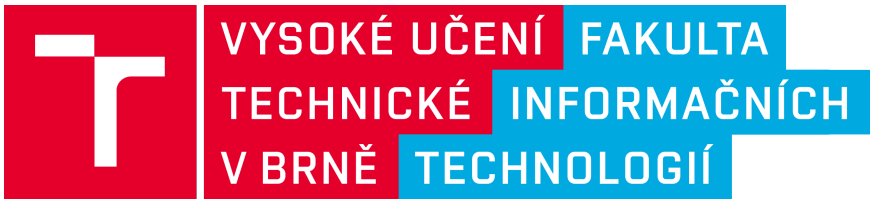


Soft Computing – Projekt

Technická zpráva

1 – Demonstrace učení BP a vlivu optimalizace, regularizace nebo dropoutu

2024/2025 Dalibor Kříčka (xkrick01)

1 Úvod

Cílem tohoto projektu je implementovat jednoduchou neuronovou síť klasifikující ručně psané číslice a demonstrovat její učení pomocí metody *backpropagation*. Učení bude demonstrováno prostřednictvím aplikace s grafickým rozhraním a grafů vyobrazujících hodnoty ztrátové funkce a přesnost trénovaného modelu v jednotlivých epochách. Účelem je, kromě samotného učení, prezentovat vliv *optimalizačních algoritmů*, vliv techniky *dropout* a klasifikaci uživatelem napsaných číslic konkrétními natrénovanými modely.

2 Data, neuronová síť a backpropagation

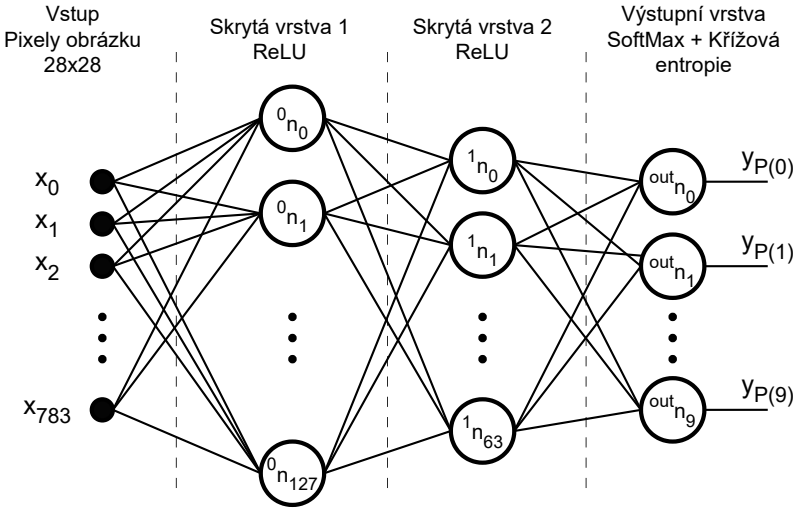
2.1 Dataset MNIST

Pro učení neuronové sítě byl využit již existující dataset *MNIST*, který obsahuje 60000 trénovacích obrázků ručně psaných číslic s pevnými rozměry 28x28 pixelů. Testovacích dat je 10000 vzorků. Ke každému vzorku z databáze MNIST je poskytnuto požadované výstupní ohodnocení pro daný obrázek, tedy jakou číslici obrázek obsahuje. Proto, aby pro trénovaný model byla zachována vlastnost *generalizace*, byly trenovací a testovací ručně psané číslice psány několika sty lidí. Množina lidí podílejících se na tvorbě trénovacích dat je disjunktní s množinou lidí podílejících se na tvorbě testovacích dat, aby byla přesnost modelu testována na dosud nespatřených datech (viz [1]).

Pixely obrázků původní datové sady jsou reprezentovány hodnotami 0 až 255, které představují odstín šedi barvy pixelu, avšak pro účely tohoto projektu a pozdější demonstraci klasifikace uživatelem napsaných číslic se všechny nenulové obrazové body převádí na hodnotu 1. Prakticky jsou tedy body obrázků označeny jako obarvený (hodnota 1) nebo neobarvený (hodnota 0).

