Detekce zavřených očí s využitím strojového učení

Bc. Adam Holomek

Diplomová práce
2019

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2018/2019

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Adam Holomek**
Osobní číslo:	**A17668**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Detekce zavřených očí s využitím strojového učení**
Téma anglicky:	**Closed Eye Detection using Machine Learning**

Zásady pro vypracování:

1. **Proveďte rešerši metod detekce objektů se zaměřením na detekci očí.**
2. **Seznamte se s deep learning systémy.**
3. **Vytvořte aplikaci s využitím deep learning systémů pro detekci zavřených a otevřených očí ve videu v reálném čase.**
4. **Ověřte funkčnost, přesnost a rychlost navrženého řešení.**
5. **Srovnejte navržená řešení s dostupnými metodami.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **ROSEBROCK, Adrian. Deep learning for computer vision with Python. Místo vydání není známé: PyImageSearch, 2017, 3 svazky (330, 280, 319 stran). ISBN 978-1-986538138.**
2. **GULLI, Antonio a Sujit PAL. Deep learning with Keras: implement neural networks with Keras on Theano and TensorFlow. Birmingham: Packt Publishing, 2017, iv, 303. ISBN 978-1-78712-842-2.**
3. **GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. Deep learning. Cambridge, Massachussetts: The MIT Press, [2016], xxii, 775. Adaptive computation and machine learning. ISBN 978-0-262-03561-3.**
4. **TAFRESHI, M. and FOTOUHI, A. M. A fast and accurate algorithm to distinguish between open and closed eye by efficient combining of texture and appearance features. 2014 22nd Iranian Conference on Electrical Engineering (ICEE). Tehran. 2014. pp. 1013-1017.doi: 10.1109/IranianCEE.2014.6999684**
5. **MAHDI, R. and KLETTE, R. 3D Cascade of Classifiers for Open and Closed Eye Detection in Driver Distraction Monitoring. In Computer Analysis of Images and Patterns. p. 171--179. Springer Berlin Heidelberg. 2011.**
6. **ROSEBROCK, A.: Eye blink detection with OpenCV, Python, and dlib. dostupné z https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/**
7. **SONG, F., TAN, X., LIU, X., & CHEN, S. (2014). Eyes closeness detection from still images with multi-scale histograms of principal oriented gradients. Pattern Recognition, 47(9), 2825-2838.**

Vedoucí diplomové práce: **doc. Ing. Zuzana Komínková Oplatková, Ph.D.**
Ústav informatiky a umělé inteligence

Konzultant: **Ing. Alžběta Turečková**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **3. prosince 2018**

Termín odevzdání diplomové práce: **15. května 2019**

Ve Zlíně dne 7. prosince 2018

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ Fakulta aplikované informatiky

L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
garant oboru

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomové práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

Adam Holomek, v.r.

. . . . . .

podpis autora

ABSTRAKT

Cílem této diplomové práce je navrhnout řešení pro detekci zavřených očí ve videu v reálném čase. V práci budou teoreticky popsány principy neuronových sítí a Spatial Transformer Network. Následovat bude praktické seznámení s tvorbou vybraných modelů konvolučních neuronových sítí za pomoci frameworků Keras a Tensorflow. Tyto vytvořené modely budou poté použity při samotné detekci zavřených očí. Bude ověřena funkčnost a přesnost navrženého řešení. Součástí práce bude i rešerše metod pro detekci objektů, se zaměřením na detekci očí. Při vývoji samotné aplikace byl použit scriptovací jazyk Python a knihovna počítačového vidění OpenCV.

Klíčová slova: Deep learning, Neuronové sítě, Konvoluční neuronové sítě, Spatial Transformer Network, Detekce oči, Detekce mrkání, Python, OpenCV, Keras, TensorFlow

ABSTRACT

The aim of this thesis is to design a solution for the detection of closed eyes in video in real time. The principles of neural networks and Spatial Transformer Network will be theoretically described. There will be a practical introduction to the creation of selected models of convolutional neural networks using the frameworks Keras and Tensorflow. These models will then be used to detect closed eyes. The functionality and accuracy of the proposed solution will be verified. Part of the work will be a research of methods for object detection, focusing on eye detection. The Python scripting language and OpenCV computer vision library were used to develop the application itself.

Keywords: Deep learning, Neural networks, Convolution neural networks, Spatial Transformer Network, Eye detection, Blink detection, Python, OpenCV, Keras, TensorFlow

OBSAH

ÚVOD

Deep Learning je technika strojového učení, při které se počítačový model učí dělat to, co je pro člověka přirozené, a tím je učit se příkladem. Deep Learning je jednou z klíčových technologií autonomního řízení. Tato technologie umožňuje rozpoznání dopravních značek či rozlišovat chodce od lampy. Také je klíčem pro hlasové ovládání telefonů, tabletů, televizí a dalších.

Při Deep learning se počítačový model učí provádět klasifikační úkoly přímo z obrázku, hlasu nebo textu. Modely Deep learning mohou v některých případech dosahovat přesnosti srovnatelné s výkonem člověka. Tyto modely jsou trénovány při použití rozsáhlého množství dat a architektur neuronových sítí. Například vývoj autonomních vozidel vyžaduje milióny obrázků a hodin videa. Trénování modelů Deep learningu může trvat několik dnů až týdnů. Použití GPU může podstatně urychlit tento proces, a to řádově z dnů na hodiny.

Ospalost je jednou z hlavních příčin vzniku dopravních nehod. Autonomní vozidla představují velký potenciál v eliminaci dopravních nehod způsobené ospalostí. Ačkoli autonomní vozy nabízejí slibné řešení, nelze se pouze spoléhat na budoucnost vozidel bez řidiče. V současné době je široce rozšířena technologie monitorující chování řidiče, jenž zahrnuje identifikaci ospalosti. Na základě sledování očí lze detekovat únavu řidiče v dostatečném předstihu.

Cílem této práce je seznámení se systémy Deep Learningu, které povedou k tvorbě aplikace pro detekci zavřených očí ve videu v reálném čase. Při vývoji bude kladen důraz na rychlost, přesnost a funkčnost navržené aplikace. Teoretická část je zaměřena především na popis neuronových sítí, konvolučních sítí a mechanismu Spatial Transformer Network. V praktické části budou demonstrovány a aplikovány znalosti získané z teoretické části. Navržené řešení bude testováno a zhodnoceno.

I. TEORETICKÁ ČÁST

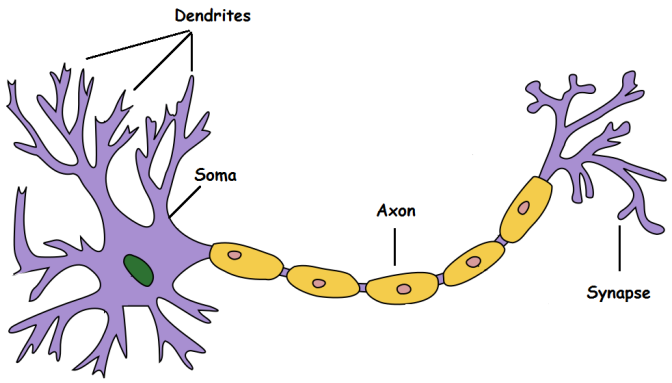
1 Neuronové sítě

Neuronové sítě jsou jádrem Deep learningu, oboru, který má praktické využití v mnoha různých oblastech. Dnes jsou neuronové sítě používány pro klasifikaci obrazu, rozpoznávání řeči, detekci objektů atd.

Koncept umělé neuronové sítě byl v minulosti inspirován a vytvářen na základě biologického nervového systému. Nervový systém můžeme rozdělit na centrální nervový systém (CNS), složen z mozku a míchy, a periferní nervový systém. Hlavním úkolem CNS je řízení organismu, zpracovávat signály, které se do CNS šíří ze smyslových receptorů a na základě vyhodnocení vstupních signálu vydat pokyn, který se opět šíří ve formě signálu směrem k efektorům (např. svalům). Umělé neuronové sítě byly na svém začátku široce inspirovány přiblíženou biologickou analogií, nicméně po ustavení základních technických konceptů probíhal jejich další rozvoj už zcela samostatně, bez jakékoli další přímé vazby na původní motivaci. Matematická reprezentace a modely umělých neuronových sítí jsou tedy oproti známé biologické realitě pouze velmi zjednodušenými modely, navíc často poměrně vzdálenými [1], [2].

1.1 Biologický neuron

Neuron neboli nervová buňka je základním kamenem CNS. Lidský mozek je složen v průměru z 10^{11} neuronů, kde každý z těchto neuronů muže mít až 5000 spojů s ostatními neurony. S rostoucím věkem člověka počet neuronu klesá. Neuron je schopen přijímat různé formy určitých signálů a odpovědět na ně speciálními signály, vést je a vytvářet funkční kontakty (synapse) s ostatními neurony, efektory nebo receptory. Strukturálně může být neuron rozdělen na tři hlavní části: buněčné tělo (soma), dendrity a axon [1], [2]. Schematicky je model biologického neuronu zobrazen na obrázku 1.1.



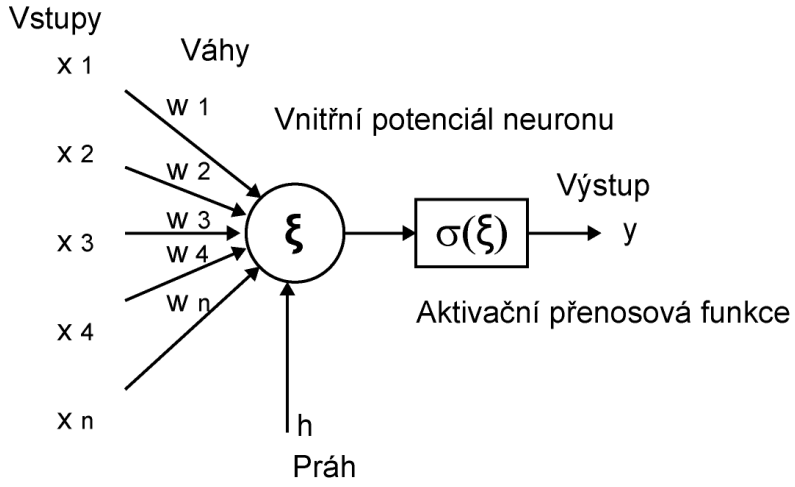
Obr. 1.1 Model biologického neuronu [2].

Tělo buňky obsahuje organely neuronu a také v něm vznikají dendrity. Dendrity jsou velmi tenká a široce rozvětvená vlákna v různých směrech, jenž představují vstup do

těla neuronu, kterým přichází signály z okolních neuronů. Na každý neuron je přidělen právě jeden axon. Jedná se o jediné dlouhé vlákno, které šíří výstupní signál z těla neuronu a které je pomocí synapsí spojen s dendrity neuronů ostatních [1], [2], [3].

1.2 Umělý neuron

V roce 1943 navrhli neurolog Warren S. McCullosh a matematik Walter Pitts první konceptuální model umělé neuronové sítě [3], který představoval teoretický model části nervové soustavy (biologického neuronu). Ve své práci popsali koncept neuronu, jedné buňky v síti buněk, která je schopná přijmout vstupy, zpracovávat je a vygenerovat výstup, přičemž váha všech vstupů je stejná a výstup je v binární formě. Schematicky je model umělého neuron zobrazen na obrázku 1.2.



Obr. 1.2 Model umělého neuronu [3].

Obecně lze o McCullosh-Pitts (MCP) neuronu, že je tvořen několika vstupy $x_1, x_2, ..., x_n$ a jedním výstupem y. Vážená suma vstupních hodnot představuje vnitřní potenciál neuronu ζ, který lze vypočítat ze vztahu:

$$\zeta = \sum_{i=1}^{N} x_i w_i - h \tag{1.1}$$

Každá vstupní hodnota je tedy násobena příslušnou vahou, tato hodnota je v těle neuronu sečtena pro všechny vstupní hodnoty a od výsledné hodnoty je odečten práh. Takto získaná hodnota vnitřního potenciálu se stává následně argumentem obecně nelineární aktivační (přenosové) funkce.

Z praktických důvodů se práh zpravidla modeluje jako jedna z vah tak, že vstupní hodnoty i jejich váhy jsou rozšířeny o nultou pozici. Vstup na nulté pozici je vždy uvažován za roven 1 a nultá váha je nastavena na hodnotu $-h$. Výstup neuronu můžeme

tedy matematicky vyjádřit jako

$$y = \sigma(\zeta) = \sigma(\sum_{i=0}^{N} x_i w_i) \tag{1.2}$$

kde $x_0 = 1$, $w_0 = -h$ a $\sigma(\zeta)$ představuje aktivační (přenosovou)funkci, v tomto případě se jedná o funkci skokovou.

Jestliže je vnitřní potenciál neuronu nižší než práh h, bude na výstupu neuronu 0, pokud vyšší, bude na výstupu 1. Tuto funkci je možné matematicky zapsat jako

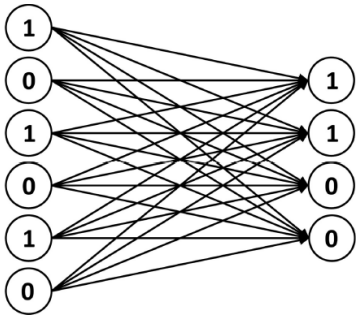
$$y = \begin{cases} 1 & \text{pokud } \zeta \geq 0 \\ 0 & \text{pokud } \zeta < 0 \end{cases} \tag{1.3}$$

1.2.1 Hebbův zákon učení

V roce 1949 definoval D. Hebb na základě biologické analogie a studia podmíněných reflexů pravidlo, které vychází právě z představy posilování vazeb mezi neurony. Hebbův zákon učení je považován za základ všech učících algoritmů, přičemž se jedná o nejstarší pravidlo učení. Toto pravidlo předpokládá, že pokud se dva sousední neurony aktivují nebo deaktivují současně, vazba mezi nimi se posiluje, zatímco při nesouhlasné aktivaci oslabuje. Pokud je aktivní jen jeden neuron z dvojice, nedochází k žádné modifikaci vazby. V neuronových sítích můžeme toto pravidlo definovat jako

$$w_{ij}^{new} = w_{ij}^{old} + \eta y_i y_j \tag{1.4}$$

kde w_{ij}^{old} je váha před modifikaci, w_{ij}^{new} je váha po modifikaci, η je učící koeficient (learning rate) a y_i, y_j výstupy neuronu. Pokud jsou oba neurony y_i a y_j současně aktivní, to odpovídá hodnotě 1 pro binární aktivační funkci, budou váhy mezi nimi v příštím kroku upraveny tak, že k nim bude přičten učící koeficient. Jeho vhodným nastavením lze tedy ovlivnit, o kolik se budou měnit hodnoty vah v jednom kroku. Pokud bude alespoň jeden z neuronů neaktivní (hodnota jeho výstupu bude 0), hodnota vah mezi nimi se nezmění. Na obrázku 1.3. je znázorněna ukázka aplikace Hebbova zákona učení. Jedná se o příklad ukázkového podmíněného reflexu, který je znázorněn pomocí jednovrstvé sítě [10].



