VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

KLASIFIKACE OBJEKTŮ ZPRACOVÁNÍM OBRAZU NA ZÁKLADĚ ZMĚNY TOPOLOGIE
OBJECT CLASIFICATION BASED ON ITS TOPOLOGY CHANGE USING IMAGE PROCESSING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Tomáš Zbavitel**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Jiří Krejsa, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Bc. Tomáš Zbavitel**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Inženýrská mechanika a biomechanika
Vedoucí práce:	**doc. Ing. Jiří Krejsa, Ph.D.**
Akademický rok:	2021/22

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Klasifikace objektů zpracováním obrazu na základě změny topologie

Stručná charakteristika problematiky úkolu:

Díky pokroku v metodách strojového učení je v současné době možné využívat obrazu získaného z běžné kamery pro klasifikaci objektů. Předmětem diplomové práce je vytvoření klasifikátoru, který bude schopný rozpoznat jednotlivé stavy objektu na základě změny jeho topologie. Konkrétně se jedná o úlohu detekce písmen jednoruční abecedy českého znakového jazyka. Klasifikátor bude založen na metodách strojového učení, konkrétně konvolučních neuronových sítích. Předpokládá se využití předpřipravených softwarových nástrojů (TensorFlow).

Cíle diplomové práce:

1. Prostudujte problematiku klasifikace objektů zpracováním obrazu metodami strojového učení.
2. Seznamte se se softwarovým balíkem TensorFlow a programovacím jazykem Python.
3. Na úloze klasifikace znaků jednoruční abecedy českého znakového jazyka proveďte sběr dat s dostatečnou mohutností výsledné datové množiny.
4. Proveďte preprocessing dat.
5. V prostředí TensorFlow navrhněte topologii konvoluční neuronové sítě pro danou úlohu, síť natrénujte a otestujte.
6. Vyhodnoťte úspěšnost metody.

Seznam doporučené literatury:

PATHAK A. et.al.: Application of Deep Learning for Object Detection, Procedia Computer Science, Vol. 132, 2018, pp. 1706-1717, DOI:10.1016/j.procs.2018.05.144.

ZHONG--QIU Z. et. al.: Object Detection with Deep Learning: A Review, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, 2019.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku.

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

Cílem předkládané práce je vybrat vhodnou metodu klasifikace objektů pro rozpoznání znaků jednoruční prstové abecedy. Za tímto účelem byla vytvořena dostatečně mohutná množina dat, která je součástí této práce. Vytvoření množiny dat je nutné pro trénink konvoluční neuronové sítě. Dále byla nalezena vhodná topologie pro klasifikaci dat. Celá práce je implementována pomocí programu Python a byla použita open-source knihovna Keras.

Klíčová slova: klasifikace objektů, rozpoznávání znaků jednoruční prstové abecedy, konvoluční neuronová síť, Python, Keras

Abstract

The aim of the present work is to select a suitable object classification method for the recognition of one-handed finger alphabet characters. For this purpose, a sufficiently robust dataset has been created and is included in this work. The creation of the dataset is necessary for training the convolutional neural network. Furthermore, a suitable topology for data classification was found. The whole work is implemented using Python and the open-source library Keras was used.

Keywords: object classification, one-handed finger alphabet character recognition, convolutional neural network, Python, Keras

BIBLOGRAFICKÁ CITACE

ZBAVITEL, Tomáš. *Klasifikace objektů zpracováním obrazu na základě změny topologie*. Brno, 2022. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/137270. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky. Vedoucí práce Jiří Krejsa.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně na základě svých znalostí, odborných konzultací, literatury uvedené v seznamu literatury a pod vedením vedoucího diplomové práce pana doc. Ing. Jiřího Krejsy, Ph.D.

PODĚKOVÁNÍ

Chtěl bych moc poděkovat vedoucímu diplomové práce doc. Ing. Jiřímu Krejsovi, Ph.D., za jeho čas, velkou trpělivost, podporu a cenné připomínky při zpracování této práce.

Rád bych srdečně poděkoval paní BcA. Mgr. Radce Kulichové, DiS., a paní Mgr. Jitce Hořanské za tlumočení a skvělou podporu během studia. Dále pak i paní MUDr. Zdeňce Zigalové a celému Poradenskému centru Alfons.

A velký dík patří mé rodině a nejbližším, kteří mě podporovali a věřili mi ve všech okamžicích mého studia.

Obsah

1. Úvod

Klasifikační úloha je úloha rozdělení jednotlivých vzorků do tříd. Tyto třídy mohou být předem známé nebo může být znám pouze jejich počet. Klasifikační úloha je často prvním a základním krokem v širším spektru úloh, jako je například vyhledávání objektů, jejich sledování a další. Klasifikace objektů je vždy založena na popisu vlastností daného objektu. Jedním z možných popisů je obraz objektu, který má řadu výhod: získání obrazu je pasivní metodou nevyžadující aktivní interakci s objektem, díky technickému vývoji kamer je pořízení obrazu levné a získaný obraz může být velmi kvalitní. Strojovému zpracování obrazů se věnuje oblast zvaná počítačové vidění.

Z obecného hlediska lze říci, že počítačové vidění je určitá snaha o nápodobu lidského vidění. Uskutečňuje se díky snímání obrazu elektronickými prostředky a porozumění jejich obsahu prostřednictvím počítačového zpracování. Významnou rolí počítačového vidění je rozpoznání objektu, kdy je konečný obraz výsledkem perspektivního zobrazení části trojrozměrného prostředí do dvourozměrného prostředí.

Počítačové vidění je disciplína, která se snaží technickými prostředky aspoň částečně napodobit lidské vidění. Pro počítačové vidění je typická snaha porozumět obecné trojrozměrné scéně, např. takové, jakou zahlédnete při pohledu z okna do zahrady. Postupy počítačového vidění jsou značně složité, s těžištěm v interpretaci obrazových dat, která jsou nejčastěji reprezentována symbolicky. Jádrem pokročilejších postupů jsou znalostní systémy a techniky umělé inteligence. Počítačové vidění je jak vědou, tak i technologií usilující o vytváření „strojů schopných vidět a vnímat“.

Z historického hlediska zaznamenáváme zájem o počítačové vidění v 60. letech minulého století. V roce 1966 zadává Marvin Minsky (výzkumník na poli umělé inteligence) úlohu počítačového vidění jednomu ze svých studentů jako prázdninový projekt. V roce 1970 je zaznamenán jistý pokrok v interpretaci obrazu v omezeném světě.

V 80. letech minulého století se objevují umělé neuronové sítě, později se díky inspiraci biologického vidění (David Marr a spolupracovníci) zájem posunuje více ke geometrii a k použití matematiky.

V letech 1990 až 2000 je zaznamenána detekce a rozpoznávání lidských obličejů a roste popularita statistické analýzy i zájem o geometrické úlohy vidění. Rozpoznávání se využívá ve větším měřítku a začínají být k dispozici rozsáhlé anotované databáze. Jedná se o počátek prakticky použitelných metod analýzy videa.

První software, který byl schopný rozpoznávat objekty, se jmenoval DIDAK. Byl vytvořen v roce 1980 Joachimem Wieselem, německým vedoucím pracovníkem Ústavu fotogrammetrie.

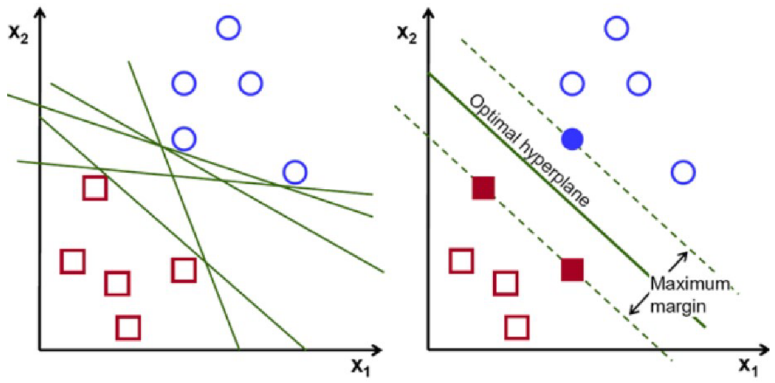
Vstupem byly uměle vytvořené matice a malé obrázky, které byly za pomoci algoritmu rozpoznány. V roce 1977 se uskutečnila konference o počítačovém vidění a rozpoznávání (CVPR), v současnosti patří, společně s mezinárodní konferencí o počítačovém vidění (ICCV), ke konferencím největšího významu v rámci počítačového vidění.

2. Možné metody klasifikace objektů

Pro klasifikaci objektů na digitálním záznamu je v praxi využíváno více různých přístupů. Každý z těchto přístupů má určité vlastnosti, které se mohou lépe nebo hůře hodit pro nějaký konkrétní případ použití. Následující kapitola proto bude věnována přehledu nejvíce používaných metod ke klasifikaci objektů a výběru té nejvhodnější metody.

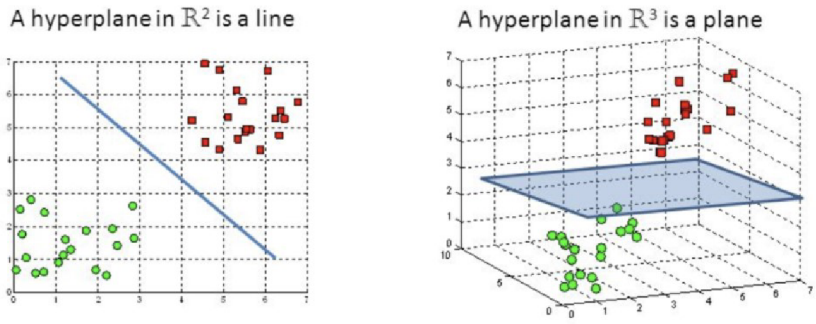
2.1. Support vector machines (SVM)

Klasifikační metoda SVM byla navržena Vladimirem Vapnikem a snaží se nalézt optimální rozhodovací nadrovinu pro správnou klasifikaci mezi datovými body různých tříd (obr. 1). Metoda SVM je původně navržena pro klasifikaci se dvěma třídami, mezi kterými se hlasuje v algoritmu (jeden proti jednomu). Pro více než dvě třídy algoritmus klasifikace SVM je spuštěn pro každou možnou dvojici tříd. Pro každou dvojici obdrží vybraná třída bod a třída, která nasbírá nejvíce bodů ze všech SVM dvou tříd, je vítězem (jeden proti všem). Doba trénování pro jeden proti jednomu je kratší než pro jeden proti všem. Dimenzionalita nadroviny se rovná počtu vstupních prvků mínus jedna, například při práci se třemi prvky bude nadrovinou dvojrozměrná rovina. [1][2]



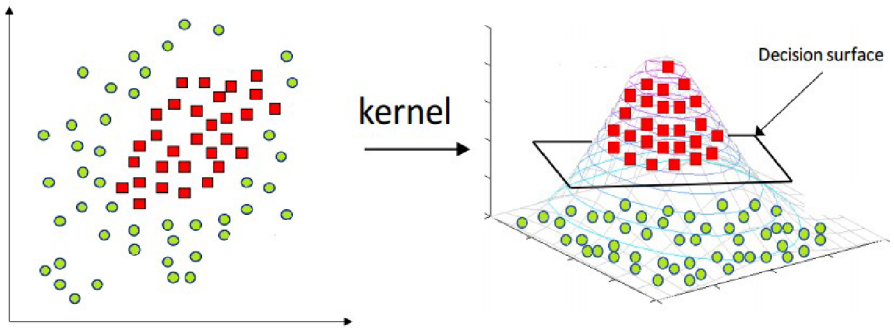
Obr. 1: Možné nadroviny (vlevo) a optimální nadrovina (vpravo) [2]

Datové body na jedné straně nadroviny budou klasifikovány do určité třídy, stejně tak budou datové body na druhé straně nadroviny klasifikovány do jiné třídy. Toto je znázorněno na obrázku č. 2, kde nadrovina leží mezi zelenými a červenými body, které budou klasifikovány do své třídy. Vzdálenost mezi nadrovinou a prvním bodem (pro všechny různé třídy) na obou stranách nadroviny je měřítkem jistoty, že algoritmus je o klasifikačním rozhodnutí. Čím větší jsou vzdálenosti, tím větší je jistota, že SVM činí správné rozhodnutí. Datové body, které jsou nejblíže k nadrovině, se nazývají podpůrné vektory a určují orientaci a polohu nadroviny, aby se maximalizoval okraj klasifikátoru. Za okraj je považován prostor dělící se na dvě strany, přičemž v každé se nachází jiná klasifikovaná třída, a jehož středem prochází rozhodovací linie. Počet podpůrných vektorů lze libovolně zvolit v závislosti na aplikacích. [2]



Obr. 2: Nadroviny [2]

Za předpokladu, že data nejsou lineárně oddělitelná, vede klasifikace SVM ke špatným lineárním výsledkům. Tento problém lze obejít pomocí techniky známé jako Kernel Trick. Tato technika je schopna namapovat nelineární oddělitelná data do vyššího dimenzionálního prostoru, což činí tato data lineárně oddělitelnými. Pomocí tohoto nového dimenzionálního prostoru lze potom snadno implementovat SVM (obr. 3). Existuje mnoho různých typů funkcí pro Kernel Trick, které lze použít k vytvoření tohoto vyššího dimenzionálního prostoru, a to lineární, polynomiální, sigmoidní a radiální základní funkce (RBF).[2][3]



Obr. 3: Kernel Trick [3]

Výhody

- nemá předpoklady pro normální rozdělení dat.
- lze využít pro lineární i pro nelineární klasifikaci.

