Seminární práce

Neuronové sítě a strojové učení pro rozpoznávání obrazu

Tomáš Pražák

Gymnázium Jaroslava Seiferta

Septima 2023/24

Vedoucí práce: Mgr. Adam Dominec

Obsah

Anotace

Tato seminární práce vysvětluje koncepty neuronových sítí a konvolučních neuronových sítí v kontextu zpracovávání a rozpoznávání obrazu. Zároveň také ukazuje jejich reálnou aplikaci implementací neuronové sítě pro populární datasety MNIST a Fashion-MNIST za pomoci Pythonu a knihoven pro lineární algebru.

Annotation

This seminary work explains the underlying concepts of artificial neural networks and convolutional neural networks in the context of computer vision. It demonstrates their application by implementing a computer vision neural network on the popular MNIST and Fashion-MNIST datasets completely from scratch only using Python and libraries for linear algebra.

Pojmy a zkratky

AI - zkratka anglického *"Artificial Inteligence"*; doslova umělá inteligence; "schopnost strojů napodobovat lidské schopnosti, jako je uvažování, učení se, plánování nebo kreativita." (EP 2020)

CT-sken - snímek počítačové tomografie

Dataset - strukturovaný soubor dat, může být například ve formě obrázků s popisky nebo třeba ve formě tabulárních dat

Gradient - hodnota vyjadřující okamžitý "sklon" funkce, vypočítáváno při back-propagaci neurnovou sítí pro každý parametr, umožňuje optimalizaci sítě

Knihovna - software s konkrétním účelem volně dostupný na internetu, použitelný v rámci konkrétního programovacího jazyku

Konvoluční neuronová síť (**CNN)** - architektura neuronových sítí využívající konvolučních vrstev, vhodná hlavně pro obrazová data (viz kapitola 3)

Metaparametr sítě - parametry sítě, které určuje programátor a které se neupravují v průběhu učení; často také *hyper-parametr*; např. rychlost učení, počet neuronů ve vrstvách (Brownlee 2019b)

ML - zkratka anglického "*Machine Learning"*; doslova "strojové učení"; idea, že počítače mají možnost se naučit opakující se vzorce z dat

MNIST - jednoduchý dataset obrázků ručně psaných jednociferných čísel (viz 5.1) (LeCun 1998)

Neuronová síť (NS) - struktura strojového učení napodobující biologický mozek pomocí sítě vzájemně interagujících neuronů rozmístěných ve vrstvách (IBM 2024b); vysvětleno v kapitole 2.1

One-hot encoding - způsob značení používaný pro cílové hodnoty výstupu klasifikační sítě; cílový výstup sítě je reprezentován vektorem o délce rovné počtu klasifikačních tříd, všechny hodnoty vektoru jsou nula až na ty, které korespondují se správnými klasifikačními třídami a mají tak hodnotu jedna

Overfitting - problém při využívání neuronových sítí, kdy si neuronová síť zapamatuje trénovací dataset a má na něm vysokou úspěšnost, ale nenaučí se zobecňovat a má tak špatnou úspěšnost na testovacím datasetu

Rozpoznávání obrazu - oblast aplikace strojového učení; model je schopen na obrázku označit klíčové oblasti jako například obličeje nebo rozhodnout, co se na něm nachází

Tenzor/vektor - pro účely této seminární práce - n-dimenzionální seznam čísel

YOLO (You Only Look Once) - architektura modelu využívaná pro rozpoznávání obrazu, využívá CNN, celé rozpoznávání však probíhá pomocí jednoho modelu, to umožňuje výrazně rychlejší rozpoznávání (Wu 2024)

Úvod

Umělá inteligence (AI) se v posledním roce dostala do popředí v médiích v kontextu velkých jazykových modelů. Ty mírně odsunuly do ústraní modely strojového učení (ML) pro rozpoznávání obrazu. V této práci se věnuji neuronovým sítím a matematice, na nichž ML a AI stojí. Tato práce se věnuje specificky oblasti rozpoznávání obrazu, která lidstvu otevírá nové možnosti například v podobě autonomních vozidel, ale také v medicíně pro efektivnější prevenci rakoviny a jiných závažných onemocnění (Hunter 2022).

Cíl práce

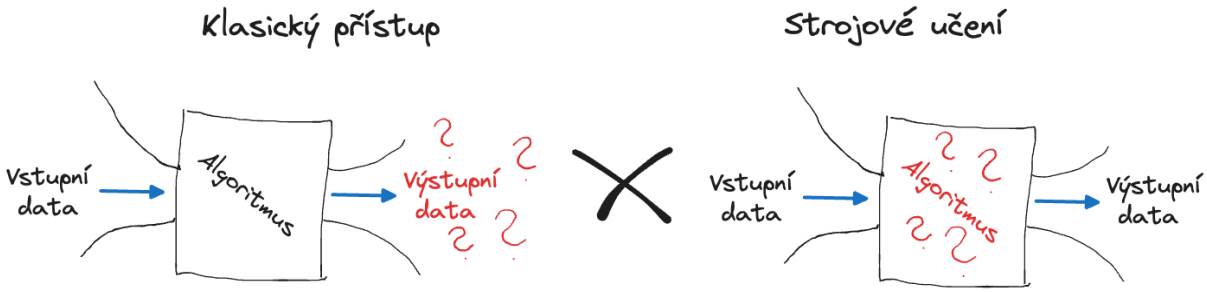
Touto prací vysvětluji zásadní koncepty stojící za ML a rozpoznáváním obrazu. Zároveň bych rád alespoň částečně demystifikoval tuto oblast vědy, která je často vnímána jako něco relativně záhadného a magicky fungujícího. Celkovou jednoduchost tématu neuronových sítí nakonec demonstruji naprogramováním vlastní neuronové sítě sloužící k rozpoznávání objektů z obrázků. Síť bych rád naprograoval pouze s pomocí komputačních knihoven Numpy a Scipy a bez již existujících ML-knihoven jako TensorFlow nebo PyTorch. Ty celý proces vytváření modelů strojového učení ještě velice zjednodušují a zpřístupňují.

1 Strojové učení vs. klasický způsob řešení

1.1 Klasický způsob řešení

Problémy v programování lze typicky strukturovat tak, že ze vstupních dat pomocí předem daného algoritmu získáváme požadovaný výstup.

Tento způsob řešení je sice přímočarý, ale má i své nevýhody. Primární nevýhodou je náročnost vytváření efektivních algoritmů pro řešení daného problému. Pro mnoho problémů jsou efektivní algoritmy již často vymyšleny, jako například třídící algoritmy nebo algoritmy pro hledání nejkratší cesty mezi body grafu. Pro komplexnější problémy jako rozpoznávání obrazu může být ale designování algoritmu náročné, ne-li nemožné. V tu chvíli přichází na scénu strojové učení. (Inisightsoftware 2023)



