Rīgas 64. vidusskola

Cilvēka darbības atpazīšana, izmantojot ilgās īstermiņa atmiņas modeli, salīdzinot ar konvolūcijas neirona tīkla modeli.

Zinātniski pētnieciskais darbs datorzinātņu un informātikas nozarē

Darba autors: Andris Andersons
Darba vadītājs: Edvards Bukovskis

Rīga, 2024

Anotācija

Cilvēka darbību atpazīšana izmantojot ilgās īstermiņa atmiņas modeli salīdzinot ar konvolūcijas neirona tīkla modeli. Darbu izstrādājis Andris Andersons, darbu vadījis Rīgas 64.vidusskolas programmēšanas skolotājs Edvards Bukovskis.

Darbā apskatīts teorētiskais pamats ilgās īstermiņa atmiņas modelim, konvolūcijas neironu tīkla modelis un to abu implementācija, kas balstās uz jau izveidotiem modeļiem, cilvēka darbības atpazīšanai, kā arī modeļu veiktspējas novērtējums, izmantojot daudzveidīgu žestu datu kopu.

Atslēgvārdi: LSTM, CNN, cilvēka darbības atpazīšana, ilgās īstermiņa atmiņas modelis, konvolūcijas neirona tīkla modelis.

Abstract

Human action recognition using a long-short-term memory model compared to a convolutional neural network. The work was developed by Andris Andersons, supervised by Edvard Bukovskis, programming teacher at Riga Secondary School 64.

This paper examines the theoretical basis for human action recognition, a long short-term memory model and its implementation based on an existing model for human action recognition, as well as performance evaluation of the model using a diverse gesture dataset.

Key words: LSTM, CNN, human action recognition, long short term memory model, convolutional neural network model.

Satura rādītājs

Ievads

Ilgā īstermiņa atmiņas modeļa (LSTM) radīšana veicināja ievērojamu progresu secīgu datu analīzē. Sākotnēji LSTM modelis tika ieviests dabiskās valodas apstrādes jomā, kur modelis guva popularitāti savas izcilās kontekstuālās informācijas uztveres un izmantošanas dēļ. LSTM tīkla atkārtojošā struktūra ļauj veikt uzdevumus ar iekšējā laika atkarībām, kas padara to ļoti pieprasītu. LSTM tīkla pielietošana zīmju valodas atpazīšanā bija loģisks solis uz priekšu, lai papildinātu modeli, padarītu to precīzāku un vairāk pieejamu. Izmantojot LSTM tīklu, lai atpazītu zīmju valodu, tiktu radīta iespēja novērst komunikācijas šķēršļus sabiedrībā, piemēram, saziņā starp nedzirdīgajiem un dzirdīgajiem.

Darba tēma: Ilgā īstermiņa atmiņas modeļa (LSTM) pielietojums cilvēka darbības atpazīšanā.

Darba mērķis: Objektīvi izvērtēt ilgā īstermiņa atmiņas modeļa (LSTM) efektivitāti un veiktspēju dažādu žestu atpazīšanā, salīdzinot to ar konvolūcijas neirona tīkla modeli (CNN).

Pētnieciskais jautājums: Kādas ir ilgā īstermiņa atmiņas modeļa (LSTM) veiktspējas priekšrocības, to salīdzinot ar konvolūcijas neirona tīkla modeli (CNN) cilvēka darbības atpazīšanā?

Darba uzdevumi:

1. Izpētīt teorētiskos pamatus ilgā īstermiņa atmiņas modelim (LSTM) un konvolūcijas neirona tīkla modelim (CNN).
2. Implementēt jau iepriekš izveidotu ilgā īstermiņa atmiņas modeli (LSTM), kas pielāgots cilvēka darbības atpazīšanai.
3. Izvērtēt ilgās īstermiņa atmiņas modeļa veiktspēju, izmantojot dažāda veida žestus.
4. Implementēt jau iepriekš izveidotu konvolūcijas neirona tīkla modeli, kas pielāgots cilvēka darbības atpazīšanai.
5. Izvērtēt konvolūcijas neirona tīkla modeļa veiktspēju, izmantojot dažāda veida žestus.

Metodes:

1. Literatūras apskats, lai iepazītos ar ilgā īstermiņa atmiņas modeļa (LSTM) un konvolūcijas neirona tīkla modeļa (CNN) teorētisko pamatu, kas iekļauj modeļu datu apmācības procesu.

2. Salīdzinošā metode, lai izvērtētu jau izgatavota ilgā īstermiņa atmiņas modeļa priekšrocības attiecībā pret jau izgatavotu konvolūcijas neironu tīkla modeli cilvēka darbību atpazīšanai.

3. *Python* programmēšanas valoda un nepieciešamās bibliotēkas, lai varētu izvērtēt LSTM priekšrocības un salīdzināt to ar CNN modeli.

