Umelá inteligencia

FIIT STU

Nazar Biriukov, 122473

Úloha 3 – A

Podúloha: MNIST klasifikátor

Úloha:

Vytvoríte neurónovú sieť na klasifikáciu ručne písaných číslic z dátového súboru MNIST. Súbor údajov pozostáva zo 60 000 obrázkov na trénovanie a 10 000 obrázkov na testovanie, pričom každý obrázok je v odtieňoch sivej, veľkosti 28 x 28 a reprezentuje jednu číslicu od 0 do 9.

Zadanie:

Na riešenie úlohy použite doprednú neurónovú sieť (viacvrstvový perceptrón) a natrénujte ju pomocou algoritmov SGD, SGD s momentom a ADAM. Okrem trénovacej a testovacej chyby odmerajte aj presnosť modelu (t. j. koľkokrát model predikoval správnu triedu na testovacej množine). Model je možné natrénovať s výslednou presnosťou väčšou ako 97%. V úlohe určite použite knižnicu **PyTorch** (stačí verzia pre CPU), iné knižnice podľa svojho uváženia.

Náčrt modelu:

1. Načítajte dataset (je voľne dostupný v knižnici **PyTorch**), rozdeľte ho na trénovaciu a testovaciu množinu.
2. Dáta je vhodné predspracovať pomocou normalizácie (hodnoty pixelov budú v intervale 0 až 1), cieľové hodnoty môžete zakódovať pomocou one-hot kódu. Záleží akú chybovú funkciu si zvolíte.
3. Navrhnite hyperparametre modelu - počet vrstiev a ich veľkosti, zvoľte si vhodné aktivačné funkcie, nastavte rýchlosť učenia a veľkosť dávky (batch) pre optimalizačný algoritmus.
4. Postupne vyskúšajte všetky tri optimalizačné algoritmy spomenuté v zadaní. Trénujte model vždy rovnaký počet epoch, aby ich bolo možné porovnať.

Riešenie

Dataset MNIST pozostáva z ručne písaných číslic (0 – 9). Každý príklad je obrázok v odtieňoch sivej o rozmeroch 28 × 28 pixelov. Cieľom je naučiť neurónovú sieť klasifikovať tieto číslice do 10 tried, pričom každá trieda zodpovedá jednej číslici.

Dáta boli rozdelené nasledovne:

Tréningová množina: 60 000 obrázkov

Testovacia množina: 10 000 obrázkov

Obrázky boli normalizované na strednú hodnotu 0.1307 a štandardnú odchýlku 0.3081.

Popis architektúry modelu

Vstupná vrstva: Transformácia 28×28 obrázku na vektor o veľkosti 784.

1. skrytá vrstva: Lineárna vrstva s 512 neurónmi a aktivačnou funkciou ReLU

2. skrytá vrstva: Lineárna vrstva s 256 neurónmi a aktivačnou funkciou ReLU

Výstupná vrstva: Lineárna vrstva s 10 neurónmi (výstup pre 10 tried)

```python
def build_model():
    return nn.Sequential(
        nn.Flatten(),
        nn.Linear(784, 512),
        nn.ReLU(),
        nn.Linear(512, 256),
        nn.ReLU(),
        nn.Linear(256, 10)
    )
```

V experimente sme vyskúšali tri optimalizačné stratégie:

- SGD
- SGD s momentom (momentum = 0.5,0,7,0,9)
- Adam

```python
print("Learning with SGD:")
model_sgd = build_model()
optimizer_sgd = optim.SGD(model_sgd.parameters(), lr=0.01)
train_losses_sgd, test_losses_sgd, acc_sgd = train_model(model_sgd, optimizer_sgd, criterion, train_loader, test_loader, epochs)
plot_results(train_losses_sgd, test_losses_sgd, acc_sgd, title_prefix="SGD")

print("\nLearning with SGD with momentum:")
model_sgdm = build_model()
optimizer_sgdm = optim.SGD(model_sgdm.parameters(), lr=0.01, momentum=0.5)
train_losses_sgdm, test_losses_sgdm, acc_sgdm = train_model(model_sgdm, optimizer_sgdm, criterion, train_loader, test_loader, epochs)
plot_results(train_losses_sgdm, test_losses_sgdm, acc_sgdm, title_prefix="SGD with Momentum")

print("\nLearning with Adam:")
model_adam = build_model()
optimizer_adam = optim.Adam(model_adam.parameters(), lr=0.001)
train_losses_adam, test_losses_adam, acc_adam = train_model(model_adam, optimizer_adam, criterion, train_loader, test_loader, epochs)
plot_results(train_losses_adam, test_losses_adam, acc_adam, title_prefix="Adam")
```

Spoločné hyperparametre pre všetky experimenty:

Počet epoch: 10

Veľkosť batchu: 64

Loss funkcia: CrossEntropyLoss

V tomto experimente sa na trénovanie modelu použila funkcia CrossEntropyLoss. Táto funkcia sa široko používa v klasifikačných úlohách a meria „vzdialenosť“ medzi predpovedaným rozdelením pravdepodobnosti nad triedami a skutočnou značkou.

Ako funguje:

Model na výstupnej vrstve generuje logity - číselné hodnoty bez normalizácie zodpovedajúce každej triede. Funkcia CrossEntropyLoss interne najprv aplikuje operáciu softmax (prostredníctvom log_softmax), čím prevádza logity na pravdepodobnosti nad všetkými triedami, a potom vypočíta krížovú entropiu medzi týmto rozdelením a skutočnou triedou.

Transformácie na vstup: Normalizácia s parametrami (0.1307, 0.3081)

Optimalizátor	Learning rate(1)	Momentum (1)	Learning rate(2)	Momentum (2)	Learning rate(3)	Momentum (3)
SGD	0.01	x	0.05	x	0.03	x
SGD s momentom	0.01	0.9	0.05	0.07	0.03	0.05
Adam	0.01	x	0.005	x	0.001	x

Prvý experiment Druhý experiment Tretí experiment

Experiment 1

Načítanie dát: Pomocou torchvision.datasets.MNIST sme stiahli a načítali tréningovú a testovaciu množinu, aplikovali sme transformácie.

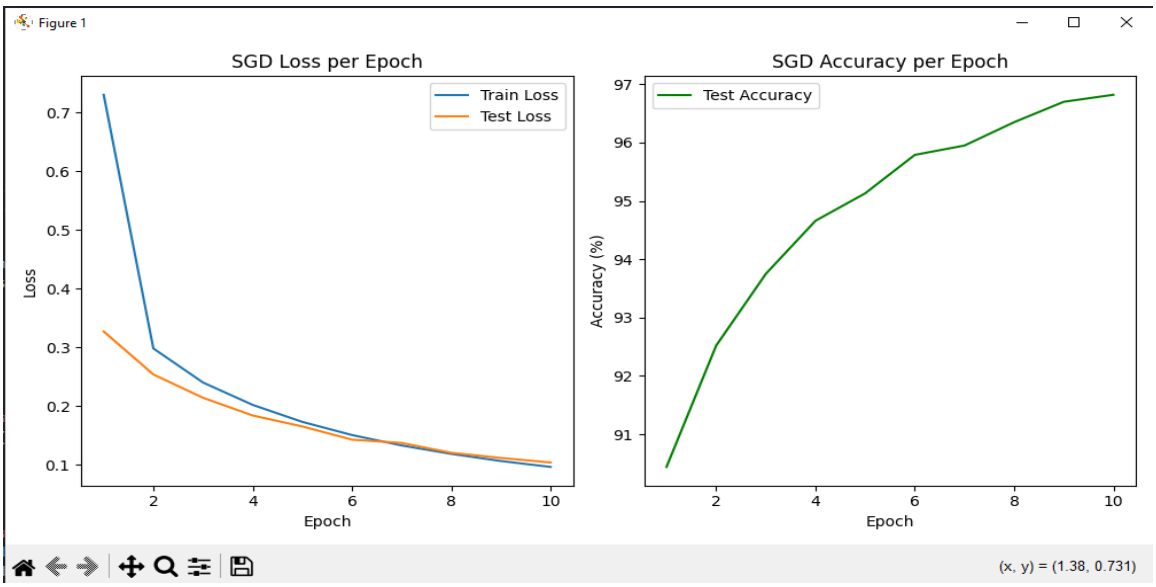
Nastavenie optimizérov:

