

## MNIST klasifikátor

### Zadanie 3.a

## Contents

MNIST klasifikátor Zadanie 3.a .....	1
Riešený problém .....	3
Opis riešenia .....	3
Architektúra modelu .....	3
Experimentovanie.....	3
Zhodnotenie.....	11
Podúloha: Backpropagation algoritmus .....	12
Riešený problém .....	13
Riešenie .....	13
Experimentovanie.....	14
Problem XOR .....	14
Problém OR .....	15
Problém AND .....	17
Zhodnotenie.....	18

## Riešený problém

Cieľom experimentu je navrhnuť a implementovať viacvrstvovú neurónovú sieť (MLP) na klasifikáciu datasetu MNIST, ktorý obsahuje obrázky číslíc (0-9). Model má správne identifikovať číslicu na obrázku.

## Opis riešenia

Ako základnu triedu neuronoveho modelu som použil `torch.nn.Module`. Na jeho inicializáciu sme použili vrstvy opísané v `architecture` modelu.

Použili sme architektúru MLP s tromi plne prepojenými (fully connected) vrstvami a funkciami aktivácie ReLU. Model bol trénovaný na 30 epoch pomocou troch optimalizačných algoritmov:

SGD (Stochastic Gradient Descent)

SGD s momentom (momentum = 0.9)

Adam (Adaptive Moment Estimation)

Na meranie chyby sa použila funkcia `CrossEntropyLoss`.

## Architektúra modelu

Ako aktivačné funkcie som použil ReLU.

Vstupná vrstva: Rozmer  $28 \times 28$  (sploštené do 784 prvkov) keďže trénujeme model na obrázkoch s rozmermi 28 na 28 pixelov

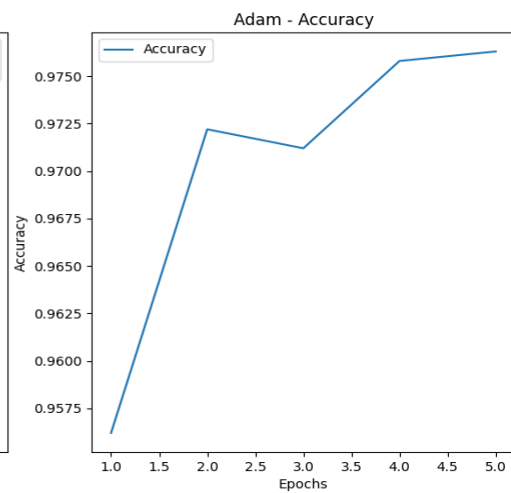
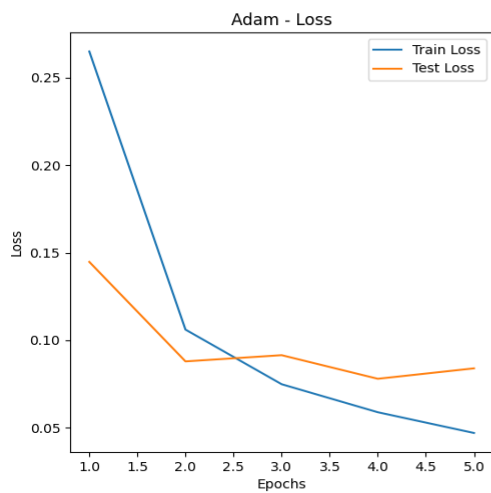
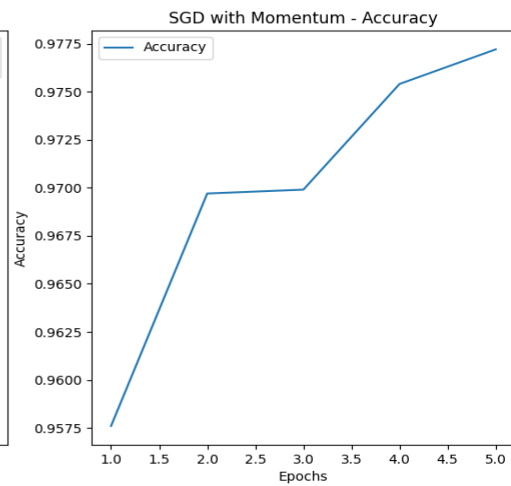
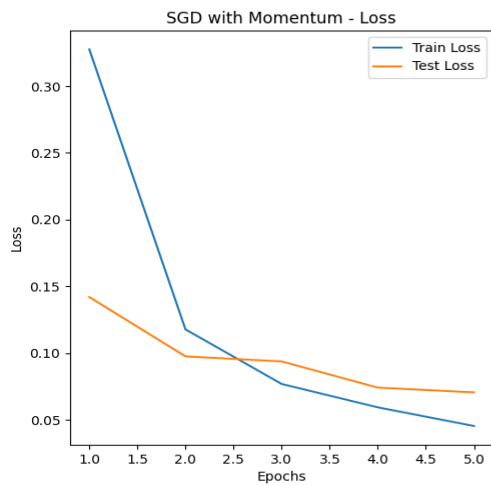
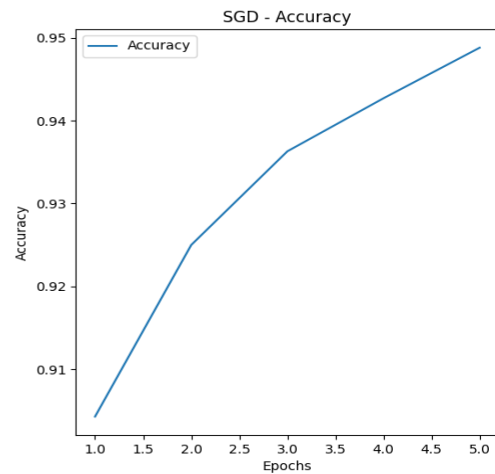
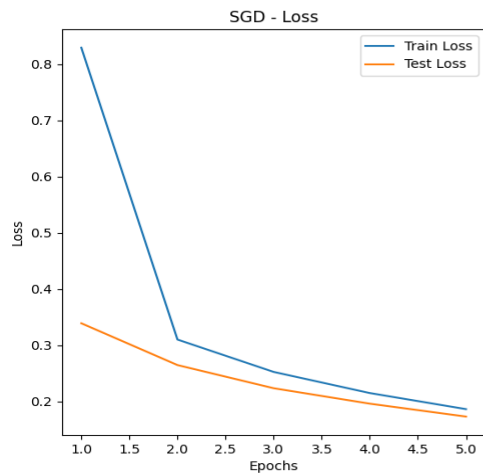
Skrytá vrstva 1: 128 neurónov + ReLU

