STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Rozpoznání ručně psaných číslic pomocí konvoluční neuronové sítě

Filip Chytil
Olomoucký kraj

Přerov 2020

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Rozpoznání ručně psaných číslic pomocí konvoluční neuronové sítě

Recognition of Handwritten Digits Using Convolutional Neural Network

Autor: Filip Chytil
Škola: Střední průmyslová škola Přerov, Havlíčkova 2, 750 02 Přerov
Kraj: Olomoucký kraj
Konzultant: Mgr. Magdalena Gažarová

Přerov 2020

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Přerov dne 18. 3. 2020 …….

Filip Chytil

Anotace

Cílem této práce je vytvořit grafické uživatelské rozhraní (GUI) zaměřující se na řešení problému rozpoznávání ručně psaných číslic pomocí konvoluční neuronové sítě společně s její problematikou. Dále se práce věnuje principům umělých neuronových sítí, struktuře neuronových sítí a vizualizaci konvolučních vrstev. Mnou vytvořená architektura FilChy model je použita v praktické části spolu s dalšími před vytvořenými strukturami modelů. V rámci práce bylo zpracováno mnoho dalších témat souvisejících s počítačovým viděním a jejich implementací.

Klíčová slova

Neuronové sítě; konvoluční neuronové sítě; hluboké učení; počítačové vidění; klasifikace číslic

Annotation

The aim of this work is to create a graphical user interface (GUI) focused on solving the problem of Handwritten digits recognition using a convolutional neural network together with its problems. Furthermore, the thesis deals with the principles of artificial neural networks, neural network structure and visualization of convolutional layers. The FilChy model I created is used in the practical part together with other pre-created model structures. Many other topics related to computer vision and its implementation were elaborated.

Keywords

Neural networks; convolutional neural networks; deep learning; computer vision; digit classification

Obsah

1 ÚVOD

Jednou z prvních průkopnických konvoluční neuronových sítí (CNN) byl model pojmenovaný LaNet5 vytvořený Yannem LeCunem v roce 1994. Architektura tohoto modelu byla zásadní v tom, že vstupní obrazové prvky byly distribuovány [1]. CNN se dostala do popředí vědců, jenže poté delší dobu nepřicházely převratnější výsledky, a proto mezi roky 1998 až 2010 nebyly CNN tolik populární. Zlom nastal, když v roce 2010 publikoval Dan Claudiu Ciresan a Jurgen Schmithuber jednu z prvních implementací CNN využívající GPU [1]. Tímto objevem se zrychlil proces učení. CNN se v praxi ukázala jako velice spolehlivá v problematice rozeznání obrazu.

CNN jsou v dnešní době klíčovým faktorem v aplikacích strojového učení. Stěžejní jsou v mnoha problémech (detekci objektů, segmentace obrazu a rozpoznávání tváře, …). Jedním z těchto problémů je i rozpoznávání ručně psaných číslic, které je využíváno v mnoha oborech (pošta, bankovnictví, …). CNN jsou základní kamenem dnes již modernějších metod zabývajících se zpracováním obrazu, jako je například R-CNN (Region CNN) vytvořen pro detekci objektů, Mask R-CNN, který dokáže klasifikovat každý pixel v předaném obrázku a mnoho dalších.

Tato práce se zabývá tvorbou HDR (handwritten digits recognition) softwaru pro detekci a klasifikaci ručně psaných číslic. Vzhledem k odlišnosti všech rukopisů a možnosti získat co možná nejobsáhlejší datovou sadu byl model natrénován na datové sadě MNIST database (Modified National Institute of Standarts and Technology). Hlavní program dostane na vstupu fotografii bílé stránky popsané číslicemi desítkové soustavy a na výstupu vrátí detekované všechny číslice.

2 TECHNOLOGIE PROJEKTU

Pro vytvoření programu byl použit programovací jazyk Python 3, protože obsahuje velkou standartní knihovnu a mnoho operačních knihoven, které implementují funkce strojového učení ve velice efektivní formě [2].

2.1 Využité knihovny

2.1.1 Tensorflow

Tensorflow je knihovna se zaměřením na strojové učení. Umožňuje vytváření modelů pomocí Keras API na vysoké úrovni s jednoduchou implementací. V tomto projektu je Tensorflow použit pro natrénování modelů a predikce nakresleného čísla. V této práci se pracuje s verzí 2.1.0.

2.1.2 PyQt5

PyQt5 poskytuje multiplatformní sadu knihoven, implementující API pro přístup k aspektům GUI (grafického uživatelského rozhraní) aplikací. V tomto projektu je tato knihovna využita pro tvorbu GUI. V této práci se pracuje s verzí 5.14.1.

2.1.3 Numpy

Numpy je knihovna poskytující práci s n-dimenzionálními maticemi a vektory, společně se základními operacemi. V tomto projektu je Numpy použita pro zpracování dat do požadovaného formátu *float32*. Ve zdrojovém kódu se používá pod zkratkou *np*. V této práci se pracuje s verzí 1.16.5.

2.1.4 Matplotlib

Matplotlib poskytuje funkce pro vykreslování grafů a zobrazování množiny obrázků. V tomto projektu je tato knihovna využita pro zobrazení grafu modelů a pro vykreslování obrázků vrstev. V této práci se pracuje s verzí 3.1.1.

2.1.5 OpenCV

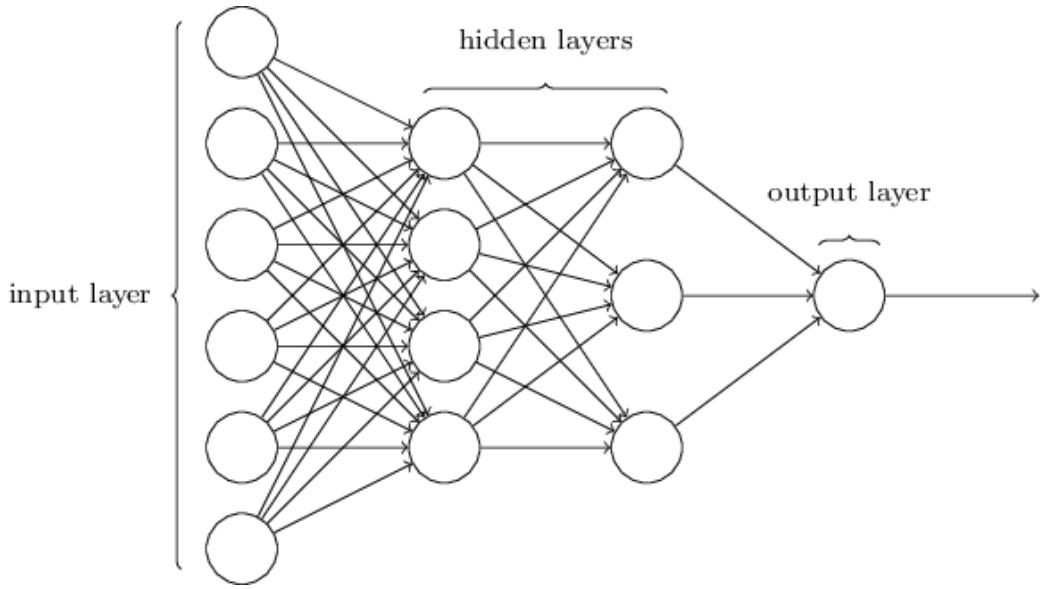
OpenCV (Open Source Computer Vision) je knihovna obsahující algoritmy pro počítačové vidění a práci s médii (obrázkem nebo videem). Ve zdrojovém kódu se OpenCV používá pod zkratkou *cv2* a je využit pro operace s obrázky. V této práci se pracuje s verzí 4.2.0.

3 UMĚLÁ NEURONOVÁ SÍŤ

Umělé neuronové sítě (UNS) se inspirovaly strukturou lidské nervové soustavy. Základním prvkem přirozené i umělé neuronové sítě je neuron neboli perceptron. Neurony jsou navzájem propojeny a předávají si signály. Platí přitom, že každý neuron může mít více vstupů, ale jen jeden výstup (tento výstup však může být poslán více než jednomu dalšímu neuronu) [3].

3.1 Princip sítě

Neuronová síť (NS) je uskupení matematických modelů zpracovávajících váhové uskupení, na jehož základě vygeneruje výstup. Neurony jsou mezi sebou propojeny a topologicky uspořádány do struktury komunikující přes orientované ohodnocené spoje. Každá neuronová síť může být naprosto jiná, záleží pouze na typu přenosové funkce, topologickém uspořádání [4]. Neurony jsou uspořádány do vrstev viz. obr. 1. Vrstvy v neuronové síti dělíme na přijímací vrstvu, skryté vrstvy a výstupní vrstvu (input layer, hidden layers, output layers).



