitekeela, kamararen posizioa oso gutxi aldatu arren.

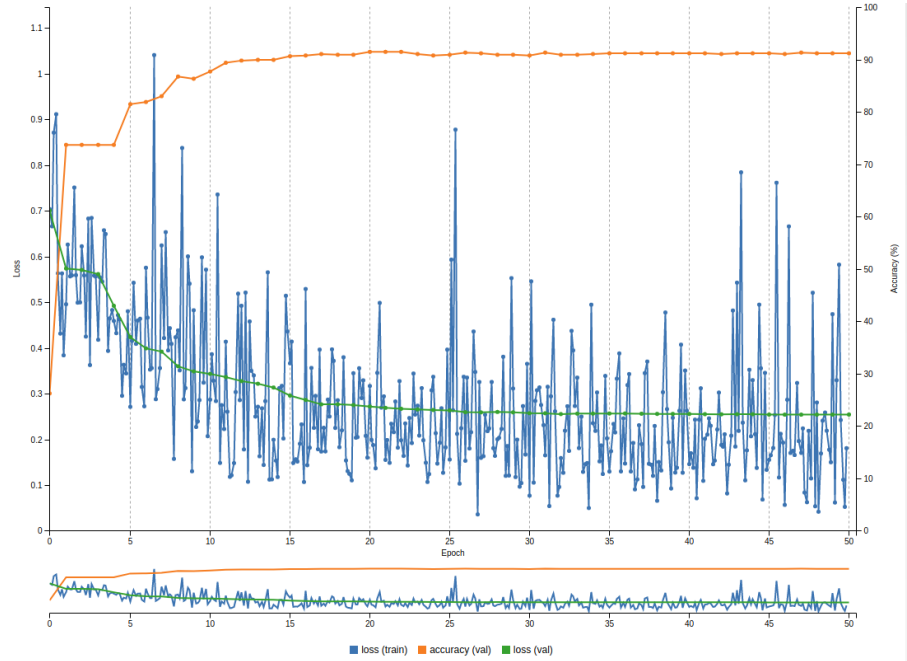


6.3 Irudia: Web-kamararen bidez egindako iragarpenak

6.4 Berrentrenamendua: korridoreko irudiak

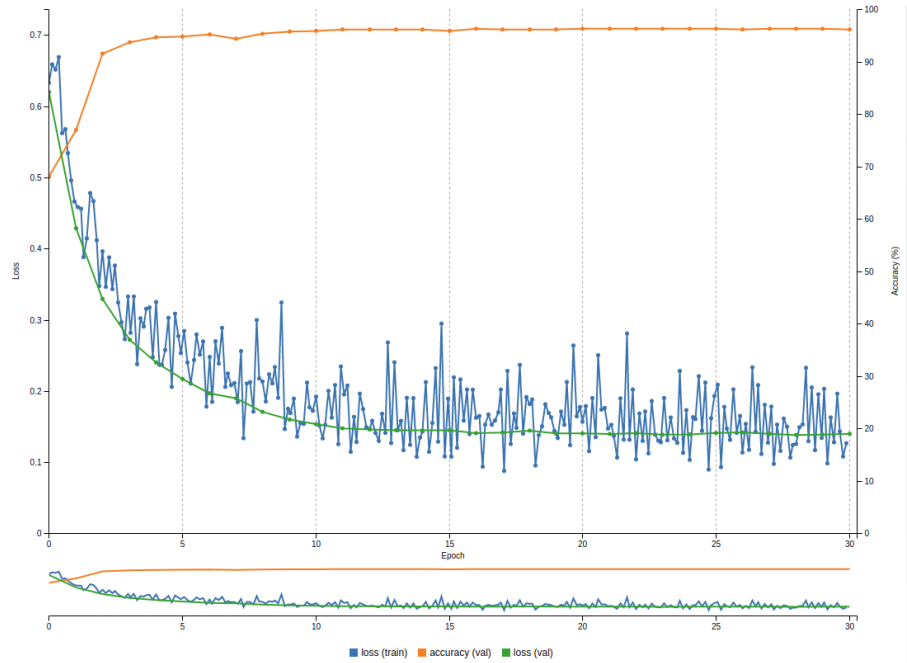
Honako hontan, korridoreko irudiekin lortutako emaitzak aztertuko dira. Lehenik eta behin, korridoreko irudiak eta garatutako sare konboluzionala erabiliko dira eredu berri bat entrenatzeko. Eredu berri honen emaitzak, 6.4 grafikoan ikusi daitezke.