(Nákres 1) Klasický způsob řešení vs. strojové učení

1.2 Od vstupu a výsledku k postupu

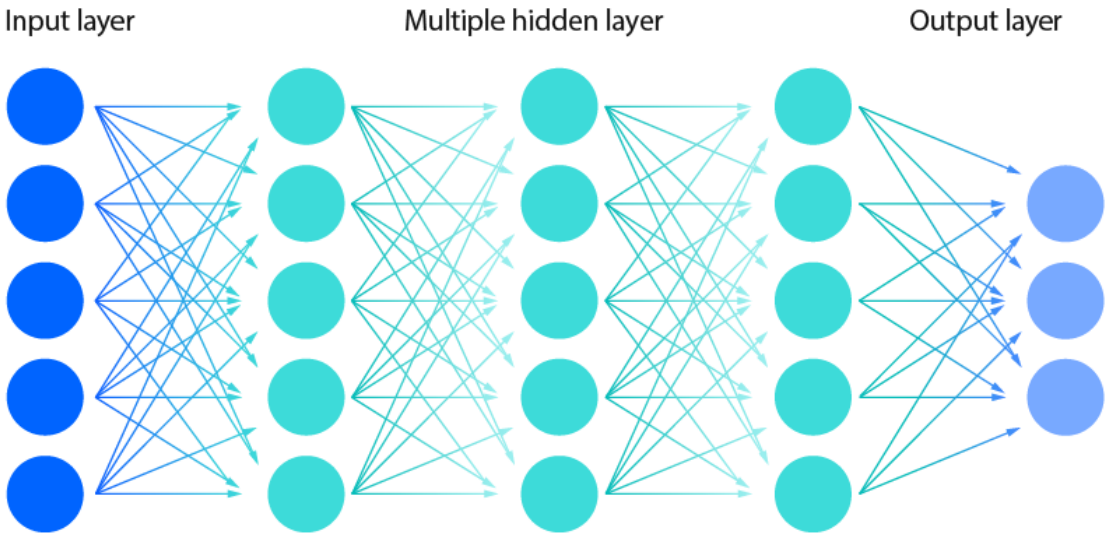
Podobně jako u klasického způsobu máme při strojovém učení vstupní data, ale místo algoritmu na získání výstupu máme také očekávaná výstupní data (viz nákres 1). Toto mohou být například obrázky a popisky toho, co se na obrázcích nachází. Pomocí strojového učení je počítač schopen se sám naučit algoritmus nutný k získání výstupu ze vstupních dat. Algoritmus se tak naučí generalizovat i pro příklady, které ještě nikdy neviděl. (Insightsoftware 2023)

Slabinou strojového učení je nutnost velkého množství dat, z nichž se algoritmus může učit. Jeho aplikace jsou tak limitovány pro situace, kde je možné dobře sbírat data. (Insightsoftware 2023) Strojové učení je ale také efektivní pro aplikace jako hraní šachů nebo jiných her, kdy se algoritmus může učit z velkého množství odehraných her, které klidně může umělá inteligence hrát sama proti sobě. (Silver 2018)

2 Základní principy neuronových sítí

2.1 Neuronová síť

Neuronová síť je architektura modelu používaná při strojovém učení. Na popularitě nabyla především díky svojí schopnosti přizpůsobovat se i komplexním datasetům. Ta je umožněna vysokým množstvím optimalizovatelných parametrů sítě, které vycházejí ze snahy napodobovat biologický mozek pomocí neuronů. (IBM 2024b)

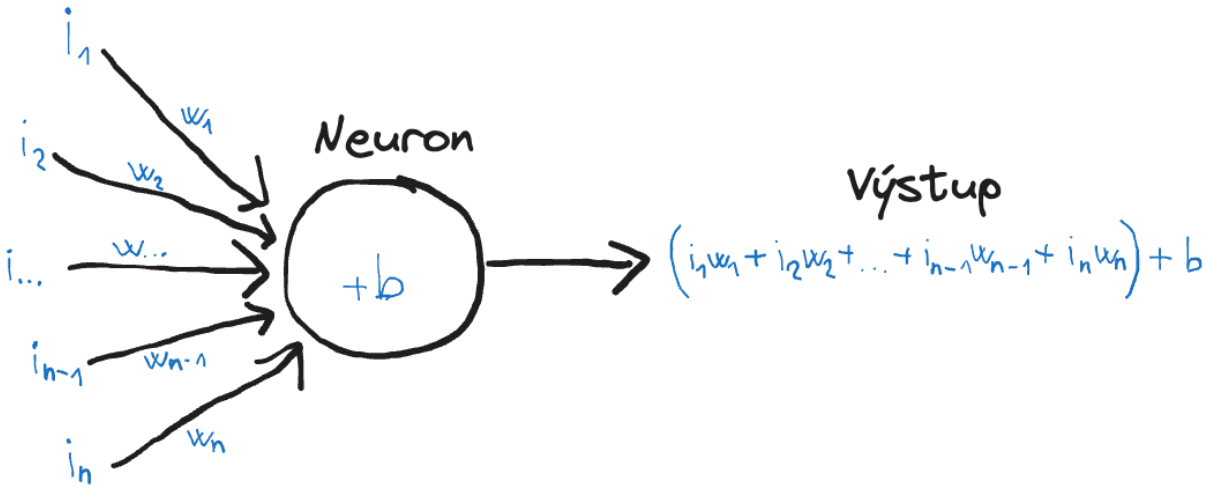


(Obrázek 1) Neuronová síť (IBM 2024a)

Neurony jsou strukturované do vrstev a každý neuron ve vrstvě je spojen s nějakými neurony v předešlé i v následující vrstvě, často i se všemi (viz obrázek 1). Standardní neuronová síť má jednu vstupní vrstvu, nějaký počet skrytých vrstev a jednu výstupní vrstvu. Počet neuronů vstupní vrstvy kopíruje dimenze vstupních dat. Pokud jsou vstupní data obrázky o velikosti 20x20 pixelů, pak vstupní vrstva bude mít 400 neuronů. Počet neuronů ve výstupní vrstvě se odvíjí od toho, co má síť dělat. V případě binární obrazové klasifikace stačí neuron jeden, pokud máme klasifikačních tříd více, závisí počet neuronů na nich. Počty neuronů ve skrytých vrstvách mohou být libovolné a jedná se tzv. metaparametr sítě. (IBM 2024b)

2.2 Neuron

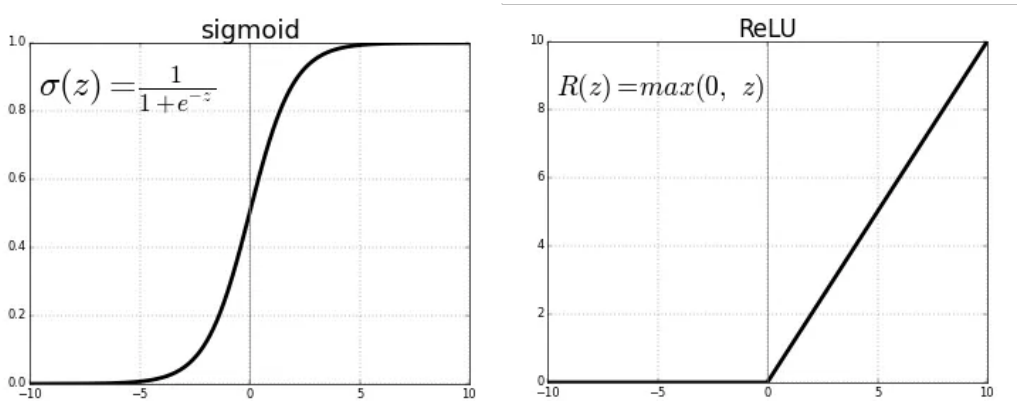
Nejzákladnější jednotkou neuronové sítě je neuron. Podobně jako jeho biologická analogie má i tento neuron nějaký počet vstupů a také výstup, kterým předává informace ostatním neuronům (viz nákres 2). Každý neuron má sadu koeficientů přiřazených ke každému vstupu (anglicky *weights*, proto na nákresu $w_1 \dots w_n$) a také jednu hodnotu navíc, která přímo posouvá výstup neuronu (anglicky *bias*), a to při jakých hodnotách se aktivuje nebo neaktivuje. Tyto parametry model při učení upravuje za účelem získání žádoucího výstupu. V neuronové síti jsou pak neurony řazeny do vrstev.



(Nákres 2) Schematická reprezentace neuronu

2.3 Aktivační funkce

Další důležitou částí neuronové sítě jsou aktivační funkce. Tyto funkce se aplikují na výstup z neuronu a téměř vždy mají prvek nelinearity, který umožňuje celé síti adaptovat se i nelineárním datům. Příkladem takových aktivačních funkcí mohou být funkce sigmoid a ReLU. Sigmoid "zmáčkne" vstup mezi hodnoty 0 a 1. Funkci ReLU pak lze popsat rovnicí $ReLU(x) = \max(0, x)$ (viz obrázek 2) (Zhang 2020).