2.2 Struktura neuronové sítě

V rámci řešení projektu je implementována dopředná neuronová síť (acyklická, vazby pouze mezi sousedními vrstvami), které je kladeno za cíl klasifikovat obrázek s ručně psanou číslicí na vstupu. Vstupem je tedy 784 hodnot pixelů (0 nebo 1) obrázku a výstupem pravděpodobnost pro každou číslici 0 až 9, se kterou je dle trénovaného modelu tato číslice na vstupu vyobrazena. Síť se skládá ze dvou skrytých vrstev s aktivační funkcí typu *ReLU*. První vrstva zahrnuje 128 neuronů a druhá 64 neuronů. Výstupní vrstva implementuje aktivační funkci *SoftMax* a obsahuje právě 10 neuronů jakožto pravděpodobnostní výstupy pro jednotlivé číslice. V celé síti se využívá *lineární bázová funkce*. Jako objektivní funkce je v modelu použita *křížová entropie* (ztrátová funkce L). Struktura sítě je vyobrazena grafem na obrázku 1.



Obrázek 1: Schéma neuronové sítě pro klasifikaci ručně psaných číslic.

ReLu

Aktivační funkce *ReLU* pro skryté vrstvy zachovává kladné vstupy a nuluje vstupy negativní.

$$ReLU(x) = max(0, x)$$

Poskytuje lepší propagaci gradientu a celkově rychlejší konvergenci, protože se vyhýbá problému mizejícího gradientu. Derivace aktivační funkce pro výpočet gradientu má následující tvar (viz [3]).

$$\delta_j = \begin{cases} 1 & \text{pokud } x_j > 0 \\ 0 & \text{pokud } x_j \leq 0 \end{cases}$$

SoftMax a křížová entropie

Aktivační funkce *SoftMax* umožňuje řešit klasifikační problémy více tříd (v tomto případě 10 tříd) takovým způsobem, že pro každý výstup stanoví určitou pravděpodobnost. Součet výstupů je tedy roven 1. SoftMax je běžně využívaná pro výstupní vrstvy klasifikačních neuronových sítí.

$$SoftMax(x_j) = \frac{e^{x_j}}{\sum_{k=0}^{9} e^{x_k}}$$

Ztrátová funkce (křížová entropie), kterou se model snaží svým učením minimalizovat, lze díky kombinaci s aktivační funkcí SoftMax spočítat následovně (za předpokladu, že je vždy přepokládaný právě jeden výstup s pravděpodobností 1 a ostatní s pravděpodobností 0).

$$L = -\ln(y_{predicted})$$

Vzorec pro výpočet gradientu pro zpětnou propagaci má tento tvar.

$$\delta_j = \begin{cases} 1 - y_j & \text{pokud } j = predicted \\ -y_j & \text{pokud } j \neq predicted \end{cases}$$

Inicializace vah a biasů

Inicializační hodnoty vah skrytých vrstev s aktivační funkcí ReLU vychází z tzv. *He* inicializace. Každé váze je tedy před začátkem trénování přiřazena hodnota z normálního rozdělení $N(0, \frac{2}{n_{in}})$, kde n_{in} je počet neuronů předchozí vrstvy (viz [2]).

Inicializační hodnoty vah pro výstupní vrstvu s aktivační funkcí SoftMax jsou počítány za pomocí tzv. *Xavier* inicializace. Každé váze je tedy před začátkem trénování přiřazena hodnota z normálního rozdělení $N(0, \frac{2}{n_{in}+n_{out}})$, kde n_{in} je počet neuronů předchozí vrstvy a n_{out} je počet neuronů aktuální vrstvy (viz [2]). Hodnoty biasů všech neuronů v síti jsou po inicializační fázi nulové.

2.3 Backpropagation

Backpropagation je algoritmus pro učení neuronových sítí prostřednictvím zpětné propagace gradientu ztrátové funkce. Učením je obrazně myšlena úprava vah a biasů, tak aby byl model schopen produkovat správné výsledky i pro doposud neviděná data (tzv. generalizace). Cílem učení je minimalizovat ztrátovou funkci pomocí gradientního sestupu. Gradientní sestup je metoda, kdy jsou pomocí gradientů neuronů určeny směry nejrychlejšího klesání ztrátové funkce, na jejichž základě jsou upraveny parametry sítě.