Obr. 1: Příklad neuronové sítě [22]

3.2 Model umělého neuronu

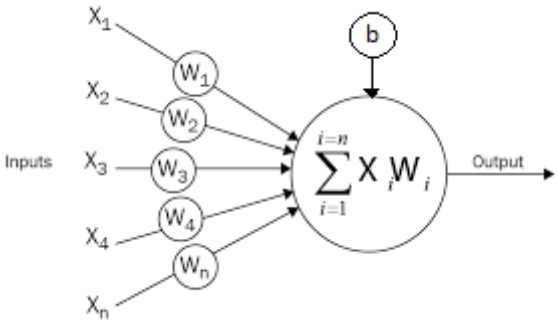
Základním algoritmem využívající vztahu (1), kde každý neuron dostane vstup *x* o *i* vstupních dat neuronu, *w* vyjadřuje uložení zkušeností (důležitost) do neuronu a *b* vyjadřuje prahovou hodnotu [5].

$$y = \sum_{i=1}^{i=n}(w_i x_i) + b \tag{1}$$

Čím vyšší vyjde hodnota *y* tím více si je neuron jistý svým výsledkem. Jeli *b* větší jak násobek *w* a *x*, pak je neuron v pasivním stavu viz. vztah (2) [5].

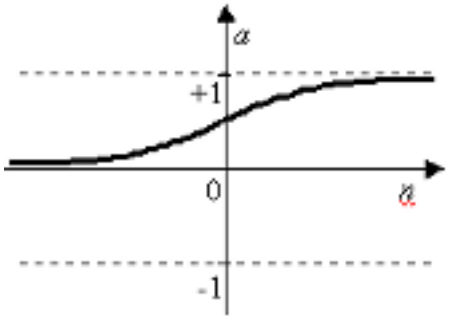
$$b > \sum_{i=1}^{i=n}(w_i x_i) \tag{2}$$

Jelikož neuron vypočítá vždy právě jeden výstup z obdržených výstupů předešlých neuronů jsou všechny neurony parametricky závislé na topologii, která se v průběhu nemění. Z tohoto vyplývá, že se neuronová síť využívá k vlastním výsledkům.

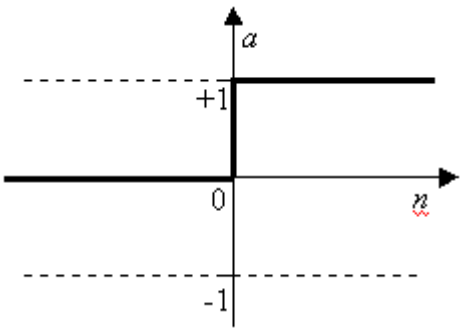


Obr. 2: Matematický model neuronu [23]

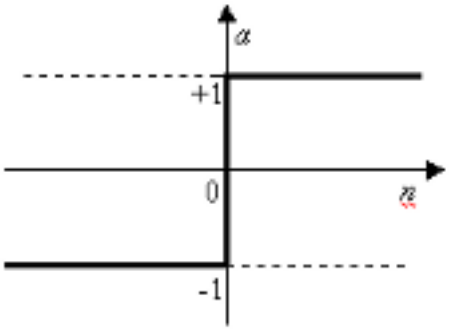
Přenosová funkce, někdy též aktivační funkce moduluje výstup neuronu do námi zvolené datové množiny, dle výběru aktivační funkce. V procesu testování se nejvíce osvědčila funkce sigmoid (viz. obr. 3), funkce jednotkového skoku (viz. obr. 4), aktivační funkce signum (viz. obr. 5) a funkce lineární (viz. obr. 6) [6]. Výběr vhodné funkce má velký vliv na konečný výsledek neuronové sítě.



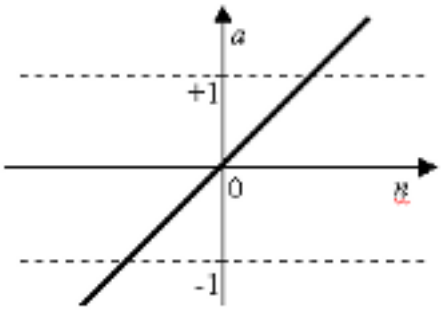
Obr. 3: Aktivační funkce sigmoid [24]



Obr. 4: Aktivační funkce jednotkového skoku [24]



Obr. 5: Aktivační funkce signum [24]



Obr. 6: Aktivační lineární funkce [24]

3.3 Principy učení

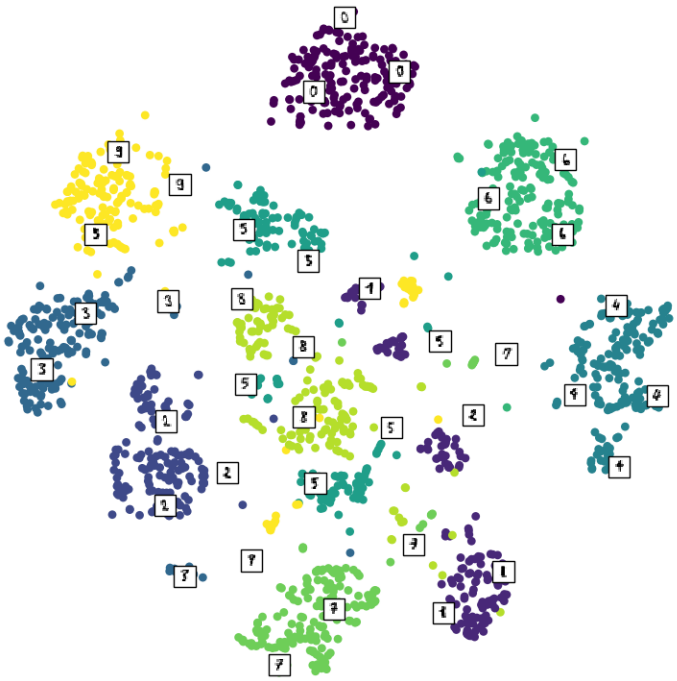
Princip učení neuronu spočívá v průběhu aktivní dynamiky, která nastává při transformaci vstupních vektorů na hodnotu výstupní. Parametry neuronu jsou v tuto dobu konstantní. Oproti tomu adaptační dynamika spočívá v nastavení parametrů tak, aby byl schopen neuron vypočítat požadovanou transformaci. Parametry, které jsou v průběhu učení zpravovány, jsou zpravidla jen váhy vstupních synapsí neuronu. Vstupní váhy *w* tedy představují paměť neuronu. Učení neuronové sítě obstarává adaptační algoritmus, který nastavuje hodnotu vah neuronu. Tento proces probíhá iterativně spolu s předloženými vstupními hodnotami, s kterými nadále pracuje.

Největší výhodou neuronových sítí je schopnost neuronové sítě najít transformaci i u těch úloh, které jsou analyticky obtížně řešitelné, či neřešitelné. Úspěšnost neuronové sítě ovlivňuje poskytnuté množství dat. Ačkoli jsou neuronové sítě perfektním nástrojem pro řešení obtížných úloh, není zde zaručeno, že vždy dojde k požadovanému výsledku [7].

3.3.1 Učení bez učitele

Učení bez učitele (unsupervised learning) je samoorganizovanou metodou strojového učení a pracuje na principu shlukování, přičemž hledá ve vstupních datech sobě podobné elementy, které poté rozřadí do skupin s nejpodobnějšími vlastnostmi (viz. obr. 7) [8]. Do učení není

možné zasahovat, což znamená, že nelze v průběhu kontrolovat správnost postupu učení. Celé učení je založeno pouze na informacích obsažených ve vstupních datech.