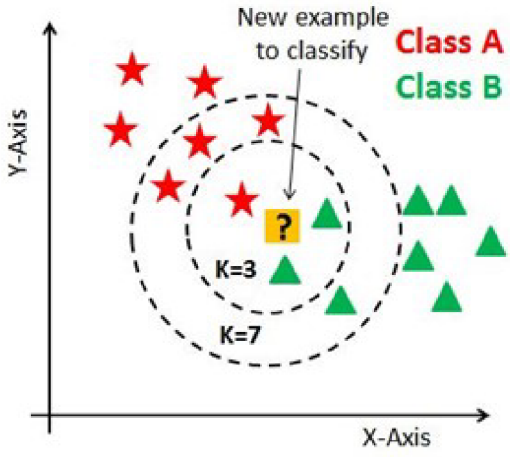
Nevýhody

- vyžaduje stanovení parametrů (např. C, kde vyjadřuje poměr vlivu obou členů kriteriální funkce) a případně i typu jádra.
- velmi často je obtížné učení, protože prakticky vždy existuje riziko uvíznutí v lokálním minimu chybové funkce a navíc je učení silně komplikováno hledáním vysokého počtu vah ve vícerozměrovém prostoru.
- algoritmus SVM není vhodný pro velké soubory dat.[2][3]

2.2. Paměťové klasifikátory (IBL)

Paměťové klasifikátory (Instance Based Learning, zkratka IBL) jsou učení založená na instancích, to znamená, že tyto metody jsou schopné nalezení podobnosti neznámého vzoru se známým vzorem. Učení v těchto algoritmech spočívá v prostém ukládání předložených trénovacích dat, jež jsou složena z atributů, a informace, do které klasifikační třídy patří. Při výskytu nové instance dotazu se z paměti načte množina podobných příbuzných instancí a použije se ke klasifikaci nové instance dotazu. Důležitým parametrem je výběr metriky vzdálenosti mezi sousedy, kde u metody IBL se využívá metrika Euklidovské vzdálenosti. K učení tato metoda používá především techniku zapamatování. Metody IBL se obvykle nazývají lazy (líné) metody učení, které v procesu učení nevytváří žádný model, a bývají často označovány jako k-NN (k nearestneighbor).

Algoritmy IBL používají jako model celý soubor dat. Algoritmus k-NN zkoumá blízké okolí vstupní instance v prostoru příznakových vektorů a vypisuje označení, které v tomto blízkém okolí viděl nejčastěji. [4][5]



Obr. 4: Líné učení, k-NN [4]

Výhody:

- tyto metody mohou také používat složitější symbolické reprezentace instancí.
- místo odhadu pro celou sadu instancí lze provést cílové funkce místní aproximace.
- tento algoritmus se snadno přizpůsobí novým datům, která jsou shromažďována za pochodu.

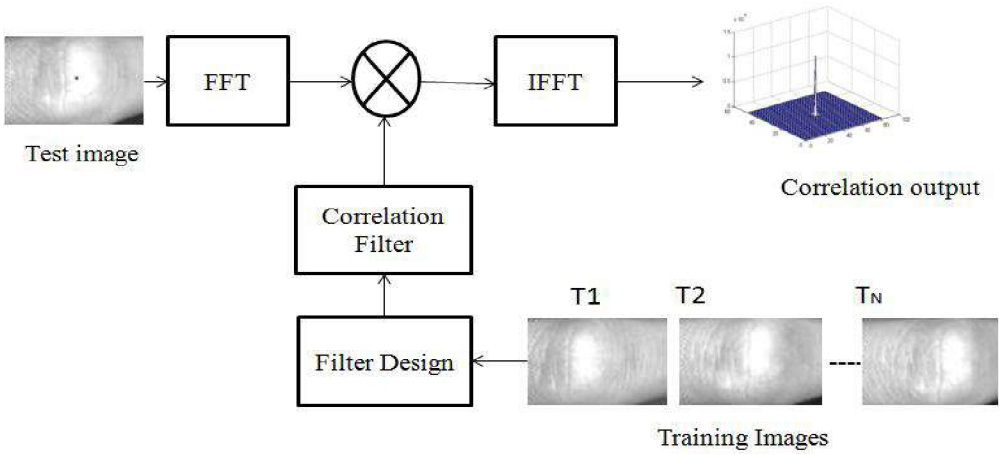
Nevýhody:

- náklady na klasifikaci nových případů mohou být vysoké.
- k uložení dat je třeba mít k dispozici velké množství paměti a každý dotaz zahrnuje spuštění identifikace místního modelu od začátku.
- velká citlivost na šum v trénovacích datech při nízkém *k* (s rostoucím *k* se zvyšuje náročnost na výpočet). [5][6]

2.3. Minimum AverageCorrelationEnergy (MACE)

Jedná se o slibný přístup k automatickému rozpoznávání cílů, který kombinuje výhody filtru minimální průměrné korelační energie (Minimum Average Correlation Energy) ve frekvenční oblasti s vynikajícími klasifikačními schopnostmi vícevrstvých sítí.

Vstupní obraz je nejprve převeden do frekvenční oblasti pomocí diskrétních Fourierových transformací a přetransformován na sloupce vektoru. Nadále je korelován korelačním filtrem a pasuje-li, pak vytváří vrcholy na těchto místech. Vrcholy objevující se na jiném místě než cílovém se nazývají falešné vrcholy. Cílem návrhu algoritmu detekce je nalézt vrcholy, kterých se vyskytuje v cílových místech co nejvíc, a zároveň minimalizovat počet falešných poplachů. K tomu, aby došlo k minimalizaci počtu falešných poplachů, byl vytvořen filtr MACE. Filtr MACE minimalizuje průměrnou korelační energii snímků získaných při korelaci snímků v trénovací množině pomocí filtru a vytváří také ostrý korelační vrchol v poloze trénovaného objektu. K tomu, aby se vrcholy vyskytovaly co nejvíce v cílových místech, obsahuje minimalizační problém omezení, a to takový, že výstup v cílovém středu je nějaká nenulová hodnota. [7]



Obr. 5: MACE filtr [8]

Výhody

- je výpočetně efektivní, tedy méně náročný
- dokáže zpracovat i vstupy se značným množstvím šumu.
- ve frekvenční oblasti je korelační operace rychlá.

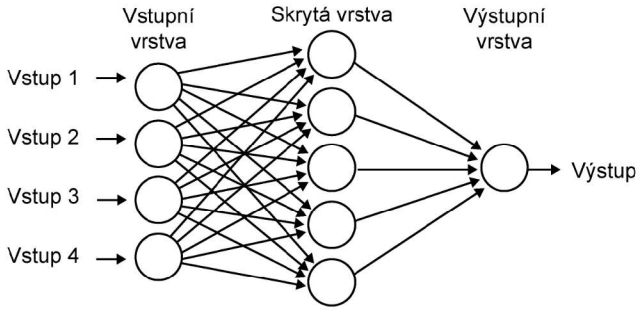
Nevýhody

- je stále závislý na množství trénovacích dat. [7][9]

2.4. Neuronové sítě

Neuronová síť je propojená skupina matematických modelů neuronu, která používá pro zpracování informací konekcionistický přístup* k výpočtu. Tato síť byla navržena na základě fungování lidského mozku. Lidský mozek pracuje na principu posílání impulsů mezi mnoha spojenými buňkami, které se nazývají neurony. Z toho důvodu bylo cílem vytvořit algoritmus, jenž pracuje na podobném principu a má schopnost učit se, přizpůsobovat se vstupním datům a měnit vnitřní parametry, které ovlivňují úspěšnost predikce. Všechna spojení neuronu jsou upravena váhou a sečtena (lineární kombinace). Nakonec aktivační funkce řídí amplitudu výstupu obvykle v rozsahu mezi 0 a 1, někdy -1 až 1. Každá síť může být upravena jinak. Tato úprava záleží na typu neuronu, topologickém uspořádání a strategii učení.

Na obrázku jsou na vstupech zobrazeny umělé neurony, které nejsou mezi sebou propojené v jedné vrstvě, za to jsou plně propojené mezi vrstvami. Každý spoj má svoji váhu, která představuje intenzitu procházejícího signálu. Vrstvy máme jednu vstupní a jednu výstupní, mezi nimi najdeme jednu nebo více skrytých vrstev. [10][11]



Obr. 6: Umělé neuronové sítě [11]

Výhody

- má schopnost učit se bez nutnosti explicitní znalosti algoritmu řešení.
- má odolnost proti šumu, menší změny na vstupu neovlivňují výstup.
- lze využít u nelineárních vztahů, řešení komplexních a velmi složitých úloh.
- je možnost implementovat neuronové sítě hardwarově, lze navrhnout integrované obvody jako stavební prvky neuronových sítí.

Nevýhody

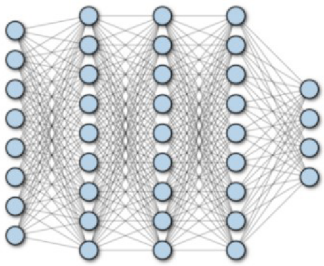
- velká náročnost na volbu parametrů sítě (např. počty vrstev, aktivační funkce apod.).
- sklony k přeučení.
- vysoká výpočetní náročnost při trénování rozsáhlých sítí. [10][11]

***Konekcionistický přístup** je přístup, který modeluje mentální reprezentaci v síti uzlů vzájemně propojených různě silnými spoji, jejichž síla se může měnit při učení.

2.4.1. Typy neuronových síti

Plně propojená neuronová síť

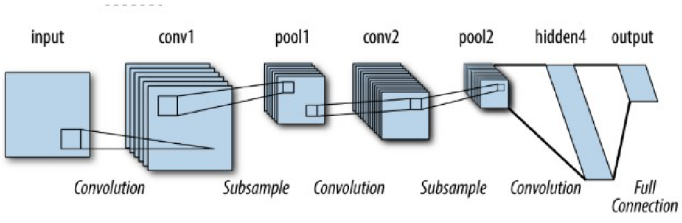
Plně propojená neuronová síť je tvořena tak, že všechny neurony v jedné vrstvě jsou propojeny s dalšími neurony v další vrstvě. Díky tomu, že je strukturálně agnostická, tj. není třeba přijímat zvláštní předpoklady pro vstupy, je velmi široce použitelná. V praxi ale má v rozpoznávání a klasifikaci obrazu určité problémy, je výpočetně náročná a obtížně srozumitelná pro člověka, protože je „hluboká“, to znamená, že má mnoho vrstev uzlů nebo neuronů. [12]



Obr. 7: Plně propojená neuronová síť [12]

Konvoluční neuronová síť

Konvoluční neuronová síť je vícevrstvá síť a nejčastěji se používá v oboru zpracování obrazu. V minulosti se používala pro rozpoznávání rukopisu jen omezeně, později dosáhla velkých úspěchů při řešení problémů počítačového vidění, jako je superrozlišení, sledování, klasifikace, segmentace obrazu a rozpoznávání objektů. Konvoluční neuronové sítě (CNN) se obvykle skládají z několika různých vrstev jako konvolučních nebo pooling vrstev. K práci s CNN lze s výhodou využívat GPU výpočty. [13]



Obr. 8: Schéma konvoluční neuronové síť [12]

Rekurentní neuronová síť

Rekurentní neuronovou síť lze využít ke klasifikaci nebo regresi stejně jako konvoluční neuronovou síť. Pracuje s 1D signály a časovými řadami a je schopná naučit se dlouhodobé závislosti v signálech (závislosti ze vzdálených časových kroků). Využívá se v technických oborech, např. v oblastech medicíny, biologie nebo finančnictví. [14]

3. Formulace problému

Klasifikace objektů může být založena na celé řadě vlastností těchto objektů, jako jsou například barva, textura, tvar obrysu atd. Většina technických objektů je pro klasifikaci pomocí metod počítačového vidění vhodná díky uniformitě vlastností v dané třídě objektů (např. strojní součásti). Existuje ovšem řada objektů, které mohou spadat do stejné třídy, ale klíčovou vlastností pro klasifikaci je topologie jejich tvaru.

Zástupcem takových objektů jsou například jednoruční znaky prstové abecedy, které mají z hlediska řešené úlohy řadu vlastností, které činí klasifikaci obtížnou. V první řadě je to skutečnost, že každý člověk má různé tělesné proporce, které se projeví i na tvaru ruky, například ruka a prsty jsou silné, úzké, krátké apod. Každý člověk také při znakování formuje ruku do mírně odlišného tvaru. V neposlední řadě je ruka v obraze na různých místech, různě natočená, a to obecně ve všech třech rozměrech (i když rozdíly v úhlech natočení nejsou velké).

Jednou z motivací výběru této množiny tříd ke klasifikaci je i to, že jsem od narození neslyšící a zajímalo mě, jaké rozdíly se projeví mezi jednotlivými objekty využívajícími znaky prstové abecedy. Měl jsem také možnost získat dostatečně velkou množinu dat pro trénování a testování. Mohutnost trénovací množiny je pro úspěšné řešení takto formulované úlohy klíčová.

Velký vliv mohou sehrát i podmínky pro samotný obrazový záznam. Tedy to, v jakém prostředí se bude pořizovat a jakou roli bude hrát dostatek osvětlení. Natáčení lze uskutečnit v rozličném prostředí, například v kanceláři, klubu, restauraci, doma apod., kde jsou odlišné barvy pozadí a různé světelné podmínky (stíny, šero, ostré světlo, sluneční osvětlení). Určitý vliv mohou mít i různé věci na pozadí. Tyto překážky klasifikaci objektů dále znesnadňují. Pokud tedy bude možné úspěšně klasifikovat znaky jednoruční znakové abecedy, může být vytvořený klasifikátor obecně použitelný pro další úlohy bližší technické praxi. Aplikací je nepřeberné množství, od klasifikace objektů při třídění odpadu až po klasifikaci rostlin při robotickém ošetřování plodin.

Z provedené rešerše a získaných teoretických podkladů vyplývá, že pro klasifikaci jakéhokoliv objektu je jednou ze slibných metod využití natrénované umělé neuronové sítě. Řešenou úlohou je tedy klasifikace znaků jednoruční prstové abecedy zpracováním obrazu. S uvažováním výše uvedených skutečností pak lze naformulovat hlavní dílčí cíle této práce:

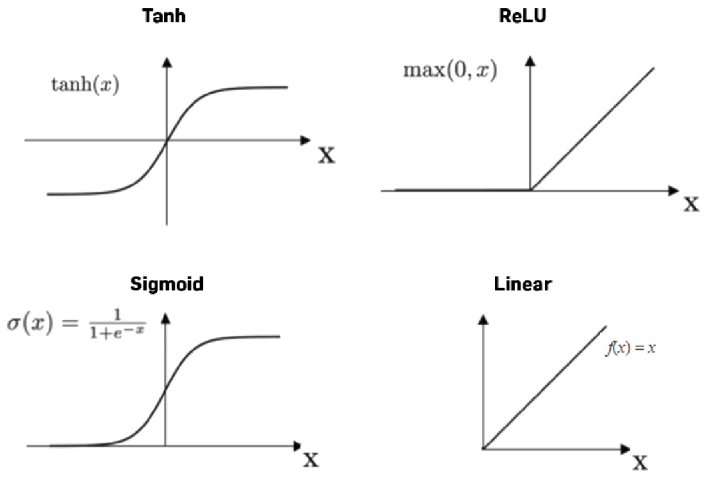
- získat dostatečně mohutnou datovou sadu vzorů.
- najít vhodnou topologii konvoluční neuronové sítě.
- tuto topologii implementovat pomocí vhodného SW nástroje.
- síť natrénovat a vyhodnotit její chování na testovacích datech.