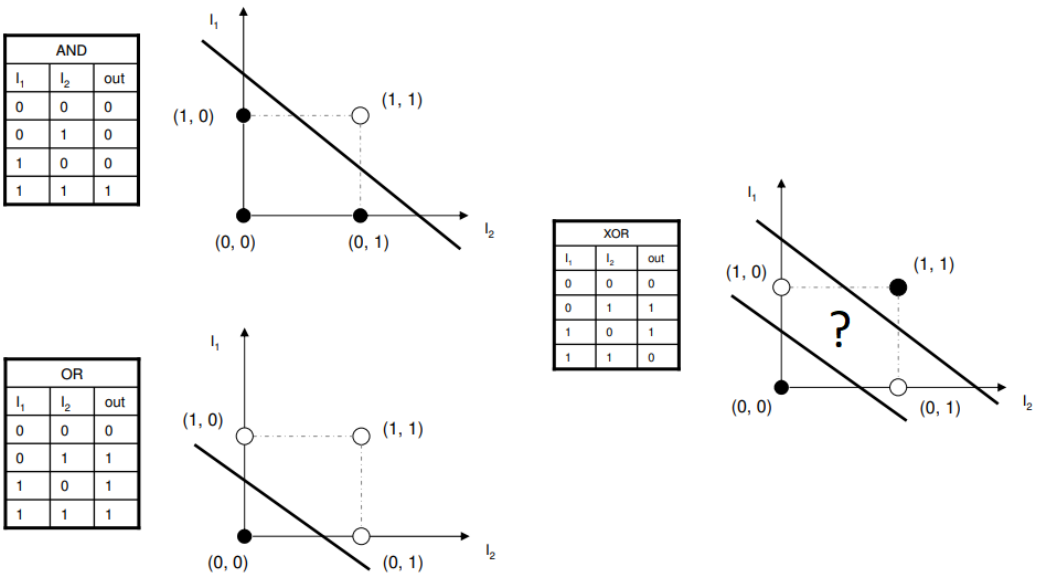
Obr. 1.3 Jednovrstvá neuronová síť - znázornění asociací [10].

1.3 Perceptron

V roce 1957 představil americký psycholog Frank Rosenblatt umělou neuronovou síť zvanou perceptron. Perceptron můžeme považovat za jediný neuron. Nechal se přitom inspirovat dřívější prací W. McCulloche a W.Pittse. Jedná se o obecnější výpočetní model MCP neuronu. Tato síť měla sloužit k rozpoznávání znaků promítaných na plátno. Modelovala percepci, odtud tedy pochází pojem perceptron [4].

Jediným perceptronem lze řešit pouze lineárně separabilní data, pomocí základních logických funkcí (NOT, AND, OR) rozděluje prostor výsledků na dvě části pomocí rovné plochy. Funkce XOR není lineárně separabilní, tudíž je nemožné ji řešit jediným perceptronem. XOR znamená, že výsledná hodnota bude 1, pokud A nebo B bude 1, nesmí však A a B nabývat 1 současně [7].

Pro lepší znázornění problému je vhodné tyto body zanést do dvourozměrného prostoru, přičemž problém lineární separace spočívá v úkolu nalézt přímku, která oddělí body s různou funkční hodnotou. Tento problém je znázorněn na obrázku 1.4 [7].



Obr. 1.4 Logické funkce - problém XOR [7].

Z obrázku je patrné, že jednotlivé množiny funkce XOR nejsou lineárně separabilní, a tedy pro klasifikaci nemůžeme použít klasický jednovrstvý perceptron.

Zřejmým řešením XOR-u by bylo poskládat více perceptronů dohromady. To však sebou neslo několik problémů. F. Rosenblatt představil spolu s perceptronem také pravidlo učení. Jednalo se o algoritmus, který jednotlivým vstupům automaticky dopočítal korektní váhy. Toto pravidlo však nebylo zevšeobecněné pro vícevrstvou síť perceptronů, což učinilo učící proces těchto sítí mnohem složitější.

1.3.1 Aktivační funkce

Aktivační funkce jsou matematické rovnice, které určují výstup neuronové sítě. Funkce je připojena ke každému neuronu v síti neuronu v rozsahu mezi 1 a 0 nebo mezi -1 a 1.

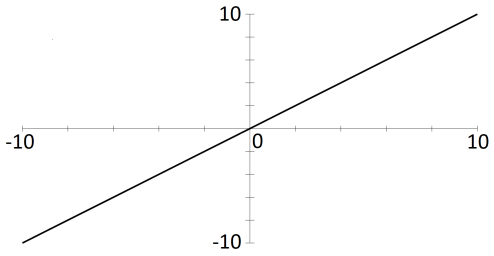
Nyní si uvedeme několik nejčastěji používaných aktivačních funkcí v dopředných neuronových sítích [5].

1. **Lineární funkce**
   Jedná se o nejjednodušší aktivační funkci. Výstup nabývá stejných hodnot jako vstup, funkce je definována v rozsahu $(-\infty, \infty)$.

   Matematicky je funkce definována následujícím vzorcem:

   $$f(x) = x \tag{1.5}$$

   Průběh této funkce je znázorněn na obrázku 1.5.

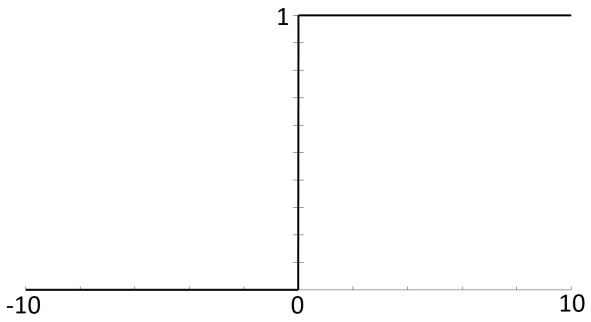


Obr. 1.5 Lineární funkce.

2. **Skoková funkce**
   Skoková (binární) funkce je v neuronových sítích velmi využívanou funkcí. Tento typ funkce je užitečný pro binární schémata. Transformuje záporné hodnoty x na 0 a nezáporné na 1. Výstup je tedy binární povahy [5]. Matematicky je funkce definována následujícím vzorcem:

   $$f(x) = 0 \text{ pokud } x < 0, 1 \text{ pokud } x \geq 0 \tag{1.6}$$

Průběh této funkce je znázorněn na obrázku 1.6.

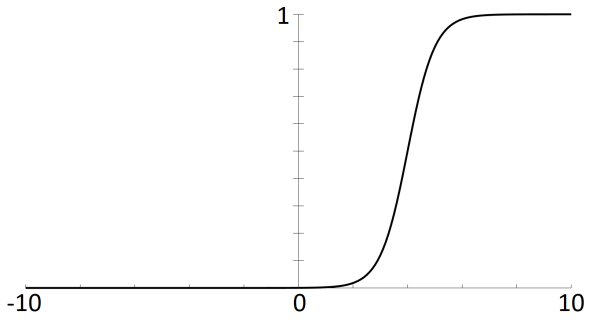


Obr. 1.6 Skoková funkce.

3. **Logistická sigmoida**
   Funkce sigmoid je matematická funkce, která vytváří sigmoidní křivku charakteristickou pro její tvar S. Jedná se o nejstarší a často používanou aktivační funkci. Vstup má libovolnou hodnotu mezi 0 a 1 a vytvoří logistický charakter [5]. Tato funkce odkazuje na speciální případ logistické funkce definovaný následujícím vzorcem:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1.7}$$

   Průběh této funkce je znázorněn na obrázku 1.7.



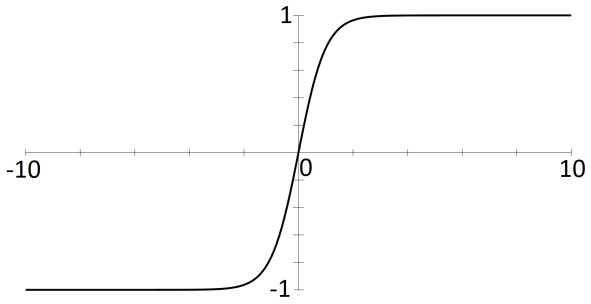
Obr. 1.7 Logistická sigmoida.

4. **Hyperbolický tangens**
   Další velmi populární a široce používanou aktivační funkcí je funkce tanh. Pokud se podíváte na následující obrázek, můžete si všimnout, že vypadá velmi podobně jako sigmoid. Jedná se o nelineární funkci, definovanou v rozsahu hodnot (-1, 1) [5].

Matematicky je funkce definována následujícím vzorcem:

$$f(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{1.8}$$

Průběh této funkce je znázorněn na obrázku 1.8.

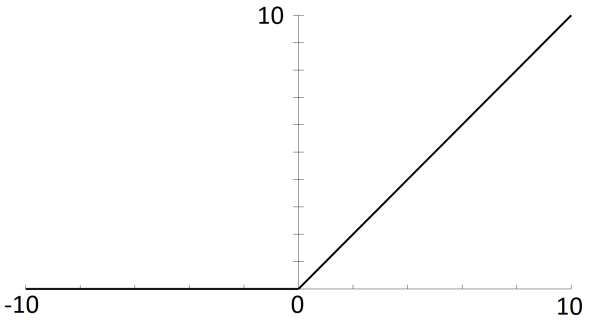


Obr. 1.8 Hyperbolický tangens.

5. **ReLU (Rectified Linear Unit)**
   ReLU je nejčastěji používanou aktivační funkcí. Nejčastěji nachází uplatnění v počítačovém vidění a rozpoznávání řeči pomocí neuronových sítí. Transformuje záporné hodnoty x na 0 a nezáporné nechá identické jako x. Rozsáh výstupu je tedy 0 až ∞ [5]. Matematicky je funkce definována následujícím vzorcem:

$$f(x) = 0 \text{ pokud } x < 0, x \text{ pokud } x \geq 0 \tag{1.9}$$

Průběh této funkce je znázorněn na obrázku 1.9.



Obr. 1.9 ReLU funkce.

6. **Softmax**
   Typicky se Softmax používá pouze pro výstupní vrstvu, pro neuronové sítě, které potřebují klasifikovat vstupy do více kategorií. Funkce Softmax zpracuje výstupy každé jednotky mezi 0 a 1, stejně jako funkce sigmoid. Rozděluje také každý

výstup tak, že celkový součet výstupů je roven 1. Průběh této funkce je znázorněn na obrázku 1.10 [6].

Obr. 1.10 Softmax [6].

Výstup funkce Softmax je ekvivalentní kategorickému rozdělení pravděpodobnosti, tedy udává pravděpodobnost, že některá z tříd je pravdivá. Matematicky je funkce definována následujícím vzorcem:

$$f(x) = \frac{e_j^z}{\sum_{k=1}^{K} e_k^z} \tag{1.10}$$

1.4 Vícevrstvá síť

Vícevrstvá síť je síť neuronů, která je tvořena minimálně třemi vrstvami neuronů: obsahuje vstupní vrstvu, jednu nebo více skrytých vrstev a vrstvu výstupní. Je-li použita jedna skrytá vrstva spolu se vstupní a výstupní, hovoříme o třívrstvé neuronové síti. Pokud obsahuje síť více skrytých vrstev, hovoříme o hlubokých neuronových sítích. Na přesný počet neuronů (perceptronů) ve skryté vrstvě neexistuje pravidlo. Obsahuje-li skrytá vrstva až příliš neuronů, dochází k jevu zvanému přeučení - síť se až příliš přesně naučí množinu testovacích dat a to včetně náhodných chyb nebo šumu. Naopak při malém počtu neuronů nemá síť dostatečnou kapacitu k vyřešení problému [9].

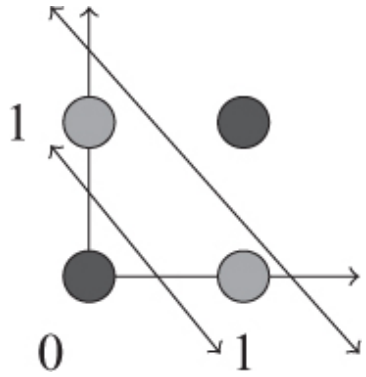
Při určovaní počtu neuronů postupujeme experimentálně a to tak, že začneme menším počtem. Funkcí, jenž určí výpočet chybovosti je dále určeno, jak přesně je síť naučená a podle toho zvyšujeme nebo snižujeme počet neuronů do doby, kdy chybovost klesne pod přijatelnou mez [9].

1.4.1 Vícevrstvý perceptron

Vícevrstvý perceptron (MLP) je nejčastěji používaným typem neuronové sítě. Jak ze samotného názvu napovídá, základním stavebním kamenem této sítě je jednovrstvý perceptron. Jedná se o dopřednou neuronovou síť, během normálního provozu není mezi vrstvami žádná zpětná vazba [8], [32].

