# Virtualna plutajuća tipkovnica zasnovana na kameri

**Abjanović, Fabijan**

**Master's thesis / Diplomski rad**

**2024**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:200:841773

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2025-03-21**

*Repository / Repozitorij:*

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**
**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK**

**Sveučilišni studij**

# VIRTUALNA PLUTAJUĆA TIPKOVNICA

**Diplomski rad**

**Fabijan Abjanović**

**Osijek, 2024.**

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

**Obrazac D1: Obrazac za ocjenu diplomskog rada na sveučilišnom diplomskom studiju**

## Ocjena diplomskog rada na sveučilišnom diplomskom studiju

| | |
|---|---|
| **Ime i prezime pristupnika:** | Fabijan Abjanović |
| **Studij, smjer:** | Sveučilišni diplomski studij Automobilsko računarstvo i |
| **Mat. br. pristupnika, god.** | D-63 ARK, 07.10.2022. |
| **JMBAG:** | 0165068614 |
| **Mentor:** | prof. dr. sc. Marijan Herceg |
| **Sumentor:** | |
| **Sumentor iz tvrtke:** | Zvonimir Kaprocki |
| **Predsjednik Povjerenstva:** | prof. dr. sc. Mario Vranješ |
| **Član Povjerenstva 1:** | prof. dr. sc. Marijan Herceg |
| **Član Povjerenstva 2:** | izv. prof. dr. sc. Ratko Grbić |
| **Naslov diplomskog rada:** | Virtualna plutajuća tipkovnica zasnovana na kameri |
| **Znanstvena grana diplomskog rada:** | **Telekomunikacije i informatika (zn. polje elektrotehnika)** |
| **Zadatak diplomskog rada:** | Funkcija virtualne tipkovnice je prepoznavanje i lokaliziranje pokreta korisnikovih prstiju na ravnoj površini korištenje kamere, kako bi se odredio odgovarajući pritisak na virtualnu tipku. Virtualna tipkovnica je projicirana na ravnoj površini i kada korisnik dodirne sliku tipke, uređaj prepoznaje dodir i zatim ispisuje znak na ekranu ili izvršava određenu funkciju, kao što je na primjer pritisak tipke "enter". U okviru ovog diplomskog rada potrebno je razviti algoritam zasnovan na dubokom učenju koji će koristiti jednu kameru za prepoznavanje |

| | |
|---|---|
| **Datum ocjene pismenog dijela diplomskog rada od strane mentora:** | 05.09.2024. |
| **Ocjena pismenog dijela diplomskog rada od strane mentora:** | Izvrstan (5) |
| **Datum obrane diplomskog rada:** | 13.09.2024. |
| **Ocjena usmenog dijela diplomskog rada (obrane):** | Izvrstan (5) |
| **Ukupna ocjena diplomskog rada:** | Izvrstan (5) |
| **Datum potvrde mentora o predaji konačne verzije diplomskog rada čime je pristupnik završio sveučilišni diplomski studij:** | 13.09.2024. |

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

## IZJAVA O IZVORNOSTI RADA

**Osijek,** 13.09.2024.

| | |
|---|---|
| **Ime i prezime Pristupnika:** | Fabijan Abjanović |
| **Studij**: | Sveučilišni diplomski studij Automobilsko računarstvo i komunikacije |
| **Mat. br. Pristupnika, godina upisa:** | D-63 ARK, 07.10.2022. |
| **Turnitin podudaranje [%]:** | 9 |

Ovom izjavom izjavljujem da je rad pod nazivom**: Virtualna plutajuća tipkovnica zasnovana na kameri**

izrađen pod vodstvom mentora prof. dr. sc. Marijan Herceg

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

# Sadržaj

1. UVOD ..... 1
2. PREGLED POSTOJEĆIH RADOVA IZ PODRUČJA IZRADE VIRTUALNIH TIPKOVNICA ... 3
3. OPIS VLASTITOG RJEŠENJA VIRTUALNE TIPKOVNICE ..... 7
3.1. Instalacija potrebnih biblioteka i konfiguracija radne okoline ..... 7
3.1.1. Korištene *Python* biblioteke ..... 8
3.1.2. Korištene *Pyhton* biblioteke za treniranje modela neuronske mreže ..... 10
3.2. Izrada vlastitog skupa podataka ..... 11
3.3. Općenito o neuronskoj mreži ..... 12
3.4. Konvolucijska neuronska mreža ..... 13
3.5. Rekurentna neuronska mreža ..... 14
3.6. Dugoročno-kratkoročna memorija neuronske mreže ..... 14
3.7. Dugoročna rekurentna konvolucijska mreža ..... 15
3.7.1. Opis arhitekture dugoročne rekuretne konvolucijske mreže ..... 15
3.7.2. Treniranje LRCN mreže ..... 20
3.8. Algoritam virtualne tipkovnice ..... 24
4. EVALUACIJA CJELOKUPNOG SUSTAVA VIRTUALNE TIPKOVNICE ..... 28
4.1. Evaluacija i odabir najbolje treniranog modela LRCN mreže ..... 30
4.2. Evaluacija i testiranje cjelokupnog algoritma virtualne tipkovnice u realnim uvjetima ..... 34
5. ZAKLJUČAK ..... 39
LITERATURA ..... 40
SAŽETAK ..... 43
ABSTRACT ..... 44
ŽIVOTOPIS ..... 45
PRILOZI ..... 46

# 1. UVOD

U današnjem dobu tehnologija se neprestano razvija te kao takva prilagođava potrebama korisnika, čineći tako svakodnevni život lakšim i praktičnijim. Jedan od aspekata tehnološke revolucije upravo je način interakcije s uređajima. Naime, tipkovnice su oduvijek bile neizostavni dio računalne opreme, međutim s vremenom su se pojavile ideje za njihovom modernizacijom. Klasične fizičke tipkovnice koje poznajemo, iako funkcionalne, imaju određene nedostatke koji se odnose na higijenu, a također zauzimaju određeni prostor na radnom stolu, što može ograničiti radnu površinu stola i smanjiti fleksibilnost u organizaciji radnog prostora. Upravo zbog navedenih razloga dolazi do razvoja virtualnih tipkovnica koje kao takve mogu značajno unaprijediti iskustvo mnogih korisnika.

Virtualne tipkovnice predstavljaju tehnologiju koja omogućuje unos slova, brojeva i znakova bez fizičkog kontakta s tipkovnicom. Naime, virtualne tipkovnice projiciraju se na određenim površinama koje se lako mogu očistiti ili se pojavljuju na zaslonu uređaja, omogućujući tako korisnicima tipkanje putem gesti ili dodira određenih površina. Ovakva inovacija za sobom nosi mnoge prednosti. Najprije, eliminira se fizički kontakt korisnika s tipkovnicom što značajno doprinosi higijeni, što je u današnje vrijeme posebno važno u kontekstu sprječavanja širenja raznih bolesti. Osim higijenskih prednosti, virtualne tipkovnice također omogućuju veću fleksibilnost u korištenju. Virtualne tipkovnice mogu se koristiti na različitim uređajima, uključujući pametne telefone, tablete, prijenosna računala, kao i stolna računala. Ova fleksibilnost korištenja omogućava korisnicima konzistentno iskustvo tipkanja bez obzira na uređaj koji koriste. Ujedno, virtualne tipkovnice omogućuju i personalizaciju rasporeda tipki, brojeva i znakova, što dodatno doprinosi prilagodbi individualnim potrebama korisnika.

Još jedna značajka virtualne tipkovnice je upravo njezina praktičnost, ponajviše u smislu uštede prostora. Svima poznate fizičke tipkovnice zauzimaju određeni prostor na radnome stolu ili čak u torbi u kojoj se ista prenosi, dok virtualne tipkovnice omogućuju veću slobodu i mobilnost. Korištenjem virtualne tipkovnice, korisnici imaju pristup istoj kad god im je to potrebno, bez dodatne opreme.

U ovome diplomskom radu izrađen je algoritam virtualne tipkovnice zasnovan na dubokom učenju, koji će korištenjem kamere prepoznati korisnikovu namjeru pritiska tipke na virtualnoj tipkovnici, odnosno „pritisak lijeve ruke“ i „pritisak desne ruke“ te sve ostalo

klasificirati u „nema pritiska“. U okviru diplomskog rada bilo je potrebno kreirati odgovarajući vlastiti skup podataka, koji se sastoji od video sekvenci tri različite, prethodno spomenute klase, na kojima je ujedno i trenirana neuronska mreža. Nakon treniranja dobiven je istrenirani model neuronske mreže visoke točnosti koji se pokazao dobro prilikom klasificiranja novih podataka. Tako istrenirani model predstavlja glavni dio algortima virtualne tipkovnice koji svojim predviđanjima omogućava nastavak rada algoritma. Nadalje, algoritam uključuje lociranje korisnikovog vrha prsta, usporedbu njegovog položaja s pozicijama tipki na virtualnoj tipkovnici te unos kada predviđanje pokaže da je došlo do pritiska.

Ovaj rad strukturiran je na sljedeći način. U uvodnome poglavlju opisana je tema rada, odnosno predstavljen je predmet rada, te je ujedno objašnjena struktura rada. U drugom poglavlju opisana su postojeća rješenja koja se odnose na problematiku rješenja virtualne tipkovnice, metode koje su korištene za izradu iste te prednosti i nedostatci takvih rješenja. Zatim slijedi poglavlje u kojem je opisano vlastito rješenje virtualne tipkovnice, počevši s instalacijom potrebnih biblioteka za rad algoritma te biblioteka koje su potrebne prilikom treniranja modela neuronske mreže. Osim toga, slijedi detaljno opisana izrada vlastitog skupa podataka, odgovarajućeg za izradu ovoga rješenja virtualne tipkovnice. Nakon toga, u istom poglavlju objašnjena je neuronska mreža te određene vrste neuronskih mreža, poput konvolucijske neuronske mreže (engl. *Convolutional Neural Network, CNN*), rekurentne neuronske mreže (engl. *Recurrent Neural Network, RNN*), dugoročno-kratkoročne memorije (engl. *Long short-term memory, LSTM*) te naposljetku dugoročna rekurentna konvolucijska neuronska mreža (engl. *Long-term Recurrent Convolutional Network, LRCN*) koja je kao takva korištena za izradu ovoga rješenja virtualne tipkovnice, čija se arhitektura i treniranje dodatno opisuju te se treće poglavlje završava detaljnim opisom i objašnjenjem rada algoritma virtualne tipkovnice. U četvrtom poglavlju provodi se evaluacija cjelokupnog sustava virtualne tipkovnice, koja uključuje evaluaciju i odabir nabolje treniranog modela LRCN mreže, te završava evaluacijom i testiranjem cjelokupnog algoritma virtualne tipkovnice u realnim uvjetima. Na kraju, u petom poglavlju dan je zaključak.

# 2. PREGLED POSTOJEĆIH RADOVA IZ PODRUČJA IZRADE VIRTUALNIH TIPKOVNICA

Napredak u tehnologiji omogućio je nove načine interakcije čovjeka i računala kakvi su prije bili gotovo pa ne zamislivi. Naime, postoje razna inovativna rješenja zasnovana na strojnom učenju koja omogućavaju novi način interakcije čovjeka i računala bazirana na metodama prepoznavanja pomoću algoritama stojnog učenja, kao što su prepozavanje lica, ruku, prstiju, raznih gestikulacija, pokreta, objekata u prostoru i slično. Jedno od inovativnih rješenja upravo je virtualna tipkovnica čija je izrada zasnovana na već spomenutim različitim vrstama interakcije.

Mnogi dosadašnji radovi iz područja izrade virtualne tipkovnice potaknuti su globalnom pandemijom Covid-19. Ljudi u strahu od navedene bolesti izbjegavaju dodirivanje stvari pa tako i tipkovnice koje su neophodne za interakciju s računalima, a upravo razna inovativna rješenja interakcije čovjeka i računala poput virtualne tipkovnice omogućuju metodu korištenja gestikulacija ruku za interakciju s računalnim sustavom bez dodira tipkovnice. U radu [1] predložen je sustav virtualne tipkovnice koja koristi kameru za praćenje pokreta ruku. Ideja opisana u navedenom radu proizašla je iz globalne pandemije COVID-19, kada su ljudi izbjegavali dodirivati predmete zbog straha od zaraze, uključujući tipkovnice koje su ključne za rad na računalima. Stoga, ovim radom predstavljen je način kako koristiti pokrete ruku za upravljanje računalima bez potrebe za fizičkim dodirom tipkovnice. Ova metoda smatra se inovativnom jer ne zahtijeva nikakve dodatne uređaje osim web kamere, koja može raditi u bilo kojoj orijentaciji. Koristi se kombinacija već izrađenih modula poput *OpenCV, Mediapip*e, *PyVDA, Win32API* i sličnih modula kako bi se postigla potrebna funkcionalnost. Rješenje je testirano na prijenosnom računalu s operativnim sustavom *Windows 10, Intel Core i5* procesorom, 8 GB RAM-a i web kamerom rezolucije 720p. Sustav može raditi s petnaest sličica u sekundi (engl. *frames per second, FPS*) te ima točnost (engl. *accuracy*) praćenja ruku od oko 90%. Iako se metoda pokazala uspješnom, postoje određeni nedostaci u usporedbi s drugim postojećim rješenjima. Na primjer, ova metoda nije u stanju obraditi kompleksne geste, može doći do pogreške uzrokovane pozadinskom bukom, postoji mogućnost nekompatibilnosti s drugim operativnim sustavima i uređajima. Međutim, tipkovnica na bazi kamere predstavlja praktično i inovativno rješenje za interakciju s računalima bez fizičkog dodira te unatoč nekim ograničenjima, pokazala se funkcionalno, točno i prenosivo rješenje koje je lako implementirati i koristiti u različitim uvjetima.

U radu [2] predložen je sustav virtualne tipkovnice zasnovan na tehnikama obrade slike. Jedna kamera koristi se za snimanje pokreta prstiju, a algoritam segmentacije slike koristi se za identifikaciju područja koje zauzima prst, umjesto oslanjanja na oblik ili konturu prsta. Također se koristi tehnika dinamičkog pragiranja, tehnika u obradi slike koja dinamički prilagođava prag (engl. *threshold*) za razdvajanje korisnih informacija od pozadinskih smetnji u slici ovisno o različitim uvjetima osvjetljenja i tonovima kože, te Kalmanov filter koji smanjuje šum i poboljšava preciznost u praćenju položaja prsta. U navedenom radu predstavljen je novi način detekcije i praćenja pokreta prstiju na projektiranoj tipkovnici i njihova pretvorba u tipke. Rješenje je testirano na prijenosnom računalu s projektorom i web kamerom, koristeći bijeli papir kao površinu za projekciju. Rezultati su uspoređeni s dva postojeća rješenja, virtualnom tipkovnicom baziranom na laseru i zaslonskom dodirnom tipkovnicom. Na temelju navedene usporedbe, detektirane su određene prednosti. Virtualna tipkovnica u navedenom radu pokazala se bržom od virtualne tipkovnice bazirane na laseru jer nije potrebno fizički pritiskati tipke, čime se smanjuje latencija i povećava frekvencija sličica. Također, virtualna tipkovnica u navedenom radu pokazala se točnijom od zaslonske dodirne tipkovnice jer nema problem zaklanjanja dijelova zalona prstima te ima veći postotak prepoznavanja i manji postotak pogrešaka. Ujedno, korisnicima je pruženo prirodnije iskustvo tipkanja u odnosu na druge dvije tipkovnice. Međutim, kao nedostaci takvoj virtualnoj tipkovnici, navedeni su osjetljivost na vanjske faktore poput uvjeta osvjetljenja, površine projekcije i pozicije kamere što može utjecati na kvalitetu i stabilnost sustava. Sustav je također manje prenosiv jer zahtijeva projektor i web kameru, te je manje siguran jer pritisci tipki mogu biti izloženi kameri ili promatračima, što može ugroziti privatnosti ili sigurnost podataka korisnika.

U radu [3] predložena je virtualna tipkovnica koja koristi tehnologiju prepoznavanja i obrade slike kako bi omogućila unos teksta na bilo kojoj površini. Konkretno, ovaj sustav koristi kamere za snimanje pokreta prstiju korisnika, a zatim pomoću algoritama za obradu slike identificira i prepoznaje te pokrete kao određene tipke na virtualnoj tipkovnici. Ovi algoritmi analiziraju vizualne informacije kako bi precizno prepoznali koje tipke korisnik pritišće. Ovakva virtualna tipkovnica prikladna je za mobilne uređaje i prijenosna računala te se može koristiti na bilo kojoj površini. Rezultati testiranja na različitim uređajima, poput pametnih telefona i prijenosnih računala, pokazali su da je virtualna tipkovnica učinkovita i jednostavna za upotrebu te su se korisnici brzo naviknuli na njezino korištenje. Stoga, konačni rezultat je jeftina, jednostavna i prenosiva virtualna tipkovnica koja šalje upisane

znakove na bilo koji uređaj sa zaslonom. Virtualna tipkovnica navedenog rada ima nekoliko prednosti u odnosu na druga rješenja. Najprije, može se koristiti na bilo kojoj površini, što je vrlo praktično. S druge strane, jeftina je i jednostavna za korištenje te vrlo učinkovita. Međutim, postoji nekoliko nedostataka. Budući da se tipkovnica temelji na prepoznavanju i obradi slike, može doći do pogrešaka u prepoznavanju znakova. Osim toga, nije prikladna za upotrebu u uvjetima slabog osvjetljenja.

U radu [4] predložen je sustav virtualne tipkovnice koji omogućuje korisnicima unos teksta s pomoću treptaja oka. Na zaslonu se prikazuje tipkovnica, a pokazivač miša automatski prolazi kroz sva slova. Korisnik jednostavno treba trepnuti kada pokazivač označi željeno slovo. Kamera prati lice korisnika te algoritam prepoznaje treptaj kao pritisak tipke. Rješenje koristi jednostavan i jeftin hardverski setup koji uključuje *Raspberry Pi* uređaj, web kameru i projektor. Softverski dio temelji se na robusnim algoritmima poput *Haarcascade* klasifikatora, *Canny* detektora i *Hough* transformacije, koji precizno prate oči korisnika i prepoznaju treptaje. Korisničko sučelje je intuitivno, omogućujući korisnicima odabir slova bez fizičkih dodira ili glasovnih komandi. Sustav je testiran s *Raspberry Pi 3 Model B+*, web kamerom s pet mega piksela i mini projektorom koji projicira tipkovnicu na bijeli zid te je uspoređen s dvije postojeće metode; zaslonskom dodirnom tipkovnicom i glasovno kontroliranom tipkovnicom. Na temelju toga, virtualna tipkovnica navedenog rada ima nekoliko prednosti. Preciznija je jer nema problema s velikim prstima ili slučajnim dodirom, brža je jer ne zahtijeva izgovaranje riječi ili slova, niti ovisi o točnosti prepoznavanja govora, te naposljetku je praktičnija jer ne zahtijeva fizički dodir ili glasno govorenje, što može biti neudobno ili nepraktično u određenim situacijama. Međutim, ističu se i određeni nedostaci poput osjetljivosti na vanjske faktore kao što su uvjeti osvjetljenja, kvaliteta projekcije i kut kamere što može utjecati na točnost sustava. Osim toga, često pomicanje očiju i treptanje može opteretiti mišiće očiju, uzrokujući umor ili napetost.