6.4 grafikoan ikusten denez, gutxi gorabehera %91ko asmatze-tasa lortzen da. Kasu honetan, bi klase besterik ez daudenez, garatutako sare konboluzionalak portaera egokiagoa du.



6.4 Irudia: Korridoreko irudiak erabiliz sortutako eredua

Beste alde batetik, aurreko datu-basearekin entrenatutako eredua erabiliko da, korridoreko irudi berri hauekin entrenatzen jarraituz eredu berri bat lortzeko. 6.5 grafikoan eredu honi dagokion ikasketa prozesua ikusi daiteke.



6.5 Irudia: Aurreko eredua eta korridoreko irudiak erabiliz lortutako eredu berria

6.5 grafikoari erreparatuz, emaitzak hobetu direla ikusi daiteke, %96ko asmatze-tasa lortuz. 6.4 grafikoarekin konparatuz, berrentrenamendua erabiliz, azkarrago lortzen da asmatze-tasa hobea, hau da, epoka gutxiago behar ditu eredu honek emaitza hobeak lortzeko.

Gainera, %96ko emaitza lortu nahi den helburuarentzako nahikoa da. Datu-base hau erabiltzearen arrazoia honako hau da: etorkizunean korridoretik robot mugikor bat ibiltzen jarri nahi da eta irakasleen bulegoak antzemateko ahalmena lortu nahi da. Hortaz, robotak uneoro irudiak aztertuko ditu altuera berdin batera eta orden jakin batean. Beraz, klase positiboko irudiak (karteldunak) jarraian joango direnez, batean okertzen bada ere, jarraian hainbat iragarpen positibo ematerakoan, positibotzat hartuko da. Zehazki, azken 10 irudien erregistroa gorde eta hauen artean gehien errepikatzen den klasea iragarriko litzateke.

Hori dela eta, lortutako emaitzekin jokaera onargarria lortzea espero da.

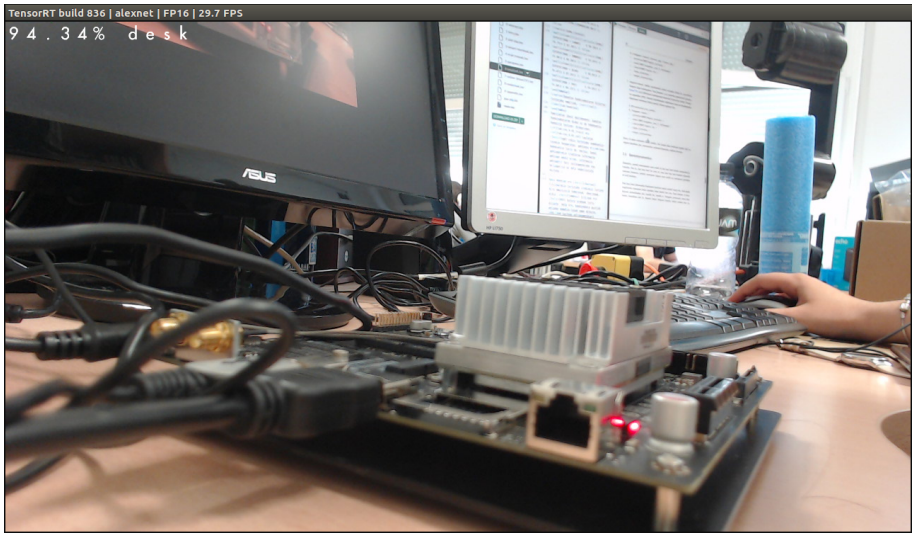
6.5 Jetson TX1

Bukatzeko, Jetson TX1 plakarekin egindako probekin lortutako emaitzak azalduko dira. 5.5 atalean azaldutako bi aukera desberdinak landuko dira. Horretarako, hiru eredu desberdin kargatu dira moduluan: sare konboluzionala + 14 klaseko datu-basea erabiliz lortutakoa, sare konboluzionala + korridoreko irudiak erabiliz lortutakoa eta aurretik 14 klaseko datu-basearekin entrenatutako sare konboluzionala + korridoreko irudiak erabiliz lortutakoa.

