Česká zemědělská univerzita v Praze

Technická fakulta

Katedra fyziky

Bakalářská práce

Systém pro detekci poléhání pšenice z obrazových dat

Artem Khokhlov

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Technická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Artem Khokhlov

Informační a řídicí technika v agropotravinářském komplexu

Název práce

Systém pro detekci poléhání pšenice z obrazových dat

Název anglicky

Wheat lodging detection system from image data

Cíle práce

Cílem práce je navrhnout a realizovat softwarové řešení pro detekci poléhání porostu pšenice na základě obrazových dat. Výsledné řešení zhodnotit z hlediska spolehlivosti a výpočetní náročnosti.

Metodika

- Vypracování literární rešerše s využitím vědeckých databází: Web of Science, Scopus, Google Scholar a dalších.

- Na základě literární rešerše zvolit vhodný způsob detekce poléhání pšenice a ten implementovat.

- Provést statistické zhodnocení výsledků.

Doporučený rozsah práce

30-40

Klíčová slova

Deep learning, obrazová analýza, zpracování dat

Doporučené zdroje informací

Cao, W., Qiao, Z., Gao, Z., Lu, S. & Tian, F. 2021. Use of unmanned aerial vehicle imagery and a hybrid algorithm combining a watershed algorithm and adaptive threshold segmentation to extract wheat lodging, Physics and Chemistry of the Earth, Volume 123, Pages 103016.

Hasan, M.M., Chopin, J.P., Laga, H. & Miklavcic, SJ. 2018. Detection and analysis of wheat spikes using Convolutional Neural Networks. Plant Methods 14, 100. https://doi.org/10.1186/s13007-018-0366-8

Chollet, F. Deep learning v jazyku Python. 2019. ISBN: 978-80-247-3100-1

Předběžný termín obhajoby

2023/2024 LS – TF

Vedoucí práce

Ing. Jakub Lev, Ph.D.

Garantující pracoviště

Katedra fyziky

Elektronicky schváleno dne 28. 1. 2022

prof. Ing. Martin Libra, CSc.

Vedoucí katedry

Elektronicky schváleno dne 23. 2. 2022

doc. Ing. Jiří Mašek, Ph.D.

Děkan

V Praze dne 28. 03. 2024

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci „Systém pro detekci poléhání pšenice z obrazových dat“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 28.03.2024 _____

Poděkování

Rád bych poděkoval Ing. Jakubu Lvovi, Ph.D., vedoucímu této bakalářské práce, za pomoc a rady při zpracování této práce. Také bych chtěl poděkovat své manželce a rodině, kteří mě po celou dobu studia podporovali.

Systém pro detekci poléhání pšenice z obrazových dat

Abstrakt

Polehání pšenice, nežádoucí jev ohýbání nebo lámání rostlin pšenice, představuje významnou hrozbu pro zemědělskou produktivitu a potravinovou bezpečnost. Včasná detekce polehání je klíčová pro provádění nápravných opatření a optimalizaci postupů řízení plodiny. Tato bakalářská práce představuje komplexní výzkum v oblasti vývoje systému pro detekci polehání pšenice s využitím obrazových dat.

Hlavním cílem práce je prozkoumat technologii hlubokého učení pro klasifikaci obrázků a vytvořit systém pro detekci poléhání pšenice s využitím těchto teoretických materiálů. Účinnost systému při detekci událostí polehání je demonstrována prostřednictvím empirických experimentů, což zdůrazňuje jeho potenciál pro zlepšení postupů řízení plodiny pšenice a zvýšení potravinové bezpečnosti.

Klíčová slova: Poléhání pšenice, analýza obrazových dat, počítačové vidění, strojové učení, precizní zemědělství.

Wheat lodging detection system from image data

Abstract

Wheat lodging, the undesirable phenomenon of wheat plants bending or breaking, poses a significant threat to agricultural productivity and food security. Early detection of lodging is crucial for implementing corrective measures and optimising crop management practices. This bachelor thesis presents a comprehensive research in the development of a wheat lodging detection system using image data.

The main objective of this thesis is to explore deep learning technology for image classification and develop a wheat lodging detection system using these theoretical materials. The effectiveness of the system in detecting lodging events is demonstrated through empirical experiments, highlighting its potential for improving wheat crop management practices and enhancing food security.

Keywords: Wheat lodging, image data analysis, computer vision, machine learning, precision agriculture.

Obsah

1 Úvod

Pšenice je jednou z nejdůležitějších plodin v zemědělském komplexu. Produkce pšenice je složitý a dlouhodobý proces, který vyžaduje pozornost věnovanou mnoha aspektům, včetně kvality a kvantity polehlé produkce. K poléhání dochází typicky ke konci vegetace.

V dnešní době, kdy dochází k technologickému pokroku a počítače se stále více podílejí na monitorování a analýze dat, lze některé technologie, jako je strojové učení a vizuální zpracování dat, využít ke zvýšení přesnosti a efektivity analýzy a kontroly plodin. Díky tomu mohou tyto technologie pomoci zemědělcům předcházet neúrodě a ztrátám na úrodě a také přesněji vyhodnocovat kvalitu plodin.

Cílem této práce je vyvinout a vyhodnotit nástroj pro detekci poléhání pšenice pomocí neurálních sítí a vyhodnocování obrazových dat. Tento přístup může v budoucnu pomoci zemědělské výrobě s včasnou detekcí a prevencí poléhání, což v konečném důsledku zvýší množství a kvalitu úrody

2 Současný stav vědění

Poléhání je stav pšenice, kdy jsou rostliny staženy k zemi, což může být způsobeno vnějšími faktory i vlastnostmi konkrétní odrůdy pšenice (Obr. 1). Tento problém je příčinou značných ztrát výnosů v oblastech, kde pšenice roste za obtížných povětrnostních podmínek. Proto je včasné odhalení a prevence poléhání prioritní v oblastech, kde je tento jev častý [1].

Nejčastěji se poléhání objevuje ve fázích růstu, kdy je pšenice již zralá a má výraznou hmotnost a dlouhý stonek. V této fázi je rostlina nejvíce vystavena vnějším faktorům, kterým ne vždy dokáže odolat. Včasné odhalení a preventivní opatření proto pomohou snížit počet zkažených rostlin. Vzhledem k tomu, že rostlina v tomto stavu leží na zemi, může od vlhkosti a půdy zplesnivět, choroba se může dále šířit na dobré rostliny a v době sklizně může být za nakažené považováno všechno dobré zrno, které se dostane do stejného skladovacího prostoru s nakaženým zrnem[1].

Obvykle se poléhání zjišťuje při vizuální prohlídce pole, která však vzhledem k různým faktorům není vždy účinná. Například na velkém poli se může polehnutí objevit uprostřed, člověk, který se dívá z okraje, ho nebude schopen odhalit a problém se může zhoršit. V takových případech se používají satelitní snímky a snímky z dronů. Nicméně analýza je manuální, což vede k vysoké závislosti na lidském faktoru. Člověk si prostě může poléhání nevšimnout, označit oblast za dobrou a při sklizni se ukáže, že tam ve skutečnosti poléhání bylo. Čím déle zůstává poléhání bez dozoru, tím větší je pravděpodobnost, že se poléhání rozšíří a zvýší se ztráty na úrodě[2].

Obr. 1 Polehlá pšenice [3]

V době psaní této práce je detekce polehání většinou manuální, ale existují sofistikovanější přístupy, které zahrnují automatickou detekci ze satelitních snímků a snímků z dronů v reálném čase. To zemědělcům usnadňuje práci a šetří čas [4].

Pokud jde o prevenci tohoto jevu, může to být různé. Od šlechtění speciálních odrůd pšenice, které vydrží i horší povětrnostní podmínky, až po vytváření speciálních lepších podmínek přímo na poli. Pozornost se vyplatí věnovat i použitému hnojivu, protože i to pomůže rostlině k plnému rozvoji a díky plnému spektru živin roste zdravější a odolnější. Tyto opatření společně pomáhají minimalizovat riziko a dopad poléhání na produkci pšenice [5].

2.1 Konvoluční site

Konvoluční neuronové sítě (dále jen CNN) jsou v době psaní této práce jedním z hlavních trendů v oblasti zpracování obrazových dat. CNN lze použít jak pro běžnou klasifikaci, tj. určení přítomnosti určitého znaku na obrázku (např. pravděpodobnosti, že je na obrázku zobrazen člověk), tak pro sofistikované analýzy (např. určení konkrétního typu a stupně zdraví rostliny z fotografie). Tato technika se aktivně studuje a vědecké průlomy ve strojovém učení obecně se objevují se záviděníhodnou pravidelností, v době psaní tohoto článku dokáže neuronová síť vygenerovat celé nestatické video, které může ukazovat lidi nebo zvířata (Sora od OpenAI), ještě před několika lety bylo obtížné si něco takového představit. Rychlý vývoj přímo souvisí s rychlostí růstu výpočetního výkonu, protože trénování neuronové sítě je výpočetně náročný proces.[24].