U radu [5] predložen je sustav virtualne tipkovnice koji koristi algoritam prepoznavanja gesti ruku korisnika, baziran na dubokom učenju kako bi u stvarnom vremenu detektirao i klasificirao geste ruku korisnika. Sustav se sastoji od dubinske kamere, projektora i računala. Virtualna tipkovnica navedenog rada dizajnirana je poput tipki na starim telefonima, gdje se pritiskom na jednu tipku birala grupa od 3 do 4 slova, a zatim bi se ponovnim pritiscima odabralo željeno slovo unutar te grupe. Za detekciju pritiska tipke koristi se udaljenost između palca i kažiprsta. Kako bi se riješio problem unosa istog slova zaredom, tipke tipkovnice su podijeljene na lijevu i desnu stranu prema frekvenciji korištenja

slova. Lijevim slovima pristupa se lijevom rukom, a desnim slovima desnom rukom, čime se postiže veća brzina tipkanja jer dok jedna ruka unosi slovo, druga može započeti unos sljedećeg slova. Algoritam prepoznavanja gesti kombinira CNN i RNN mreže kako bi se postigla visoka točnost i pouzdanost. CNN mreža izdvaja prostorne značajke gesti ruku iz dubinskih slika, dok RNN mreža bilježi temporalne značajke gesti iz sekvenci dubinskih slika. Također, za potrebe navedenog rada prikupljen je novi skup podataka koji sadrži 12000 dubinskih slika deset različitih gesti ruku, izvedenih od strane dvadeset osoba u različitim uvjetima. Prototip sustava testiran je s *Microsoft Kinect v2* kao dubinskom kamerom, *BenQ MW632ST* kao projektorom i računalom s operativnim sustavom *Windows 10, Intel Core i7-6700K* procesorom i *Nvidia GeForce GTX 1080* grafičkom karticom. Rezultati su pokazali da predloženi algoritam prepoznavanja gesti postiže prosječnu točnost od 98,3% na skupu podataka. Ujedno, algoritam radi s prosječnom brzinom od 30,8 FPS-a. Međutim, postoje i određeni nedostaci. Sustav zahtijeva proces kalibracije kako bi se uskladila dubinska kamera i projektor, što može biti nepraktično za korisnika, te je sustav ograničen rezolucijom i vidnim poljem dubinske kamere i projektora, što može utjecati na veličinu i poziciju virtualne tipkovnice.

Sigurno je neupitno kako mnoga inovativna rješenja omogućuju raznoliku interakciju čovjeka i računala te svaki od njih za sobom nose određene prednosti, ali i nedostatke. Neke od prednosti ovakvoga sustava interakcije čovjeka i računala jesu upravo to što se ne zahtijeva fizički dodir, time se smanjuje rizik od mogućnosti zaraze, zatim u većini slučajeva nema potrebe za dodatnim hardverima s obzirom na to da se koristi postojeća web kamera računala te naposljetku ovakav sustav primjenjiv je u različitim situacijama, odnosno koristan je kada su recimo ruke prljave ili u drugim situacijama kada ne postoji mogućnost fizički dodirivati tipkovnicu. Međutim, ovakvi sustavi interakcije za sobom nose i određene nedostatke poput kvalitete kamere, što bi značilo da bi sustav mogao biti manje precizan s kamerama slabije kvalitete. Također, dovodi se u pitanje funkcionalnost koja je uglavnom promjenjiva ovisno o uvjetima osvjetljenja u prostoru te će za pojedina rješenja biti potrebna dodatna kalibracija i prilagodba ovisno o korisniku. Osim toga, u nekim slučajevima može se pojaviti problem neiskusnih korisnika kojima će biti potrebno više vremena da se naviknu na tipkanje u drugačijim uvjetima nego onima na koje su naviknuti, što onda dovodi u pitanje učinkovitost u samom početku korištenja virtualne tipkovnice.

# 3. OPIS VLASTITOG RJEŠENJA VIRTUALNE TIPKOVNICE

Velika većina postojećih rješenja i izvedbe virtualne tipkovnice koriste nove načine interakcije s tipkovnicom koje širem krugu korisnika nisu uvijek prirodne odnosno intuitivne. Na primjer, neke metode zahtjevaju tipkanje dodirivanjem dva prsta međusobno u zraku, što može otežati precizan odabir tipki, dok druga rješenja zatijevaju dodatnu opremu koja može biti vrlo skupa. Upravo zato, ideja ovoga rada bila je izraditi virtualnu tipkovnicu što sličniju fizičkoj tipkovnici uz korištenje već ugrađene web kamere. Stoga ovako zamišljena virtualna tipkovnica zahtijeva samo praznu ravnu radnu površinu poput stola, web kameru te zaslon odnosno monitor na kojemu će biti prikazan raspored i pozicija svih tipki. Za funkcionalnost ovako zamišljene virtualne tipkovnice, jedan od najvažnijih dijelova je LRCN mreža čiji je zadatak predvidjeti je li se dogodio pritisak lijeve ruke, pritisak desne ruka ili detektirati da se pritisak nije dogodio. Kako bi spomenuta neuronska mreža mogla obavljati funkciju predviđanja, najprije ju je potrebno trenirati. Treniranje neuronske mreže ne može se započeti bez skupa podataka, koji kao takav mora biti što raznovrsniji odnosno treba se sastojati od više različitih slika ruku kako bi naposljetku i konačni model bio što precizniji i točniji. Međutim, s obzirom na to da niti jedan od javno dostupnih skupova podataka nije bio odgovarajući za ovaj tip rješenja, bilo je potrebno prikupiti te napraviti novi odgovarajući skup podataka. Nakon uspješnog treniranja neuronske mreže, potrebno je napraviti algoritam u *Python* programskom jeziku koji će u stvarnom vremenu snimati videozapise s kamere te napraviti predobradu svakog pojedinog okvira kako bi odgovarali ulazu u neuronsku mrežu i tako prikupiti petnaest uzastopnih okvira koji će zajedno biti proslijeđeni modelu neuronske mreže na predviđanje. Algoritam grafički prikazuje interaktivnu virtualnu tipkovnicu koja omogućuje korisniku tipkanje i unos željenih slova, znakova, brojeva ili izvršavanje određenih funkcija poput *enter-a*, razmaka ili brisanja.

## 3.1. Instalacija potrebnih biblioteka i konfiguracija radne okoline

Za razvoj virtualne tipkovnice korišten je *Python* 3.7.0. programski jezik, a da bi se olakšalo upravljanje projektom i instalacija potrebnih paketa, korištena je *Anaconda* distribucija koja pruža jednostavan način stvaranja i upravljanja virtualnim okruženjima, dok je za pisanje i uređivanje koda korišten *PyCharm IDE*. Radna okolina konfigurirana je na operacijskom sustavu *Microsoft Windows 10* kako bi se osigurala kompatibilnost s alatima i paketima korištenim tijekom razvoja virtualne tipkovnice. Također, ključne biblioteke koje omogućavaju funkcionalnost algoritma su *Tensorflow* 2.8.0.[6], *Keras* 2.8.0. [7], *NumPy* 1.21.6. [8], *CVZone* 1.4.1. [9], *Pynput* 1.7.3 [10], te *Mediapipe* 0.8.7.2. [11].

Za potrebe treniranja korištene su dodatne biblioteke poput *Scikit-learn* 0.24.0. [12], *Matplotlit* 3.5.3. [13] te da bi se ubrzao proces treniranja korištenje grafičke kartice bilo je potrebno instalirali i *CUDA Toolkit* 11.2.2. [14] te *cuDNN* 8.1.0.77. [15].

### 3.1.1. Korištene *Python* biblioteke

*TensorFlow* je besplatna i fleksibilna biblioteka otvorenog koda (engl. *open source*), posebno razvijena za rad s algoritmima strojnog učenja i umjetnom inteligencijom. Ova biblioteka omogućava znanstvenicima, istraživačima te inženjerima razvijanje i implementiranje modela dubokog učenja s velikom lakoćom. Naime, *TensorFlow* omogućuje korištenje kolekcije različitih slojeva (engl. *layers*), aktivacijskih funkcija (engl. *activation functions*), optimizatora i metoda za izradu različitih arhitektura neuronskih mreža. Također *TensorFlow* korisnicima omogućuje jednostavan način izrade modela neuronske mreže za treniranje te obavljanje složenih zadataka. Jedna od glavnih prednosti *Tensorflow-a* jest njegova sposobnost obavljanja složenih proračuna na različitim hardverskim platformama što uključuje centralnu procesorsku jedinicu (engl. *Central Processing Unit, CPU*), grafičku procesorsku jedinicu (engl. *Graphics Processing Unit, GPU*) te specijalizirane procesore poput jedinica za obradu tenzora (engl. *Tensor Processing Unit, TPU*), koji su optimizirani za brze matematičke operacije koje su ključne za treniranje dubokih neuronskih mreža. *TensorFlow* omogućuje korisnicima brzo prototipiziranje i implementiranje svojih modela te također podržava rad u distribuiranim okruženjima, čime se omogućava treniranje modela na više uređaja istovremeno, što znatno ubrzava proces treniranja, osobito kada se radi s velikim skupovima podataka. Za izradu rješenja virtualne tipkovnice u ovome radu korištena je verzija *2.8.0. TensorFlow* biblioteke.

*Keras* biblioteka apstraktna je biblioteka otvorenoga koda za duboko učenje koja omogućava jednostavnu i brzu izradu složenih neuronskih mreža. Ova biblioteka osmišljena je kako bi olakšala istraživačima i inženjerima rad sa složenim modelima strojnog učenja. *Keras* biblioteka dizajnirana je za rad s različitim slojevima i modelima neuronskih mreža, pružajući intuitivno aplikacijsko programsko sučelje (engl. *Application Programming Interface, API*) za jednostavno definiranje slojeva, aktivacijskih funkcija, optimizatora i funkcija gubitaka (engl. *loss*). Ova biblioteka je interoperabilna s drugim bibliotekama za duboko učenje kao što su *Tensorflow* i *Theano*, što omogućava veću fleksibilnost u izboru platformi za treniranje modela. Jedna od ključnih značajki *Keras* biblioteke je njezina sposobnost vizualizacije i praćenja napretka treniranja modela. Korisnici mogu pratiti razne metrike performansi, uključujući gubitak i točnost, kroz cijeli proces treniranja i validacije.

Za izradu rješenja virtualne tipkovnice u ovome radu korištena je verzija 2.8.0. *Keras* biblioteke.

*NumPy* je fundamentalna biblioteka za numeričko računanje u *Python* programskom jeziku. Ova biblioteka razvijena je s ciljem pružanja efikasne i brze operacije na velikim nizovima i matricama što je ključno za znanstvene i analitičke aplikacije. *NumPy* biblioteka omogućava korisnicima obavljanje raznih matematičkih i logičkih operacija nad nizovima, koristeći se funkcionalnostima koje nisu dostupne u običnim listama *Python* programskog jezika. *NumPy* biblioteka nezaobilazna je u mnogim projektima jer pruža moćne alate za učitavanje, spremanje te manipulaciju podacima različitih formata, uključujući vrijednosti odvojene zarezima (engl. *Comma Separated Values, CSV*), tekstualne, binarne i druge vrste datoteka. Za izradu rješenja virtualne tipkovnice u ovome radu korištena je verzija 1.21.6. *NumPy* biblioteke.

*OpenCV (*engl. *Open Source Computer Vision Library*) je biblioteka otvorenoga koda koja sadrži jako velik broj algoritama računalnog vida (engl. *computer vision*), strojnog učenja i algoritma za obradu slike. *OpenCV* napravljen je kako bi osigurao zajedničku infrastrukturu za aplikacije računalnog vida i ubrzao korištenje strojne percepcije u komercijalnim proizvodima. Ova biblioteka sadrži više od 2500 optimiziranih algoritama, što uključuje i opsežan skup klasičnih i najsuvremenijih algoritama računalnog vida i strojnog učenja koji služe za otkrivanje i prepoznavanje ljudskoga lica, klasifikaciju ljudskih aktivnosti tijekom video sekvenci, identifikaciju različitih objekata, praćenje pokretnih objekata u videozapisima i slično. Za izradu rješenja virtualne tipkovnice u ovome radu korištena je verzija 1.4.1. *CVZone*, paket računalnog vida koji olakšava pokretanje funkcije obrade slike i umjetne inteligencije koji u osnovi koristi *OpenCV* i *Mediapipe* biblioteke.

*Pynput* biblioteka omogućuje kontrolu, upravljanje i nadzor ulaznih uređaja, odnosno unosa s miša i tipkovnice. Ova biblioteka omogućuje kreiranje programa koji mogu izvršavati radnje poput pritiska tipki, pomicanja miša i druge interakcije s korisnikovim uređajima. Za izradu rješenja virtualne tipkovnice u ovome radu korištena je verzija 1.7.3.*Pynput* biblioteke.

*Mediapipe* je biblioteka otvorenoga koda razvijena od strane *Google*-a koji omogućuje jednostavnu izradu rješenja za strojno učenje na različitim platformama, uključujući web, *Android* i *iOS*. Koristi grafove za organiziranje tijeka podataka, što omogućuje laku prilagodbu i ponovnu upotrebu komponenti. *Mediapipe* biblioteka dolazi s mnoštvom unaprijed pripremljenih modula i primjera, uključujući funkcionalnosti poput

trajnog praćenja objekata, prepoznavanja lica, praćenja poza i slično. Korištenje C++ koda osigurava optimalnu izvedbu, a okvir (engl. *framework*) je posebno pogodan za mobile uređaje. Za izradu rješenja virtualne tipkovnice u ovome radu korištena je verzija 0.8.7.2. *Mediapipe* biblioteke.

### 3.1.2. Korištene *Pyhton* biblioteke za treniranje modela neuronske mreže

*Scikit-learn* popularna je biblioteka otvorenoga koda za strojno učenje u *Python* programskom jeziku koja je izgrađena na temeljima drugih biblioteka poput *NumPy, SciPy* i *Matplotlib*. Ova biblioteka pruža širok spektar alata za analizu podataka te modeliranje, uključujući i razne algoritme za klasifikaciju, regresiju, grupiranje i smanjenje dimenzionalnosti. *Scikit-learn* biblioteka pruža alate za pretprocesiranje podataka, uključujući standardizaciju, normalizaciju, kategorizaciju te popunjavanje nedostajućih vrijednosti. Osim toga, omogućuje jednostavno spajanje niza transformacija i modela kroz cjevovod (engl. *pipeline*), olakšavajući time treniranje te evaluaciju modela. Modularnost i jednostavnost ove biblioteke omogućuje korisnicima dodavanje vlastitih modela i algoritama. Za potrebe treniranja modela neuronske mreže u ovome radu korištena je verzija 0.24.0. *Scikit-learn* biblioteke.

*Matplotlib* je sveobuhvatna biblioteka za stvaranje statičnih, animiranih i interaktivnih vizualizacija u *Python* programskom jeziku. Pomoću ove biblioteke mogu se stvarati kvalitetni interaktivni grafikoni koji se mogu uvećavati, pomicati i ažurirati, te se omogućava prilagodba vizualnog stila, izgleda i formata datoteka. Za potrebe grafičkog prikaza krivulje točnosti i gubitka prilikom treniranja u ovome radu korištena je verzija 3.5.3. *Matplotlit* biblioteke.

*NVIDIA CUDA Toolkit* pruža razvojno okruženje za stvaranje *GPU*-ubrzanih aplikacija visokih performansi. Pomoću *CUDA Toolkit-a* mogu se razviti, optimizirati te implementirati aplikacije na *GPU*-ubrzanim ugrađenim sustavima, stolnim radnim stanicama, poslovnim podatkovnim centrima, platformama temeljenim na oblaku (engl. *cloud-based*) te superračunalima. Skup alata *CUDA Toolkit-a* uključuje *GPU*-ubrzane biblioteke, alate za uklanjanje pogrešaka i optimizaciju, C/C+++ kompajler i biblioteku za vrijeme izvođenja. Radi ubrzanja procesa treniranja korištenjem grafičke kartice korištena je verzija 11.2.2. *CUDA Toolkit* biblioteke.

*NVIDIA CUDA Deep Neural Network (cuDNN)* je biblioteka *GPU*-ubrzanih operacija dizajniranih za duboke neuronske mreže. Ova biblioteka koristi *CUDA* okvir kako bi

iskoristila snagu *NVIDIA GPU*-ova za opće namjene računarstva. Ovo visoko-performansno *GPU* ubrzanje značajno ubrzava računanja, smanjujući ukupno vrijeme obrade. Izgradnja i optimizacija dubokih neuronskih mreža zahtijeva skup funkcija visoke razine i operacija niske razine, koje pruža upravo ova biblioteka. Među njima su konvolucija, poduzorkovanje (engl. *pooling*), normalizacija, aktivacijske funkcije, rekurentni slojevi i druge tehnike. Ova biblioteka optimizira navedene postupke i stoga dramatično ubrzava izvršavanje modela neuronskih mreža na *NVIDIA GPU*-ovima. Radi ubrzanja procesa treniranja korištenjem grafičke kartice korištena je verzija 8.1.0.77. *cuDNN* biblioteke.

## 3.2. Izrada vlastitog skupa podataka

S obzirom na to da niti jedan od javno dostupnih skupova podataka nije bio odgovarajući, u okviru diplomskog rada prikupljen je te izrađen vlastiti skup podataka potreban za ovakvo rješenje virtualne tipkovnice. Kako bi konačni model bio što precizniji i točniji, bilo je potrebno napraviti što veći skup podataka, uz naglasak na raznolikost podataka, odnosno što veći broj različitih ruku, kako bi model u konačnici mogao prepoznati nove ruke korisnika, a ne samo one na kojima je treniran.

