**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE**

**FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

**STROJOVÉ VIDENIE A VÝPOČTOVÁ INTELIGENCIA**

**NEURÓNOVÁ SIEŤ V PYTHONE**

**ZADANIE 1**

|  |  |
| --- | --- |
| **11/2024** | **Jakub Hubáček** |

**Zadanie:**

Vytvorte a overte neurónovú sieť (NS), ktorá bude schopná rozpoznávať znaky číslic od 0 po 9, s nasledovnými parametrami:

* Vstupom do NS je mriežka buniek 4x7 (napr. CCD snímač), t.j. NS má 28 vstupov s rozsahom , kde hodnota 0 predstavuje biele a 1 čierne políčko. Medzi tým sú stupne šedej farby.
* NS má 10 výstupov s rozsahom , kde každý výstup prislúcha jednej rozpoznanej číslici. Hodnotu 1 môže nadobúdať práve jeden výstup.

NS sa natrénuje na vstupno-výstupných dátach pre číslice 0 až 9 zakódovaných do 28-prvkových riadkových vektorov:  
Vstup: znak 1 = [0 0 1 0 0 1 1 0 1 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 ]  
Výstup: znak 1 = [1 0 0 0 0 0 0 0 0 0]  
...  
Vstup: znak 0 = [0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 0 1 1 0 ]  
Výstup: znak 0 = [0 0 0 0 0 0 0 0 0 1]

**Úlohy:**

1. Experimentálne zistite, aké veľké porušenie obrazovej informácie ešte NS zvládne, t.j. nájdite prípady, kde sieť znak nerozpozná. Zašumením sa chápu iné hodnoty než 0 alebo 1. Porušením sa chápe, že niektoré prvky matice sú vymenené, posunuté alebo majú doplnkové hodnoty.
2. Overte vplyv rôznych architektúr neurónovej siete (iná aktivačná funkcia, iný počet neurónov v skrytej vrstve, iný počet skrytých vrstiev...).

**Riešenie:**

**Tvorba trénovacieho datasetu:**

A screenshot of a computer

Description automatically generatedPre natrénovanie neurónovej siete budeme potrebovať dataset, ktorý si vytvoríme svojpomocne. V zadaní máme jasne zadaný formát, ako musia byť reprezentované vstupné a výstupné dáta siete. Pre tvorbu vstupných údajov sme si vytvorili pomerne jednoduchý program s grafickým rozhraním. Tento pomocný program nám zjednodušil tvorbu vstupných a výstupných znakov teda všetkých prvkov datasetu. Na základe vyfarbenia políčok v poli o veľkosti 4x7 a zadaného čísla, ktoré má hovoriť o tom aké číslo sme vykresľovali v poli sa nám vygenerujú hodnoty 1 a 0 oddelené čiarkou, ktoré môžeme ukladať do súboru CSV a tým vytvoríme dataset.

Obrázok 1 Ukážka nástroja použitého pre tvorbu datasetov

Do trénovacieho datasetu sme zahrnuli aj rôzne predpokladané deformácie. Nie všetky číslice ľudia píšu rovnako a preto sme sa snažili do trénovacieho datasetu dostať čo najviac možností. Takisto sme zahrnuli aj istú mieru posunutia číslic v mriežke. To všetko sme robili s cieľom čo najlepšieho natrénovania siete, pre čo najvyššiu presnosť.

**Tvorba testovacieho datasetu:**

Na vytvorenie testovacieho datasetu sme použili ďalšie ručne vytvorené dáta, do ktorých sme dodatočne pridali aj šum. Testovacie dáta vytvárali dva testovacie subjekty, ktoré boli požiadané aby napísali v našom nástroji každú číslicu. Z týchto údajov sme zostavili dataset pre testovanie. Testovací dataset avšak nepoužívame vo forme, ktorá bola vygenerovaná našim nástrojom, ale dotvárame v ňom zašumenie. Teda políčka už nie sú reprezentované iba jednotkou alebo nulou, ale desatinnými číslami z rozsahu nula až jeden. Na obrázku si to môžeme predstaviť tak, že nula reprezentuje bielu farbu, jednotka čiernu a všetko medzi sú odtiene sivej.