2.1.1 Konvoluce

Konvoluce je matematická operace, která slouží k získání invertované oblasti překrývání(z) dvou jiných funkcí (f a g) [6]. Obecný vzorec je:

$$z(t) = f(t) \times g(t) = \sum_{\tau=-\infty}^{+\infty} f(\tau)g(t-\tau) \quad (1)$$

CNN pracují s tenzorem třetího řádu jako vstupem, což může být například obrázek s výškou *H*, šířkou *W* a třemi barevnými kanály (*R, G, B*). Nicméně CNN může zpracovávat i tenzory vyššího řádu, přičemž vstup postupuje skrz různé vrstvy zpracování. Mezi tyto vrstvy mohou patřit konvoluční, pooling, normalizační a další.[6].

2.1.2 Tenzor

V souvislosti s CNN je tenzor jedním z nejdůležitějších pojmů v tomto tématu, protože je generalizujícím označením pro několik datových struktur. Tenzor je datová struktura, která může mít jakýkoli počet dimenzí a často se označuje jejich počtem, např. 3d tensor znamená tensor se třemi dimenzemi. Tento přístup slouží k lepšímu zobecnění a umožňuje popisovat různé typy dat jedním společným termínem. V kontextu CNN je to velmi výhodné, protože tenzory se používají všude a popis jejich specifického typu dat by vedl k nadměrnému nárůstu počtu pojmů a problematika by se stala méně srozumitelnou pro lidi, kteří s ní nemají mnoho zkušeností.

Zobecnění napříč dimenzemi:

Užitečnost tensorů vychází z jejich schopnosti zobecňovat napříč různými počty dimenzí:

- Skalár, reprezentující jednu číselnou hodnotu, je považován za 0-rozměrný tenzor.
- Vektor, jednorozměrné pole čísel, je rozpoznán jako 1-rozměrný tenzor.
- Matice, dvourozměrné pole čísel, je identifikována jako 2-rozměrný tenzor.

Struktura s více než dvěma dimenzemi se nazývá podle počtu indexů potřebných k tomu, aby se do určité části struktury dostalo, tj. struktura s 4 dimenzemi se bude nazývat 4d-tenzor apod. Tenzor je spojení pro pojmy jako matice, skalár, vektor. Práce s pojmem tenzor zjednodušuje popis datové struktury, stačí specifikovat počet dimenzí tenzoru, aby bylo zřejmé, o čem se jedná [7].

Aplikace v CNN:

V CNN tenzory usnadňují reprezentaci a manipulaci s daty v různých fázích[8]:

- **Vstupní Tenzor:** Obvykle matice dat, která se odesílá do modelu. Například obrázek, který má tři dimenze.
- **Tenzor Váhy:** Malá matice pro extrakci specifických rysů z obrázku. Každá oblast má svou vlastní váhu a násobí se jí příslušná část obrazu, čímž vznikne nová hodnota.
- **Výstupní Tenzor:** Výsledek operace konvoluce, který je ovlivněn rozměry obrazu a vlastnostmi filtru, jako je velikost a krok.
- **Tenzory vyšších dimenzí:** Používají se pro složitější data jako video nebo objemové informace, zavádějí další rozměry pro hloubku nebo čas.

Tenzory se v CNN používají všude, takže jsou jednou z hlavních věcí v kontextu pochopení výkonnosti neuronových sítí. Pokud se pro optimalizaci tenzorových operací používají frameworky hlubokého učení, zvyšuje se také rychlost neuronových sítí.

2.1.3 Architektura CNN

Aby se model naučil rozpoznávat složité závislosti v obrazech, potřebuje mnoho vrstev (Obr. 6), z nichž každá je určena pro konkrétní účel, což pomáhá rozdělit práci v modelu a urychlit ji, v důsledku čehož jsou výsledky také přesnější, než jakých by bylo možné dosáhnout pomocí tradičních algoritmů strojového učení [9].

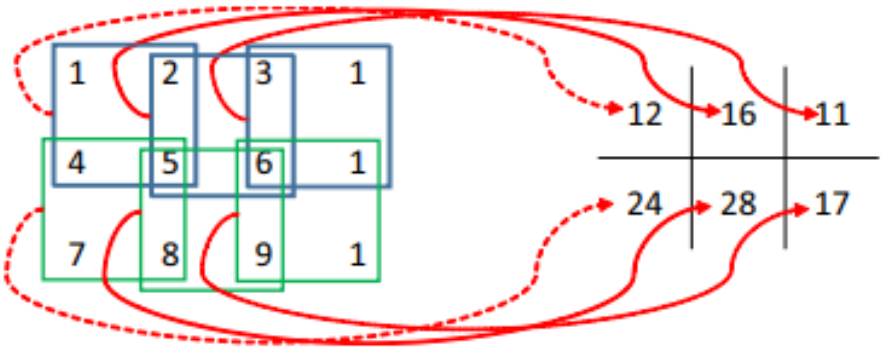
Typy vrstev CNN:

Konvoluční vrstva (Convolutional Layer)

Hlavní typ vrstvy v CNN, která je zodpovědná za zpracování a analýzu dat. Vyhledává a zvýrazňuje specifické rysy na obrázku. Filtr je vždy určen k detekci jednoho konkrétního rysu, takže pro složitější detekci je možné zvýšit hloubku sítě přidáním nových konvolučních vrstev, z nichž každá bude sloužit k detekci konkrétního rysu v obrázku a konečný výsledek bude součtem výstupů všech filtrů.[9].

V rámci CNN konvoluční vrstva aplikuje na zadaný obraz filtry a vytváří mapu rysů (Obr. 2), která zdůrazňuje určité vlastnosti nebo vzory ve vstupu. Sousední části se sčítají a výsledkem je matice složená z nových polí, která vyplývají z tohoto sčítání [9].

Obr. 2 Vizuální interpretace konvoluční operace. Čísla v čtvercích jsou sečtena, aby vytvořila novou hodnotu, která zaujímá pozici v zmenšené mřížce odpovídající pozici čtverce ve vztahu k ostatním čtvercům. [9]



Komponenty Konvoluční Operace

- **Vstupní Obraz** (Mapa rysů): Data představující 2d tenzor, který je přijímán a zpracováván konvoluční vrstvou. V případě CNN může být vstupem obrázek nebo mapa rysů získaná při aplikaci předchozích vrstev.
- **Filtr** (Kernel): Matice malé velikosti (obvykle 2-3 pixely na výšku a šířku), která se používá k extrakci specifických prvků z obrázku.
- **Krok** (Stride): Filtr se na obraz aplikuje opakovaně, a aby bylo zajištěno zpracování všech částí obrazu, měl by mít filtr určitý krok, o který bude posunut poté, co byly provedeny všechny potřebné operace s jednou částí obrazu. Krok 1 znamená, že se filtr posouvá o jeden pixel najednou. Volba kroku ovlivňuje prostorové rozměry výstupní mapy rysů.
- **Ohraničení** (Padding): Užitečná komponenta, která pomáhá omezit oblast zpracování, může být použita buď k přidání rámečku kolem obrázku, aby nedošlo ke ztrátě dat, nebo naopak může omezit zpracovávanou oblast tak, aby filtr nebyl použit v místech, kde to není nutné. Padding je obvykle nula, protože při zpracování filtrem násobení nulou nedovolí, aby padding data jakkoli ovlivnila výsledek.

Proces Konvoluční Operace

- **Inicializace:** Inicializace probíhá umístěním filtru do rohu obrazu.
- **Násobení**: Každá část filtru procházející obrazem je vynásobena odpovídající překrývající se částí obrazu.
- **Součet**: Výsledky překrytí filtrů se poté sečtou a získá se nová hodnota. Takže například při použití filtru 2x2 vznikne ze čtverce odpovídající hodnoty čtverec 1x1, kde hodnota odpovídá výsledku operace součtu. Výsledný součet může být následně zpracován aktivační funkcí ReLU(Rectified Linear Unit).
- **Výstupní Mapa Vlastností**: Výsledek sčítání se umístí na novou mapu prvků v místě, kde byl použit. Například při použití filtru 2x2 na matici bude počáteční čtverec této velikosti v nové mapě pod indexem [1, 1].