Za izradu vlastitog skupa podataka korištena je kamera *White Shark GWC-004 Owl*. Koristeći već ugrađenu *Windows* aplikaciju *Camera*, snimane su ruke ukupno šest osoba kako tipkaju po praznoj površini. Zatim je videozapise bilo potrebno pretvoriti u nekoliko tisuća okvira, sve pomno pregledati te sortirati sekvence od po petnaest okvira u tri klase: „pritisak lijeve ruke“, „pritisak desne ruke“ i „nema pritiska“. Naime, petnaest okvira odabrano je proizvoljno jer je empirijski utvrđeno kako je to dovoljan broj uzastopnih okvira u koje se može pohraniti cijeli pokret pritiska tipke. S obzirom na to da kamera snima u 30 FPS-a, ovih petnaest uzastopnih okvira predstavlja video sekvencu u trajanju od pola sekunde. Veći broj okvira uzrokovao bi veći memorijski otisak, dok se korištenjem manjeg broja uzastopnih okvira ne bi mogao pohraniti cijeli pokret pritiska tipke, čime bi se izgubile važne informacije. Nakon navedenog dugotrajnog procesa pregledavanja te sortiranja videosekvenci u klase, bilo je potrebno napraviti pred obradu podataka kako bi oni bili prikladni za ulaz u neuronsku mrežu. Najprije je bilo potrebno izrezati svaki okvir na jednako određenu regiju od interesa (engl. *region of interest, ROI*), kako bi se iz snimljenih okvira uklonili nepotrebni detalji. Konkretno, iz svakog okvira se reže područje stola jer su važni samo oni pokreti koji se događaju iznad i na površini stola. Na taj su način početni okviri snimljeni u 1920x1080 *full HD* rezoluciji rezanjem postali rezolucije 1664x624. S obzirom

na to da nisu potrebni tonovi boja, odnosno *RGB* slika, okviri su prebačeni u nijanse sive (engl. *grayscale*), te se na taj način smanjio memorijski otisak svakog okvira tri puta - okviri dubine 3 postaju okviri dubine 1. Zatim su okviri smanjeni na rezoluciju 104x39 kako bi još dodatno i znatno smanjili memorijski otisak. U slučaju da *ROI* ne bi bila prethodno izrezana, okviri bi i dalje bili smanjeni na istu rezoluciju, ali bi ujedno ostali i nepotrebni detalji, poput pokreta stolica, nogu i svega onoga što se ne događa iznad površine stola, tako ostaju samo bitni detalji koji čine okvir u cijelosti. Također, okviri, ali i kamera u algoritmu su zrcaljeni kako bi se ostvario osjećaj poznatog tipkanja poput onoga po fizičkoj tipkovnici. Naposljetku, cijela tako obrađena klasa prebačena je u *NumPy* polje kako bi se mogla učitati za daljnje treniranje modela.

Krajnji skup podataka podijeljen je u tri klase i sastoji se od: 1072 video sekvence prve klase odnosno *LEFT_BTN_PRESS* od kojih je 804 video sekvenci određeno za treniranje i validaciju, a 268 za testiranje. Zatim 2393 video sekvence druge klase odnosno *NOT_BTN_PRESS* od kojih je 1795 video sekvenci određeno za treniranje i validaciju, a 598 za testiranje, te naposljetku 1136 video sekvence treće klase odnosno *RIGHT_BTN_PRESS* od kojih je 853 video sekvenci određeno za treniranje i validaciju, a 283 za testiranje.

## 3.3. Općenito o neuronskoj mreži

Prije nego što bude objašnjeno kako zapravo funkcionira LRCN mreža korištena za rješenje virtualne tipkovnice u ovome radu, najprije treba nešto reći o neuronskoj mreži općenito. Neuronske mreže su jedan od ključnih koncepata u području umjetne inteligencije. Temelje se na strukturi i funkciji ljudskog mozga, a koriste se za rješavanje složenih problema poput prepoznavanja slika, obrade jezika i predviđanja obrazaca. U neuronskoj mreži, podaci se obrađuju kroz slojeve međusobno povezanih čvorova, neurona. Svaki neuron prima ulazne podatke, na temelju kojih se izračunava izlazna vrijednost. Ova vrijednost se potom prenosi na sljedeći sloj neurona. Proces učenja u neuronskoj mreži odvija se prilagođavanjem težina povezanih s tim čvorovima kako bi se minimizirala pogreška između predviđenih i stvarnih rezultata. Neuronske mreže treniraju se pomoću velikih količina podataka te koriste algoritme poput gradijentnog spusta (engl. *gradient descent*) za optimizaciju svojih performansi. Kako bi se postigli najbolji rezultati, mreža prolazi kroz više iteracija učenja, tijekom kojih se težine kontinuirano prilagođavaju. Složenost neuronskih mreža omogućava im rješavanje zadataka koji bi za tradicionalne računalne sustave bili izuzetno teški ili čak nemogući. Posebno se ističu duboke neuronske mreže koje sadrže više

slojeva između ulaza i izlaza, omogućujući prepoznavanje i apstrahiranje složenih obrazaca u podacima. Neuronske mreže predstavljaju snažan alat u suvremenoj tehnologiji, omogućujući napredak u brojnim područjima kao što su autonomna vozila, medicina, financije te mnogi drugi sektori [16].

## 3.4. Konvolucijska neuronska mreža

CNN se koristi za prepoznavanje obrazaca u podacima, osobito kod analize slika i videozapisa, te predstavlja ključnu tehnologiju u području računalnog vida. U središtu ove tehnologije nalazi se proces konvolucije, osnovni matematički postupak koji omogućuje automatsko izdvajanje značajki iz slike. Konvolucija uključuje primjenu filtera na sliku kako bi se dobila nova, obrađena slika. Filter je mala matrica, najčešće dimenzija 3x3 ili 5x5, koja sadrži brojčane vrijednosti određene za izvođenje specifičnih operacija na pikselima slike. Proces počinje tako što se filter postavlja na dio slike, obično počevši od gornjeg lijevog kuta. Potom se izračunava umnožak elemenata filtera s odgovarajućim pikselima ispod njega. Rezultati tih umnožaka se zbrajaju stvarajući novu vrijednost piksela na izlaznoj slici. Filter se zatim pomiče preko slike, korak po korak, sve dok cijela slika nije obrađena. Rezultat konvolucije je nova slika, poznata i kao mapa značajki (engl. *feature map*), koja je obično smanjena u dimenzijama i naglašava određene značajke izvorne slike, poput rubova, tekstura ili drugih obrazaca. Ovako transformirana slika omogućuje CNN mreži da prepozna i nauči značajke koje su važne za zadatke poput prepoznavanja objekata i klasifikacije slika. U CNN mreži, proces konvolucije odvija se u konvolucijskom sloju. Nakon njega slijedi sloj poduzorkovanja, koji dodatno smanjuje dimenzionalnost mape značajki, zadržavajući pritom ključne informacije. Time se povećava robusnost mreže na promjene u ulaznim podacima, kao što su translacije i deformacije. Na kraju mreže, koriste se potpuno povezani slojevi (engl. *fully connected layers*) za donošenje konačnih odluka na temelju značajki izdvojenih u prethodnim slojevima. Učenje u CNN mreži postiže se optimizacijom težina filtera i potpuno povezanih slojeva, što omogućuje mreži minimiziranje pogreške u predikcijama. Taj proces izvodi se pomoću metode propagiranja unazad (engl. *backpropagation*) u kombinaciji s algoritmom gradijentnog spuštanja. Kroz sve ove procese, CNN mreža omogućuje dubinsku analizu i razumijevanje složenih struktura u podacima, postižući visoku preciznost i učinkovitost u zadacima prepoznavanja obrazaca [17].

## 3.5. Rekurentna neuronska mreža

RNN mreža se koristi za obradu sekvencijalnih podataka, gdje redoslijed elemenata ima ključnu ulogu. Za razliku od klasičnih neuronskih mreža, koje pretpostavljaju kako su svi ulazi neovisni jedan o drugom, RNN-ovi omogućuju prijenos informacija iz jednog koraka u sekvencijalnom podatku u sljedeći, čime se omogućuje modeliranje vremenskih ovisnosti. To je izuzetno važno za zadatke poput analize vremenskih serija, obradu prirodnog jezika i predviđanja u raznim domenama. Međutim, treniranje RNN mreža suočava se s izazovima poznatim kao eksplodirajući gradijent (engl. *exploding gradient*) i nestajući gradijent (engl. *vanishing gradient*). Ovi problemi nastaju tijekom metode propagiranja unazad, kad se gradijenti funkcije gubitka propagiraju unazad kroz slojeve mreže kako bi se ažurirale težine. U slučaju eksplodirajućeg gradijenta, gradijenti postaju izuzetno veliki, što uzrokuje ekstremno velika ažuriranja težina. To može dovesti do nestabilnosti u učenju, gdje funkcija gubitka oscilira ili čak proces treniranja divergira, čime mreža ne može pronaći optimalna rješenja. S druge strane, kod nestajućeg gradijenta, gradijenti se smanjuju dok propagiraju unazad, što rezultira time da se težine prestaju značajno ažurirati. Ovaj problem posebno pogađa duboke mreže, gdje slojevi bliže ulazu dobivaju sve manje korisnih informacija za učenje, što onemogućuje mreži učinkovito učenje složenijih obrazaca i odnosa s podacima. Kako bi se ovi problemi ublažili, razvijene su tehnike poput rezanja gradijenta (engl. *gradient clipping*) za eksplodirajuće gradijente te korištenje naprednih arhitektura poput LSTM mreže i jedinice s upravljanim povratom (engl. *Gated Recurrent Unit, GRU*), koje omogućuju mreži pamćenje važnih informacija tijekom duljih sekvenci i zadržava stabilnost tijekom treniranja [18].

## 3.6. Dugoročno-kratkoročna memorija neuronske mreže

LSTM je poseban tip RNN mreže, dizajnirana kako bi se prevladali problemi dugoročne ovisnosti u standardnim RNN-ovima. Glavni izazovi kod standardnih RNN-ova jest problem eksplodirajućeg i nestajućeg gradijenta, što onemogućuje učinkovito učenje dugoročnih ovisnosti u podacima. Kako bi se ovaj problem prevladao, LSTM-ovi koriste posebne strukture, poznate kao ćelije (engl. *cells*), koje omogućuju dugoročno pamćenje informacija. LSTM ćelije se sastoje od tri osnovne vrste vrata (engl. *gates*): vrata za zaborav (engl. *forget gate*), ulazna vrata (engl. *input gate*) i izlazna vrata (engl. *output gate*). Svaka od ovih vrata kontrolira tok informacija kroz ćeliju, dopuštajući ili onemogućavajući prijenos podataka s jednog vremenskog koraka na drugi. Vrata za zaborav odlučuju koje će se informacije zadržati, a koje zaboraviti. To se postiže pomoću sigmoidne funkcije koja

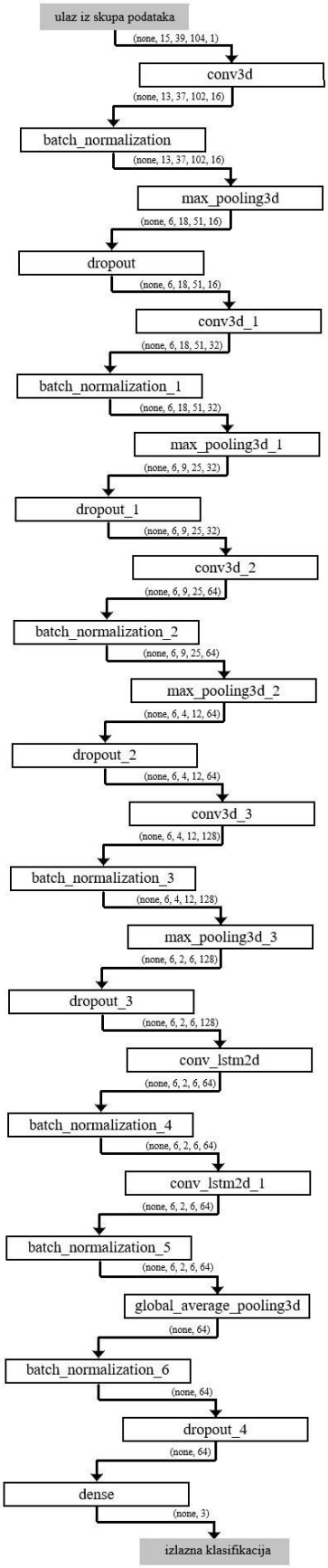
generira vrijednosti između 0 i 1, gdje 0 predstavlja potpuno zaboravljanje, a 1 potpuni prijenos informacija. Ulazna vrata kontroliraju koje će nove informacije biti pohranjenje u memoriju ćelije. To se također radi pomoću sigmoidne funkcije, ali u kombinaciji s tangens funkcijom koja osigurava prijenos samo relevantnih informacija. Izlazna vrata određuju koje će informacije iz ćelije biti prenesene u sljedeći vremenski korak. Tako model može odlučiti koje su informacije iz dugoročne memorije potrebne za trenutnu procjenu. Jedna od ključnih značajki LSTM-ova jest mogućnost pamćenja informacija kroz mnoge vremenske korake, što ih čini iznimno korisnima za zadatke koji zahtijevaju analiziranje sekvencijalnih podataka, kao što su obrada prirodnog jezika, prepoznavanje govora i predviđanje vremenskih serija [19].

## 3.7. Dugoročna rekurentna konvolucijska mreža

LRCN mreže su predstavljene kao model koji kombinira CNN i RNN mreže, a glavna svrha im je poboljšanje obrade sekvencijalnih podataka u različitim zadacima računalnog vida i obrade prirodnog jezika. LRCN je razvijen kako bi se iskoristila sposobnost CNN mreže u ekstrakciji značajki iz slika te sposobnost RNN mreže u modeliranju vremenskih ovisnosti unutar sekvenci. U LRCN modelu, CNN se koristi za izdvajanje prostorno hijerarhijskih značajki iz pojedinačnih okvira video sekvenci ili slika. Ove značajke zatim se prosljeđuju RNN-u, obično u obliku LSTM ili GRU jedinica, koje su zadužene za hvatanje vremenskih ovisnosti među značajkama. Ovaj pristup omogućuje modelu učenje kako se značajke mijenjaju kroz vrijeme i kako se one mogu koristiti za donošenje predikcija. LRCN se ističe po svojoj sposobnosti učinkovite obrade složenih zadataka poput prepoznavanja aktivnosti u videozapisima, opisivanja sadržaja slika i prepoznavanja govora. Jedna od ključnih prednosti LRCN mreže je njezina modularnost, koja omogućuje jednostavnu prilagodbu različitim zadacima, bez potrebe za značajnim promjenama u osnovnoj arhitekturi. CNN i RNN komponente mogu se trenirati zajedno, krajem na kraj (engl. *end-to-end*), čime se postiže bolja usklađenost modela s ciljevima zadatka. LRCN mreže predstavljaju značajan napredak u dubokom učenju, omogućujući integraciju prostorne i vremenske informacije na način koji je bio nedostižan ranijim modelima [20].

### 3.7.1. Opis arhitekture dugoročne rekuretne konvolucijske mreže

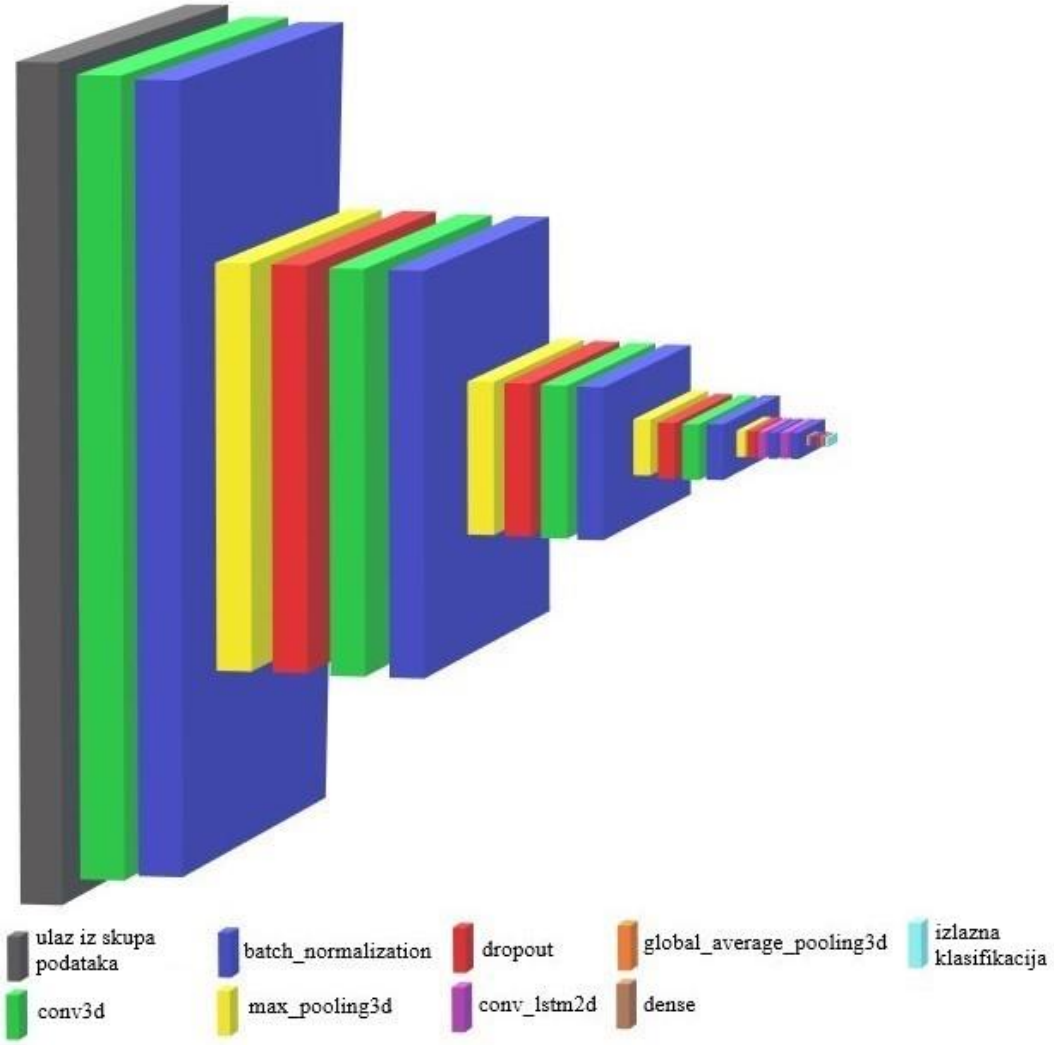
Svi slojevi s prikazanim dimenzijama izlaza iz svakog sloja koji zajedno čine arhitekturu korištene LRCN mreže prikazani su na slici 3.1.