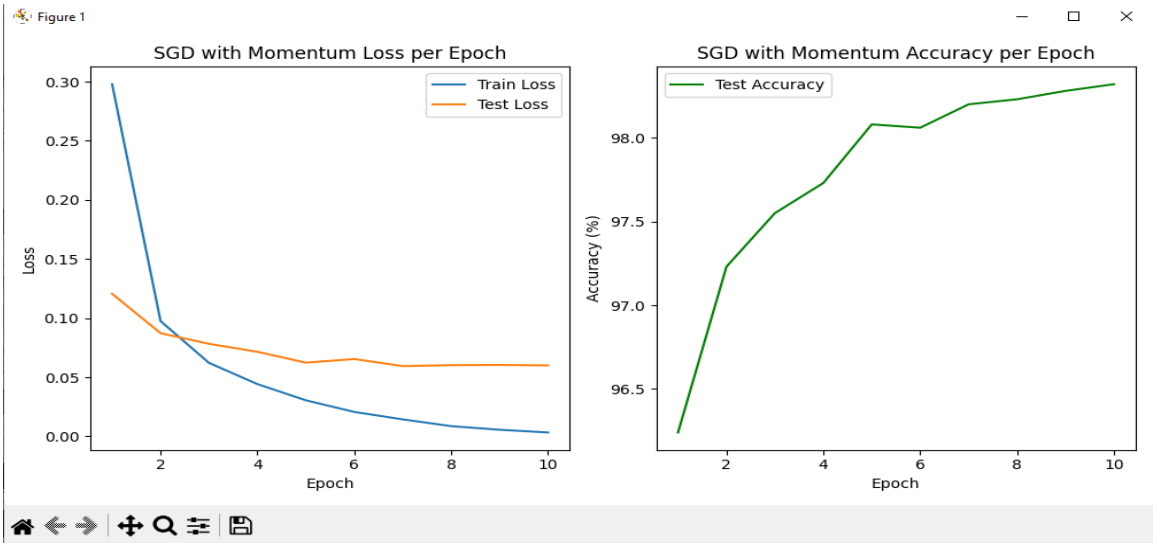
Model 1: SGD s Learning rate = 0.01

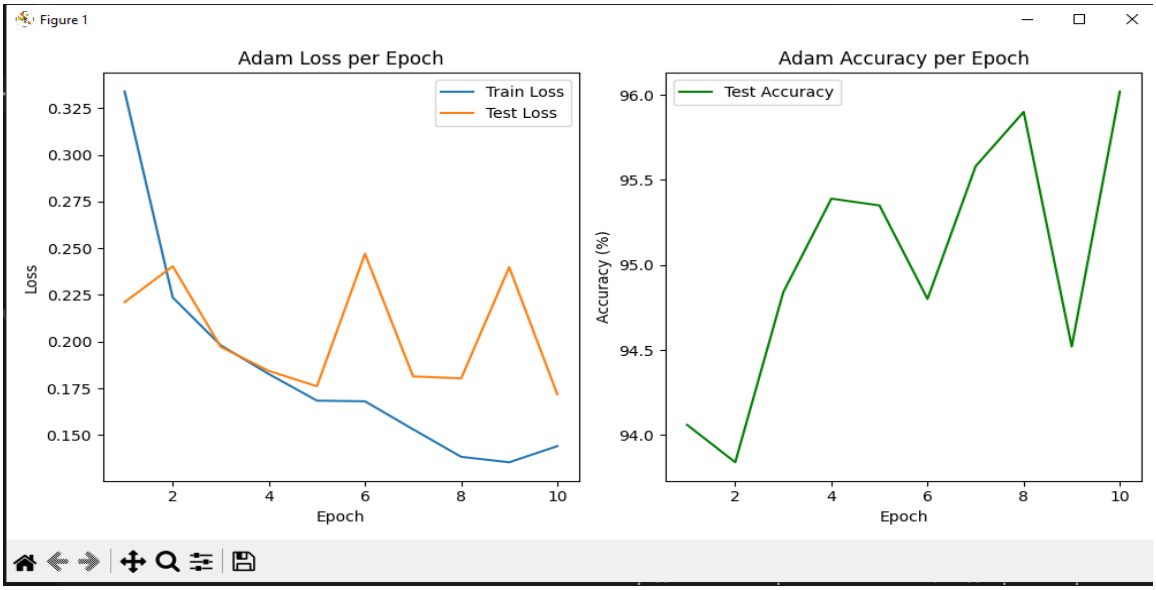
Model 2: SGD s momentum (0.9) a Learning rate = 0.01

Model 3: Adam s Learning rate = 0.01

Tréning: Každý model sme trénovali 10 epoch, zaznamenali sme tréningovú chybu, testovaciu chybu a presnosť na testovacej množine po každej epoche.







Z testu vyplýva, že SGD with momentum dosiahol najvyššiu presnosť na testovacej množine. Výsledná presnosť sa pri SGD with momentum 98%.

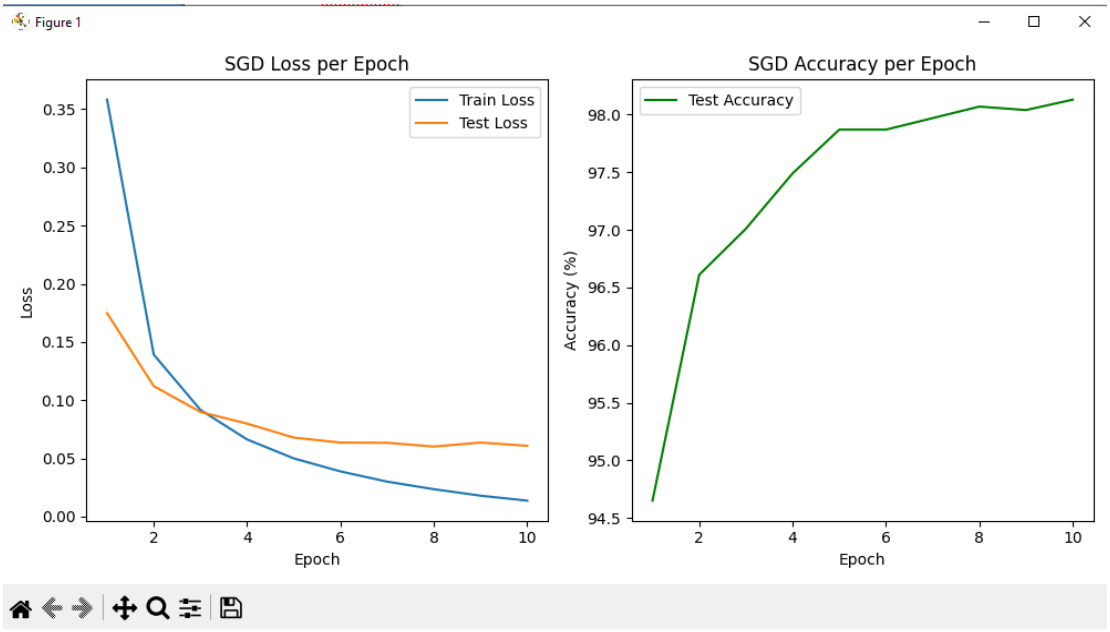
Experiment 2

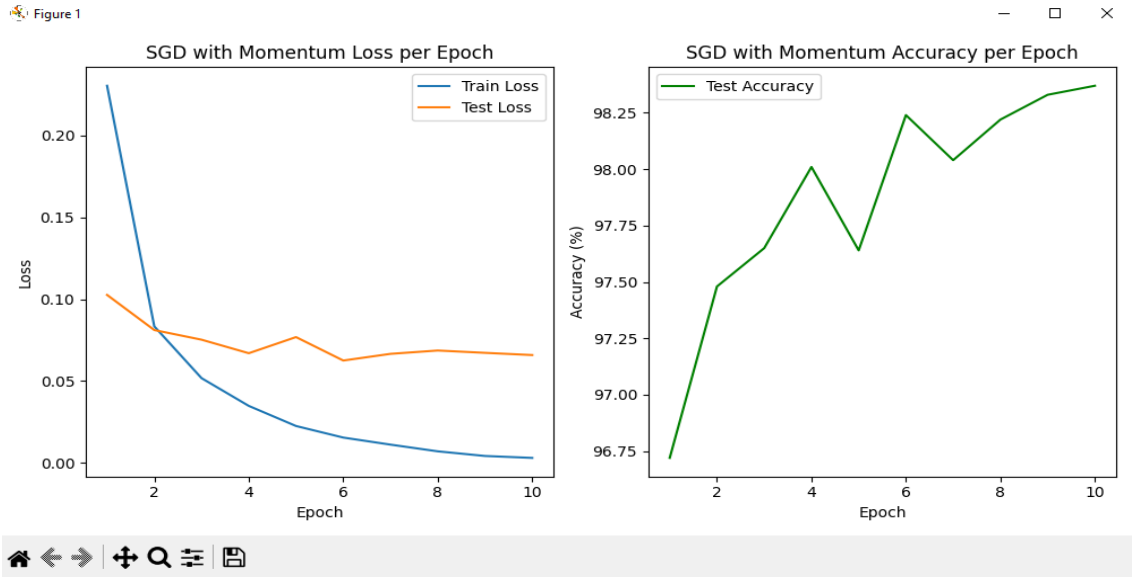
Nastavenie optimizérov:

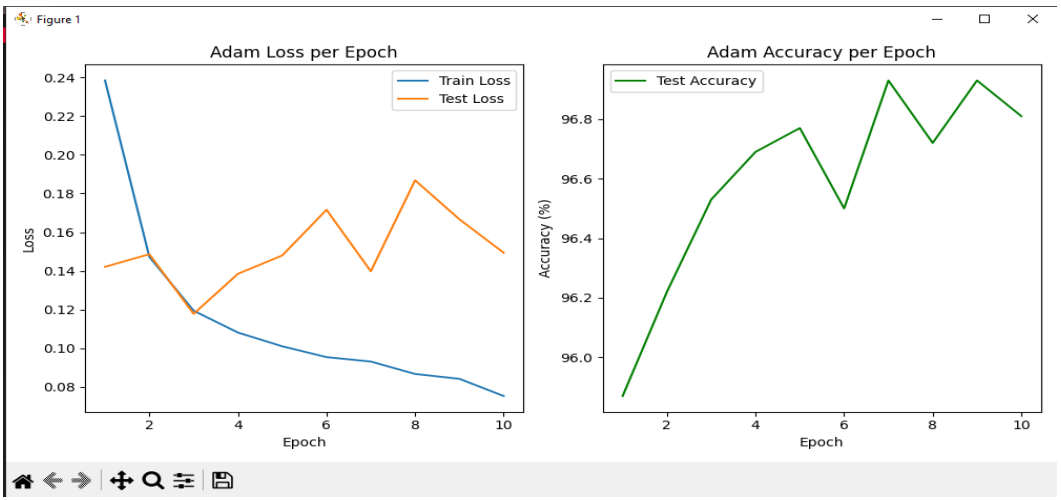
Model 1: SGD s Learning rate = 0.05

Model 2: SGD s momentum (0.7) a Learning rate = 0.05

Model 3: Adam s Learning rate = 0.005







Z testu vyplýva, SGD s momentum že dosiahol najvyššiu presnosť na testovacej množine. Výsledná presnosť sa pri SGD with momentum 98%.

Results:

SGD: 98.13%

SGD with momentum: 98.37%

Adam: 96.81%

Experiment 3

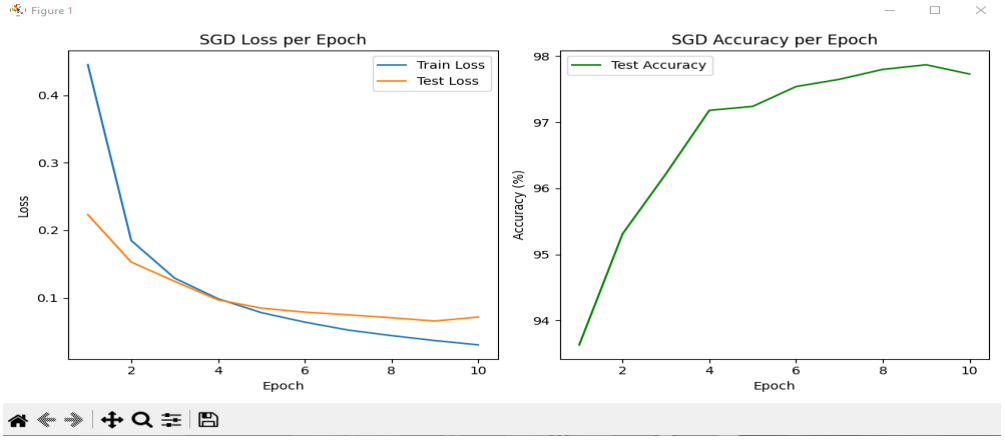
Načítanie dát: Pomocou torchvision.datasets.MNIST sme stiahli a načítali tréningovú a testovaciu množinu, aplikovali sme transformácie.

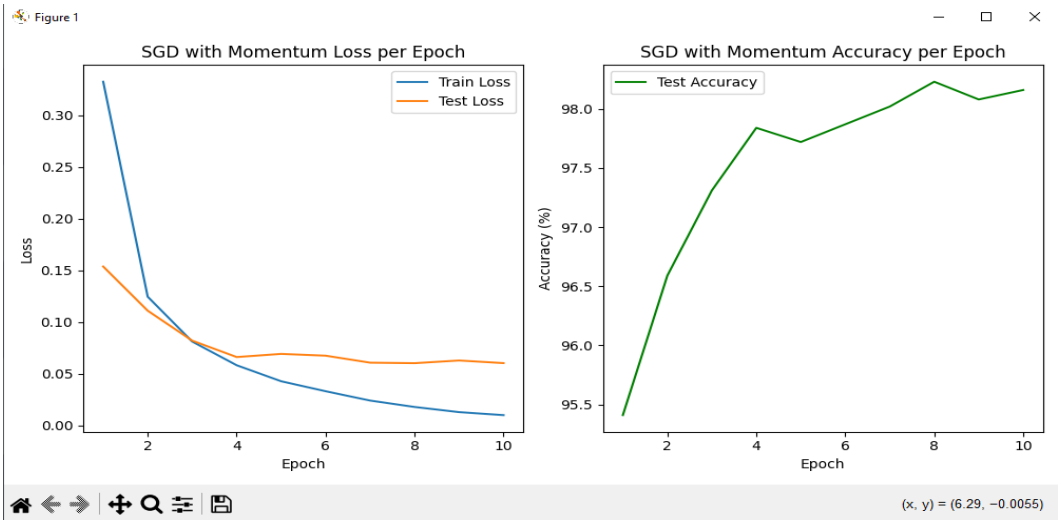
Nastavenie optimizérov:

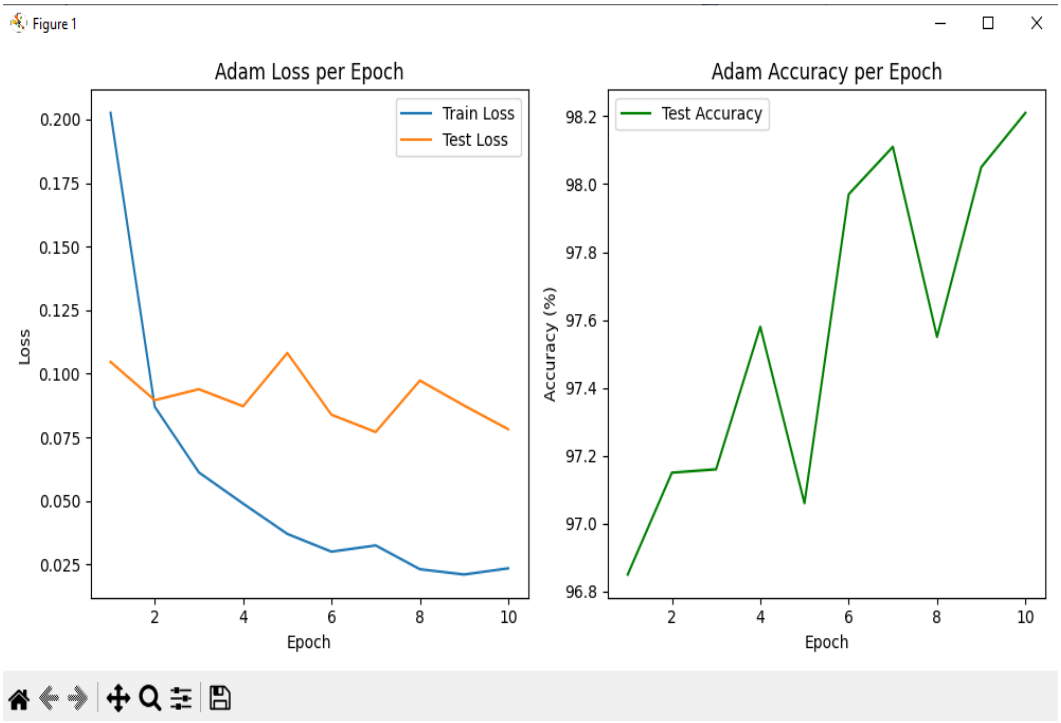
Model 1: SGD s Learning rate = 0.03

Model 2: SGD s momentum (0.5) a Learning rate = 0.03

Model 3: Adam s Learning rate = 0.001







Z testu vyplýva, že Adam dosiahol najvyššiu presnosť na testovacej množine. Výsledná presnosť sa pri SGD with momentum 98.21%

Záver

Architektúra s dvoma skrytými lineárnymi vrstvami a aktivačnou funkciou ReLU dosiahla na MNIST vysokú presnosť.