Skrytá vrstva 2: 64 neurónov + ReLU

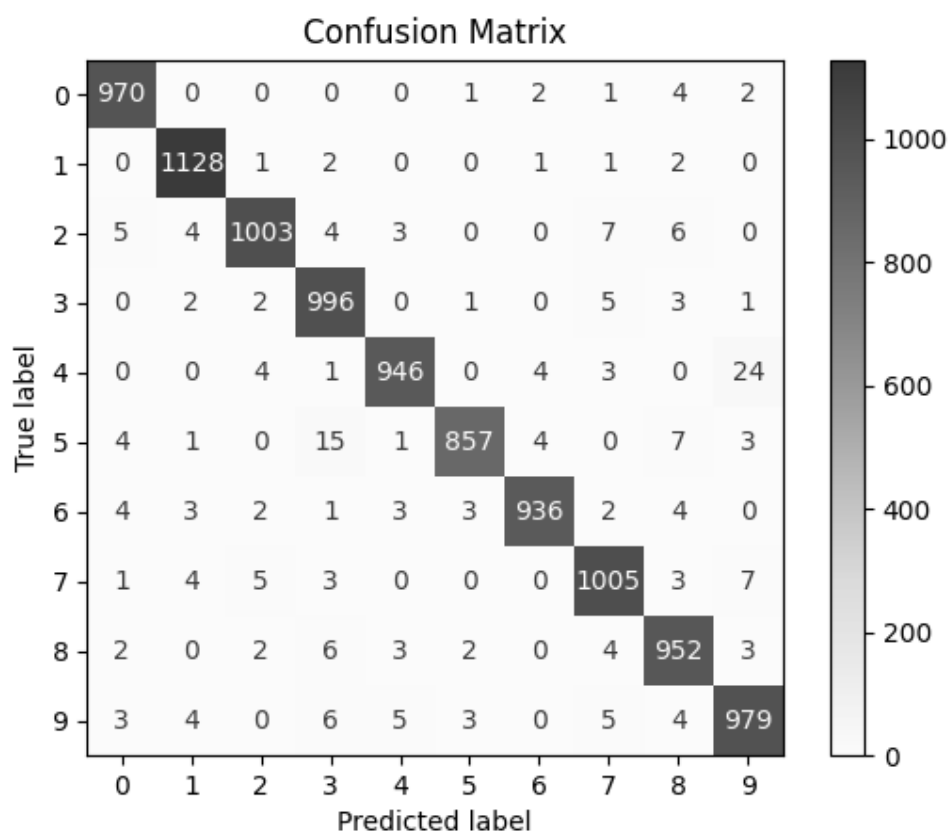
Výstupná vrstva: 10 neurónov (softmax implicitne zahrnutý v `CrossEntropyLoss`)

## Experimentovanie

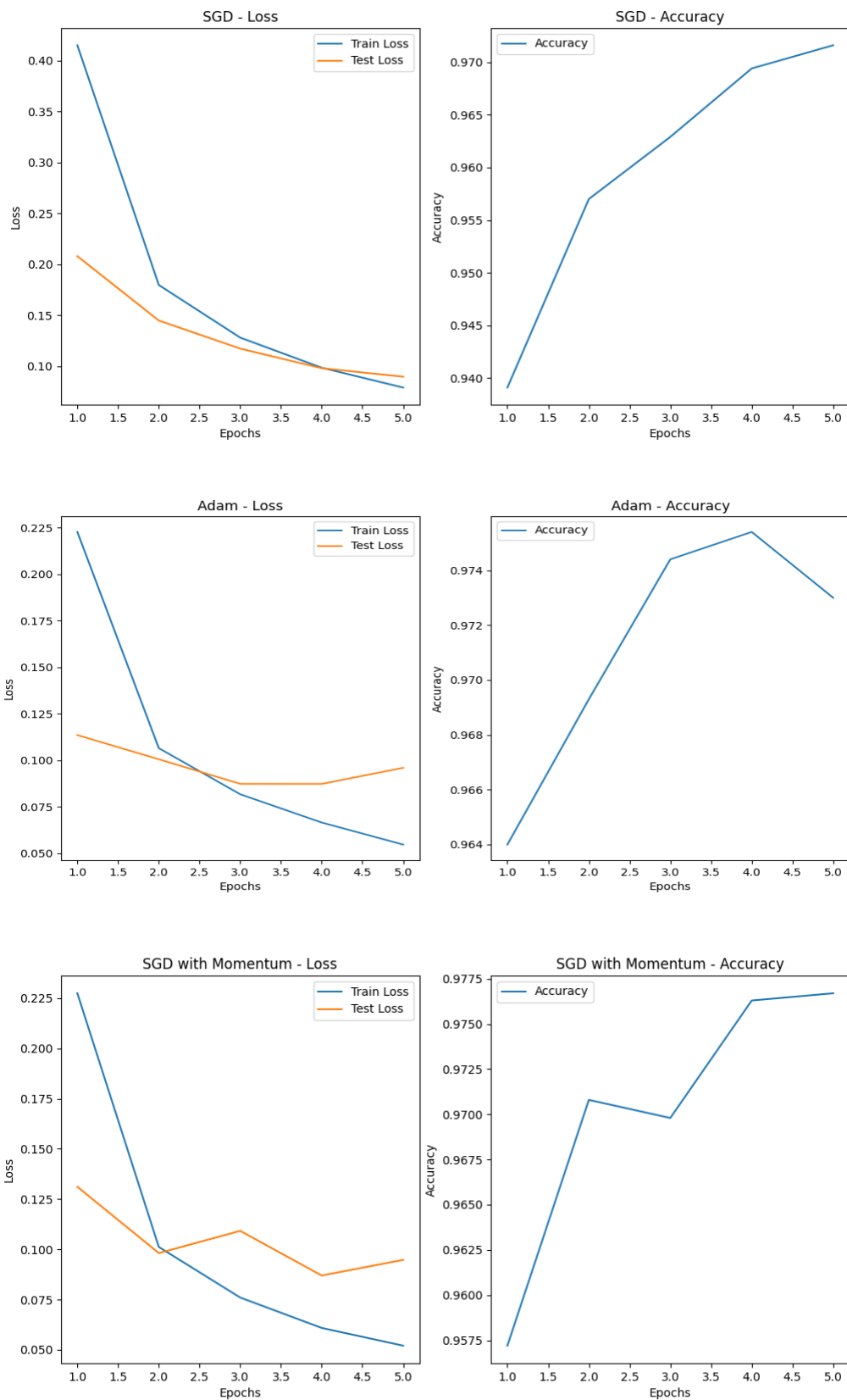
Epochy	5
Batch size	64
Learning r. - SGD	0,01
Learning r. – ADAM	0,001
Momentum	0,9