A screenshot of a grid

Description automatically generated

Obrázok 2 Príklad zašumenej číslice 6 z testovacieho datasetu

Proces pridania šumu do testovacieho datasetu prebieha nasledovne. Najskôr sa načíta testovací dataset z CSV súboru pomocou knižnice pandas do objektu DataFrame s názvom dataset. Následne sa definujú parametre pre Gaussovský šum. Priemer šumu noise\_mean je nastavený na hodnotu 0, čo znamená, že šum bude centrovaný okolo nuly, zatiaľ čo smerodajná odchýlka noise\_std\_dev je nastavená na 0.1, čo riadi intenzitu šumu. Vyššia hodnota smerodajnej odchýlky by spôsobila väčšiu variabilitu hodnôt s pridaným šumom oproti pôvodným hodnotám.

Potom sa vytvorí kópia pôvodného datasetu s názvom noisy\_data, aby sa zachovala pôvodná verzia bez zmien. Šum sa pridáva do všetkých stĺpcov datasetu s výnimkou posledných desiatich, ktoré sú považované za cieľové

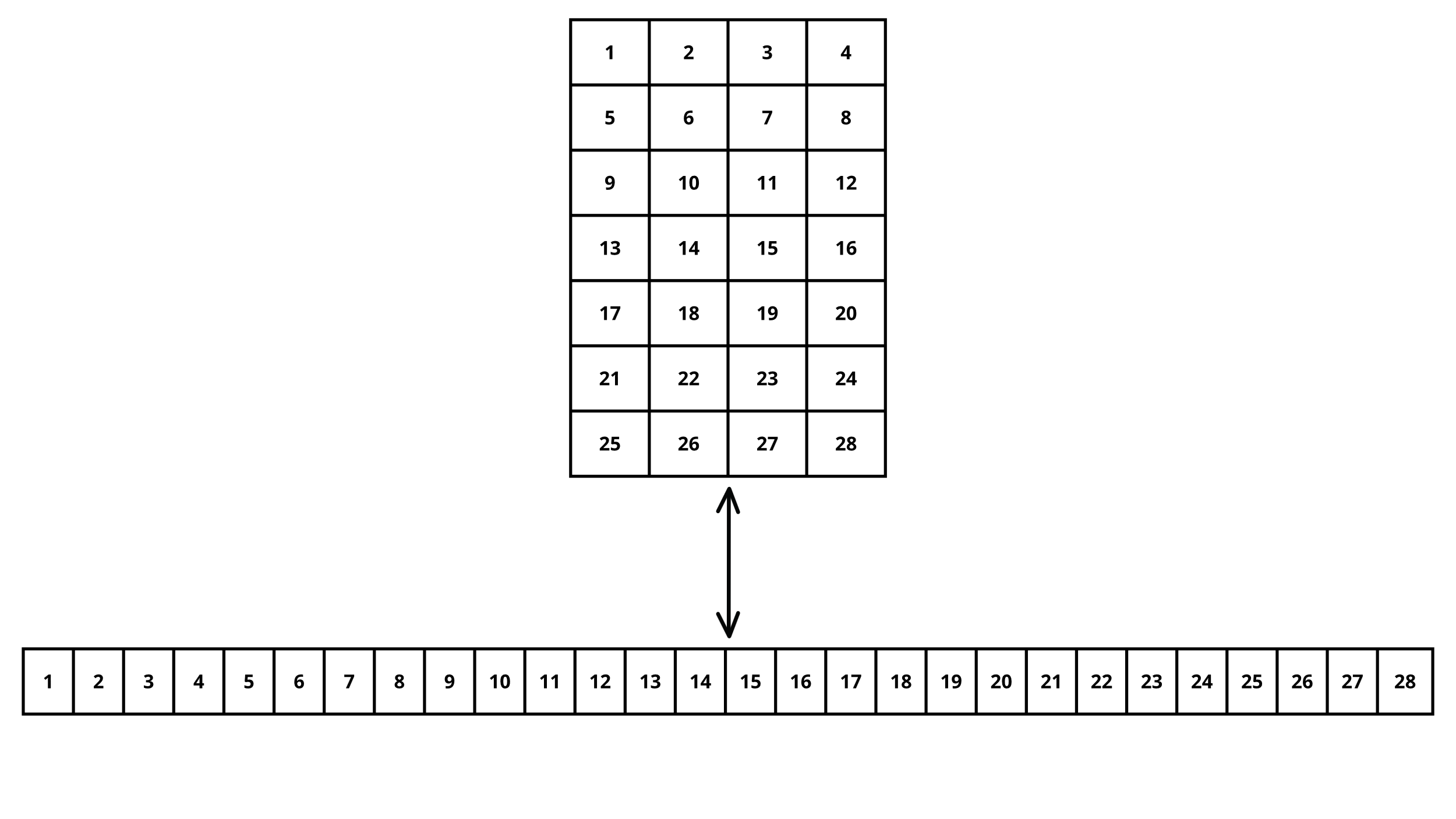
hodnoty a nemajú byť ovplyvnené šumom. Pre každý stĺpec, ktorý podlieha pridaniu šumu, sa generuje náhodný šum pomocou normálneho rozdelenia s priemerom noise\_mean a smerodajnou odchýlkou noise\_std\_dev. Tento generovaný šum sa pripočíta k pôvodným hodnotám v stĺpci a výsledné hodnoty sa obmedzia do rozsahu [0, 1] pomocou metódy clip(0, 1), aby sa zabránilo prekročeniu povolených hraníc.