- **Posunutí Filtru**: Po dokončení všech operací se filtr dále pohybuje po zadané trajektorii, dokud tímto způsobem neprojde celý obraz. V případě, že filtr narazí na okraj obrazu - pohybuje se dále vertikálně a projde nový řádek obrázku.

Výsledná Mapa Rysů

Výsledek konvoluční vrstvy, který je produktem aplikace filtru na obraz. Každá mapa je výsledkem aplikace jednoho konkrétního filtru. Je možné použít více filtrů, což povede k vytvoření mapy pro každý filtr.

ReLU vrstva (Activation Layer)

Rectified Linear Unit, známá také jako aktivační vrstva. Prostřednictvím této vrstvy se do modelu zavádí nelinearita. Toho je dosaženo nastavením všech záporných hodnot na nulu. Tím se model naučí lépe aproximovat nelineární funkce a bude se moci učit ze komplexnějších vzorců [10].

Při popisu architektury sítě se vrstvy ReLU často neuvádějí, protože v nich neprobíhá žádné učení, takže se pouze nazývají "vrstvou" a často se považují za součást konvoluční vrstvy, protože následují vždy bezprostředně za ní[10].

Do aktivační vrstvy se vloží vstupní objem $W \times H \times D$ a poté se použije aktivační funkce. Výstupní rozměry aktivační vrstvy vždy stejné jako vstupní rozměry, tj. $Win = Wout, Hin = Hout, Din = Dout.$ [11].

Pooling vrstva (Subsampling nebo Down-sampling Layer)

Obvykle se tato vrstva umisťuje do sítě za každou konvoluční, protože pomáhá snížit množství vstupních dat pro další vrstvu, aby se snížil další počet parametrů a operací. Tato vrstva se obvykle vyskytuje po aplikaci nelinearity, například ReLU, na mapu rysů vytvořenou konvoluční vrstvou.

Vrstva pooling se aplikuje na každou mapu rysů zvlášť, což nakonec vede k vytvoření nové mapy rysů pro každý výstup konvoluční vrstvy [11].

Filtr je obvykle 2x2 a má velikost kroku 2, ale může mít i jinou velikost, která bude vždy menší než zadaná mapa rysů [10]. Vrstva pooling tedy vždy zmenší velikost následující mapy rysů na polovinu (Obr. 3), což následně sníží počet hodnot v každé mapě rysů na čtvrtinu původní velikosti[10]. Například při použití vrstvy pooling na mapu rysů 6×6 (celkem 36 pixelů) je výsledkem zmenšená mapa 3×3 (celkem 9 pixelů) [11].

Používají se dva typy sdružování[12]**:**

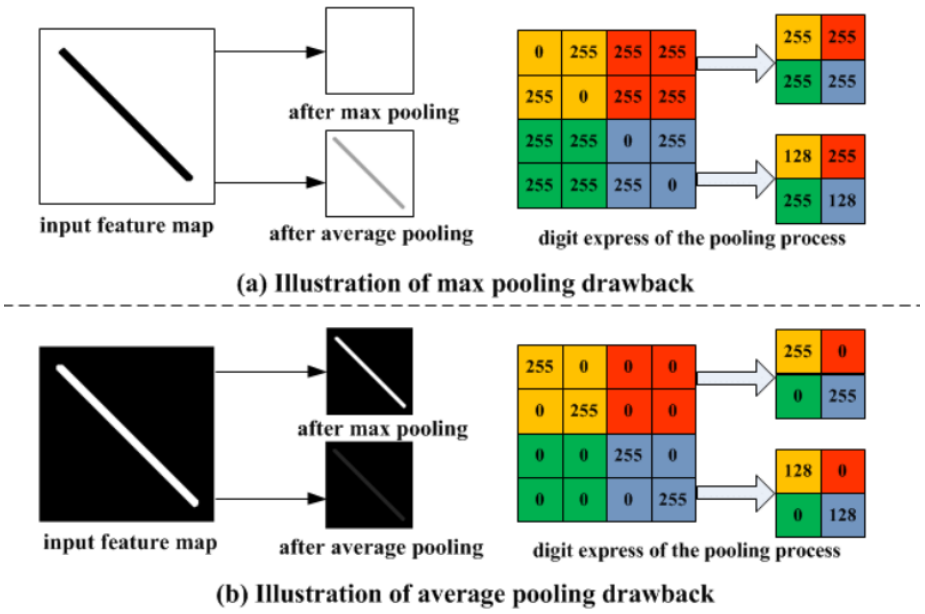
Average pooling:

Podstatou je zmenšení vzorku vstupní reprezentace (obrazu, výstupní matice skryté vrstvy atd.), čímž se sníží její dimenzionalita a bude možné předpokládat vlastnosti obsažené v dílčích oblastech. To se provádí tak, že se vezme průměrná hodnota pixelů v rámci předem definovaného okna (nazývaného "pool size"), které se posouvá napříč vstupní maticí. Operace se aplikuje zvlášť v každém řezu vstupních dat.

Max pooling:

Funguje podobně, ale místo průměru pixelů v oblasti bere maximální hodnotu. To vychází z myšlenky, že nejvýraznější rysy obrazu, jako jsou hrany, mohou být lépe reprezentovány maximální hodnotou v určité oblasti obrazu než průměrem.

Obr. 3 Ilustrace principů sdružování [13]



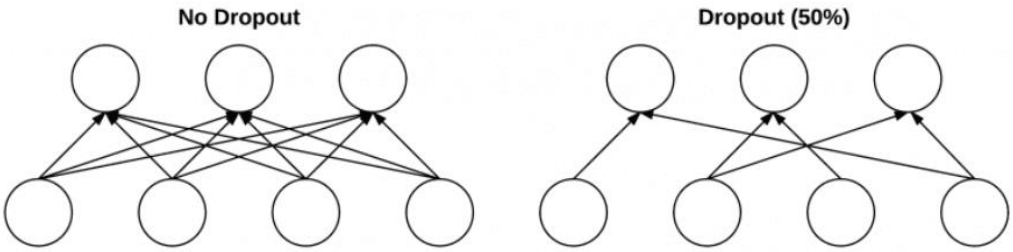
Plně propojená vrstva (Fully Connected Layer, FC Layer)

Plně propojená vrstva předpokládá, že každý neuron je propojen s každým neuronem z předchozí a následující vrstvy. Každé spojení je označeno jinou váhou, aby se rozlišila důležitost uzlu. Tato vrstva v podstatě přijímá vstupní objem (výstup z předchozí vrstvy) a výstupem je N-rozměrný vektor, kde N je počet neuronů ve vrstvě FC. Tento výstup může představovat například skóre kategorií v klasifikační úloze, takže vrstva FC je v mnoha architekturách neuronových sítí klíčovou složkou pro odvození konečného výstupu.

Dropout vrstva

Dropout se používá k dynamické změně architektury sítě během trénování. Dropout náhodně a v náhodné fázi zpracování vypíná některé neurony, aby se model mohl přizpůsobit práci bez nich (Obr. 4).

Obr. 4 Ilustrace použití vrstvy dropout[14]



Dropout řeší přeučení tím, že dynamicky mění architekturu sítě během tréninku [14]. Uzly se zapínají a vypínají náhodně, což modelu umožňuje přizpůsobit se stávající architektuře a naučit se pracovat se sadou filtrů a dat, která jsou v daném časovém období k dispozici.

Vrstva dávkové normalizace (Batch Normalization Layer)**:**

Vrstva slouží k omezení vystupujících dat. Tím se odstraní hodnoty, které se od středních hodnot velmi liší ve větším nebo menším směru, a dosáhne se větší konzistence dat, což vede k vyšší robustnosti systému.

Protože jsou data v této vrstvě normalizovaná, práce dalších vrstev je snazší a rychlejší, protože pracují vždy s podobnými hodnotami a nejsou závislé na tom, jaká data poskytuje předchozí vrstva. Hlavním úkolem této vrstvy je normalizovat a regularizovat data. Toho se dosáhne normalizací dat po částech. Tato vrstva je velmi užitečná a v mnoha

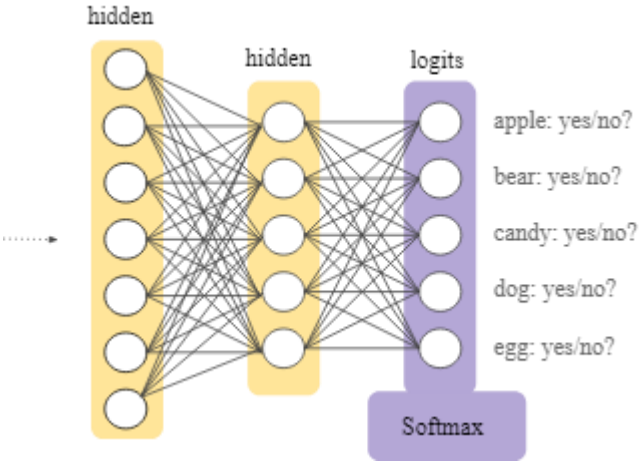
modelech se obvykle umisťuje hned po inicializaci, ale uživatel může tuto vrstvu použít vícekrát, vše závisí na tom, jaké výsledky chce dosáhnout [12].