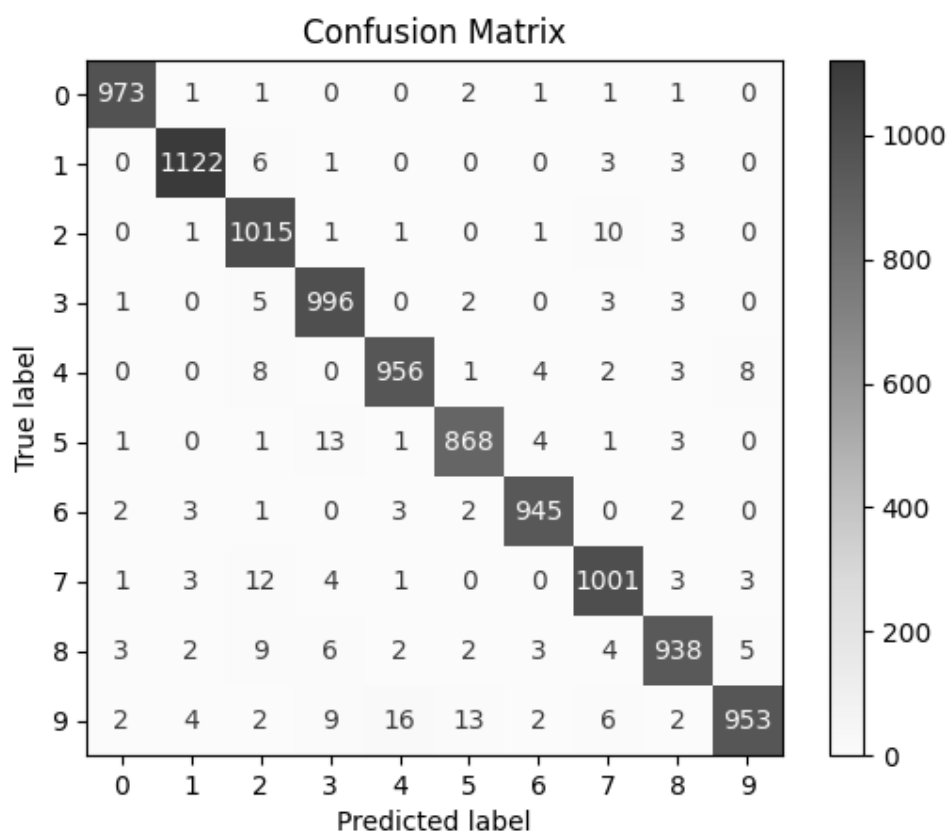
Najúspešnejšia optimalizačná funkcia je SGD s momentom s najmenšou chybovosťou. No s porovnaním s ADAM funkciou je praktický ekvivalentná