**Slika 3.1.** Prikaz arhitetkure LRCN mreže korištene u ovome radu

Ulaz u mrežu je 5D (peterodimenzionalno) *NumPy* polje, koje je dobiveno u zadnjem koraku kreiranja skupa podataka. Gledano s lijeva na desno, prvi broj unutar 5D *NumPy* polja predstavlja veličinu serije (engl. *batch size*), koja označava broj podataka koji se obrađuju zajedno u jednom prolazu kroz mrežu. Taj broj označava se kako „ništa“ (engl. *none*) u definiciji slojeva zato što prilikom dizajniranja mreže veličina serije ne mora biti specificirana – ona se određuje tijekom treniranja modela. Veličina serije unaprijed se određuje kada se pokrene treniranje modela, ali mreža je projektirana tako da se može raditi s bilo kojom veličinom serije pa je zato ta dimenzija označena kao „ništa“ u arhitekturi. Drugi broj u polju predstavlja video sekvence odnosno koliko video okvira video sekvenca sadrži. Prilikom ulaska u mrežu taj broj je uvijek fiksiran na 15 jer svaka video sekvenca prikupljena u skupu podataka koji prikazuje određenu gestu sastoji se od točno 15 okvira. Treći i četvrti broj u polju predstavljaju rezoluciju svakog okvira, pritom treći broj predstavlja broj piksela vertikalno po visini, dok četvrti broj u polju predstavlja broj piksela horizontalno po širini. Posljednji broj u polju predstavlja dubinu svakog okvira, koji je na samom ulazu postavljen na 1 jer se radi o okvirima pretvorenima u nijanse sive. Kako video sekvence prolaze kroz mrežu, tako se mijenja rezolucija slike unutar 5D polja, kao i broj video sekvenci te dubina filtera, dok se dimenzija polja mijenja tek u zadnja četiri sloja. Na kraju, od ulaznog 5D polja ostaje 2D polje dimenzije (*none*, 3) koje kao izlaz ima samo tri mogućnosti – "pritisak lijeve ruke", "pritisak desne ruke" te "nema pritiska". Vizualizirana arhitektura korištene LRCN mreže prikazana je na slici 3.2, gdje se jasno mogu vidjeti promjene u dimenziji okvira video sekvenci. U ovome slučaju arhitektura LRCN mreže sadrži 3D konvolucijske slojeve. Naime, tipičan način pretvaranja 2D video okvira u 3D video podatke je slaganje uzastopnih okvira u novu dimenziju koja predstavlja vrijeme. Upravo zato je ulaz u ovakvu arhitekturu LRCN mreže (prikazan na slici 3.1.) 5D polje. To je zato što se 4D polje, koje se koristi kao ulaz u CNN mrežu za obrađivanje slika, proširuje dodavanjem dodatne dimenzije, koja predstavlja broj okvira u video sekvenci. Dakle, arhitektura korištene LRCN mreže sastoji se od četiri 3D konvolucijska sloja i dva konvolucijska LSTM 2D sloja. CNN slojevi su 3D jer rade s trodimenzionalnim prostornim podacima - video okvirima (visina, širina, dubina), dok su LSTM slojevi 2D jer fokusiraju svoju analizu na dvije prostorne dimenzije (visina i širina) unutar sekvencijalnog okvira. Veličina filtra svakog 3D konvolucijskog sloja je (3,3,3) dok broj filtera dvostruko raste, stoga je prvi konvolucijski sloj definiran sa 16 filtera, drugi s 32, treći sa 64 te četvrti sa 128. Ujedno, svaki 3D konvolucijski sloj koristi *ReLu* aktivacijsku funkciju. Prvi konvolucijski sloj nema definiran *padding* pa kao posljedica toga smanjuje rezoluciju okvira za dva po visini i po širini, dok ostala tri konvolucijska sloja imaju definiran

*padding* pa se rezolucija okvira ne mijenja. Nakon svakog konvolucijskog sloja nalazi se *batch_normalization*, *max_pooling* te *dropout*. *Batch_normalization* normalizira aktivacije unutar svakog mini *batch*-a, odnosno unutar manjeg skupa podataka koji se obrađuje odjednom, čime se smanjuje varijacija između pojedinačnih uzoraka unutar toga skupa. Time se osigurava da ulazi u sljedeće slojeve budu stabilniji i dosljedniji, što znači da su manje podložni promjenama u distribuciji podataka i brže konvergiraju tijekom učenja modela.




**Slika 3.2.** Vizualizirani prikaz korištene arhitekture LRCN mreže

*Max_pooling* je sloj maksimalnog udruživanja, koji smanjuje dimenzionalnost okvira – tako na primjer u slučaju (2,2) filtra uzima četiri susjedna piksela koje uspoređuje te četiri piksela svodi na jedan najznačajniji piksel. U ovoj mreži *Max_pooling* filter nakon prvog konvolucijskog sloja ima filter (2,2,2) dok nakon drugog, trećeg i četvrtog ima filter (1,2,2). *Dropout* je sloj koji nasumično nulira neke od neurona prethodnih slojeva kako bi se spriječio

*overfitting*. *Overfitting* je koncept koji se pojavljuje u neuronskoj mreži kada se model previše uskladi prema podacima koji su mu dani prilikom treniranja, a kao posljedica toga jako slabo radi u predviđanju novih do sada ne viđenih podataka. Nakon svakog sloja *dropout* isključuje 25% neurona prethodnih slojeva, dok na samom kraju mreže prije krajnje klasifikacije *dropout* isključuje 50% neurona prethodnih slojeva. Nakon svih 3D konvolucijskih i dodatnih slojeva, dolaze dva LSTM sloja. Veličina filtra oba LSTM sloja je (3,3) i broj filtera je u oba slučaja isti, odnosno 64. Nakon svakog LSTM sloja ponovno je dodan *batch_normalization*. Zatim slijedi *Global Average Pooling3D* sloj koji 5D dimenzionalnost smanjuje na 2D dimenzionalnost. Na samom kraju nalazi se *dense* sloj sa *softmax* aktivacijskom funkcijom koji sadrži tri jedinice koje izravno predstavljaju tri potrebne klase. Za potrebe treniranja koristi se *Adam Optimizator* i rijetka kategorijalna unakrsna entropija (engl. *Sparse categorical cross entropy)* za praćenje gubitka točnosti i validacije. Rijetka kategorijalna unakrsna entropija koristi se u zadacima klasifikacije s više klasa, gdje su oznake predstavljene cijelim brojevima koji označavaju indekse tih klasa. Funkcija gubitka unakrsne entropije (engl. *Cross entropy loss function*) koristi se za mjerenje uspješnosti modela, osobito u zadacima u kojima model treba svrstati podatke u jednu od mnogih mogućih kategorija. Svrha ove funkcije je kvantificirati razliku između stvarne raspodjele klasa i raspodjele vjerojatnosti koju model predviđa. U ovom kontekstu, oznake podataka obično se predstavljaju kao „*one-hot encoding*“ vektori. To znači da su sve vrijednosti u vektoru nule, osim na mjestu koje odgovara ispravnoj klasi. Tako se modelu signalizira koja je točna klasa. Kod rijetke kategorijalne unakrsne entropije, koristi se pojednostavljeni pristup. Umjesto „*one-hot encoding*“ vektora, koristi se samo jedan cijeli broj koji označava ispravnu klasu. Ovaj pristup smanjuje potrebu za memorijom jer više nije potrebno pohranjivati čitav vektor, već samo jedan broj, čime se i kodiranje oznaka pojednostavljuje [21]. Nakon što model izračuna predikcije, te predikcije se transformiraju pomoću *softmax* funkcije. Predikcija u dubokom učenju odnosi se na rezultat koji model daje nakon obrade ulaznih podataka. U zadacima klasifikacije, predikcije modela predstavljaju vjerojatnosti povezane s različitim mogućim klasama. Na primjer, ako model klasificira slike životinja, predikcija može izgledati kao skup vjerojatnosti koji model pridužuje svakoj vrsti životinje (npr. 70% mačka, 20% pas, 10% zec). *Softmax* funkcija [22] pretvara predikcije modela u raspodjelu vjerojatnosti. Svaka predikcija dobiva vrijednost između 0 i 1, pri čemu zbroj svih tih vrijednosti iznosi 1. To omogućuje modelu da izrazi svoju sigurnost u kojoj klasi pripada svaki ulazni podatak. Nakon primjene *softmax* funkcije, za izračunavanje gubitka koristi se negativni logaritam vjerojatnosti točne klase. Gubitak [23] je funkcija koja

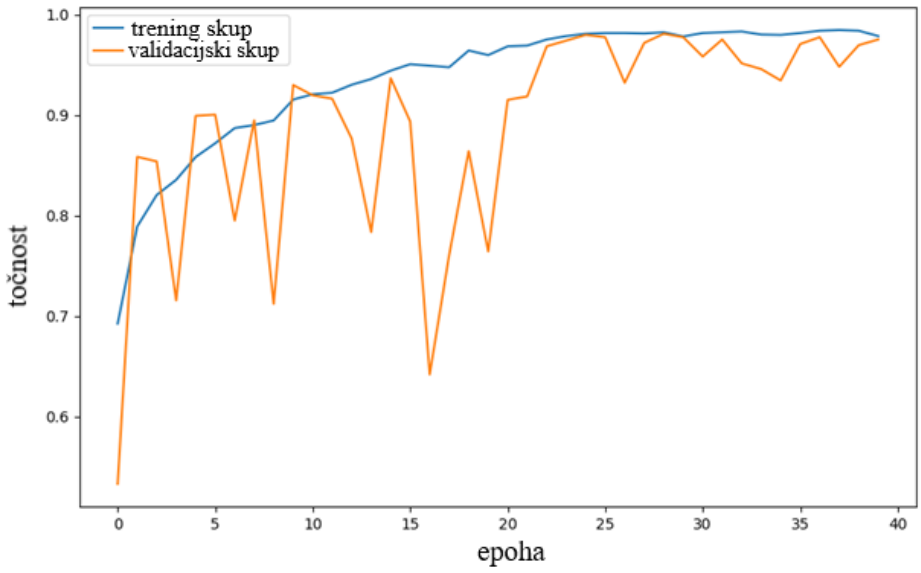
uspoređuje ciljane i predviđane izlazne vrijednosti te mjeri koliko dobro neuronska mreža uči iz podataka za treniranje. Cilj treniranja je minimizirati gubitak između predviđenih i ciljnih rezultata. U ovom radu koristi se rijetka kategorijalna unakrsna entropija kao funkcija gubitka, koja se računa prema formuli: $loss = -\sum_{i=1}^{n} \mathrm{t_i} * \log(\mathrm{p_i})$ , gdje $t_i$ predstavlja stvarnu oznaku *i*-te klase, $p_i$ je vjerojatnost koju *softmax* daje za *i*-tu klasu, *n* je broj klasa, u ovome slučaju tri klase [24]. Ova operacija omogućuje penaliziranje modela kada daje niske vjerojatnosti za ispravne klase. Ako model predvidi nisku vjerojatnost za ispravnu klasu, gubitak će biti visok, dok će u suprotnom slučaju gubitak biti nizak. Tako se model potiče na davanje visoke vjerojatnosti za točne klase kako bi se minimizirala funkcija gubitka. Nakon što je izračunat negativni logaritam vjerojatnosti za sve primjere u skupu podataka, računa se prosječni gubitak. Prosječni gubitak predstavlja ukupnu kvalitetu predikcija modela za cijeli skup podataka. Taj prosjek omogućuje optimizaciju modela na globalnoj razini, uzimajući u obzir sve primjere. Ovakav LRCN model neuronske mreže sastoji se od ukupno 1,030,691 parametara od kojih 864 parametra ne prolaze proces treniranja. U prilogu P.3.1. se nalazi objašnjena arhitektura ovoga modela definirana u *Python* programskom jeziku.

### 3.7.2. Treniranje LRCN mreže

Za potrebe treniranja neuronske mreže najprije je bilo potrebno učitati *NumPy* polja koja sadrže pred obrađene video sekvence iz kreiranog skupa podataka. Svako *NumPy* polje predstavlja jednu klasu pa je tako svakom učitanom *NumPy* polju dodijeljena oznaka klase (engl. *label*) od 0 do 2. Oznaka „0“ označava sve video sekvence klase „pritisak lijeve ruke“, „1“ označava sve video sekvence klase „nema pritiska“, te „2“ označava sve video sekvence klase „pritisak desne ruke“. Zatim, sva tri *NumPy* polja spojena su u jedno polje u kojem su potom nasumično izmiješane sve ulazne sekvence koje imaju svoje oznake klase, kako model prilikom treniranja ne bi učio prvo sve značajke prve klase pa zatim druge i treće, nego bi učio podjednako na izmiješanim podacima sve tri klase. Nakon što je testni skup već unaprijed izdvojen, kao što je objašnjeno u potpoglavlju *3.2. Izrada vlastitog skupa podataka,* u posljednjem trenutku prije ulaska u model, podaci su dodatno podijeljeni nasumičnim odabirom, pri čemu je 25% podataka izdvojeno za validaciju, dok je preostalih 75% podataka proslijeđeno modelu za treniranje. Treniranje je izvršeno više puta pri čemu se mijenjao broj epoha (engl. *epoch*) i veličina serije.

Proces treniranja izvršavan je na osobnom računalu s ugrađenom *NVIDIA GTX 1070 Ti* grafičkom karticom s 8GB VRAM-a (engl. *Video Random-Access Memory*) te je vrijeme

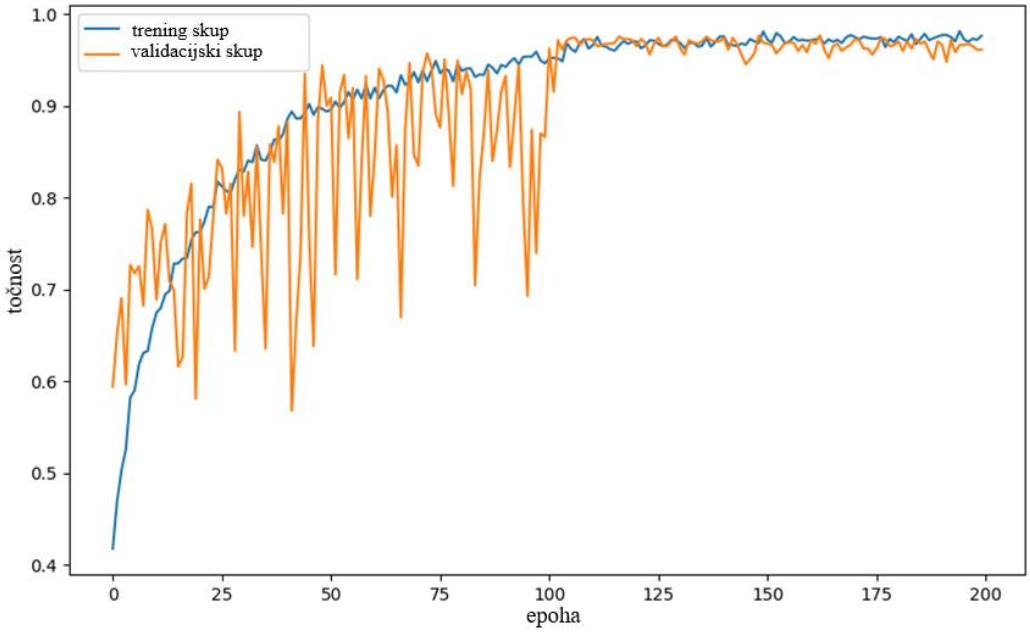
jednog treniranja iznosilo oko 90 min. U svakom eksperimentalnom koraku obavljena su višestruka treniranja modela s različitim konfiguracijama, odnosno različitim hiperparametrima. Nakon svakog koraka, na temelju evaluacijskih metrika (poput točnosti, preciznosti itd.), odabran je model s najboljim performansama. U posljednjem koraku provedeno je sedamnaest treniranja, a rezultati tih treniranja detaljno su opisani u potpoglavlju *4.1. Evaluacija i odabir najbolje treniranog modela LRCN mreže,* te je model s najboljim performansama iz navedenog koraka odabran za izradu virtualne tipkovnice ovog diplomskog rada. Prva treniranja napravljena su kao binarna klasifikacija. Naime, početna ideja bila je napraviti bazu s dvije klase od kojih je jedna „pritisak tipke“, a druga „nema pritiska“. Taj koncept je činio prvi skup podataka. Klasa „pritisak tipke“ sastojala se od 1418 video sekvenci, dok se klasa „nema pritiska“ sastojala se od 1605 video sekvenci koje su bile odvojene za treniranje i validaciju. Na slici 3.3. prikazana je krivulja točnosti na trening i validacijskom skupu modela s najboljim performansama takvog binarnog modela tijekom treninga. Točnost je metrika koja mjeri koliko često model strojnog učenja ispravno predviđa izlaze. Računa se prema formuli: $\textit{točnost}=\frac{broj\ ispravnih\ predviđanja}{ukupni\ broj\ predviđanja}$ [25].




**Slika 3.3.** Krivulja točnosti na trening i validacijskom skupu kroz epohe binarnog modela

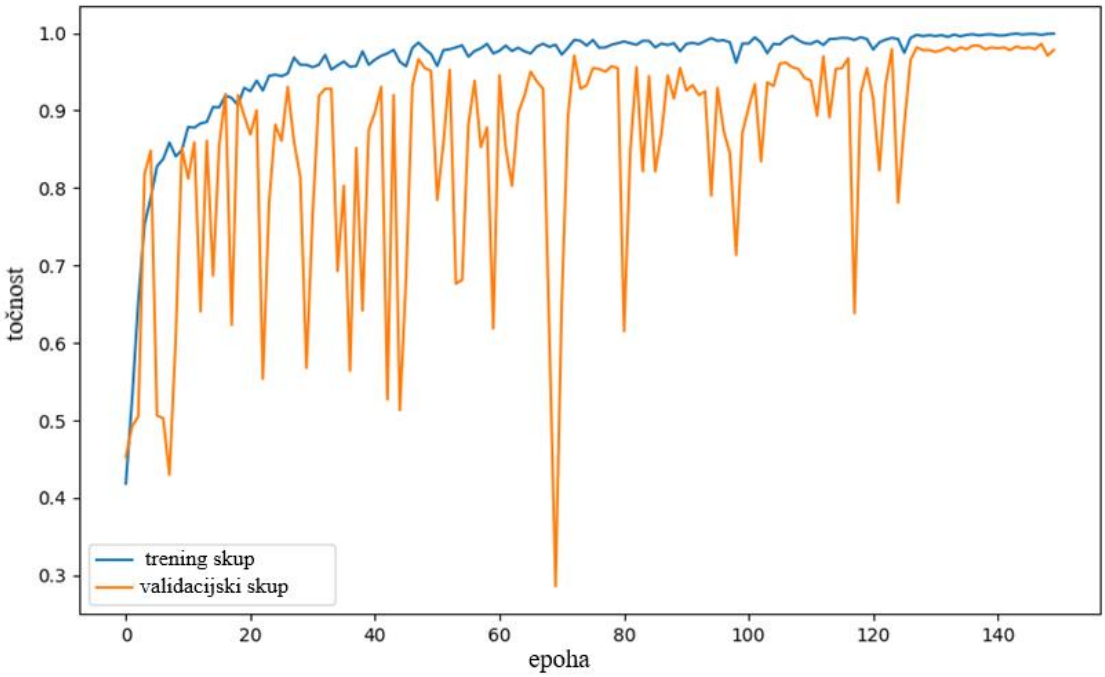
Treniranje takvog modela izvršeno je s veličinom serije 10, brojem epoha 40 te brzinom učenja (engl. *learning rate*) od 0.001. Hiperparametri su odabrani nakon eksperimentiranja s različitim kombinacijama tijekom višestrukih treniranja. Kombinacija s navedenim hiperparametrima pokazala se najboljom u smislu postignute točnosti i minimiziranja gubitka te je korištena za prikaz najboljeg rezultata treniranja korištenjem

prvog skupa podataka. Postignuta je točnost od 97.84% uz gubitak od 0.0584 na trening skupu, dok je na validacijskom skupu točnost iznosila 97.51% uz gubitak od 0.0872. Ovi rezultati predstavljaju spremljene skupove težina modela nakon 40 epoha treniranja. Model kao takav izgledao je i funkcionirao odlično, dok je korisnik virtualne tipkovnice tipkao jednom rukom. Međutim, pojavio se problem prilikom korištenja virtualne tipkovnice s obje ruke istovremeno. Naime, binarna klasifikacija daje odgovor je li se dogodio pritisak ili nije. U trenutku korištenja obje ruke model prepozna da se dogodio pritisak te se postavlja pitanje kako sa sigurnošću utvrditi koja ruka obavlja akciju pritiska. Navedeni problem pokušao se riješiti raznim algoritmima poput – detekcije koja je ruka niža u trenutku detekcije pritiska ili detekcije koja se ruka u posljednjih 15 okvira više pomjerala. Međutim, niti jedan pristup nije doveo do zadovoljavajućih rezultata odnosno odluke koja ruka obavlja pritisak. Navedeni problem je riješen razdvajanjem klase „pritisak tipke“ u dvije klase od kojih je jedna „pritisak lijeve ruke“ te druga „pritisak desne ruke“, dok klasa „nema pritiska“ ostaje ne promijenjena. Tako, umjesto binarne klasifikacije napravljena je tro-klasna klasifikacija, te je slijedom toga potrebno ponovno trenirati model neuronske mreže tro-klasne klasifikacije, koja ujedno čini drugi skup podataka. Klasa „pritisak lijeve ruke“ sastojala se od 678 video sekvenci, klasa „pritisak desne ruke“ sastojala se od 743 video sekvenci, a klasa „nema pritiska“ sastojala se od 1605 video sekvenci koje su bile odvojene za treniranje i validaciju. Na slici 3.4. prikazana je krivulja točnosti na trening i validacijskom skupu modela s najboljim performansama u tom koraku.