4. Aplikace konvoluční neuronové sítě

V hlubokém učení je konvoluční neuronová síť (CNN nebo ConvNet) třída umělých neuronových sítí, které se nejčastěji používají k analýze vizuálních obrazů jako rozpoznávání obrazu a videa, segmentace obrazu, klasifikace obrazu, analýza lékařských obrazů, zpracování přirozeného jazyka, finančních časových řad a doporučovacích systémů. Konvoluční sítě byly inspirovány tím, že vzor spojení mezi neurony připomíná organizaci zrakové kůry živočichů. Jednotlivé korové neurony reagují na podněty pouze v omezené oblasti zorného pole známé jako receptivní pole. CNN jsou verze vícevrstvých perceptronů, kterými se obvykle rozumí plně propojené sítě, což znamená, že každý neuron v jedné vrstvě je propojen se všemi neurony v další vrstvě. Díky „úplnému propojení“ jsou tyto sítě náchylné k nadměrnému přizpůsobování. Mezi typické způsoby regularizace se zahrnuje penalizace parametrů během trénování (přidání nějaké formy váhy ke ztrátové funkci). V porovnání s jinými algoritmy klasifikace obrazu používají CNN relativně málo předzpracování. Jinými slovy se síť učí optimalizovat filtry (nebo jádra) prostřednictvím automatického učení, zatímco v tradičních algoritmech jsou tyto filtry vytvářeny ručně. Tato nezávislost na předchozích znalostech a lidské snaze při navrhování funkcí je velkou výhodou. [14][15]

Aktivační funkce

Aktivační funkce se používá po konvoluční vrstvě a snižuje linearitu výstupních informací. Jedná se tedy o nelineární funkci, která převádí sumu vážených vstupů na výstup. Dříve se používala sigmoida nebo hyperbolický tangens, v současnosti se používá ReLU (rectified linear unit), která byla zavedená profesorem Richardem Hahnloserem a je nejpoužívanější aktivační funkcí. Oproti předchozím dvěma typům je rychlejší, ale její problém spočívá v tom, že všechny záporné hodnoty se okamžitě stávají nulovými. To znamená, že jakýkoli záporný vstup zadaný aktivační funkci ReLU okamžitě změní v grafu hodnotu na nulu, což snižuje schopnost modelu správně se přizpůsobit nebo trénovat z dat. [16][17]



Obr. 9:Běžné typy aktivační funkce [18]

Ztrátová funkce (Chybová funkce)

Během trénování ztrátová funkce měří, jak se výstup sítě liší od požadovaného výstupu. Kategorizací aplikování ztrátové funkce (loss function) je vypočítána ztráta, kterou každý neuron provedl při dopředném výpočtu. Následně je ztráta propagována v opačném směru a každý neuron si dle ztráty upraví jednotlivé váhy. Neuronová síť se pomocí úprav vah snaží minimalizovat ztrátovou funkci. [16][17]

4.1. Vrstvy sítí

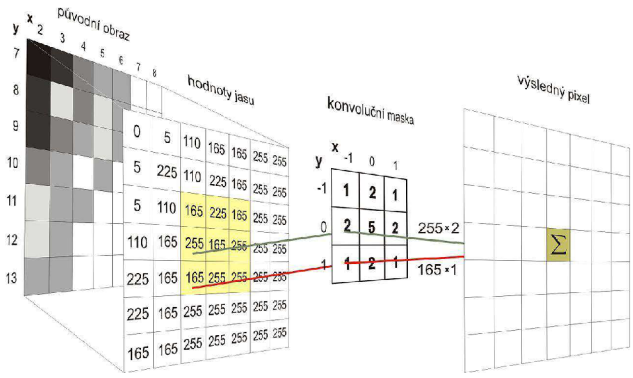
Vrstvy neuronových sítí mohou být různě uspořádané a nastavené, tj. mohou to být vrstvy různých typů v závislosti na druhu neuronů, které je tvoří. Udávají z velké části vlastnosti a schopnosti sítě, například pořadí různých typů vrstev nebo vstupní rozměry.

Plně propojená vrstva

Jedná se o uspořádání neuronů, kde každý neuron má na svém vstupu výstup každého neuronu předchozí vrstvy. Ve stejné vrstvě tak nejsou neurony navzájem napojené mezi sebou, tudíž se signál šíří pouze jedním směrem. Jejich aktivace lze vypočítat pomocí maticového násobení. Hodnoty v použité matici jsou parametry, které lze trénovat a aktualizovat pomocí zpětného šíření (anglicky backpropagation). Výstupem je vektor o určitém rozměru a používá se ke změně rozměrů vektoru. Lze také aplikovat vrstvu na vektor operace jako rotace, škálování a translace. [29]

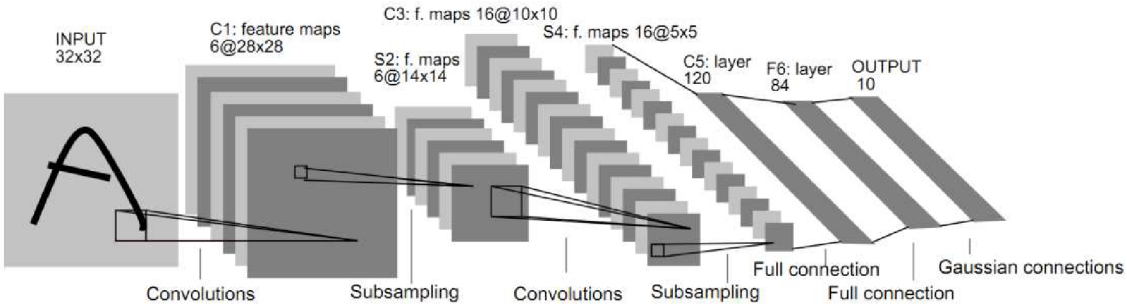
Konvoluční vrstva

Jedná se o dopřednou neuronovou vrstvu, která vykonává operaci konvoluce. V každé operaci je konvoluční maska, jedná se o matice o různých rozměrech podle potřeby. Například lze zvolit 5×5×3, což definuje výšku a šířku masky a taky její hloubku, ve které budou definovány barevné vrstvy RGB, kde každý pixel má svou barvu, která je popsána číslem. Následně maskou prochází celý obraz a násobí se jednotlivé pixely s hodnotami masky. Výsledky se sečtou a získá se tak jedno číslo. Celý postup je znázorněn na obrázku. Nadále je také možné zvolit velikost kroku, po kterém se maska bude posouvat. Tím dojde k redukci rozměrů původní matice. [15]



Obr. 10: Konvoluční vrstva [20]

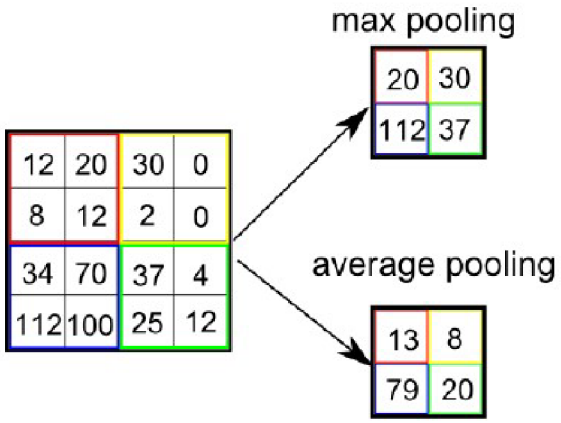
V praxi většinou datový vstup prochází více filtry najednou, což pak ovlivňuje rozměr výstupního tvaru. Na obrázku č. 11 je znázorněno, že roste počet aktivačních map v jednotlivých vrstvách a zároveň se mapy zmenšují, a to díky velikosti masky a použití sdružovací vrstvy (pooling), která bude popsána níže. [15]



Obr. 11: Schéma vrstev [21]

Sdružovací vrstva (Pooling vrstva)

Sdružovací vrstvy se používají v konvolučních sítích, kde redukují množství parametrů předávaných do sítě, výpočetní náročnost a celý proces může pomoci vyhnout se nežádoucímu přeučení sítě. Při výpočtu v konvoluční vrstvě se několikrát opakují konvoluce, čímž vznikají nadbytečná data. Sdružovací vrstva odstraňuje právě tato nadbytečná data. Existují dva typy této funkce. První vybírá maxima z dané masky (max pooling) a druhá vypočítá průměr (average pooling), viz obrázek č. 12. [15]



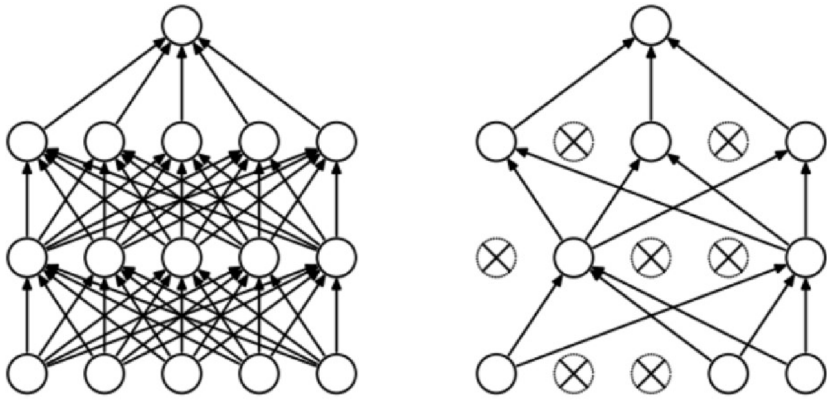
Obr. 12: Funkce sdružovací vrstvy [22]

Flatten vrstva

Vrstva flatten slouží ke zploštění vstupu, což upraví vícerozměrné data do jednorozměrného vektoru. Například Flatten([[1;2];[3;4]];[[5;6];[7;8]]) = [1;2;3;4;5;6;7;8]. [23]

Dropout vrstva

Dropout je operace, která se používá k odstranění problému přílišného přizpůsobení, tzn. přeučení. Při každé iteraci ignoruje daná nežádoucí data u vstupních dat, která se obvykle nazývají šum. Tato operace se snaží odstranit šumová data, a tím zabrání nadměrnému přizpůsobení modelu a přeučení neuronové sítě. [23]



Obr. 13: Funkce dropoutu (vpravo) [24]

4.2. Trénování neuronové sítě

V této kapitole bude stručně vysvětleno učení neuronové sítě, křížová validace a přeučení.

4.2.1. Učení neuronové sítě

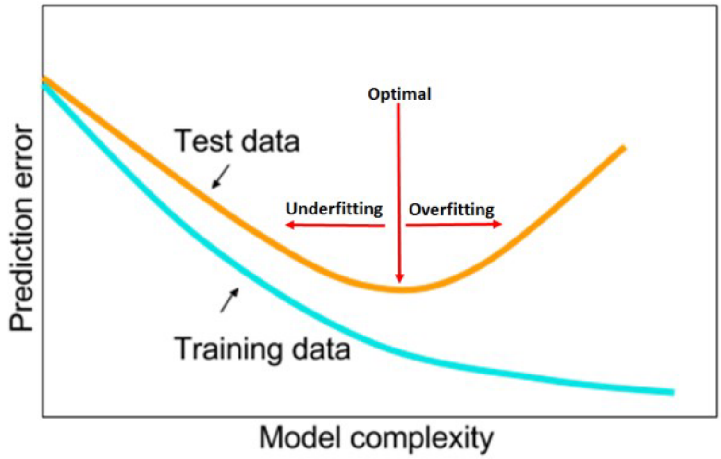
Každé dítě se s pomocí rodičů nejprve učí rozpoznávat předměty, které mu dávají zpětnou vazbu. Tímto principem byla inspirována i tato neuronová síť, která dostává informace o úspěšnosti své predikce. Při učení neuronové sítě se tak používá algoritmus zpětného šíření. Účelem je optimalizovat její vnitřní parametry a zároveň minimalizovat chybu predikce. Samotný průběh učení se pak skládá z mnoha cyklů/iterací, vyhodnocení chyby predikce, zpětného šíření a optimalizace parametrů. [25][26]

4.2.2. Křížová validace

Křížová validace je metoda, která zkontroluje, jak moc vstupní data ovlivňují daný model. Mezi metody křížové validace patří *k*-násobná křížová validace. Při ní jsou trénovací data rozdělena na *k* částí a následně se jedna z těchto částí použije k testování a zbylé budou použity k trénování. Celý postup se *k*-krát opakuje tak, aby každá část byla použita k testování jednou. [11]

4.2.3. Přeučení

Za určitých podmínek, např. pokud proces učení není včas ukončen, se může síť dostat do stavu tzv. přeučení (overfitting). Je to stav, kdy se síť perfektně naučí testovací sadu, ale při použití na datech, která nebyla použita na trénink, by je síť měla být schopna správně určit, ale selhává. Při procesu učení jsou jediné informace, které získáváme jako zpětnou vazbu, trénovací a testovací chyba. Trénovací chyba je měřena při validaci validační množinou, částečně vycházející z trénovací množiny, a testovací chyba je měřena při validaci testovací množinou, která není zahrnuta v trénovací množině. Obě by ideálně měly konvergovat k nule. Ve skutečnosti však takto obvykle konvergují jen po určitou dobu, než začne testovací chyba opět růst, zatímco trénovací chyba stále klesá. Za účelem co největší eliminace přeučení se používají různé techniky jako například křížová validace, která je jednou z nejběžnějších. V této technice je trénovací množina rozdělena na jednu podmnožinu, na níž se systém trénuje, a jednu podmnožinu vyhodnocující aktuální naučenost. Tato oddělená data se nazývají validační množina. Během trénování je třeba zastavit ve správném momentě, kdy testovací chyba nejvíc konverguje k nule, a to ještě před tím, než testovací chyba začne stoupat. Jednou z možností je zjistit po každé iteraci aktuální chybu na validační množině a uložit si současné nastavení sítě, pokud bude lepší než v předchozí iteraci. Jestli síť nedosahuje lepších výsledků, pak se trénování zastaví a za nejlepší nastavení sítě se prohlásí poslední uložené parametry. [27][28]



Obr. 14: Přeučení [28]

4.3. Nástroje pro neuronové sítě

Nástroje určené k práci s konvolučními neuronovými sítěmi se používají ke zpracování matic různých rozměrů a obsahují implementace jednotlivých vrstev a prostředků potřebných k učení. Existují v podobě knihoven programovacích jazyků jako Python, C++ apod. V této práci byla zvolena knihovna Keras (pro Python), která je postavena nad knihovnou TensorFlow.