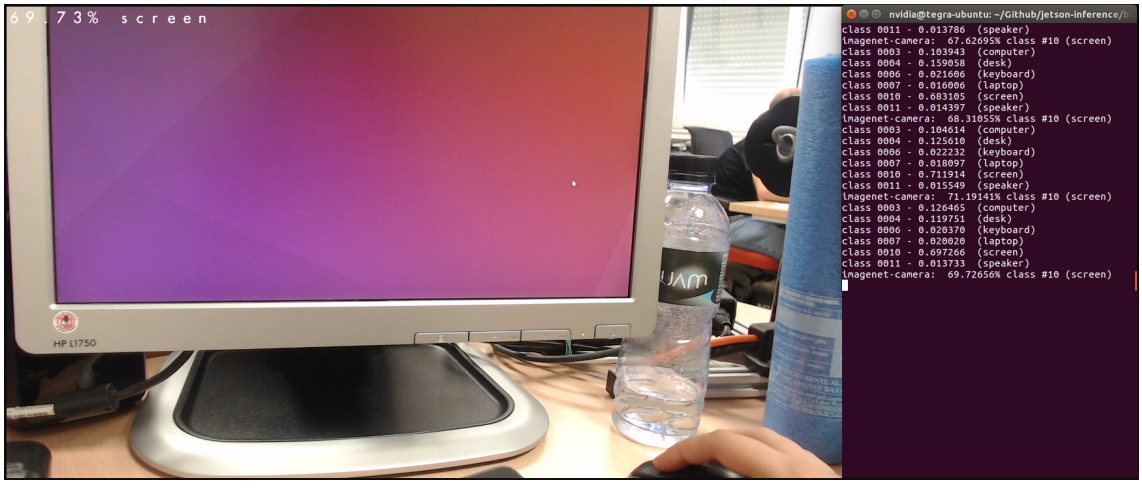
Denbora errealeko probak egiteko, *./imagenet-camera*, 14 klaseko datu-basea erabiliko da. Kasu honetan, ez dira klase horien irudiak gorde ondoren *test* moduan erabiltzeko eta beraz, hauek Jetson TX1 modulura lotutako kamaratik lortu eta bertan kargatutako eredua erabiliz klasifikatuko dira.

6.6 irudiko adibideetako bakoitzean, leihoaren goikaldean ezkerreko aldean, ereduak iragarritako klasea ikusi daiteke, klase hori izateko probabilitatearekin batera. Azkeneko adibidean, terminalaren irudia ere gehitu da, bertan sei klase desberdin izateko probabilitateak agertzen dira eta horietatik probabilitate handiena duena iragartzen da.