(Obrázek 2) Funkce ReLU a sigmoid (Medium 2017)

Pro účely klasifikačních sítí je dobré zmínit také aktivační funkci Softmax. Ta se aplikuje na celou výstupní vrstvu klasifikační neuronové sítě a vyjadřuje, jaký podíl na celkovém výstupu sítě má který neuron. To v praxi vypadá tak, že hodnota výstupu každého neuronu je vydělena součtem výstupů všech neuronů v dané vrstvě (Zhang 2020).

2.4 Loss

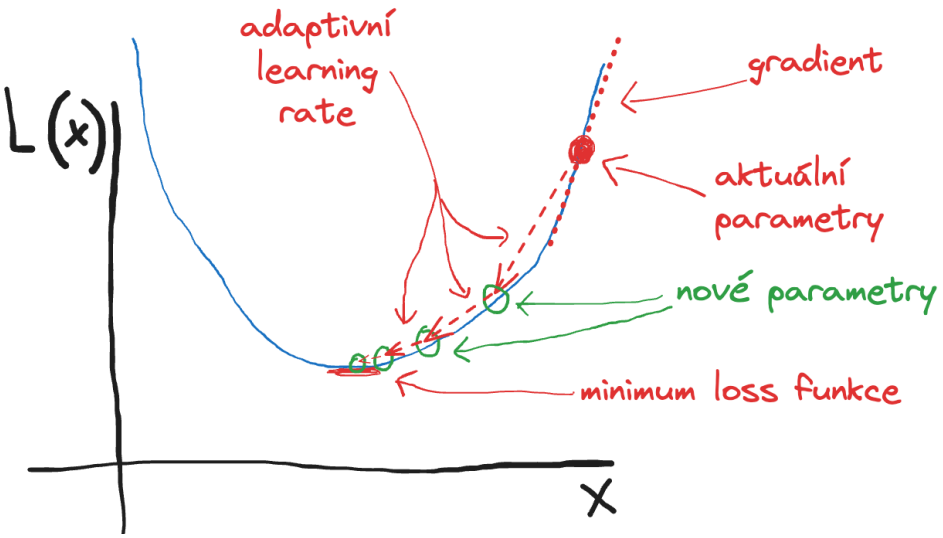
Strojové učení je ve své podstatě optimalizační problém. Je tedy nutné získat hodnotu ukazující, jak dobře nebo špatně si tento model vede. K tomu slouží loss funkce, která na základě výstupů ze sítě a cílových výstupů spočítá celkovou chybovost sítě. Při optimalizaci je pak cílem minimalizovat loss funkci a tak maximalizovat přesnost.

V kontextu klasifikačních sítí je důležitá loss funkce "cross-entropy", která počítá odchylku jedné nebo několika klasifikačních tříd od výstupů ze sítě. Tuto funkci lze popsat rovnicí $CE = -\sum_i^C t_i \log(s_i)$, kde t je vektor obsahující one-hot-encoded správné klasifikační třídy (tj. vektor, jehož délka se rovná počtu klasifikačních tříd, a je definován jako $t_i = 1$ pro správné klasifikační třídy a $t_i = 0$ pro ty špatné) Dále s je vektor výstupů ze sítě, C je pak počet klasifikačních tříd a subskript i označuje konkrétní prvek vektoru. (Brownlee 2019a)

2.5 Optimalizační algoritmy

V procesu back-propagace (vysvětlen v 2.7) získáme pomocí loss funkce gradient vzhledem ke všem parametrům. Optimalizační algoritmy rozhodují jak tyto parametry upravit, aby došlo ke snížení výstupu loss funkce.

Obecně všechny optimalizační algoritmy fungují tak, že v malých krocích sestupují u každého parametru směrem dolů podle gradientu (viz nákres 3). Rychlost tohoto sestupu je určená metaparametrem rychlosti učení (*learning rate*). Různé optimalizační algoritmy pak přidávají další aspekty ve snaze se vyrovnat s častými optimalizačními problémy.



(Nákres 3) Sestup po gradientu (inspirace Raschka 2024)

Adaptivní rychlost učení je způsob, jak se optimalizační algoritmy vyrovnávají s náročností procesu optimalizace. Rychlost učení je průběžně snižována, což modelu umožňuje dělat velké změny v parametrech ze začátku a následně pomocí menších změn najít vhodnou hodnotu parametrů pro minimalizování loss funkce. Větší změny znamenají menší výpočetní náročnost, protože při učení stačí méně iterací, a zefektivňují tak proces učení.

Momentum je metoda užívaná v optimalizačních algoritmech s cílem adaptivně zvyšovat learning rate, aby se funkce loss chovala vzhledem k aktuálním parametrům jednoduše. Optimalizační algoritmus s momentem si udržuje vážený průměr gradientu za několik posledních iterací pro každý parametr, který je pak používán namísto okamžitého gradientu. (Zhang 2020)

2.6 Forward-propagace

Forward-propagace nebo též inference je proces probíhající v neuronové síti za účelem získání nějakého výsledku ze vstupních dat. Tento proces probíhá tak, že vstupní vrstva dostane data, ta jsou násobena koeficienty (*weights*), které korespondují s každou spojnicí mezi dvěma neurony. A všechny tyto hodnoty se pak v neuronu sečtou i s hodnotou *biasu.* Následně je na každý neuron aplikována aktivační funkce nelineárního charakteru. Výstupní hodnota je pak použita jako vstup pro další vrstvu sítě. Data jsou takto sítí propagována dopředu (od toho *forward-propagation*) až do poslední vrstvy sítě. Tam je v případě klasifikačních sítí aplikována aktivační funkce Softmax a z jejího výstupu se vypočítá míra chybovosti sítě, neboli *loss*.

2.7 Back-propagace (Zpětné šíření chyby)

Back-propagace je proces s cílem najít gradient vzhledem ke každému parametru sítě. Ten je následně využit optimalizačním algoritmem pro optimalizaci sítě pro daný dataset.

Proces back-propagace začíná spočítáním gradientu loss funkce vzhledem k aktivaci výstupní vrstvy sítě. Pokud tedy aktivaci poslední vrstvy vyjádříme jako a^L a loss funkci pro příklad n jako C_n, pak počítáme $\frac{\partial C_n}{\partial a^L}$. Pomocí tzv. řetízkového pravidla (pravidlo počítání derivace složených funkcí) jsme pak schopni spočítat derivaci loss funkce vzhledem ke kterémukoliv parametru. (Sanderson 2017)

Například pro spočítání gradientu parametru w_i^L, který má vliv na hodnotu některého neuronu výstupní vrstvy, musíme propagovat zpět od gradientu loss funkce. Postupujeme-li od konce, nachází se v tomto řetězci loss funkce, aktivační funkce a neuron. Součet vstupů pro výstupní neuron je pak přímo ovlivněn parametrem w_i^L. Gradient pro tento parametr by se tedy během back-propagace počítal následovně: $\frac{\partial C_n}{\partial w_0^L} = \frac{\partial C_n}{\partial a^L} \cdot \frac{\partial a^L}{\partial N_n^L} \cdot \frac{\partial N_n^L}{\partial w_i^L}$. Všechny proměnné reprezentují to, co v předešlých příkladech, a N_n^L je součet vstupů pro n-tý neuron ve vrstvě L, tedy poslední vrstvě. (Sanderson 2017)