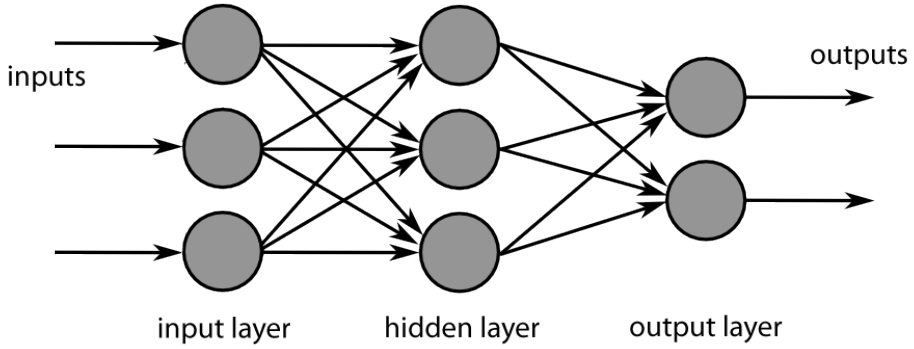
Učení sítě je v tomto případě prováděno prostřednictvím algoritmu Backpropagation. Neurony pracují s nelineárními aktivačními funkcemi, na rozdíl o jednovrstvého per-

ceptronu umí řešit nelineárně separabilní data (obrázek 1.11). Díky správnému nastavení hodnot vah může poskytnout potřebné oddělení pro přesné třídění vstupů XOR [7], [8].



Obr. 1.11 Nelineární separace.

MLP se skládá z alespoň 3 vrstev, vstupní, výstupní a jedné nebo více skrytých vrstev [32]. Architekturu MLP demonstruje obrázek 1.12.



Obr. 1.12 Vícevrstvý perceptron [33].

Vstupy každého neuronu jedné vrstvy jsou spojeny s výstupy všech neuronů vrstvy předchozí. Neexistují žádné vztahy mezi vzdálenějšími vrstvami nebo mezi neurony v rámci jedné vrstvy. Každý neuron má tedy právě tolik vstupů, kolik je neuronů v předchozí vrstvě. Vstupní vrstva sítě slouží pouze k distribuci vstupních hodnot [8].

Při volbě aktivační funkce u vícevrstvých perceptronů není vhodné použití funkce skokové ani lineární. Důvod je takový, že jedna vrstva lineárních neuronů provádí lineární zobrazení a skládáním více lineárních zobrazení dostaneme opět jen lineární zobrazení. Správnou volbou je použití perceptronů s nelineární aktivační funkcí. Taková to funkce transformuje součet impulsů do intervalu <0,1>, a to tak, že u vstupních hodnot v blízkosti nuly prudce roste, zatímco u vysokých a nízkých hodnot se mění jen nepatrně (tato vlastnost je převzata od biologických neuronů). Tyto požadavky splňují již zmíněné nelineární aktivační funkce, z nichž nejčastěji se používá již dříve zmíněná Logistická sigmoida [8].

1.5 Učení neuronové sítě

Cílem učení neuronové sítě je nastavení váh modelu neuronové sítě tak, aby docházelo k minimalizaci odchylky (chyby) mezi skutečným (aktuálním) a požadovaným výstupem. V biologických sítích jsou zkušenosti uloženy v dendritech. V umělých neuronových sítích jsou tyto zkušenosti uloženy v jejich matematickém ekvivalentu a tím je váha. Učení neuronové sítě lze obecně rozdělit na učení s učitelem a učení bez učitele [9], [10].

U učení s učitelem je obdobně jako u biologických sítí využita zpětná vazba. Na základě aktuálního nastavení je zjištěna výstupní hodnota, která je porovnána s požadovanou hodnotou a na základě rozdílu těchto hodnot určíme chybu. Poté spočítáme nutnou korekci a upravíme hodnoty vah či prahů tak, aby došlo ke snížení hodnoty chyby. Tento proces je opakován až do dosažení stanovené minimální chyby [9], [10].

U učení bez učitele nedochází k vyhodnocování výstupu, takže během učení není znám výstup. Síť dostává na vstup sadu vzorů, které si sama třídí. Vzory jsou tříděny buď do skupin a reaguje na typického zástupce, nebo si přizpůsobí topologii vlastnostem vstupu [9], [10].

1.5.1 Delta pravidlo

Delta pravidlo, nazývané taky jako metoda LMS (Least Mean Square), bylo poprvé představeno B. Widrowem a M. Hoffem v síti ADALINE (Adaptive Linear Element) [1], [10], [11]. Jedná se o učící pravidlo pro aktualizaci vah vstupů v jednovrstvé neuronové síti. Snahou pravidla je najít ideální váhy pomocí minimalizace celkové chybové funkce δ. Tato chyba je vypočtena jako rozdíl mezi požadovaným a skutečným výstupem dle rovnice:

$$\delta_j = t_j - y_j \tag{1.11}$$

kde t_j je cílový (požadovaný) výstup a y_j je skutečný výstup. Dosazením této rovnice dostaneme finální podobu delta pravidla

$$w_{ij}^{new} = w_{ij}^{old} \eta (t_j - y_j) x_{ij} \tag{1.12}$$

kde η je učící koeficient v rozmezí (0,1). Delta pravidlo je ideální pro neurony s lineární aktivační přenosovou funkcí, v případě modifikace je však použitelné i pro neurony s nelineární aktivační přenosovou funkcí.

1.5.2 Backpropagation

Algoritmus Backpropagation byl původně představen v roce 1970, jeho význam byl však doceněn až v roce 1986 z článku od Davida Rumelharta, Geoffrey Hintona a Ronalda Williamse. Backpropagation je metoda zpětné propagace chyby a je příkladem zobecnění delta pravidla, používá se pro výpočet chybové funkce podle jednotlivých vah ve vícevrstvých sítích. V současné době se jedná o nejčastěji používaný algoritmus k učení neuronových sítí. Aby bylo možné algoritmus používat i pro nelineární klasifikační úlohy, můžeme použít aktivační funkce jako například logistickou sigmoidu nebo hyperbolický tangens. Výstupní řešení se porovná s očekávaným a tím se zjistí, o kolik se neuronová síť zmýlila. Zpětně se pak na základě tohoto výsledku vypočítá, o kolik se mají váhy neuronů změnit, aby se odchylka od správného řešení co nejvíce snížila [4], [12].

Chybová funkce může být popsána jako průměr ztrát v jednotlivých příkladech:

$$E = \frac{1}{2}\sum_{j=1}^{n}(t_j - y_j)^2 \tag{1.13}$$

Odvození algoritmu Backpropagation začíná použitím pravidla řetězce na částečnou derivaci chybové funkce

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{\partial E}{\partial a_j^k}\frac{\partial a_j^k}{\partial w_{ij}^k} \tag{1.14}$$

kde a_j^k je aktivace uzlu j ve vrstvě k před jeho předáním do nelineární aktivační funkce (v tomto případě funkce sigmoid) pro generování výstupu. Tato dekompozice parciálního derivátu v zásadě říká, že změna chybové funkce E v důsledku váhy w_{ij}^k je výsledkem změny chybové funkce E v důsledku aktivace a_j^k krát změna aktivace a_j^k na váhu w_{ij}^k [12].

První termín δ_j^k se obvykle nazývá chybou a lze vyjádřit vztahem

$$\delta_j^k \equiv \frac{\partial E}{\partial a_j^k} \tag{1.15}$$

Druhý termín lze vypočítat z rovnice pro a_j^k uvedené výše

$$\frac{\partial a_j^k}{\partial w_{ij}^k} = \frac{\partial}{\partial w_{ij}^k}(\sum_{l=0}^{r_k-1} w_{lj}^k o_l^{k-1}) = o_i^{k-1} \tag{1.16}$$

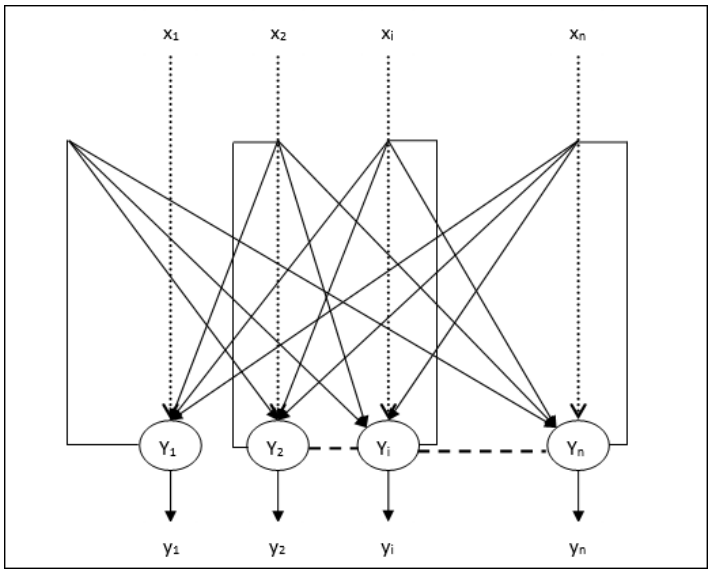
Parciální derivace chybové funkce E vzhledem k váze w_{ij}^k je tedy

$$\frac{\partial E}{\partial w_{ij}^k} = \delta_j^k o_i^{k-1} \tag{1.17}$$

Parciální derivace váhy je tedy výsledkem chyby δ_j^k v uzlu j vrstvy k a výstupu o_i^{k-1} uzlu i vrstvy $k-1$.

1.6 Hopfieldova síť

Poprvé byla tato síť představena v roce 1982 Johnem Hopfieldem. Hopfieldova síť je příkladem sítě se zpětnou vazbou (tzv. Rekurentní síť). Síť je tvořená z n neuronů, kde jsou výstupy neuronů spojeny se vstupem každého neuronu pomocí příslušných vah. Následující obrázek 1.13 demonstruje chování Hopfieldovi sítě [14], [15], [16].



Obr. 1.13 Ukázka Hopfieldovi sítě [16].

Výstup z této sítě je tvořen hodnotami -1 nebo 1. Jestliže je vnitřní potenciál neuronu nižší než práh h, bude na výstupu neuronu -1, v opačném případě bude na výstupu 1 [13]. Tuto funkci je možné matematicky zapsat jako

$$y = \begin{cases} 1 & \text{pokud } \zeta \geq 0 \\ -1 & \text{pokud } \zeta < 0 \end{cases} \tag{1.18}$$

Mezi každou dvojicí neuronů i a j existuje v Hopfieldovi síti spojení, které je označováno váhou w_{ij}. Jedná se o symetrické spojení, to znamená, že mezi neurony i a j je stejná váha jako mezi neurony j a i, matematicky vyjádřeno:

$$w_{ii} = 0, \forall i$$
$$w_{ij} = w_{ji}, \forall i, j$$

Hopfieldova síť implementuje Hebbův zákon učení a to následujícím způsobem:

$$w_(ij) = w_(ji) = -\frac{1}{N}\sum_{k=1}^{p} x_i^{(k)} x_j^{(k)} \tag{1.19}$$

kde k značí jednotlivé kroky, tedy hodnota odpovídá celkovému počtu kroků adaptace. Tato adaptace probíhá podle Hebbova zákona v jednotlivých krocích, během nichž dochází k postupnému upravování synaptických vah na základě předkládaných vzorů.

Hopfieldova síť nově zavádí pojem energetická funkce. Tato funkce matematicky vyjadřuje jak velké chyby se síť dopouští v každém jejím okamžiku [13].

Energetická funkce je vyjádřena vztahem:

$$E(y) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} y_i y_j \tag{1.20}$$

kde N udává celkový počet neuronů sítě, w_{ij} váhu mezi neurony i, j a $y \in -1, 1$. Během samotného výpočtu by se měla energie E snižovat, dokud nedosáhne svého lokálního minima.

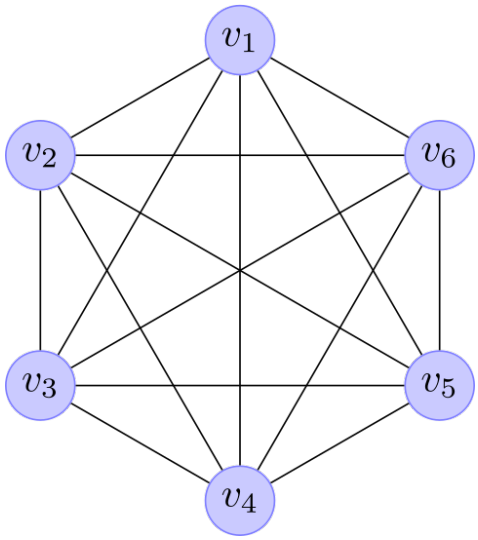
1.6.1 Boltzmannův stroj

Boltzmannův stroj je inspirován binární Hopfieldovou sítí a je běžně známý jako jeho stochastická verze. Tento model neuronové sítě vynalezl Geoffrey Hinton a Terry Sejnowski v roce 1985 [34].

Boltzmannův stroj se skládá ze stochastických neuronů, které mají jeden ze dvou možných stavů, buď 1 nebo 0. Změna stavu neuronu na opačný (0 na 1 a 1 na 0) závisí jak na hodnotě aktivační funkce, tak na veličině nazývané teplota sítě $T \geq 0$. Tato teplota se obvykle mění v průběhu práce sítě a je zodpovědná za stochastické chování neuronů a energie sítě tak nemusí na rozdíl od Hopfieldovy sítě v každém kroku sítě jen klesat. Obrázek 1.14 zobrazuje jednoduchou architekturu Boltzmannova stroje [2], [35].

Základem Boltzmannova stroje je tedy stochastický neuron, u kterého známe pravděpodobnost P, se kterou se bude nacházet v daném stavu na základě hodnoty vnitřního potenciálu neuronu. Funkce, jenž vyjadřuje závislost pravděpodobnosti změny stavu bývá sigmoida, která je modifikována právě parametrem T. V závislosti na teplotě tedy může stochastický neuron s jistou pravděpodobností změnit svůj stav bez ohledu na hodnotu vnitřního potenciálu ζ_i [2].