1. Literatūras apskats

1.1. Ilgās īstermiņa atmiņas neironu tīkls (LSTM)

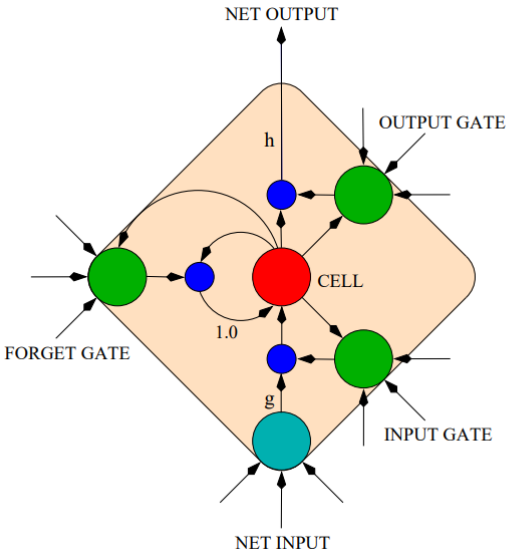
Ilgās īstermiņa atmiņas modelis (*long short term memory* jeb LSTM) ir rekurento neironu tīklu (RNN) paveids, kam raksturīga rekurenta struktūra, kas ļauj veikt uzdevumus, kur dati vai notikumi ir atkarīgi no laika un to secības. Atmiņas modeli 1997.gadā ieviesa vācu datorzinātnieki Seps Hohreiters un Jurgens Šmīdhūbers (Hochreiter and Schmidhuber, 1997). Atšķirībā no rekurento neironu tīkla, ilgās īstermiņa atmiņas modelis izmanto atgriezeniskās saites savienojumus, kā arī tas izvairās no ilgtermiņa atkarības problēmas, kas piemīt parastam RNN (Staudemeyer and Morris, 2019).

LSTM tika izstrādāts, analizējot fundamentālu mašīnmācīšanās problēmu – gradientu izzušana un uzsprāgšana (Schmidhuber, 2022). Pateicoties tam, ka LSTM palīdz risināt gradientu izzušanu un uzsprāgšanu, to sāka lietot uzdevumos saistībā ar rokraksta un runas atpazīšanu (Graves et al., 2009; Sak, Senior and Beaufays, 2014).

LSTM pēc savas struktūras tika uzbūvēts, lai varētu uzturēt informāciju lielos laika intervālos, kamēr vien šī informācija ir nepieciešama. Tā sastāv no šūnas, ievades vārtiem, izvades vārtiem un aizmiršanas vārtiem (Staudemeyer and Morris, 2019).

LSTM centrā ir atmiņas šūna, kas spēj visu doto informāciju atcerēties. Tā uztur informāciju par visām ievades vērtībām, kuras novērotas līdz noteiktam solim. Atmiņas šūnai ir tādi paši ievades un izvades vārti, kā rekurentiem neirona tīklam (RNN), bet LSTM ir papildus vārti, kas kontrolē šo informāciju. Ievades un izvades vārti kontrolē informācijas ievadi un izvadi atmiņas šūnā (Chen, 2016). LSTM vārtu aktivizēšanai izmanto loģistikas funkciju - tā ir matemātiska funkcija ar raksturīgu S-veida ieliekumu (Staudemeyer and Morris, 2019).

Aizmiršanas vārti var atiestatīt atmiņu, izmantojot jebkādu matemātisku funkciju, kurai raksturīgs S-veida ieliekums. Aizmiršanas vārti izveido lineāru funkciju no ievadītajām vērtībām un tam seko loģistikas funkcija, kas nozīmē, ka izejošā vērtība būs no 0 līdz 1. Aizmiršanas vārti izvērtē, kādu informāciju paturēt no iepriekšējā stāvokļa, salīdzinot to ar nākamā stāvokļa ievadīto vērtību. Ja vērtība ir 0, tad aizmiršanas vārti šo informāciju aizmirst jeb dzēš, bet, ja vērtību noapaļojot sanāk 1, tad informāciju saglabā. (Staudemeyer and Morris, 2019).



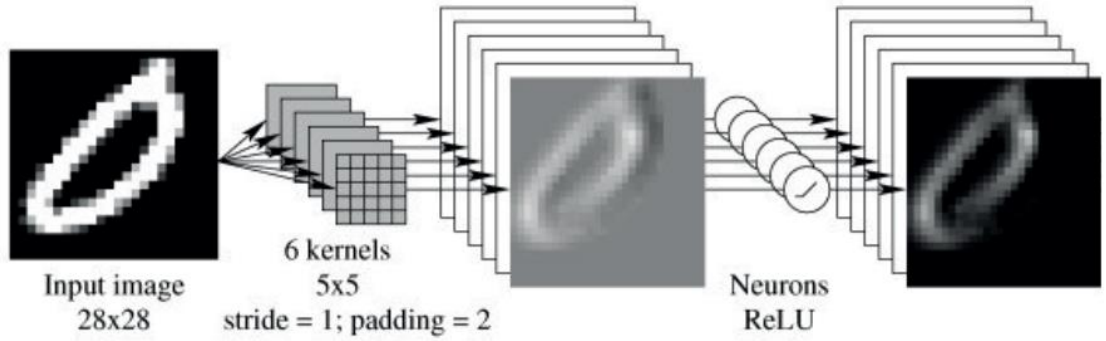
1.attēls. LSTM atmiņas bloks ar vienu šūnu (Graves et al., 2009).

1.2. Konvolūcijas neirona tīkls (CNN)

Konvolūcijas neirona tīkls (*convolutional neural network* jeb CNN) ir dziļās apmācības modelis, kas satur konvolūcijas slāņus. Tā galvenā funkcija ir atpazīt noteiktus elementus, kuri atrodas bildē, piemēram, noteiktu krāsu punkti un kontūras (University of Helsinki and MinnaLearn, 2018). CNN nosaka svarīgumu šiem elementiem, izmantojot svarus un novirzes, lai varētu tos atšķirt (Saha, 2018). Svari kontrolē to, cik liela ietekme būs ievadei uz izvadi, un novirze kontrolē to, cik izteikti kāds elements ietekmēs gala rezultātu.