Obr. 7: Rozřazení ručně psaných číslic do skupin [25]

3.3.2 Učení s učitelem

Učení s učitelem (supervised learning) je metoda strojového učení, která na vstupu dostane množinu *M*, která obsahuje *n* dvojic *x* (vstupní vektor) a *y* (odpovídající výstupní vektor) (vztah (3)). Má tedy konkrétní příklady správné transformace vstupních vektorů na vektory výstupní. Množina *M* je rozdělena na část testovacích a trénovacích dat. Poměr mezi těmito dvěma částmi není konstantně určen, a proto je třeba jej zvolit vzhledem k řešené úloze. Obvykle množina trénovacích příkladů obsahuje 70 % - 85 % dat z celé množiny dat [9]. Často používaný je i způsob, kde množina *M* je rozdělena místo do dvou na tři disjunktní části; mimo množiny testovací a trénovací je vytvořena také množina validační. V průběhu učení je periodicky vyhodnocována chyba na množině validační, díky které se může trénování předčasně zastavit, pokud nebude na validační množině klesat nějakou dobu chyba.

$$M = \{[x^1, y_p^1], [x^2, y_p^2], \dots, [[x^{nmax}, y_p^{nmax}]]\} \tag{3}$$

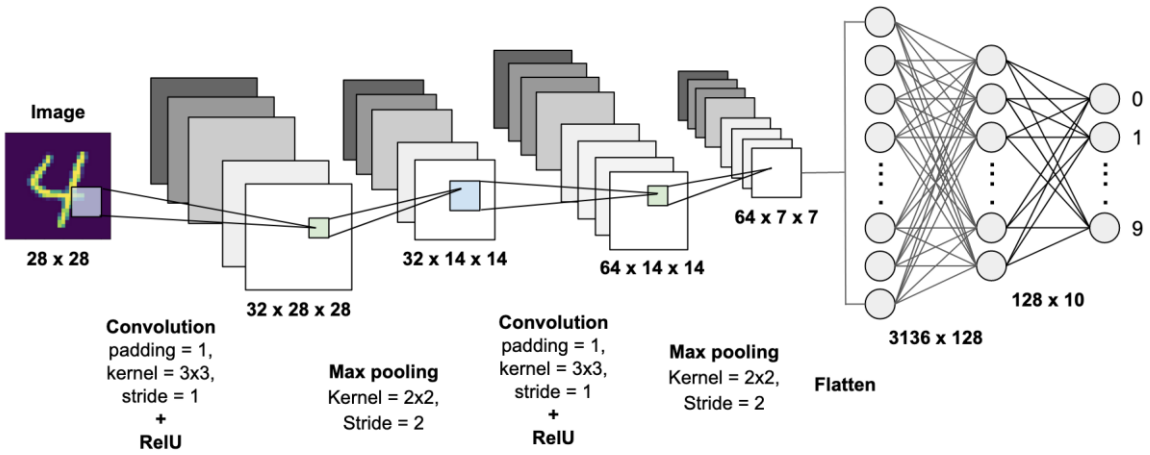
Algoritmus postupně předkládá neuronu jednotlivé prvky trénovací množiny, zjišťuje jeho výstupní hodnotu a na základě odchylky výstupní hodnoty neuronu a výstupní hodnoty požadované provádí korekci vah v neuronu. Interval, ve kterém algoritmus předloží neuronu alespoň jednou všechny vzory z trénovací množiny, se nazývá epocha. Pro správnost naučení neuronové sítě může být v závislosti na komplexnosti problému zapotřebí desítky až tisíce epoch.

4 KONVOLUČNÍ NEURONOVÁ SÍŤ

Konvoluční neuronová síť (CNN) je neuronová síť využívající matematickou operaci zvaná konvoluce. Konvoluční sítě slouží jako nástroj pro předzpracování obrazu, který zpracuje a předá ho neuronovým sítím jako vektor [10].

Typická konvoluční síť je ukázaná na obrázku 8. Jak je vidět, skládá se z několik vrstev. Mezi charakteristické vlastnosti patří získávání příznaků z recepčních polí, technologie sdílení vah mezi neurony a prostorové vzorkování.

Konvoluční neuronová síť dokáže z obrazu extrahovat ty nejužitečnější informace, které následně předá klasifikátoru (neuronové síti) ve formě vektoru, který dokáže poté vyhodnotit výstupní hodnotu. Z toho vyplývá, že konvoluční vrstva dokáže lépe využít informace ze vstupních hodnot.



Obr. 8: Konvoluční neuronová síť [26]

4.1 Architektura konvoluční neuronové sítě

Konvoluční neuronová síť se rozděluje na extrakci příznaků konvolučními vrstvami a klasifikace neuronovou sítí (viz. obr. 8). Konvoluční síť se skládá z několika vrstev. Každá vrstva má svou danou funkci a je implementována v síti za nějakým účelem. Hlavní typy funkcí, které jsou použity v praktické části zde budou popsány.

4.1.1 Vrstva zpracování obrazu

Vrstva zpracování obrazu (image processing layer) slouží k možnému předdefinováním sítě filtrů, které jsou konstantní během procesu trénování. Můžeme do této vrstvy předdefinovat velikost vstupního tenzoru (4) obsahující počet, šířku, výšku a hloubku obrazů.

$$x_{shape} = (x_n; x_w; x_h; x_d) \tag{4}$$

kde

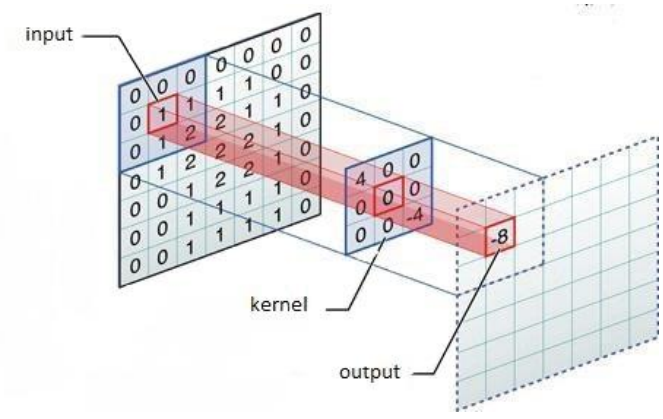
- x_{shape} – představuje velikost vstupního tenzoru;
- x_n – představuje velikost množiny vstupních obrazů;
- x_w – představuje velikost šířky obrazu v pixelech;
- x_h – představuje velikost výšky obrazu v pixelech;
- x_d – představuje hloubku obrazu, tzn. počet vrstev matice;

4.1.2 Konvoluční vrstva

Konvoluční vrstva (convolutional layer) je hlavní stavebním blokem používaným v konvolučních neuronových sítích. Výhodou konvolučních vrstev je schopnost automaticky se naučit velké množství filtrů. Konvoluční vrstva obsahuje sadu filtrů, jejichž parametr je třeba naučit přizpůsobit se datům [11]. Vždy je velikost filtrů menší nebo shodný jako vstupní hodnota. Maskou postupně projde každý pixel označující hodnotu barvy obrazu a vynásobí se hodnotou pixelu v masce, poté budou všechny hodnoty pixelu sečteny. Tento proces se nazývá obecná forma maticové konvoluce (5) [12].

$$\begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix} \cdot \begin{bmatrix} y_{11} & \cdots & y_{1n} \\ \vdots & \ddots & \vdots \\ y_{m1} & \cdots & y_{mn} \end{bmatrix} = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} x_{(m-i)(n-j)} y_{(1+i)(1+j)} \tag{5}$$

Výsledná suma maticové konvoluce bude novou hodnotou aktuálního pixelu, který se překrývá se středem jádra (viz. obr. 9). Pokud je jádro vstupních hodnot symetrické s jádrem masky, výsledná hodnota bude umístěna tam. Pokud symetrické není, tak se musí překlopit kolem její vodorovné i svislé osy.



Obr. 9: Umístění výsledné hodnoty konvoluce [27]

Pro zvolení filtrovací masky (kernel) je třeba znát množství parametrů, které daná funkce bude muset mít. Pro znázornění uvedeme příkladný filtr velikosti 4x4x1 (to znamená, že filtr se použije pro zpracování černobílého obrázku, který má pouze jeden barevný kanál vyjadřující světlost pixelu). Filtr 4x4x1 je roven 16 vahám, které budou určeny jako parametr. Vždy je nutno počítat s prahovou hodnotou, což znamená přičtení jedničky. Praktický příklad výpočtu

tohoto filtru: pro obraz o velikosti 28x28x1 aplikujeme 6krát filtr velikosti 4x4x1. Pro tento výpočet je nutno použít vzorec (6):

$$a = \frac{N-F}{krok+b} \tag{6}$$

kde

- a – představuje velikost strany výsledného obrazu;
- N – představuje velikost obrázku;
- F – představuje velikost masky;
- b – představuje prahovou hodnotu.

Po dosazení uvedených parametrů do zmíněného vzorce (7) získáme výslednou hodnou.

$$a = \frac{28-4}{1+1} = 12 \tag{7}$$

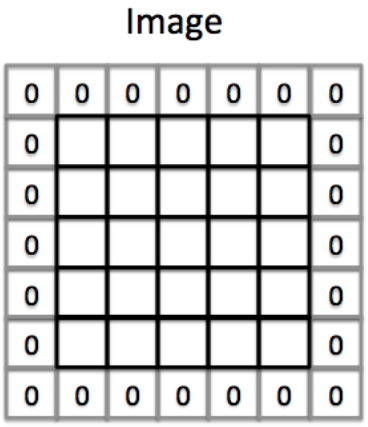
Výsledný obraz tedy bude ve tvaru (8), tedy 12x12x6.

$$(a; a; n) \tag{8}$$

kde

- a – představuje velikost strany výsledného obrazu;
- n – představuje počet filtrů aplikovaných na obraz.