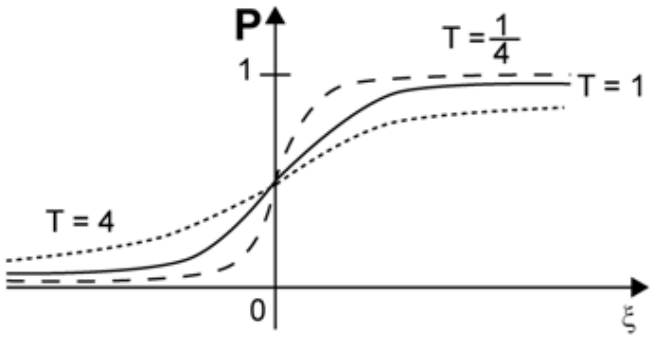
Pravděpodobnost, že síť přijme změnu stavu, je dána následujícím vztahem



Obr. 1.14 Architektura Boltmannova stroje [35].

$$P = \sigma(\zeta_i) = \frac{1}{1 + e^{\frac{-\zeta_i}{1+T}}} \tag{1.21}$$

Pro $T \geq 0$ dostáváme pro pravděpodobnost změny stavu standardní sigmoidální funkci a neuron se tedy s vysokou pravděpodobností bude chovat dle hodnoty svého vnitřního potenciálu ζ_i. Dále vyplývá, že s rostoucí hodnotou parametru T dochází k vyrovnání pravděpodobnosti volby obou stavů bez ohledu na hodnotu vnitřního potenciálu ζ_i a pro $T \to 0$ se neuron chová zcela náhodně, oba stavy jsou stejně pravděpodobné. Na následujícím obrázku 1.15 je znázorněn graf pravděpodobnostní aktivační funkce vykreslený pro různé teploty [2], [10].



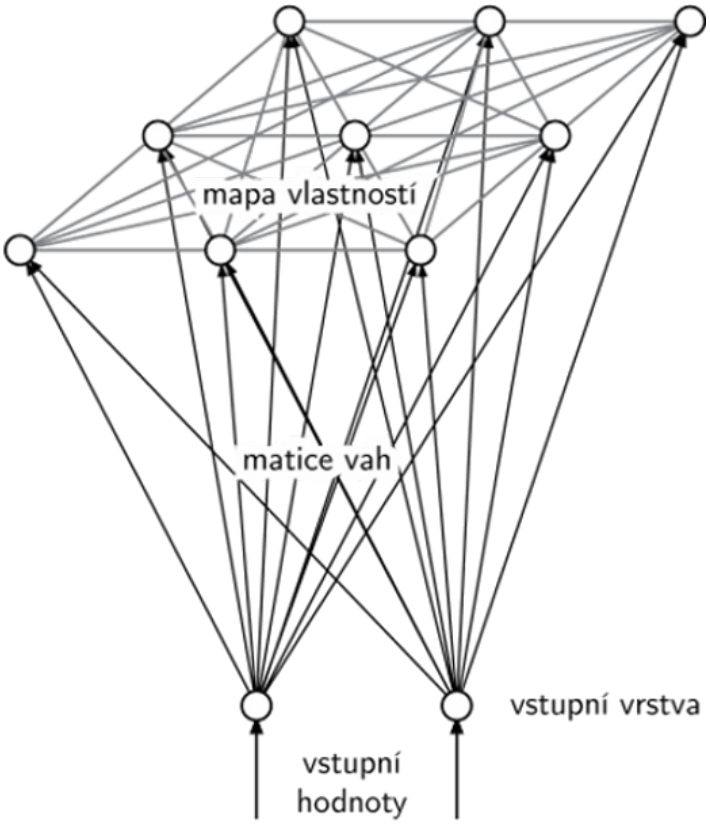
Obr. 1.15 Průběh pravděpodobnostní aktivační funkce [2].

1.7 Kohonenova mapa

Kohonenova mapa nebo taky Kohonenova síť byla poprvé představena v 80. letech 20. století profesorem T. Kohonenem. Patří do skupiny samoorganizujících se neuronových sítí. To znamená, že ke svému učení nepotřebuje přítomnost učitele [9], [14].

Samoorganizující se mapy se liší od jiných umělých neuronových sítí tím, že uplatňují konkurenční učení na rozdíl od učení chybových korekcí (např. algoritmus Backpropagation) a v tom smyslu, že používají funkci sousedství k zachování topologických vlastností vstupního prostoru [17], [18].

Kohonenova síť je tvořena vstupní vrstvou a jedinou vrstvou neuronů, tzv. Kohonenovou (kompetiční) vrstvou, kde každý neuron v této vrstvě je propojen se všemi vstupy, takže má přesnou informaci o každé vstupní hodnotě. Následující obrázek 1.16 znázorňuje strukturu Kohonenovi sítě [18].



Obr. 1.16 Struktura Kohonenovy mapy [9].

Váhy spojení mezi neurony představují souřadnice udávající konkrétní polohu neuronu v prostoru. Pro neurony v Kohonenově vrstvě je typické, že neobsahují práh. Jsou uspořádány do předem daných struktur pomocí laterálních spojů, jenž určují vazby mezi nimi a topologickou mřížku, která může představovat např. kruh nebo čtverec. Výstup neuronů je binární (0 - neaktivní, 1 - aktivní), přičemž právě jeden neuron může být v daný okamžik aktivní [2].

Učení sítě je prováděno na základě Eukleidovské vzdálenosti, pomocí které je vybrán tzv. vítězný neuron.

$$w_{ij}^{new} = \sum_{j=1}^{m}(x_i - w_{ij})^2 \tag{1.22}$$

Za vítězný neuron je považován neuron s nejmenší vzdálenosti, tento neuron se stává dále aktivním. Pomocí laterálních spojů je pak okolo něj vytvořeno tzv. sousedství, jedná se o okolní neurony které se jemu nejvíce podobají. Váhy těchto neuronů se dále modifikují [2]. Aktualizace váhy jak vítězného neuronu, tak i jeho sousedství je vypočítána ze vztahu

$$w_{ij}^{new} = w_{ij}^{old} + \eta_(x_i - w_{ij}^{old}) \tag{1.23}$$

Proces učení je ukončen, jakmile dojde k vyčerpání stanoveného počtu iterací.

2 Konvoluční neuronové sítě

Konvoluční neuronová síť - CNN (Convolutional Neural Network) je třídou umělé neuronové sítě, která používá konvoluční vrstvy k filtraci užitečných informací na vstupu. Operace konvoluce zahrnuje kombinaci vstupních dat (mapování funkcí) s jádrem konvoluce (filtr) pro vytvoření transformované mapy funkcí. Filtry ve vrstvách konvoluce (konvoluční vrstvy) jsou upraveny na základě naučených parametrů pro získání nejužitečnějších informací pro konkrétní úkol. Konvoluční sítě se automaticky přizpůsobují, aby nalezly nejlepší funkci založenou na úkolu [6], [36]. Aplikace konvolučních neuronových sítí zahrnují různé obrazové systémy (rozpoznávání obrazu, klasifikaci obrázků, označování videa, textovou analýzu) a zpracovávání řeči (rozpoznávání řeči, zpracování přirozeného jazyka, textové klasifikace) spolu s nejmodernějšími systémy AI, jako jsou roboti, virtuální asistenti nebo autonomní vozidla [6], [36].

2.1 Architektura

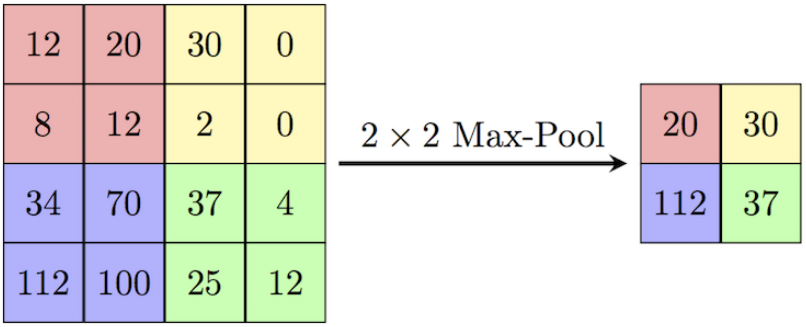
Konvoluční neuronové sítě se skládají z řady konvolučních a škálovacích vrstev, které jsou volitelně následovány plně propojenými vrstvami. Vstupem konvoluční vrstvy je $m \times m \times r$ obrázek, kde m je výška a šířka obrazu a r je počet kanálů, např. obraz RGB $r = 3$. Konvoluční vrstva má filtry (nebo jadra) velikosti $n \times n \times x \times q$, kde n je menší než rozměr obrázku a q může být stejný jako počet kanálů r nebo menší a může se lišit pro každé jádro. Velikost filtrů dává vzniknout lokálně propojené struktuře, která je každá konvolvována s obrazem, aby se vytvořily k mapy funkcí o velikosti $m - n + 1$. Každá mapa je pak škálována typicky s průměrem nebo maximálním sdružením $p \times p$ souvislé oblasti, kde p se pohybuje mezi 2 pro malé obrazy a 5 pro ty větší. Před nebo po škálovací vrstvě se na každou mapu vlastností aplikuje aditivní zkreslení a sigmoidní nelinearita. Po konvolučních vrstvách může být libovolný počet plně propojených vrstev. Hustě spojené vrstvy jsou identické s vrstvami ve standardní vícevrstvé neuronové síti [13], [19], [36].

2.1.1 Vrstvy konvoluční sítě

- Konvoluční vrstva (Convolution Layer) - aplikuje speciální počet filtrů na obrázek. Pro každý subregion obrázku vrstva vypočítá pomocí množiny matematických operací jedinou hodnotu na výstupu. Konvoluční vrstva používá typickou ReLu aktivační funkci, která vkládá do modelu nelinearitu [6], [36].

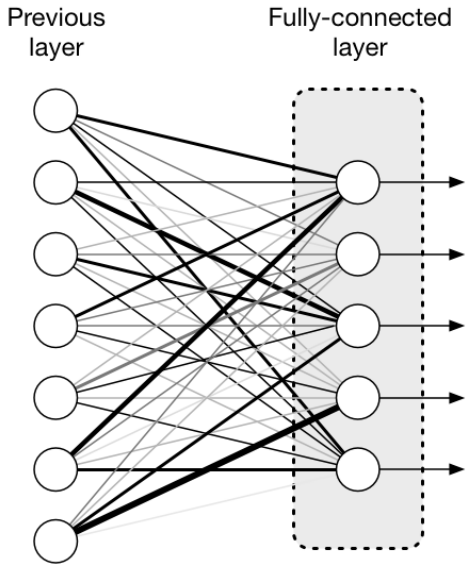
- Škálovací vrstva (Pooling Layer) - snižuje velikost obrázků extrahovaných konvolučními vrstvami, tak aby se zmenšila dimenze mapy a zkrátila čas zpracování. Obvykle se používá max-pooling (obrázek 2.1), který vytváří subregiony mapy s

velikostí 2×2 pixelů, udržuje jejich maximální hodnotu a zbavuje se všech ostatních hodnot. Lze ale použít i mean-pooling nebo jiný [6], [36].



Obr. 2.1 Max-pooling [20].

- Plně propojená vrstva (Full Connected Layer) - provádí klasifikaci na extrahovaných datech z konvolučních a škálovacích vrstev. V plně propojené vrstvě je každý uzel vrstvy propojen s každým uzlem z předchozí vrstvy (obrázek 2.2) [6], [36].



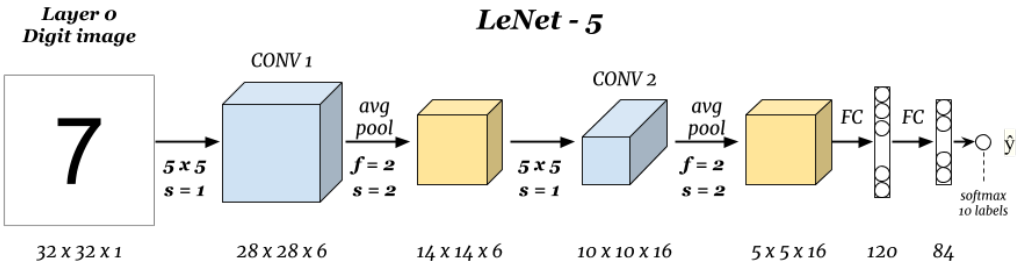
Obr. 2.2 Plně propojená vrstva [21].

Ve výše uvedeném obrázku 2.2 bude matice map funkcí převedena na vektor $(x1, x2, x3, \ldots)$. S plně propojenými vrstvami jsme tyto vlastnosti spojili dohromady a vytvořili model. Nakonec máme aktivační funkci, jako je softmax nebo sigmoid pro klasifikaci výstupů jako kočka, pes, auto, kamion a jiné [6].

2.2 Modely konvoluční sítě

2.2.1 LeNet-5

Model LeNet-5 byl představen Yeannem LeCunem v roce 1998, aby identifikoval ručně psané číslice pro rozpoznávání PSČ v poštovní službě. Tento průkopnický model do značné míry zavedl konvoluční neuronovou síť, jak ji známe dnes. Podle moderních standardů je architektura LeNet velmi jednoduchá a paměťově nenáročná, a tak je ideální volbou pro výuku základů CNN [22], [23].



Obr. 2.3 Architektura LeNet [24].