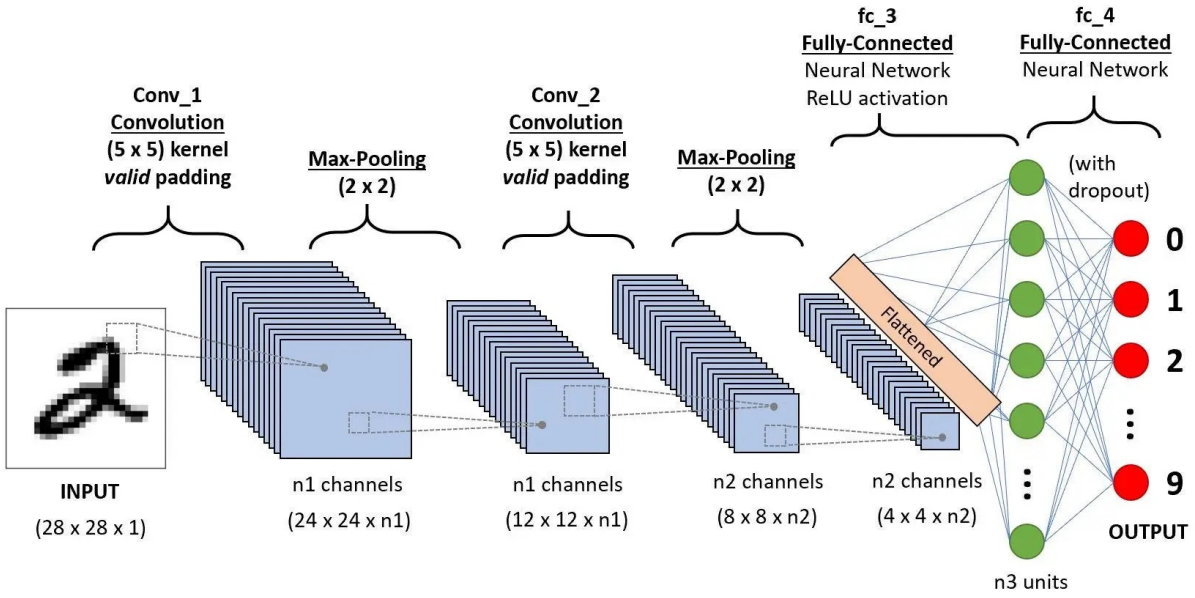
Během back-propagace se takto spočítají gradienty vzhledem ke každému parametru, podle kterých pak optimalizační algoritmus tyto parametry upravuje.

3 Konvoluční neuronové sítě

3.1 Architektura CNN

Pro rozpoznávání obrazu pomocí neuronových sítí se koncem 20. století objevila nová architektura - konvoluční neuronová síť (CNN). Klasická architektura narážela na problém v počtu parametrů, například obrázky o rozlišení pouhých 1 megapixel mají 1 milion pixelů neboli vstupních parametrů. K těm pak korespondují optimalizovatelné parametry alespoň o počtu *počet_vstupů · počet_výstupních_neuronů*, což dělá trénování modelů této komplexity v praxi nepraktické (Zhang 2020).

Zatímco standardní, plně propojená, architektura sítě je schopna naučit se různé typy dat, architektura CNN je vhodná primárně pro obrazová data. Konvoluční sítě mají dva klíčové typy vrstev, konvoluční vrstvy a poolingové (sdružovací) vrstvy (viz obrázek 3) (Zhang 2020).



(Obrázek 3) Konvoluční neuronová síť (Saha 2024)

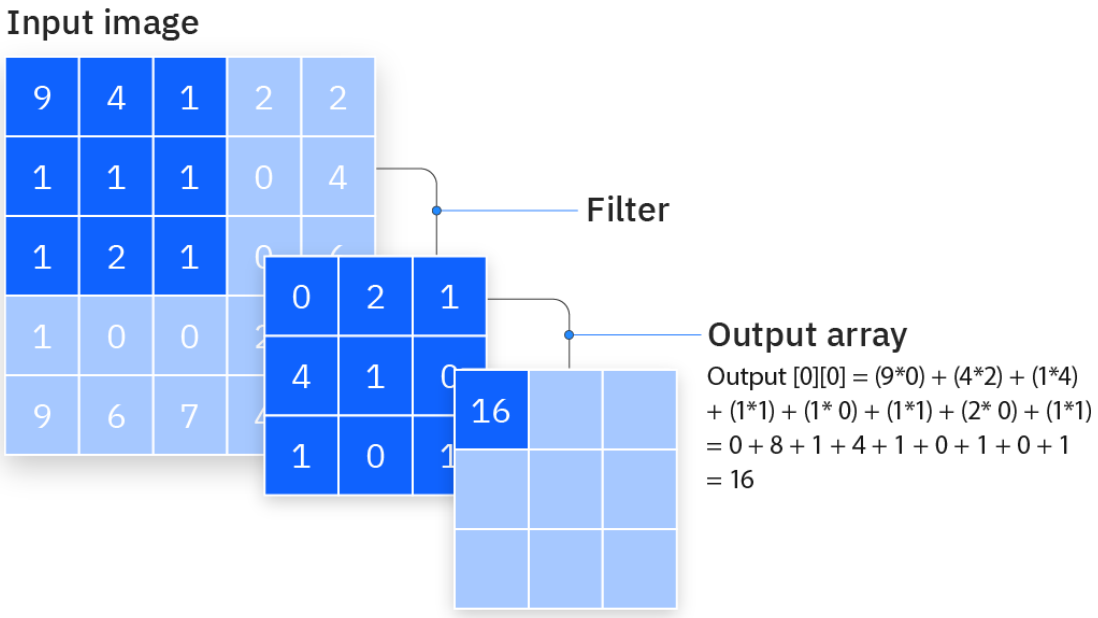
Konvoluční vrstvy vybírají z dat obrazové vzory, například horizontální nebo vertikální hrany. Reagují podobně na podobné obrazové vzory, aniž by záleželo na tom, kde na obrázku se vzor nachází (Zhang 2020).

Poolingové vrstvy pak sdružují data a tak snižují rozlišení obrázků a zrychlují zpracovávání. Zároveň ale v obrázcích zanechávají reprezentaci původních vzorů a lze je tedy připodobnit ke kompresi (Zhang 2020).

3.2 Konvoluční vrstva - forward-propagace

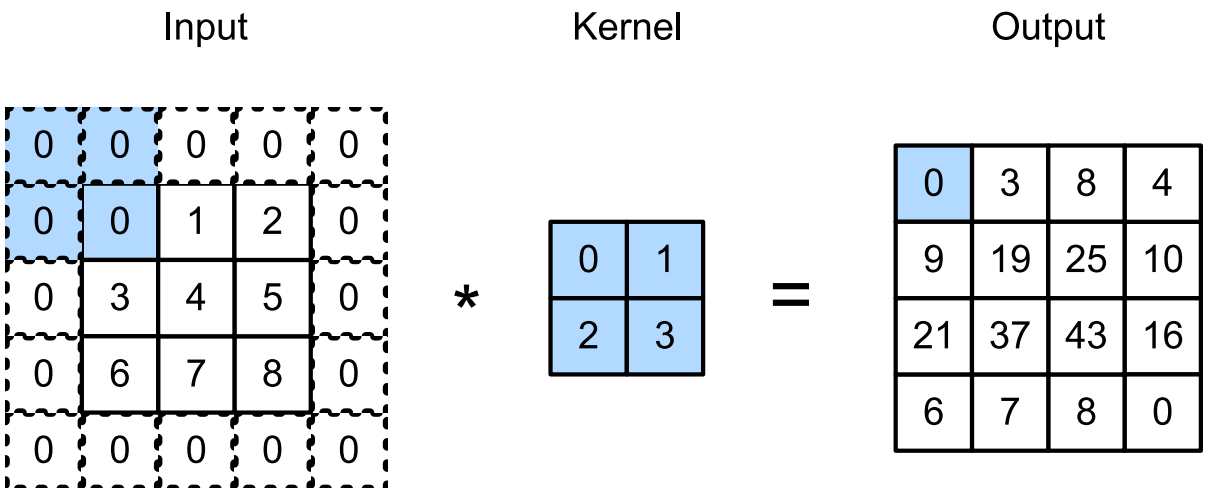
Pro porozumění konvolučním vrstvám je třeba pochopit operaci konvoluce. Pro konvoluci máme vstupní data v podobě 2D-tenzoru pixelů (v případě obrázku s jedním obrazovým kanálem) a také menší 2D-tenzor parametrů, zvaný filtr. Častá velikost filtru je 3x3 nebo 5x5 (Far1din 2023).