Pro případy, kdy zvolíme jeden z výše uvedených parametrů jinak, a to tak, že nám výsledná velikost nebude vycházet jako celé číslo, budeme muset použít funkci zvanou zero-padding (viz. obr. 10) [13]. Funkce zero-padding cíleně zvětší obraz o jeden pixel na každé straně.



Obr. 10: Zero-padding [28]

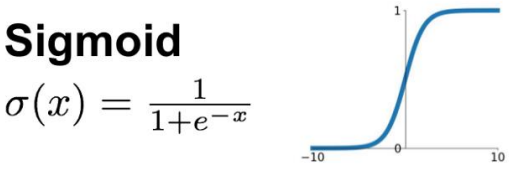
Pokud máme v modelu více konvolučních vrstev za sebou, což v praktické části této práce máme, můžeme již hovořit jako o hlubokém učení (deep learning).

4.1.3 Přenosové funkce

V kapitole 3.3 je popsáno fungování umělého neuronového modelu. V této kapitole je přiblížení k jeho přenosovým funkcím (aktivačním funkcím). Při volení aktivační funkce je nutné brát ohledy na problematiku dané práce. Špatná volba aktivační funkce může zapříčinit neúspěch modelu. Mezi základní aktivační funkce patří:

Sigmoida

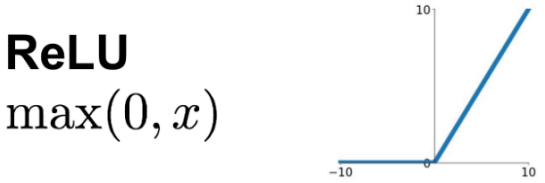
Hlavním důvodem, proč používáme funkci sigmoid je, že vrací hodnoty v intervalu $\langle 0; 1\rangle$. Používá se převážně u modelů určujících pravděpodobnost, což je číslo mezi 0 a 1 [14]. Problémem funkce Sigmoid je mizející gradient (vanishing gradient problem), který ztěžuje učení a možnost vyladění parametrů dřívějších vrstev. Tento problém se zhoršuje se zvyšujícím se počtem vrstev v architektuře naší umělé neuronové sítě [15].



Obr. 11: Přenosová funkce sigmoid [29]

ReLU

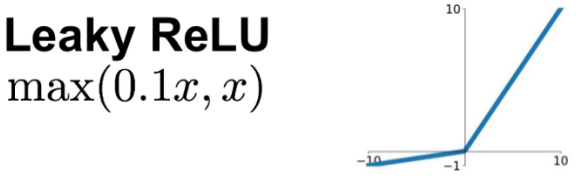
Funkce ReLU je v dnešní době nejpoužívanější aktivační funkcí v konvolučních neuronových sítích anebo v hlubokém učení. Funkce ReLU a její derivace jsou monotónní. Problémem této funkce je, že všechny záporné hodnoty jsou ve funkci automaticky nula, což snižuje modelu schopnost správně se přizpůsobit či vycvičit z přidělených dat. Tím se zapříčiní nepřiměřené mapování negativních hodnot v modelu [14].



Obr. 12: Přenosová funkce ReLU [29]

Leaky ReLU

Funkce Leaky ReLU je pokusem vyřešení problému funkce ReLU. Leaky ReLU zvětšuje rozsah funkce ReLU hodnotou *y* osy. Obvykle tato hodnota je 0,1 [14].



Obr. 13: Přenosová funkce Leaky ReLU [29]

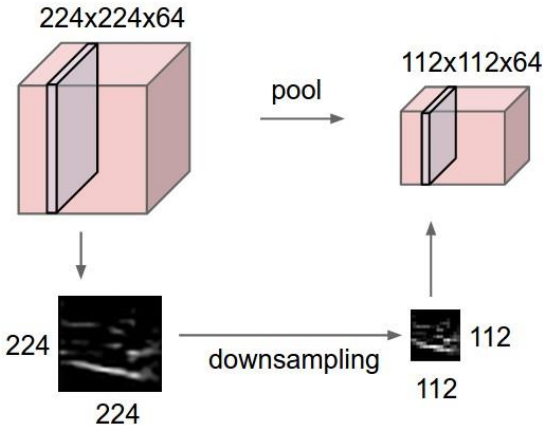
TanH

Funkce TanH je nelineární funkcí, což umožňuje ukládat vrstvy. Pro TanH je gradient silnější než u funkce Sigmoid. Stejně tak jak u funkce Sigmoid i TanH má problém s mizejícím gradientem, ale ve funkci TanH je optimalizací funkce Sigmoid, a proto vyřešení tohoto problému je snazší. V praxi se preferuje funkce TanH [16].

Obr. 14: Přenosová funkce TanH [29]

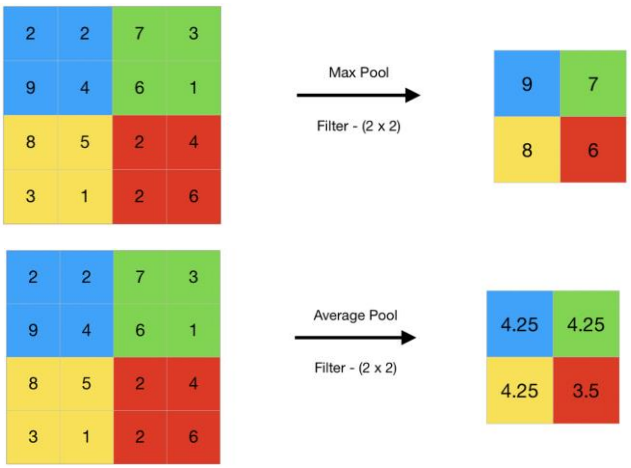
4.1.4 Sdružující vrstva

Operace sdružující vrstvy (Pooling Layer) zahrnuje posouvání dvourozměrného filtru přes každý kanál mapy funkcí a shrnutí prvků v oblasti pokryté filtrem. Jinak řečeno se sdružující vrstva využívá ke zmenšování prostorových rozměrů nezávisle na hloubce vstupu. Důvodem využívání sdružovacích vrstev je zredukování prostorových informací, čímž se docílí zvýšení výpočetního výkonu. Díky menšímu počtu informací také snížíme šanci přeučení modelu.



Obr. 15: Princip sdružovací vrstvy [30]

Sdružující vrstvy jsou rozděleny do dvou typů. Prvním typem je Maximální sdružování (Max Pooling), při kterém se vybere maximální prvek z oblasti mapy prvků pokryté filtrem. Výstupem tedy je mapa funkcí, která obsahuje pouze nejdůležitější vlastnosti předchozích map (viz. obr. 16). Druhým typem je Průměrné sdružování (Average Pooling), které z oblasti mapy pokryté filtrem vypočítá průměrnou hodnotu prvků (viz. obr. 16).



Obr. 16: Maximální sdružování-Max Pooling (nahoře),
Průměrné sdružování-Average Pooling (dole) [31]

4.1.5 Normalizační vrstva

Normalizační vrstva (Batch Normalization) slouží k transformaci vstupních dat a škálování aktivací. Lze použít v případech, kdy jsou hodnoty funkcí příliš vzdálené. Jejím normalizováním zvětšíme výpočetní výkon a zrychlíme samotný proces tréninku sítě. Princip normalizování spočívá v normalizaci výstupu předchozí aktivační vrstvy [17].

4.1.6 Plně propojené vrstvy

Plně propojené vrstvy (Fully Connected Layers) jsou nezbytnou součástí konvolučních neuronových sítí. Cílem plně propojené vrstvy je zpracování výsledku konvolučních vrstev v jediném vektoru. Plně propojené vrstvy v konvoluční neuronové síti pracují stejně jako v neuronových sítích (viz. kapitola 3) s tím rozdílem, že vstupy jsou vytvořené předchozími fázemi konvoluční neuronové sítě. To tedy znamená, že každý z neuronů v průběhu tréninku dokáže zachytit charakteristické prvky vstupních dat, které mu umožní správnou predikci [18].

5 STRUKTURA PROGRAMU

Praktická část této práce je program složený z několika částí. Ne všechny moduly jsou stěžejní pro tuto práci (slouží pouze pro vytvoření grafického uživatelského rozhraní). Proto zde bude kladen důraz pouze na ty, které řeší otázku počítačového vidění a klasifikace čísel. Hlavní struktura programu je postavena na podpůrné knihovně Tensorflow, která obsahuje algoritmy strojového učení. Moduly jsou mezi sebou propojeny předávanými daty, které jsou nezbytné pro finální klasifikaci obrazu.