Z troch testovaných optimalizačných metód (SGD, SGD s momentom a Adam) dosiahol najlepšie a najrýchlejšie konvergujúce výsledky optimalizátor Adam.

Výsledná presnosť sa pri SGD with momentum 98%.

Celkovo experiment potvrdil, že výber optimalizačného algoritmu má značný vplyv na rýchlosť a kvalitu konvergencie trénovaného modelu. Adam sa ukázal ako najlepšia voľba pre danú úlohu a architektúru.

Podúloha: Backpropagation algoritmus

Úloha:

V tejto úlohe budete implementovať plne funkčný algoritmus backpropagation, ktorý je kľúčovým komponentom tréningu neurónovej siete. Umožňuje jej učiť sa pomocou minimalizácie zadanej chybovej funkcie. Treba implementovať doprednú aj spätnú časť pre jednotlivé operátory a funkcie, ako aj aktualizácie parametrov siete. Algoritmus overíte natrénovaním jednoduchej doprednej neurónovej siete (viacvrstvového perceptrónu).

Zadanie:

Na riešenie úlohy použite knižnicu **NumPy** pre maticové a vektorové operácie. Použitie knižníc ako *PyTorch* a *TensorFlow*, ktoré obsahujú autograd je *zakázané*. Implementujte modulárnu architektúru, v ktorej bude možné jednotlivé moduly reťaziť. Implementácia bude obsahovať lineárnu vrstvu, aktivačnú funkciu sigmoid, tanh, relu a chybovú funkciu MSE (mean squared error).

Náčrt modelu:

1. Pre validáciu algoritmu použite XOR problém, použite dvojvrstvovú sieť, ktorá bude mať na skrytej vrstve 4 neuróny, na výstupnej vrstve 1 neurón a trénovať sa bude pomocou MSE chybovej funkcie. Rýchlosť učenia môže byť 0.1 až 0.01 a počet epoch by mal stačiť okolo 500.
2. Najskôr implementujte dopredný smer pre jednotlivé moduly a spojte ich do jedného modelu (zoznam modulov), ktorý vám zo vstupných údajov vráti výstup.
3. Začnite od MSE funkcie a pre každý modul spočítajte jeho deriváciu a implementujte spätný smer. Dajte si pozor, lebo pri počítaní derivácie je

potrebné udržiavať aj niektoré výsledky z dopredného smeru. Moduly môžu mať svoj stav, alebo môžu byť aj bezstavové a výsledky môžu byť odložené niekde inde. To ponechávame na vašej voľbe.

4. Implementujte pravidlo pre aktualizáciu váh bez momentu a potom aj s momentom.
5. Volanie modelu by malo byť jednoduché, pre dopredný smer model.forward(vstup) a pre spätný smer model.backward(chyba) a následne volanie aktualizácie parametrov (či už to bude implementovať model, alebo iná samostatná trieda je opäť na vás).

Riešenie

Tento program demonštruje implementáciu algoritmu backpropagation na neurónovej sieti, ktorá rieši logické úlohy AND, OR a XOR. ,výpočet chyby MSE , spätné šírenie, ako aj aktualizáciu parametrov (váh a posunov) pomocou gradientného zostupu s možnosťou využitia hybnosti.

Hlavnými prvkami implementácie sú:

Priechod vpred (forward pass):

Vstupné údaje sa privádzajú na vstup siete a potom postupne prechádzajú lineárnymi vrstvami a nelineárnymi aktiváciami (napr. tanh pre skrytú vrstvu a sigmoid pre výstupnú vrstvu). Na výstupe sieť generuje predpovede.

Stratová funkcia (MSE):

Ako chybová funkcia sa používa stredná kvadratická chyba (MSE):

MSE (Mean Squared Error - stredná kvadratická chyba) je stratová funkcia, ktorá sa bežne používa na riešenie regresných problémov a v tomto prípade na hodnotenie presnosti predpovedí neurónovej siete. Meria stredný štvorcový rozdiel medzi predpovedanými a skutočnými hodnotami. Úloha trénovania sa obmedzuje na výber takých parametrov modelu, pri ktorých bude MSE čo najbližšie k nule.

Spätné šírenie:

Na základe parciálnych derivácií chyby výstupov a parametrov (váh a posunov) sa vypočítajú gradienty. Používajú sa derivácie aktivačných funkcií (napr. tanh a sigmoid) a lineárne transformácie. Takto získame gradienty, ktoré naznačujú, ako by sa mali zmeniť parametre, aby sa znížila chyba.

Aktualizácia parametrov:

Parametre sa aktualizujú pomocou gradientného zostupu

Gradientný zostup s hybnosťou je tiež možný

Experimenty s rôznymi úlohami a architektúrami

Grafy počas trénovania

Počas trénovania sa chyba (MSE) podľa epoch ukladá do kódu v zozname strát. Po skončení trénovania môžete vykresliť graf zmeny chyby na trénovanej vzorke. Tieto grafy by mali byť pripojené k dokumentácii a mali by zobrazovať, ako sa chyba počas trénovania znižuje pre rôzne konfigurácie (napr. bez a s krútiacim momentom, s rôznymi rýchlosťami trénovania).

Rôzne konfigurácie trénovania

S momentom vs. bez momentu:

Porovnajte tréning s momentom=0,0 a napr. s momentom=0,9. Očakáva sa, že konvergencia bude stabilnejšia a rýchlejšia s momentom.

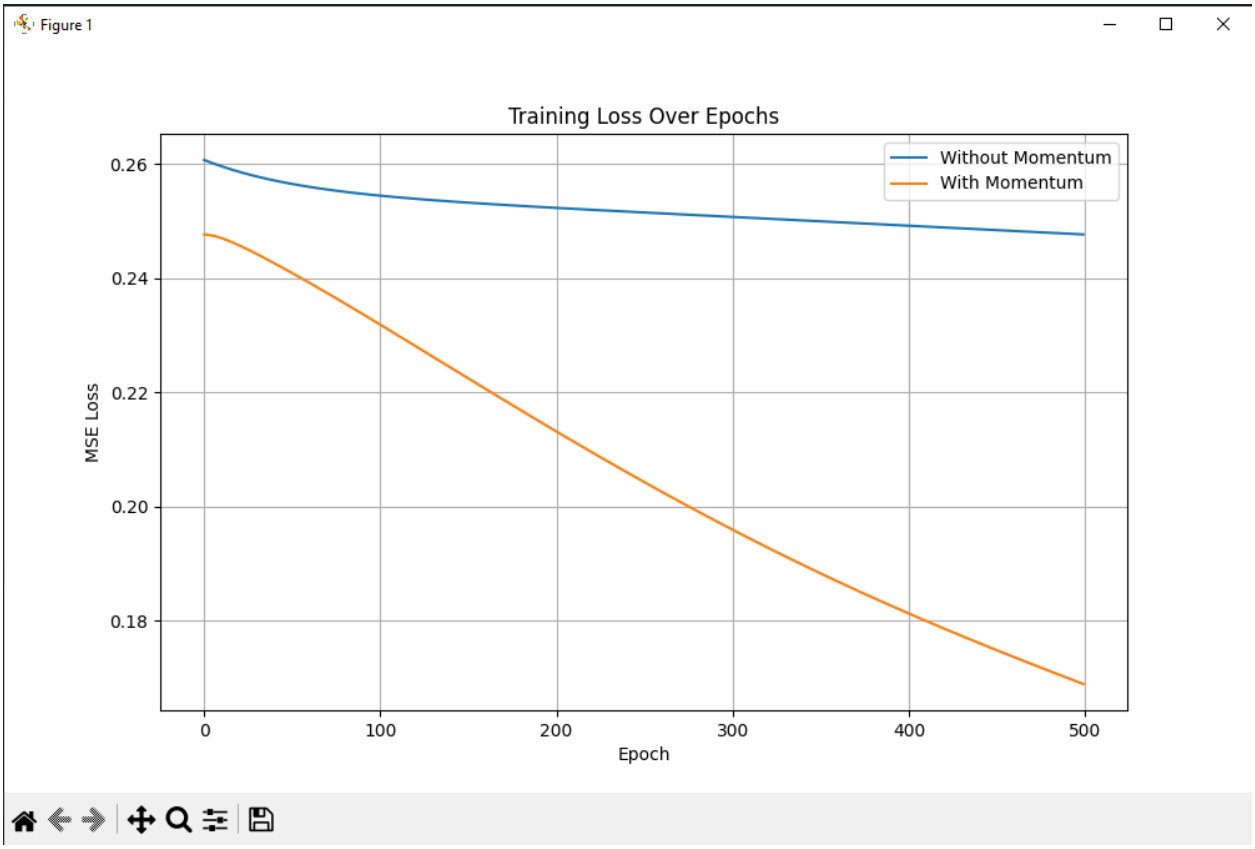
Rôzne miery učenia pre XOR:

Pre problém XOR vyskúšajte aspoň dve rôzne hodnoty learning_rate (napr. 0,01 a 0,05 alebo 0,05 a 0,1) a vykreslite konvergenciu. Zvýšenie learning_rate môže urýchliť učenie, ale príliš vysoká hodnota môže spôsobiť nestabilitu procesu.