Apmācīt parastu neirona tīklu, kas neizmanto konvolūcijas slāņus, bet vadās pēc ievadītā attēla pikseļiem, būtu sarežģīti, jo, ja attēlā elementi ir sagrozīti vai atrodas citā attēla malā, tad elementa atpazīšanas precizitāte būtu salīdzinoši zemāka. Lai iemācītu atpazīt elementus neatkarīgi no to atrašanās vietas bildē, būtu nepieciešams liels mācību datu apjoms, jo tīkls atpazīst elementu tikai tādās pozīcijās, kurās elements ticis atpazīts mācību datos (University of Helsinki and MinnaLearn, 2018). Piemēram, attēls, kur ābols atrodas apakšējā kreisajā stūri, tiks atpazīts tad, ja mācību datos ir bijis attēls ar ābolu apakšējā kreisajā stūrī, bet, ja mācību datos arī tiktu iekļauts ābols, kas atrastos augšējā labā stūri, tad attēlā, kur atrodas ābols gan apakšējā kreisajā stūrī, gan augšējā labā stūrī, tiktu atpazīts. CNN piemīt spēja elementu atpazīt neatkarīgi no tā, kādā pozīcijā elements ticis atpazīts mācību datos (University of Helsinki and MinnaLearn, 2018).

CNN izmanto trīs slāņus – konvolūcijas slāni, apvienošanas slāni un pilnībā savienotu slāni. CNN var sastāvēt no vairākiem katra tipa slāņiem, atkarībā no nepieciešamības. Konvolūcijas slānis ir kā pamatelements CNN darbībai. Konvolūcijas slāņa parametrus veido kodoli (angļu valodā *kernels)*. Kodols ir virkne ar apgūstamiem filtriem, kuri ir nelieli platumā un augstumā, bet sniedzas cauri visam ievades datu apjomam. Kad dati nonāk konvolūcijas slānī, slānis sasaista katru filtru cauri ievades datu apjoma platumam un augstumam. Pēc tā tiek aprēķināts punktu reizinājums starp filtru ierakstiem un ievades datiem jebkurā pozīcijā, kam seko aktivizācijas funkcija (sigmoid, ReLU) un iegūto izvadi sauc par aktivizācijas mapi. Šīs aktivizācijas mapes tiek sakopotas dziļuma dimensijā, lai iegūtu pilno izvades datu apjomu (Bezdan and Bačanin Džakula, 2019).

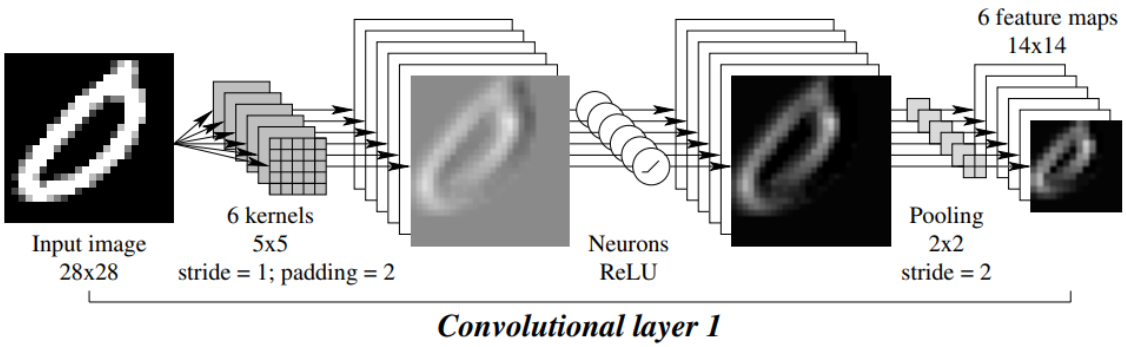


2. attēls. Piemērs konvolūcijas tīklam, kam seko aktivizācijas funkcija (Bezdan and Bačanin Džakula, 2019).

Izvades datu apjoms ir atkarīgs no trīs hiperparametriem – dziļums (*depth*), solis (*stride*) un polsterējums (*padding*) (O'Shea and Nash, 2015; Bezdan and Bačanin Džakula, 2019). Izvades datu apjoma dziļumu raksturo filtru skaits, kuri tiek izmantoti konvolūcijas slānī. Katrs filtrs mācās kaut ko citu no dotās ievades, piemēram, krāsas, malas un tekstūras (Bezdan and

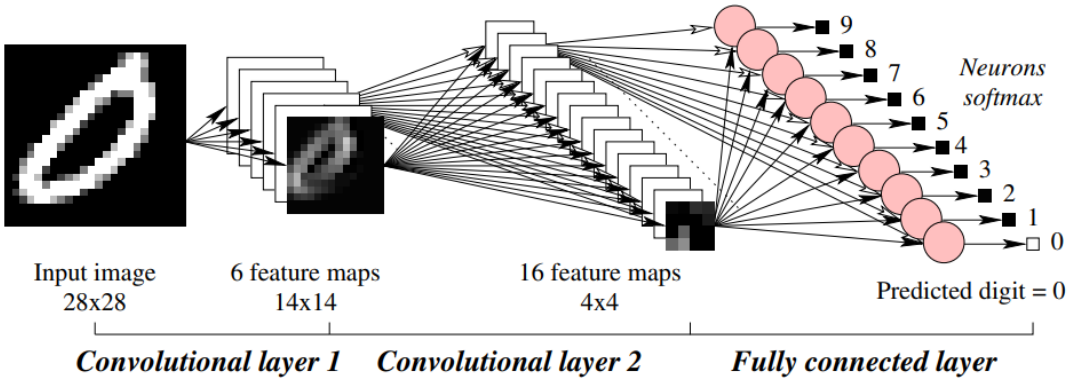
Bačanin Džakula, 2019). Solis norāda to, cik liela būs pikseļu kustība kodolam. Ja soli samazina, izvades datu apjoms palielinās, bet tiek apgūtas vairākas elementa pazīmes. Turpretī, ja soli palielina, tad tiek apgūts mazāk elementa pazīmju un iegūts mazāks izvades datu apjoms. Arhitektūras projektētājs ir atbildīgs par to, ka kodols simetriski slīd pāri ievadei, kad tiek implementēts CNN (Wang et al., 2020). Polsterējums nodrošina kontroli pār izejas datu apjoma lielumu. Tas ir nepieciešams, ja kodols pārsniedz aktivizācijas mapi. Polsterējums saglabā datus pie robežām, kas palīdz arī saglabāt ievades telpisko izmēru (Dumoulin and Visin, 2016). Viena no pazīstamākajām polsterējuma metodēm ir nulles polsterējums. Tas simetriski ievieto nulles gar ievades malām, kas palīdz novērst ievades datu zudumu un nokontrolēt izvades datu apjomu efektīvā veidā (Wang et al., 2020).