4.3.1. TensorFlow

TensorFlow je nejznámější open-source knihovna používaná k vytváření neuronových sítí a modelů hlubokého učení vyvinutá společností Google. Jedná se o kombinaci nízkoúrovňového výpočetního frameworku typu Theano, který je open-source knihovna a umožňuje efektivně definovat, optimalizovat a vyhodnocovat matematické výrazy zahrnující vícerozměrná pole. TensorFlow má vestavěnou podporu CNN, ale implementace RNN je složitá. TensorFlow je velmi výkonná knihovna, ale je obtížně pochopitelná pro vytváření neuronových sítí. Z toho důvodu byla vystavěna knihovna Keras pro uživatelské použití, což umožňuje jednodušší práci s neuronovými sítěmi. [29][30]

4.3.2. Keras

Keras běží nad open-source strojovými knihovnami jako TensorFlow pro modelování neuronových sítí s vysokou mírou abstrakce. Vyvinul ho výzkumník umělé inteligence ve společnosti Google jménem Francois Chollet. Jedná se o jednu z nejvýkonnějších a nejsnadněji použitelných knihoven pro jazyk Python, která je postavena nad knihovnami hlubokého učení jako Theano, TensorFlow atd. pro vytváření modelů hlubokého učení. Keras byl založen na minimální struktuře, která poskytovala čistý a snadný způsob definování modelů hlubokého učení založených na TensorFlow nebo Theano. Lze provádět výpočty i přes GPU. [23][30]

Výhody

Keras je vysoce výkonný a dynamický framework, který přináší následující výhody:

- větší podpora komunity.
- snadné testování.
- neuronové sítě Keras jsou napsány v uživatelsky přívětivém jazyce Python.
- Keras podporuje konvoluční i rekurentní sítě.
- modely hlubokého učení jsou diskrétní komponenty, což je možné kombinovat mnoha způsoby. [23]

Pro tuto práci byl zvolen z toho důvodu, že je schopen testovat různé přístupy a má širokou škálu různých architektur neuronových sítí.

5. Řešení úlohy

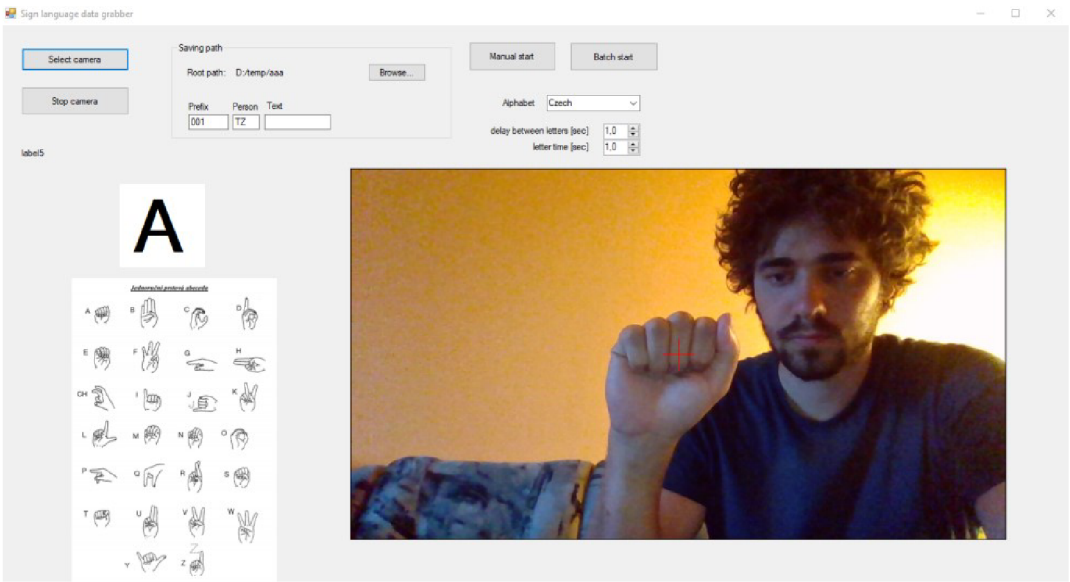
5.1. Získávání dat

Celkový počet figurantů je 30 osob (z toho 15 mužů a 15 žen) v různých věkových kategoriích (13 v období plné dospělosti, 12 v období zralosti a 5 starších). Figuranti byli natáčeni v nejrůznějších prostředích s různými světelnými podmínkami, kde i několikrát měnili místo. Změnou místa docházelo i ke změně pozadí, a to odstíny, barvami, světlem apod.

K této práce byla vytvořena aplikace, která umožňuje různé funkce, např. focení, ostřihnutí fotek, rotaci fotek nebo změnu měřítka fotek a tak dále. Pro získání dat potřebných pro výzkumnou část diplomové práce, bylo potřeba nahrát a uložit prstovou abecedu do počítače.

Díky této vytvořené aplikaci se zjednodušuje a usnadňuje práce s jednotlivými znaky (daty). V praxi to znamená, že se znaky vyfotí v určitých intervalech – lze tedy nastavit čas tak, aby mezi jednotlivými písmeny byla malá pauza, která dovolí figurantovi přecházet z pozice aktuálního písmena na písmeno další. To vše v nějakém čase, než se zase vyfotí a uloží nové.

Další užitečný přínos je štítkování (anglicky labeling), které fotky při jejich ukládání přejmenuje určitým písmenem tak, aby se snadněji rozpoznalo, o který znak prstové abecedy se jedná (později je vhodné pro trénování a testování). Mojí snahou bylo získat data pomocí znaků prstové abecedy. Svůj zájem projevili neslyšící i slyšící lidé, kteří se do tohoto diplomového projektu zapojili. Pomocí webové kamery jsme natočili videa s využitím prstové abecedy. Zúčastnění lidé viděli na obrazovce monitoru abecedu a následně znakovali podle abecedy.



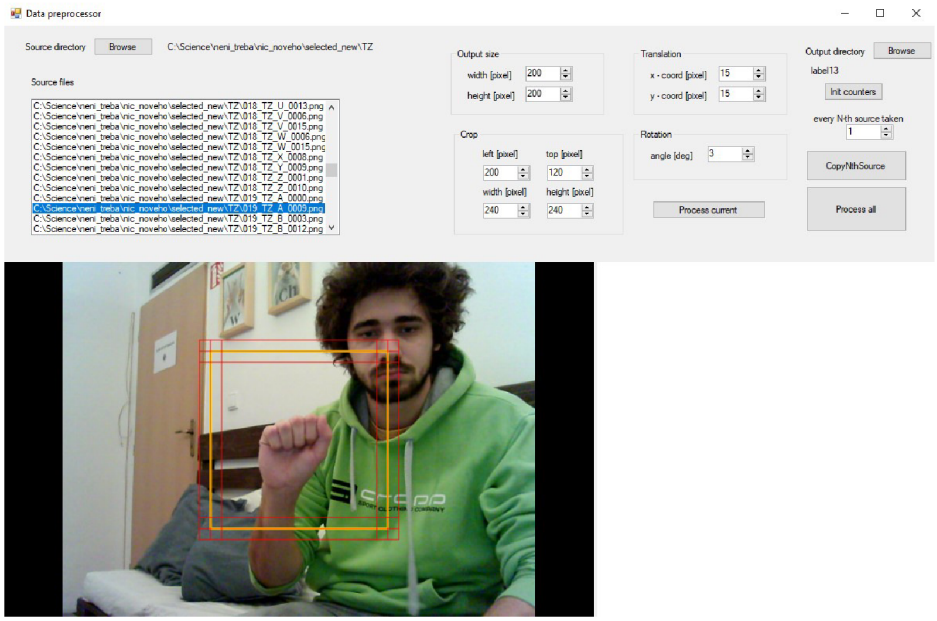
Obr. 15: Aplikace při natáčení prstové abecedy

Nebylo organizačně náročné zajistit zájemce, kteří jsou praváci a ne leváci. Mezi neslyšícími i slyšícími je více praváků než leváků. Lateralita je tedy v tomto případě pro tvorbu znaků velmi důležitá. Většina lidí používá pro znakový jazyk pravou ruku, kdy ruka praváka je aktivní a levá ruka je pasivní. Proto bylo mou prioritou oslovit lidi, kteří používají prstovou abecedu nebo znaky v pravém znaku. To představovalo jistou obtíž, protože jsem musel upozornit účastníky, že musí používat pouze jednoruční abecedu. V České republice se zpravidla (většina Neslyšících) používá dvouruční abeceda. To znamená, že se používá abeceda pomocí obou rukou (levé a pravé). Některým lidem to vyhovovalo, ale pro některé to bylo hodně náročné, protože dosud nebyli zvyklí používat jednoruční prstovou abecedu.

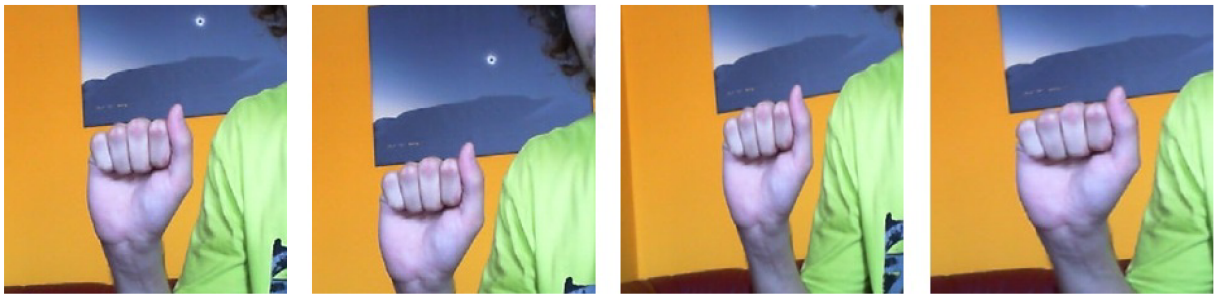
5.2. Augmentace dat

Množina dat, která se získala od 30 osob, není dostatečná pro trénování, protože existují jiné tvary ruky (dlouhé nebo krátké prsty, tloušťka apod.) a pohledy kamery na ruce (pozice artikulace – tj. poloha ruky vůči tělu při znakování, různé úhly apod.). K tomu, aby konvoluční neuronová síť fungovala perfektně, je třeba získat ještě mnoho dalších dat od dalších, jiných osob. Kvůli nedostatečnému času a opatřením vlády souvisejících s onemocněním COVID-19, a to hlavně kvůli omezení kontaktu s lidmi, bylo postupováno trochu jinak, a to pomocí augmentace dat.

Augmentace dat proběhla pro každý vzorový obrázek posuvem, rotací, změnou měřítka ve všech směrech, změnou měřítka v jedné ose a jejich kombinací. Posuvy obrázků se dějí v náhodném směru a uloží se jako další data, takže artikulace probíhá v různých pozicích. Rotace natočí obrazová data v různých malých stupních, jako by předloktí nebylo rovno, ale trochu natočeno. Změna měřítka ve všech směrech by se nahradila tím, že ruka by byla blíž ke kameře nebo dále od kamery. Změna měřítka v jedné ose může pomoci rozšiřovat množinu dat tím, že ruce budou tlustší nebo hubenější apod.



Obr. 16: Aplikace při augmentaci



Obr. 17: Ukázka dat po augmentaci

5.3. Rozdělení dat

Celkem jsem získal 580070 obrazových dat, která byla rozdělena do čtyř množin. Jedna velká trénovací množina se vytvořila k tréninku (577866 dat). Následně tři množiny sloužily k testování. U dvou testovacích množin (681 a 342 dat) byli vybráni dva figuranti (MK a TZ), kteří byli taky součástí tréninku, ale tato data z něho byla částečně oddělena. Jinými slovy bylo natáčeno jindy a jinde, ale se stejným figurantem v jiném oblečení. Tito figuranti byli zvoleni z určitých důvodů. Figurant MK je slyšící a má zvláštní tvar ruky, přesněji řečeno má dlouhé tenké prsty, dokonce má dlouhý palec. Další figurant TZ je neslyšící a byl přiřazen k testovací množině z důvodu opačného pohlaví. U zbylé testovací množiny (1181) byl zvolen figurant (PG), který není součástí tréninkové množiny. Figurant PG je neslyšící a dobře ovládá jednoruční prstovou abecedu. S ohledem na tuto výhodu a velkou množinu dat byl proto primárně právě on zvolen k validaci.

5.4. Návrh topologií konvoluční neuronové sítě

Navrhnout vhodnou topologii bylo obtížné. Ideální topologie by byla, kdyby měla nastavitelných parametrů (vah) co nejméně a zároveň nejlépe klasifikovala. Bylo zapotřebí několika návrhů, než se našla vhodná topologie. Návrh první topologie byl navržen náhodně a po menších krocích byl upraven tak, aby se blížil k ideálnímu návrhu topologie. Přesněji řečeno, byly nejprve přidány konvoluční, sdružovací a další vrstvy, dále byly zadány rozměry výstupy (neurony). Po výpočtu byla navržena další topologie, v níž byly přidány nebo odebrány některé vrstvy a zároveň byly různě upraveny rozměry výstupu vůči původní topologii. Po výpočtech bylo obtížné předpovědět, co je třeba upravit, protože například zdvojení konvoluční vrstvy někdy pomůže a někdy zase ne. Tyto malé úpravy uspořádání vrstev a výstupních rozměrů byly provedeny náhodně. Všechny topologie byly psány v programovacím jazyce Pythonu (obr. 18).

```
model = models.Sequential()
model.add(layers.Conv2D(12, (5, 5), activation='relu', input_shape=(224, 224, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(12, (5, 5), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(14, (3, 3), activation='relu'))
model.add(layers.Conv2D(14, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(18, (3, 3), activation='relu'))
model.add(layers.Conv2D(18, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(22, (3, 3), activation='relu'))
model.add(layers.Conv2D(22, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(20, activation='relu'))
model.add(layers.Dense(27, activation='softmax',kernel_regularizer=regularizers.l2(0.01)))
```

Obr. 18: Ukázka z jedné topologie v Pythonu

Návrhy topologií nazvanými třemi písmeny jsou znázorněny tabulkou (viz níže). Rozdíly mezi topologiemi jsou malé. Například topologie NNN má stejné uspořádání u topologie III, ale po třetí sdružovací vrstvě byly upraveny výstupní rozměry, čímž došlo ke změně parametrů. Topologie PPP má jiné uspořádání, tedy má oproti ostatním topologiím více konvolučních vrstev. V topologiích QQQ, KKK, OOO, RRR a SSS je o plně propojenou vrstvu navíc a výstupní rozměry byly různě upraveny na začátku nebo konci.