Experimentujte s rôznymi úlohami a architektúrami:

	Learning rate(1)	Momentum(1)	Learning rate(2)	Momentum(2)	Learning rate(3)	Momentum(3)
XOR	0.01	0.9	0.05	0.07	0.07	0.05
OR	0.01	0.9	0.05	0.07	0.07	0.05
AND	0.01	0.9	0.05	0.07	0.07	0.05

Experiment 1 with XOR (0.01 , 0.9)



Training without momentum

Input: [0 0], Predicted: 0.4245, Pred Label: 0, Actual: 0

Input: [0 1], Predicted: 0.5827, Pred Label: 1, Actual: 1

Input: [1 0], Predicted: 0.4418, Pred Label: 0, Actual: 1

Input: [1 1], Predicted: 0.5698, Pred Label: 1, Actual: 0

Accuracy: 50.00%

Training with momentum

Input: [0 0], Predicted: 0.2585, Pred Label: 0, Actual: 0

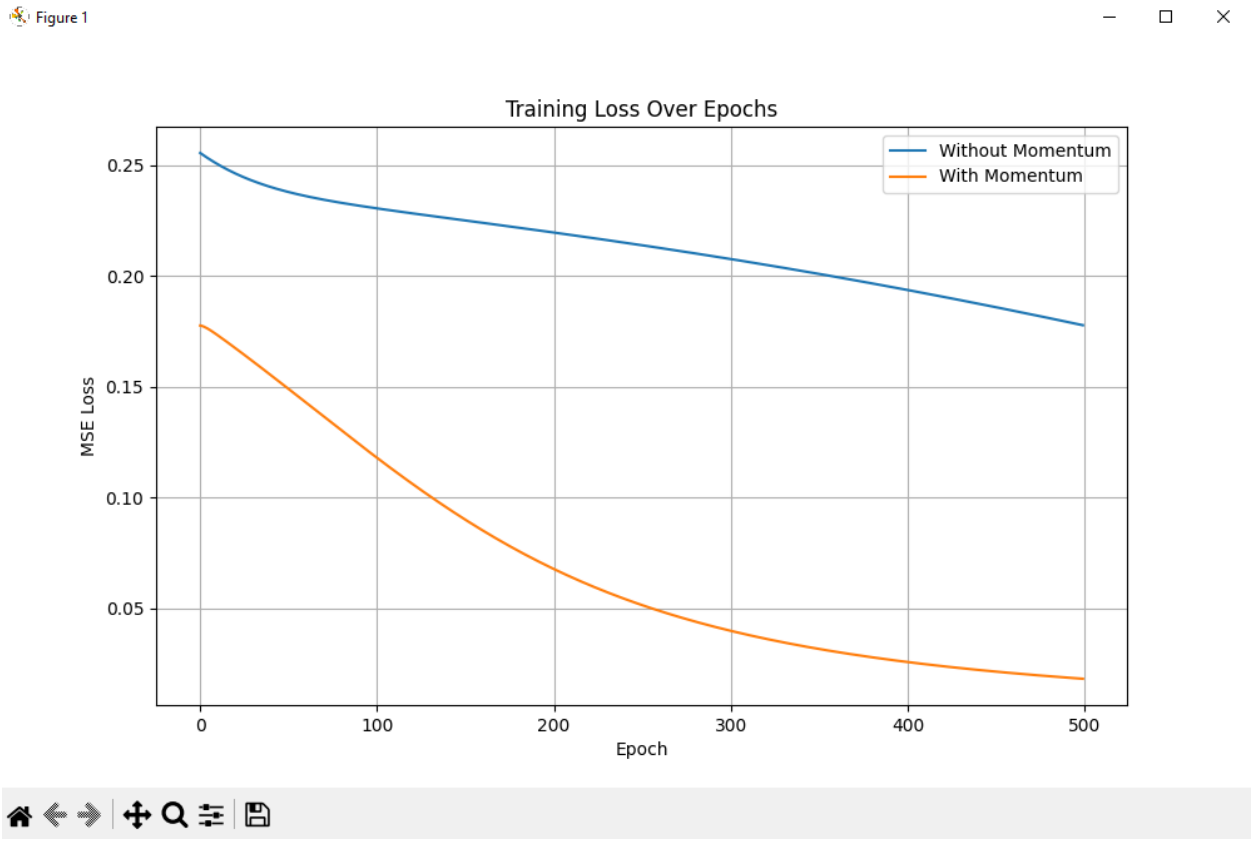
Input: [0 1], Predicted: 0.7123, Pred Label: 1, Actual: 1

Input: [1 0], Predicted: 0.5309, Pred Label: 1, Actual: 1

Input: [1 1], Predicted: 0.5526, Pred Label: 1, Actual: 0

Accuracy: 75.00%

Experiment 2 with XOR (0.05 , 0.7)



Training without momentum

Input: [0 0], Predicted: 0.3611, Pred Label: 0, Actual: 0

Input: [0 1], Predicted: 0.5341, Pred Label: 1, Actual: 1

Input: [1 0], Predicted: 0.6028, Pred Label: 1, Actual: 1

Input: [1 1], Predicted: 0.4526, Pred Label: 0, Actual: 0

Accuracy: 100.00%

Training with momentum

Input: [0 0], Predicted: 0.0873, Pred Label: 0, Actual: 0

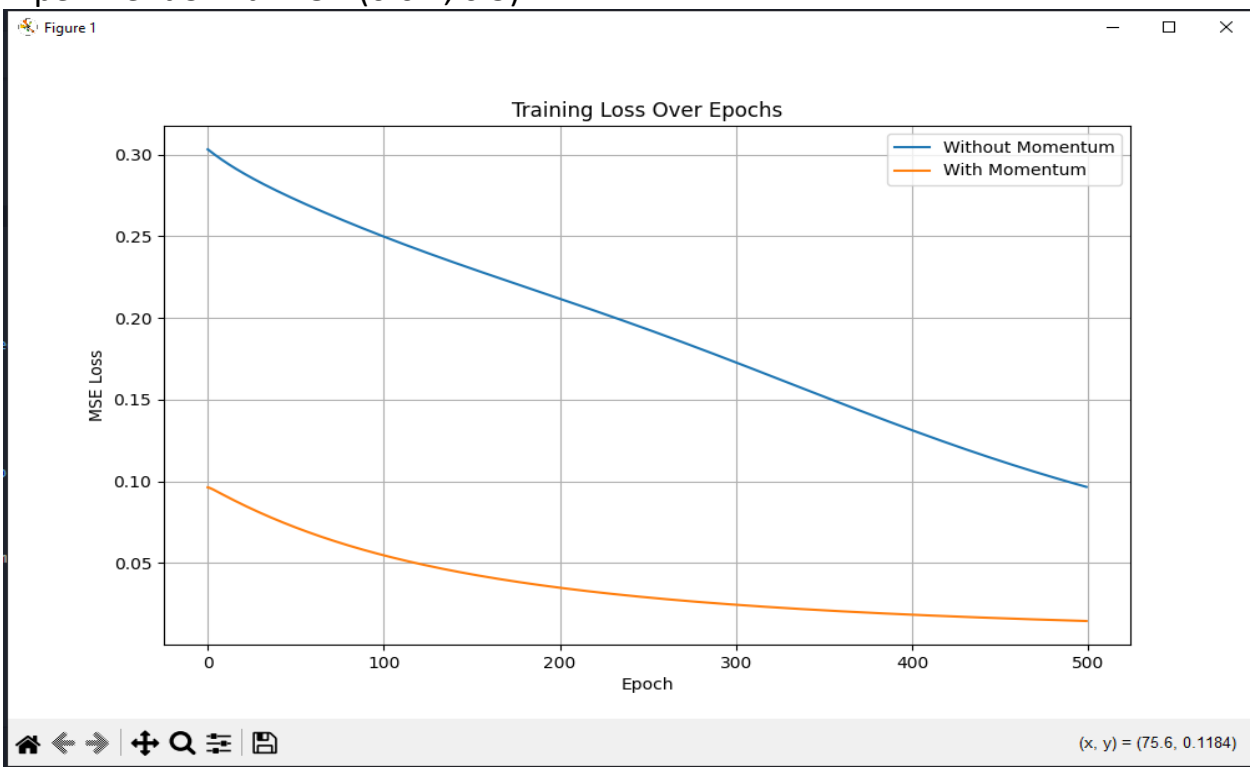
Input: [0 1], Predicted: 0.8571, Pred Label: 1, Actual: 1