Po pridaní šumu sa nový dataset uloží do CSV súboru pod názvom "test\_dataset\_with\_noise.csv", pričom sa indexy nezapisujú, aby formát súboru zodpovedal pôvodnému datasetu. Nakoniec sa tento dataset so šumom načíta z uloženého súboru a rozdelí sa na vstupné znaky X\_test a cieľové hodnoty y\_test. Predpokladá sa, že vstupné znaky sú obsiahnuté v prvých 28 stĺpcoch, zatiaľ čo cieľové hodnoty sa nachádzajú v nasledujúcich desiatich stĺpcoch. Tento proces pridania šumu simuluje možné chyby vo vstupných dátach, čo pomáha otestovať odolnosť modelu voči šumu. Šum by mohol v realite vzniknúť napríklad vynechávaním písacieho nástroju alebo roztečením atramentu na papieri.

**Vytvorenie architektúry siete:**

**Vstupné údaje:**

Sieť má na vstupe 28 hodnôt, čo zodpovedá počtu políčok v našej mriežke 4x7. Každé políčko má hodnotu medzi 0 (biela) a 1 (čierna), čím sme získali jednorozmerný vstupný vektor dĺžky 28. Takýto vstup je jednoduchý a sieť ho vie ľahko spracovať.



**Výber vrstiev a počtu neurónov:**

Výber počtu neurónov bol vykonaný experimentálnou metódou, kde sme skúšali rôzne hodnoty.

Prvá vrstva (64 neurónov): Táto vrstva pomáha nájsť rôzne tvary a črty v obrázkoch číslic. Počet 64 neurónov sme vybrali, aby sa model lepšie naučil odlíšiť medzi rôznymi číslicami. Používame aktivačnú funkciu ReLU, ktorá pomáha modelu učiť sa rýchlo a účinne.

Druhá vrstva (32 neurónov): Táto vrstva ďalej spracováva to, čo sa model naučil v prvej vrstve, ale už s menším počtom neurónov. Toto pomáha modelu sústrediť sa len na tie najdôležitejšie črty. Takisto používa aktivačnú funkciu ReLU.

Použitie dvoch vrstiev je dostatočné na to, aby sieť rozoznávala číslice bez zbytočnej zložitosti.

**Výstupná vrstva (10 neurónov):**

Táto vrstva má 10 neurónov, keďže máme 10 možných výstupov, kde každý neurón zodpovedá jednej číslici (0 až 9). Použitá je funkcia softmax, ktorá zaručí, že výstup siete nám povie, ktorá číslica má najvyššiu pravdepodobnosť. Model tak vie určiť, ktorú číslicu považuje za najpravdepodobnejšiu.

**Optimalizácia a stratová funkcia:**

Optimalizátor Adam (Adaptive Moment Estimation) je iteratívny optimalizačný algoritmus používaný na minimalizáciu stratovej funkcie počas trénovania neurónovej siete

Stratová funkcia categorical\_crossentropy je štandardom pre viactriednu klasifikáciu. Pomáha modelu správne rozlišovať medzi jednotlivými výstupnými triedami tým, že maximalizuje pravdepodobnosť správneho výstupu pre daný vstup.

**Trénovacie Parametre: Epochy a Batch size:**

Počet epoch sme experimentálne určili na 70. Je dostatočne veľký na to, aby sa model dobre naučil rozpoznávať vzory. Zároveň nie je príliš vysoký a to preto, aby sme zabránili pretrénovaniu modelu. Ak by bol počet epoch

vyšší, model by sa mohol začať učiť detaily, ktoré sú špecifické len pre trénovací dataset, a tým by mohla klesnúť jeho schopnosť všeobecne rozpoznávať číslice.