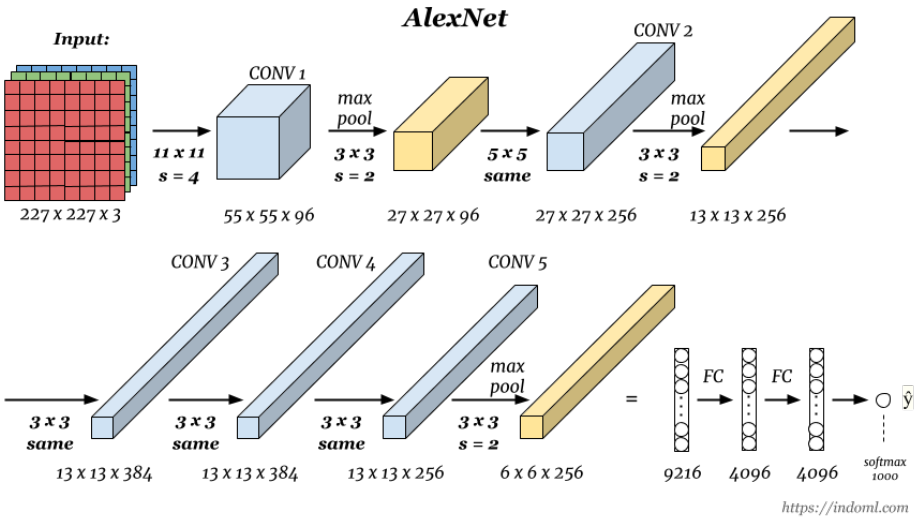
Architektura LeNet (obrázek 2.3) je rozdělena do dvou částí, z bloku konvolučních vrstev a jedné plně propojené vrstvy. Základními jednotkami v konvolučním bloku jsou konvoluční vrstva a následná průměrná vrstva škálovací (mean-pooling v současné době nahrazen max-pooling). Konvoluční vrstva se používá k rozpoznávání prostorových vzorů v obraze, jako jsou čáry a části objektů, průměrná vrstva škálovací se používá ke zmenšení rozměrů. Konvoluční vrstvový blok se tudíž skládá z opakovaných svazků těchto dvou základních jednotek. Každá konvoluční vrstva v tomto bloku používá okno tvaru 5×5, kroku 1 a aktivační funkci Sigmoidu (v současné době použití ReLU). Pro první konvoluční vrstvu je počet výstupních kanálů 6, pro druhou konvoluční vrstvu byl tento počet zvýšen na 16. To proto, že výška a šířka vstupu druhé konvoluční vrstvy je menší než výška kanálu první konvoluční vrstvy. Tvar okna průměrných škálovacích vrstev v bloku je 2×2 a krok činí 2 [22].

2.2.2 AlexNet

Ačkoli LeNet dosáhl dobrých výsledků v brzkých malých souborech dat, jeho výkonost na větších souborech dat nebyla uspokojivá. AlexNet, pojmenován podle Alexe Krizhavského, byl představen v roce 2012. Ačkoli jsou si architektury AlexNet a LeNet velmi podobné, existuje několik významných rozdílu [23], [25].

AlexNet je složen dohromady z osmi vrstev, z toho z pěti konvolučních vrstev, dvou plně propojených skrytých vrstev a jedné plně propojené výstupní vrstvy. Dále došlo u AlexNet k nahrazení původní aktivační funkce sigmoid (LeNet) novou funkcí ReLU.

To významně zlepšilo konvergenci během trénování. Architekturu AlexNet popisuje obrázek 2.4 [23], [25].

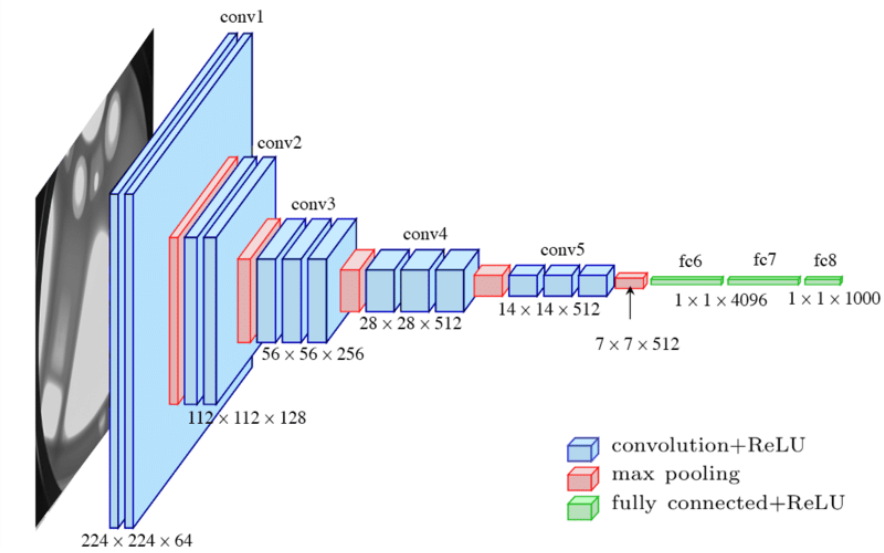


Obr. 2.4 Architektura AlexNet [24].

V první vrstvě AlexNet je tvar okna s konvolucí 11×11. Tvar konvolučního okna ve druhé vrstvě se zmenší na 5x5 následován 3×3. Po první, druhé a páté konvoluční vrstvě přidává síť maximální škalovací vrstvy (max-pooling) s tvarem okna 3×3. Kromě toho má AlexNet zhruba 10× více kánálu než jeho předchůdce LeNet [23].

2.2.3 VGGNet

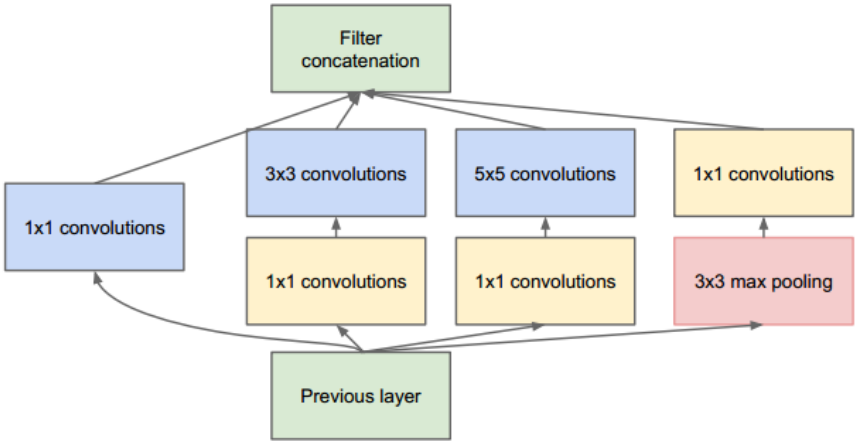
VGG je zkratka pro skupinu Visual Geometry Group z Oxfordské univerzity. Architektura sítě VGG byla poprvé představena Simonyanem a Zissermanem v jejich dokumentu z roku 2014. Tvůrci VGGNet dosud vymodelovali několik různých variant sítě. Například varianta VGG-16 se skládá z 16 vrstev. Narozdíl od AlexNetu, jenž používá velikost okna první konvoluční vrstvy 11×11 a 5×5 pro druhou vrstvu, používá velikost okna 3x3 pro jednu konvoluční vrstvu za druhou. Dále obsahuje 2 plně propojené vrstvy, kde každá vrstva má 4096 uzlů a jedné plně propojené vrstvy s 1000 uzly. Nakonec následuje klasifikátor Softmax. Obrázek 2.5 vykresluje architekturu VGG-16 [23].



Obr. 2.5 Architektura VGG-16 [24].

2.2.4 GoogLeNet/Inception

V roce 2015 navrhl Szegedy strukturu, která překonala všechny ostatní. Hlavní otázkou při vývoji bylo, jaká velikost konvoluce je ideální pro zpracování. Koneckonců máme k dispozici výběr z možností 1×1 nebo 3×3, 5×5 nebo dokonce větší. A není vždy jasné, který z nich je nejlepší. Jak se ukázalo, kombinace všech výše uvedených velikostí konvoluce funguje nejlépe. Základní konvoluční blok v GoogLeNet je označován jako Inception blok [23], [25].



Obr. 2.6 Struktura počátečního bloku GoogLeNet [25].

Jak je vidět na obrázku 2.6, ve vstupním bloku jsou čtyři paralelní cesty. První tři cesty používají konvoluční vrstvy s velikostí okna 1×1, 3×3 a 5×5 k získání informací z různých prostorových velikostí. Střední dvě cesty provedou 1×1 konvoluci na vstupu ke snížení počtu vstupních kanálů, tím se sníží složitost modelu. Čtvrtá cesta používá 3×3 maximální škálovací vrstvu, za kterou následuje 1×1 konvoluční vrstva pro změnu počtu kanálů. Všechny čtyři cesty používají vhodné vyplnění (padding), které dává vstupu a výstupu stejnou výšku a šířku. Nakonec zřetězíme výstup každé cesty kanálu a vložíme ji do další vrstvy. Přizpůsobitelné parametry počátečního bloku jsou počet výstupních kanálů na vrstvu, které mohou být použity pro řízení složitosti modelu [25].

3 Spatial Transformer Network

Spatial Transformer Network (STN) byla poprvé představena Maxem Jaderbergem a jeho spolupracovníky společnosti Google Deepmind v roce 2016. STN umožňuje prostorovou manipulaci s daty v rámci sítě [25].

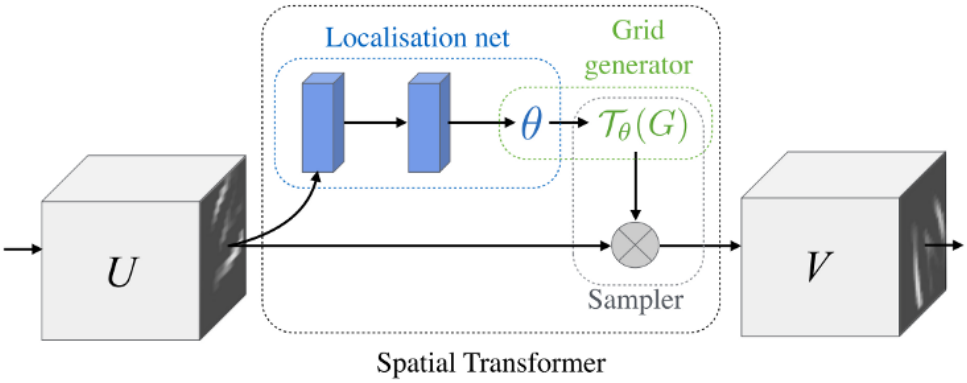
Konvoluční neuronové sítě definují mimořádně silnou skupinu modelů. Tyto modely jsou však stále omezeny na to, být prostorově neměnné vzhledem ke vstupním datům efektivním způsobem výpočetně i parametricky [25], [26].

STN je samostatný modul, který lze kdykoliv vložit do architektury CNN, a v jakémkoliv počtu, což vede k vytvoření sítí prostorových transformátorů. Tento modul je výpočetně velmi rychlý a nezhoršuje tak výrazně samotnou rychlost tréninku. Umístění prostorových transformátorů do CNN umožňuje síti naučit se aktivně transformovat mapy funkcí, aby se minimalizovala celková nákladová funkce sítě během tréninku. Znalosti o tom, jak transformovat každý tréninkový vzorek, jsou komprimovány a ukládány do mezipaměti váhách lokalizační sítě (a také vahách vrstev předcházejících prostorovému transformátoru) během tréninku [26].

V CNN je možné mít více prostorových transformátorů. Umístění více prostorových transformátorů do rostoucí hloubky sítě umožňuje transformaci stále abstraktnějších reprezentací a také dává lokalizačním sítím potenciálně více informativní reprezentace, na nichž se zakládají předpokládané transformační parametry. Lze také použít více prostorových transformátorů paralelně, to může být užitečné, pokud je v mapě objektů několik objektů nebo částí zájmu, které by měly být zaměřeny individuálně. Omezení této architektury v čistě předávací síti je, že počet paralelních prostorových transformátorů omezuje počet objektů, které může síť modelovat [26].

3.1 Prostorová transformace

Mechanismus prostorové transformace je rozdělen do tří částí, znázorněných na obrázku 3.1. V pořadí výpočtu nejprve *lokalizační síť* přebírá mapu vstupních prvků a přes množství skrytých vrstev vydává parametry prostorové transformace, které by měli být aplikovány na mapu funkcí. Pak se předpokládané transformační parametry použijí k vytvoření vzorkovací mřížky, což je sada bodů, ve které by měla být vstupní mapa vzorkována, aby se vytvořil transformovaný výstup. To se provádí *generátorem sítě*. Nakonec mapa funkcí a vzorkovací mřížka jsou převzaty jako vstupy do *vzorkovače*, ve kterém se vytváří výstupní mapa vzorkovaná ze vstupu v bodech mřížky [25], [26].



Obr. 3.1 Spatial Transformer [25].

3.1.1 Lokalizační síť

Lokalizační síť vezme mapu vstupních funkcí $U \in \mathbb{R}^{H \times W \times C}$ se šířkou W, výškou H a C kanály a výstupy θ, parametry transformace $\mathcal{T}_\theta$, které mají být aplikovány na mapu funkcí. Velikost θ se může měnit v závislosti na typu transformace, který je parametrizován, např. pro afinní transformaci θ je 6-rozměrná [26].

Lokalizační síťová funkce může mít jakoukoliv formu, jako je plně propojená síť nebo konvoluční síť, měla by však zahrnovat konečnou regresní vrstvu pro vytvoření transformačních parametrů θ [26].

3.1.2 Parametrizovaná vzorkovací mřížka

Úkolem generátoru sítě je výstup parametrizované vzorkovací mřížky, což je sada bodů, kde by vstupní mapa měla být vzorkována, aby se vytvořil požadovaný transformovaný výstup [26].

Pro provedení zkreslení mapy vstupních funkcí se každý výstupní pixel vypočítá použitím vzorkovacího jádra centrovaným na určitém místě v mapě vstupních prvků. Pixelem označujeme prvek obecné mapy prvků, nemusí se nutně jednat o obraz. Obecně jsou výstupní pixely definovány tak, aby ležely na pravidelné mřížce $G = G_i$ pixelů $G_i = (x_i^t, y_i^t)$, tvořících výstupní mapu funkcí $V \in \mathbb{R}^{H' \times W' \times C}$, kde H' a W' jsou výška a šířka mřížky a C je počet kanálů, který je stejný ve vstupu a výstupu [26].