Input: [1 0], Predicted: 0.8575, Pred Label: 1, Actual: 1

Input: [1 1], Predicted: 0.1562, Pred Label: 0, Actual: 0

Accuracy: 100.00%

Experiment 3 with XOR (0.07 , 0.5)



Training without momentum

Input: [0 0], Predicted: 0.2878, Pred Label: 0, Actual: 0

Input: [0 1], Predicted: 0.6793, Pred Label: 1, Actual: 1

Input: [1 0], Predicted: 0.6918, Pred Label: 1, Actual: 1

Input: [1 1], Predicted: 0.3238, Pred Label: 0, Actual: 0

Accuracy: 100.00%

Training with momentum

Input: [0 0], Predicted: 0.0741, Pred Label: 0, Actual: 0

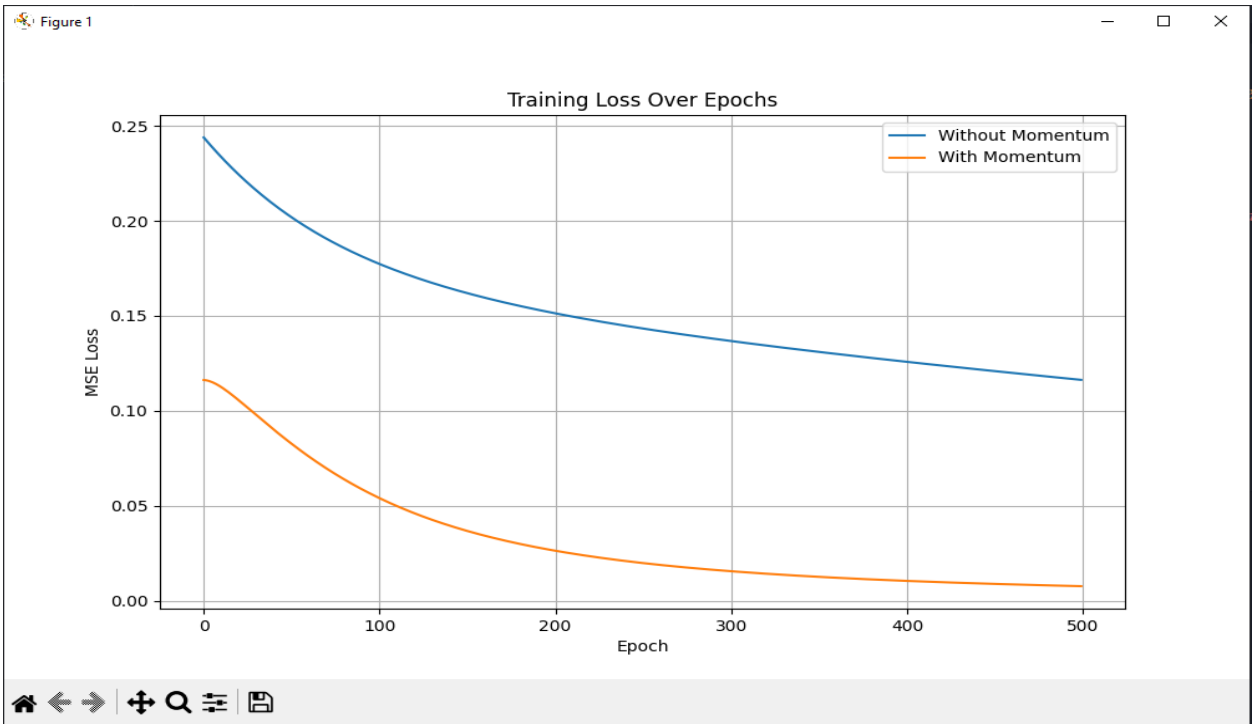
Input: [0 1], Predicted: 0.8692, Pred Label: 1, Actual: 1

Input: [1 0], Predicted: 0.8819, Pred Label: 1, Actual: 1

Input: [1 1], Predicted: 0.1471, Pred Label: 0, Actual: 0

Accuracy: 100.00%

Experiment 1 with OR (0.01 , 0.9)



Training without momentum

Input: [0 0], Predicted: 0.5495, Pred Label: 1, Actual: 0

Input: [0 1], Predicted: 0.6893, Pred Label: 1, Actual: 1

Input: [1 0], Predicted: 0.8016, Pred Label: 1, Actual: 1

Input: [1 1], Predicted: 0.8354, Pred Label: 1, Actual: 1

Accuracy: 75.00%

Training with momentum

Input: [0 0], Predicted: 0.1278, Pred Label: 0, Actual: 0

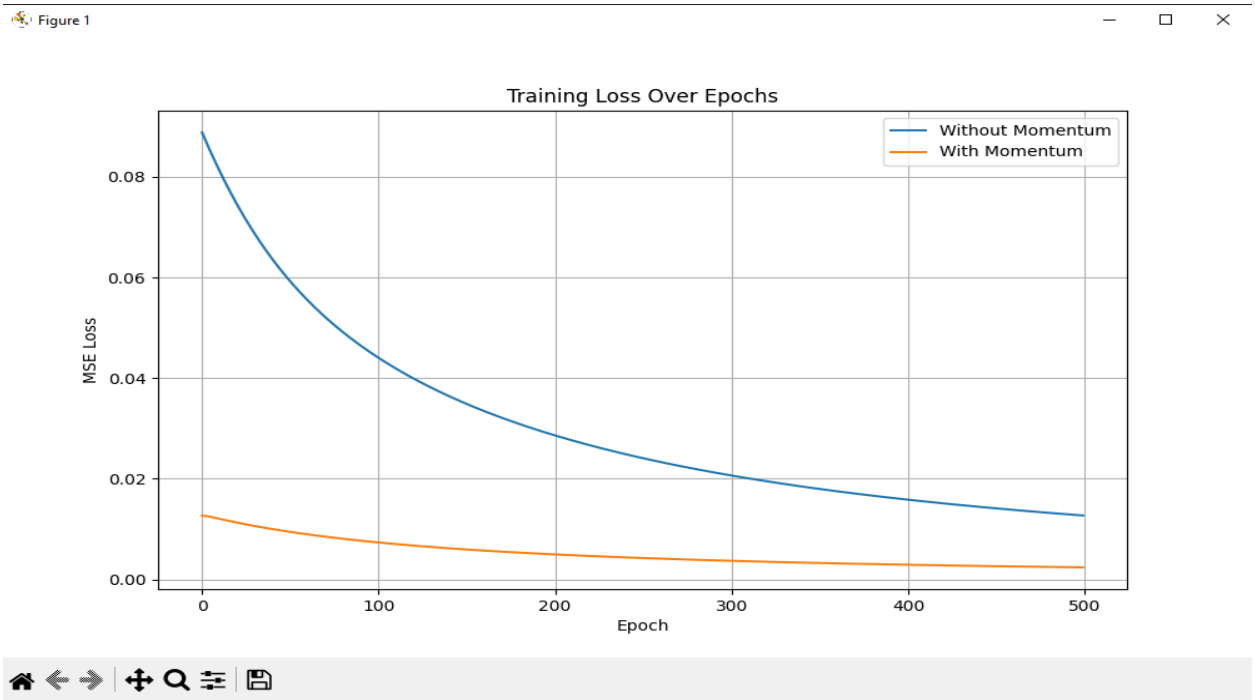
Input: [0 1], Predicted: 0.9170, Pred Label: 1, Actual: 1

Input: [1 0], Predicted: 0.9155, Pred Label: 1, Actual: 1

Input: [1 1], Predicted: 0.9815, Pred Label: 1, Actual: 1

Accuracy: 100.00%

Experiment 2 with OR (0.05 , 0.7)