Při samotné konvoluci začíná filtr v levém horním rohu obrázku. Pixely překrývané filtrem jsou vynásobeny korespondujícími hodnotami filtru. Tyto součiny se pak sečtou dohromady společně s hodnotou biasu a tvoří hodnotu prvního pixelu ve výstupu konvoluční vrstvy (viz obrázek 4). Filtr se pak posune o jeden nebo více kroků doprava a z nových pixelů pod filtrem se vypočítá hodnota druhého pixelu výstupu, tj. pixelu o jeden krok doprava. Počet kroků, o které se filtr posune, nazýváme anglicky stride. Tento proces se opakuje, než se filtr dotýká pravé strany vstupu, následně se vrátí na levý kraj a posune se o jeden nebo více řádků dolů podle hodnoty stridu. Vypočítaná hodnota pak koresponduje s prvním pixelem dalšího řádku výstupu. Tímto způsobem je filtr posouván, dokud nespočítáme každý pixel výstupu (Zhang 2020; Far1din 2023). Podobně jako klasická vrstva i konvoluční vrstva využívá nelineární aktivační funkci jako např. ReLU, která se aplikuje na každý pixel ve výstupu (Far1din 2023).



(Obrázek 4) Konvoluce obrázku (IBM 2024b)

Velikost výstupu konvoluční vrstvy lze vypočítat pomocí vzorečku $d_c = \frac{d_i - d_k}{s} + 1$, kdy d značí dimenzi tj. buď šířku nebo výšku. Spodní index c značí výstup konvoluce, i značí vstupní obrázek a k značí filtr, s pak značí velikost kroků, neboli stride. Pokud je tedy filtr větší než 1x1, je výstupní obrázek menší než vstupní a ztrácíme tak určité množství informací na okrajích, jelikož krajní pixely projdou menším počtem aplikací filtru. Tomuto jevu se dá zabránit pomocí techniky jménem padding, neboli výplň (viz obrázek 5). Ta okolo celého vstupního obrázku přidá několik řad pixelů, podle toho, jakou velikost filtru používáme. Hodnoty pixelů paddingu jsou obvykle 0. Padding lze také použít abychom, zabránili průběžnému zmenšování obrázků působením konvolucí (Zhang 2020)



(Obrázek 5) Padding v konvoluční vrstvě (Zhang 2020)

Konvoluční vrstvy také většinou nepoužívají pouze jeden filtr, ale často má jedna konvoluční vrstva filtrů hned několik. Více filtrů umožňuje vrstvě detekovat větší počet základních vzorů. Jeden filtr tak může detekovat vertikální hrany, druhý horizontální hrany a třetí např. rohy. K počtu filtrů pak koresponduje počet výstupních obrázků neboli kanálů výstupu. Více kanálů je také využíváno při konvoluci barevných obrázků. Konvoluční filtr tak není vždy jen dvojrozměrný, ale může mít i hloubku. Ta se rovná počtu kanálů vstupu a konvoluce se tím mění z operace mezi dvěma 2D tenzory na operaci mezi dvěma 3D tenzory (Far1din 2023).

3.3 Konvoluční vrstva - back-propagace

Back-propagací konvoluční vrstvou potřebujeme získat dva gradientové vektory. Je třeba získat gradient vzhledem k hodnotám jádra, aby optimalizační algoritmus mohl snížit výstup loss funkce, a také gradient pro vstupní hodnoty, aby mohla back-propagace pokračovat do předešlých vrstev.

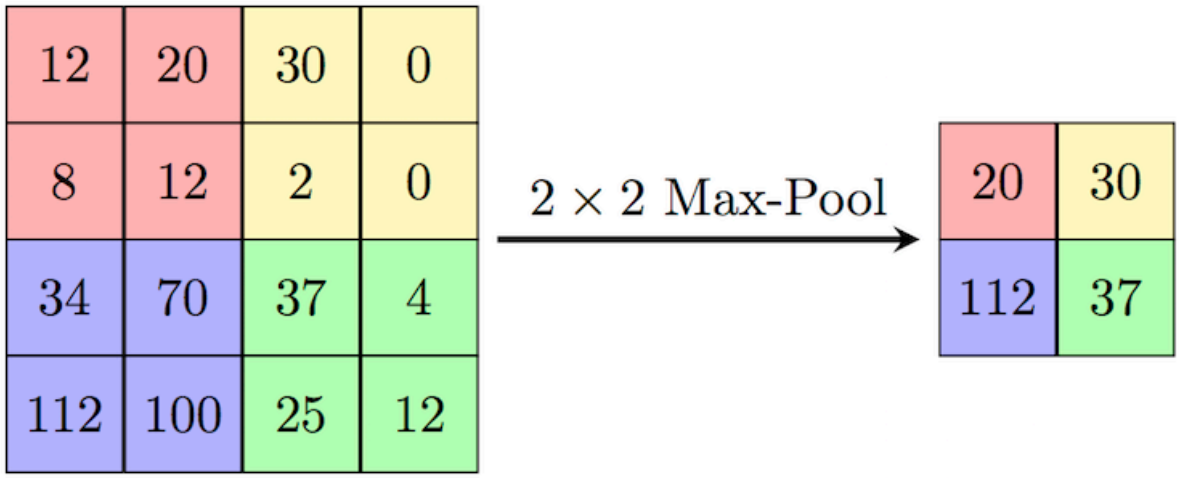
Z následující vrstvy dostane při back-propagaci konvoluční vrstva gradient loss funkce vzhledem ke svému výstupu. Pro získání gradientu filtru provedeme konvoluci původního obrázku jako při forward-propagaci, a to včetně paddingu. Jako konvoluční filtr ale v tomto případě použijeme gradient který jsme dostali z následující vrstvy. Výsledný vektor této konvoluce bude mít tvar filtru a bude obsahovat jeho derivace. Lze jej pak optimalizačním algoritmem použít ke zlepšení filtru a zvýšení přesnosti sítě (Lambert 2021).

K získání gradientu vzhledem ke vstupu konvoluční vrstvy opět použijeme gradient získaný z následující vrstvy. Ten vypaddujeme z každé strany tolika vrstvami nul, kolik je šířka nebo výška filtru mínus jedna. Tj. pokud máme gradient z následující vrstvy o velikosti 4x5 pixelů a filtr o velikosti 3x3 pixely, pak gradient vypaddujeme nulami do velikosti 8x9 pixelů (Raj 2021). Na takto vypaddovaném gradientu provedeme konvoluci s původním filtrem, který je otočený o 180 stupňů, tj. středově převrácený (Lambert 2021). Tímto způsobem získáme gradient loss funkce vzhledem ke vstupu konvoluční vrstvy, který pak slouží jako gradient vzhledem k výstupu předcházející vrstvy.

3.4 Pooling vrstva - forward-propagace

Pooling vrstva agreguje výstup konvoluční vrstvy na základě předem daného pravidla a tím zmenšuje rozměry obrázku, ale stále zachovává důležitá data. Hlavními dvěma typy pooling vrstev jsou max pooling a mean pooling, tj. průměrový pooling (IBM 2024a).

Podobně jako u konvoluce posouváme při poolingu přes obrázek filtr. Pooling filtr však nemá žádné vlastní hodnoty, namísto toho aplikuje na hodnoty pod filtrem předem danou agregační operaci - zjistí maximum nebo spočítá aritmetický průměr. Pooling obvykle používá stride, který se rovná velikosti filtru, protože při agregaci chceme každý pixel použít pouze jednou (viz obrázek 6). Populární poolingové metaparametry jsou velikost filtru 2x2 a stride 2 (Zhang 2020; IBM 2024a).