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = A_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} \tag{3.1}$$

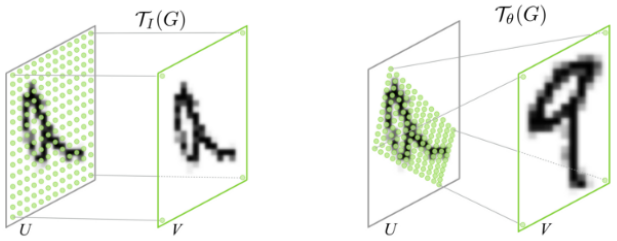
kde (x_i^t, y_i^t) jsou cílové souřadnice pravidelné mřížky na výstupní mapě funkcí, (x_i^s, y_i^s) jsou zdrojové souřadnice ve vstupní mapě funkcí, které definují vzorkovací body a A_θ

je matice afinní transformace.

Transformační matice umožňuje oříznutí, překlad, rotaci, měřítko a zkosení. Chceme-li transformaci omezit, zapíšeme ji ve formě

$$A_\theta = \begin{bmatrix} s & 0 & t_x \\ 0 & s & t_y \end{bmatrix} \tag{3.2}$$

kde parametry s, t_x a t_y představují oříznutí, překlad a isotropní škálování. Vliv transformace na mřížku ve srovnání s identickou transformací je znázorněn na obrázku 3.2.



Obr. 3.2 Příklady použití parametrizované vzorkovací mřížky na obraz U produkující výstup V [26].

3.1.3 Vzorkování s rozlišitelným obrazem

Pro provedení prostorové transformace vstupní mapy funkcí musí vzorkovač odebrat sadu vzorkovacích bodů $\mathcal{T}_\theta(G)$, spolu se vstupní mapou funkcí U a vytvářejí vzorovou výstupní mapu funkcí V [26].

Každé souřadnice (x_i^s, y_i^s) v $\mathcal{T}_\theta(G)$ definuje prostorové umístění ve vstupu, kde je jádro vzorkování použito pro získání hodnoty u konkrétního pixelu ve výstupu V. To může být napsáno jako

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c k(x_i^s - m; \Phi_x) k(y_i^s - n; \Phi_y) \forall i \in [1...H'W'] \forall c \in [1...C] \tag{3.3}$$

kde Φ_x a Φ_y jsou parametry generického vzorkovacího jádra $k()$, které definuje obraz interpolace (např. bilineární), U_{nm}^c je hodnota v místě (n, m) kanálu c vstupu a V_i^c je výstupní hodnota pixelu i v místě (x_i^t, y_i^t) kanálu c. Vzorkování se provádí identicky pro každý kanál vstupu, takže každý kanál je transformován stejným způsobem (to zachovává prostorovou konzistenci mezi kanály) [26].

II. PRAKTICKÁ ČÁST

4 Vývoj

Vývoj byl implementován na notebooku Acer Swift 3 s procesorem Intel®Core™ i3-7100U CPU 2,4GHz, 4GB operační pamětí DDR4 a operačním systému Windows 10. Veškerá implementace byla provedena v jazyce Python. Jedná se o multiplatformní, interpretovaný, objektově orientovaný programovací a skriptovací jazyk. Tvorba konvolučních sítí byla prováděna pomocí frameworku Keras s backend Tensorflow.

Keras je knihovna na úrovni modelu psaná v jazyce Python, která poskytuje stavební bloky na vysoké úrovni pro vývoj modelů hlubokého učení. Běží jak na CPU, tak GPU. Podporuje konvoluční sítě, rekurentní sítě, jakož i kombinaci obou. Keras je navržen tak, aby umožnil rychlé experimentování s hlubokými neuronovými sítěmi, byl uživatelsky příjemný, modulární a rozšiřitelný. Nezpracovává samotné operace nízké úrovně, jako jsou produkty tensor, konvoluce a tak dále. Místo toho spoléhá na specializovanou, dobře optimalizovanou tensorovou manipulační knihovnu, která má sloužit jako „backend engine“ Kerasu. Namísto výběru jediné knihovny tensorů a provádění implementace Kerasu vázané na tuto knihovnu řeší Keras tento problém modulárním způsobem a lze do něj bez problémů zapojit několik různých backend engine. V současné době má Keras k dispozici tři implementace backend engine: backend TensorFlow, backend Theano a backend CNTK [28].

TensorFlow je open-source softwarová knihovna pro numerické výpočty pomocí dataflow grafů. Uzly grafu představují matematické operace, zatímco hrany grafů představují vícerozměrná pole dat (tensory), která mezi nimi proudí. Tato flexibilní architektura umožňuje nasazení výpočtů na jeden nebo více CPU nebo GPU na počítačích, serverech nebo mobilních zařízeních bez přepisování kódu. TensorFlow se především používá pro aplikace strojového učení, jako jsou neuronové sítě [29].

5 Implementace

Cílem této práce je implementace algoritmu pro detekci zavřených očí, který bude pracovat s webovou kamerou v reálném čase. Tento algoritmus by měl představovat zjednodušený systém detekce únavy řidiče, ve smyslu identifikace mikrospánku. Mrkání je zdrojem falešných detekcí a musí být pro správnost systému ignorováno. Zjednodušeně bude systém probíhat takto:

1. Detekce tváře v každém snímku vytvořeném webovou kamerou.
2. Pro detekovanou tvář jsou detekovány oči.
3. U každého detekováno oka je zjištěno, zda je oko otevřené nebo zavřené.
4. Pokud bylo zjištěno, že během krátkého časového úseku byli oči otevřeny, pak zavřeny a otevřeny, dospěli jsme k závěru, že osoba mrkla.
5. Samotné mrknutí trvá zhruba 0,3 sekund [37]. Pokud po této době budou oči stále zavřeny, systém detekuje mikrospánek a zahlásí varování.

Existuje několik způsobů, jak je možné detekovat zavřené oči. V této práci k tomu použijeme modely konvoluční neuronové sítě, které povedou k vytvoření binárních klasifikátorů.

Vybranými modely konvoluční neuronové sítě jsou LeNet-5, VGGNet a speciální případ Spatial Transform Network. Tyto modely budou vytrénovány na datové sadě Closed Eyes In The Wild (CEW). Ta se skládá přibližně z 2400 očních snímků ve velikosti 24×24. Tyto snímky byly extrahovány z tváří lidí různé pleti, barvy očí, lidí s brýlemi i bez nich [27].

5.1 Sestavení modelů konvoluční sítě

Před samotným tréninkem modelů je potřeba správně nakonfigurovat proces učení. K tomu je potřeba určit optimalizační funkci, ztrátovou funkci, popřípadě některé metriky jako přesnost.

Ztrátová funkce je měřítkem toho, jak dobrý je náš model při dosahování daného cíle. Měří nepřesnost klasifikace po projití každé dávky v síti a závisí pouze na výstupních hodnotách sítě.

Optimalizační funkce se používá pro minimalizaci ztrátové funkce pomocí aktualizací vah při použití gradientů. Váhy jsou nastavovány tak, aby došlo ke snížení celkové chyby sítě a postupně tak bylo dosaženo jejího globálního minima. Metoda *Adam* (adaptive

moment estimation) je často používaná pro svou rychlou konvergenci, protože ke změně rychlosti učení dochází už v průběhu učení sítě.

Rychlost učení určuje, jak rychle se vypočítávají optimální váhy pro model. Menší rychlost učení může vést k přesnější váze, ale doba potřebná k výpočtu váhy bude delší.

V rámci této práce je použita jako ztrátová funkce *binární crossentropie* a optimalizační funkce *Adam* s rychlostí učení $lr = 0,001$.

Vstupem všech tří modelu jsou parametry

1. width- šířka snímku
2. height - výška snímku
3. depth - počet kanálu (hloubka) snímku
4. classes - počet tříd v úkolu klasifikace

Všechny vstupní snímky v naší datové sadě budou 24 pixelů široké, 24 pixelů vysoké a mají hloubku 1. Vzhledem k tomu, že v datové sadě existují jen dvě třídy (zavřené/otevřené), nastavili jsme počet tříd na 2.

5.2 Trénink modelů

Rozdělení validace náhodně rozdělí data pro trénování a testování. Během tréninku je moci vidět validační ztrátu, která udává střední kvadratickou chybu našeho modelu na validační sadě. Provedeme validaci rozdělení na 0,2, což znamená, že 20% tréninkových dat, které poskytujeme modelu, bude vyhrazeno pro testování výkonnosti modelu.

Počet epoch je počet opakovaných cyklů dat. Čím více epoch provozujeme, tím více se model zlepší, až do určitého okamžiku. Po tomto okamžiku se už model během každé epochy nezlepšuje. Čím více epoch je, tím déle bude trénink modelu trvat. Naše zvolené modely jsou trénovány pro 20 epoch s velikostí dávky 64.

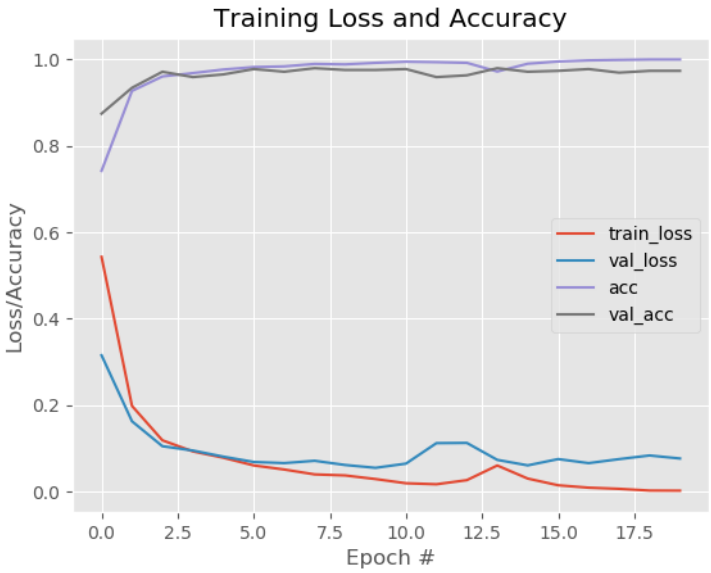
Všechny naše trénované modely jsou sekvenčního typu. To umožňuje vytvářet model vrstvu po vrstvě. Jedná se o nejjednodušší způsob, jak vytvářet model v Kerasu.

Dropout je parametr, jenž pomáhá zabránit k přeučení sítě. Přeučení sítě znamená, že je síť přizpůsobena svým trénovacím datům natolik, že její výstup není možné úspěšně použít pro klasifikaci dat jiných. Dropout během učení vynechá náhodnou množinu aktivací tak, že jejich hodnoty nastaví na hodnotu 0.

5.2.1 LeNet-5

Naše první konvoluční vrstva se učí s 20 filtry, každý o velikosti 5×5. Poté je použita aktivační funkce ReLU následovaná škálovací vrstvou (max-pooling) 2×2 s krokem

2×2, čímž se sníží velikost vstupu o 75%. Pak pokračujeme další konvoluční vrstvou, v pořadí druhou, tentokrát se však učí s 50 filtry namísto 20. Jako další je plně propojená vrstva s 500 uzly a aktivační funkcí ReLU. Model zakončuje klasifikátor Softmax.



Obr. 5.1 Graf přesnosti a ztrát při tréninku LeNet.

```
[INFO] evaluating network...
              precision    recall  f1-score   support

       close       0.97      0.97      0.97       239
        open       0.98      0.97      0.97       246

   micro avg       0.97      0.97      0.97       485
   macro avg       0.97      0.97      0.97       485
weighted avg       0.97      0.97      0.97       485
```

Obr. 5.2 Výstupní tabulka přesnosti LeNet.

kde *micro avg* představuje průměr na základě sumarizace jednotlivých pravdivých pozitivních vzorků, falešných pozitivních vzorků a falešných negativních vzorků každé třídy, *macro avg* je průměr z průměrů každé třídy, *f1-score* je vážený průměr mezi *precision* a *recall*, *recall* vyjadřuje intuitivní schopnost klasifikátoru najít všechny pozitivní vzorky a *support* počet výskytů v každé třídě [38].

Z pohledu na graf ztrát a přesnosti v čase (obrázek 5.1) lze vyčíst, že se naše síť chová dobře. Po pouhých pěti epochách již LeNet dosahuje přesnosti klasifikace 95%. Ztráta jak na tréninkových, tak na testovacích datech postupně klesá, protože naše míra učení zůstává konstantní a neupadá. Na konci dvacáté epochy dosahujeme 97% přesnosti na našem testovacím souboru, viz. obrázek 5.2. Celkový čas tréninku byl 10 minut a 35 sekund.

5.2.2 MiniVGGNet

Model MiniVGGNet je menší verzí VGGNet, který umožní rychlejší a méně náročnější trénink na našem systému. Pro snazší popis architektury zavedeme zkratky jako CONV pro konvoluční vrstvu, ACT pro aktivační funkci, BN pro BatchNormalization, FC pro plně propojenou vrstvu a POOL pro škálovací vrstvu (MaxPooling).

Architektura se skládá z (CONV => RELU => BN) * 2 => POOL => DROPOUT. Jako první je definována CONV s 32 filtry, každý o velikosti 3×3. Poté je aplikována aktivační funkce ReLU, která je vzápětí přiváděna do vrstvy BN. Abychom zmenšili prostorové dimenze našeho vstupu, namísto použití POOL použijeme jinou sadu CONV => RELU => BN. To umožní naší síti učit se bohatším funkcím, což je běžnou praxí při tréninku hlubších CNN. Pokračujeme použitím POOL s velikostí 2×2. Vzhledem k tomu, že není explicitně nastaven krok, Keras implicitně předpokládá, že náš krok bude roven maximální velikosti škálování, tedy 2×2. Poté je použit DROPOUT s pravděpodobností $p = 0,25$, a to znamená, že uzel z POOL se náhodně odpojí od další vrstvy s pravděpodobností 25% během tréninku.