Apvienošanas slānis pakāpeniski samazina tīkla telpisko apjomu, kas samazina parametru apjomu un kopējo tīkla skaitļošanas sarežģītības apjomu. Apvienošanas slānis darbojas pāri katrai aktivizācijas kartei un izmantojot funkciju "MAX", mēro tās dimensiju (O'Shea and Nash, 2015). Apvienošanas slānis, kas izmanto funkciju "MAX" neironu tīklos parādās kā *"Max-pooling"*, un tās operācijai arhitektūras laikā ir nepieciešams izvēlēties kodola un soļa izmēru. Kad ir veikta izvēle, tad operācija liek kodolam slīdēt ar noteikto soli pāri ievadei, atlasot lielāko vērtību katrā kodola šķēlumā no ievades, lai iegūtu izejas vērtību izvadei (Wang et al., 2020).



3. attēls. Konvolūcijas slānis, kam seko apvienošanas slānis (Bezdan and Bačanin Džakula, 2019).

Pilnībā savienots slānis sastāv no neironiem, kuri ir tieši savienoti ar neironiem blakus esošajos slāņos, un tie nav savienoti ne ar vienu neironu to iekšienē (O'Shea and Nash, 2015). Seklā CNN modelī pēdējā konvolūcijas slāņa pazīmes atbilst daļai no ievades attēla, jo uztveres lauks neaptver visu attēla dimensiju. Šādā gadījumā ir nepieciešams pilnībā savienots slānis (Basha et al., 2020).



4. attēls. Divi konvolūcijas slāņi, kam seko pilnībā savienots slānis (Couchot et al., 2016).

2. Cilvēka darbības atpazīšana, izmantojot ilgā īstermiņa atmiņas modeli un konvolūcijas neirona tīkla modeli

Šī zinātniski pētnieciskā darba ietvaros salīdzinātie modeļi ir iepriekš izgatavoti un ir publiski pieejami (skatīt 1., 2. pielikumu).

Cilvēka darbības atpazīšanas process norisināsies ar jau izgatavotiem modeļiem, kuri ir pielāgoti cilvēka darbības atpazīšanai, bet nepieciešams tos apmācīt. Šie modeļi ir izstrādāti *Python* programmēšanas valodā.

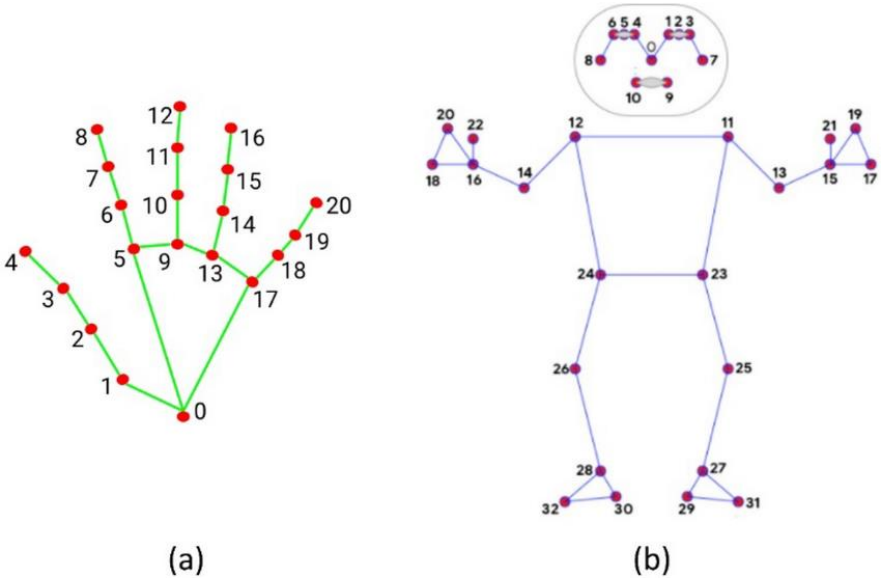
LSTM un CNN modeļiem nepieciešams ir nodrošināt bibliotēkas (*dependencies*), lai būtu balsts neirona tīkla apmācībai. Šim nolūkam tiks izmantots *TensorFlow*, jo tas nodrošina rīku kopumu modeļu izveidei un to apmācībai. Lai iegūtu vizuālo ievadi, tiks izmantots *OpenCV*, kas nodrošina reālā laika datora redzi, šajā gadījumā – kameru, lai varētu nodrošināt attēlus, kuri tiks izmantoti apmācībai. Datu ieguvei tiks izmantots *MediaPipe*, lai varētu atlasīt atslēgpunktus no vizuālās ievades. Novērtēšanas rādītājiem tiks izmantots *Scikit-learn* un *Matplotlib*, lai palīdzētu ar vizualizāciju. *NumPy*, lai varētu strādāt ar masīviem un strukturēt dažādu datu kopas. Bibliotēka *os* nepieciešama, lai varētu strādāt ar failu adresēm un *time* nepieciešama, lai nodrošinātu pauzi starp vizuālo datu ieguvi.