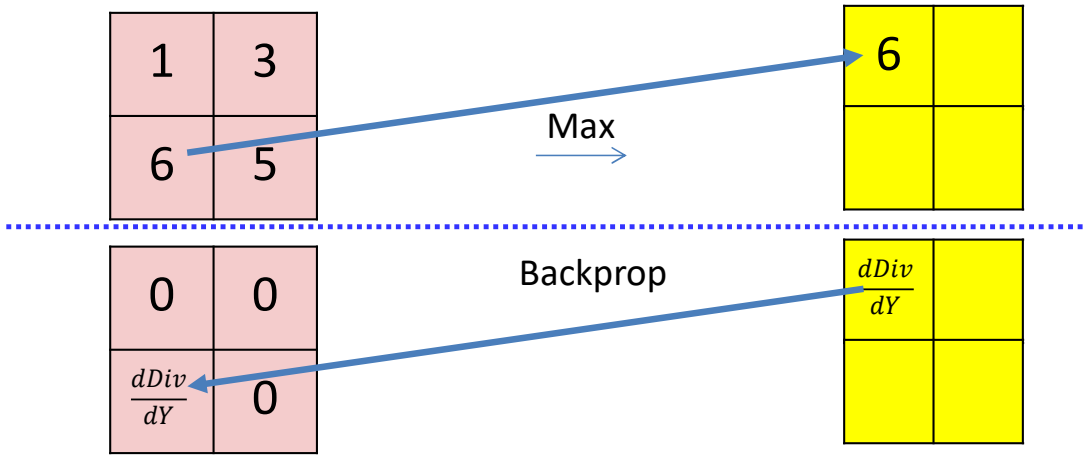
(Obrázek 6) Max-Pooling (Pictorial example of max-pooling. 2018)

Díky agregaci pomáhá pooling konvolučním sítím snížit citlivost k tomu, kde se objekt na obrázku nachází (Zhang 2020). Zároveň také pomáhá CNNs řešit problém overfittingu tím, že snižuje počet parametrů sítě. To napomáhá i rychlosti trénování (IBM 2024a). Max-pooling je obecně považován za výhodnější typ poolingu (Zhang 2020).

3.5 Max-pooling vrstva - back-propagace

Jelikož pooling vrstva nemá žádné vlastní parametry, slouží její back-propagace pouze k získání gradientu vzhledem ke vstupu vrstvy, aby back-propagace mohla pokračovat v rámci sítě dál.

Gradient získaný z následující vrstvy bude mít tvar výstupu pooling vrstvy. Pro odvození derivace vzhledem ke vstupu max-pooling vrstvy je dobré si nejdříve představit tenzor ve tvaru vstupu této vrstvy, který je celý plný nul. Do tohoto tenzoru pak vkládáme hodnoty derivace z gradientu následující vrstvy. U každého pixelu výstupu konvoluční vrstvy zjistíme, z kterého pixelu původního vstupu tato hodnota pochází, a na její korespondující místo v tenzoru nul vložíme korespondující derivaci (viz Obrázek 7). (Raj 2021)



$$\frac{dDiv}{dy(l-1,m,k,l)} = \begin{cases} \frac{dDiv}{dy(l,m,i,j)} & if\ (k,l) = P(l,m,i,j) \\ 0\ otherwise \end{cases}$$

(Obrázek 7) Derivace max-pooling vrstvy (z prezentace Raj 2021)

4 Aplikace rozpoznávání obrazu pomocí NS

Rozpoznávání obrazu je jednou z aplikací neuronových sítí a je používáno v celé řadě odvětví. V této kapitole rozebírám několik z těchto odvětví.

4.1 Doprava

Rozpoznávání obrazu má v dopravě celou řadu využití - například automatická detekce poznávacích značek, detekce dopravních značek, detekce překážek nebo jiné úlohy související s autonomními vozidly (Dilek 2023).

Obecně jsou neuronové sítě využívány ke dvěma účelům v dopravě. První z nich je detekce konkrétního objektu. Například je tak model schopen na obrázku rozpoznat dopravní značku a označit část obrázku, ve které se značka nachází. Druhý účel je pak klasifikace obrazového vstupu. Model je tak například schopen vyhodnotit, o kterou konkrétní značku se jedná (Dilek 2023).

Oblast rozpoznávání značek je už dobře prozkoumaná a mnohé modely byly schopny dosáhnout vysokých úrovní přesnosti, často přes 95%. Nejčastěji byly k tomuto účelu používány sítě CNN, například architektura YOLO (You Only Look Once). Zásadní potíží pro tyto modely však zůstává fakt, že dopravní značky jsou mezi zeměmi různé (Dilek 2023).

4.2 Zdravotnictví

Brzká detekce rakoviny má extrémně vysokou korelaci s přežitím pacienta v následujících letech. Modely využívající architekturu konvolučních neuronových sítí se ukázaly jako efektivní způsob predikce některých typů rakoviny v prvotních fázích (Hunter 2022).

Modely využívající CNN jsou aplikovány na CT-skeny nebo obyčejné rentgenové snímky. Na základě těchto snímků jsou některé modely s vysokou mírou přesnosti schopny diagnostikovat různé druhy rakoviny. Jsou také schopny určit, ve které fázi se onemocnění nachází, nebo o jaký druh nádoru se jedná. Modely přesností predikcí dosahují nebo i předčují zkušené doktory s přesnostmi predikcí i přes 90 % (Hunter 2022).

Celkově mají modely strojového učení, konkrétně konvoluční neuronové sítě, již nyní velký dopad na obor zdravotnictví. Jejich rozšíření umožňuje kvalitní a velkokapacitní rakovinový screening, čímž umožňuje nacházet rakovinu již v jejím brzkém stádiu. To zvyšuje šance pacientů na přežití a rychlejší uzdravení a snižuje výdaje zdravotnického systému (Hunter 2022).

4.3 Zemědělství

Množství pracovní síly v zemědělství se v poslední době snižuje, a tak dochází i v tomto relativně tradičním oboru k využívání strojového učení pro zefektivnění práce (Wu 2024).

Architektura konvolučních neuronových sítí je v zemědělství používána například při detekci zralosti ovoce. Díky automatické detekci může docházet i k automatické sklizni a lze tak snížit výdaje na pracovní sílu. V zemědělství jsou také využívány i složitější architektury neuronových sítí, konkrétně například architektura YOLO, která je v porovnání na modely využívající několik specializovaných modelů CNN schopná běžet rychleji. Postupy využívající YOLO modely slouží například k detekci škůdců a následné lokální aplikaci pesticidů. Místo nutnosti aplikovat pesticidy na celé pole je pak takový postup cenově efektivnější a také ekologičtější (Wu 2024).

5 Vlastní implementace NS pro rozpoznávání obrazu

5.1 Datasety

Pro vlastní implementaci neuronových sítí jsem se rozhodl využít dva veřejně dostupné datasety - MNIST a Fashion MNIST. Dataset MNIST je soubor 60 tisíc trénovacích plus 10 tisíc testovacích obrázků ručně psaných jednociferných čísel o velikosti 28x28 pixelů (LeCun 1998). Tento dataset je snad nejjednodušším datasetem pro rozpoznávání obrazu a bývá často považován za příliš jednoduchý. V moji práci bych ho rád využil hlavně na základní demonstraci funkčnosti modelů.

Dataset Fashion MNIST slouží jako přímá a modernější náhrada datasetu MNIST. Používá stejného formátovaní i velikosti obrázků. Velikost datasetu je také identická. Komplexita samotných obrázků je však o dost vyšší než u MNIST; obrázky jsou získané ze stránky Zalando a obsahují 10 druhů zboží. Představují tak větší výzvu i pro modernější modely ML a umožňují reálnější testování výkonnosti modelu (Xiao 2017). Tento dataset chci v mé práci využít jako menší výzvu co se stavění a trénování modelů týká.