Praktická část je složena ze dvou aplikací (test1, test2) zabývajících se problémem klasifikace a detekce ručně psaných číslic. Tyto dvě aplikace jsou komplexně shrnuty do třetí aplikace (test3). Z tohoto důvodu budou v této kapitole popsány pouze moduly třetí aplikace.

Tento program není finálním řešením převedení množiny čísel do textového formátu, pouze ukazuje princip detekce a klasifikace ručně psaných číslic.

5.1 Detekce stránky

Prvním stěžejním modulem je modul detekce stránky, který je nezbytný pro větší úspěšnost aplikace. Dokáže rozeznat stránku a její pozadí, díky čemuž dokáže vstup oříznout a nechat pouze stránku s psanými číslicemi. Tento modul bude správně fungovat pouze za splnění ideálních podmínek kontrastu mezi stránkou a jejím pozadím. Počítá se s tím, že pokud v praxi bude tento modul využit, bude brán na tuto chybu ohled a požadovaný list bude vždy focen za dobrých světelných podmínek na tmavém podkladu.

5.2 Detekce číslic

Modul detekce číslic obdrží vstupní obraz již předzpracovaný modulem prvním. Úkolem je detekovat číslice na základě triviální metody počítačového vidění. Tento proces lze vyřešit i efektivnější cestou aplikováním speciálního typu neuronové sítě. Modul dokáže vyhledat číslice na základě prahování a následném nalezení všech kontur. Toto řešení může být v praxi nedostačující, protože v případě, kdy je číslice zapsána tak, že se některé z jejich čar neprotínají, modul je bere jako dvě odlišné číslice.

5.3 Normalizace obrazu

Modul pro normalizaci obrazu slouží k úpravě obrazu tak, aby námi natrénovaná konvoluční síť obdržela ke klasifikaci vstupní obraz ve stejném formátu, v jakém byla trénovací data v průběhu učení konvoluční sítě. Rozdíly, které tento modul normalizuje, jsou ve velikosti obrazu pro klasifikaci, jelikož každé číslo může být napsáno jinou velikostí. Proto jsou všechny obrazy zmenšeny na velikost 28x28 pixelů. Další odchylka může nastat v hodnotách pixelů. Jelikož pro trénink byly pixely převedeny do intervalu od 0 do 1, pro lepší výpočetní výkon se musí převést taktéž při klasifikaci.

5.4 Klasifikace číslic

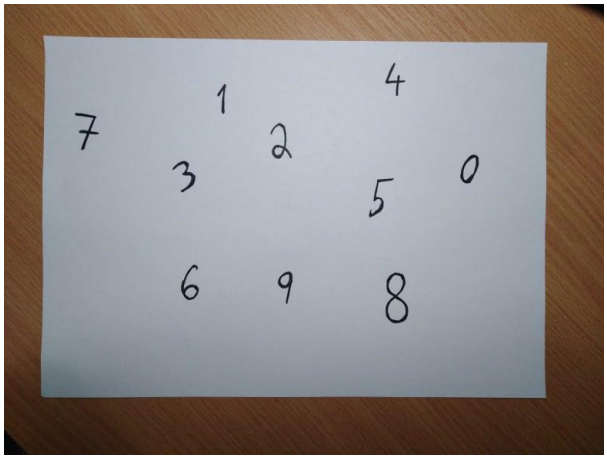
Finální fází celého procesu je modul klasifikace číslic, který rozliší jednotlivé číslice již zpracovaného vstupního obrazu pomocí předtrénované konvoluční sítě. Tento modul dokáže klasifikovat obrázky do deseti různých kategorií. Jedná se o kategorickou klasifikaci a ne binární, takže je zapotřebí velké množství trénovacích dat.

6 TECHNICKÉ ZPRACOVÁNÍ

V této kapitole je popsán postup zpracování 3. části praktické přílohy, jelikož je nejobsáhlejší a vyskytují se v ní funkce z předcházejících částí práce.

6.1 Detekce stránky

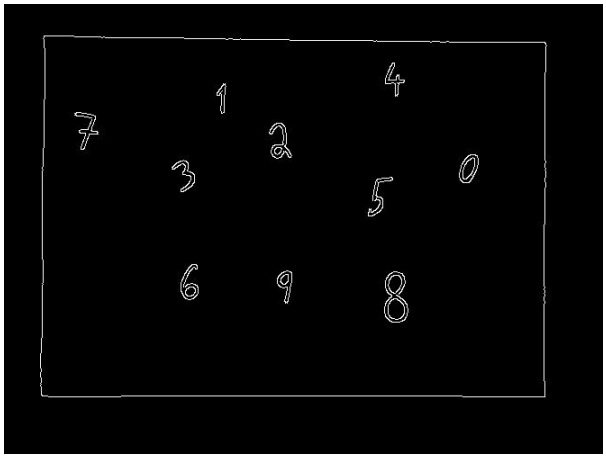
Metoda pro detekci stránky je nezbytná pro další detekci číslic na papíru. Cílem tohoto modulu je najít rohové body papíru, ze kterých se následně vyřízne detekovaná stránka a z ní bude nadále program čerpat. Předpokladem pro úspěšné fungování tohoto modelu je kontrast mezi papírem a jeho podkladem a také viditelnost všech čtyř rohů papíru.



Obr. 17: Originální fotka

6.1.1 Detekce hran

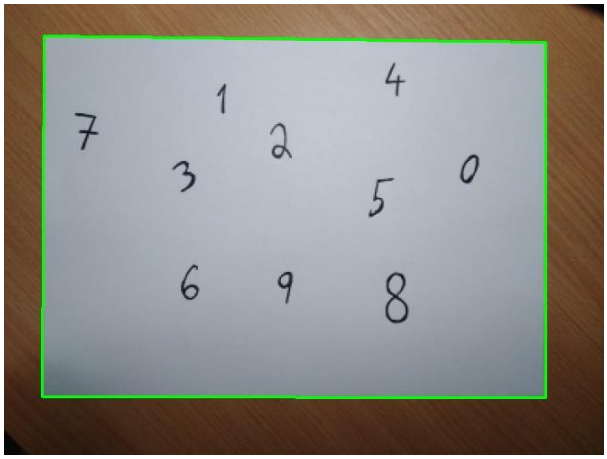
Pro detekci hran je využito Cannyho detekce hran (funkce *cv2.Canny*). Jelikož je Cannyho detekce citlivá na šum, prvním krokem musí být odstranění šumu v obraze (funkce *cv2.GaussianBlur*) čímž se také předejde následné detekci méně významných hran [19].



Obr. 18: Cannyho detekce hran

6.1.2 Nalezení kontur

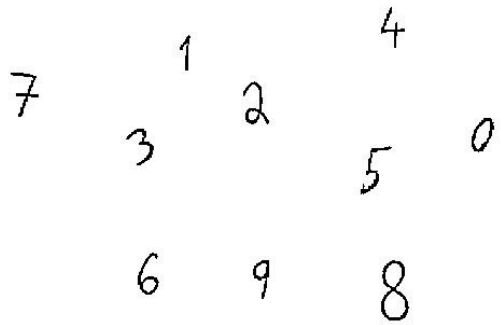
Tato část modulu plní funkcí nalezení výsledných 4 hran strany. K tomuto účelu je aplikována funkce pro hledání kontur (funkce *cv2.findContours*). Platí, že kontura je pouze nepřerušená uzavřená hrana. Po nalezení všech kontur vybereme největší konvexní konturu, která má při jisté míře aproximace čtyři rohy.



Obr. 19: Detekované hrany

6.1.3 Vyříznutí stránky

Pro vytvoření výřezu stránky musíme z nalezených 4 bodů (rohů) vytvořit obraz z ptačí perspektivy (anglicky „birds eye view“). Po vyříznutí obrazu musí být aplikována funkce převedení vstupního obrazu z jednoho barevného prostoru do druhého [20].



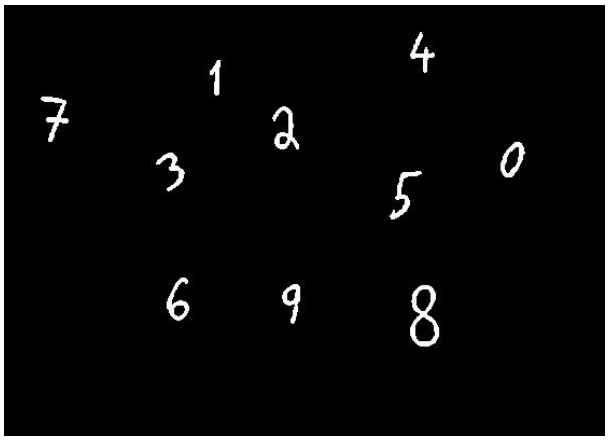
Obr. 20: Vyříznutá stránka

6.2 Detekce číslic

Metoda pro detekci číslic je stěžejní pro výslednou predikci. Cílem tohoto modulu je najít kontury číslic, ze kterých se vypočítá pravoúhlý ohraničující čtyřúhelník, díky kterému lze předat natrénovanému modelu pro klasifikaci pouze vystřižený kus obsahující jen jedno číslo.