3. Ilgās īstermiņa atmiņas modeļa apmācība

3.1. Datu ieguves metode

Izmantojot *OpenCV* bibliotēku, tiks izmantota kamera, lai iegūtu attēlus, kuri ies cauri otram slānim. Otrs slānis būs atbildīgs par atslēgdatu ieguvi, kur tiks izmantota bibliotēka *MediaPipe*. Šajā slānī tiek izveidoti divi mainīgie, kur viens mainīgais iekļaus *MediaPipe Holistic*, lai varētu izsekot cilvēka kustībai ar vizuāli parādītiem punktiem. Otrs mainīgais iekļaus *MediaPipe drawing utilities*, kas tiks izmantots, lai atvieglotu punktu vizualizāciju uz cilvēka ķermeņa daļām (skat. 5.attēlu). Atslēgdati tiks iegūti no otrajā slānī iegūtajiem punktiem. Izmantojot *MediaPipe* funkciju *landmark*, tiks noteiktas visu atsevišķo punktu *x*, *y*, un *z* koordinātes, kuras tiks savienotas masīvā, izmantojot bibliotēku *NumPy*. Jāpiemin, ka, ja netiek reģistrēti punkti kādā noteiktā brīdi, tad tiek izveidoti masīvi, kuri tiek aizpildīti ar nullēm, bet ir vienāda lieluma ar *x*, *y,* un *z* koordinātu masīviem. Izmantojot bibliotēkas *NumPy* funkciju *flatten*, šie atsevišķie masīvi tiks konvertēti vienā lielā masīvā, jo šāds formāts būs nepieciešams, kad šie dati tiks izmantoti LSTM neirona tīkla apmācībai. Kad masīvi ir

konvertēti, tad nākamais solis ir saglabāt šos datus, lai varētu iegūt atslēgpunktus. Atslēgpunkti tiks iegūti saglabājot noteiktu attēli, kur tiek reģistrētas punktu *x, y* un *z* koordinātes un saglabātas masīvā, bet tā kā cilvēka darbības atpazīšana notiek reālā laikā, ir nepieciešams video. Video ir sekvence ar mainīgām bildēm, tādēļ tiks izveidotas 30 sekvences, kuras satur 30 attēlus, kur katrs attēls tiks saglabāts kā masīvs.



5. attēls. Vizuāls attēlojums punktiem, kurus iegūst ar MediaPipe bibliotēku (Kim and Baek, 2023).

3.2. Kolekcionēšanas mapju uzstādīšana

Lai atpazītu cilvēka darbību izmantojot LSTM neirona tīklu, tiek izmantoti trīs žesti no zīmju valodas, tādēļ, izmantojot bibliotēku *os*, tiek izveidota failu adrese uz mapi *"Data"*, kur tiek izveidotas 3 mapes ar nosaukumiem *"hello", "thanks"* un *"iloveyou"*. Šīs mapes saturēs 30 sekvences, tāpēc katrā mapē tiek izveidotas sekvenču mapes, kurām secīgi tiek iedalīts numurs no 1 līdz 30. Katra mape saturēs 30 bildes, kuras būs saglabātas ar *NumPy* funkciju *np.array*, lai saglabātu attēlus kā datu masīvus.

3.3. Datu ievākšana modeļa trenēšanai un testēšanai

Izmantojot 3.1. nodaļā minēto datu ieguves metodi, tiks ievākti attēli, kuri tiks saglabāti masīvos, lai izmantotu tos modeļa apmācībai. Datu ieguve notiek, saglabājot 30 individuālus attēlus, kuras tiek saglabātas masīvos, attiecīgā sekvences mapē. Tā kā LSTM modelim ir jāatpazīst žests reālā laikā, tad nepieciešams uztvert veselu žesta kustību, tādēļ tiek iniciēta sekvence, kur ir jāveic noteiktā žesta darbība un tikmēr tiek saglabāti 30 attēli ar notverto kustību - šādi jāveic katra sekvence. Process atkārtojas līdz brīdim, kad sekvences ir aizpildītas ar nepieciešamajiem datu masīviem katram žestam.

3.4. Datu pirmapstrāde

Lai LSTM modelis spētu atpazīt cilvēka darbību, ir nepieciešams apstrādāt datus, tā, lai LSTM neironu tīkls atpazīst attiecību starp sekvenci un izveidotajām mapēm *"hello", "thanks"* un *"iloveyou"*. Lai to panāktu, ir jāizveido no mapju *"hello", "thanks"* un *"iloveyou"* sekvencēm viens kopīgs masīvs un mapēm ir jāiedod indeksi, lai tās kategorizētu. Izmantojot bibliotēkas *TensorFlow Keras utilities* funkciju *to_categoricial*, mapēm *"hello", "thanks"* un