MBGD - Mini-Bacth Gradient Desdent

MBGD je přístup k realizaci gradientního sestupu takový, že se změna parametrů sítě provede až po zpracování předem určného počtu náhodně vybraných prvků trénovací datové sady. Právě tento přístup byl zvolen pro realizaci tohoto projektu. MBGD je kompromisem mezi rychlou konvergencí (SGD) a stabilním klesáním k minimu funkce (MGD). Velikost dávky (*bacth size*) má vliv na průběh učení modelu. Menší

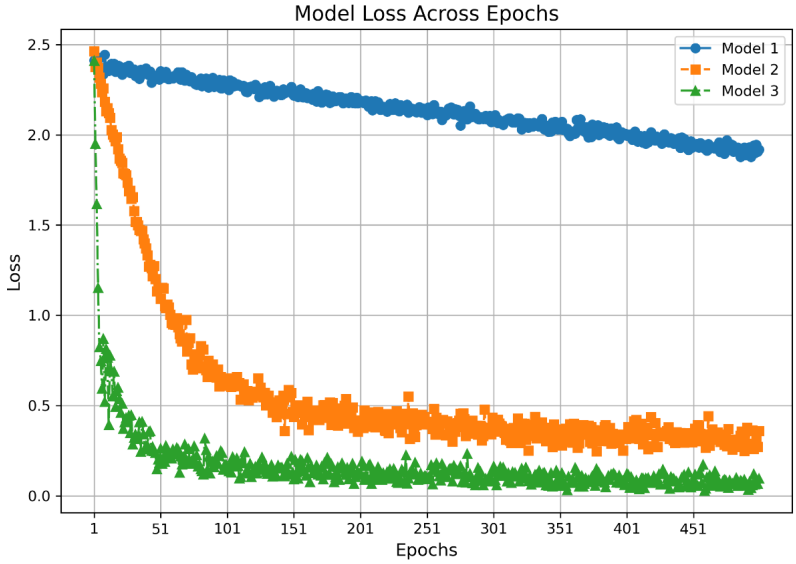
dávky jsou výpočetně méně náročné a dochází k rychlé konvergenci, ale dochází k větším oscilacím po cestě k minimu ztrátové funkce a pro větší dávky opačně.

Adam

Adam je optimalizační algoritmus, který dynamicky upravuje koeficient učení pro každý parametr sítě kombinací principů hybnosti (momentum) a měnícího průměru kvadrátu gradientů. V rámci trénování představené neuronové sítě Adam vykazuje výrazné zlepšení konvergence oproti trénování bez optimalizace. Nicméně jeho konvergence je oproti AMSGrad pomalejší a navíc dosahuje vyšší konečné ztráty (viz obrázek 2). Adam je tedy výkonný, ale ani v dlouhém běhu nemusí být tak stabilní jako AMSGrad.

AmsGrad

AmsGrad vychází z algoritmu Adam s tím rozdílem, že AmsGrad využívá výběru *maxima* minulého a současného kvadrátu gradientů. AMSGrad dosahuje nižší ztráty mnohem rychleji a stabilněji než Adam, což potvrzuje jeho schopnost lépe zvládnout přetrvávající problémy, jako je přeskakování optimálních bodů. Při trénování dříve zmíněné neuronové sítě se ukazuje AMSGrad jednoznačně nejvýkonnější, jelikož i při dlouhodobém tréninku udržuje stabilní nízkou ztrátu (viz obrázek 2).



Obrázek 2: Vývoj ztráty v průběhu učení neuronové sítě pro jednotlivé optimalizační algoritmy (Model 1 – bez optimalizace, Model 2 – Adam, Model 3 – AmsGrad), kde koeficient učení = 0.001 a batch size = 128.

Dropout

Dropout je technika, která s určitou pravděpodobností vyřadí během trénování vstupy nebo neurony vnitřních vrstev, kdy v každém kroku je vyřazení vyhodnoceno znovu. Tato technika slouží k zabránění přetrénování neuronové sítě. Vliv dropoutu se při trénovaní sítě, která je předmětem tohoto projektu, projeví především tak, že modely trénované s dropoutem dosahují vyšší přesnosti na doposud neviděných datech při vyšších hodnotách ztrátové funkce.

3 Instalace a ovládání aplikace

3.1 Instalace, stažení dat a spuštění

Nainstalovat nezbytné python knihovny (numpy a matplotlib) a stáhnout datové sady pro trénování modelů je možné spuštěním skriptu `install_and_run.sh`, který aplikaci poté spustí (skript pro instalaci

knihoven vyžaduje autentizaci heslem). Pro samotné spuštění aplikace stačí využít skript `run.sh`.