6.2.1 Prahování

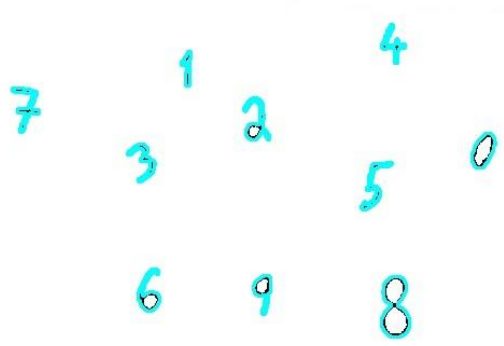
Pro detekci číslic je nezbytná funkce prahování (funkce *cv2.threshold*), které předchází vyhlazení obrázku pomocí Gaussianova filtru pro zamezení detekování nevýznamných rysů.



Obr. 21: Prahování

6.2.2 Nalezení kontur

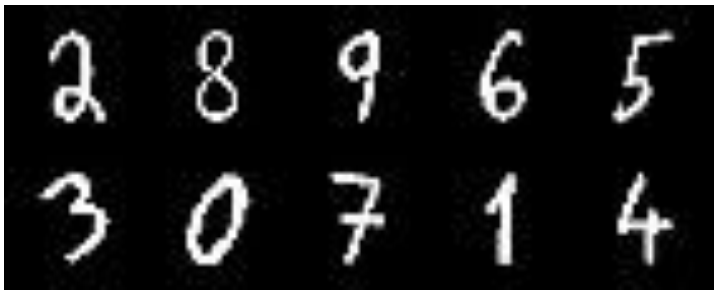
Tato část modulu vyhledá na obrázku všechny kontury (hranice se stejnou intenzitou) díky funkci počítačového vidění (funkce *cv2.findContours*) [21].



Obr. 22: Detekce kontur (modře)

6.2.3 Vyříznutí číslic

Pro vyříznutí je potřeba mít nalezené kontury, které jsme získali v předchozím modulu. Díky funkci počítačového vidění (funkce *cv2.boundingRect*), která dokáže vypočítat pravoúhlý ohraničující čtyřúhelník. Ten vystřihneme.

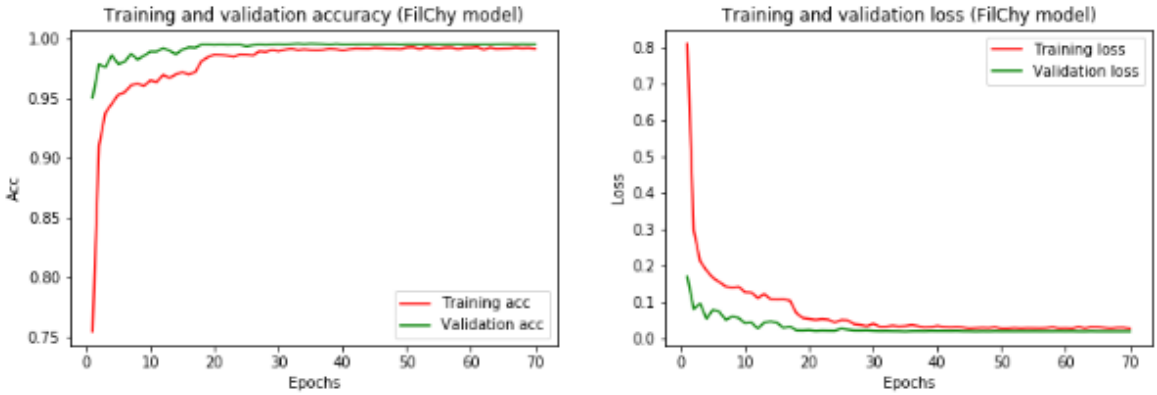


Obr. 23: Vystřižené číslice z obrázku

6.3 Klasifikace číslic

Ještě před aplikací modelu pro klasifikaci číslic převedeme detekovaná písmena na společnou velikosti 28x28. Po klasifikaci všech písmen jsou ke každému detekovanému číslu vypsány jejich hodnoty do levého horního rohu jejich ohraničení.

Pro klasifikaci je využit mnou vytvořený CNN model (*filchy model*), skládající se z bloků obsahujících konvoluční vrstvy, sdružující vrstvy, normalizační vrstvy a regulující vrstvy. Klasifikaci obstarává plně propojená neuronová síť. Výstup neuronové sítě je matice 10x1 obsahující procentuální šanci pro každé číslo desítkové soustavy. Z této matice je vybráno číslo s největší pravděpodobností.



Obr. 24: Graf úspěšnosti (vlevo) a graf ztráty (vpravo)

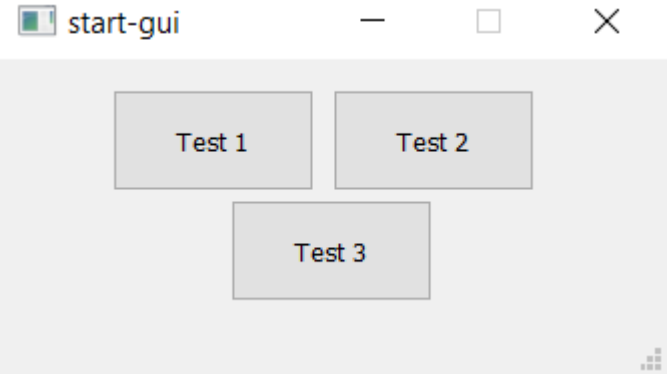
Síť byla natrénována na datové sadě obsahující 65000 příkladů, z čehož 60000 příkladů je určeno pro trénování a 5000 příkladů je určeno pro testování. Z grafu je zřejmé, že při trénovacím procesu nedošlo k žádnému přeučení, nebo naopak nedostatečnému naučení. Model při trénování dosáhl validační úspěšnosti 99.7 %.

7 GUI

GUI (anglicky Graphical User Interface) využívá knihoven PyQt5. GUI tohoto projektu se skládá z hlavního menu a tří částí.

7.1 Hlavní stránka

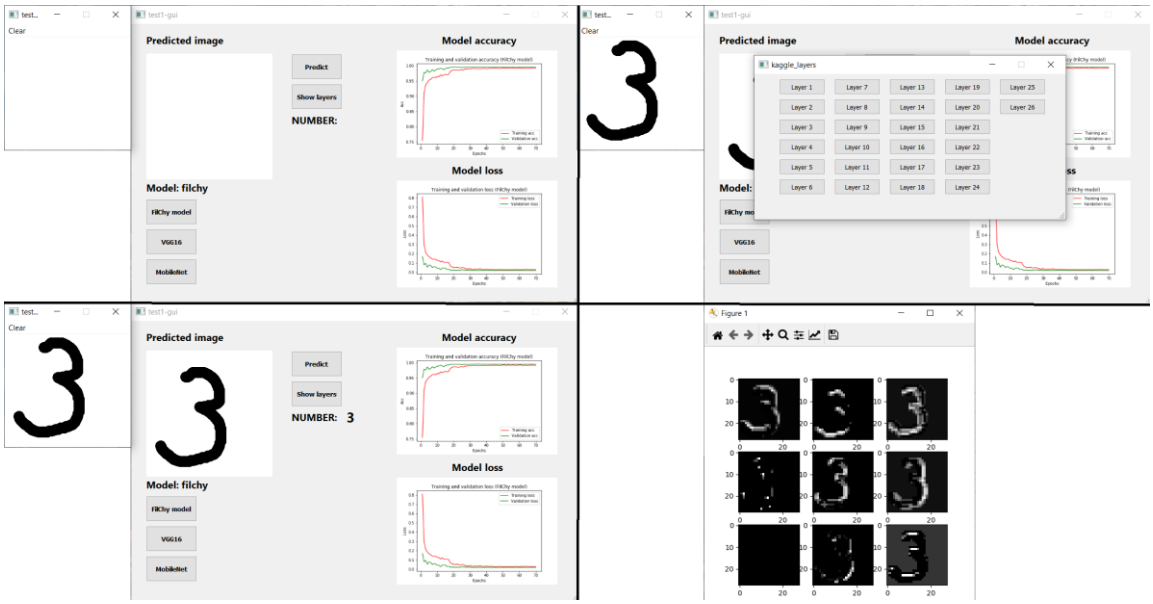
Hlavní stránka (*start-gui*) poskytuje rozhraní pro výběr jedné ze tří implementací CNN. Umožňuje nezávisle přepínat mezi nimi.