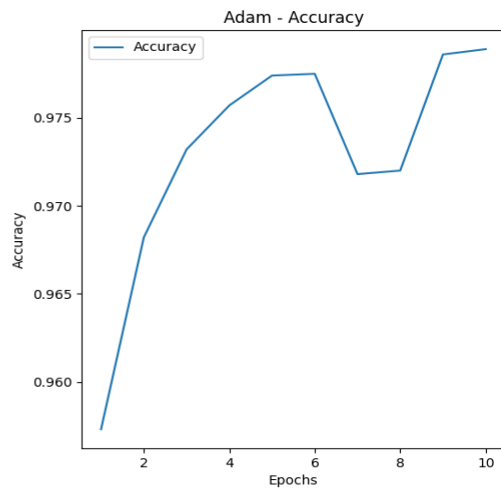
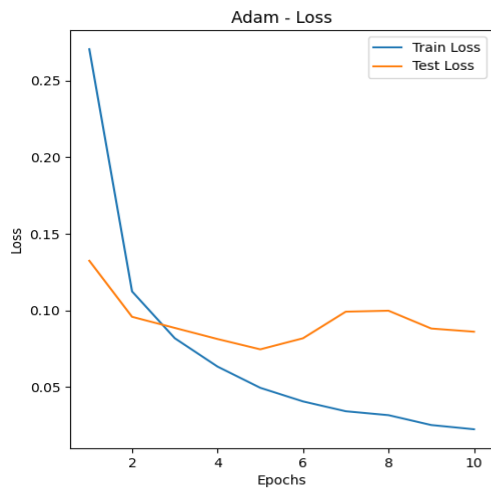
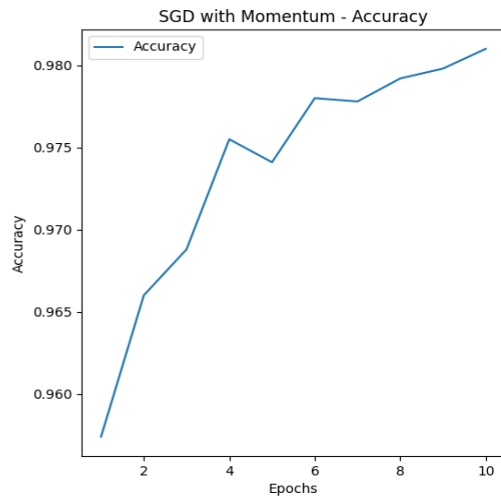
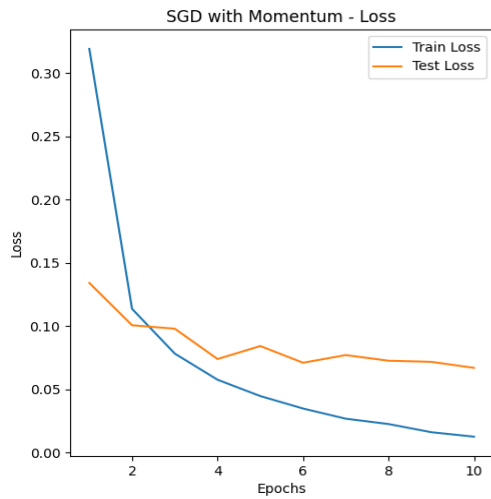
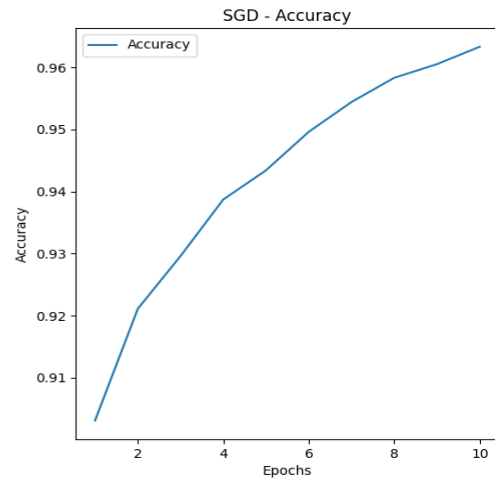
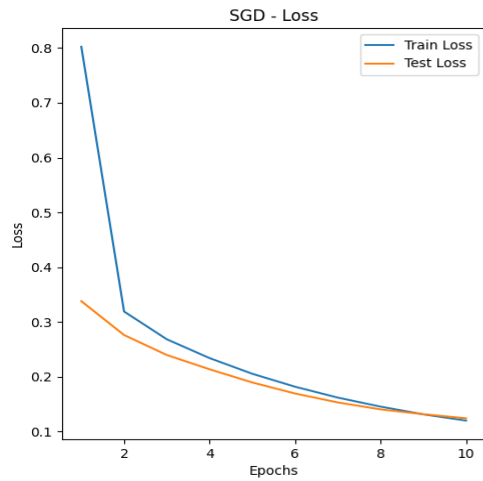
Epochy	5
Batch size	16
Learning r. - SGD	0,01
Learning r. – ADAM	0,001
Momentum	0,9



Opäť je SGD s momentum najúspešnejšia optimalizačná funkcia no oproti veľkosti batch size 64 je veľkosť 16 o niečo úspešnejšia

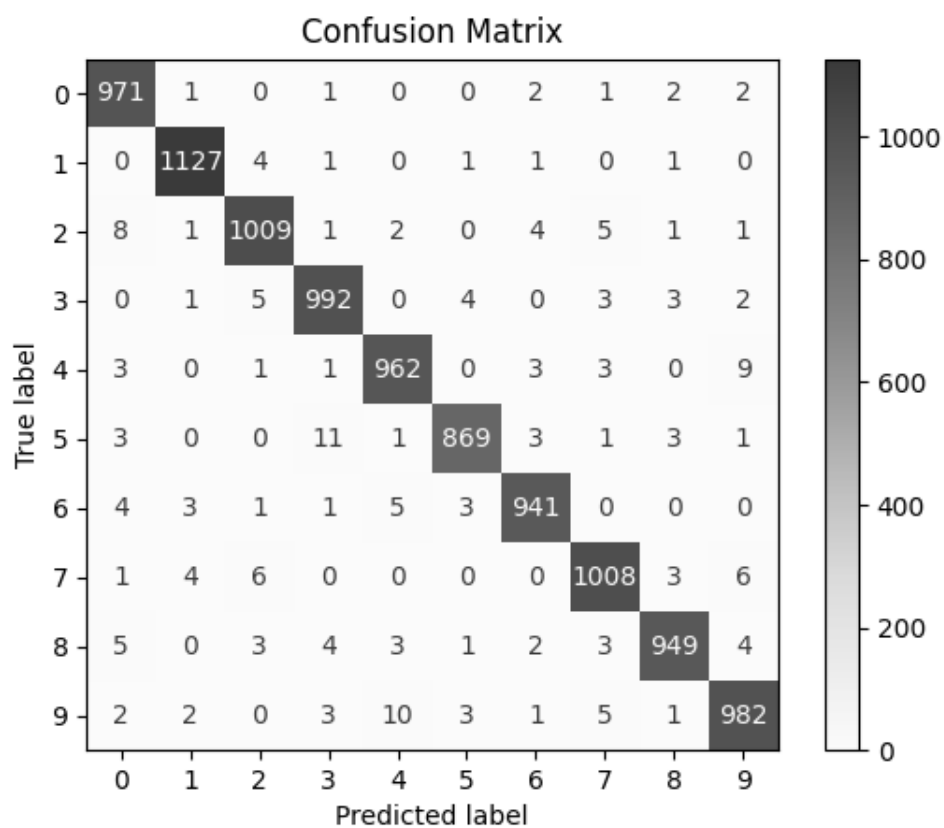


Epochy	10
Batch size	64
Learning r. - SGD	0,01
Learning r. – ADAM	0,001
Momentum	0,9