Softmax vrstva

Hlavním úkolem této vrstvy je zobecnit získané výsledky tak, aby v úlohách, kde je cílem klasifikace, uživatel nepracoval s různými čísly, která se od sebe často velmi liší. Místo toho je při zadávání výsledků do této vrstvy výstupem číslo pravděpodobnosti od 0 do 1, které říká, jak pravděpodobné je, že klasifikovaný objekt obsahuje určité vlastnosti. Každý výsledek je exponenciálně zvýšen a vydělen součtem všech exponentů [15].

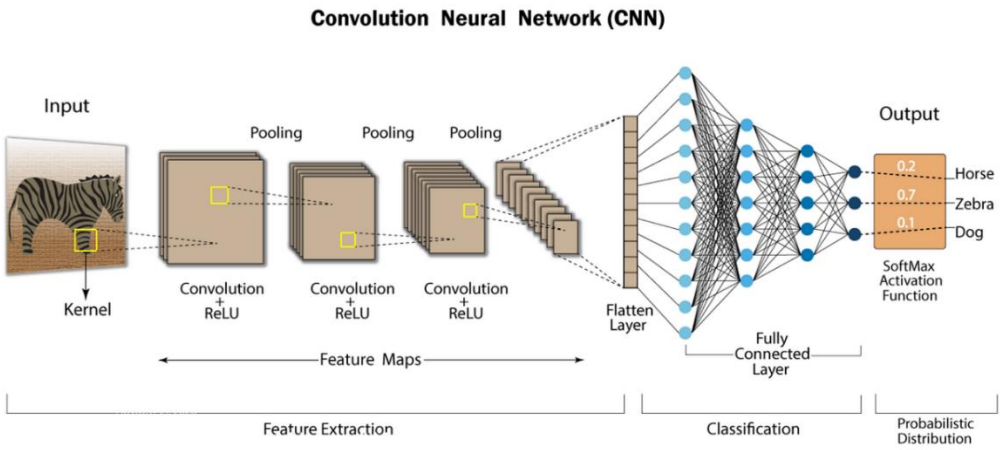
Funkce softmax slouží k převodu výsledných čísel do formátu vhodnějšího pro zpracování. Logity tedy mohou mít libovolnou hodnotu, úkolem funkce softmax je převést tuto hodnotu do běžného tvaru: čísla od 0 do 1. Na vstupu funkce softmax mohou být použita libovolná čísla, ale výstupem bude vždy číslo od 0 do 1. Větší čísla budou převedena na číslo s vysokou pravděpodobností blížící se 1, záporná nebo nejmenší čísla budou převedena na číslo s nižší pravděpodobností (Obr. 5).

Obr. 5 Ilustrace použití vrstvy softmax[15]



Softmax je podobný sigmoidní funkci, takže je považován za zobecnění logistické regrese. Po použití této funkce získá uživatel lepší výsledky, se kterými se lépe pracuje, místo velkých čísel s velkým rozsahem je výstup tvořen čísly od 0 do 1, což je čitelnější a zřetelnější.

Obr. 6 Příklad celého modelu neuronové sítě [16]



2.1.4 Druhy architektury konvolucnich siti

Konvoluční neuronové sítě jsou velkou skupinou neuronových sítí, které mohou mít velmi odlišnou architekturu a sloužit k různým účelům. V zásadě se CNN liší architekturou vrstev, některé architektury jsou implementovány s větší hloubkou (Deep CNN), takže obraz prochází desítkami konvolučních vrstev, některé naopak kladou důraz na jednoduchost a jsou určeny pro jednodušší úlohy. Vývojáři neuronových sítí zpravidla soupeří o rychlost trénování, protože rozsáhlá síť se složitou architekturou vyžaduje velké množství dat a výpočetní kapacity [17].

Například rekurentní neuronové sítě (RCNN) využívají kombinace rekurentních struktur s konvolučními vrstvami, což vede ke schopnosti pozorovat a analyzovat nejen statické obrázky, ale také videa nebo sekvence obrázků[18].

Transformer-based CNN se zaměřují na rozvoj pozornosti modelu tak, aby mohl pracovat s daty poskytnutými spolu s kontextem. Například vyhledávání konkrétních věcí na obrázku nebo generování popisu obrázku. Takové architektury využívají velké množství dat, takže mohou k úloze přistupovat abstraktněji a pracovat s parametry, které nejsou jednoznačně definovány[19].

Vzhledem k velkému množství různých dostupných architektur se CNN stávají stále častěji používaným nástrojem v mnoha oblastech klasifikace dat. Vývojáři začínají stále více upřednostňovat tyto technologie před obvyklými algoritmy strojového učení.

Široce používané architektury neuronových sítí:

- **AlexNet**:

Docela stará architektura představená v roce 2012. Používá pouze 8 vrstev, z nichž pět je konvolučních. Celkově ji lze označit za hlubokou. Proti přetrénování se používají vrstvy ReLu a Droput. Jedná se o jednu ze základních architektur pro práci s obrázky[17].

- **VGGNet**:

VGGNet je Velmi hluboká síť, která využívá 19 vrstev - všechny filtry jsou 3x3, což zdůrazňuje kritičnost hloubky při zpracování, protože malé filtry je často nutné aplikovat opakovaně, aby byl výsledek přesnější.[20].

- **GoogLeNet** (Inception):

Jedna z nejpřesnějších sítí pro práci s obrázky. Hlavní výhodou je modul Inception, který umožňuje dynamicky volit velikost použitého filtru, což zvyšuje hloubku zpracování při zachování rychlosti a výkonu[21].

- **ResNet**:

Sada architektur CNN, která je tvořena skupinou velmi hlubokých sítí. Každá jednotlivá síť je označena číslem, které odpovídá počtu vrstev, například ResNet-50, která má 50 vrstev. Tohoto velkého počtu je dosaženo použitím náhodného přeskakování vrstev. To umožňuje síti učit se různými způsoby, což v konečném důsledku povede ke snížení ztrát a zvýšení přesnosti u sítí s velkým počtem vrstev [22].

- **DenseNet**:

Hlavním charakteristickým prvkem je, že všechny vrstvy jsou plně propojeny se svými sousedy, což pomáhá zlepšit data a gradientní tok. Uživatel také nemusí zadávat velké množství parametrů a celková architektura je jednodušší. Síť je vhodná pro klasifikační úlohy.[22].

2.1.5 Odborné publikace

Dále jsou uvedeny některé příklady výzkumných prací zaměřených na využití hlubokého učení v zemědělství, včetně prací zaměřených přímo na řešení problému detekce poléhání pšenice:

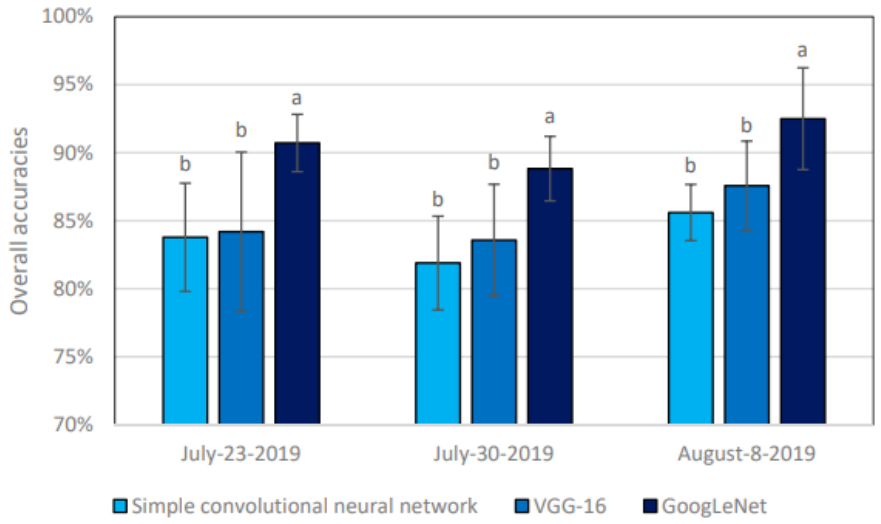
V první studii [23] se porovnává výkonnost standardního strojového učení a hlubokého učení. Výsledkem je stanovení a vzájemné porovnání úrovní směrodatné odchylky a přesnosti. Pro hluboké učení byly použity různé varianty extrakce příznaků, které byly následně využity v několika algoritmech strojového učení.