Obr. 25: Hlavní stránka aplikace

7.2 Test1

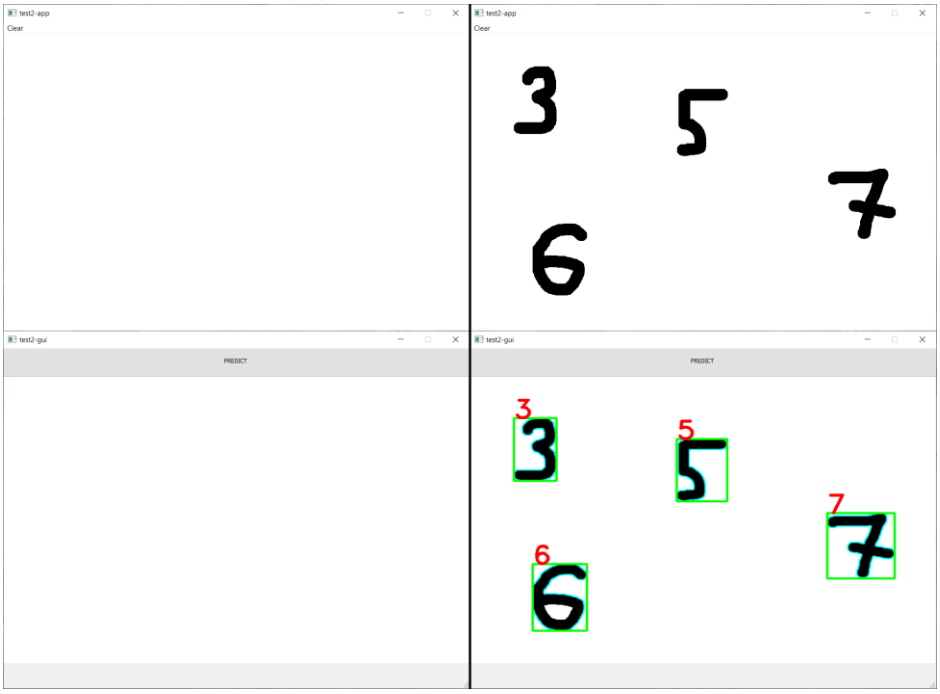
Tato stránka umožňuje uživateli nakreslit libovolné číslo a následně klasifikovat jedním ze tří modelů (modely: *filchy*, *vgg16*, *mobilenet*) jeho hodnotu. Dále tato stránka také umožňuje zobrazit jednotlivé vrstvy každého modelu.



Obr. 26: Prázdná aplikace (vlevo nahoře), aplikace predikuje číslo (vlevo dole), rozhraní pro zobrazení vrstev (vpravo nahoře), zobrazena druhá konvoluční vrstva modelu (vpravo dole)

7.3 Test2

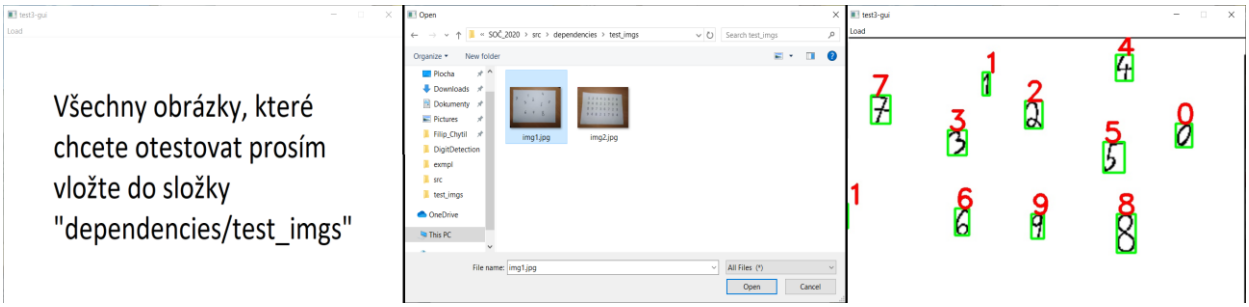
Tato stránka umožňuje uživateli napsat libovolný počet číslic do interaktivního okna, které může následně nechat celé klasifikovat v okně druhém.



Obr. 27: Před vyplněním programu (levý sloupec), po vyplnění a klasifikaci programu (pravý sloupec)

7.4 Test3

Tato část programu se nejvíce přibližuje aplikování CNN v praxi. Tato aplikace umožňuje načíst libovolnou fotku s papírem popsaným čísly, který program zpracuje a provede výslednou klasifikaci všech čísel na papíru.



Obr. 28: Nevyplněný program (levý sloupec), výběr fotografie pro klasifikaci (prostřední sloupec), vyplněný program (pravý sloupec)

8 ZÁVĚR

V této práci byl vytvořen fungující program, jehož úspěšnost činila při detekci a klasifikaci ručně psaných číslic z fotografie 96 % (měřeno ze 100 mnou ručně psaných číslic) a při klasifikace ručně psaných číslic testovacích dat MNIST handwritten digit database 99,7 % (měřeno z 5000 ručně psaných číslic). Tento model byl natrénován na datové sadě obsahujících 60000 příkladů, což je poměrně velká datová sada. Program by měl být schopen rozeznat rukopisy velkého množství lidí.

Jelikož program nedostává žádnou zpětnou vazbu, dalším krokem by mělo být vytvoření kontroly od samotného uživatele, který by při nalezení chybné klasifikace chybu nahlásil a určil správnou hodnotu. Následně by se obrázek přidal do datové sady a po dosažení určitého počtu chybných klasifikací by došlo k vytrénování nového modelu na datové množině obsahují již i chybné klasifikace minulého modelu.

Součástí práce bylo vytvoření vlastního konvolučního modelu, který byl také aplikován v mezinárodní soutěži Kaggle Kannada MNIST, ve kterém se umístil na 60. – 63. místě z celkových 1212 účastníků s úspěšností 99,1 % ze znaků, které nikdy předtím neviděl. V této soutěži se model musel naučit rozpoznávat jazyk Kanarese (Kannada) nikoli desítkovou soustavu číslic. Celkový počet různých typů písmen k naučení byl taktéž roven deseti.

Kromě zpětné vazby programu je také možné zvýšit jeho efektivitu implementováním R-CNN (Region CNN) určeným pro detekci objektů namísto primitivního řešení použitého v tomto programu, který vyhledává všechny kontury na obrázku. Tímto způsobem by bylo možné se vyhnout detekci stránky, jelikož R-CNN nepracuje na principu kontur jako tento program.

9 POUŽITÁ LITERATURA

[1] CULURCIELLO, Eugenio. THE HISTORY OF NEURAL NETWORKS. *Dataconomy* [online]. 2017 [cit. 2020-03-08]. Dostupné z: https://dataconomy.com/2017/04/history-neural-networks/

[2] HÁJEK, Břetislav. *Středoškolská odborná činnost: Optické rozpoznávání ručně psaného textu* [online]. 2017, 30 [cit. 2020-03-08]. Dostupné z: https://socv2.nidv.cz/archiv39/getWork/hash/a01b90e6-ea1b-11e6-9404-005056bd6e49

[3] DURČÁK, Pavel. *NaPočítači: Neuronové sítě a princip jejich fungování* [online]. 8.9.2017 [cit. 2020-03-08]. Dostupné z: https://www.napocitaci.cz/33/neuronove-site-a-princip-jejich-fungovani-uniqueidgOkE4NvrWuNY54vrLeM670eFNQh552VdDDulZX7UDBY/

[4] LIANG, Ming a Xiaolin HU. *Recurrent Convolutional Neural Network for Object Recognition* [online]., 9 [cit. 2020-03-08]. Dostupné z: http://openaccess.thecvf.com/content_cvpr_2015/papers/Liang_Recurrent_Convolutional_Neural_2015_CVPR_paper.pdf

[5] Umělá neuronová síť: Model umělého neuronu. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2020-03-08]. Dostupné z: https://cs.wikipedia.org/wiki/Um%C4%9Bl%C3%A1_neuronov%C3%A1_s%C3%AD%C5%A5

[6] *Neuronové sítě* [online]. In: [cit. 2020-03-08]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=21471

[7] *Umělá inteligence: Principy učení neuronu obecně* [online]. In: [cit. 2020-03-08]. Dostupné z: https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--umela-inteligence--neuronove-site-jednotlivy-neuron--adaptacni-dynamika-neuronu--principy-uceni-neuronu-obecne