**Slika 3.4.** Krivulja točnosti na trening i validacijskom skupu kroz epohe tro-klasnog modela

Treniranje takvog modela izvršeno je s veličinom serije 15, brojem epoha 200 te brzinom učenja od 0.0001, te je postignuta točnost od 97.27% uz gubitak 0.0868 na trening skupu, dok je na validacijskom skupu točnost iznosila 97.14% uz gubitak od 0.0821. Važno je za napomenuti kako su navedena točnost i skup težina spremljeni na 109. epohi jer se u sljedećih 90 epoha vidi znatna stagnacija točnosti na trening i validacijskom skupu. Hiperparametri su ovdje ujedno odabrani nakon eksperimentiranja s različitim kombinacijama tijekom višestrukih treniranja. Kombinacija s navedenim hiperparametrima pokazala se najboljom u smislu postignute točnosti i minimiziranja gubitka te je korištena za prikaz najboljeg rezultata treniranja korištenjem drugog skupa podataka. Ovakvim tro-klasnim modelom riješen je problem korištenja obje ruke. Naime, prilikom predviđanja model detektira točno koja ruka pritišće tipku. Međutim, s obzirom na to kako je skup podataka prikupljen na površini jednog stola, promjena boje ili površine stola mogla bi znatno pogoršati performanse modela. Zbog toga je skup podatak proširen video sekvencama snimanim na kartonskoj podlozi koja je lako prenosiva i može se postaviti na bilo koju ravnu radnu površinu, čime se uklanja utjecaj boje stola na model neuronske mreže. Tako je napravljen krajnji skup podataka koji je opisan u potpoglavlju *3.2. Izrada vlastitog skupa podataka.* Na slici 3.5. prikazana je krivulja točnosti na trening i validacijskom skupu modela s najboljim performansama tijekom treninga.

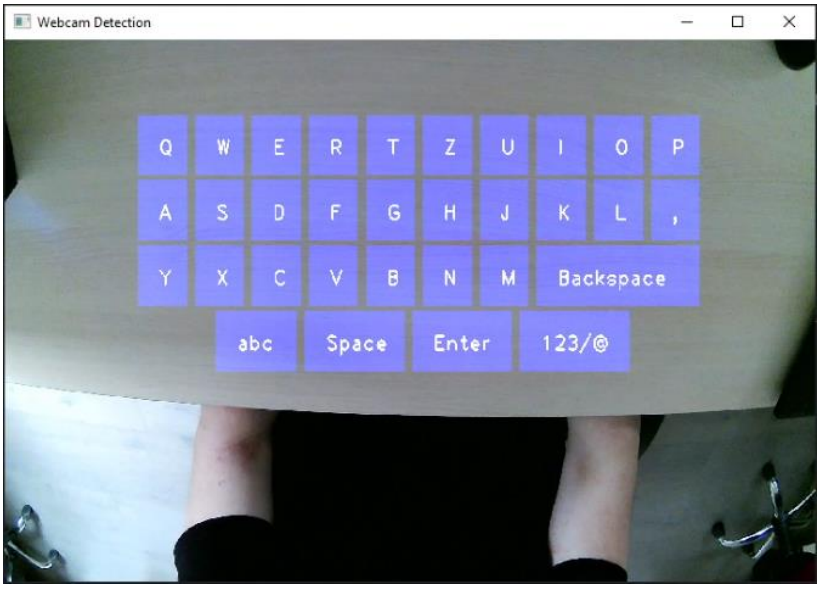



**Slika 3.5.** Krivulja točnosti na trening i validacijskom skupu kroz epohe tro-klasnog modela s najboljim performansama

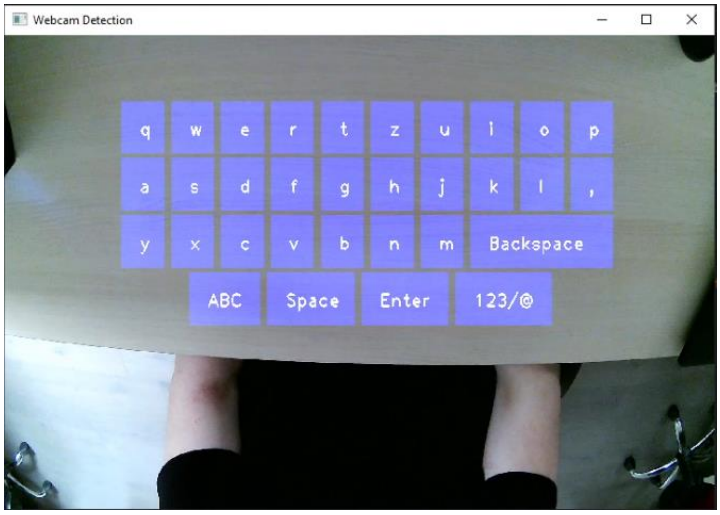
Treniranje takvog modela izvršeno je s veličinom serije 17, brojem epoha 150 te brzinom učenja od 0.001, koja se pred kraj treniranja spušta na vrijednost 0.0001 kako bi se gubitak dodatno smanjio, a točnost povećala. Hiperparametri su odabrani nakon eksperimentiranja s različitim kombinacijama tijekom višestrukih treniranja. Kombinacija s navedenim hiperparametrima pokazala se najboljom u smislu postignute točnosti i minimiziranja gubitka te je korištena za prikaz najboljeg rezultata treniranja korištenjem krajnjeg skupa podataka. Postignuta je točnost od 99.77% uz gubitak 0.0084 na trening skupu, dok je na validacijskom skupu točnost iznosila 98.61% uz gubitak od 0.091. Skup težina ovoga modela spremljen je u 147. epohi u trenutku kada se na validacijskoj krivulji dogodio posljednji vrh prije pada, kako je prikazano na slici 3.5. u zadnje tri epohe. Ovaj model u konačnici je korišten za ostvarivanje funkcionalnosti virtualne tipkovnice.

## 3.8. Algoritam virtualne tipkovnice

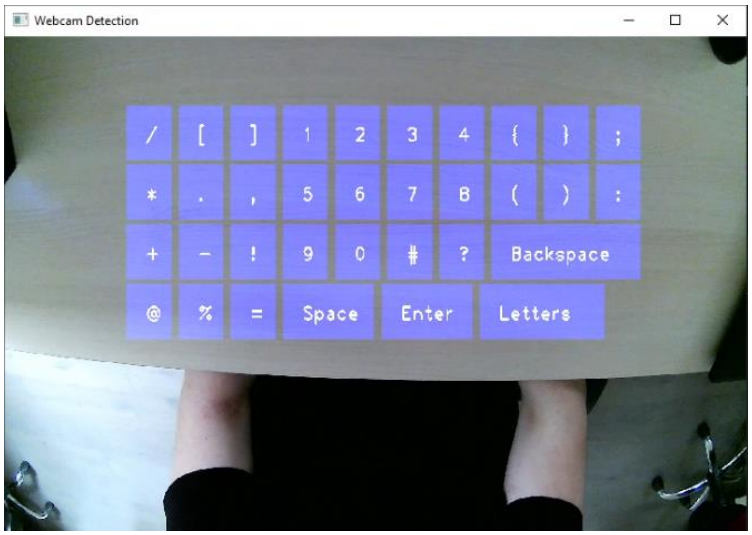
Nakon uspješnog treniranja neuronske mreže, potrebno je napraviti algoritam u *Python* programskom jeziku. Algoritam u stvarnom vremenu snima videozapise s kamere, radi pred obradu svakog pojedinog okvira kako bi odgovarali ulazu u neuronsku mrežu te prikuplja petnaest uzastopnih okvira koji se zajedno prosljeđuju modelu neuronske mreže na predviđanje. Također, algoritam grafički prikazuje transparentnu interaktivnu virtualnu tipkovnicu koja omogućuje korisniku tipkanje i unos željenih slova, znakova, brojeva ili izvršavanje određenih radnji poput *enter-a, space-a* i *backspace-a,* prikazano na slikama 3.6., 3.7. i 3.8.




**Slika 3.6.** Prikaz interaktivne virtualne tipkovnice (velika slova)



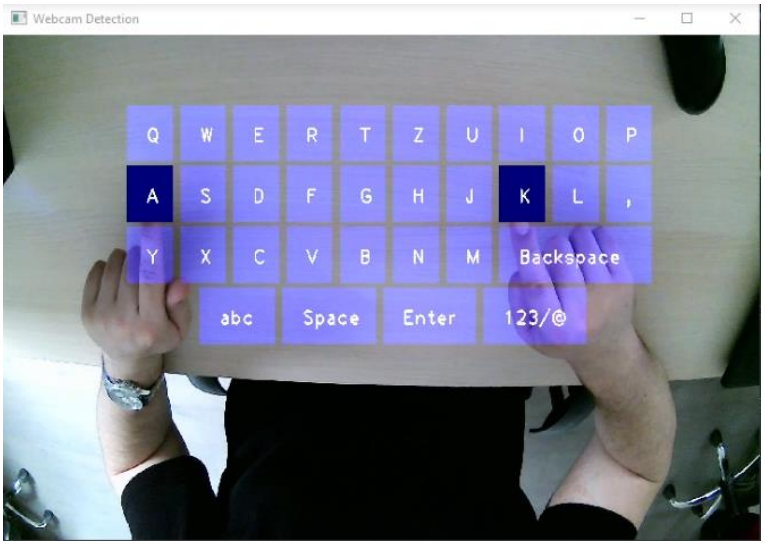

**Slika 3.7.** Prikaz interaktivne virtualne tipkovnice (mala slova)




**Slika 3.8.** Prikaz interaktivne virtualne tipkovnice (brojevi i znakovi)

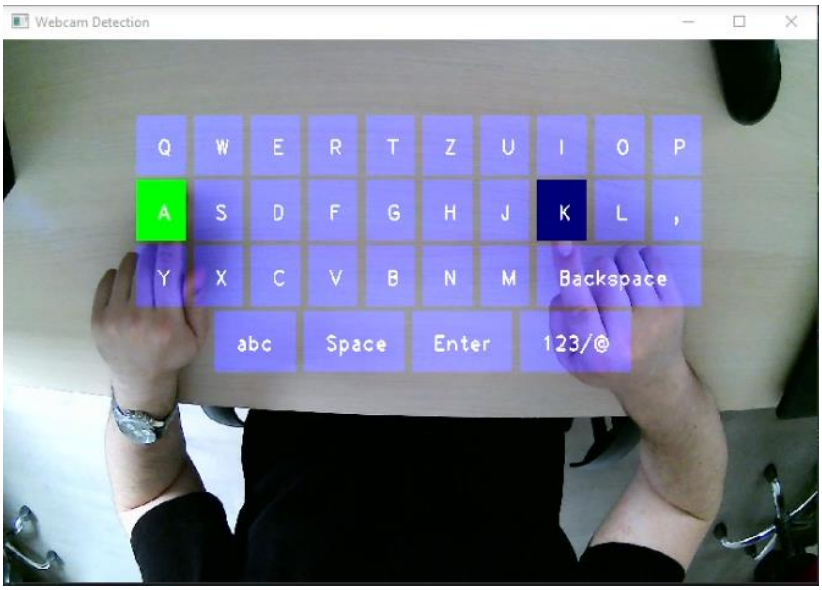
Tijek rada algoritma je sljedeći – korisniku je prikazana transparentna interaktivna virtualna tipkovnica na zaslonu monitora te korisnik gledajući poziciju svoje ruke na istom tom zaslonu odabire lokaciju zamišljenog virtualnog slova, znaka ili radnje iznad površine stola i zatim izvršava pokret pritiska takve virtualne tipke. Usporedno s time, algoritam obrađuje i hvata 15 uzastopnih okvira i prosljeđuje ih modelu neuronske mreže na predviđanje, dok ujedno *Google-ova* biblioteka za detekciju i označavanje ključnih točaka ruke *Mediapipe,* detektira korisnikovu ruku na okvirima snimljenih kamerom te pohranjuje pozicije ključnih točaka ruke od kojih je najpotrebnija pozicija vrha kažiprsta. Naime, korisniku se dodatno olakšava traženje i odabir znakova tako što se u svakom trenutku

uspoređuju pozicije vrhova kažiprsta obje ruke s pozicijama tipki virtualne tipkovnice i potamnjuju one tipke, kako je prikazano na slici 3.9., iznad koje se trenutno nalaze kažiprsti.



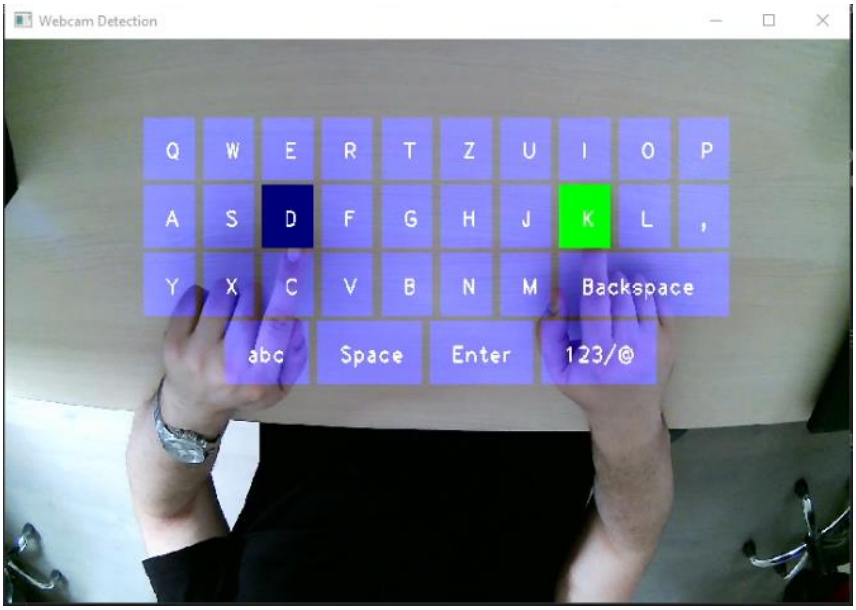

**Slika 3.9.** Prikaz potamljenih tipki na interaktivnoj virtualnoj tipkovnici

Tako na primjer, korisnik lijevom rukom pritišće tipku virtualne tipkovnice "A", zatim model predviđa da se dogodio pritisak lijeve ruke te algoritam od *Mediapipe-a* dobiva lokaciju vrha kažiprsta lijeve ruke, potom algoritam uspoređuje lokaciju kažiprsta lijeve ruke s lokacijom iscrtanih tipki virtualne tipkovnice te ako lokacija kažiprsta odgovara lokaciji tipke "A" virtualne tipkovnice, algoritam ispisuje slovo A i daje korisniku povratnu informaciju da je ta tipka pritisnuta mijenjanjem boje u zelenu u tom trenutku, kako je prikazano na slici 3.10.

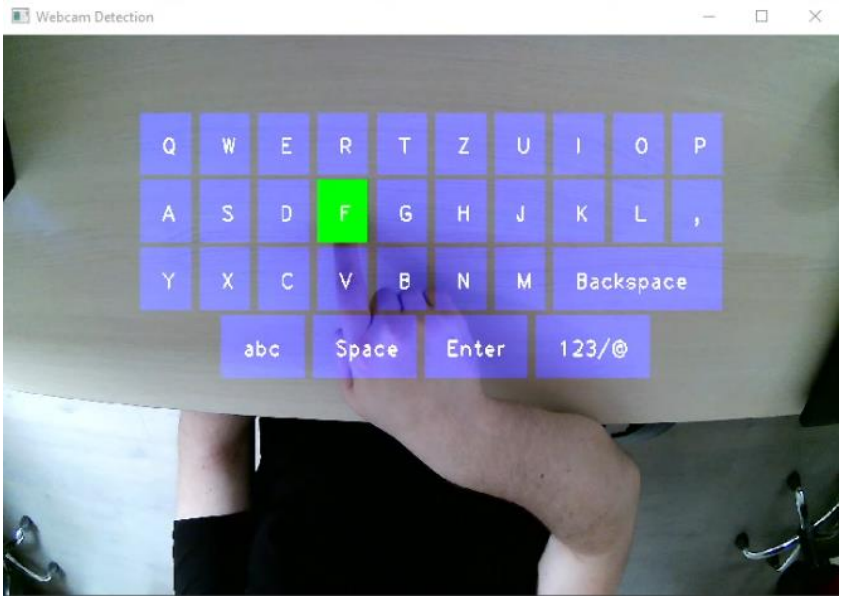



**Slika 3.10.** Prikaz pritiska tipke lijevom rukom na interaktivnoj virtualnoj tipkovnici

Osim toga, kako je prikazano na slici 3.10. na kojoj je očitan pritisak tipke lijevom rukom na interaktivnoj virtualnoj tipkovnici, pritisak tipke može se učiniti i desnom rukom, kako je prikazano na slici 3.11. Ujedno, moguće je i korištenje, odnosno tipkanje na interaktivnoj virtualnoj tipkovnici samo jednom rukom, kako je prikazano na slici 3.12.