Training without momentum

Input: [0 0], Predicted: 0.1572, Pred Label: 0, Actual: 0

Input: [0 1], Predicted: 0.9136, Pred Label: 1, Actual: 1

Input: [1 0], Predicted: 0.8679, Pred Label: 1, Actual: 1

Input: [1 1], Predicted: 0.9663, Pred Label: 1, Actual: 1

Accuracy: 100.00%

Training with momentum

Input: [0 0], Predicted: 0.0681, Pred Label: 0, Actual: 0

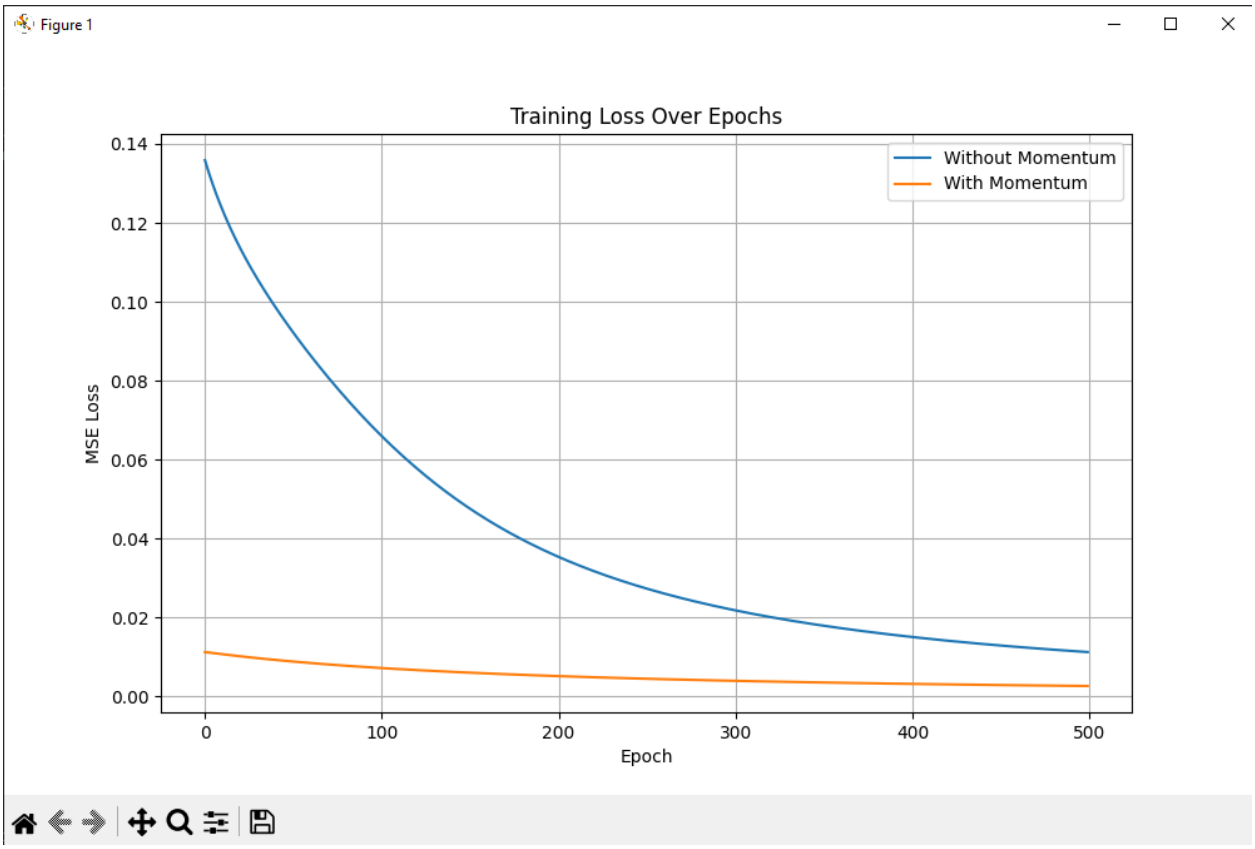
Input: [0 1], Predicted: 0.9629, Pred Label: 1, Actual: 1

Input: [1 0], Predicted: 0.9419, Pred Label: 1, Actual: 1

Input: [1 1], Predicted: 0.9883, Pred Label: 1, Actual: 1

Accuracy: 100.00%

Experiment 3 with OR (0.07 , 0.5)



Training without momentum

Input: [0 0], Predicted: 0.1542, Pred Label: 0, Actual: 0

Input: [0 1], Predicted: 0.9131, Pred Label: 1, Actual: 1

Input: [1 0], Predicted: 0.8884, Pred Label: 1, Actual: 1

Input: [1 1], Predicted: 0.9652, Pred Label: 1, Actual: 1

Accuracy: 100.00%

Training with momentum

Input: [0 0], Predicted: 0.0742, Pred Label: 0, Actual: 0

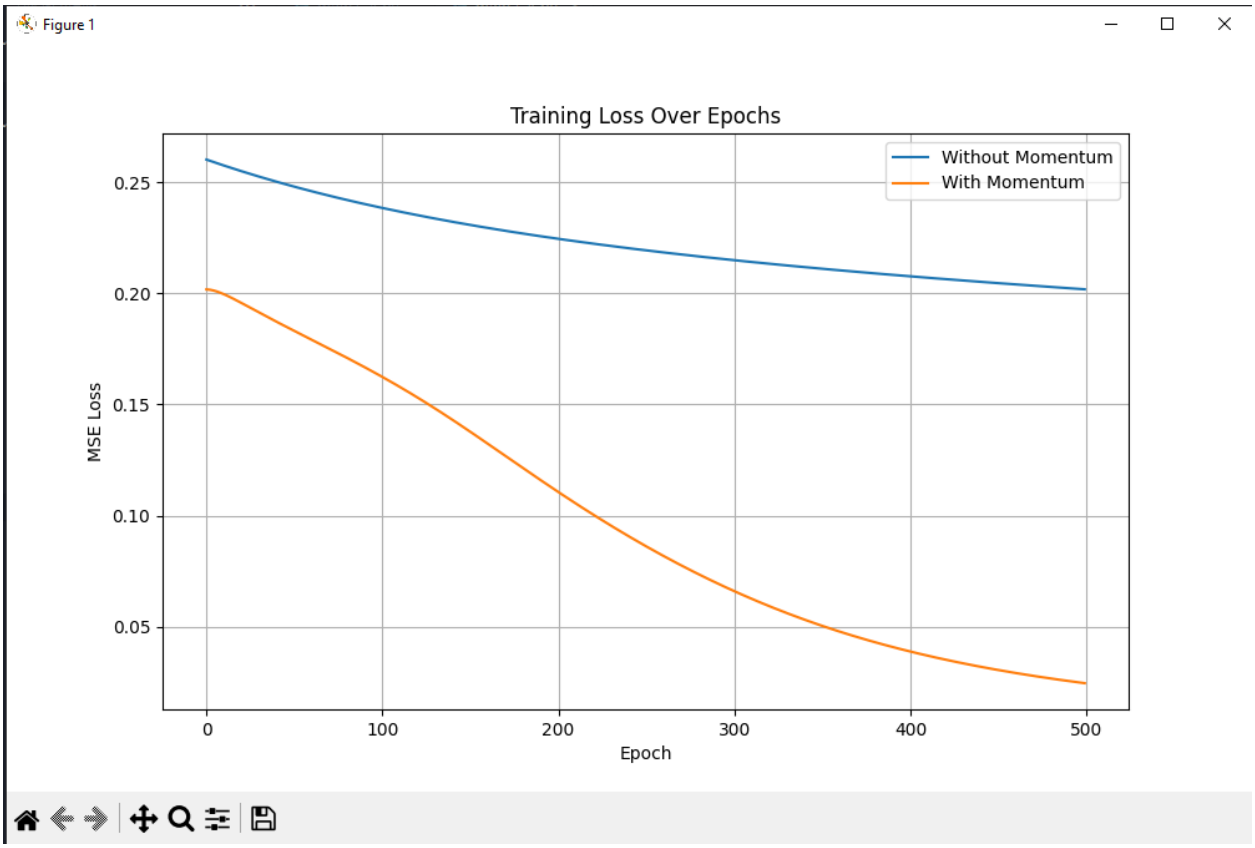
Input: [0 1], Predicted: 0.9557, Pred Label: 1, Actual: 1

Input: [1 0], Predicted: 0.9447, Pred Label: 1, Actual: 1

Input: [1 1], Predicted: 0.9868, Pred Label: 1, Actual: 1

Accuracy: 100.00%

Experiment 1 with AND (0.01 , 0.9)