Ze všech použitých algoritmů byl jako nejlepší vybrán random forest. To bylo možné zjistit až ve fázi porovnání směrodatné odchylky, protože při porovnání přesnosti vykazovaly všechny systémy přibližně stejné výsledky (Tabulka 1).

V případě hlubokého učení byly testovány tři architektury: GoogLeNet, jednoduchá konvoluční síť a VGG-16 (obr. 7). GoogLeNet vykazoval v každém testu nejlepší výsledky, což je vidět v grafu. Nakonec byly GoogLeNet a Random forest porovnány mezi sebou a

vykazovaly přibližně stejné výsledky, ale ve výsledku GoogLeNet překonal algoritmus v přesnosti o 2 procenta (93% proti 91%), což znamenalo jasnou volbu v jeho prospěch. Proto bylo doporučeno použít pro detekci poléhání pšenice systém GoogLeNet[23].

Obr. 7 Srovnání tří různých architektur CNN[23]



Tabulka 1 Srovnání preciznosti algoritmů a neuronových sítí[23]

Performance Metrics	Random Forest	Neural Network	Support Vector Machine	Average
Precision	1.40%	7.83%	4.84%	4.69%
Recall	14.94%	25.00%	22.68%	20.87%
Overall Accuracy	3.23%	6.01%	4.72%	4.66%
F1 Score	8.67%	16.62%	13.13%	12.81%

Další studie [24] byla zaměřena spíše na aspekt práce s různými obdobími růstu a kombinování dat pro detekci.

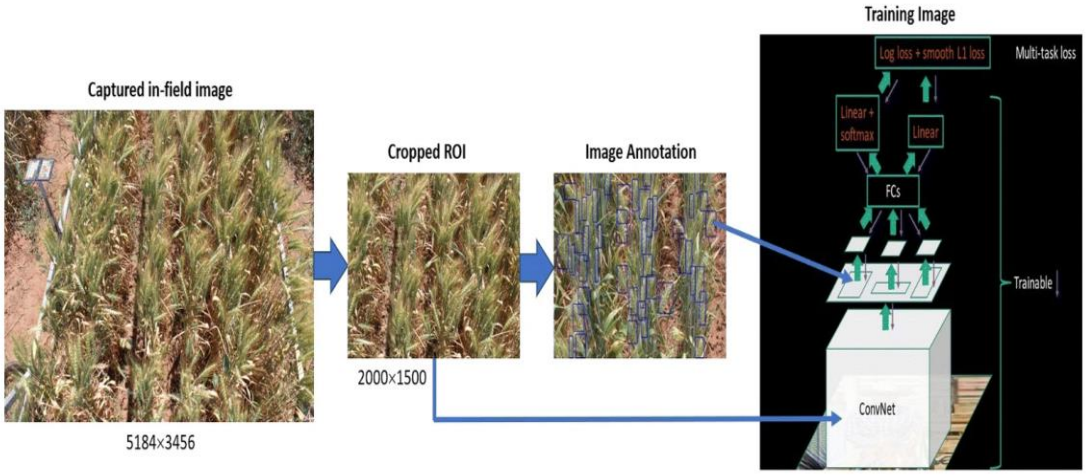
V tomto článku autoři přistupují k problematice výkonnosti neuronových sítí z hlediska dat použitých pro klasifikaci. Byly vycvičeny tři modely CNN, každý s použitím jiné sady dat. Datové sady se lišily tím, že snímky byly pořízeny v různých obdobích růstu pšenice, celkem byla zvolena dvě období. Vznikly tak tři sady dat z jedné fáze růstu, druhé a smíšené sady, kde je možné najít data z první i druhé sady. Výsledky tréninku byly nejlepší u smíšené sady, která vykázala přesnost 89,23%, další dvě sady vykázaly výsledky 53,32% a 84,9% (Tabulka 2). Ačkoli se výsledky jedné ze sad s jednou růstovou fází blížily výsledkům smíšené sady, stále prohrávaly v přesnosti, z čehož plyne, že bohatší sada snímků má kladný vliv na přesnost další klasifikace[24].

Tabulka 2 Srovnání přesnosti modelu na snímcích z různých časových období[24]

		Predicted		Precision (%)	Recall (%)	F_1-Score (%)	OA (%)	Kc
		Lodging	Non-Lodging					
Model_grain filling † and tested at early grain filling stage								
Actual	Lodging	563426	328653	71.79	63.16	67.20	90.22	0.61
	Non-lodging	221430	4508971					
Model_grain filling † and tested at physiological maturity								
	Lodging	569850	5495	9.41	99.04	17.18	14.41	0.01
	Non-lodging	5487168	354986					
Model_physiological maturity † and tested at early grain filling stage								
	Lodging	320180	147047	30.03	68.53	41.79	84.13	0.32
	Non-lodging	745032	4410221					
Model_physiological maturity † and tested at physiological maturity								
	Lodging	588172	320238	49.54	64.75	56.13	85.67	0.48
	Non-lodging	599088	4908403					
Model_both † and tested at early grain filling stage								
	Lodging	620320	271759	65.96	69.54	67.70	89.47	0.61
	Non-lodging	320180	4410221					
Model_both † and tested at physiological maturity								
	Lodging	488492	419918	62.96	53.77	58.01	88.98	0.52
	Non-lodging	287324	5220167					

V další studii [25] byl použit hluboký učící přístup k přesné detekci, počítání a analýze pšeničných klasů pro odhad výnosu.

Obr. 8 Vizualizace architektury[25]



V této práci byla použita rozsáhlá datová sada obrázků ve formátu RGB. Data byla získána fotografováním pšenice na poli, přičemž všechny snímky byly pořízeny ve stejné růstové fázi, což je celkově možné, protože cílem bylo detekovat objekty, aniž by bylo nutné posuzovat nějaké vlastnosti rostlin, a silueta pšenice je vždy přibližně stejná (Obr. 8)

V této práci bylo natrénováno několik oblastních konvolučních sítí (R-CNN), které byly následně použity k počítání pšeničných klasů na obrázku. Pro trénink byl vytvořen soubor dat, ve kterém byl každý klas ručně označen, což umožnilo sítím rozpoznat rysy a naučit se počítat klasy na neanotovaných datech. Výsledkem bylo, že různé sítě dosáhly výsledků v rozmezí 88 až 94 %, což lze považovat za dobrý výsledek vzhledem k tomu, že při pořizování některých snímků nebyly nejlepší povětrnostní a světelné podmínky.

2.1.6 Knihovna Tensorflow

V této práci byla pro vývoj CNN využita knihovna TensorFlow.

Díky své široké funkcionalitě a vysoké efektivitě zaujímá tato knihovna jedno z klíčových míst ve výzkumu a vývoji neuronových sítí. Tento nástroj pomáhá rychle a efektivně trénovat sítě na velkém množství dat, což je užitečné zejména pro nezávislé vývojáře a výzkumníky [28].

Hlavní vlastnosti

- **Komplexní knihovny:**

Framework obsahuje mnoho knihoven, které pomáhají při práci s tensorflow. Například Keras API pomáhá při vytváření neuronových sítí na velmi jednoduché úrovni. Uživateli stačí přidat do modelu všechny potřebné vrstvy, nakonfigurovat je a Keras se postará o zbytek.

- **Použití v klasifikaci obrazů**

Tensorflow se často používá právě pro práci s CNN, protože podporuje jejich vývoj a dokáže poměrně efektivně počítat a analyzovat data.

Tento framework je vhodný pro vývojáře neuronových sítí, kteří mají málo anotovaných dat k trénování. To je možné díky tomu, že některé architektury, jako ResNet a Inception, jsou součástí tensorflow a uživatel je musí pouze nakonfigurovat a uspořádat tok dat.

- **Využití GPU**

Framework je schopen provádět výpočty na grafické kartě, což výrazně urychluje proces trénování sítí, protože grafické karty jsou schopny zpracovávat mnoho operací paralelně. Tensorflow je také schopen automaticky rozdělit zátěž mezi několika grafickými kartami, což také zvyšuje rychlost zpracování dat. [28].