3.2 Ovládání aplikace

Aplikace s grafickým uživatelským rozhraním byla implementována v jazyce Python pomocí knihoven *numpy* a *PyQt5*. Aplikace má dvě hlavní okna *Train* a *Classify*.

Okno Train

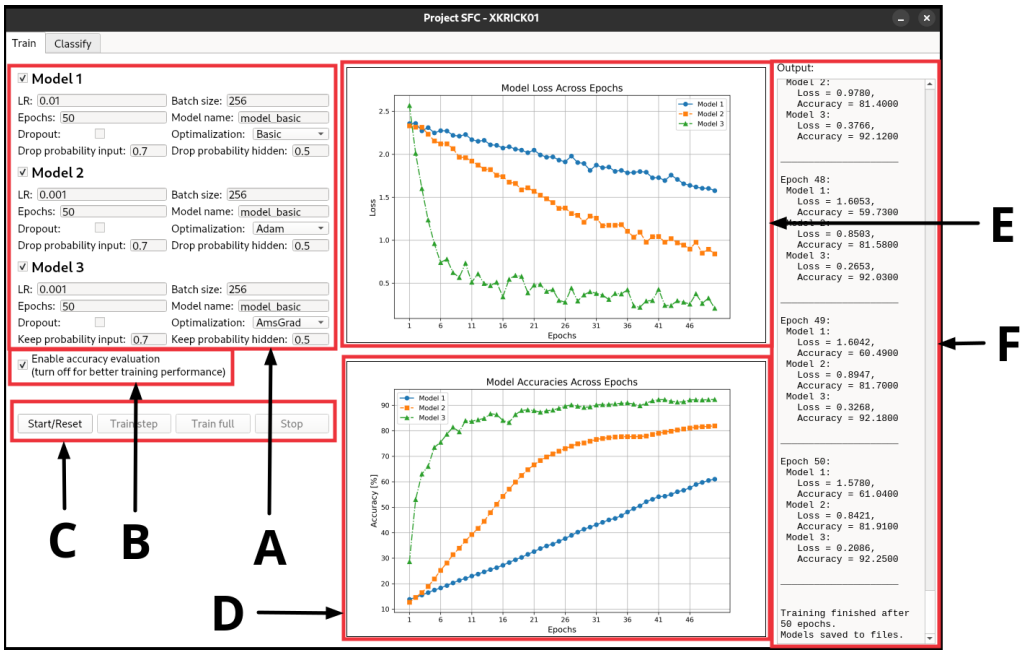
Okno *Train* umožňuje uživateli natrénovat modely dle zvolených parametrů, včetně krokování trénování. Mějme toto okno rozděleno na sekce A – F, viz obrázek 3, jejichž funkce je popsána následovně:

- **A** – Nastavení parametrů až pro tři modely, které budou trénovány zároveň v jednom trénovacím běhu. Možnost deaktivace jednolivých modelů pro daný trénovací běh (zaškrtávací políčko vedle „Model X“). Pro každý model je možné nastavit: koeficient učení (*LR*), velikost trénovací dávky (*Batch size*), počet trénovacích epoch (*Epochs*), vlastní název modelu, pod kterým bude po natrénování uložen do souboru (*Model name*), optimalizační algoritmus (*Optimalization*), zapnutí/vypnutí techniky dropout (*Dropout*), pravděpodobnosti deaktivace vstupů a neuronů vnitřních vrstev (*Drop probability input a hidden*).
- **B** – Možnost povolit vykreslování grafu přesnosti modelů v jednotlivých epochách (viz sekce D). Zapnutí této možnosti však výrazně zpomaluje trénování, kvůli vyhodnocování přesnosti modelů na testovacích datech v každém kroku učení.
- **C** – Ovládací tlačítka řídící demonstraci učení: *Start/Reset* – resetuje výsledky z předchozího trénování a inicializuje modely dle nastavení parametrů v sekci A, *Train step* – vyhodnotí jednu epochu všech aktivních modelů a trénování zastaví, *Train full* – spustí trénování všech aktivních modelů od aktuální epochy až do stanoveného počtu epoch (do konce) a uloží modely do souborů, *Stop* – ukončí trénování v aktuální epoše a uloží modely do souborů.
- **D** – Graf prezentující úspěšnost klasifikace všech aktivních modelů na testovacích datech během jednotlivých epoch.
- **E** – Graf prezentující hodnoty ztrátových funkcí všech aktivních modelů během jednotlivých epoch.
- **F** – Textový výstup aplikace poskytující konkrétní hodnoty, které jsou vyobrazeny v grafech.