(a) Erlojua



(b) Idazmahaia

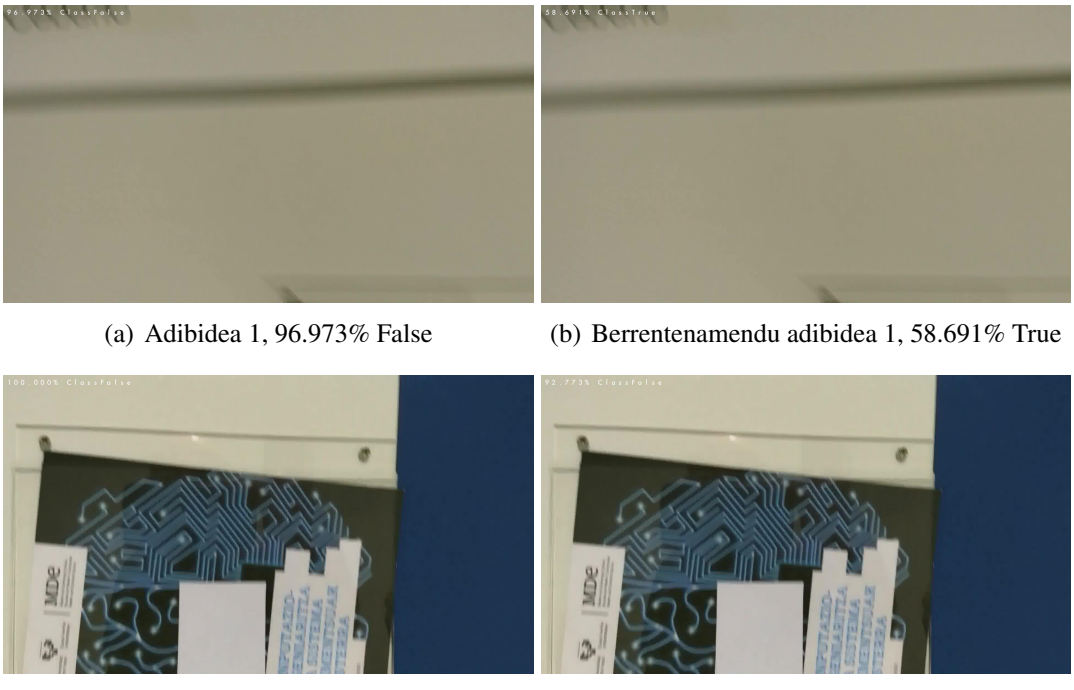


(c) Pantaila

6.6 Irudia: Irudien klasifikazioa *Jetson TX1* erabiliz

Bestalde, korridoreko hainbat irudi *test* moduan erabiltzeko gorde direnez, *./imagenet-console* erabiliz gordetako irudi hauen iragarpena egingo da. Korridoreko irudiak erabiliz sortutako bi ereduak erabiltzea erabaki da hauen emaitzak benetazko kasuetan konparatu ahal izateko.

6.7 irudian, kartelik gabeko klaseko adibideak agertzen dira. Lehenengo adibidean ikusi daiteke nola berrentrenatutako ereduak karteldun klasea iragartzen duen, kartelaren izkina desberdintzen duelako. Emaitzei erreparatuz, esan daiteke lehenengo ereduak, berrentrenatu gabea, orokorrean ezezko klasea (kartelik gabekoa) probabilitate handiagoarekin iragartzen duela.

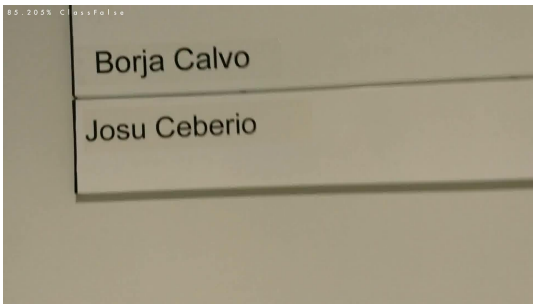


(a) Adibidea 1, 96.973% False (b) Berrentenamendu adibidea 1, 58.691% True

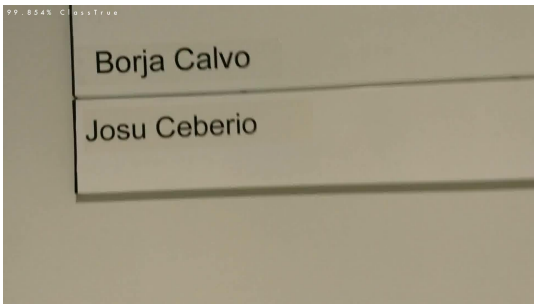
(c) Adibidea 2, 100.00% False (d) Berrentenamendu adibidea 2, 92.773% False