"iloveyou" tiek piešķirti indeksi, kuras funkcionē kā etiķetes. Tiek izveidoti divi masīvi – sekvences un etiķetes. Izmantojot *for loop*, *NumPy* funkciju *np.load* un *os* bibliotēkas funkciju *os.path.join*, tiek secīgi iets cauri sekvencēm un katras sekvences datu masīvu apvieno sekvences masīvā, kurā ietilpst 90 atsevišķi sekvenču masīvi, kuri sastāv no 30 attēliem, kuri saglabāti kā masīvi, tikmēr etiķetes masīvam tiek piešķirta katra sekvence, kurai iziets cauri. Izmantojot *NumPy* funkciju *numpy.array* iespējams apskatīt masīva datus, kur sekvences masīvā atrodas 90 sekvences, 30 attēli un noteikts punktu skaits, kuri reģistrēti bildē un etiķetes masīvā atrodas 90 sekvences. Pirms LSTM modelis tiek apmācīts, ir nepieciešams pielietot *train-test_split* funkciju, kuru iegūst no bibliotēkas *scikit-learn*. Šī funkcija sadalīs datus masīvā divās daļās – *training set* un *testing set*. Izmantojot šo funkciju tiek sadalīts sekvences masīvs un etiķetes masīvs ar norādītu parametru *test_size* = 0.05. Ar *test_size* parametru var noteikt to, cik procentuāli no visiem datiem tiks izmantoti, lai pārbaudītu LSTM modeļa veiktspēju atpazīt neredzētus datus.

3.5 Apmācības process

Lai sāktu LSTM modeļa apmācību, nepieciešams izmantot *TensorFlow* bibliotēku, kur papildus tiek izsaukts *Keras* API (Lietojumprogrammas saskarne), lai nodrošinātu slāņus un modeļus. LSTM modelis sastāvēs no 3 LSTM slāņiem un 3 pilnībā savienotiem slāņiem. No bibliotēkas *TensorFlow.Keras.models* tiek importēts *sequential* modelis, kas nodrošina sekvencētu lineāru datu ievadi un izvadi caur noteiktiem slāņiem. No *TensorFlow.Keras.layers* tiek importēts *LSTM* slānis un pilnībā savienots slānis *Dense*. Lai izsekotu un grafiski attēlotu LSTM modeļa apmācību, no *TensorFlow.Keras.callbacks* importē *TensorBoard*.

Pirms LSTM modeļa apmācības uzsākšanas, nepieciešams sastādīt modeli. Tas tiek paveikts padodot trīs parametrus – *optimizer*, *loss*¸ un *metrics*. *Optimizer* parametrs ir atbildīgs par to, kā tiek atjaunināti modeļa svari, lai samazinātu izvēlēto *loss* parametru. *Loss* parametrs mēra starpību starp modeļa prognozēm un faktiskajām mērķa vērtībam. *Metrics* parametrs sniedz informāciju par modeļa veiktspēju. Zinātniski pētnieciskā darbā izmantotais LSTM modelis izmanto "*Adam*" kā *optimizer*, "*categorical_crossentropy*" kā *loss* un "*categorical_accuracy*" kā *metrics*.

Izmantojot *TensorFlow* bibliotēkas funkciju *model.fit*, argumentā tiek padots sekvences un etiķetes *training set* masīvs, kam seko *epoch* ar noteiktu vērtību, kas būs atbildīgs par to, ka LSTM modelis iziet cauri dotajam datu masīvam noteiktas reizes apmācības procesā, piemēram, ja *epoch* vērtība ir 2000, tad LSTM modelis ies cauri dotajam datu masīvam 2000 reizes. Lai izsekotu apmācības procesu, papildus jāpadod argumentā *callbacks*.

4. Konvolūcijas neirona tīkla modeļa apmācība

4.1. Datu iegūšana

Dati priekš CNN modeļa apmācības tiek iegūti no internetā atrastām bildēm, kurās redzams kāds no *"hello", "thanks"* un *"iloveyou "*žestiem. Attēli tiek lejupielādēti un saglabāti mapītē *"data"*, kura sastāv no 3 papildus mapēm *"hello", "thanks"* un *"iloveyou "*. Šajās mapēs atrodas 30 attēli ar attiecīgiem žestiem. Šādi tiek iegūta mācību datu apakškopa *batch*.

4.2. Datu pirmapstrāde

CNN modelis izmanto attēlus kā pievada vērtību. Katras bildes pikseļa vērtības diapazons sniedzas no 0 līdz 255, bet, samazinot šo diapazonu, lai vērtības sniegtos no 0 līdz 1, tiek iegūta ātrāka datu vispārināšana un augstāka precizitāte. *Batch* sastāv no bildēm un indeksiem un, lai iegūtu vēlamo diapazonu, kurā vērtības sniedzas no 0 līdz 1, bildes tiek dalītas ar 255. Lai to pārbaudītu, izmanto funkciju *batch[0].max()*. Ja maksimālā vērtība ir 1, tad ir izdevies iegūt vēlamo diapazonu. Tad seko datu sadalīšana, kur dati tiks sadalīti trijās daļās – *training data*, *validation data* un *testing data*. Tieši šim CNN modelim tiek iedalīti 70% datu priekš *training data*, 20% datu priekš *validation data* un 10% datu priekš *testing data*. *Training data* ir dati, kuri tiks izmantoti, lai apmācītu CNN modeli, *validation data* ir dati, kuri netiek izmantoti, lai apmācītu CNN modeli, bet gan novērotu modeļa veiktspēju apmācības laikā un pielāgotu parametrus, lai novērstu *overfitting*. *Testing data* ir dati, kuri netika izmantoti apmācībai un validēšanai, bet tiek izmantoti, lai pārbaudītu modeļa veiktspēju, kad saskaras ar jauniem datiem.