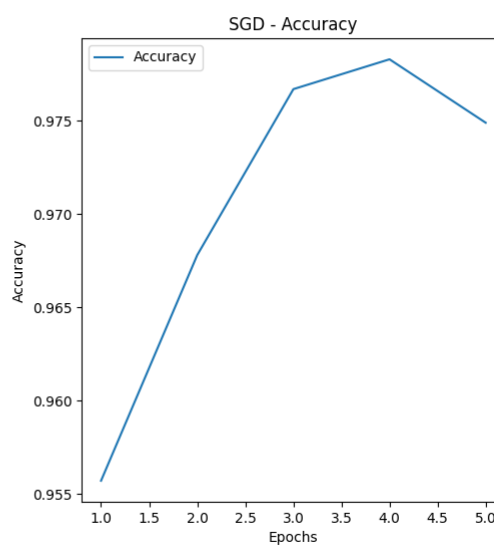
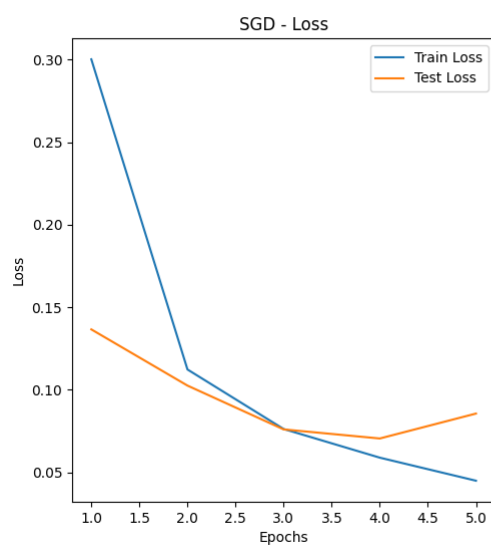


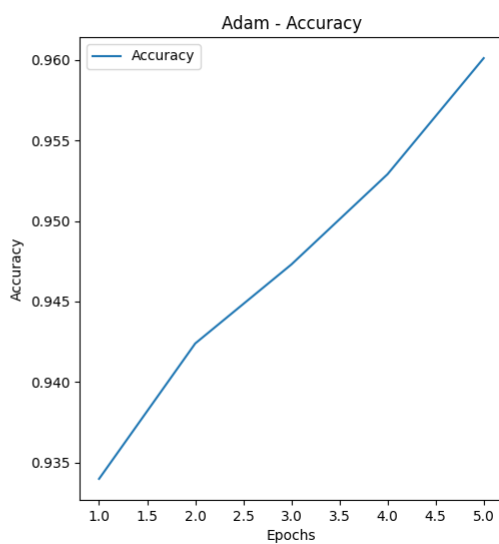
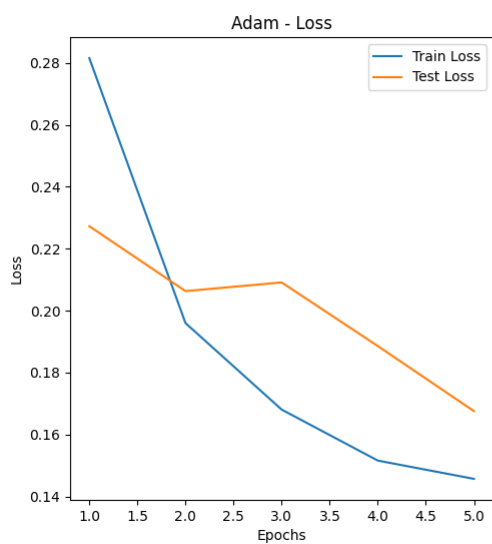
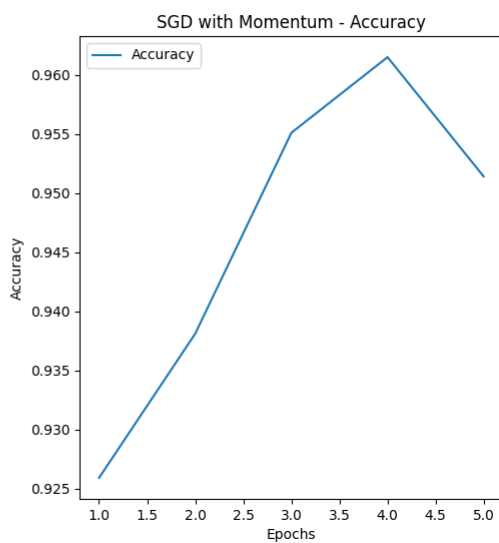
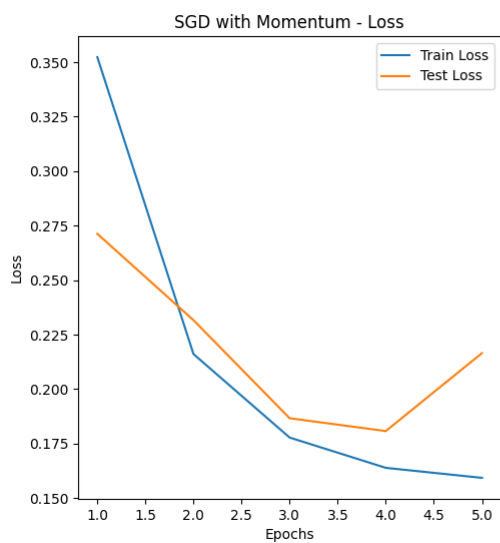
Opět je SGD s momentumem najoptimalnější algoritmus.