6.7 Irudia: Irudien klasifikazioa *Jetson TX1* erabiliz, **false** klasea

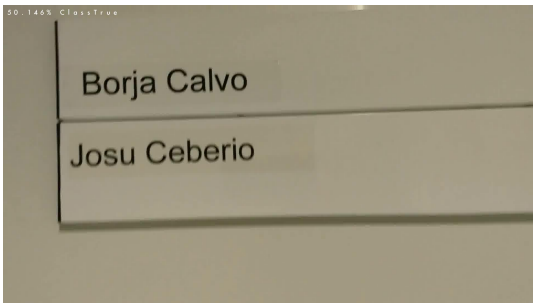
6.8 irudian aldiz, karteldun klaseko adibideak aurkitzen dira. Kasu honetan, zehatzagoa da berrentrenatutako eredua, iragarpen guztiak modu egokian egin ditu eta. Berrentrenamendu gabeko ereduak aldiz, kartelik gabeko klasea iragarri du adibideetako batean eta bere probabilitatea baxuagoa da beste adibidean ere.



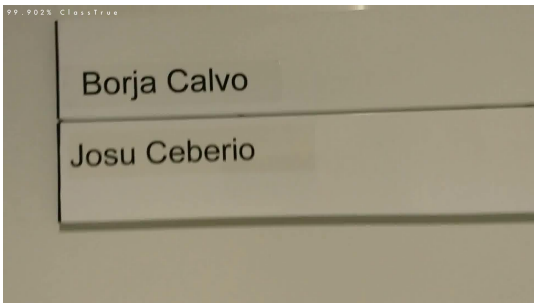
(a) Adibidea 1, 85.205% False



(b) Berrentenamendu adibidea 1, 99.854% True



(c) Adibidea 2, 50.156% True



(d) Berrentenamendu adibidea 2, 99.902% True

6.8 Irudia: Irudien klasifikazioa *Jetson TX1* erabiliz, **true** klasea

7. KAPITULUA

Ondorioak

Atal honetan, proiektua garatu ondoren ateratako ondorioak azalduko dira, bai landutako kontzeptuen eta lortutako emaitzen ondorioak, bai pentsatutako ondorio pertsonalak ere. Gainera, proiektuarekin zerikusia izan dezaketen etorkizunerako lanak ere aurkeztuko dira.

7.1 Proiektuaren ondorioak

7.1.1 Emaitza esperimentalak

Proiektuari dagokionez, emaitza esperimentalak ez dira esperotakoak izan. Arazo honen arrazoi nagusienetako bat honakoa izan daiteke: erabilitako irudien artean, irudi batzuetan agertzen diren objektuen identifikazioa oso zaila da, adibidez 7.1 irudia.

7.1 Irudia: *Desk* klaseko irudi baten adibidea

Hau da, irudi batzuetan aukeratutako hainbat klaseko objektuak agertzen dira. 7.1 irudiko klasea idazmahia (*desk*) da. Hala ere, ordenagailua (*computer*), pantaila (*screen*) eta teklatua (*keyboard*) agertzen dira ere.

Gainera, beste irudi batzuei klaseetako bat esleitzea ia ezinezkoa da, baita gizakientzako ere, 7.2 irudian ikusi daitekeen moduan. Adibideetan agertzen diren irudiak, mugikorra (*phone*) eta idazmahia (*desk*) klaseetakoak dira. Hala ere, pertsona bati irudian ikusten duena deskribatzeko eskatzerako orduan, klase horiek nekez esleituko lituzke.

(a) *Phone* (b) *Desk*

7.2 Irudia: Klasifikatzeko zailak diren irudien adibideak

Horrez gain, filtro desberdinak erabiliz lortutako irudien kanalen konbinaketa eginez emaitzak hobetzea espero zen, baina emaitzetan ikusi denez, kanalak gehituz eskuratzen den informazio hori ez da lagungarria izan irudien klasifikaziorako.

Hala ere, *deep learning* metodoko sare neuronalek *machine learning*-eko emaitzak hobetzen dituzte, irudien sailkapenerako eredu hobeak lortuz.

Bestalde, berrentrenamendua lantzerako orduan, emaitzak hobetu egiten direla frogatu da eta atal horretan lortutako asmatze-tasa onargarria dela ondorioztatu daiteke. Azaldutako moduan, korridoreko irudiak robot mugikor baten bitartez lortzea espero da, irudiak altura jakin batean jasoko dira eta beraz, ordena bat eramango dute. Irudiak ordena bat jarraituko dutenez, hainbat klase positibo jarraian iragarriz, nahiz eta baten bat negatibo eman, positibotzat hartuko da. 7.3 irudian segida honen adibide bat ikusi daiteke.