[8] Unsupervised learning. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2020-03-08]. Dostupné z: https://en.wikipedia.org/wiki/Unsupervised_learning

[9] *Umělá inteligence: Učení s učitelem* [online]. In: [cit. 2020-03-08]. Dostupné z: https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--umela-inteligence--neuronove-site-jednotlivy-neuron--adaptacni-dynamika-neuronu--uceni-s-ucitelem

[10] Convolutional neural network. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2020-03-08]. Dostupné z: https://en.wikipedia.org/wiki/Convolutional_neural_network

[11] KE, Qiuhong a Farid BOUSSAID. *Computer Vision for Human–Machine Interaction: Convolutional Layers* [online]. 2018 [cit. 2020-03-08]. Dostupné z: https://www.sciencedirect.com/topics/engineering/convolutional-layer

[12] Kernel (image processing): Convolution. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2020-03-08]. Dostupné z: https://en.wikipedia.org/wiki/Kernel_(image_processing)#Convolution

[13] KRAJČOVIČOVÁ, Bc. Mária. *Konvoluční neuronová síť pro zpracování obrazu* [online]. Brno, 2015 [cit. 2020-03-08]. Dostupné z: https://core.ac.uk/download/pdf/30299309.pdf . Diplomová práce. Vysoké učení technické v Brně. Vedoucí práce doc. Ing. Radim Burget, Ph.D.

[14] Activation Functions in Neural Networks: Sigmoid, TanH, Softmax, ReLU, Leaky ReLU EXPLAINED!!! [online]. 2017 [cit. 2020-03-08]. Dostupné z: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[15] GARG, Nikhil. *What is the vanishing gradient problem?* [online]. [cit. 2020-03-08]. Dostupné z: https://www.quora.com/What-is-the-vanishing-gradient-problem

[16] Complete Guide of Activation Functions: A practical guide comparing advanteges, problems, and solutions of Activation functions [online]. 2017 [cit. 2020-03-08]. Dostupné z: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[17] IOFFE, Sergey a Christian SZEGEDY. *Batch Normalization: Accelerating Deep Network Training b y Reducing Internal Covariate Shift* [online]., 11 [cit. 2020-03-08]. Dostupné z: https://arxiv.org/pdf/1502.03167v3.pdf

[18] SIDDIQUI, Kashif Ali. What happens in the fully connected layer in a convolutional neural network? *Quora* [online]. [cit. 2020-03-08]. Dostupné z: https://www.quora.com/What-happens-in-the-fully-connected-layer-in-a-convolutional-neural-network

[19] Canny Edge Detection: Canny Edge Detection in OpenCV. *OpenCV: Python tutorials* [online]. [cit. 2020-03-08]. Dostupné z: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html#canny-edge-detection-in-opencv

[20] Miscellaneous Image Transformations: cvtColor. *OpenCV: Documentation* [online]. [cit. 2020-03-08]. Dostupné z: https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html#cvtcolor

[21] Structures Analysis and Shape Descriptors: findContours. *OpenCV: Documentation* [online]. [cit. 2020-03-08]. Dostupné z: https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=boundingrect#findcontours

[22] Using neural nets to recognize handwritten digits: The architecture of neural networks. In: *Http://neuralnetworksanddeeplearning.com/chap1.html* [online]. [cit. 2020-03-08]. Dostupné z: http://neuralnetworksanddeeplearning.com/chap1.html

[23] Practical Convolutional Neural Networks: Building blocks of a neural network. In: *Https://subscription.packtpub.com/book/data/9781788392303/1/ch01lvl1sec10/building-blocks-of-a-neural-network* [online]. [cit. 2020-03-08]. Dostupné z: https://subscription.packtpub.com/book/data/9781788392303/1/ch01lvl1sec10/building-blocks-of-a-neural-network

[24] Activation function: Functions. In: *Https://en.wikipedia.org/wiki/Activation_function* [online]. [cit. 2020-03-08]. Dostupné z: https://en.wikipedia.org/wiki/Activation_function

[25] Mnist clustering. In: *Https://www.google.com/search?q=mnist+clustering&sxsrf=ACYBGNSUkzxhafu3LZ7emxnqsuUIfulTgg:1581257446756&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiq9KWD08TnAhUKUcAKHYIHAAAQ_AUoAXoECAsQAw&biw=1536&bih=722#imgrc=zafuXr8-XofDQM* [online]. [cit. 2020-03-08]. Dostupné z: https://www.google.com/search?q=mnist+clustering&sxsrf=ACYBGNSUkzxhafu3LZ7emxnqsuUIfulTgg:1581257446756&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiq9KWD08TnAhUKUcAKHYIHAAAQ_AUoAXoECAsQAw&biw=1536&bih=722#imgrc=zafuXr8-XofDQM

[26] MNIST Handwritten Digits Classification using a Convolutional Neural Network (CNN): Defining the Model. In: *Https://towardsdatascience.com/mnist-handwritten-digits-classification-using-a-convolutional-neural-network-cnn-af5fafbc35e9* [online]. [cit. 2020-03-08]. Dostupné z: https://towardsdatascience.com/mnist-handwritten-digits-classification-using-a-convolutional-neural-network-cnn-af5fafbc35e9

[27] Convolutional Neural Networks from the ground up: How Convolutional Neural Networks learn. In: *Https://towardsdatascience.com/convolutional-neural-networks-from-the-ground-up-c67bb41454e1* [online]. [cit. 2020-03-08]. Dostupné z: https://towardsdatascience.com/convolutional-neural-networks-from-the-ground-up-c67bb41454e1

[28] CHEN, Ting-Hao. What is “padding” in Convolutional Neural Network? In: *Https://medium.com/machine-learning-algorithms/what-is-padding-in-convolutional-neural-network-c120077469cc* [online]. 2017 [cit. 2020-03-08]. Dostupné z: https://medium.com/machine-learning-algorithms/what-is-padding-in-convolutional-neural-network-c120077469cc

[29] UDOFIA, Udeme. Basic Overview of Convolutional Neural Network (CNN): Activation Function. In: *Https://medium.com/dataseries/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17* [online]. 2018 [cit. 2020-03-08]. Dostupné z: https://medium.com/dataseries/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17

[30] Convolutional Neural Networks (CNNs / ConvNets): Pooling Layer. In: *Http://cs231n.github.io/convolutional-networks/* [online]. [cit. 2020-03-08]. Dostupné z: http://cs231n.github.io/convolutional-networks/

[31] CNN | Introduction to Pooling Layer: Types of Pooling Layers. In: *Https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/* [online]. [cit. 2020-03-08]. Dostupné z: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

10 SEZNAM OBRÁZKŮ A TABULEK

11 PŘÍLOHA 1: ZDROJOVÝ KÓD

Zdrojový kód lze nalézt na úložišti GitHub: https://github.com/filchy/digit_recognition_gui. Pro spuštění programu je nutný Python 3.5 a tyto knihovny: PyQt5 (verze 5.14.1), Numpy (verze 1.16.5), OpenCV-Python (verze 4.2.0), Matplotlib (verze 3.1.1) a Tensorflow (verze 2.1.0). Při použití jiných verzích těchto knihoven nezaručuji správný chod programu.

Pro spuštění celého programu se nacházejte v příkazovém řádku ve složce **src** a spusťte hlavní soubor **main.py** příkazem *python main.py*.

Struktura:

Zdrojový kód

1. Hlavní soubor – main.py
2. Detekce stránky – img_procces.py/find_roi
3. Detekce číslic – img_procces.py/roi_img_predict
4. Modely:
   4.1. FilChy – filchy_model.py
   4.2. VGG16 – vgg16_model.py
   4.3. MobileNet – mobilenet_model.py
5. Vizualizace vrstev modelů:
   5.1. FilChy vizualizace – filchy_visualizer.py
   5.2. VGG16 vizualizace – vgg16_visualizer.py
   5.3. MobileNet vizualizace – mobilenet_visualizer.py
6. Zpracování dat – load_data.py
7. GUI:
   7.1. Start GUI – main.py
   7.2. Test 1 – test1.py
   7.3. Test 2 – test2.py
   7.4. Test 3 – test3.py
8. Malovací aplikace:
   8.1. Test 1 malování – test1_app.py
   8.2. Test 2 malování – test2_app.py

Soubory

dependencies/ - hlavní složka
dependencies/images – průběžné ukládání obrázků ke klasifikaci
dependencies/models – uložené předtrénované modely
dependencies/test_imgs – vkládání fotek pro test 3
dependencies/graphs – uložené grafy průběhu trénování