Batch size sme experimentálne určili na 8. Nižšia hodnota batch size umožňuje modelu častejšie aktualizovať váhy a tým sa lepšie prispôsobiť dátam. Tento prístup zvyšuje stabilitu učenia a pomáha modelu sa efektívnejšie učiť v prípade obmedzeného množstva údajov.

**Vyhodnotenie výkonu modelu:**

Počas 70 epoch sa model postupne zlepšoval v presnosti na trénovacích aj testovacích dátach. Model dosiahol na konci trénovania nasledujúce výsledky:

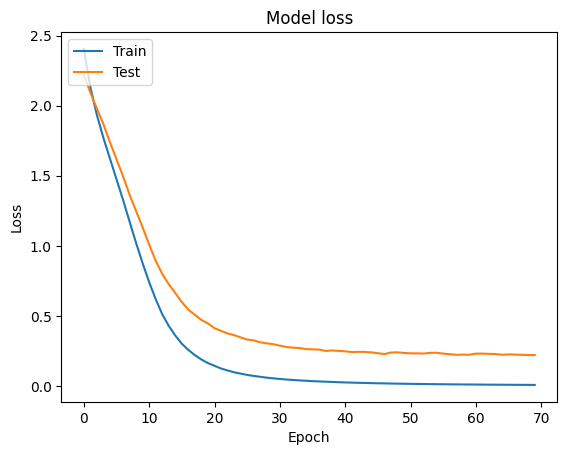
* **Konečná presnosť na trénovacích dátach:** 100%
* **Konečná strata na trénovacích dátach:** 0.0260
* **Konečná presnosť na validačných (testovacích) dátach:** 95%
* **Konečná strata na validačných dátach:** 0.1341

A graph of a graph

Description automatically generatedVývoj presnosti a straty počas trénovania ukazuje, že model stabilne zlepšoval svoje predikčné schopnosti. V počiatočných epochách bola presnosť nízka, čo naznačuje, že model sa ešte len učil základné vzory v dátach. Postupne, okolo 20. epochy, model začal vykazovať stabilnú presnosť na validačných dátach (90–95%), pričom sa minimalizovala aj chyba (strata). Od približne 60. epochy už zlepšovanie modelu nebolo výrazné, čo naznačuje, že model dosiahol svoje maximálne schopnosti na daných dátach.

Obrázok 3 Graf presnosti modelu počas trénovania a testovania

Graf (obr. 3) ukazuje, že model dosiahol veľmi vysokú presnosť na trénovacích dátach (100 %) a tiež vysokú presnosť na testovacích dátach (95 %). Stabilita krivky po približne 20. epoche naznačuje, že model sa naučil rozoznávať číslice efektívne a jeho výkon je konzistentný.



Obrázok 4 Graf priebehu straty modelu počas trénovania a testovania

A diagram of a confused matrix

Description automatically generatedGraf (obr. 4) ukazuje, že model bol schopný efektívne znížiť chybu a naučiť sa rozpoznávať číslice na základe trénovacích dát. Oranžová krivka validačnej straty sa po 20–30 epochách stabilizuje a zostáva na nízkej hodnote, čo potvrdzuje, že model má dobrú generalizáciu na testovacích dátach.

Obrázok 5 Confusion Matrix modelu pre testovacie dáta

Graf (obr. 5) zobrazujeconfusion matrix pre náš model klasifikácie číslic.

* **Hodnoty na hl. diagonále** predstavujú správne klasifikácie – model správne predikoval každú číslicu. V tomto prípade model správne klasifikoval všetky číslice od 0 do 9, okrem niektorých prípadov čísla 8.
* **Chyby** sú zobrazené v políčkach mimo diagonálu. Číslica 8 bola 4-krát zamieňaná za číslicu 2. Táto matrica ukazuje, že model dosahuje vysokú presnosť, ale občas zamieňa číslice 8 a 2. Táto zámena pravdepodobne vzniká v dôsledku spôsobu písania číslice 8 jedným z testovacích subjektov.

**Zhrnutie:**

Model dosiahol veľmi dobrú presnosť na trénovacích aj testovacích dátach, pričom s presnosťou 95% na testovacích dátach potvrdzuje svoju použiteľnosť na rozpoznávanie číslic v mriežke 4x7. Tento výkon naznačuje, že model je dostatočne robustný a schopný rozoznať číselné vzory, ktoré sa odlišujú od tých použitých počas trénovania.