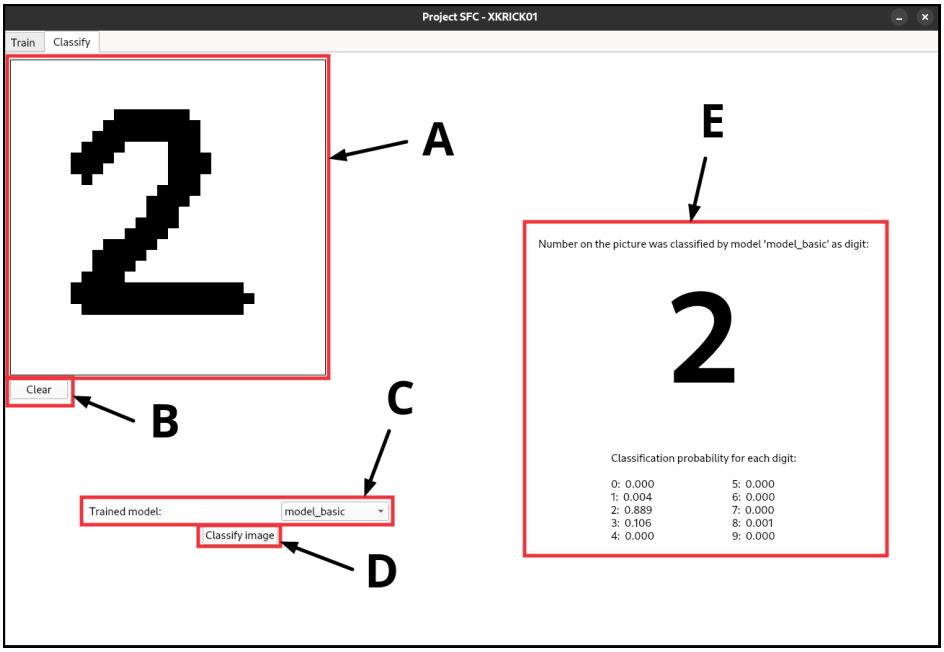
Okno Classify

Okno *Classify* uživateli umožňuje nakreslit přímo v aplikaci myší číslici a poté ji klasifikovat vybraným natrénovaným modelem. Mějme toto okno rozděleno na sekce A – E, viz obrázek 4, jejichž funkce je popsána následovně:

- **A** – Plátno, na které je možné stiskem pravého tlačítka myši a následným tahem psát číslice pro následnou klasifikaci. Levé tlačítko myši je možné použít jako „gumu“.
- **B** – Tlačítko pro vyčištění plátna a výsledků klasifikace.
- **C** – Možnost výběru dříve natrénovaného modelu pro klasifikaci podle jeho názvu.
- **D** – Tlačítko pro vyhodnocení klasifikace číslice napsané na plátně pomocí zvoleného modelu.
- **E** – Výsledky klasifikace.



Obrázek 3: Náhled okna *Train* rozděleného na dílčí UI sektory A – F.



Obrázek 4: Náhled okna *Classify* rozděleného na dílčí UI sektory A – E.

Literatura

[1] LECUN, Y., CORTES, C. et al. *THE MNIST DATABASE of handwritten digits* [online]. 1998 [cit. 2024-11-29]. Dostupné z: https://yann.lecun.com/exdb/mnist/index.html.

[2] MAHESHKAR, S. *A Gentle Introduction To Weight Initialization for Neural Networkss* [online]. 16. ledna 2024 [cit. 2024-11-30]. Dostupné z: https://wandb.ai/sauravmaheshkar/initialization/reports/A-Gentle-Introduction-To-Weight-Initialization-for-Neural-Networks--Vmlldzo2ODExMTg.

[3] SAKSHI_TIWARI. *Activation functions in Neural Networks* [online]. 19. listopadu 2024 [cit. 2024-11-29]. Dostupné z: https://www.geeksforgeeks.org/activation-functions-neural-networks/#1-linear-activation-function.