- **Vizualizace s TensorBoard:**

Velkou výhodou je dostupnost integrovaného nástroje pro protokolování a analýzu modelu TensorBoard, který umožňuje vizuální analýzu a zpracování tréninkových dat. Pomocí nástroje TensorBoard lze zobrazovat data v grafu (Obr. 13), zapisovat protokoly do souboru, tisknout a analyzovat statistiky [28].

Obr. 9 Rozhraní TensorBoard[30]

- **Stochastický gradientní sestup (SGD)**

Tensorflow používá různé optimalizační algoritmy, z nichž hlavní je SGD (ostatní typy jsou poddruhy SGD). Tento algoritmus je založen na tom, že model není trénován na všech datech najednou, ale na malých dávkách, které jsou náhodně generovány. Výsledkem použití dávek je častější aktualizace vah a vyšší přesnost trénování [28].

Tensorflow umožňuje trénovat nové modely bez použití předtrénovaných architektur. Uživateli stačí vytvořit model jedním řádkem kódu, nastavit vrstvy a nakonfigurovat kompilaci. Jako výstup uživatel získá model připravený k použití, který může později použít k vyhodnocení obrázků.

3 Cil prace

Cílem této práce je vytvořit algoritmus pro trénování modelu pro rozpoznávání poléhání pšenice. V této práci nebyly použity předtrénované modely, které jsou trénovány na velkém množství dat a mají za cíl lepší rozpoznávání konkrétních objektů. Model má ručně definovanou architekturu vrstev a byl trénován pro konkrétní účel klasifikace obrazu, aby bylo možné lépe pochopit proces vývoje algoritmů hlubokého učení.

Výstupem této práce je model připravený k rozpoznávání poléhání pšenice s vysokou přesností. Přesnost modelu byla hodnocena pomocí standardních parametrů, jako je přesnost a ztrátovost.

4 Materiály a metody

Data poskytla katedra fyziky a jedná se o několik sad snímků pořízených na poli experimentálních ploch společnosti Selgen, v obci Stupice, Praha východ. Snímky byly pořízeny v různém časovém období, takže není specifikováno, v jaké fázi zrání byla pšenice.

K vytvoření snimků byla použita kamera s funkcí hloubkového snímání a klasické RGB. Kamera je připevněná k rámu, který je zase namontován na sečku (Obr. 10). Secí stroj projíždí mezi řádky pšenice a fotografuje porost shora.

Obr. 10 Kamera připevněná k secímu stroji

4.1 Předzpracování a příprava dat

Pokud jde o specifický cíl klasifikace, tedy posouzení, zda je na daném snímku zobrazena polehlá pšenice, či nikoliv, lze tvrdit, že by neměl být kladen důraz na složitou algoritmickou architekturu s mnoha vrstvami, ale spíše na správné a vhodné předzpracování vstupních dat, protože chybí účel identifikovat konkrétní objekt.

Po předzpracování obrazových dat byla použitá knihovna OpenCV. Open Source Computer Vision Library je softwarová knihovna pro počítačové vidění a strojové učení. Je navržena tak, aby poskytovala komplexní sadu nástrojů pro vývoj aplikací počítačového vidění.[26].

Obr. 11 Příklad vstupních obrázků před předzpracováním

Každá sada snímků obsahuje přibližně 2 až 3 tisíce fotografií, mezi nimiž je třeba vybrat polehlé oblasti, které v poskytnutých sadách nejsou tak početné. V důsledku toho bylo provedeno i generování dat, aby se zvýšil počet snímků s polehlou pšenicí. Bylo také provedeno zrcadlení a převracení snímků, což pomohlo několikanásobně zvětšit množinu dat pro trénování, protože počítač rozpozná snímek jako jedinečný a použití stejných, ale převrácených snímků pomůže zabránit přetrénování, ke kterému v běžných případech může dojít v důsledku použití stejných snímků tak, že se některé z nich stanou rozhodujícími při posuzování a model bude přetrénován[6].

Velmi důležitou součástí předzpracování je oddělení užitečných dat od nepotřebných. V tomto případě je to provedeno pomocí knihovny OpenCV, která obsahuje funkci threshold, která se používá k aplikaci prahování na pevnou úroveň na obraz ve stupních šedi (Obr. 12 (II)). Tato operace prochází všechny pixely v obrázku a rozděluje je do dvou skupin: hodnoty menší než prahová a hodnoty nad prahovou hodnotou. Uživatel také definuje číslo, které bude považováno za novou maximální hodnotu. Všechny hodnoty

pod prahovou hodnotou tedy získají novou hodnotu 0 a hodnoty nad prahovou hodnotou získají hodnotu, kterou uživatel určí jako nové maximum. V reprezentaci barev se 255 obvykle považuje za bílou a 0 za černou. Výstupní obrázek tedy bude černobílý. Tato technika je užitečná pro oddělení objektu zájmu od pozadí, zejména v situacích, kdy objekt a pozadí mají odlišné úrovně jasu. Což je v tomto případě viditelné a pšenice je snadno odlišitelná od pozadí. Proto je možné při zkoumání prahové funkce a experimentech s různými úrovněmi nastavit rozpoznávání tak, aby na výsledném snímku zůstaly pouze klásky pšenice.

Po prahování je obraz podroben funkci canny, která detekuje hrany. Funkce Canny v OpenCV je operátor detekce hran, který používá vícestupňový algoritmus k detekci široké škály hran v obrazech[27]. Výsledkem je zachycení ostrých hran a redukce šumu (Obr. 12 (III)). V tomto případě je velmi užitečný, protože na snímcích jsou snadno rozpoznatelné pšeničné klásky. Hlavním problémem při ní je nastavení parametrů tak, aby byly detekovány co nejdelší hrany a bylo snazší určit, zda je pšenice polehlá, nebo ne.

Obrázky jsou na výstupu sjednoceny na velikost 540x960, aby se s nimi lépe pracovalo a zabíraly méně místa na disku.

Obr. 12 Příklad převodu obrázku z rgb na čb jen s okraji

Dále se data rozdělí na části pro trénování a ověřování výsledků. Na začátku je 400 obrázků dobré pšenice a 96 obrázků polehlé. 70% je pořízeno pro účely tréninku a 30% pro validaci. Model je sám o sobě "chytrý" a může pak tyto kousky dat pro tyto účely použít. Pro testování jsou k dispozici jiné snímky než uvedené.

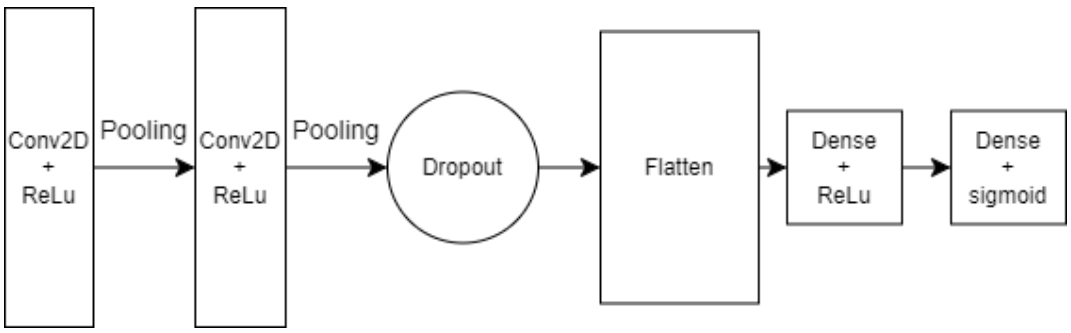
4.2 Popis modelu

Architektura modelu použitá v této práci je vytvořena s ohledem na všechna dříve uvedená pravidla a je co nejjednodušší pro daný případ použití, protože složitější architektura se ukazuje jako přehnaně přizpůsobená a vede k nesprávným výsledkům.

Základní vrstva je sestavena ze dvou konvolučních vrstev, za nimiž následuje aktivace ReLu a pooling (Obr. 13). Po konvolučních vrstvách přichází na řadu droupout. Na konci je umístěna zplošťovací vrstva a dvě zhušťovací vrstvy. Jedna z těchto zhušťovacích vrstev používá aktivaci ReLu a poslední vrstva používá sigmoidu, která v našem případě poskytuje lepší výstup: Takže při vyhodnocování považujeme vše nad 0,5 za polehlé a pod 0,5 za dobré.