NNN			III		
Vrstvy	Výstup. rozměry	parametry	Vrstvy	Výstup. rozměry	parametry
conv2d	(222,222,12)	336	conv2d	(222,222,12)	336
max_pool	(111,111,12)		max_pool	(111,111,12)	
conv2d	(109,109,12)	1308	conv2d	(109,109,12)	1308
max_pool	(54,54,12)		max_pool	(54,54,12)	
conv2d	(52,52,14)	1526	conv2d	(52,52,14)	1526
conv2d	(50,50,14)	1778	conv2d	(50,50,14)	1778
max_pool	(25,25,14)		max_pool	(25,25,14)	
conv2d	(23,23,18)	2286	conv2d	(23,23,16)	2032
conv2d	(21,21,18)	2934	conv2d	(21,21,16)	2320
max_pool	(10,10,18)		max_pool	(10,10,16)	
conv2d	(8,8,22)	3586	conv2d	(8,8,20)	2900
conv2d	(6,6,22)	4378	conv2d	(6,6,20)	3620
max_pool	(3,3,22)		max_pool	(3,3,20)	
flatten	(198)		flatten	(180)	
dropout	(198)		dropout	(180)	
dense	(27)	5373	dense	(27)	4887

Tab. 1: Návrh topologií

PPP			QQQ		
Vrstvy	Výstup. rozměry	parametry	Vrstvy	Výstup. rozměry	parametry
conv2d	(222,222,12)	336	conv2d	(222,222,12)	336
conv2d	(220,220,12)	1308	max_pool	(111,111,12)	
max_pool	(110,110,12)		conv2d	(109,109,12)	1308
conv2d	(108,108,12)	1308	max_pool	(54,54,12)	
conv2d	(106,106,12)	1308	conv2d	(52,52,14)	1526
max_pool	(53,53,12)		conv2d	(50,50,14)	1778
conv2d	(51,51,14)	1526	max_pool	(25,25,14)	
conv2d	(49,49,14)	1778	conv2d	(23,23,18)	2286
max_pool	(24,24,14)		conv2d	(21,21,18)	2934
conv2d	(22,22,18)	2286	max_pool	(10,10,18)	
conv2d	(20,20,18)	2934	conv2d	(8,8,22)	3586
max_pool	(10,10,18)		conv2d	(6,6,22)	4378
conv2d	(8,8,22)	3586	max_pool	(3,3,22)	
conv2d	(6,6,22)	4378	flatten	(198)	
max_pool	(3,3,22)		dropout	(198)	
flatten	(198)		dense	(10)	1990
dropout	(198)		dense	(27)	297
dense	(20)	3980			
dense	(27)	567			

Tab. 2: Návrh topologií

KKK			OOO		
Vrstvy	Výstup. rozměry	parametry	Vrstvy	Výstup. rozměry	parametry
conv2d	(222,222,12)	336	conv2d	(220,220,12)	912
max_pool	(111,111,12)		max_pool	(110,110,12)	
conv2d	(109,109,12)	1308	conv2d	(106,106,12)	3612
max_pool	(54,54,12)		max_pool	(53,53,12)	
conv2d	(52,52,14)	1526	conv2d	(51,51,14)	1526
conv2d	(50,50,14)	1778	conv2d	(49,49,14)	1778
max_pool	(25,25,14)		max_pool	(24,24,14)	
conv2d	(23,23,18)	2286	conv2d	(22,22,18)	2286
conv2d	(21,21,18)	2934	conv2d	(20,20,18)	2934
max_pool	(10,10,18)		max_pool	(10,10,18)	
conv2d	(8,8,22)	3586	conv2d	(8,8,22)	3586
conv2d	(6,6,22)	4378	conv2d	(6,6,22)	4378
max_pool	(3,3,22)		max_pool	(3,3,22)	
flatten	(198)		flatten	(198)	
dropout	(198)		dropout	(198)	
dense	(20)	3980	dense	(20)	3980
dense	(27)	567	dense	(27)	567

Tab. 3: Návrh topologií

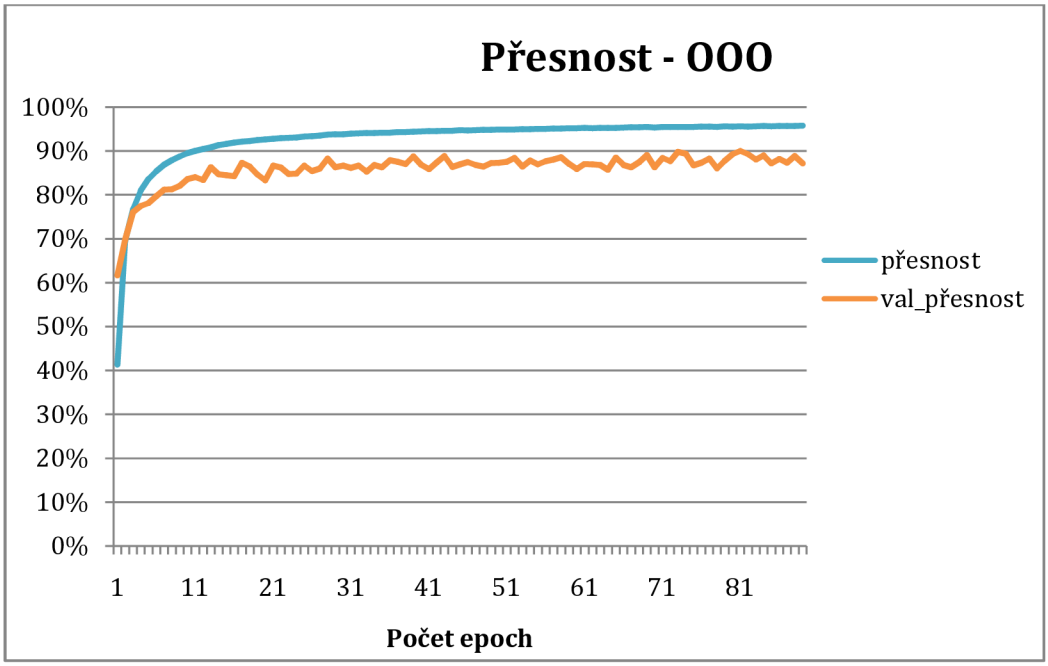
RRR			SSS		
Vrstvy	Výstup. rozměry	parametry	Vrstvy	Výstup. rozměry	parametry
conv2d	(222,222,12)	336	conv2d	(222,222,12)	336
max_pool	(111,111,12)		max_pool	(111,111,12)	
conv2d	(109,109,12)	1308	conv2d	(109,109,12)	1308
max_pool	(54,54,12)		max_pool	(54,54,12)	
conv2d	(52,52,14)	1526	conv2d	(52,52,14)	1526
conv2d	(50,50,14)	1778	conv2d	(50,50,14)	1778
max_pool	(25,25,14)		max_pool	(25,25,14)	
conv2d	(23,23,18)	2286	conv2d	(23,23,18)	2286
conv2d	(21,21,18)	2934	conv2d	(21,21,18)	2934
max_pool	(10,10,18)		max_pool	(10,10,18)	
conv2d	(8,8,22)	3586	conv2d	(8,8,22)	3586
conv2d	(6,6,22)	4378	conv2d	(6,6,22)	4378
max_pool	(3,3,22)		max_pool	(3,3,22)	
flatten	(198)		flatten	(198)	
dropout	(198)		dropout	(198)	
dense	(12)	2388	dense	(14)	2786
dense	(27)	351	dense	(27)	405

Tab. 4: Návrh topologií

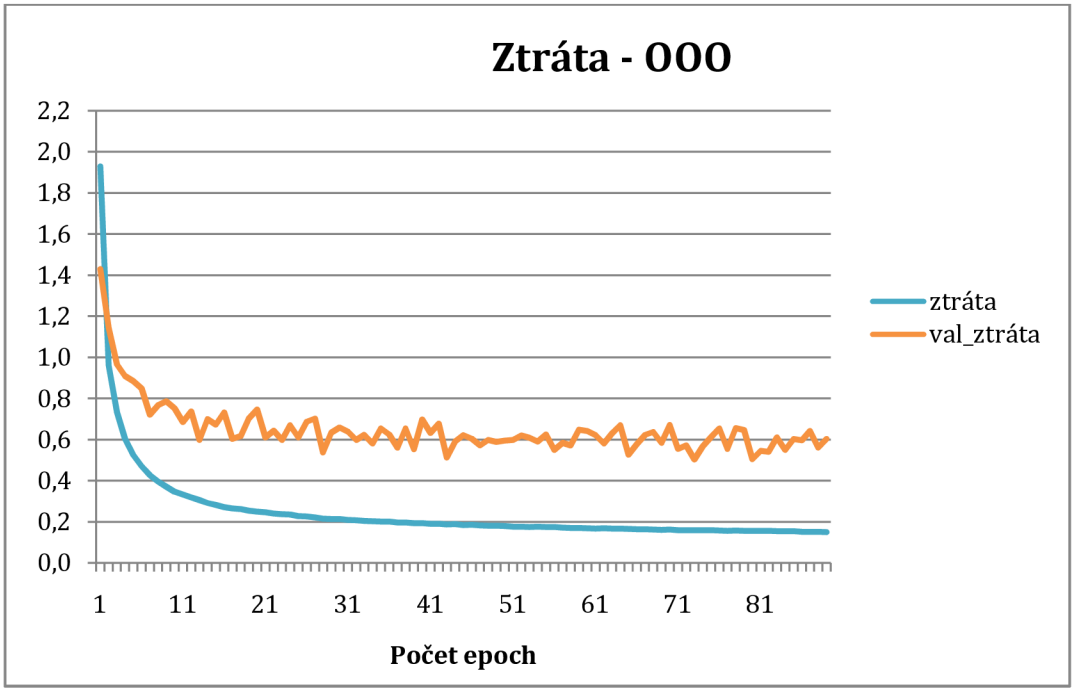
Vysvětlivky zkratek:
conv2d – *konvoluční vrstva*, max_pool – *sdružovací vrstva*, flatten – *flatten vrstva,* dropout–*dropout vrstva*, dense – *plně propojená vrstva*

5.5. Průběh tréninku

Po prvních deseti epochách (tj. učících cyklech) se uložil první model a začala se provádět validace u každé epochy. Výsledky by se měly nejprve blížit k ideálním hodnotám a pak by mělo dojít k přeučení, ale v tomto případě nedošlo k velkému přeučení, což díky dropoutu a regularizaci není moc vidět. Průběh tréninku a validace byl zapsán a převeden do dvou grafů (viz níže), které ukazují přesnost a ztrátu u jedné topologie, tedy topologie OOO, která z nich nejlépe klasifikuje. V grafu jsou popsány dvě přesnosti, popř. dvě ztráty, jedna z nich byla měřena validační množinou, částečně z trénovací množiny, a druhá testovací množinou.



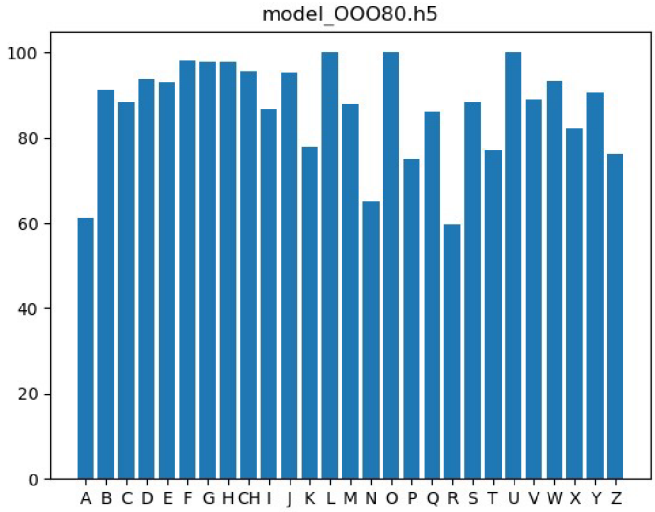
Graf 1: Přesnost u jedné topologie



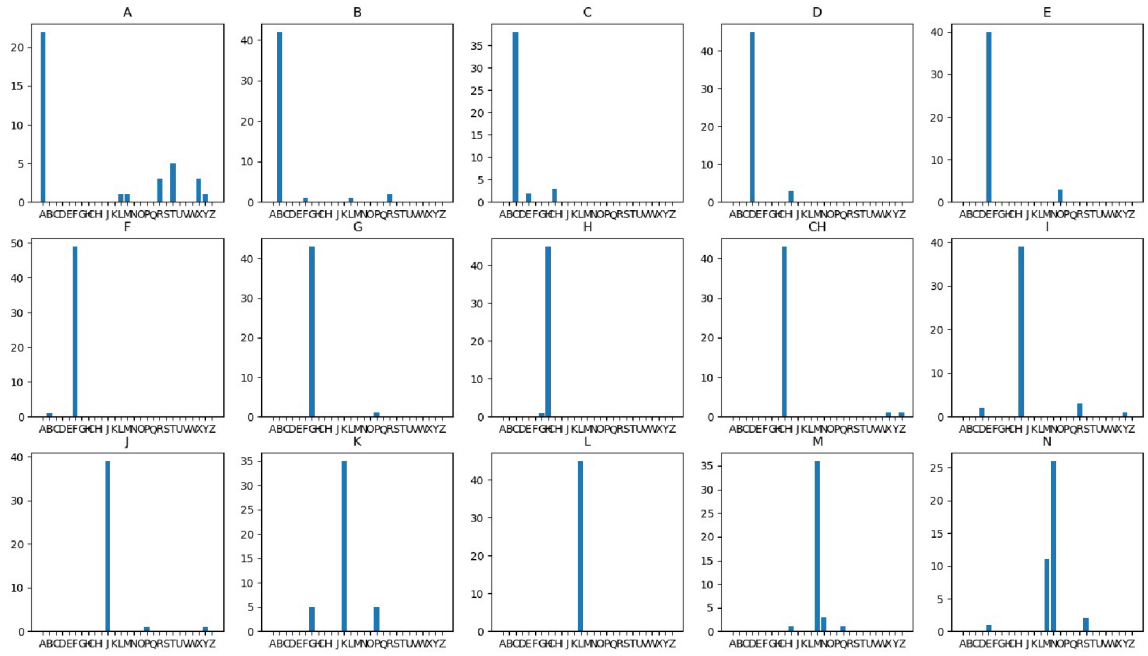
Graf 2: Ztráta u jedné topologie

5.6. Test

Po tréninku konvoluční neuronové sítě se data uloží a následně se vyhodnotí testovacími množinami, kde jsou tři množiny s různými figuranty (PG, MK a TZ), přičemž platí 1 figurant = 1 množina. Na obrázku je znázorněno, s jakou pravděpodobností byla jednotlivá písmena klasifikována a jak a kolik jiných písmen bylo rovněž detekováno u správného písmene. Kompletní výsledky jsou zobrazeny a popsány v následující kapitole.