**Slika 3.11.** Prikaz pritisnute tipke desnom rukom na interaktivnoj virtualnoj tipkovnici




**Slika 3.12.** Prikaz pritiska tipke jednom rukom na interaktivnoj virtualnoj tipkovnici

# 4. EVALUACIJA CJELOKUPNOG SUSTAVA VIRTUALNE TIPKOVNICE

Evaluacija je jedan od temeljnih koraka u procjeni rada nekog sustava i razumijevanju njegovih mogućnosti. Naime, kako je neuronska mreža jedna od najvažnijih dijelova ovoga rješenja, a ujedno je i najveća nepoznanica zbog svoje složenosti i neprozirnosti, funkcionalnosti i mogućnosti neuronske mreže mogu se evaluirati jedino raznim metrikama poput točnosti i gubitka prilikom treniranja, validacije i testiranja na testnom skupu, kao i ostalim metrikama poput preciznosti (engl. *precision*), odziva (engl. *recall*), f1-rezultata (engl. *F1-score*) i matrice konfuzije (engl. *confusion matrix*).

Preciznost je metrika koja mjeri koliko često model strojnog učenja točno predviđa pozitivnu klasu.

Računa se prema formuli: Preciznost=$\frac{broj\ ispravno\ pozitivnih\ predviđanja}{broj\ ukupno\ pozitivnih\ predviđanja}$ [25].

Odziv se odnosi na sposobnost modela neuronske mreže u otkrivanju pozitivne oznake klasa

[26]. Računa se prema formuli:

Odziv=$\frac{broj\ ispravno\ pozitivnih\ predviđanja}{Ukupan\ broj\ pozitivnih\ podataka\ iz\ skupa\ podataka}$ [25].

F1-rezultat predstavlja harmonijsku sredinu između preciznosti i odziva [25].

Računa se prema formuli: F1 rezultat=$\frac{2 * Preciznost * Odziv}{Preciznost + Odziv}$ [27].

Matrica konfuzije koristi se u strojnome učenju za procjenu učinkovitosti klasifikacijskih modela. Pruža sveobuhvatan pregled o tome koliko model uspješno predviđa rezultate. Prikazuje stvarne i predviđene rezultate klasifikacijskog modela te služi za procjenu točnosti modela, odnosno koliko dobro model razvrstava podatke u ispravne kategorije. Prilikom procjene izvedbe klasifikacijskog modela, matrica konfuzije je ključna. Nudi temeljitu analizu pravih pozitivnih (engl. *true positives, TP*), pravih negativnih (engl. *true negatives, TN*), lažno pozitivnih (engl. *false positives, FP*) i lažno negativnih (engl. *false negatives, FN*) predviđanja, olakšavajući dublje razumijevanje odziva, preciznosti i ukupne učinkovitosti modela u razlikovanju klasa. U situacijama gdje postoji neravnomjerna distribucija klasa u skupu podataka, ova matrica posebno je korisna za procjenu izvedbe modela izvan osnovne

metrike točnosti [28]. Tablica 4.1. prikazuje primjer izgleda matrice konfuzije tro-klasne klasifikacije.

Predviđena oznaka klase od strane modela

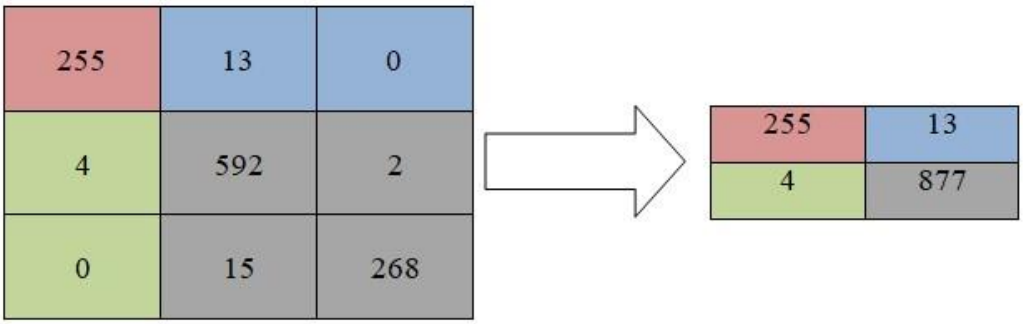
Stvarna oznaka klase

| Klasa | Pritisak lijeve ruke | Nema pritiska | Pritisak desne ruke |
|---|---|---|---|
| Pritisak lijeve ruke | 255 | 13 | 0 |
| Nema pritiska | 4 | 592 | 2 |
| Pritisak desne ruke | 0 | 15 | 268 |

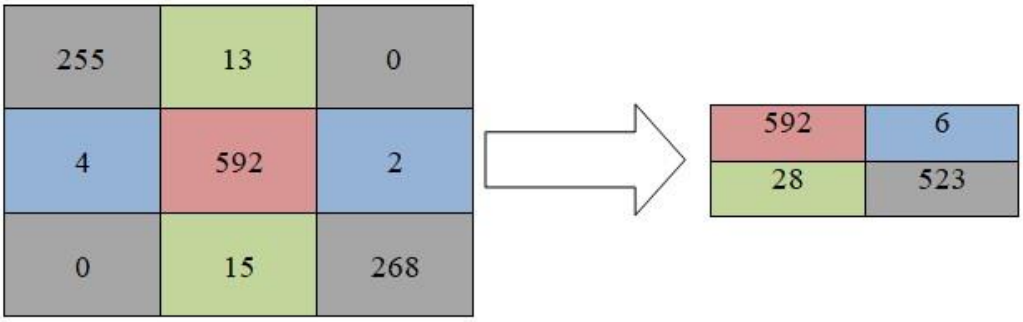
**Tablica 4.1.** Primjer izgleda matrice konfuzije tro-klasne klasifikacije

Svaka 3x3 matrica konfuzije može se svesti na matricu dimenzija 2x2 za svaku od tri klase, odnosno može se dobiti tri matrice dimenzija 2x2, kao što je prikazano na slici 4.1. Elementi označeni istom bojom zbrajaju se kako bi se formirala odgovarajuća matrica za promatranu klasu. Element označen crvenom bojom predstavlja broj ispravno klasificiranih primjera, što odgovara TP vrijednosti u matrici konfuzije za promatranu klasu. Elementi označeni plavom bojom odnose se na primjere koji su pogrešno klasificirani u promatranu klasu, odnosno predstavljaju FP primjere. Elementi označeni zelenom bojom su primjeri koji su pogrešno klasificirani u neku drugu klasu, te oni predstavljaju FN primjere. Suma preostalih elemenata, označenih sivom bojom, predstavlja broj TN primjera [30]. Tako na primjer, promatrajući klasu „pritisak lijeve ruke“ na prikazanom primjeru (slika 4.1.) može se iščitati 255 TP primjera, 13 FP primjera, 4 FN primjera i 877 TN primjera. Slično tome, promatrajući klasu „nema pritiska“, može se iščitati 592 TP primjera, 6 FP primjera, 28 FN primjera i 523 TN primjera. Na kraju, promatrajući klasu „pritisak desne ruke“, može se iščitati 268 TP primjera, 15 FP primjera, 2 FN primjera i 864 TN primjera.

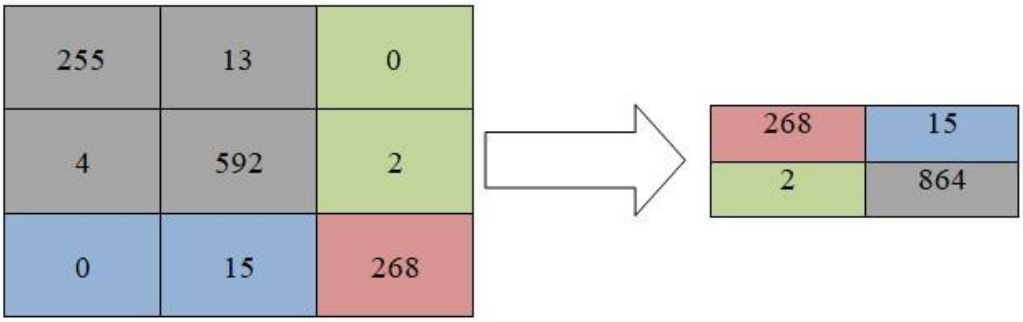
Matrica konfuzije za klasu “pritisak lijeve ruke” konstruirat će se na sljedeći način:




Matrica konfuzije za klasu “nema pritiska” konstruirat će se na sljedeći način:




Matrica konfuzije za klasu “pritisak desne ruke” konstruirat će se na sljedeći način:




**Slika 4.1.** Prikaz konstrukcije triju 2x2 matrica konfuzije iz jedne 3x3 matrice konfuzije za svaku od promatranih klasa

## 4.1. Evaluacija i odabir najbolje treniranog modela LRCN mreže

Nakon što je izrađen konačni zadovoljavajući skup podataka, provedeno je nekoliko treniranja s različitim hiperparametrima kako bi se dobio najbolji model koji će poslužiti kao rješenje za virtualnu tipkovnicu. Nakon obavljenih treniranja, prikupljeno je sedamnaest skupova težina, a točnosti i gubitci neuronske mreže na validacijskom skupu podataka,

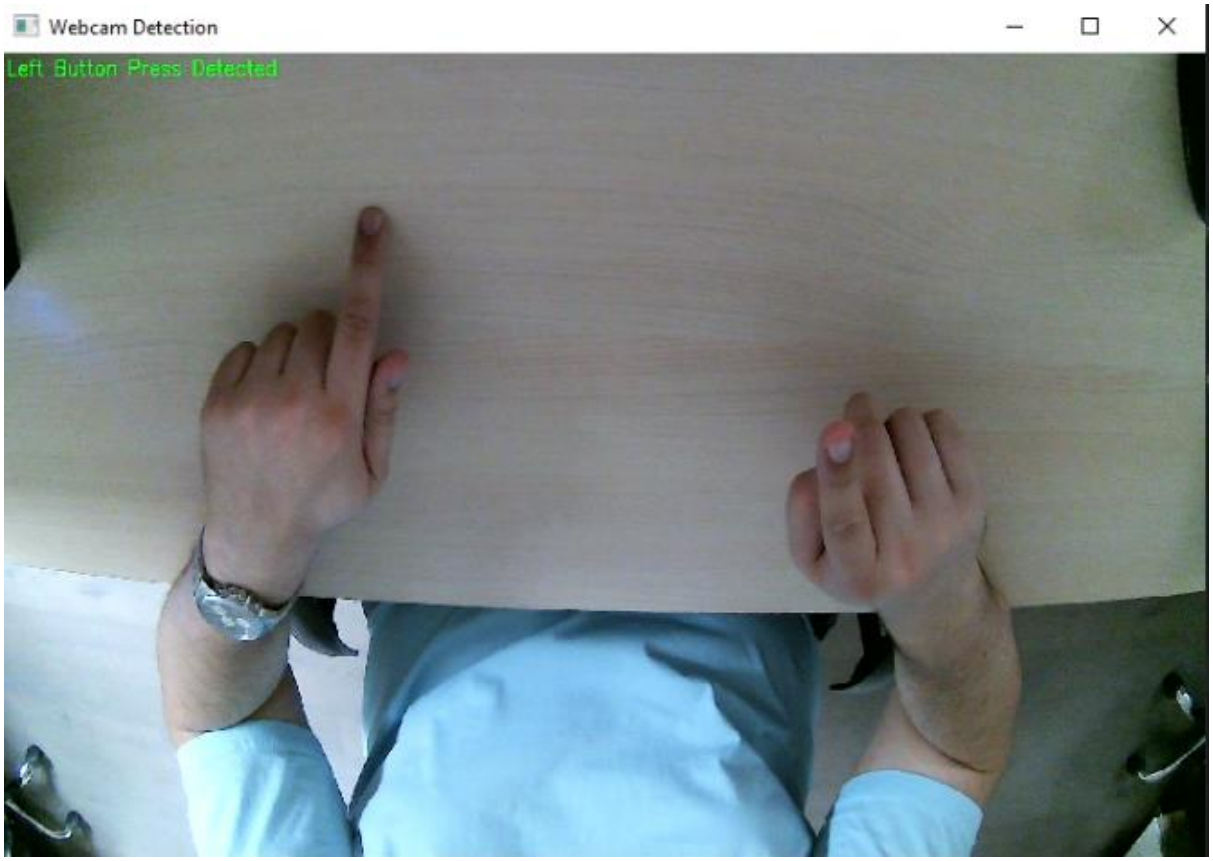
postignuti korištenjem tih skupova težina, prikazani su u tablici 4.2. Tablica prikazuje kako je neuronska mreža koja koristi skup težina pod rednim brojem 13. postigla najveću točnost na validacijskom skupu podataka, dok je korištenje nekoliko drugih skupova težina rezultiralo manjim gubicima na validacijskom skupu. Iz tog razloga, svi skupovi težina pojedinačno su ponovno učitani u model neuronske mreže, te su potom testirani na istom testnom skupu podataka kako bi se dodatno evaluirali.

| Redni broj skupova težina | Točnost na validacijskom skupu | Gubitak na validacijskom skupu |
|---|---|---|
| 1. | 96.92% | 0.1438 |
| 2. | 98.15% | 0.0705 |
| 3. | 97.22% | 0.0966 |
| 4. | 98.26% | 0.0816 |
| 5. | 98.38% | 0.0845 |
| 6. | 97.27% | 0.1079 |
| 7. | 97.63% | 0.0913 |
| 8. | 97.45% | 0.0919 |
| 9. | 98.38% | 0.0924 |
| 10. | 98.26% | 0.0877 |
| 11. | 98.49% | 0.0861 |
| 12. | 97.80% | 0.1063 |
| 13. | 98.61% | 0.0910 |
| 14. | 97.10% | 0.1273 |
| 15. | 98.26% | 0.0869 |
| 16. | 98.03% | 0.0924 |
| 17. | 98.03% | 0.0828 |

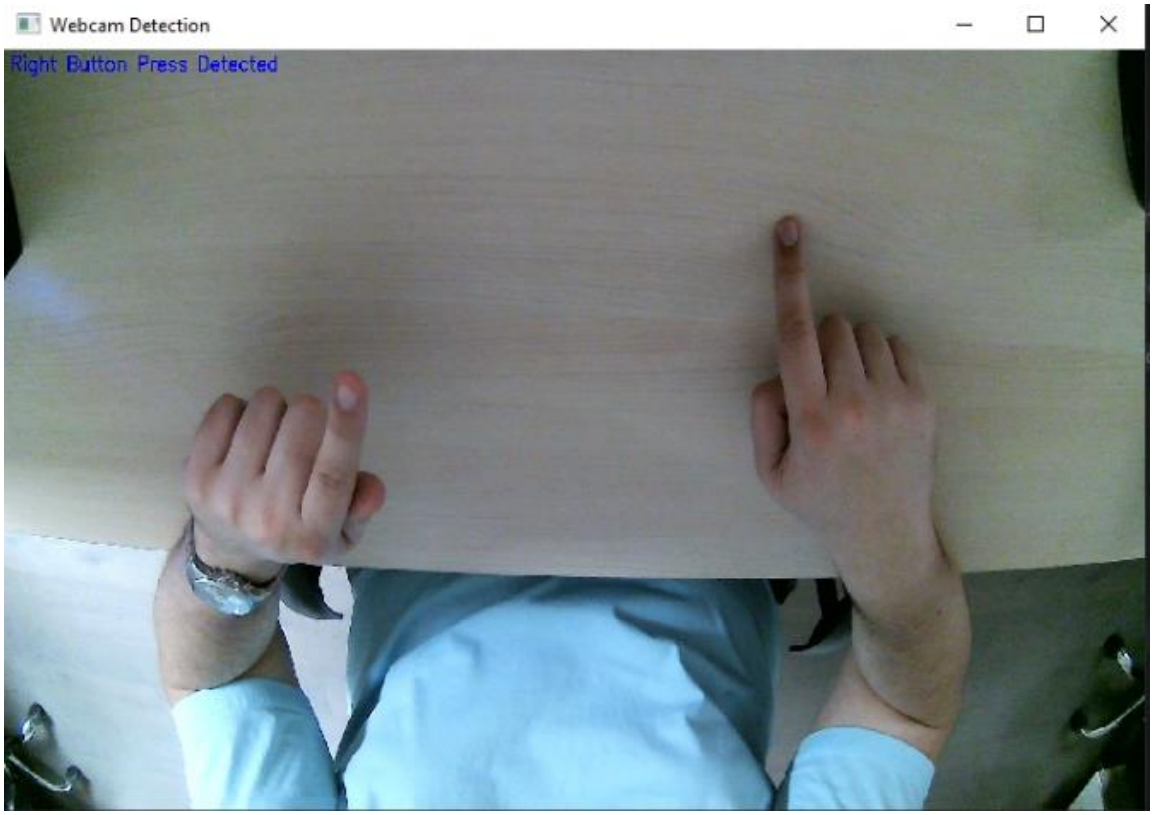
**Tablica 4.2.** Prikaz točnosti i gubitaka neuronske mreže korištenjem svih sedamnaest skupova težina na validacijskom skupu

Kao što je već objašnjeno u potpoglavlju *3.2. Izrada vlastitog skupa podataka*, 25% podataka je na samom početku bilo izdvojeno za potrebe kasnijeg testiranja. Taj skup podataka predstavlja testni skup. Testni skup je do sada bio isključen iz procesa treniranja kako bi ti podaci ostali „neviđeni“ i omogućili procjenu stvarnih mogućnosti i