Architektura modelu je navržena tak, aby byla účinná a případně snížila možnost přetrénování, což je klíčové pro dosažení vysoké přesnosti při zachování generalizace na neviděná data. Využití dropout vrstvy pomáhá redukovat přizpůsobení na tréninková data tím, že náhodně "vypíná" některé neurony během tréninku, což nutí síť učit se robustnějšími způsoby. Zplošťovací vrstva transformuje výstupní data z předchozích vrstev do jednorozměrného pole, což umožňuje jejich zpracování plně propojenými vrstvami. Zhušťovací vrstvy pak zvyšují abstrakci úrovně informací, které síť zpracovává, a umožňují modelu lépe generalizovat z naučených vzorů na reálné aplikace. Použití aktivace ReLu a sigmoidální funkce je základem pro efektivní trénink a interpretaci výsledků, kde sigmoidální aktivace na výstupní vrstvě umožňuje modelu predikovat pravděpodobnostní skóre pro klasifikační úkoly.

Obr. 13 Ilustrace architektury modelu



4.3 Trénování

Pro účely trénování se používá tensorflow od společnosti Google.

Trénování probíhá v epochách, takže program bere dávky dat a zpracovává je, přičemž uživatel získává výsledky přesnosti a ztrát pro každou epochu a může určit, zda je s výsledky spokojen, nebo zda nepoužívá příliš mnoho nebo málo epoch.

Ztráta je měřítkem toho, jak dobře si model vede na trénovacích datech. Vyjadřuje rozdíl mezi předpovídaným výstupem modelu a skutečným výstupem (ground truth) pro každý trénovací příklad.

Přesnost je měřítkem toho, jak dobře model předpovídá správný výstup pro daná vstupní data. Vypočítá se jako poměr počtu správně předpovězených příkladů k celkovému počtu příkladů.

4.4 Popis softwarového řešení

Trénovací program se skládá z jednoho skriptu v jazyce python, který přijima json se správnými cestami ke složkám. Zdrojový kód řešení je k dispozici na githubu: https://github.com/cortexpick7/deepLearningB

Program načte data a rozdělí je na dávky, z nichž každá se skládá z 32 obrázků. Každou dávku pak rozdělí na validační a tréninková data. Bude to provedeno pomocí iterátoru dat NumPy. NumPy je knihovna pro programovací jazyk Python a další, která slouží k efektivní práci s velkými multidimenzionálními poli a maticemi[30].

Model je vytvořen pomocí metody `Sequential()` z tensorflow, pomocí ní je vytvořen sekvenční model s vrstvami následujícími za sebou. Poté jsou vrstvy do modelu přidávány postupně. Uživatel definuje, kolik vrstev potřebuje a jaké tyto vrstvy jsou. V našem případě jsou to dvě konvoluční vrstvy, po každé následuje vrstva pooling a po ní jedna vrstva dropuot, jedna vrstva flattening a jedna vrstva densing.

Pak je model připraven ke kompilaci a zavolá se na něm metoda compile(). V metodě `compile()` může uživatel definovat různé parametry, mezi nejdůležitější patří optimalizační funkce(v dané funkci se používá algoritm Adam), konfigurace ztrát a typy metrik(v tomto modelu se používá přesnost).

Poté je model připraven k trénování a je na něm zavolána metoda `fit()`. Tato metoda vygeneruje data o tréninku, která lze později použít pro vizualizaci a analýzu výstupu tréninku. V této metodě je také třeba vyplnit několik parametrů, z nichž nejdůležitější jsou: data pro trénování, data pro validaci, množství tréninkových epoch a záznamy, kam může uživatel umístit obsluhu zpětného volání TensorBoard pro ukládání záznamů o trénování.

Po tréninku se pomocí knihovny matplotlib zobrazí diagramy s údaji o přesnosti. Uživateli je pak nabídnuto uložení modelu s názvem, který si může zvolit. Matplotlib je knihovna pro Python, která se používá pro 2D grafiku. Umožňuje vytvářet statické, interaktivní a animované vizualizace v Pythonu[31].

Pro další použití se pak využívá skript `run-evaluation.py`. Tento skript jednoduše přijímá cestu k obrázku a cestu k modelu, který bude použit. Na modelu je zavolána metoda `predict()`, která spotřebuje obrázek a vrátí desetinné číslo mezi 0 a 1, v tomto případě platí, že čím více se číslo blíží 1, tím je pšenice pravděpodobněji polehlá.

5 Výsledky trénování

Trénování probíhá na sadě 496 obrázků, takže počet epoch není velký. Data jsou však také jednoduchá, takže model vykazuje dobré výsledky i při malém množství dat.

Při postupu model vykazuje každou iteraci lepší výsledky a v epoše 7 dosahuje přesnosti 0,955. Ztráta se také každou epochou snižuje a na konci provádění se blíží nule (Tabulka 3).

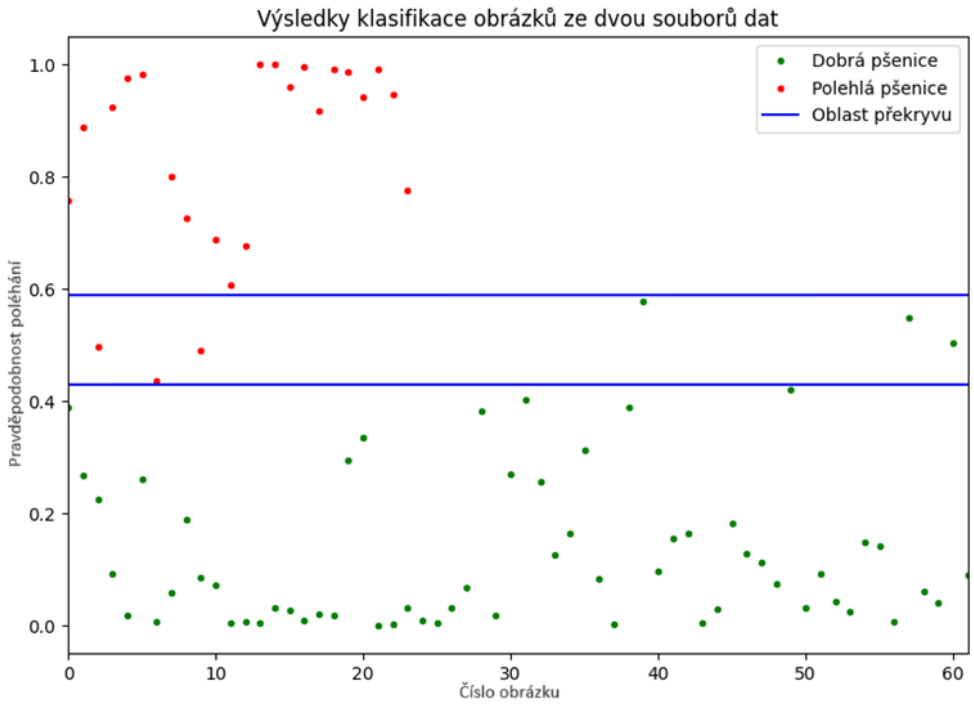
Tabulka 3 Výsledky tréninku

Epocha	Ztráta	Přesnost	Val_ztráta	Val_přesnost
1	0.6098	0.7812	0.9272	0.3958
2	0.8482	0.7656	1.1886	0.7812
3	0.8655	0.7865	0.2528	0.8854
4	0.3181	0.9115	0.3430	0.9896
5	0.3055	0.9688	0.2274	0.9896
6	0.2252	0.9323	0.1837	0.9167
7	0.1609	0.9479	0.0779	1.0000

6 Validace výsledků a diskuze

Test byl proveden na 62 snímcích dobré pšenice a 25 snímcích polehlé. Každý hromadný test ukázal dobrou přesnost, která však neodpovídá výsledku, který ukázal tensorflow. To může být způsobeno tím, že některé obrázky mohou obsahovat jak části polehlé, tak i nepolehlé pšenice a model se nemůže rozhodnout, kterou odpověď vybrat. I když při pohledu na grafy je jasně vidět, že i nesprávné odpovědi vykazují ve většině případů čísla blízká 0,5. V případě skutečného použití lze tedy stanovit prahovou hodnotu a k rozhodnutí, zda obrázek obsahuje nebo neobsahuje polehlou pšenici, je nutná interakce člověka. Obrázky se také pořizují velmi rychle a sousední obrázky vždy zobrazí koordináty, pokud na obrázku nějaké jsou, a podle toho se uživatel může rozhodnout, na které části pole došlo k polehnutí.

Obr. 14 Výsledek hromadné zkoušky na obrázcích



Z výsledků je jasně patrné, že výsledky se vždy blíží meznímu číslu, v případě dobré pšenice jsou čísla blíže 0 než 0,5, v případě polehlé jsou čísla blíže 1. Některé výsledky, které se odchylují od požadovaných čísel, jsou zobrazeny v zóně překrytí, takže uživatel může definovat hranici, od které jsou obrázky považovány za dobré nebo polehlé pouze pomocí algoritmu.