5.2 Programovací jazyk a knihovny

Můj původní plán byl naimplementovat plně propojenou a konvoluční neuronovou síť v novém programovacím jazyce Mojo, který by se měl rychlostí blížit k low-level jazykům jako je C, ale s high-level syntaxem podobným Pythonu. Jazyk je ale stále v betě a chybí v něm mnoho základní funkcionality. Jeho použití by tak tuto práci příliš překomplikovalo a rozhodl jsem se tedy od tohoto nápadu upustit.

Jazyk, který jsem nakonec zvolil, je Python. Jeho jednoduchý syntax a široký výběr komputačních knihoven je pro tuto práci ideální. Většina mé implementace využívá pouze knihovnu Numpy, která umožňuje optimalizované operace jako maticové násobení nebo agregaci. Pouze pro konvoluce využívám komputační knihovny Scipy. Ta umožňuje efektivní konvoluci n-dimenzionálních vektorů.

5.3 Zdroje a inspirace

Moje implementace je postavena na implementaci neuronové sítě v numpy z e-knihy Neural Networks from Scratch (Kinsley 2020). Z této knihy jsem si vzal implementaci LayerDense, ActivationReLU, ActivationSoftmaxLossCategoricalCrossentropy a OptimizerSGD tříd. Ty jsem následně lehce modifikoval, aby fungovaly s mými vlastními třídami, případně aby fungovaly lépe.

Z knihovny TensorFlow jsem adoptoval implementaci inicializace weights a filtrů. Konkrétně inicializaci Glorot Uniform neboli také Xavier inicializaci.

5.4 Rozsah vlastní implementace

Nad rámec tříd získaných z e-knihy Neural Networks from Scratch jsem ve své vlastní implementaci vytvořil funkce pro načítání dat, několik dalších typů vrstev a také Network třídu, která reprezentuje neuronovou síť složenou z vrstev. I přes to, že píšu o aktivační funkci Sigmoid, jsem se rozhodl ji ve své implementaci nezahrnout.

Funkce pro načítání dat jsou schopné načíst data datasetů MNIST a Fashion MNIST v jejich standardizovaném formátu a uložit je do Numpy tenzorů.

První vrstvou mé implementace je konvoluční vrstva. Ta dostává vstup ve formátu 4D-tenzoru. První dimenze představuje počet vstupních obrázků, druhá představuje počet kanálů na obrázek, třetí a čtvrtá dimenze tenzoru ukládají pixelové hodnoty každého obrazového kanálu. Konvoluční vrstva používá funkci “scipy.convolve” pro samotné

konvoluce při back- i forward-propagaci. Architektura konvoluční vrstvy je popsána v kapitolách 3.2 a 3.3.

Další vrstvou mé implementace je Max-Pooling vrstva. Její princip je popsán v kapitolách 3.4 a 3.5. Tvar vstupu je stejný jako tvar vstupu konvoluční vrstvy. Moje implementace rozdělí pixely každého kanálu všech obrázků na sub-pole o velikosti max-pooling filtru a z každého z nich pak vezme maximum. Forward-propagace tak je rychlá a efektivní. Back-propagace max-pooling byla však náročnější na implementaci; je funkční, ale neefektivní (viz kapitola 5.5).

Poslední vrstvou mé implementace je Flatten vrstva. Ta převádí 4D-tenzory do 1D tenzorů, které je možné použít jako vstup pro plně propojenou vrstvu. Při back-propagaci pak gradienty z plně propojené vrstvy změní do tvaru vhodného pro max-pooling a konvoluční vrstvy. Moje implementace využívá v obou směrech funkci “numpy.reshape”.

Třída Network implementuje automatickou forward- a back-propagaci, trénování, ukládání a načítání modelů tvořených z více vrstev. Tyto funkce výrazně zjednodušují proces trénování i evaluace sítě. Ukládání sítě zapisuje parametry a architekturu do .json souboru, ze kterého je možné model načíst zpět a znovu využívat.

5.5 Problémy při implementaci

Problémů při implementaci nastalo mnoho. První problém, který se vyskytl, byl s programovacím jazykem Mojo. Ten slibuje vysokou výkonnost, jednoduchý syntax a obecně se prezentuje jako skvělá volba pro strojové učení. Bohužel však v jazyku chybí mnoho základních funkcí. Například získávání několika hodnot z tenzoru, tzv. slicing, v jazyce vůbec není. Mnoho slibovaných funkcionalit, jako např. vektorizaci funkcí, jsem také nebyl schopen zprovoznit. Celková náročnost implementace se tak drasticky zvyšovala a rozhodl jsem se od jazyku Mojo upustit.

Dalším problémem, se kterým jsem se potýkal, byla inicializace weights a konvolučních filtrů. Inicializace z e-knihy Neural Networks from Scratch nebyla funkční a ani několik dalších inicializačních metod neprodukovalo lepší výsledky. Výstupy vrstev se skoro okamžitě dostaly do záporu a aktivační funkce ReLU pak zapříčinila, že vstup do další vrstvy už byl pouze nulový. Fungující inicializaci jsem získal až adoptováním Glorot Uniform inicializace z knihovny TensorFlow. Ta generuje normální rozdělení mezi -1 a 1, tu pak vynásobíme hodnotou $\sqrt{\frac{6}{i+o}}$, která slouží jako nejvyšší nebo nejnižší možná hodnota

parametru. Proměnná *i* reprezentuje počet vstupů, pro které weights inicializujeme, proměnná *o* reprezentuje počet výstupů (Google 2024).

S problémy jsem se potýkal i při implementaci max-pooling vrstvy. Zatímco forward-propagace využívá efektivních funkcí z knihovny Numpy, back-propagace využívá několika for-cyklů v Pythonu. Trénování konvolučních sítí s max-pooling tak trvá výrazně déle než srovnatelné plně propojené sítě. Tento problém se mi nepodařilo vyřešit, ale ukazuje na to, jak efektivní jsou implementace funkcí z knihovny Numpy, pokud jsme je schopni využít pro naše potřeby.

5.6 Výsledné modely a jejich výkonnost

V mojí závěrečné implementaci jsem se rozhodl postavit tři modely. První model pro dataset MNIST a zbylé dva pro dataset Fashion MNIST.

První model využívá jednu skrytou a jednu výstupní vrstvu, obě plně propojené. Na skrytou vrstvu se aplikuje aktivační funkce ReLU, má 784 vstupů a 10 výstupů. Druhá vrstva má 10 vstupů a 10 výstupů (jeden pro každou klasifikační třídu), jako aktivační funkci pak používá softmax; z té se pak počítá Categorical Cross-entropy loss. Model využívá optimalizér SGD s rychlostí učení 1. Při trénování byl použit batch-size 60 a model byl trénován na 20 epochách. Model dosahuje přesnosti 93 % na evaluačním datasetu MNIST o velikosti 10 000 vzorků.

Druhý model využívá tři plně propojené vrstvy. Architekturou je srovnatelný s prvním modelem, pouze je mezi vrstvy prvního modelu přidaná jedna plně propojená vrstva s 10 vstupy a 10 výstupy a aktivační funkcí ReLU. Optimalizér SGD byl využit s rychlostí učení 0.01. Model byl trénován na 30 epochách s batch-size 10. Model dosáhl na evaluačním datasetu Fashion MNIST přesnosti 81 %.

Třetí model využívá architektury CNN. Ta se sestává ze dvou konvolučních bloků, každý obsahující konvoluční vrstvu následovanou max-pooling vrstvou a aktivací ReLU. Obě konvoluční vrstvy používají velikost filtru 3x3; první z nich používá 2 filtry, druhá používá 3 filtry. Max-pooling vrstvy používají velikost filtru 2x2 a stride 2. Po těchto čtyřech vrstvách následuje flatten vrstva. Dál síť obsahuje dvě plně propojené vrstvy, první z nich má 147 vstupů a 10 výstupů, druhá má pak 10 vstupů a 10 výstupů, jeden pro každou klasifikační třídu. První využívá aktivační funkci ReLU, druhá pak softmax. Tato síť byla velice pomalá

na natrénování kvůli back-propagaci max-pooling vrstvy; i tak je však funkční s přesností na evaluačním Fashion MNIST datasetu 74 %.