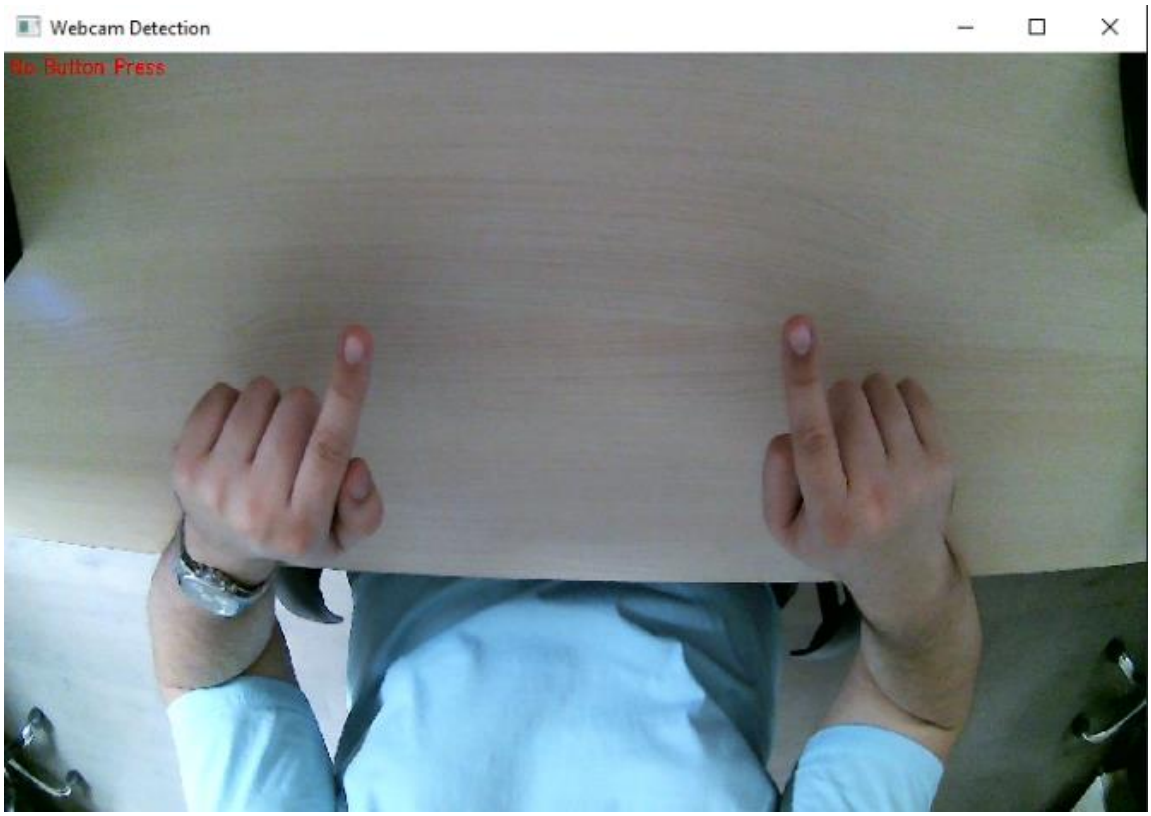
funkcionalnosti modela pri radu s novim podacima. Za potrebe testiranja, izrađena je manja *Python* skripta u koju se učitavaju željeni skupovi težina dobivenih treningom u model neuronske mreže, kao i *NumPy* polja testnog skupa, te kao rezultat dobivaju se prethodno spomenute metrike. Rezultati testiranja neuronske mreže korištenjem svih sedamnaest skupova težina na testnom skupu prikazani su u prilogu 4.1. Nakon usporedbe rezultata testiranja neuronske mreže korištenjem svih sedamnaest skupova težina, istaknuti su skupovi težina čije su neuronske mreže na testnom skupu imale gubitak manji od 0.01 i točnost veću od 98%. Među istaknutim skupovima težina, bio je i skup težina pod rednim brojem 13., čija je neuronska mreža ostvarila najveću točnost na validacijskom skupu. Dodatno, kako bi se usporedili tako istaknuti skupovi težina neuronske mreže, izrađena je još jedna dodatna manja *Python* skripta koja su stvarnom vremenu hvata okvire s kamere te ih šalje učitanom modelu neuronske mreže na predviđanje dok korisnik tipka po praznoj radnoj površini. Algoritam potom kao povratnu informaciju modela prikazuje u gornjem lijevom kutu prozora kamere detektiranu akciju: „pritisak lijeve ruke“, „pritisak desne ruke“ ili „nema pritiska“, kao što je prikazano na slikama 4.2., 4.3. i 4.4.




**Slika 4.2.** Detektiran pritisak lijeve ruke




**Slika 4.3.** Detektiran pritisak desne ruke




**Slika 4.4.** Nije detektiran pritisak

Ujedno, na temelju provedenih mjerenja, utvrđeno je kako algoritam virtualne tipkovnice može obraditi oko 15 FPS-a. Smanjenje brzine snimanja kamere sa 30 FPS-a na 15 FPS-a uzrokovano je dodatnim opterećenjem koje stvara model neuronske mreže. Zbog toga je potrebna cijela sekunda kako bi se prikupilo i obradilo 15 uzastopnih okvira, što znatno utječe na brzinu predviđanja modela.

Nakon provedenog jednostavnog testiranja u stvarnom vremenu, neki skupovi težina, čije su neuronske mreže imale veću točnost na testnom skupu, pokazali su se lošijima od onih čije su mreže imale veću točnost na validacijskom skupu. Na kraju je odabran skup težina pod rednim brojem 13., čija je neuronska mreža imala najveću točnost na validacijskom skupu, uz ostvarivanje visoke točnosti i na testnom skupu. Ujedno, u dijelu *3.7.2. Treniranje LRCN mreže,* na slici 3.5. prikazana je krivulja točnosti na trening i validacijskom skupu kroz epohe upravo tog konačnog modela s najboljim performansama.

## 4.2. Evaluacija i testiranje cjelokupnog algoritma virtualne tipkovnice u realnim uvjetima

Evaluacija i testiranje virtualne tipkovnice u realnim uvjetima provedeni su testiranjem sedam korisnika, podijeljenih u tri skupine. Naime, prvi korisnik koji je testirao rad virtualne tipkovnice je osoba koja je virtualnu tipkovnicu izradila, najviše je s njom upoznata te već ima potrebno iskustvo tipkanja na istoj. Zatim, druga dva korisnika koja su sudjelovala u testiranju, ujedno su sudjelovali i u kreiranju skupa podataka što znači da je model neuronske mreže već upoznat s pokretima i izgledom tih ruku te će ih vjerojatno lakše prepoznati, međutim navedeni korisnici nemaju iskustvo u tipkanju po virtualnoj tipkovnici. Naposljetku četiri preostala korisnika nisu niti sudjelovali u kreiranju skupa podataka, niti imaju bilo kakvog iskustva s tipkanjem po virtualnoj tipkovnici.

Svaki od korisnika dobio je zadatak natipkati određenu rečenicu odnosno pangram (*Pangram* je rečenica koja sadrži sva slova, u ovom slučaju engleske abecede) korištenjem jedne te zatim obje ruke, kao i s kartonskom podlogom te bez nje - kako bi se ispitale performanse virtualne tipkovnice u svakoj stvarnoj situaciji.

U tablici 4.3. pikazani su rezultati testiranja virtualne tipkovnice korištenjem jedne ruke bez kartonske podloge sa zadanom rečenicom "*Amazingly few discotheques provide jukeboxes" :*

| Jedna ruka<br>Bez kartonske podloge | Osoba<br>1. | Osoba<br>2. | Osoba<br>3. | Osoba<br>4. | Osoba<br>5. | Osoba<br>6. | Osoba<br>7. |
|---|---|---|---|---|---|---|---|
| Potrebno vrijeme | 1 min<br>56 sec | 2 min<br>31 sec | 2 min<br>31 sec | 3 min<br>6 sec | 5 min<br>12 sec | 5 min<br>24 sec | 3 min<br>37 sec |
| Algoritam ne prepoznaje Pritisak tipke (FN) | 0 | 0 | 0 | 3 | 9 | 3 | 6 |
| Algoritam prepoznaje pritisak tipke kada se on nije dogodio (FP) | 0 | 0 | 1 | 0 | 3 | 4 | 2 |
| Algoritam detektira pritisak tipke krive ruke | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Korisnik promašuje slovo | 1 | 1 | 2 | 0 | 3 | 1 | 0 |
| *Mediapipe* dovodi do krivog unosa zbog nesigurne pozicije kažiprsta | 0 | 1 | 1 | 0 | 2 | 3 | 0 |

**Tablica 4.3.** Rezultati testiranja korištenjem jedne ruke bez kartonske podloge

U tablici 4.4. pikazani su rezultati testiranja virtualne tipkovnice korištenjem jedne ruke s kartonskom podlogom sa zadanom rečenicom "*Amazingly few discotheques provide jukeboxes" :*

| Jedna ruka<br>+ kartonska podloga | Osoba<br>1. | Osoba<br>2. | Osoba<br>3. | Osoba<br>4. | Osoba<br>5. | Osoba<br>6. | Osoba<br>7. |
|---|---|---|---|---|---|---|---|
| Potrebno vrijeme | 2 min<br>14 sec | 1 min<br>54 sec | 2 min<br>29 sec | 2 min<br>46 sec | 3 min<br>21 sec | 4 min<br>31 sec | 3 min<br>57 sec |
| Algoritam ne prepoznaje Pritisak tipke (FN) | 0 | 0 | 0 | 2 | 3 | 6 | 6 |
| Algoritam prepoznaje pritisak tipke kada se on nije dogodio (FP) | 0 | 0 | 3 | 0 | 0 | 5 | 1 |
| Algoritam detektira pritisak tipke krive ruke | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Korisnik promašuje slovo | 0 | 0 | 3 | 1 | 3 | 2 | 1 |
| *Mediapipe* dovodi do krivog unosa zbog nesigurne pozicije kažiprsta | 4 | 1 | 4 | 0 | 4 | 7 | 1 |

**Tablica 4.4.** Rezultati testiranja korištenjem jedne ruke s kartonskom podlogom

U tablici 4.5. pikazani su rezultati testiranja virtualne tipkovnice korištenjem obje ruke bez kartonske podloge sa zadanom rečenicom:*"The quick brown fox jumps over the lazy dog":*

| Dvije ruke<br>Bez kartonske podloge | Osoba<br>1. | Osoba<br>2. | Osoba<br>3. | Osoba<br>4. | Osoba<br>5. | Osoba<br>6. | Osoba<br>7. |
|---|---|---|---|---|---|---|---|
| Potrebno vrijeme | 1 min<br>41 sec | 2 min<br>28 sec | 2 min<br>11 sec | 3 min<br>21 sec | 2 min<br>43 sec | 3 min<br>9 sec | 3 min<br>18 sec |
| Algoritam ne prepoznaje Pritisak tipke (FN) | 3 | 3 | 6 | 7 | 11 | 5 | 8 |
| Algoritam prepoznaje pritisak tipke kada se on nije dogodio (FP) | 0 | 3 | 2 | 0 | 1 | 5 | 4 |
| Algoritam detektira pritisak tipke krive ruke | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| Korisnik promašuje slovo | 1 | 2 | 2 | 1 | 0 | 0 | 0 |
| *Mediapipe* dovodi do krivog unosa zbog nesigurne pozicije kažiprsta | 0 | 0 | 2 | 0 | 1 | 1 | 0 |

**Tablica 4.5.** Rezultati testiranja korištenjem obje ruke bez kartonske podloge

U tablici 4.6. pikazani su rezultati testiranja virtualne tipkovnice korištenjem obje ruke s kartonskom podlogom sa zadanom rečenicom:*"The quick brown fox jumps over the lazy dog":*

| Dvije ruke<br>+ kartonska podloga | Osoba<br>1. | Osoba<br>2. | Osoba<br>3. | Osoba<br>4. | Osoba<br>5. | Osoba<br>6. | Osoba<br>7. |
|---|---|---|---|---|---|---|---|
| Potrebno vrijeme | 1 min<br>52 sec | 2 min<br>44 sec | 2 min<br>49 sec | 4 min<br>24 sec | 2 min<br>20 sec | 3 min<br>16 sec | 4 min<br>3 sec |
| Algoritam ne prepoznaje Pritisak tipke (FN) | 2 | 1 | 7 | 1 | 6 | 5 | 2 |
| Algoritam prepoznaje pritisak tipke kada se on nije dogodio (FP) | 0 | 5 | 3 | 2 | 2 | 10 | 1 |
| Algoritam detektira pritisak tipke krive ruke | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Korisnik promašuje slovo | 0 | 0 | 1 | 1 | 2 | 2 | 4 |
| *Mediapipe* dovodi do krivog unosa zbog nesigurne pozicije kažiprsta | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

**Tablica 4.6.** Rezultati testiranja korištenjem obje ruke s kartonskom podlogom

Tijekom testiranja mjerilo se vrijeme potrebno za tipkanje cijele zadane rečenice, te se bilježilo ako algoritam nije prepoznao pritisak tipke (FN), ako je algoritam prepoznao pritisak tipke kada se on nije dogodio (FP), ako je algoritam detektirao pritisak krive ruke, zatim ako je korisnik promašio slovo, što je utjecalo na pogrešan unos i dodatno produžilo vrijeme tipkanja, te je *Mediapipe* doveo do pogrešnog unosa zbog nesigurne pozicije kažiprsta. Naime, *Mediapipe* točno prikazuje pozicije svih ključnih točaka ruke ako su sve one vidljive. Međutim, položaj ruke i način tipkanja često skrivaju neke ključne točke, zbog čega *Mediapipe* biblioteka procjenjuje položaj skrivenih ključnih točaka. Takva procjena može utjecati i na vidljive točke, što može dovesti do pomicanja pozicije vrha kažiprsta na mjesto susjedne tipke, uzrokujući tako unos pogrešne tipke. Prikazane tablice (od 4.3. do 4.6.) i rezultati svih testiranja pokazuju kako ovo rješenje virtualne tipkovnice zahtijeva dosta vremena za tipkanje relativno kratkih rečenica. Primjerice, rečenica "*The quick brown fox jumps over a lazy dog*" sadrži 43 znaka, uključujući razmake, dok rečenica "*Amazingly few discotheques provide jukeboxes*" sadrži 44 znaka, također uključujući razmake. Testiranje u stvarnom vremenu obavljeno je korištenjem ranije spomenute web kamere, te kako je utvrđeno u prethodnom potpoglavlju, zbog dodatnog opterećenja koje stvara model kamera snima u 15 FPS-a. Također, kada bi došlo do pritiska tipke, cijeli međuspremnik (engl. *buffer*) bi se ispraznio, a zatim bi se čekala dodatna sekunda za ponovno punjenje 15 uzastopnih okvira. U suprotnom, ako pritiska tipke nije bilo, najstarijih 5 okvira bi se izbacilo, a zatim bi se čekalo 1/3 sekunde za snimanje novih 5 okvira, koji bi se ponovno slali modelu. Upravo je to uzrok sporog tipkanja, koje bi se moglo riješiti boljom optimizacijom modela, jačim računalnim hardverom ili kamerom koja snima više okvira po sekundi. Osim toga, ostali rezultati koji su se proučavali pokazali su sljedeće: "Algoritam ne prepoznaje pritisak" često se pojavljuje, ali ne predstavlja veći problem jer korisnik jednostavno treba ponoviti pritisak. S druge strane, "Algoritam prepoznaje pritisak tipke kada se on nije dogodio" predstavlja problem jer se unese pogrešna tipka pa korisnik mora koristiti dodatnu tipku *backspace* kako bi ispravio unos i ponovio pritisak ispravne tipke. Međutim, kako je prikazano u tablicama, unos pogrešne tipke nije česta pojava; značajniji broj ovih slučajeva zabilježen je samo kod jednog korisnika od ukupno sedam testiranih. "Algoritam detektira pritisak tipke krive ruke" događa se vrlo rijetko, samo tri puta tijekom provedenog testiranja. Nadalje, "Korisnik promašuje slovo" događa se zbog toga što tipkanje na virtualnoj tipkovnici zahtijeva određeno vrijeme privikavanja, što utječe na vrijeme tipkanja jer korisnik mora brisati pogrešan unos i ponoviti unos ispravne tipke korištenjem *backspace*-a. Konačno, mjerenja i pogreška uzrokovana *Mediapipe* bibliotekom, odnosno ponekad dolazi do

pomicanja pozicije kažiprsta na mjesto susjedne tipke, što rezultira unosom pogrešne tipke. Sveukupno, na temelju dobivenih rezultata testiranja, ovo rješenje virtualne tipkovnice funkcionira relativno dobro, ali najveći problem predstavlja upravo vrijeme potrebno za tipkanje, što bi se moglo riješiti napretkom tehnologije ili optimizacijom modela neuronske mreže.

# 5. ZAKLJUČAK

U ovome radu, uspješno je prikazana izrada virtualne tipkovnice koja koristi već ugrađenu web kameru te ravnu radnu površinu, omogućujući korisnicima intuitivno tipkanje. Razvijeno je rješenje koje je slično fizičkoj tipkovnici te je za funkcionalnost ključna LRCN mreža. Navedena neuronska mreža trenirana je za prepoznavanje pritiska lijeve ili desne ruke te za detekciju kada pritiska nema. Za potrebe ovog rada prikupljen je i izrađen poseban skup podataka kako bi model bio što precizniji i točniji. Nakon uspješnog treniranja, razvijen je algoritam u *Python* programskom jeziku koji u stvarnom vremenu obrađuje videozapise s kamere te predviđa pritiske tipki koristeći petnaest uzastopnih okvira. Algoritam prikazuje interaktivnu virtualnu tipkovnicu koji omogućuje korisnicima unos slova, znakova, brojeva te izvršavanje određenih akcija poput *enter*-a, razmaka ili brisanja. Evaluacija i testiranje virtualne tipkovnice provedeni su na sedam korisnika, podijeljenih u tri skupine. Prvi korisnik je izradio tipkovnicu i ima iskustvo s njom. Dva sljedeća korisnika su sudjelovala u kreiranju skupa podataka, ali nemaju iskustvo tipkanja na virtualnoj tipkovnici. Preostala četiri korisnika nisu sudjelovala u kreiranju skupa podataka niti imaju iskustva s virtualnom tipkovnicom. Virtualna tipkovnica pokazuje dobru funkcionalnost, a posebno se ističe intuitivno tipkanje na ravnoj površini uz pomoć web kamere, kao i pouzdano prepoznavanje pritiska tipki, koristeći neuronsku mrežu. Pažljivo i precizno prikupljen skup podataka značajno je utjecao na ostvarivanje visoke točnosti modela. Iako je zabilježeno sporo vrijeme tipkanja, uzrokovano performansama modela neuronske mreže, postoje realne mogućnosti za optimizaciju i poboljšanje ovog segmenta algoritma.

# LITERATURA

[1]H. Sharma, R. Saxena, S. Kumar, A. Saini, M. Saadand M. Faraz, „*VirtualHands: Real Time Keyboard, Desktop & Application Navigation using Gestures*“ 2022 International Conference on Fourth Industrial Revolution Based Technology and Practices (ICFIRTP), Uttarakhand, India, 2022, pp. 262-267, doi: 10.1109/ICFIRTP56122.2022.10059450.

[2]S. Singla, K.Banta, N. Jain, „*Virtual keyboard using Image Processing*“, Computer Science, 2019.

[3] P. Khare, K. Satya Ram, S. Sourabhand A. Mahender, „*QWERTY Keyboard in Virtual Domain Using Image Processing*“ 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 2019, pp. 237-242, doi: 10.1109/ICCS45141.2019.9065459.

[4]P.Divya, P. Ayeswariya, M. Divesh, K. Arthi , „*Eyeblink Controlled virtualkey board using Raspberry Pi*“, International Research Journal of Engineeringand Technology,2020.

[5]T. -H. Lee, S. Kim, T. Kim, J. -S. Kim and H. -J. Lee, „*Virtual Keyboards With Real-Time and Robust Deep Learning-Based Gesture Recognition*“in IEEE Transactions on Human-Machine Systems, vol. 52, no. 4, pp. 725-735, Aug. 2022, doi: 10.1109/THMS.2022.3165165.