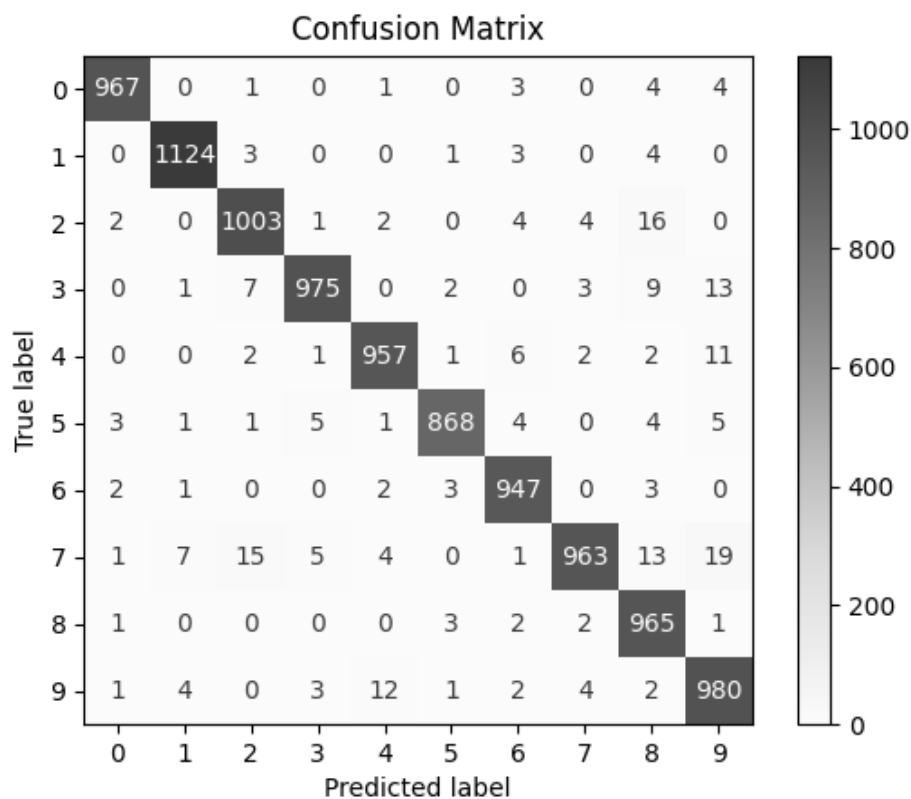


Epochy	5
Batch size	64
Learning r. - SGD	0,1
Learning r. – ADAM	0,01
Momentum	0,9





Prekvapivo v tomto prípade je SGD bez minima najúspešnejšia optimalizačná funkcia



## Zhodnotenie

Vytvorený model neuronovej siete sme trénovali na datasete obrázkov čísel od 0 až po 9. Na trénovanie modelov sme použili viacero optimalizačných funkcií, t. j. SGD, SGD s momentom a ADAM. Po experimentovaní s parametrami testovania sme zistili že zväčšovanie epoch testovania očakávane zlepšuje presnosť a znižuje chybovosť modelu. Znižovanie batch size má taktiež priaznivý efekt na efektivitu modelu no predpovedám že existuje hranica kedy to už neplatí. Learning rate má užšie okno pre úspešnosť optimalizačných funkcií pri zvýšení learning ratu najúspešnejší algoritmus bol SGD no nebol taký úspešný ako SGD s momentom s rovnakými parametrami ale menším learning ratom. Spomedzi testovaných optimalizačných funkcií SGD s momentom má najlepšie výsledky.

## Podúloha: Backpropagation algoritmus

## Riešený problém

Riešeným problémom bolo vytvorenie plne funkčného algoritmu backpropagation, ktorý je základným kameňom učenia neurónových sietí. Cieľom bolo umožniť neuronovej sieti efektívne sa učiť minimalizáciou chybovej funkcie prostredníctvom výpočtu gradientov a postupnej úpravy parametrov siete (váh a biasov).

## Riešenie

Na riešenie problému XOR bola použitá dvojvrstvová dopredná neurónová sieť (MLP) trénovaná algoritmom spätného šírenia chyby (backpropagation). Model obsahuje 4 neuróny v skrytej vrstve a 1 neurón vo výstupnej vrstve. Tréning spočíva v optimalizácii váh prostredníctvom gradientného zostupu a minimalizácie chyby.

Moja implementácia neuronovej siete sa skladá z listu prechodov medzi vrstvami. Tento list sa skladá z lineárnych a aktivačných vrstiev.

### *Linearna vrstva*

Reprezentuje prechod medzi vrstvami neuronovej siete. Využíva maticu váh a biasov na výpočet výstupov na zo vstupov. Na inicializáciu váh používam inicializačnú funkciu XAVIER, ktorú som si zvolil kvôli tomu že mojou aktivačnou funkciou je Tanh. Linearna vrstva taktiež obsahuje gradienty váh a biasov ktoré sa aplikujú v backward propagation a momentum chyby.

### *Aktivačné funkcie*

Sigmoid: Používa sa vo výstupnej vrstve na škálovanie výsledkov medzi 0 a 1.

Tanh: Používa sa v skrytej vrstve na nelinearitu, ktorá umožňuje modelu riešiť nelinearne problémy ako XOR.

### *Chybová funkcia*

Používam takzvanú chybovu funkciu Mean Squared error. Pri doprednom chode sa vypočíta priemerná chybová hodnota umocnená na 2, a následne sa pri backpropagation vypočíta jej derivácia. Podľa ktorej sa upravujú váhy neurónov.

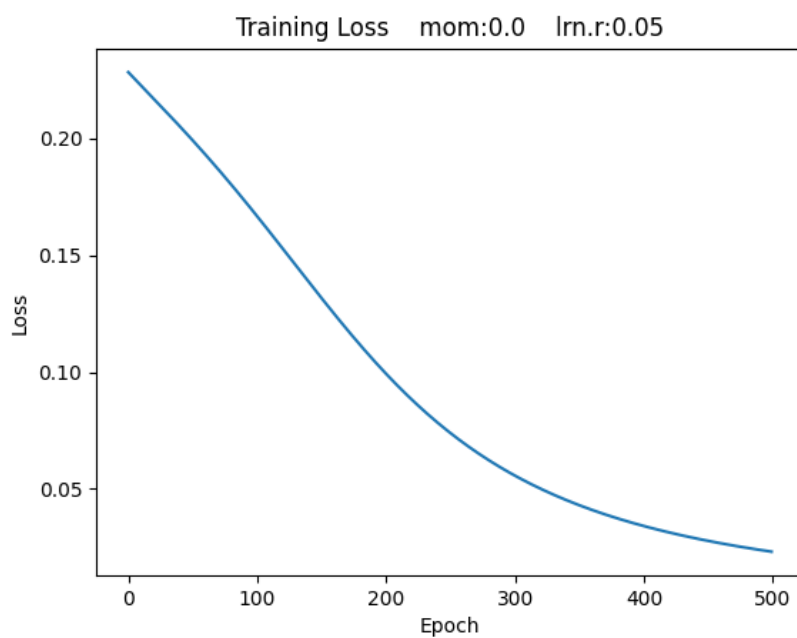
### *Použitý algoritmus: Backpropagation*

Backpropagation je postup, ktorý iteratívne aktualizuje váhy neuronovej siete na základe chyby. Skladá sa z dvoch hlavných krokov:

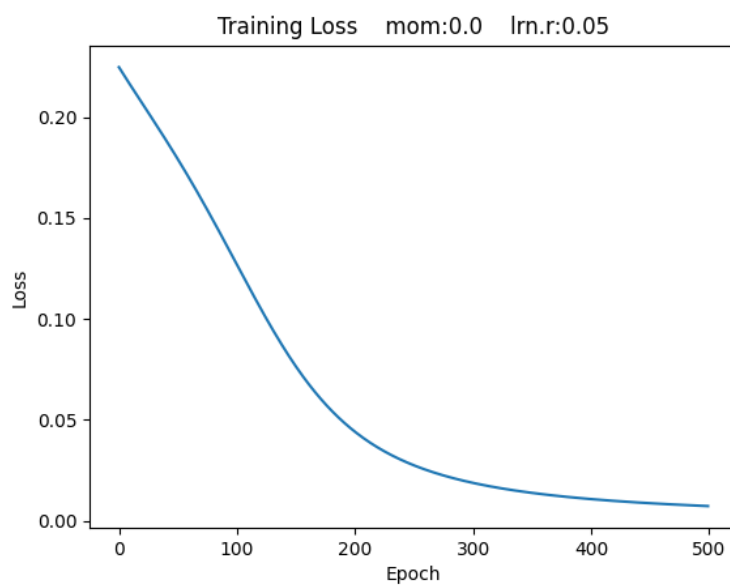
1. Dopredný prechod (Forward pass)  
Vstupné dáta sú prenášané cez sieť a na každej vrstve sa aplikujú lineárne kombinácie váh a aktivačné funkcie. Na výstupe dostávame predikciu.
2. Spätný prechod (Backward pass)  
Po výpočte chyby sa pomocou reťazového pravidla postupne vypočíta gradient chyby a upravujú sa váhy a biasy v jednotlivých vrstvách.