Druhý blok odpovídá přesně stejnému vzoru jako výše, nyní se však učí CONV s 64 filtry (každý o velikosti 3×3) oproti 32 filtrům v prvním bloku. Zvyšujeme počet filtrů, protože se prostorová vstupní velikost zmenšuje s tím, čím hlouběji jsme v síti. Dále přichází naše první a jediná sada FC => RELU vrstev. Naše plně propojená vrstva má 512 uzlů, po nichž následuje aktivace ReLU a BN. Následuje DROPOUT $p = 0,5$ aplikovaný mezi plně propojenými vrstvami. Nakonec použijeme klasifikátor Softmax.

Obr. 5.3 Graf přesnosti a ztrát při tréninku MiniVGGNet.

Z dokončeného trénování můžeme vidět, obrázek 5.3 a obrázek 5.4, že model Mi-

```
[INFO] evaluating network...
              precision    recall  f1-score   support

       close       0.97      0.98      0.98       239
        open       0.98      0.98      0.98       246

   micro avg       0.98      0.98      0.98       485
   macro avg       0.98      0.98      0.98       485
weighted avg       0.98      0.98      0.98       485
```

Obr. 5.4 Výstupní tabulka přesnosti MiniVGGNet.

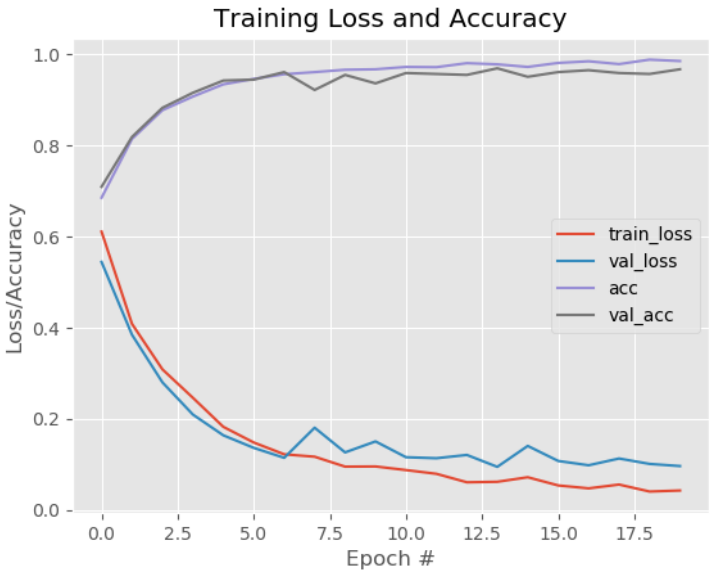
niVGGNet získává přesnost klasifikace 98%. Celkový čas trénování byl 38 minut a 38 sekund.

Tab. 5.1 Architektura MiniVGGNet

Typ vrstvy	**Velikost výstupu**	**Velikost filtru / krok**
Velikost obrázku	24×24×1	
CONV	24×24×32	3×3, K=32
ACT	24×24×32	
BN	24×24×32	
CONV	24×24×32	3×3, K=32
ACT	24×24×32	
BN	24×24×32	
POOL	12×12×32	2×2
DROPOUT	12×12×32	
CONV	12×12×64	3×3, K = 64
ACT	12×12×64	
BN	12×12×64	
CONV	12×12×64	3×3, K = 64
ACT	12×12×64	
BN	12×12×64	
POOL	6×6×64	2×2
DROPOUT	6×6×64	
FC	512	
ACT	512	
BN	512	
DROPOUT	512	
FC	2	
SOFTMAX	2	

5.2.3 Spatial Transformer Network

Spatial Transformer Network je mechanismus rozšiřující klasickou architekturu CNN. Použitá lokalizační síť (locnet) je ve formě konvoluční sítě a je tvořena ze dvou bloků. Každý blok je tvořen z vrstvy škálovací s velikostí 2×2 a vrstvy konvoluční s 20 filtry velikosti 5×5 pro první blok a velikostí 3×3 pro druhý blok, který přechází do plně propojené vrstvy s 50 uzly a aktivační funkcí ReLU. Výstup z lokalizační sítě je veden do modulu prostorové transformace, která obsahuje mechanismy jako generátor sítě a bilineární vzorkovač. Výstupem modulu prostorové transformace je interpolovaný obraz jako transformovaná vstupní mapa, která je přivedena do již známého modelu LeNet.



Obr. 5.5 Graf přesnosti a ztrát při tréninku STN.

```
[INFO] evaluating network...
              precision    recall  f1-score   support

       close       0.96      0.97      0.97       239
        open       0.97      0.96      0.97       246

   micro avg       0.97      0.97      0.97       485
   macro avg       0.97      0.97      0.97       485
weighted avg       0.97      0.97      0.97       485
```

Obr. 5.6 Výstupní tabulka přesnosti STN.

Z dokončeného trénování můžeme vidět, obrázek 5.5 a obrázek 5.6, že model Spatial Transformer Network získává přesnost klasifikace 97%. Celkový čas trénování byl 14 minut a 50 sekund.

5.3 Detekce

Nyní, když jsou natrénovány modely konvoluční neuronové sítě přejdeme k implementaci našeho detektoru zavřených očí. K tomu použijeme knihovnu počítačového vidění OpenCV.

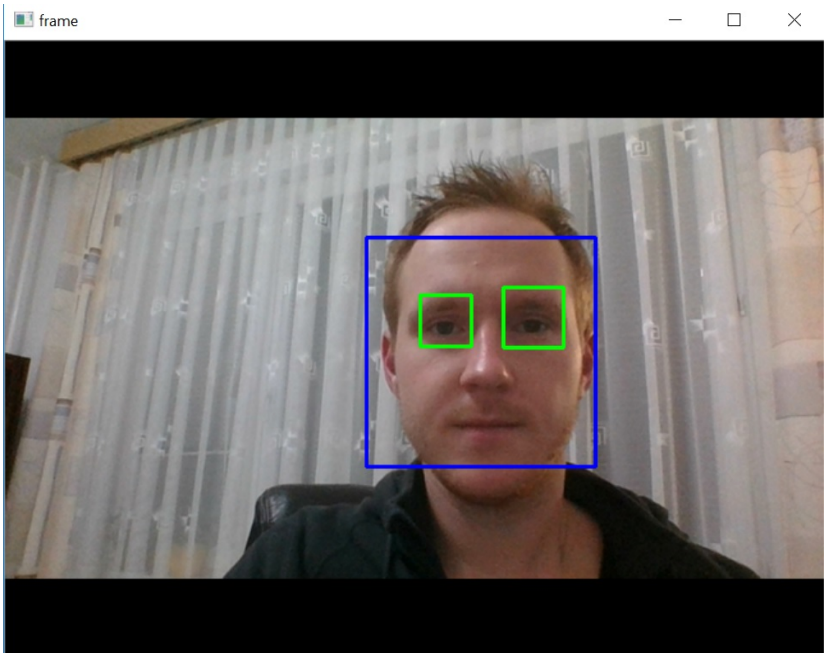
Nejdříve je přečten každý snímek z webové kamery, následuje detekce tváře a očí. Detekované oči ořízneme do snímku a ten vložíme do naší konvoluční sítě, která vrátí predikované hodnoty pravděpodobnosti otevřeného/zavřeného oka.

5.3.1 Detekce tváře a očí

Jako první je definována funkce pro detekci tváře, k tomu je použit natrénovaný Haarův kaskádový klasifikátor pro detekci tváře *haarcascade_frontalface_alt.xml* distribuovaný s balíčkem OpenCV.

Detekce objektů pomocí kaskádových klasifikátorů založených na Haarových příznacích je efektivní metoda detekce objektů navrhovaná Paulem Violou a Michaelem Jonesem v jejich článku z roku 2001. Jedná se o přístup založený na strojovém učení, kde kaskádová funkce je vycvičena z mnoha pozitivních a negativních snímků. Poté se používá k detekci objektů v různých obrazech. OpenCV již obsahuje mnoho předem připravených klasifikátorů pro detekci obličeje, očí, úsměvů a jiných [31].

Jakmile je ve snímku detekována tvář, je možné přistoupit k detekci očí. Detekovanou tváří je vymezen prostor, ve kterém oči hledat. Z tohoto prostoru lze pak snadněji získat oblasti pro levé a pravé oko. V případě detekce očí bude postup obdobným způsobem jako u detekce tváře s tím rozdílem, že bude použit vytvořený Haarův kaskádový klasifikátor *haarcascade_lefteye_2splits.xml* pro levé a *haarcascade_righteye_2splits.xml* pro pravé oko. Úspěšnou detekce tváře a očí znázorňuje obrázek 5.7.



Obr. 5.7 Příklad detekce tváře a očí

5.3.2 Detekce zavřených očí

Pro každý snímek detekce oka je predikován stav pomocí našich modelů. Aby byl náš natrénovaný model konvoluční sítě schopen predikovat ze snímku co nejpřesněji, musí být ve stejném formátu jako snímky, ze kterých se učil. Takže musíme udělat stejné úpravy i na našich očních souřadnicích. Operace jsou provedeny jak pro pravé, tak pro levé oko. Pokud jsme detekovali obě oči, změníme jejich velikost tak, aby byly ve stejné velikosti, jako u snímku, které jsme učili u našich modelů. To vše vede k správné predikce stavu oka.

Výsledek predikce obsahuje dvě hodnoty, které vyjadřují pravděpodobnost, zda je oko otevřené nebo zavřené. Pokud je hodnota pravděpodobnosti pro otevřené oko větší než hodnota pro zavřené, je nastaven stav oka jako otevřené, v opačném případě jako zavřené.

6 Testování

Cílem testování je zjistit úspěšnost detekce zavřených očí při použití různých modelů konvoluční sítě, kterými jsou LeNet, MiniVGGNet a speciální případ Spatial Transformer Network. Pro testování byl mechanismus detekce upraven tak, aby detekoval i samotné mrknutí oka. To z důvodu, že testujeme u mechanismu úspěšnost detekce zavřeného oka, nikoli, zda byl detekován mikrospánek.

Pro testování úspěšnosti detekce byly použity čtyři datové sady, kde každá sada obsahovala odlišnou osobu v různých prostředích. Tyto snímky byly zhotovené extrakcí jak z osobních, tak veřejných videosnímků. Snímky obsahovaly tvář s otevřenýma nebo zavřenýma očima (obrázek) .

Celkovou přesnost detekce vypočítáme ze vztahu:

$$presnost[\%] = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative} * 100 \tag{6.1}$$

kde $TruePositive$ je počet správně detekovaných snímků zavřených očí, $TrueNegative$ počet snímků se správnou detekcí otevřených očí, $FalsePositive$ počet snímků, u kterých došlo k chybné detekci zavřených očí a $FalseNegative$ počet snímků, u kterých došlo k chybné detekci otevřených očí. Jednotlivé výsledky testování jsou znázorněny v tabulkách 6.1, 6.2 a 6.3.

Tab. 6.1 Úspěšnost detekce modelu LeNet

LeNet	**Osoba1**	**Osoba2**	**Osoba3**	**Osoba4**
Počet snímků	145	150	134	145
TruePositive	11	4	35	5
TrueNegative	113	145	96	85
FalsePositive	5	1	3	55
FalseNegative	0	0	0	0
Přesnost	97%	99%	98%	62%
Průměr	**89%**			

Tab. 6.2 Úspěšnost detekce modelu MiniVGG

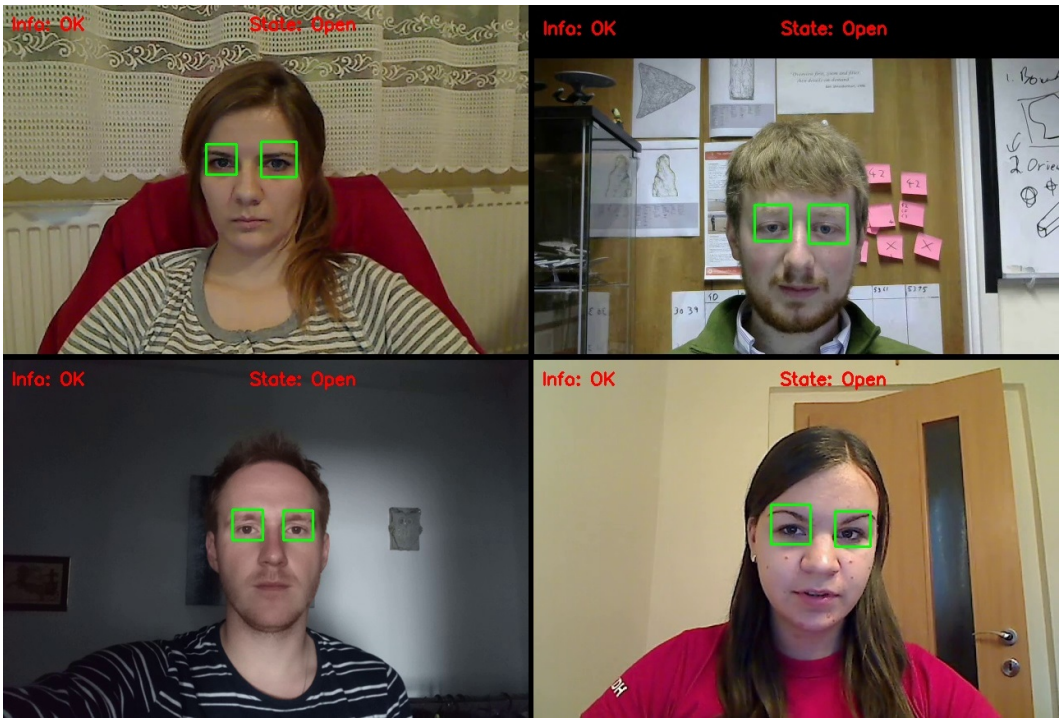
MiniVGG	**Osoba1**	**Osoba2**	**Osoba3**	**Osoba4**
Počet snímků	145	150	134	145
TruePositive	11	4	35	5
TrueNegative	7	145	95	122
FalsePositive	127	1	4	18
FalseNegative	0	0	0	0
Přesnost	95%	99%	97%	88%
Průměr	**95%**			

Tab. 6.3 Úspěšnost detekce Spatial Transformer Network

STN	**Osoba1**	**Osoba2**	**Osoba3**	**Osoba4**
Počet snímků	145	150	134	145
TruePositive	11	4	35	5
TrueNegative	122	145	96	53
FalsePositive	12	1	3	87
FalseNegative	0	0	0	0
Přesnost	92%	99%	98%	40%
Průměr	**82%**			

Nejlepšího výsledku, co do úspěšnosti detekce, dosahuje na testovacích datech konvoluční model miniVGGNet s průměrnou přesností 95%. Jeho úspěšnost se zejména projevila u testovací sady - Osoba4, ve kterých dosahoval výrazně lepších výsledku než zbývající modely. U Osoby4 docházelo často k falešným detekcím v případě, kdy snímaná osoba změnila svůj úhel pohledu tváře směrem dolů. U dalších osob vykazovaly modely velmi podobné úrovně přesnosti. Jako druhý se umístil model LeNet s průměrnou přesností 89% a jako třetí skončil model STN s 82%. Následující obrázky 6.1, 6.2 demonstrují správnou detekci systému, obrázek 6.3 detekci chybnou.