Všechny výše zmíněné sítě a celkový kód mé implementace jsou k dispozici v GitHub repozitáři na URL “https://github.com/TENBrnak/CNNs_from_scratch”. Sítě je možné zkusit si sám natrénovat, ale lze i načíst výše zmíněné, předem natrénované, modely.

Závěr

Věřím, že se mi touto prací podařilo naplnit cíle, které jsem si stanovil. Popsal a vysvětlil jsem základní prvky plně propojené neuronové sítě a také konvoluční neuronové sítě. Tyto koncepty jsem pak také aplikoval při programování vlastních neuronových sítí.

Při práci mě zaujal koncept back-propagace a existence autogradu - automatického gradientu. Jde o nástroj na úrovni programovacího jazyka nebo knihovny, který je schopen automaticky počítat gradient pro libovolný typ vrstvy, aniž by je člověk musel explicitně definovat. Rád bych se o tomto konceptu do budoucna dozvěděl více.

Seznam literatury

1. BROWNLEE, Jason. Loss and Loss Functions for Training Deep Learning Neural Networks. *Machine Learning Mastery* [online]. 2019 [cit. 2024-02-14]. Dostupné z: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/
2. BROWNLEE, Jason. What is the Difference Between a Parameter and a Hyperparameter? *Machine Learning Mastery* [online]. 2019 [cit. 2024-02-14]. Dostupné z: https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/
3. DILEK, Esma a Murat DENER. Computer Vision Applications in Intelligent Transportation Systems: A Survey [online]. Ankara, Turkey, 2023 [cit. 2024-04-07]. Dostupné z: https://doi.org/10.3390/s23062938. Article. Graduate School of Natural and Applied Sciences, Gazi University.
4. EVROPSKÝ PARLAMENT (EP). *Co je umělá inteligence a jak ji využíváme?* Online. Evropský parlament. 2020, 2023-11-21. Dostupné z: https://www.europarl.europa.eu/topics/cs/article/20200827STO85804/umela-inteligence-definice-a-vyuziti. [cit. 2024-02-13].
5. FAR1DIN. *Convolutional Neural Networks from Scratch | In Depth* [online video]. 2023, 2023 [cit. 2024-03-17]. Dostupné z: https://www.youtube.com/watch?v=jDe5BAsT2-Y
6. GOOGLE. Tf.keras.initializers.GlorotUniform. TensorFlow [online]. 2024 [cit. 2024-04-04]. Dostupné z: https://www.tensorflow.org/api_docs/python/tf/keras/initializers/GlorotUniform
7. HUNTER, Benjamin, Sumeet HINDOCHA a Richard W. LEE. The Role of Artificial Intelligence in Early Cancer Diagnosis. *MDPI* [online]. Basel, Switzerland, 2022 [cit. 2024-02-13]. ISSN 2072-6694. Dostupné z: doi:10.3390/cancers14061524
8. IBM. What are convolutional neural networks? IBM. *IBM* [online]. 2024a [cit. 2024-03-18]. Dostupné z: https://www.ibm.com/topics/convolutional-neural-networks
9. IBM. What is a neural network? *IBM* [online]. 2024b [cit. 2024-02-14]. Dostupné z: https://www.ibm.com/topics/neural-networks

10. INSIGHTSOFTWARE. Traditional Programming vs Machine Learning. *Insightsoftware* [online]. 2023 [cit. 2024-02-15]. Dostupné z: https://insightsoftware.com/blog/machine-learning-vs-traditional-programming/
11. KINSLEY, Harrison a Daniel KUKIELA. Neural Networks from Scratch (NNFS) [online]. 2020 [cit. 2024-04-04]. Dostupné z: https://nnfs.io
12. LAMBERT, John. Backpropagation through a Conv Layer. *John Lambert* [online]. 2021 [cit. 2024-03-20]. Dostupné z: https://johnwlambert.github.io/conv-backprop/
13. LECUN, Yann. The MNIST Database. Yann LeCun [online]. 1998 [cit. 2024-04-03]. Dostupné z: http://yann.lecun.com/exdb/mnist/
14. RAJ, Bhiskha. *Deep Neural Networks Convolutional Networks IV* [online]. 2021 [cit. 2024-03-20]. Dostupné z: https://deeplearning.cs.cmu.edu/S21/document/slides/Lec12.CNN4.pdf. Prezentace.
15. SANDERSON, Grant. *Backpropagation calculus | Chapter 4, Deep learning* [online - video]. 2017, 2017 [cit. 2024-02-20]. Dostupné z: https://www.youtube.com/watch?v=tIeHLnjs5U8&t=519s
16. SILVER, David, Thomas HUBERT, Julian SCHRITTWIESER a Demis HASSABIS. AlphaZero: Shedding new light on chess, shogi, and Go. GOOGLE. *Google DeepMind* [online]. 2018 [cit. 2024-02-15]. Dostupné z: https://deepmind.google/discover/blog/alphazero-shedding-new-light-on-chess-shogi-and-go/
17. WU, Ningning. Application and evaluation of deep learning based image recognition techniques in agriculture [online]. Chegdu, China, 2024 [cit. 2024-04-07]. DOI: 10.54254/2755-2721/48/20241260.Dostupné z: https://www.researchgate.net/publication/379076491_Application_and_evaluation_of_deep_learning_based_image_recognition_techniques_in_agriculture/fulltext/65f9c512a4857c796260cd1a/Application-and-evaluation-of-deep-learning-based-image-recognition-techniques-in-agriculture.pdf. Proceedings of the 4th International Conference on Signal Processing and Machine Learning. Glasgow College, University of Electronic Science and Technology of China.
18. XIAO, Han, Kashif RASUL a Roland VOLLGRAF. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms [online]. Berlin, 2017 [cit. 2024-04-03]. Dostupné z: https://arxiv.org/pdf/1708.07747.pdf. Research Paper. Zalando Research.

19. ZHANG, Aston, Zack C. LIPTON, Mu LI a Alex J. SMOLA. *Dive into Deep Learning* [online]. 2020, 2023 [cit. 2024-02-13]. Dostupné z: https://d2l.ai

Seznam obrázků

1. Deep Neural Network. In: IBM(a). *IBM* [online]. [cit. 2024-02-14]. Dostupné z: https://www.ibm.com/content/dam/connectedassets-adobe-cms/worldwide-content/cdp/cf/ul/g/3a/b8/ICLH_Diagram_Batch_01_03-DeepNeuralNetwork.png
2. RASCHKA, Sebastian. Gradient Descent. In: *Sebastian Raschka* [online]. 2024 [cit. 2024-02-15]. Dostupné z: https://sebastianraschka.com/images/faq/gradient-optimization/ball.png
3. ReLU v/s Logistic Sigmoid. In: *Medium* [online]. 2017 [cit. 2024-02-13]. Dostupné z: https://miro.medium.com/v2/resize:fit:1400/format:webp/1*XxxiA0jJvPrHEJHD4z893g.png
4. Convolutional Neural Network. In: IBM(b). *IBM* [online]. 2024 [cit. 2024-03-14]. Dostupné z: https://www.ibm.com/content/dam/connectedassets-adobe-cms/worldwide-content/creative-assets/s-migr/ul/g/ed/92/iclh-diagram-convolutional-neural-networks.png
5. A CNN sequence to classify handwritten digits. In: SAHA, Sumit. *Saturn Cloud* [online]. 2018 [cit. 2024-03-14]. Dostupné z: https://saturncloud.io/images/blog/a-cnn-sequence-to-classify-handwritten-digits.webp
6. Pictorial example of max-pooling. In: *Computer Science Wiki* [online]. 2018 [cit. 2024-03-21]. Dostupné z: https://computersciencewiki.org/images/8/8a/MaxpoolSample2.png