Training without momentum

Input: [0 0], Predicted: 0.3359, Pred Label: 0, Actual: 0

Input: [0 1], Predicted: 0.3938, Pred Label: 0, Actual: 0

Input: [1 0], Predicted: 0.2944, Pred Label: 0, Actual: 0

Input: [1 1], Predicted: 0.3273, Pred Label: 0, Actual: 1

Accuracy: 75.00%

Training with momentum

Input: [0 0], Predicted: 0.0164, Pred Label: 0, Actual: 0

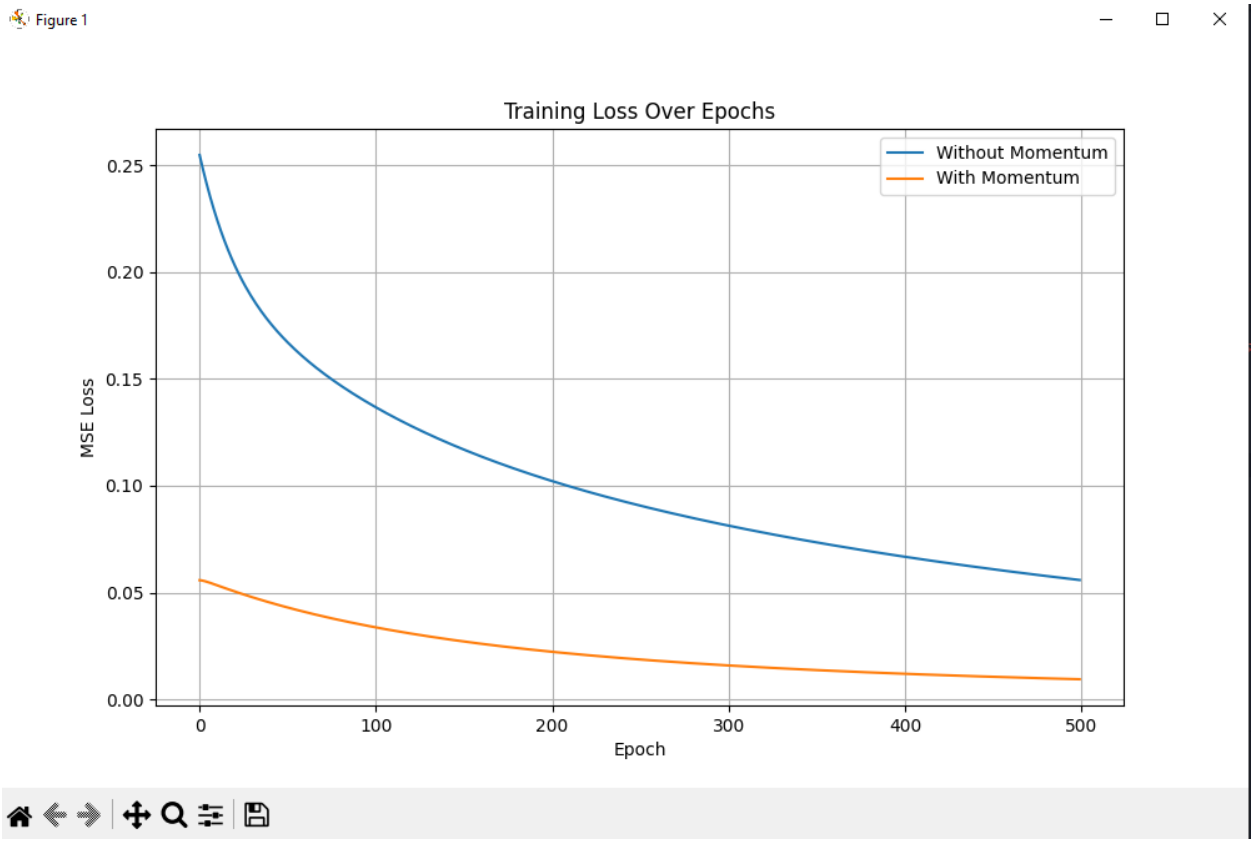
Input: [0 1], Predicted: 0.1747, Pred Label: 0, Actual: 0

Input: [1 0], Predicted: 0.1358, Pred Label: 0, Actual: 0

Input: [1 1], Predicted: 0.7785, Pred Label: 1, Actual: 1

Accuracy: 100.00%

Experiment 2 with AND (0.05 , 0.7)



Training without momentum

Input: [0 0], Predicted: 0.0484, Pred Label: 0, Actual: 0

Input: [0 1], Predicted: 0.2316, Pred Label: 0, Actual: 0

Input: [1 0], Predicted: 0.2710, Pred Label: 0, Actual: 0

Input: [1 1], Predicted: 0.6940, Pred Label: 1, Actual: 1

Accuracy: 100.00%

Training with momentum

Input: [0 0], Predicted: 0.0025, Pred Label: 0, Actual: 0

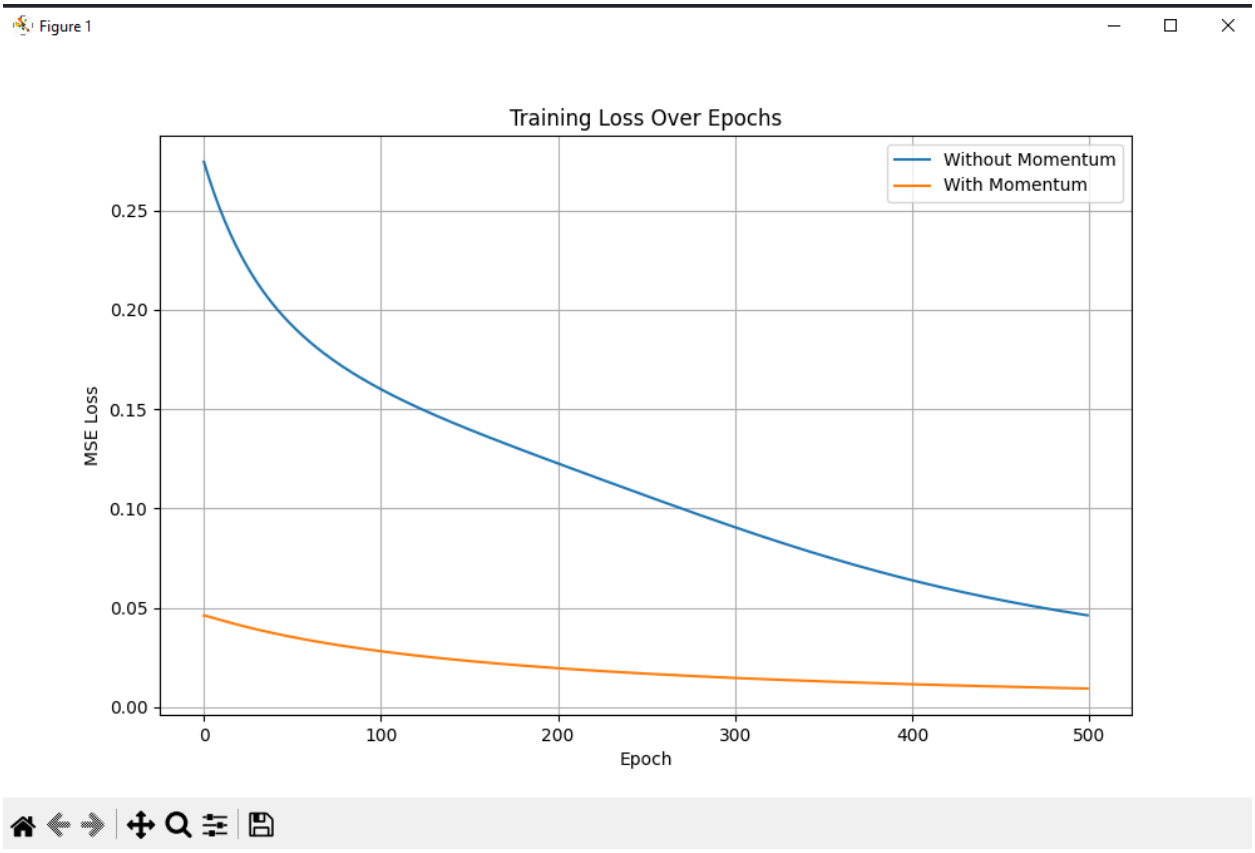
Input: [0 1], Predicted: 0.1059, Pred Label: 0, Actual: 0

Input: [1 0], Predicted: 0.1034, Pred Label: 0, Actual: 0

Input: [1 1], Predicted: 0.8755, Pred Label: 1, Actual: 1

Accuracy: 100.00%

Experiment 3 with AND (0.07 , 0.5)



Training without momentum

Input: [0 0], Predicted: 0.0701, Pred Label: 0, Actual: 0

Input: [0 1], Predicted: 0.2683, Pred Label: 0, Actual: 0

Input: [1 0], Predicted: 0.1207, Pred Label: 0, Actual: 0

Input: [1 1], Predicted: 0.6948, Pred Label: 1, Actual: 1

Accuracy: 100.00%

Training with momentum

Input: [0 0], Predicted: 0.0100, Pred Label: 0, Actual: 0

Input: [0 1], Predicted: 0.1129, Pred Label: 0, Actual: 0

Input: [1 0], Predicted: 0.0850, Pred Label: 0, Actual: 0

Input: [1 1], Predicted: 0.8675, Pred Label: 1, Actual: 1

Accuracy: 100.00%

Zhrnutie

Implementácia spätného šírenia.

Testy siete s 1 skrytou vrstvou na úlohách AND, OR a XOR.

Príkladové grafy zobrazujúce dynamiku učenia.

Porovnanie rôznych hyperparametrov: rôznych rýchlostí učenia pre úlohy XOR, OR, AND.

Tieto experimenty umožňujú lepšie pochopiť vplyv architektúry siete a výberu hyperparametrov na schopnosť modelu úspešne sa učiť jednoduché logické úlohy.