Obr. 19: Pravděpodobnost úspěšné klasifikace



Obr. 20: Četnosti detekce jiných písmen u jednoho písmena

6. Dosažené výsledky a vyhodnocení

K tomu, aby se vybral nejvhodnější návrh topologie ke klasifikaci, je třeba vyhodnotit každý model, který se uložil po každé epoše u určité topologie, a navzájem je mezi sebou porovnat.

6.1. Vyhodnocení modelu

Ideálním návrhem topologie by byl model mající nejvyšší přesnost a nejnižší ztrátu v optimální epoše. Úkolem této části je tedy najít v každé topologii nejlepší model a z nich následně vybrat ten nejlepší, u něhož byla nejvyšší přesnost, aniž by nedošlo k přeučení. Každý model byl vyhodnocen a z jejich výsledků pak byl zvolen nejvhodnější model podle přesnosti a ztráty podstatné u validace. Jako nejvhodnější model byla zvolena topologie OOO. Všechny přesnosti a ztráty byly znázorněny v tabulce č. 3. Z této tabulky vyplývá, že přesnost byla nejvyšší, popř. ztráta byla nejnižší, v předchozí epoše než v poslední, v níž došlo k malému přeučení.

	acc	val_acc	loss	val_loss	epocha
NNN	94,01%	86,71%	0,2313	0,6299	40
NNN (max_val)	93,60%	88,15%	0,2468	0,6371	29
III	91,66%	85,61%	0,3046	0,6894	39
III (max_val)	90,65%	87,04%	0,3387	0,6221	30
KKK	94,00%	85,86%	0,2043	0,6291	40
KKK (max_val)	93,83%	87,89%	0,2088	0,5275	37
OOO	95,74%	87,13%	0,1496	0,6030	80
OOO (max_val)	95,59%	90,01%	0,1551	0,5446	72
PPP	96,40%	85,27%	0,1280	0,7820	80
PPP (max_val)	95,96%	88,57%	0,1417	0,5743	57
QQQ	94,41%	88,23%	0,1902	0,7017	40
QQQ (max_val)	94,35%	89,25%	0,1933	0,5619	36
RRR	94,41%	87,72%	0,1884	0,6979	40
RRR (max_val)	94,40%	88,99%	0,1893	0,6291	38
SSS	92,22%	85,77%	0,2607	0,6952	16
SSS (max_val)	91,11%	86,71%	0,2932	0,6682	10

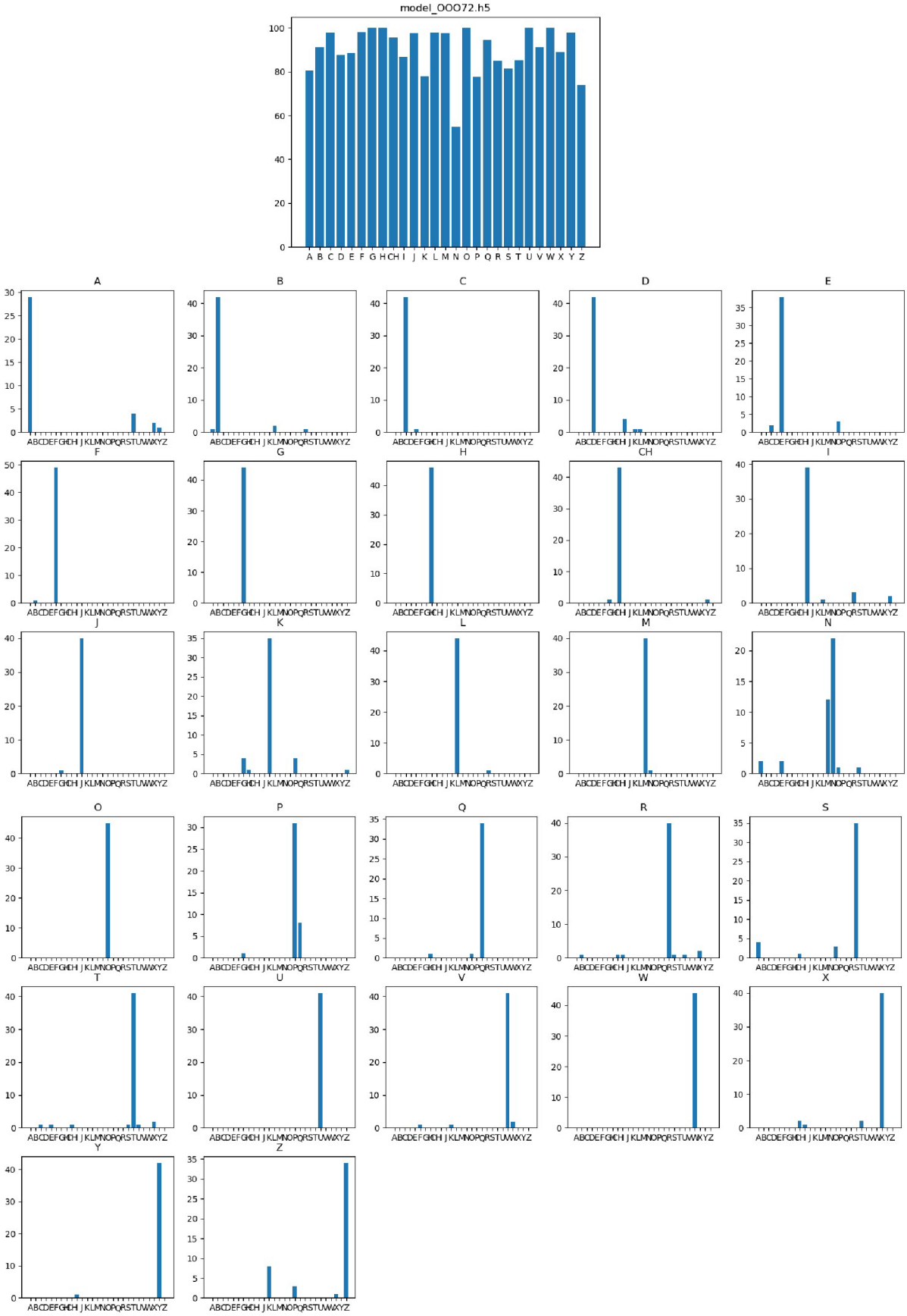
Tab. 5: Přehled přesnosti u jednotlivé topologie

Vysvětlivky zkratek:
acc – *přesnost,* loss – *ztráta,* val_acc – *přesnost u validace,* val_loss – *ztráta u validace,* max_val – *nejvyšší přesnost u validace*

6.2. Přehled výsledků

Tato část je věnována výsledkům každého figuranta. K tomu byla přidána kontingenční matice, která vytvoří přehled v kolika procentech u určitého písmena (vodorovně) byla klasifikována všechna jiná písmena (svisle).

Výsledky č. 1 – figurant PG

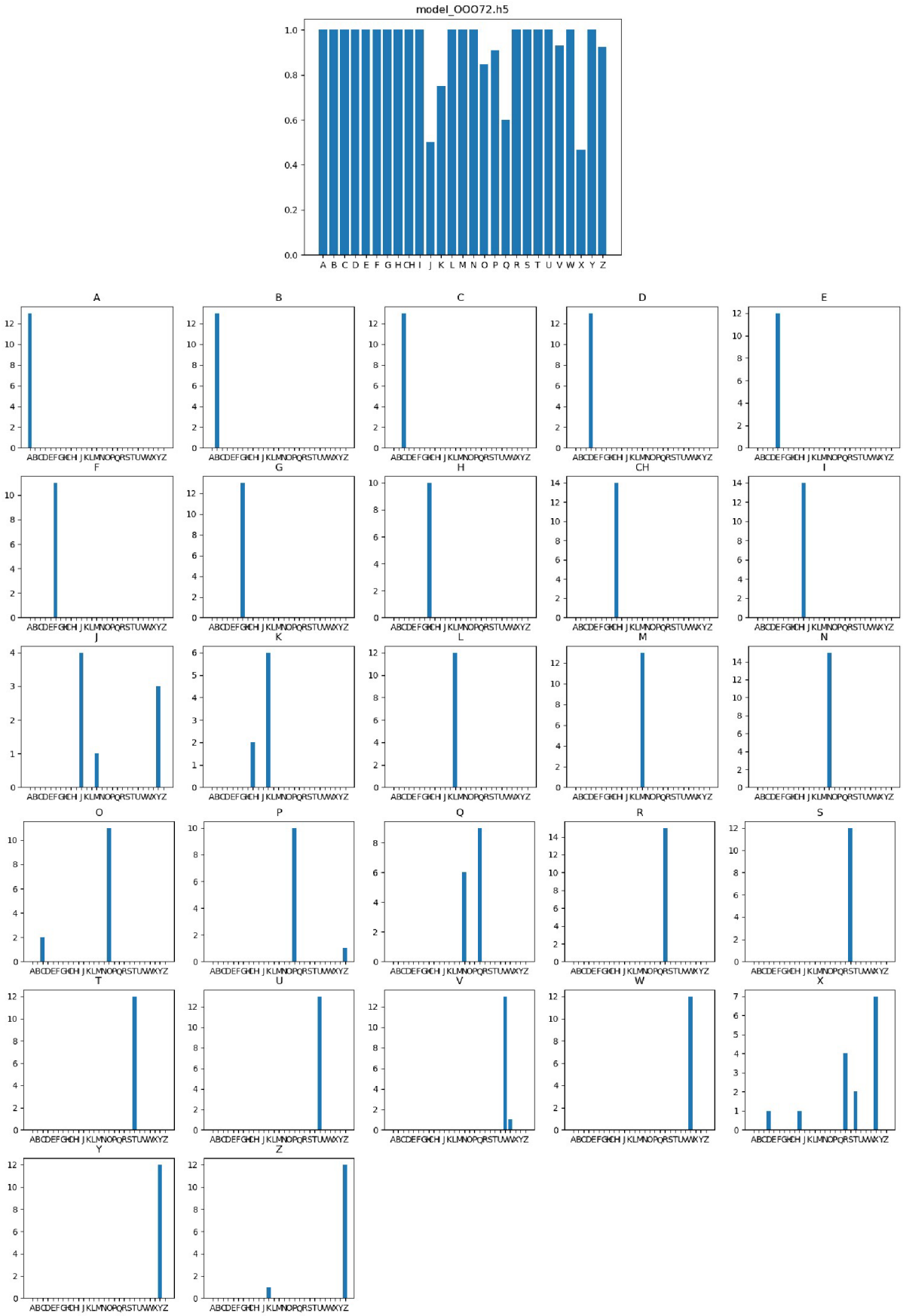


Pravděpodobnostní tabulka (Kontingenční matice) – figurant PG

	A	B	C	D	E	F	G	H	CH	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	80,6%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	8,3%	0,0%	0,0%	0,0%	4,4%	2,3%	0,0%
B	2,8%	91,3%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	4,4%	0,0%	0,0%	0,0%	0,0%	0,0%	2,1%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
C	0,0%	0,0%	97,7%	0,0%	2,3%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
D	0,0%	0,0%	0,0%	87,5%	0,0%	0,0%	0,0%	0,0%	0,0%	8,9%	0,0%	2,2%	2,2%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
E	0,0%	0,0%	4,7%	0,0%	88,4%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	6,7%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
F	0,0%	2,2%	0,0%	0,0%	0,0%	98,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
G	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
H	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
CH	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	2,3%	0,0%	95,6%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	2,3%	0,0%
I	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	86,7%	0,0%	0,0%	2,2%	0,0%	0,0%	0,0%	0,0%	0,0%	6,4%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	4,7%	0,0%
J	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	2,3%	0,0%	0,0%	0,0%	97,6%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
K	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	9,1%	2,2%	0,0%	0,0%	0,0%	77,8%	0,0%	0,0%	0,0%	0,0%	10,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	2,2%
L	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	97,8%	0,0%	0,0%	0,0%	0,0%	0,0%	2,1%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
M	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	97,6%	2,5%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
N	5,6%	0,0%	0,0%	0,0%	4,7%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	29,3%	55,0%	2,2%	0,0%	0,0%	0,0%	2,3%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
O	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
P	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	2,3%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	77,5%	22,2%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
Q	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	2,2%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	2,2%	0,0%	94,4%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
R	0,0%	2,2%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	2,2%	2,2%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	85,1%	2,3%	0,0%	2,4%	0,0%	0,0%	4,4%	0,0%	0,0%
S	11,1%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	2,2%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	6,7%	0,0%	0,0%	0,0%	81,4%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
T	0,0%	0,0%	2,3%	0,0%	2,3%	0,0%	0,0%	0,0%	2,2%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	2,3%	85,4%	2,4%	0,0%	0,0%	4,4%	0,0%	0,0%
U	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%
V	0,0%	0,0%	0,0%	0,0%	0,0%	2,0%	0,0%	0,0%	0,0%	0,0%	0,0%	2,2%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	91,1%	4,5%	0,0%	0,0%	0,0%
W	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%
X	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	4,4%	2,2%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	4,2%	0,0%	0,0%	0,0%	88,9%	0,0%	0,0%
Y	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	2,2%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	97,7%	0,0%
Z	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	17,8%	0,0%	0,0%	0,0%	0,0%	7,5%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	2,2%	0,0%	73,9%

Jak můžeme vidět v předchozí tabulce, tak písmeno A v kontingenční matici bylo klasifikováno na 80,6 % a tato neuronová síť podobně klasifikovala i písmena T (8,3 %), X (4,4 %) a Y (2,3 %). Tato čísla jsou velmi nízká oproti písmenu A, což ale není podstatné. V této matici byla nalezena nízká čísla u některých písmen, a to například u písmene P (77,5 %), ke které byla přiřazena další písmena G (2,3 %) a Q (22,2 %). Nebo u jiného písmene N (55,0 %), ke kterému byla přiřazena další písmena M (29,3 %), E (4,7 %), A (5,6 %), O (2,2 %) a S (2,3 %). Pravděpodobnost u písmena P je víc než třikrát vyšší než pravděpodobnost u písmena Q, k němuž bylo přiřazeno, což není velká chyba, ale pravděpodobnost u písmena M, které bylo přiřazeno k písmenu N a má více než polovinu pravděpodobnosti písmena N, může ovlivňovat celou klasifikaci. Příčiny a řešení tohoto problému jsou popsány v další kapitole 6.3.