[6] „*TensorFlow: An end-to-end platform for machine learning*“,dostupno na: https://www.tensorflow.org/guide, [pristupljeno: 17.lipnja 2024]

[7] „*Keras: Deep Learning for humans*“,dostupno na: https://keras.io/guides/, [pristupljeno: 17.lipnja 2024]

[8] „*NumPy*“, dostupno na: https://numpy.org/about/, [pristupljeno: 17.lipnja 2024]

[9] „*OpenCV– Open ComupterVisionLibrary*“, dostupno na: https://opencv.org/about/ [pristupljeno: 17.lipnja 2024]

[10] „*Pynput*“, dostupno na: https://pypi.org/project/pynput/, [pristupljeno: 17.lipnja 2024

[11] A. Alavi, „*A Review of Google's New Mobile-Friendly AI Framework: Mediapipe*“ ,dostupno na: https://medium.com/swlh/a-review-of-googles-new-mobile-friendly-ai-framework-mediapipe-25d62cd482a1, [pristupljeno: 29.srpnja 2024]

[12] „*Scikit-learn: machine learning in Python*“, dostupno na: https://scikit-learn.org/stable/index.html, [pristupljeno: 17.lipnja 2024]

[13] „*Matplotlib- Visualization with Python*“, dostupno na: https://matplotlib.org/, [pristupljeno: 17.lipnja 2024]

[14] „*NVIDIA CUDA Toolkit – free tools and training*“, dostupno na: https://developer.nvidia.com/cuda-toolkit, [pristupljeno: 17.lipnja 2024]

[15] „*NVIDIA cuDNN – Deep Neural Network*“, dostupno na: https://developer.nvidia.com/cudnn, [pristupljeno: 17.lipnja 2024]

[16] CLOUDFLARE, “*What is neural network?*”, 2024., dostupno na: https://www.cloudflare.com/learning/ai/what-is-neural-network/, [pristupljeno: 15. kolovoza 2024.]

[17] D. Cornelisse, „*An intuitiveguide to Convolutional Neural Network*”, FreeCodeCamp, 2018., dostupno na: https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/, [pristupljeno: 15. kolovoza 2024.]

[18] GeeksforGeeks, „*Introduction to Recurrent Neural Network”,* 2024., dostupno na: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/, [pristupljeno: 15.kolovoza 2024.]

[19] GeeksforGeeks, „*Whatis LSTM – Long Short Term Memory?*”, 2024., dostupno na: https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/, [pristupljeno: 15.kolovoza 2024.]

[20] J. Donahue, L. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, T. Darrell, „*Long-term Recurrent Convolutional Networks for Visual Recognition and Description*”, 2016., dostupno na: https://arxiv.org/pdf/1411.4389, [pristupljeno: 15.kolovoza 2024.]

[21] S.Chand, „*Choosing between Cross Entropy and Sparse Cross Entropy – The Only Guide you Need!*“, dostupno na: https://medium.com/@shireenchand/choosing-between-cross-entropy-and-sparse-cross-entropy-the-only-guide-you-need-abea92c84662, [pristupljeno: 17.kolovoza 2024.]

[22] H. Phillips, „*A Simple Introduction to Softmax*“, Medium, 2023., dostupno na: https://medium.com/@hunter-j-phillips/a-simple-introduction-to-softmax-287712d69bac, [pristupljeno: 17.kolovoza 2024.]

[23]V. Yathish, „*Loss Functions and Their Use In Neural Networks*“, Towards Data Science, 2022., dostupno na: https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9, [pristupljeno: 17.kolovoza 2024.]

[24]S. Kedia, P. Nath, „*Here is what you need to know about Sparse Categorical Cross Entropy in nutshell*“, MachineLearningDiaries, 2022. , dostupno na: https://vevesta.substack.com/p/here-is-what-you-need-to-know-about, [pristupljeno: 17.kolovoza 2024.]

[25] Evidently AI Team, *„Accuracy vs. precision vs. recall in machine learning: what's the difference?“*, EVIDENTLY AI – Collaborative AI obervability platform, dostupno na: https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall?fbclid=IwZXh0bgNhZW0CMTAAAR0PyLPQQveZgq-Cmjq9dAA3bAyNEJmcQ-FxOV630utsgt1cSC4AUYAlEis_aem_Z7DnrnXqymwwe1IwX5xFmA#:~:text=Precision%20shsho%20how%20often%20an,objects%20of%20the%20target%20class, [pristupljeno: 29.lipnja 2024.]

[26] Educative, S. M. A. Shah, *„Precision vs. recall vs. accuracy in neural networks“*, Educative: AI-powered learning, 2024. dostupno na: https://www.educative.io/answers/precision-vs-recall-vs-accuracy-in-neural-networks, [pristupljeno: 27.lipnja 2024.]

[27] R. Kundu, *„F1 Score in Machine Learning: Intro & Calculation“*, V7 – Turnimages, video & documents into trusthworty AI, 2022., dostupno na: https://www.v7labs.com/blog/f1-score-guide, [pristupljeno: 29.lipnja 2024.]

[28] GeeksforGeeks, *„Confusion Matrix in Machine Learning“*, GeeksforGeeks - A computer science portal for geeks, 2024., dostupno na: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/, [pristupljeno: 27.lipnja 2024.]

[29] A. Mishra, *„Metrics to Evaluate your Machine Learning Algorithm“*, Towards Data Science, 2018., dostupno na: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234, [pristupljeno: 27.lipnja 2024.]

[30] T. Đuričić, A. Marćep, *Strojno učenje – bilješke s predavanja*, Sveučilište u Zagrebu, Fakultet Elektrotehnike i Računarstva, 2015./2016., dostupno na: https://www.fer.unizg.hr/_download/repository/SU-2015-Vrednovanje_modela.pdf, [pristupljeno: 18. kolovoza 2024.]

# SAŽETAK

U ovom diplomskom radu opisana je izrada virtualne tipkovnice koja koristi kameru za prepoznavanje pritisaka prstiju na ravnoj radnoj površini. Sustav omogućuje tipkanje bez fizičke tipkovnice, uz minimalnu potrebnu opremu i jednostavno održavanje. LRCN mreža prepoznaje pritiske lijeve i desne ruke te razlikuje situacije kada nema pritiska. Za potrebe treniranja neuronske mreže, kreiran je novi skup podataka s raznovrsnim slikama ruku jer postojeći nisu odgovarali potrebama ovoga rada. Algoritam, razvijen u *Python* programskom jeziku, u stvarnom vremenu obrađuje videozapise, prikuplja petnaest uzastopnih okvira i koristi neuronsku mrežu za prepoznavanje pritisaka tipki. Grafički prikaz omogućuje interaktivno tipkanje slova, znakova, brojeva i osnovnih funkcija kao što su *enter*, *space* i *backspace*. Model je treniran i evaluiran na zasebnom testnom skupu, a najbolji model odabran je na temelju performansi. Testiranje s tri skupine korisnika pokazalo je funkcionalnost i pouzdanost tipkanja, uz nedostatak kao što je sporost tipkanja zbog vremenskih zahtjeva modela i kamere.

**Ključne riječi:** virtualna tipkovnica, skup podataka, strojno učenje, klasifikacija, LRCN model

# VIRTUAL FLOATING BASED CAMERA KEYBOARD


## ABSTRACT

In this thesis the creation of a virtual keyboard is described, that uses a camera to recognize finger presses on a flat working surface. The system enables typing without a physical keyboard, requiring minimal equipment and simple maintenance. An LRCN network recognizes presses from the left and right hands and distinguishes situations where there is no press. For the purpose of training the neural network, a new dataset was created with diverse hand images, as existing ones did not meet the needs of this work. The algorithm, developed in the *Python* programming language, processes videos in real-time, collects fifteen consecutive frames, and uses a neural network to recognize key presses. A graphical interface allows interactive typing of letters, symbols, numbers and basic functions such as enter, space and backspace. The model was trained and evaluated on a separate test set, with the best model selected based on performance. Testing with three user groups demonstrated the functionality and reliability of typing, with a drawback being the slow typing speed due to the time requirements of the model and the camera.



**Keywords:** virtual keyboard, dataset, machine learning, classification, LRCN model

# ŽIVOTOPIS

Fabijan Abjanović, student 2. godine Diplomskog sveučilišnog studija Automobilsko računarstvo i komunikacije. Rođen u Rijeci 27. srpnja 1997. godine. Osnovnu školu Antuna Bauera završava u Vukovaru 2012. godine., a Srednju tehničku školu Nikole Tesle u Vukovaru, smjer elektrotehnika, s odličnim uspjehom uz završni rad „Sklopnici i elementi daljinskog upravljanja“ završava 2016. godine. Nakon srednje škole upisuje sveučilišni preddiplomski studij elektrotehnike na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Na 2. godini studija opredjeljuje se na smjer Komunikacije i informatika što i završava 2022. godine uz završni rad „Točkasto zavarivanje baterija“. Iste godine upisuje sveučilišni diplomski studij Automobilsko računarstvo i komunikacije te ujedno na prvoj godini studija postaje stipendist tvrtke TTTechAuto d.o.o.

# PRILOZI

**Prilog 3.1**. Struktura LRCN mreže prikazana u *Python* kodu.

```
model = Sequential()
model.add(Conv3D( filters: 16,  kernel_size: (3, 3, 3), input_shape=(patch_size, img_cols, img_rows, 1), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling3D(pool_size=(2, 2, 2)))
model.add(Dropout(0.25))
model.add(Conv3D( filters: 32,  kernel_size: (3, 3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling3D(pool_size=(1, 2, 2)))
model.add(Dropout(0.25))
model.add(Conv3D( filters: 64,  kernel_size: (3, 3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling3D(pool_size=(1, 2, 2)))
model.add(Dropout(0.25))
model.add(Conv3D( filters: 128,  kernel_size: (3, 3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling3D(pool_size=(1, 2, 2)))
model.add(Dropout(0.25))
model.add(ConvLSTM2D(filters=64, kernel_size=(3, 3), padding='same', return_sequences=True))
model.add(BatchNormalization())
model.add(ConvLSTM2D(filters=64, kernel_size=(3, 3), padding='same', return_sequences=True))
model.add(BatchNormalization())
model.add(GlobalAveragePooling3D())
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense( units: 3, activation='softmax'))
```

**Prilog 4.1.** Prikaz rezultata svih pojedinačnih 17 skupova težina, učitanih u model neuronske mreže i testiranih na testnom skupu

1. Test Loss 14.99%
   Test Accuracy 97.13%

$$\begin{bmatrix} 255 & 13 & 0 \\ 4 & 592 & 2 \\ 0 & 15 & 268 \end{bmatrix} \text{matrica konfuzije}$$

| klasa | Preciznost | odziv | F1 - rezultat |
|---|---|---|---|
| Pritisak lijevom rukom | 98% | 95% | 97% |
| Nema pritiska | 95% | 99% | 97% |
| Pritisak desnom rukom | 99% | 95% | 97% |

2. Test Loss 9.84%
   Test Accuracy 97.74%

$$\begin{bmatrix} 257 & 10 & 1 \\ 2 & 593 & 3 \\ 0 & 13 & 270 \end{bmatrix} \text{matrica konfuzije}$$

| klasa | Preciznost | odziv | F1 - rezultat |
|---|---|---|---|
| Pritisak lijevom rukom | 99% | 96% | 98% |
| Nema pritiska | 96% | 99% | 98% |

| | | | |
|---|---|---|---|
| Pritisak desnom rukom | 99% | 95% | 97% |

3. Test Loss 11.69%
   Test Accuracy 96.95%

$$\begin{bmatrix} 248 & 19 & 1 \\ 2 & 594 & 2 \\ 0 & 20 & 263 \end{bmatrix} \text{matrica konfuzije}$$

| klasa | Preciznost | odziv | F1 - rezultat |
|---|---|---|---|
| Pritisak lijevom rukom | 99% | 93% | 96% |
| Nema pritiska | 94% | 99% | 97% |
| Pritisak desnom rukom | 99% | 93% | 96% |

4. Test Loss 10.41%
   Test Accuracy 98.35%

$$\begin{bmatrix} 258 & 10 & 0 \\ 3 & 593 & 2 \\ 0 & 9 & 274 \end{bmatrix} \text{matrica konfuzije}$$

| klasa | Preciznost | odziv | F1 - rezultat |
|---|---|---|---|
| Pritisak lijevom rukom | 99% | 96% | 98% |
| Nema pritiska | 97% | 99% | 98% |
| Pritisak desnom rukom | 99% | 97% | 98% |

5. Test Loss 10.55%
   Test Accuracy 98.17%

$$\begin{bmatrix} 260 & 8 & 0 \\ 3 & 594 & 1 \\ 0 & 13 & 270 \end{bmatrix} \text{matrica konfuzije}$$

| klasa | Preciznost | odziv | F1 - rezultat |
|---|---|---|---|
| Pritisak lijevom rukom | 99% | 97% | 98% |
| Nema pritiska | 97% | 99% | 98% |
| Pritisak desnom rukom | 99% | 95% | 97% |

6. Test Loss 9.41%
   Test Accuracy 97.56%

$$\begin{bmatrix} 256 & 12 & 0 \\ 1 & 596 & 1 \\ 0 & 16 & 267 \end{bmatrix} \text{matrica konfuzije}$$

| klasa | Preciznost | odziv | F1 - rezultat |
|---|---|---|---|
| Pritisak lijevom rukom | 99% | 96% | 98% |
| Nema pritiska | 96% | 99% | 98% |
| Pritisak desnom rukom | 99% | 94% | 97% |

7. Test Loss 8.64%
   Test Accuracy 97.65%

$$\begin{bmatrix} 256 & 10 & 2 \\ 1 & 595 & 2 \\ 0 & 16 & 267 \end{bmatrix} \text{matrica konfuzije}$$

| klasa | Preciznost | odziv | F1 - rezultat |
|---|---|---|---|
| Pritisak lijevom rukom | 99% | 96% | 98% |
| Nema pritiska | 96% | 99% | 98% |
| Pritisak desnom rukom | 99% | 94% | 96% |

8. Test Loss 13.25%
   Test Accuracy 97.91%

$$\begin{bmatrix} 256 & 12 & 0 \\ 3 & 592 & 3 \\ 0 & 13 & 270 \end{bmatrix} \text{matrica konfuzije}$$

| klasa | Preciznost | odziv | F1 - rezultat |
|---|---|---|---|
| Pritisak lijevom rukom | 99% | 96% | 97% |
| Nema pritiska | 96% | 99% | 97% |
| Pritisak desnom rukom | 99% | 95% | 97% |

9. Test Loss 12.82%
   Test Accuracy 97.74%

$$\begin{bmatrix} 252 & 16 & 0 \\ 3 & 591 & 4 \\ 0 & 11 & 272 \end{bmatrix} \text{matrica konfuzije}$$

| klasa | Preciznost | odziv | F1 - rezultat |
|---|---|---|---|
| Pritisak lijevom rukom | 99% | 94% | 96% |
| Nema pritiska | 96% | 99% | 97% |
| Pritisak desnom rukom | 99% | 96% | 97% |

10. Test Loss 8.84%
    Test Accuracy 98.26%

$$\begin{bmatrix} 259 & 9 & 0 \\ 3 & 592 & 3 \\ 0 & 9 & 274 \end{bmatrix} \text{matrica konfuzije}$$

| klasa | Preciznost | odziv | F1 - rezultat |
|---|---|---|---|
| Pritisak lijevom rukom | 99% | 97% | 98% |
| Nema pritiska | 97% | 99% | 98% |
| Pritisak desnom rukom | 99% | 97% | 98% |

11. Test Loss 8.46%
    Test Accuracy 98.26%

$$\begin{bmatrix} 259 & 8 & 1 \\ 3 & 593 & 2 \\ 0 & 10 & 273 \end{bmatrix} \text{matrica konfuzije}$$

| klasa | Preciznost | odziv | F1 - rezultat |
|---|---|---|---|
| Pritisak lijevom rukom | 99% | 97% | 98% |
| Nema pritiska | 97% | 99% | 98% |
| Pritisak desnom rukom | 99% | 96% | 98% |

12. Test Loss 10.76%
    Test Accuracy 97.65%

$$\begin{bmatrix} 257 & 10 & 1 \\ 2 & 594 & 2 \\ 0 & 11 & 272 \end{bmatrix} \text{matrica konfuzije}$$

| Klasa | Preciznost | odziv | F1 - rezultat |
|---|---|---|---|
| Pritisak lijevom rukom | 99% | 96% | 98% |
| Nema pritiska | 97% | 99% | 98% |
| Pritisak desnom rukom | 99% | 96% | 97% |

13. Test Loss 8.68%
    Test Accuracy 98.00%

$$\begin{bmatrix} 258 & 9 & 1 \\ 3 & 593 & 2 \\ 1 & 8 & 274 \end{bmatrix} \text{matrica konfuzije}$$

| Klasa | Preciznost | odziv | F1 - rezultat |
|---|---|---|---|
| Pritisak lijevom rukom | 98% | 96% | 97% |
| Nema pritiska | 97% | 99% | 98% |
| Pritisak desnom rukom | 99% | 97% | 98% |

14. Test Loss 13.24%
    Test Accuracy 96.95%

$$\begin{bmatrix} 258 & 10 & 0 \\ 7 & 583 & 8 \\ 0 & 8 & 275 \end{bmatrix} \text{matrica konfuzije}$$

| klasa | Preciznost | odziv | F1 - rezultat |
|---|---|---|---|
| Pritisak lijevom rukom | 97% | 96% | 97% |
| Nema pritiska | 97% | 97% | 97% |
| Pritisak desnom rukom | 97% | 97% | 97% |

15. Test Loss 8.82%
    Test Accuracy 98.09%

$$\begin{bmatrix} 256 & 12 & 0 \\ 3 & 593 & 2 \\ 0 & 13 & 270 \end{bmatrix} \text{matrica konfuzije}$$

| klasa | Preciznost | odziv | F1 - rezultat |
|---|---|---|---|
| Pritisak lijevom rukom | 99% | 96% | 97% |
| Nema pritiska | 96% | 99% | 98% |
| Pritisak desnom rukom | 99% | 95% | 97% |

16. Test Loss 10.79%
    Test Accuracy 97.21%

$$\begin{bmatrix} 255 & 13 & 0 \\ 5 & 588 & 5 \\ 1 & 7 & 275 \end{bmatrix} \text{matrica konfuzije}$$

| klasa | Preciznost | odziv | F1 - rezultat |
|---|---|---|---|
| Pritisak lijevom rukom | 98% | 95% | 96% |
| Nema pritiska | 97% | 98% | 98% |
| Pritisak desnom rukom | 98% | 97% | 98% |

17. Test Loss 10.00%
    Test Accuracy 96.78%

$$\begin{bmatrix} 246 & 22 & 0 \\ 1 & 592 & 5 \\ 0 & 24 & 259 \end{bmatrix} \text{matrica konfuzije}$$

| klasa | Preciznost | odziv | F1 - rezultat |
|---|---|---|---|
| Pritisak lijevom rukom | 99% | 92% | 96% |
| Nema pritiska | 93% | 99% | 96% |
| Pritisak desnom rukom | 98% | 92% | 95% |