Celkově výsledky testování a následná analýza potvrzují, že použitý model strojového učení je silným nástrojem pro klasifikaci snímků v zemědělství, a to i přes určité výzvy spojené s interpretací a aplikací v reálném světě. Tyto poznatky mohou sloužit jako základ pro další výzkum a vývoj v této oblasti, s cílem dále zlepšit přesnost a užitečnost technologie pro zemědělské aplikace.

Výsledky taky ukazují, že příprava dat hraje při trénování modelu neuronové sítě významnou roli. Model přijímá předzpracované obrázky, na kterých jsou jasně viditelné hrany k detekci, a proto se povedlo model natrénovát na malém počtu obrázků s vysokou přesností.

7 Závěr

Implementovaný model pro rozpoznávání polehlé pšenice se ukázal jako účinný nástroj pro klasifikaci obrazu a teoreticky by mohl být použit v reálném prostředí s minimálním lidským dohledem. Model prokázal schopnost rychle a efektivně se učit a provádět detekci, což dokazují výsledky testů.

V budoucnu by bylo možné tento model vylepšit a rozšířit, především zvýšením množství a rozmanitosti dat, aby bylo dosaženo většího zobecnění a robustnosti modelu.

Tato práce přispívá k oboru precizního zemědělství tím, že poskytuje podrobnou analýzu architektury a principů konvolučních neuronových sítí a příklad jejich implementace.

Výzkum provedený v této práci ukazuje účinnost CNN při přesné identifikaci polehlé pšenice, což může přispět k optimalizaci výnosů.

8 Seznam použité literatury

[1] M. J. Pinthus, „Lodging in Wheat, Barley, and Oats: The Phenomenon, its Causes, and Preventive Measures", *Advances in Agronomy*, roč. 25, č. C, s. 209–263, led. 1974, doi: 10.1016/S0065-2113(08)60782-8.

[2] R. Guo, X. Zhu, a T. Liu, „Automatic detection of crop lodging from multitemporal satellite data based on the isolation forest algorithm", *Comput Electron Agric*, roč. 215, pro. 2023, doi: 10.1016/j.compag.2023.108415.

[3] Berry PM „Lodging". Viděno: 28. únor 2024. Dostupné z: https://farmpep.net/topic/lodging

[4] A. Azizi *et al.*, „Comprehensive wheat lodging detection after initial lodging using UAV RGB images", *Expert Syst Appl*, roč. 238, s. 121788, bře. 2024, doi: 10.1016/J.ESWA.2023.121788.

[5] Y. Niu, T. Chen, C. Zhao, a M. Zhou, „Lodging prevention in cereals: Morphological, biochemical, anatomical traits and their molecular mechanisms, management and breeding strategies", *Field Crops Res*, roč. 289, s. 108733, pro. 2022, doi: 10.1016/J.FCR.2022.108733.

[6] M. Xin a Y. Wang, „Research on image classification model based on deep convolution neural network", *EURASIP J Image Video Process*, roč. 2019, č. 1, s. 1–11, pro. 2019, doi: 10.1186/S13640-019-0417-8/TABLES/2.

[7] L.-H. Lim, „Tensors and Hypermatrices" December 2013 doi:10.1201/b16113-19.

[8] Y. Ji, Q. Wang, X. Li, a J. Liu, „A Survey on Tensor Techniques and Applications in Machine Learning", *IEEE Access*, roč. 7, s. 162950–162990, 2019, doi: 10.1109/ACCESS.2019.2949814.

[9] J. Wu, „Introduction to Convolutional Neural Networks" Nanjing University, China, 2017.

[10] J. Murphy, „An Overview of Convolutional Neural Network Architectures for Deep Learning" Microway, Inc 2016.

[11] M. D. Zeiler a R. Fergus, „Visualizing and Understanding Convolutional Networks arXiv:1311.2901v3 [cs.CV] 28 Nov 2013", *Computer Vision–ECCV 2014*, roč. 8689, č. PART 1, s. 818–833, 2014, doi: 10.1007/978-3-319-10590-1_53.

[12] F. Tu, S. Yin, P. Ouyang, S. Tang, L. Liu, a S. Wei, „Deep Convolutional Neural Network Architecture with Reconfigurable Computation Patterns", *IEEE Trans Very Large Scale Integr VLSI Syst*, roč. 25, č. 8, s. 2220–2233, srp. 2017, doi: 10.1109/TVLSI.2017.2688340.

[13] R. Qayyum „Introduction To Pooling Layers In CNN – Towards AI". Viděno: 28. únor 2024. Dostupné z: https://towardsai.net/p/l/introduction-to-pooling-layers-in-cnn

[14] A. Rosebrock „Convolutional Neural Networks (CNNs) and Layer Types - PyImageSearch". 2021. Viděno: 28. únor 2024. Dostupné z: https://pyimagesearch.com/2021/05/14/convolutional-neural-networks-cnns-and-layer-types/

[15] „Multi-Class Neural Networks: Softmax | Machine Learning | Google for Developers". Viděno: 28. únor 2024. [Online]. Dostupné z: https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax

[16] N. Shahriar „What is Convolutional Neural Network — CNN (Deep Learning) | by Nafiz Shahriar | Medium". Viděno: 28. únor 2024. Dostupné z: https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5

[17] N. Aloysius a M. Geetha, „A review on deep convolutional neural networks", *Proceedings of the 2017 IEEE International Conference on Communication and Signal Processing, ICCSP 2017*, roč. 2018-January, s. 588–592, čvc. 2017, doi: 10.1109/ICCSP.2017.8286426.

[18] R. Girshick, „Fast R-CNN". s. 1440–1448, 2015. Viděno: 28. únor 2024. Dostupné z: https://github.com/rbgirshick/

[19] A. Khan *et al.*, „A survey of the vision transformers and their CNN-transformer based variants", *Artificial Intelligence Review 2023 56:3*, roč. 56, č. 3, s. 2917–2970, říj. 2023, doi: 10.1007/S10462-023-10595-0.

[20] U. Muhammad, W. Wang, S. P. Chattha, a S. Ali, „Pre-trained VGGNet Architecture for Remote-Sensing Image Scene Classification", *Proceedings - International Conference on Pattern Recognition*, roč. 2018-August, s. 1622–1627, lis. 2018, doi: 10.1109/ICPR.2018.8545591.

[21] C. Szegedy *et al*., "Going deeper with convolutions,"*2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 2015, pp. 1- 9,doi:10.1109/CVPR.2015.7298594.

[22] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 2261-2269, doi: 10.1109/CVPR.2017.243.

[23] Z. Zhang, P. Flores, C. Igathinathane, D. L. Naik, R. Kiran, a J. K. Ransom, „Wheat Lodging Detection from UAS Imagery Using Machine Learning Algorithms", *Remote Sensing 2020, Vol. 12, Page 1838*, roč. 12, č. 11, s. 1838, čer. 2020, doi: 10.3390/RS12111838.

[24] B. Zhao *et al.*, „Automatic Wheat Lodging Detection and Mapping in Aerial Imagery to Support High-Throughput Phenotyping and In-Season Crop Management", *Agronomy 2020, Vol. 10, Page 1762*, roč. 10, č. 11, s. 1762, lis. 2020, doi: 10.3390/AGRONOMY10111762.

[25] G. Cheng, J. Han, a X. Lu, „Remote Sensing Image Scene Classification: Benchmark and State of the Art", *Proceedings of the IEEE*, roč. 105, č. 10, s. 1865–1883, říj. 2017, doi: 10.1109/JPROC.2017.2675998.

[26] G. Bradski. (2000). The OpenCV Library. *Dr. Dobbs Journal of Software Tools*.

[27] Brahmbhatt, S. (2013). Embedded Computer Vision: Running OpenCV Programs on the Raspberry Pi. In: Practical OpenCV. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4302-6080-6_11

[28] Pang, B., Nijkamp, E., & Wu, Y. N. (2020). Deep Learning With TensorFlow: A Review. Journal of Educational and Behavioral Statistics, 45(2), 227-248. https://doi.org/10.3102/1076998619872761.

[29] „TensorBoard- A Visualization suite for Tensorflow models | by Renu Khandelwal | Towards Data Science". Dostupné z: https://towardsdatascience.com/tensorboard-a-visualization-suite-for-tensorflow-models-c484dd0f16cf

[30] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.

[31] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007