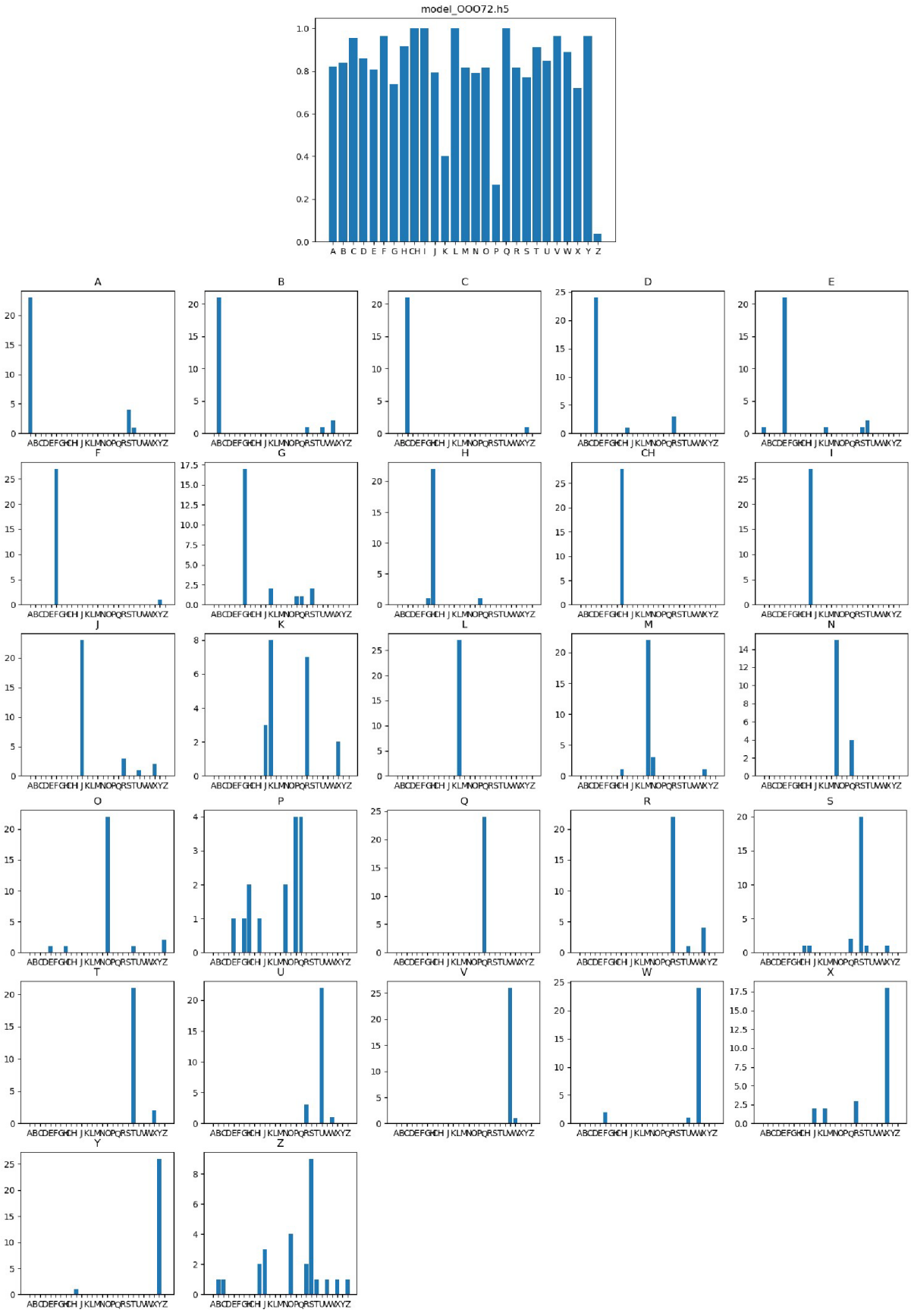
Výsledky č. 2 – figurant MK



Pravděpodobnostní tabulka (Kontingenční matice) – figurant MK

	A	B	C	D	E	F	G	H	CH	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
B	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
C	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
D	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
E	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
F	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
G	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
H	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
CH	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
I	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
J	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	50,0%	0,0%	0,0%	7,7%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	25,0%	0,0%
K	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	14,3%	0,0%	0,0%	75,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
L	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
M	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
N	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
O	0,0%	0,0%	15,4%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	84,6%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
P	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	90,9%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	7,7%
Q	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	40,0%	0,0%	0,0%	60,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
R	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
S	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
T	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
U	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%
V	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	92,9%	8,3%	0,0%	0,0%	0,0%
W	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%
X	0,0%	0,0%	0,0%	7,7%	0,0%	0,0%	0,0%	0,0%	0,0%	7,1%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	26,7%	0,0%	16,7%	0,0%	0,0%	0,0%	46,7%	0,0%	0,0%
Y	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%
Z	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	12,5%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	92,3%

Výsledky č. 3 – figurant TZ



Pravděpodobnostní tabulka (Kontingenční matice) – figurant TZ

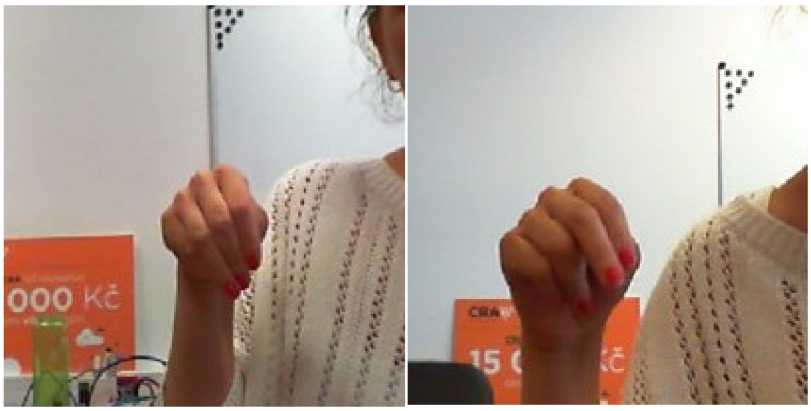
	A	B	C	D	E	F	G	H	CH	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	82,1%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	15,4%	4,3%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
B	0,0%	84,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	3,7%	0,0%	0,0%	3,8%	0,0%	7,4%	0,0%	0,0%	0,0%
C	0,0%	0,0%	95,5%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	3,7%	0,0%
D	0,0%	0,0%	0,0%	85,7%	0,0%	0,0%	0,0%	0,0%	0,0%	3,7%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	11,1%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
E	3,6%	0,0%	0,0%	0,0%	80,8%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	3,7%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	3,8%	8,7%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
F	0,0%	0,0%	0,0%	0,0%	0,0%	96,4%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	3,7%	0,0%
G	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	73,9%	0,0%	0,0%	0,0%	0,0%	10,0%	0,0%	0,0%	0,0%	0,0%	6,7%	4,2%	0,0%	7,7%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
H	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	4,3%	91,7%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	6,7%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
CH	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
I	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
J	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	79,3%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	11,1%	0,0%	0,0%	3,8%	0,0%	0,0%	8,0%	0,0%	0,0%
K	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	10,3%	40,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	25,9%	0,0%	0,0%	0,0%	0,0%	0,0%	8,0%	0,0%	0,0%
L	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
M	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	3,6%	0,0%	0,0%	0,0%	0,0%	81,5%	15,8%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	4,0%	0,0%	0,0%
N	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	78,9%	0,0%	0,0%	16,7%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
O	0,0%	0,0%	0,0%	0,0%	3,8%	0,0%	0,0%	4,2%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	81,5%	0,0%	0,0%	0,0%	0,0%	4,3%	0,0%	0,0%	0,0%	0,0%	0,0%	7,7%
P	0,0%	0,0%	0,0%	0,0%	3,8%	0,0%	4,3%	8,3%	0,0%	3,7%	0,0%	0,0%	0,0%	0,0%	10,5%	0,0%	26,7%	16,7%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
Q	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
R	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	81,5%	0,0%	0,0%	3,8%	0,0%	0,0%	16,0%	0,0%	0,0%
S	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	3,6%	3,7%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	8,3%	0,0%	76,9%	4,3%	0,0%	0,0%	0,0%	4,0%	0,0%	0,0%
T	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	91,3%	0,0%	0,0%	0,0%	8,0%	0,0%	0,0%
U	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	11,1%	0,0%	0,0%	84,6%	0,0%	3,7%	0,0%	0,0%	0,0%
V	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	96,3%	3,7%	0,0%	0,0%	0,0%
W	0,0%	0,0%	0,0%	0,0%	0,0%	7,1%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	3,8%	0,0%	88,9%	0,0%	0,0%	0,0%
X	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	6,9%	0,0%	7,4%	0,0%	0,0%	0,0%	0,0%	0,0%	11,1%	0,0%	0,0%	0,0%	0,0%	0,0%	72,0%	0,0%	0,0%
Y	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	3,7%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	96,3%	0,0%
Z	0,0%	4,0%	4,5%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	7,4%	10,3%	0,0%	0,0%	0,0%	0,0%	14,8%	0,0%	0,0%	7,4%	34,6%	4,3%	0,0%	3,7%	0,0%	4,0%	0,0%	3,8%

6.3. Vyhodnocení jednotlivého figuranta

K tomu, aby celá neuronová síť zafungovala dobře, je třeba zjistit, proč byla ke klasifikaci jednoho písmene přiřazena i jiná písmena, a to proto, aby se v příštím procesu vědělo, na co dát pozor, například nedostatečně velká množina, chyby při natáčení apod. Tato část podkapitoly tomu byla věnována a bylo zjištěno, proč u jednotlivého figuranta došlo k detekci jiného písmene než toho, které bylo znakováno.

Figurant PG

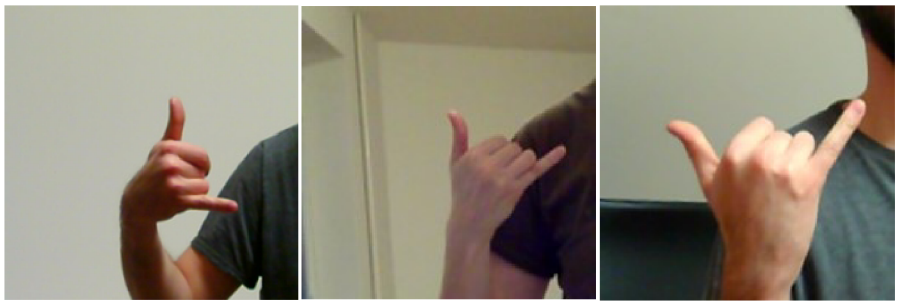
Ve výsledku bylo znázorněno, že pravděpodobnost u písmene N se vyskytla v 55,0 %. Podle grafu bylo k písmenu N přiřazeno i jiné písmeno, a to M. Z obrázku č. 21 je patrné, že se tato písmena velmi podobají. K dosažení přesnějšího výsledku, a to i když neuronová síť správně klasifikovala, je proto vhodné rozšířit trénovací množinu.



Obr. 21: Podobná písmena M (vlevo) a N (vpravo) u figuranta PG

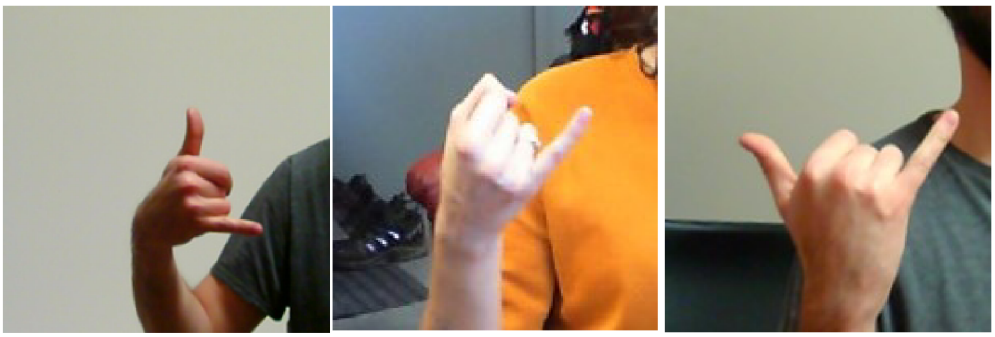
Figurant MK

Jak bylo zmíněno, je tato testovací množina výjimečná. Pravděpodobnost u písmena J byla 50,0 % a k němu bylo přiřazeno písmeno Y. U písmena J se často vyskytl problém v samotném provádění jeho pohybu, který většinou provádějí neslyšící. Při snímání do trénovací či testovací množiny je třeba, aby ruka byla v klidu. Na obrázku č. 22 je snímán začátek a konec pohybu písmena J, a právě na konci vypadá stejně jak písmeno Y.



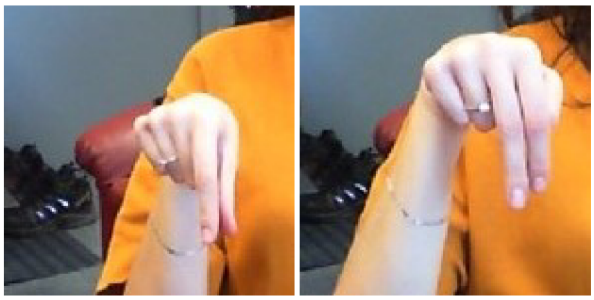
Obr. 22:Začátek a konec pohybu písmena J (vlevo, uprostřed) a písmeno Y (vpravo)

Na obrázku č. 23 je vidět, že písmeno J u figuranta MK, kterému je přiraženo i písmeno Y, nebylo jasné. Kvůli nesprávnému snímání, jež může velmi ovlivnit validaci, je nutné překontrolovat celou trénovací/testovací množinu a smazat nesprávná snímaní, v horším případě znovu správně vytvořit množinu.



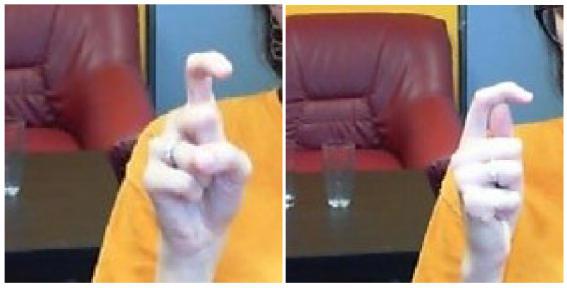
Obr. 23: Písmena J v trénovací množině (vlevo), u figuranta MK (uprostřed) a Y (vpravo)

Další problém byl u písmena Q (pravděpodobnost 60,0 %), ke kterému při klasifikaci bylo přiřazeno písmeno N. Kvůli figurantově dlouhému palci skoro vypadá písmeno N, jež bylo znázorněno na obrázku č. 24.