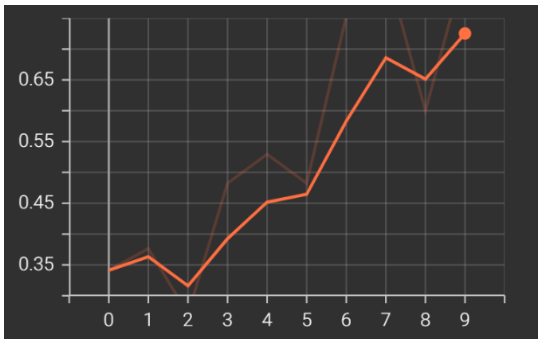
4.3. Apmācības process

No bibliotēkas *TensorFlow.keras.models* tiek importēts *sequential* modelis. No *TensorFlow.keras.layers* tiek importēts *Conv2D* konvolūcijas slānis, *MaxPooling2D* slānis, kas samazinās ievades apjoma telpiskos izmērus, tādējādi samazinot parametru un aprēķinu skaitu modelī. Tiek importēts arī pilnībā savienots slānis *dense* un *flatten* slānis, kas pārveidos ievades datus viendimensionālā masīvā. CNN modeļa sastādīšanai izmanto *"Adam" optimizer* parametrs, *"categorical_crossentropy"* kā *loss* parametrs un *"accuracy"* kā *metrics* parametrs.

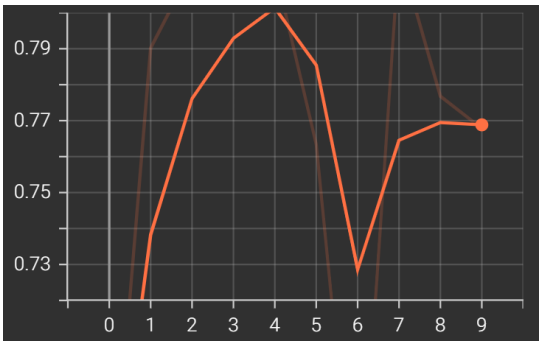
5. Apmācības rezultāti

Tiks novērotas 4 vērtības: *epoch*, iterācijas ilgums, apmācības precizitāte un testa precizitāte (precizitāte, ar kuru nosaka neredzētus datus). LSTM modelis tiek apmācīts ar 34,5MB lielu datu kopu, bet CNN modelis tiek apmācīts ar 9,09MB lielu datu kopu.

	LSTM modelis	**CNN modelis**
Epoch, ms/step	10	10
Iterācijas ilgums, sekundes	7,75	30.13
Apmācības precizitāte	0,8353 (83,53%)	0,7688 (76,88%)
Testa precizitāte	0,8 (80%)	0,8 (80%)

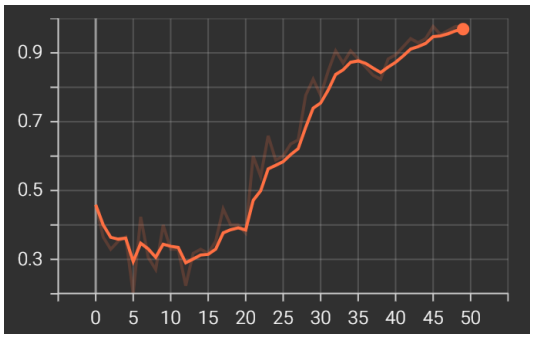


6. attēls. LSTM modeļa kategoriskā precizitāte, kur epoch = 10.

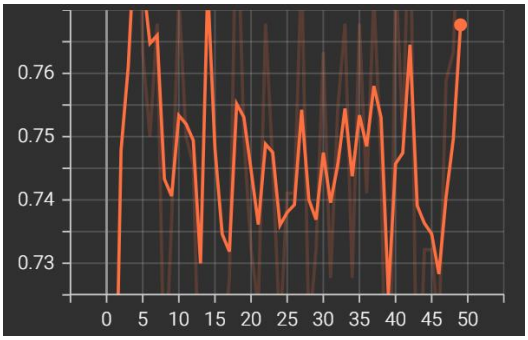


7. attēls. CNN modeļa precizitāte, kur epoch = 10.

	LSTM modelis	**CNN modelis**
Epoch, ms/step	50	50
Iterācijas ilgums, sekundes	14,65	166,83
Apmācības precizitāte	0,9765 (97,65%)	0,7946 (79,46%)
Testa precizitāte	0,8 (80%)	0,5 (50%)

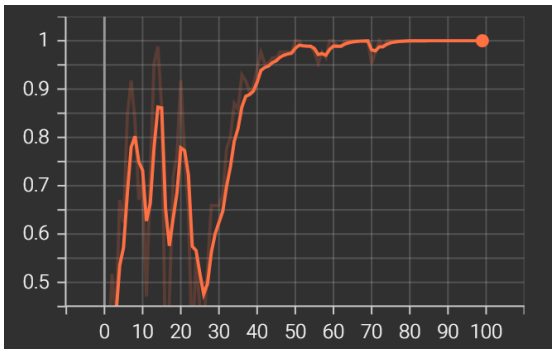


8. attēls. LSTM modeļa kategoriskā precizitāte, kur epoch =50.

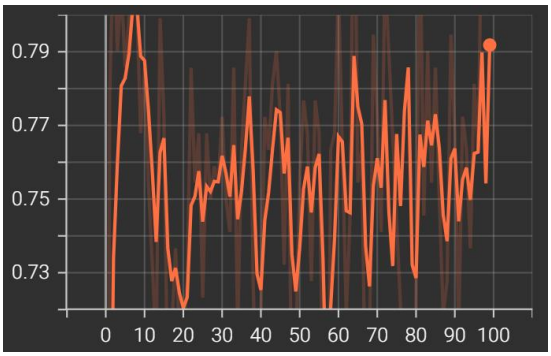


9. attēls. CNN modeļa precizitāte, kur epoch =50.

	LSTM modelis	CNN modelis
Epoch, ms/step	100	100
Iterācijas ilgums, sekundes	22,33	346
Precizitāte	1.00 (100%)	0,8482 (84,82%)
Testa precizitāte	1.0 (100%)	0,7 (70%)



10. attēls. LSTM modeļa kategoriskā precizitāte, kur epoch = 100.