7.3 Irudia: Korridoreko segida baten adibidea

7.1.2 Emaitza teknikoak

Bestetik, emaitza teknikoak aztertu daitezke. Nahiz eta emaitza esperimentalak oso egokiak ez izan, proiektuaren ondorioz emaitza tekniko aproposak lortu dira. Hau da, proiektu honen garapenari esker, software eta hardware teknologia ugarirekin lan egin da.

Sare neuronal konboluzionala sortuz, programazio lengoaia lantzeaz gain, *deep learning* liburutegi eta lan-ingurune desberdinak erabiltzera ikasi da. Horrez gain, DIGITS softwarearen instalazioa egin eta honek eskaintzen dituen baliabideak aztertu eta landu dira. Aipatzekoa da ere *machine learning*-eko sailkatzaileak garatzeko liburutegiekin lan egin dela.

Jetson TX1 moduluaren instalazioaren bidez, NVIDIA-k eskainitako hardwarea erbiltzeko ahalmena lortu da, honen bidez proba desberdinak egiteko gaitasuna lortuz. Beraz, proiektu honen bidez hardware berri batekin aritzeko aukera lortu da.

Halaber, *machine learning* eta *deep learning* metodo desberdinak ikastea derrigorrezkoa izan da eta beraz, proiektuaren atal handi bat kontzeptu teorikoak ikastera bideratu da. Ulertzeko zaila gertatzen den gaia denez, proiektu honen ondorioz lortutako ikasketa oso baliagarria suertatu da.

Hortaz, emaitza esperimentalak espero bezain onak ez izan arren, proiektuaren garapenaren bidez lortutako ezagutza handia izan dela ondorioztatu daiteke.

7.2 Etorkizunerako lana

Etorkizunean, aurretik aipatutako moduan, korridoreko irudiekin lan egiten jarraitzea da helburua. Kartela dagoen ala ez erabakitzeaz gain, irakasle bakoitzaren bulegoa desberdintzea lortu nahi da. Hau da, karteletan agertzen diren izenak ezagutzeko ahalmena es-

kuratu nahi da sare konboluzionalak erabiliz.

Gainera, hamalau klase desberdineko datu-basearekin lortutako emaitzak hobetzekotan, neurona-sare konboluzionalaren arkitekturan aldaketak eginez proba gehiago egin daitezke, edota sareko hiperparametroei balio desberdinak emanez.

7.3 Ondorio pertsonalak

Niri dagokidanez, garatutako proiektu honi esker, irudien prozesamendurako gaur egun gehien erabiltzen diren metodoetako batzuk ulertu ahal izan ditut. Nahiz eta gai hau oso zabala izan, hasierako pausuak ematea lortu dut.

Gainera, proiektuan zehar azaldutako problemak konpontzea ere ikasketa handia da, geroz eta independenteago bihurtuz eta informazioa nire kabuz bilatuz. Proiektua aurrera eramateko ere, egutegi bat jarraitzeko beharra ikusi dut, betiere ataza guztiak adostutako uneetan bukatuz proiektua data egokian entregatu ahal izateko. Hau da, maila honetako proiektu batek arduratsu izatera behartzen zaitu eta hau etorkizunean oso lagungarria izango da.

Bukatzeko, RSAIT taldeak eskainitako teknologiari esker, ezagutzen ez nituen eta asko erabiltzen diren tresnak ezagutzea posiblea izan dut. Eskerrak eman nahi ditut taldeko kideei eskainitako laguntza guztiarengatik, mila esker.

Bibliografia

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*, page 198. Springer, 2006.

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[3] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[4] R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, pages 947–951, 2000.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385:770–778, 2015.

[6] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[7] D. H. Hubel and T. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[8] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, abs/1412.6980:1–13, 2015.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *In Advances in neural information processing systems*, pages 1097–1105, 2012.

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. pages 2278–2324, 1998.

[11] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[12] S. Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.