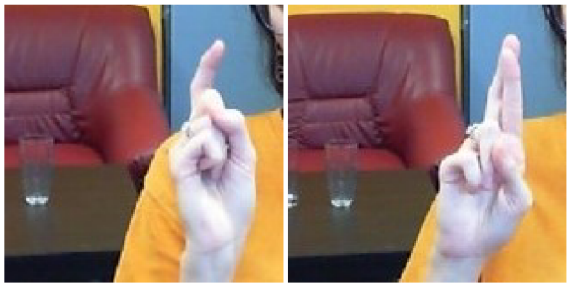


Obr. 24: Písmena Q (vlevo) a N (vpravo)

Další zajímavý případ se objevil u písmena X (pravděpodobnost 46,7 %). Figurant MK má zvláštní dlouhé prsty, díky nimž může občas dojít k chybě. Písmeno X se tak podobá písmenům R a T (obr. 25 a 26).



Obr. 25: Písmena X (vlevo) a T (vpravo)

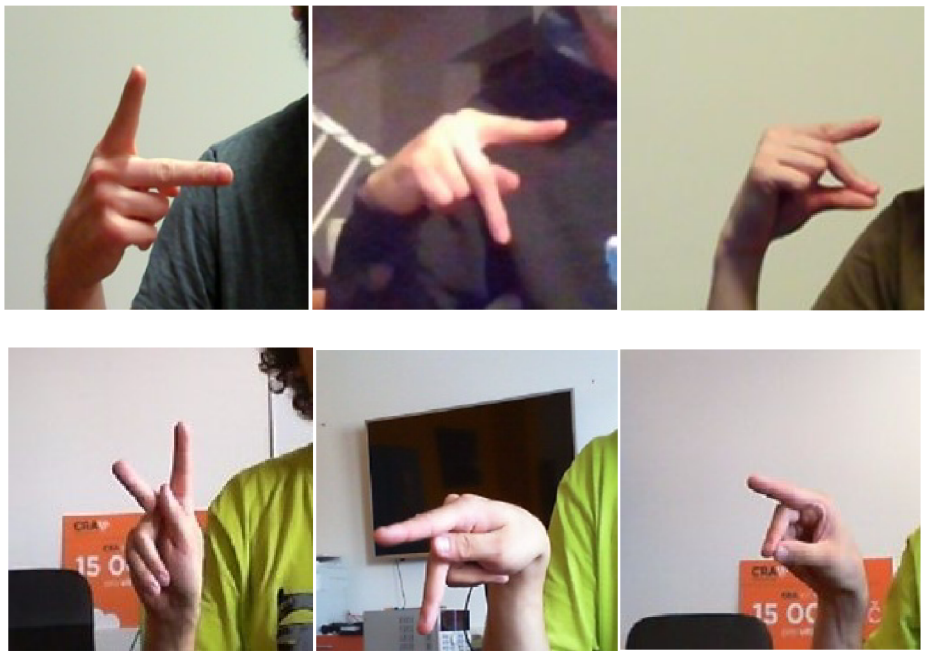


Obr. 26: Písmena X (vlevo) a R (vpravo)

K tomu, aby se dalo vyhnout podobné situaci u písmen Q a X, je dobré ještě rozšířit trénovací množinu.

Figurant TZ

U písmen K, P a Z validace nedopadla tak dobře, jelikož při ní bylo zjištěno, že figurant TZ znakoval sice pravou rukou, ale orientace ruky byla jiná (obr. 27), a v trénovací množině jsou pouze písmena K a P, která byla znakována jinou orientací ruky, a to pouze u jediného figuranta. Důvodem orientace ruky je prostor natáčení, ve kterém figuranti vybrali orientaci ruky, při které se dobře cítí. Pravděpodobnost klasifikace u těchto písmen byla nízká z důvodu malého množství těchto písmen v trénovací množině.



Obr. 27: Písmena v trénovací množině (nahoře) a u figuranta TZ (dole)

7. Závěr

Práce se zabývala klasifikací jednoruční prstové abecedy s použitím metod strojového učení, přesněji řečeno konvolučních neuronových sítí. Součástí práce také byly různé možnosti použití různých metod klasifikace. Cílem práce bylo vybrat vhodnou metodu pro klasifikaci znaků jednoruční prstové abecedy a navrhnout topologie pro tuto klasifikaci ve vhodném programu.

Konvoluční neuronové sítě byly vybrány jako vhodná metoda pro klasifikaci znaků jednoruční prstové abecedy. Tato metoda má řadu výhod, především dobře zobecňuje z tréninkových dat, což je nezbytné kvůli různým tvarům ruky jednotlivých znakujících osob, dále je do určité míry invariantní vůči posunutí, natočení a také změně měřítka. V klasifikačních úlohách, které zpracovávají data lišící se topologií, a nikoliv běžnými příznaky jako je barva nebo textura, jsou proto konvoluční neuronové sítě aktuálně nejrozšířenější a nejpoužívanější. Za tímto účelem byla použita knihovna Keras, která je postavena na kvalitní knihovně (TensorFlow), a proto má široké možnosti využití, je výkonná a přitom uživatelsky přívětivá. Nadále je knihovna Keras open-source, takž není potřeba licence. Celá neuronová síť byla napsána v Pythonu.

Data byla získána snímáním jednoruční prstové abecedy a po augmentaci dat byla množina dostatečně velká pro trénink a test. Po vyhodnocení byly zjištěny některé nedostatky, které je třeba doplnit dalšími daty. Při snímání je třeba dát pozor například na písmena J nebo K, u kterých figuranti používají pohyb, kvůli kterému se mohou vytvořit nevhodná data. Další nedostatek spočíval v opačné orientaci ruky u písmen, která se málo nebo vůbec nenachází v trénovací množině (například jiné varianty znaků pro některá písmena). Mimo zmíněné nedostatky je ale nutné vyzdvihnout, že validace primární testovací množinou, ve které figurant PG postupoval správně, proběhla úspěšně.

Návrh topologie byl obtížný, protože cílem bylo nalezení ideálního návrhu topologie a její epochy, v níž byla přesnost nejvyšší. Byly navrženy různé topologie s malými úpravami uspořádání vrstev a výstupních rozměrů. Po vyhodnocení byly vzájemně porovnány mezi sebou podle přesnosti měřenou testovací množinou a po porovnání výsledků byla pro účely této práce zvolena jako nejvhodnější topologie OOO.

Zvolená metoda splňuje požadavky na dostatečně přesnou metodu pro klasifikaci znaků jednoruční prstové abecedy pomocí zpracování obrazových dat. Vzhledem k vytvoření množiny dat a návrhu topologie v Pythonu úspěšně klasifikuje znaky jednoruční prstovou abecedu a splňuje všechny cíle práce.

Seznam použité literatury

[1] GUENTHER, Nick a Matthias SCHONLAU. Support Vector Machines. *The Stata Journal*. 2016, **16**(4), 917-937. Dostupné z: doi:10.1177/1536867X1601600407

[2] IPPOLITO, Pier Paolo. SVM: Feature Selection and Kernels. *Towards Data Science Inc.* [online]. 2019-06-03 [cit. 2021-02-02]. Dostupné z: https://towardsdatascience.com/svm-feature-selection-and-kernels-840781cc1a6c

[3] ZHANG, Grace. What is the kernel trick? Why is it important? *Medium* [online]. 2018-11-11 [cit. 2021-04-22]. Dostupné z: https://medium.com/analytics-vidhya/how-to-classify-non-linear-data-to-linear-data-bb2df1a6b781

[4] SHAIER, Sagi. ML Algorithms: One SD (σ)- Instance-based Algorithms. *Towards Data Science Inc.* [online]. 2019-02-02 [cit. 2021-03-23]. Dostupné z: https://towardsdatascience.com/ml-algorithms-one-sd-%CF%83-instance-based-algorithms-4349224ed4f3

[5] AHA, David W., Dennis KIBLER a Marc K. ALBERT. Instance-based learning algorithms. *Machine Learning*. 1991, **6**(1), 37-66. ISSN 0885-6125. Dostupné z: doi:10.1007/BF00153759

[6] Instance-based learning. *Slide Share* [online]. 2021-04-02 [cit. 2021-05-26]. Dostupné z: https://www.slideshare.net/swapnac12/instance-based-learning-245522096

[7] HOBSON, Gregory L., S. Richard F. SIMS, Paul D. GADER, James M. KELLER a Firooz A. SADJADI. *Minimum average correlation energy (MACE) prefilter network for automatic target recognition*. 1994-7-29, , 402-409. Dostupné z: doi:10.1117/12.181037

[8] VERMA, Gaurav, S. Richard F. SIMS, Paul D. GADER, James M. KELLER a Firooz A. SADJADI. FINGER KNUCKLE PRINT BASED VERIFICATION USING MINIMUM AVERAGE CORRELATION ENERGY FILTER. *International Journal of Electronic Commerce Studies*. 2014, 1994-7-29, **5**(2), 233-246. ISSN 20739729. Dostupné z: doi:10.7903/ijecs.1089

[9] MAHALANOBIS, A. a David P. CASASENT. Performance evaluation of minimum average correlation energy filters. *Applied Optics*. 1991, **30**(5). ISSN 0003-6935. Dostupné z: doi:10.1364/AO.30.000561

[10] *Neural Network* [online]. [cit. 2021-04-10]. Dostupné z: https://www.byclb.com/TR/Tutorials/neural_networks/ch7_1.htm

[11] HOLČÍK, Jiří, KOMENDA, Martin (eds.) a kol. *Matematická biologie: e-learningová učebnice* [online]. 1. vydání. Brno: Masarykova univerzita, 2015. ISBN 978-80-210-8095-9.

[12] MAHAJAN, Pooja. Fully Connected vs Convolutional Neural Networks. *Medium* [online]. 2020-10-23 [cit. 2021-04-13]. Dostupné z: https://medium.com/swlh/fully-connected-vs-convolutional-neural-networks-813ca7bc6ee5

[13] UCHIDA, Kazutaka, Masayuki TANAKA a Masatoshi OKUTOMI. Coupled convolution layer for convolutional neural network. *Neural Networks*. 2018, **105**, 197-205. ISSN 08936080. Dostupné z: doi:10.1016/j.neunet.2018.05.002

[14] PILÁT, Martin. *Přírodou inspirované algoritmy* [online]. [cit. 2021-04-20]. Dostupné z: https://martinpilat.com/cs/prirodou-inspirovane-algoritmy/

[15] *Convolutional Neural Networks* [online]. [cit. 2021-04-02]. Dostupné z: https://cs231n.github.io/convolutional-networks/

[16] BARUAH, Indraneel Dutta. Activation Functions and Loss Functions for neural networks — How to pick the right one? *Medium* [online]. 2021-07-21 [cit. 2021-07-30]. Dostupné z: https://medium.com/analytics-vidhya/activation-functions-and-loss-functions-for-neural-networks-how-to-pick-the-right-one-542e1dd523e0

[17] SHARMA, Nikita. Exploring Activation and Loss Functions in Machine Learning. *Heartbeat* [online]. 2020-08-11 [cit. 2021-05-13]. Dostupné z: https://heartbeat.fritz.ai/exploring-activation-and-loss-functions-in-machine-learning-39d5cb3ba1fc

[18] Obrázek z webové stránky. [online]. [cit. 2021-05-16]. Dostupné z: https://iot4beginners.com/deep-learning-and-the-internet-of-things/

[19] SHARMA, Palash. Keras Dense Layer Explained for Beginners. *MLK* [online]. 2020-10-20 [cit. 2021-04-28]. Dostupné z: https://machinelearningknowledge.ai/keras-dense-layer-explained-for-beginners/

[20] Obrázek z webové stránky. [online]. 2006-07-19 [cit. 2021-04-30]. Dostupné z: https://commons.wikimedia.org/wiki/File:Konvoluce_2rozm_diskretni.jpg

[21] YALÇIN, Orhan G. 4 Pre-Trained CNN Models to Use for Computer Vision with Transfer Learning. *Towards Data Science Inc.* [online]. 2020-9-23 [cit. 2021-07-12]. Dostupné z: https://towardsdatascience.com/ml-algorithms-one-sd-%CF%83-instance-based-algorithms-4349224ed4f3

[22] Obrázek z webové stránky [online]. [cit. 2021-07-30]. Dostupné z: https://miro.medium.com/max/3288/1*uAeANQIOQPqWZnnuH-VEyw.jpeg

[23] Keras. *Tutorials Point* [online]. [cit. 2021-7-06]. Dostupné z: https://www.tutorialspoint.com/keras/

[24] GYLBERTH, Roan. Understanding Dropout. *Medium* [online]. 2019-07-21 [cit. 2021-06-13]. Dostupné z: https://medium.com/konvergen/understanding-dropout-ddb60c9f98aa

[25] QUEGUINER, Jean-Louis. What does Training Neural Networks mean? *OVHcloud* [online]. 2020-04-22 [cit. 2021-05-28]. Dostupné z: https://www.ovh.com/blog/what-does-training-neural-networks-mean/

[26] BROWNIEE, Jason. Why Training a Neural Network Is Hard. *Machine Learning Mastery* [online]. 2019-05-01 [cit. 2021-06-13]. Dostupné z: https://machinelearningmastery.com/why-training-a-neural-network-is-hard/

[27] GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. Deep Learning. *MIT Press* [online]. [cit. 2021-06-20]. Dostupné z: https://www.deeplearningbook.org/

[28] SETH, Yashu. A Disciplined Approach to Neural Network Hyper-Parameters: Learning Rate, Batch Size, Momentum, and Weight Decay – Paper Dissected. *Let the Machines Learn* [online]. 2018-11-26 [cit. 2021-06-15]. Dostupné z: https://yashuseth.blog/2018/11/26/hyper-parameter-tuning-best-practices-learning-rate-batch-size-momentum-weight-decay/

[29] MAY, Madison. An Overview of Python Deep Learning Frameworks. *KDnuggets* [online]. [cit. 2021-07-02]. Dostupné z: https://www.kdnuggets.com/2017/02/python-deep-learning-frameworks-overview.html

[30] TensorFlow. *Tutorials Point* [online]. [cit. 2021-07-03]. Dostupné z: https://www.tutorialspoint.com/tensorflow/index.htm