11. attēls. CNN modeļa precizitāte, kur epoch = 100.

Salīdzinot LSTM modeli ar CNN modeli, ir redzams, ka LSTM modelis ir gan ātrāks, gan precīzāks visos attēlos. Paaugstinot *epoch* skaitu, abi modeļi apmācības laikā palielināja savu precizitāti, bet CNN modelī ir novērojama svārstīga precizitāte, kamēr LSTM modelim precizitāte auga. Ar *epoch* vērtību 10, LSTM modeļa apmācības precizitāte ir par 6,65% augstāka nekā CNN modelim, bet abiem saskaroties ar jauniem datiem, abi modeļi sasniedza 80% precizitāti. Ar *epoch* vērtību 50, starp LSTM modeli un CNN modeli, apmācības precizitātes starpība sasniedza 18,19% un saskaroties ar jauniem datiem, LSTM modeļa precizitāte ir par 30% augstāka nekā CNN modeļa precizitāte. Ar *epoch* vērtību 100, LSTM modelis sasniedza 100% precizitāti gan apmācībā, gan saskaroties ar jauniem datiem, bet CNN modeļa apmācības precizitāte sasniedza tikai 84,82% un saskaroties ar jauniem datiem, CNN modeļa precizitāte ir par 30% mazāka nekā LSTM modeļa precizitāte.

Secinājumi

1. Palielinoties *epoch* vērtībai, LSTM modelis apmācības laikā spēja atpazīt dotos žestu *”hello”, ”thanks”* un *”iloveyou”* datus ar lielāku precizitāti nekā CNN modelis.

2. LSTM modeļa apmācības ilgums ir vidēji 10,25 reizes ātrāks nekā CNN modeļa apmācības ilgums, tādēļ LSTM modelis gūst priekšrocību apmācības ilguma ziņā.

3. Saskaroties ar jauniem datiem, LSTM modelis tos atpazina ar aizvien augstāku precizitāti *epoch* vērtībai palielinoties, kamēr CNN modeļa precizitāte samazinājās, ieturot 30% starpību ar LSTM modeli.

Izmantotās literatūras saraksts

1. Basha, S.H.S. *et al.* (2020) ‘Impact of fully connected layers on performance of convolutional neural networks for image classification’, *Neurocomputing*, 378, pp. 112–119. Pieejams: https://doi.org/10.1016/j.neucom.2019.10.008.
2. Bezdan, T. and Bačanin Džakula, N. (2019) ‘Convolutional Neural Network Layers and Architectures’, in. Singidunum University, pp. 445–451. Pieejams: https://doi.org/10.15308/sinteza-2019-445-451.
3. Chen, G. (2016) ‘A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation’. Pieejams: http://arxiv.org/abs/1610.02583.
4. Couchot, J.-F. *et al.* (2016) ‘Steganalysis via a Convolutional Neural Network using Large Convolution Filters for Embedding Process with Same Stego Key’. Pieejams: http://arxiv.org/abs/1605.07946.
5. Dumoulin, V. and Visin, F. (2016) ‘A guide to convolution arithmetic for deep learning’. Pieejams: http://arxiv.org/abs/1603.07285.
6. Graves, A. *et al.* (2009) ‘A Novel Connectionist System for Unconstrained Handwriting Recognition’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5), pp. 855–868. Pieejams: https://doi.org/10.1109/TPAMI.2008.137.
7. Hochreiter, S. and Schmidhuber, J. (1997) ‘Long Short-Term Memory’, *Neural Computation*, 9(8), pp. 1735–1780. Pieejams: https://doi.org/10.1162/neco.1997.9.8.1735.
8. Kim, H.-J. and Baek, S.-W. (2023) ‘Implementation of wearable glove for sign language expression based on deep learning’, *Microsystem Technologies*, 29(8), pp. 1147–1163. Pieejams: https://doi.org/10.1007/s00542-023-05454-5.
9. O’Shea, K. and Nash, R. (2015) ‘An Introduction to Convolutional Neural Networks’. Pieejams: http://arxiv.org/abs/1511.08458.
10. Saha, S. (2018) ‘A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way’, 15 December. Pieejams: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (Skatīts: 9.februāris, 2024).
11. Sak, H., Senior, A. and Beaufays, F. (2014) ‘Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition’. Pieejams: http://arxiv.org/abs/1402.1128.
12. Schmidhuber, J. (2022) ‘Annotated History of Modern AI and Deep Learning’. Pieejams: http://arxiv.org/abs/2212.11279.
13. Staudemeyer, R.C. and Morris, E.R. (2019) ‘Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks’. Pieejams: http://arxiv.org/abs/1909.09586 (Skatīts: 16.janvāris, 2024).
14. University of Helsinki and MinnaLearn (2018) *Sarežģītākas neironu tīklu tehnoloģijas*. Pieejams: https://course.elementsofai.com/lv/5/3 (Skatīts: 9.februāris, 2024).
15. Wang, Z.J. *et al.* (2020) ‘CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization’. Pieejams: https://doi.org/10.1109/TVCG.2020.3030418.

Pielikumi

1. pielikums.
Pirmkods LSTM modelim, kas pielāgots cilvēka darbības atpazīšanai
https://github.com/nicknochnack/ActionDetectionforSignLanguage, materiāls skatīts 17. novembrī, 2023.

2. pielikums.
Pirmkods CNN modelim, kas pielāgots atpazīt cilvēka darbību bildēs
https://github.com/nicknochnack/ImageClassification/tree/main, materiāls skatīts 24. novembrī, 2023.