Obr. 6.1 Ukázka správné detekce zavřených očí u Osob 1-4 v pořadí zleva doprava, z vrchu dolů.



Obr. 6.2 Ukázka správné detekce otevřených očí u Osob 1-4 v pořadí zleva doprava, z vrchu dolů.

Modely byly dále porovnány i z hlediska časové náročnosti rychlosti detekce. Rychlost byla měřena od načtení snímku až po samotné zobrazení výsledku detekce, tedy

Obr. 6.3 Ukázka špatné detekce u otevřených očí u Osob 1-4 v pořadí zleva doprava, z vrchu dolů.

zahrnuje detekci tváře, očí i jejich stavu. Následující tabulka 6.4 obsahuje průměrnou rychlost detekce jednotlivých modelů v milisekundách (ms).

Tab. 6.4 Rychlost detekce

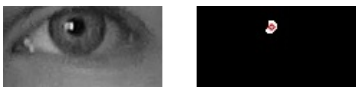
Průměrný čas detekce stavu oka ve snímku	
LeNet	29,3ms
miniVGG	52,9ms
STN	30,8ms

Z tabulky můžeme vidět, že nejkratšího času dosahuje model LeNet s necelými 30 milisekundami těsně následován modelem STN. Nejdelší čas patří modelu miniVGG-Net, který je řádově o 20 milisekund pomalejší, než první dva modely.

7 Srovnání

Ve své dřívější práci jsem se rovněž zabýval detekcí mrknutí v obrazech pomocí tří různých metod.

První z nich byla detekce zorničky pomocí prahování. Princip této metody spočíval v hledání nejtmavšího bodu celého oka - zorničky (obrázek 7.1). Pokud je tento bod v obraze nalezen, předpokládá, že je oko otevřené. Tato metoda je choulostivá na příliš velký odraz v oku, jenž zastínil zorničku. Další problém jsou tmavé řasy, které falešně detekovaly zavřené oko. Výhodou metody je robustnost v případě, kdy se osoba dívá do stran [30].



Obr. 7.1 Detekce zorničky pomocí prahování [30].

Další použitou metodou byla Houghova transformace pro nalezení kružnic. Oční duhovka představuje zdroj takovéto kružnice, pokud je v oku detekována, předpokládá se, že je oko otevřené (obrázek 7.2). Nevýhodou této metody je absence nalezení kružnice v případě, kdy se osoba dívá do stran nebo mžourá [30].

Obr. 7.2 Detekce kružnice, levý snímek - otevřené oko, pravý snímek - zavřené oko [30].

Poslední zvolenou metodou byla detekce zavřeného oka pomocí natrénovaného kaskádového klasifikátoru. Trénování probíhalo za pomoci knihovny OpenCV, k tomu byla zapotřebí sada snímků zavřeného a otevřeného oka. Tato metoda dosahovala nejlepších výsledků ze všech tří zmíněných a jako jediná tak může konkurovat úspěšnosti natrénovaných modelů konvoluční neuronové sítě [30]. Úspěšnost všech tří metod zobrazuje tabulka 7.1.

Tab. 7.1 Srovnání cizích metod

Metody detekce - srovnání	
Prahování	70%
Houghova transformace	73%
Kaskádový klasifikátor	93%

ZÁVĚR

Deep learning, neuronové sítě, umělá inteligence a strojové učení. Tato pojmy se víc než často používají při pokročilé analýze rozsáhlých dat, které v současné době berou svět útokem. Technicky jde o složitější datové modely, které dokážou například rozpoznat co je na obrázku, porozumět lidské řeči či detekovat únavu řidiče. Únava řidiče je významným faktorem velkého počtu dopravních nehod. Vývoj technologií pro detekci nebo prevenci ospalosti při řízení je velkou výzvou v oblasti systémů pro předcházení nehodám.

Navrhovaný systém by mohl zabránit nehodám způsobeným ospalostí při řízení. Informace o poloze hlavy a očí se získají pomocí různých algoritmů pro zpracování obrazu. Během monitorování je systém schopen rozhodnout, zda jsou oči otevřené nebo zavřené. Pokud jsou oči zavřené delší dobu než je doba mrknutí, vydá se varovný signál. Samotná detekce očí je nezbytnou součástí pro přesné detekování stavu očí (zavřené/otevřené). Pokud tedy není systém schopný detekovat oči ve videu, jsou další operace spjaté s detekcí stavu očí zbytečné.

V této práci jsme se jak teoreticky, tak prakticky seznámili s prací neuronových síti. Byly představeny modely konvolučních neuronových sítí, z nich byly vybrány tři, které jsem dále trénoval na sadě snímků. Vybranými modely byly LeNet, VGGNet a Spatial Transformer Network. Výstup z těchto modelů byl dále použit pro detekci stavu očí. Modely byly testovány jak časově, tak co do úspěšnosti. Nakonec byly výsledky porovnány s ostatními metodami detekce.

V porovnání s ostatními metodami dosahuje detekce zavřených očí pomocí neuronových sítí o poznání lepších výsledků, přičemž nejlepšího výsledku dosáhl model konvoluční neuronové sítě VGGNet, který prokázal svou schopnost především u snímku se špatnou kvalitou.

SEZNAM POUŽITÉ LITERATURY

[1] *Neuronové sítě - jednotlivý neuron* [online]. [cit. 2019-03-18]. Dostupný z WWW: http://portal.matematickabiologie.cz/res/f/neuronove-site-jednotlivy-neuron.pdf

[2] *McCulloch-Pitts Neuron - Mankind's First Mathematical Model Of A Biological Neuron* [online]. 2018 [cit. 2019-03-18]. Dostupné z WWW: https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1

[3] SHIFFMAN, Daniel. *Chapter 10. Neural Networks* [online]. [cit. 2019-03-18]. Dostupné z WWW: https://natureofcode.com/book/chapter-10-neural-networks/

[4] NIELSEN, Michael. *Neural Networks and Deep Learning* [online]. [cit. 2019-03-11]. Dostupný z WWW: http://static.latexstudio.net/article/2018/0912/neuralnetworksanddeeplearning.pdf

[5] *Different activation functions* [online]. [cit. 2019-03-11]. Dostupný z WWW: https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788397872/1/ch01lvl1sec16/different-activation-functions

[6] *Terms used in Neural Networks* [online]. [cit. 2019-03-18]. Dostupné z WWW: https://iq.opengenus.org/terms-used-in-neural-networks/

[7] *Solving XOR with a single Perceptron* [online]. 2018 [cit. 2019-03-18]. Dostupné z WWW: https://medium.com/@lucaspereira0612/solving-xor-with-a-single-perceptron-34539f395182

[8] GUPTA, Tushar *Deep Learning: Feedforward Neural Network* [online]. 2017 [cit. 2019-03-18]. Dostupné z WWW: https://towardsdatascience.com/deep-learning-feedforward-neural-network-26a6705dbdc7

[9] *Neuronové sítě* [online]. [cit. 2019-03-18]. Dostupné z WWW: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=21471

[10] BOHATÁ, Tereza. *Využití hlubokých neuronových sítí pro predikci schizofrenie* [online]. 2018 [cit. 2019-03-28]. Dostupné z WWW: https://is.muni.cz/th/dfpcy/DP_Bohata.pdf

[11] *Learning and Adaptation* [online]. 2018 [cit. 2019-04-03]. Dostupné z WWW: https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_learning_adaptation.htm

[12] MCGONAGLE, John, George SHAIKOUSKI, Christopher WILLIAMS, Andrew HSU a Jimin KHIM *Backpropagation* [online]. 2009 [cit. 2019-04-07]. Dostupné z WWW: https://brilliant.org/wiki/backpropagation/

[13] KUCHAŘ, David. *Deep learning v analýze obrazu* [online]. 2018 [cit. 2019-04-07]. Dostupné z WWW: https://dspace.vsb.cz/bitstream/handle/10084/128561/KUC0257_FEI_B2647_2612R025_2018.pdf?sequence=1&isAllowed=y

[14] VONDRÁK, Ivo. *Neuronové sítě* [online]. 2009 [cit. 2019-04-07]. Dostupné z WWW: http://vondrak.cs.vsb.cz/download/Neuronove_site.pdf

[15] ROJAS, Raúl. *The Hopfield Model* [online]. [cit. 2019-04-10]. Dostupné z WWW: https://page.mi.fu-berlin.de/rojas/neural/chapter/K13.pdf

[16] *Artificial Neural Network - Hopfield Networks* [online]. [cit. 2019-04-10]. Dostupný z WWW: https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_hopfield.htm

[17] ROJAS, Raúl. *Kohonen Networks* [online]. [cit. 2019-04-10]. Dostupné z WWW: https://page.mi.fu-berlin.de/rojas/neural/chapter/K15.pdf

[18] *Self-Organizing Maps* [online]. 2014 [cit. 2019-04-10]. Dostupné z WWW: http://gorayni.blogspot.com/2014/10/self-organizing-maps.html

[19] *Convolutional Neural Network* [online]. [cit. 2019-04-14]. Dostupné z WWW: http://deeplearning.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/

[20] *Max-pooling / Pooling* [online]. [cit. 2019-04-14]. Dostupné z WWW: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

[21] *Matrix Multiplication with Metal Performance Shaders* [online]. 2017 [cit. 2019-04-14]. Dostupné z WWW: https://machinethink.net/blog/mps-matrix-multiplication/

[22] ROSEBROCK, Adrian. *Deep Learning for Computer Vision with Python* [online]. 2017 [cit. 2019-04-14]. Dostupné z WWW: https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/

[23] *CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more* [online]. [cit. 2019-04-14] Dostupné z WWW: https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

[24] *Student Notes: Convolutional Neural Networks (CNN) Introduction* [online]. [cit. 2019-04-14] Dostupné z WWW: https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

[25] DESHPANDE, Adit. *The 9 Deep Learning Papers You Need To Know About (Understanding CNNs Part 3)* [online]. 2015 [cit. 2019-04-18]. Dostupné z WWW: https://adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html

[26] JADERBERG, Max, Karen SIMONYAN, Andrew ZISSERMAN a Koray KAVUKCUOGLU. *Spatial Transformer Networks* [online]. 2015 [cit. 2019-04-18]. Dostupné z WWW: https://arxiv.org/abs/1506.02025

[27] *Closed Eyes In The Wild (CEW)* [online]. [cit. 2019-04-18]. Dostupné z WWW: http://parnec.nuaa.edu.cn/xtan/data/ClosedEyeDatabases.html

[28] *Keras Documentation* [online]. [cit. 2019-05-10]. Dostupné z WWW: https://keras.io/

[29] *Tensorflow* [online]. [cit. 2019-05-10]. Dostupné z WWW: https://www.tensorflow.org/

[30] HOLOMEK, Adam. *Detekce mrkání očí v obrazech* [online]. 2016 [cit. 2019-05-10]. Dostupné z WWW: https://dspace.vsb.cz/bitstream/handle/10084/118851/HOL0229_FEI_B2647_2612R025_2016.pdf?sequence=1&isAllowed=y

[31] *Face Detection using Haar Cascades* [online]. [cit. 2019-05-10]. Dostupné z WWW: https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html

[32] *Multilayer perceptron example* [online]. [cit. 2019-05-12]. Dostupné z WWW: https://github.com/rcassani/mlp-example

[33] *Artificial Neural Networks/Feed-Forward Networks* [online]. [cit. 2019-05-12]. Dostupné z WWW: https://en.wikibooks.org/wiki/Artificial_Neural_Networks/Feed-Forward_Networks

[34] *Boltzmann Machine* [online]. [cit. 2019-05-13]. Dostupné z WWW: https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_boltzmann_machine.htm

[35] *Boltzmann Machines* [online]. 2014 [cit. 2019-05-13]. Dostupné z WWW: http://gorayni.blogspot.com/2014/06/boltzmann-machines.html

[36] SAHA, Sumit. *A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way* [online]. 2018 [cit. 2019-05-13]. Dostupné z WWW: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[37] *Proč je tak důležité mrkání?* [online]. [cit. 2019-05-15]. Dostupné z WWW: https://www.vasecocky.cz/mrkani-proc-je-dulezite.html

[38] *sklearn.metrics.precision_ recall_ fscore_ support* [online]. [cit. 2019-05-15]. Dostupné z WWW: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html#sklearn.metrics.precision_recall_fscore_support

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

CNS	Centrální nervový systém
MCP	McCullosh-Pitts
ReLU	Rectified Linear Unit
LMS	Least Mean Square
ADALINE	Adaptive Linear Element
CNN	Convolution Neural Network
STN	Spatial Transformer Network
CPU	Central Processing Unit
GPU	Graphic Processing Unit
CNTK	Microsoft Cognitive Toolkit
Adam	Adaptive moment
MLP	Multi Layered Perceptron
RGB	Red Green Blue
VGG	Visual Geometry Group

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I. NÁZEV PŘÍLOHY

Obsah přílohy