**Riešenie úlohy 1:**

**Zašumenie:**

Pre otestovanie toho, ako veľké zašumenie ešte sieť zvládne, môžeme meniť parameter noise\_std\_dev v našom kóde a sledovať ako to ovplyvní presnosť na testovacích dátach, nakoľko tento šum sa bude aplikovať práve na testovací dataset.

**Predpoklad:**

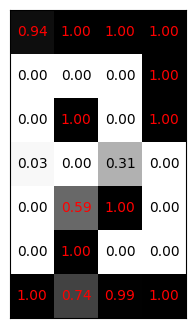
Nízka hodnota noise\_std\_dev (do 0.2): Pridá malý šum, čo znamená, že zmeny v obraze budú minimálne. Model by mal stále správne rozpoznať väčšinu číslic.

Stredná hodnota noise\_std\_dev (od 0.2 po 0.3): Zvyšuje množstvo šumu, čo môže viesť k častejším nesprávnym klasifikáciám, najmä ak sú číslice podobné (napr. 3 a 8, alebo 2 a 8).

Vysoká hodnota noise\_std\_dev (od 0.3 vyššie): Pri veľmi vysokých hodnotách sa obraz stáva menej čitateľným. Očakávajme, že presnosť modelu výrazne klesne, pretože číslice budú príliš zmenené na spoľahlivé rozpoznanie.

**Výsledky experimentu:**

|  |  |
| --- | --- |
| **noise\_std\_dev** | **Presnosť (Accuracy)** |
| 0.05 | 0.9500 |
| 0.1 | 0.9500 |
| 0.15 | 0.9375 |
| 0.2 | 0.9625 |
| 0.25 | 0.9500 |
| 0.3 | 0.9250 |
| 0.35 | 0.8625 |
| 0.4 | 0.8125 |
| 0.45 | 0.7375 |
| 0.5 | 0.7125 |

**A black and white squares with red and black numbers

Description automatically generated**

Obrázok 6 Porovnanie zašumenia pri minimálnom testovanom zašumení (vľavo) a maximálnom testovanom zašumení (vpravo)

**A diagram of a confused matrix

Description automatically generatedA diagram of a confused matrix

Description automatically generated**

Obrázok 7 Porovnanie confusion matíc pri noise\_std\_dev = 0.3 a 0.5

**Zhrnutie**

Model sa správal približne ako sme očakávali, avšak prekvapil nás nárast presnosti pri parametri noise\_std\_dev rovnom 0.2, čo si vysvetľujeme tak, že šum dopomohol v niektorých prípadoch k tomu, aby model vedel lepšie určiť o akú číslicu sa jedná.

**Porušenie:**

Na otestovanie toho, aké veľké porušenie zvládne model, vytvoríme si manuálne porušené dáta. Skúsime posun celej číslice doprava, následne pridávanie alebo uberanie náhodných prvkov číslice. Najskôr uberieme postupne 1, 2, 3 prvky z pôvodnej číslice, následne pridáme 1, 2, 3 prvky do pôvodnej číslice. Spolu teda budeme mať 7 porušených vzoriek pre jednu číslicu. Celý porušený dataset sa nachádza v súbore porusene.csv a obsahuje všetky testované dáta s porušením a na konci je údaj o tom ako boli dáta klasifikované modelom.

**Predpoklad:**

Očakávame, že posunutie do rôznych smerov by model mal zvládnuť, nakoľko určité posuny sme zakomponovali aj do trénovacích dát. Takisto náhodné prvky, resp. inak napísané číslice boli použité počas trénovania. Preto pri miernom porušení by to model mohol zvládnuť.

**Výsledky experimentu:**

Číslice 1 a 7 boli rozpoznané v 7/7 testovaných prípadoch. Uspeli aj keď sme ich posunuli, odobrali až tri prvky, či pridali tri prvky do ich matice.

Číslice 2, 4, 5, 6, 8, 9, 0 neboli rozpoznané len v prípade posunu. V ostatných prípadoch nebol problém a boli správne klasifikované.

Výnimkou je číslica 3, ktorá zlyhala najviac a to pri posune a aj pridaním už len jedného prvku bola klasifikovaná ako číslica 8.

Presnosť na porušených dátach je teda 59/70 čo predstavuje 84.29%.