## Experimentovanie

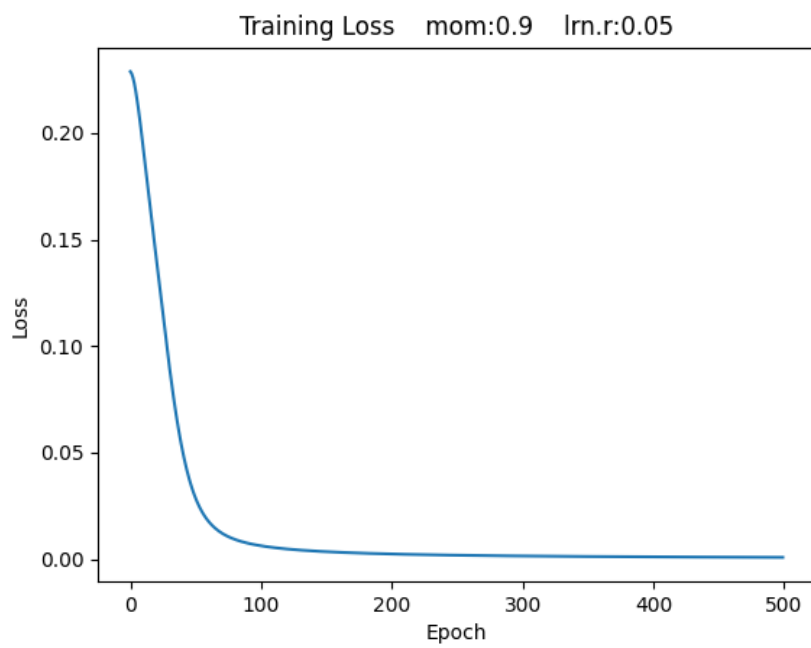
### Problem XOR



*1 skrytá vrstva*

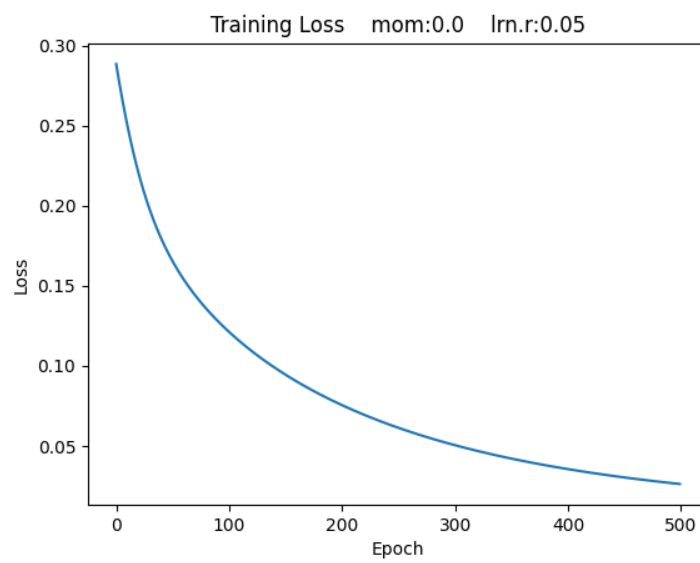


*2 skryté vrstvy*

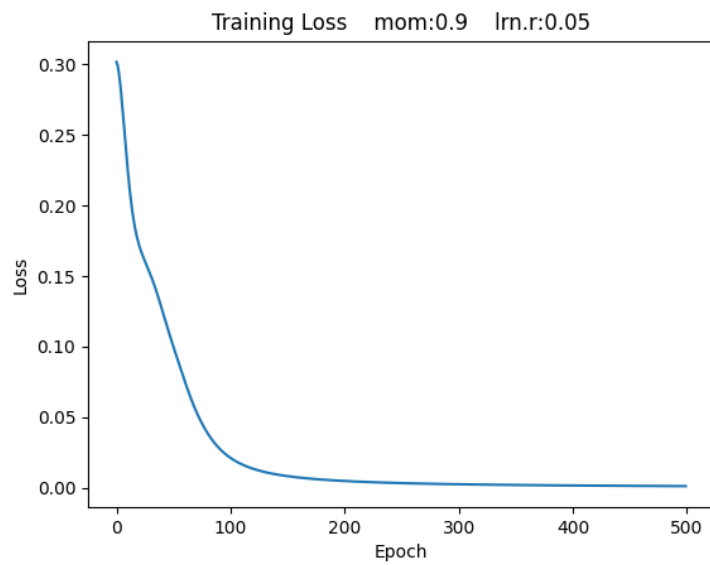


1 skrytá vrstva

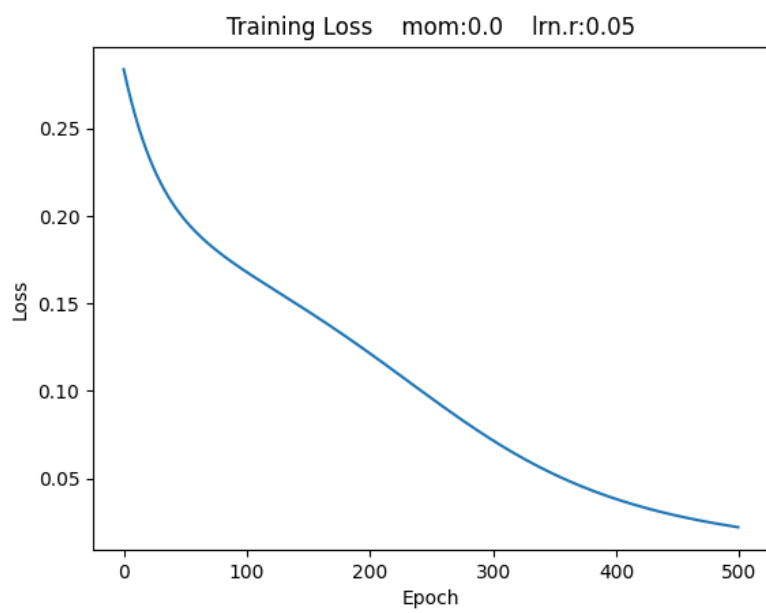
Problém OR



1 skrytá vrstva



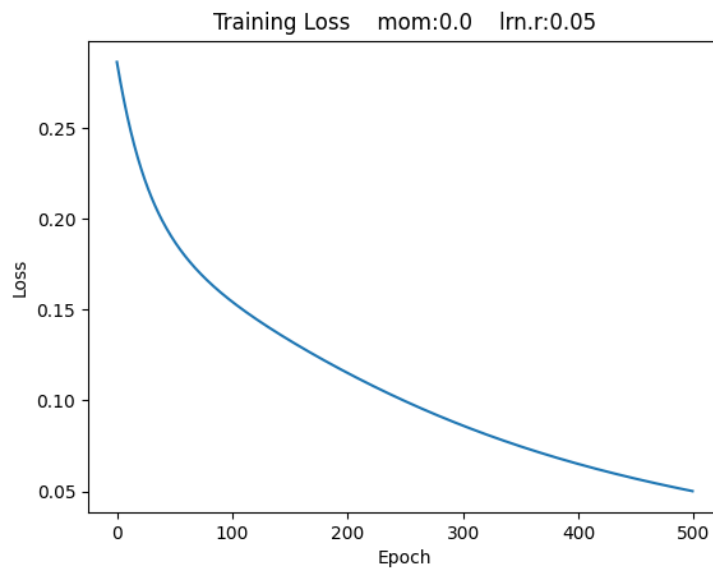
1 skrytá vrstva



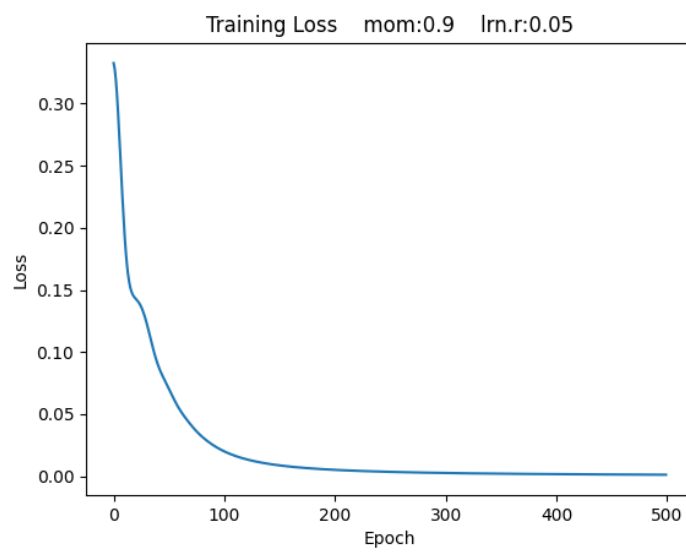
2 skryté vrstvy



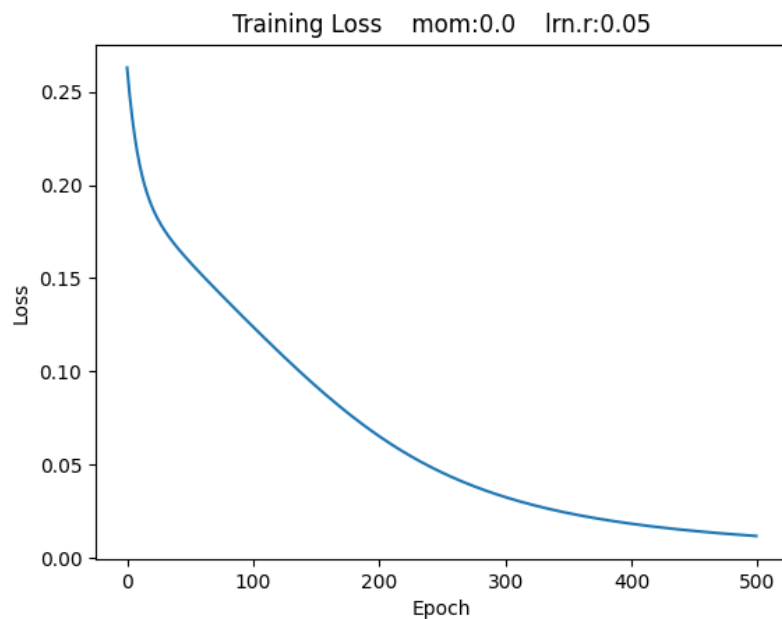
## Problém AND



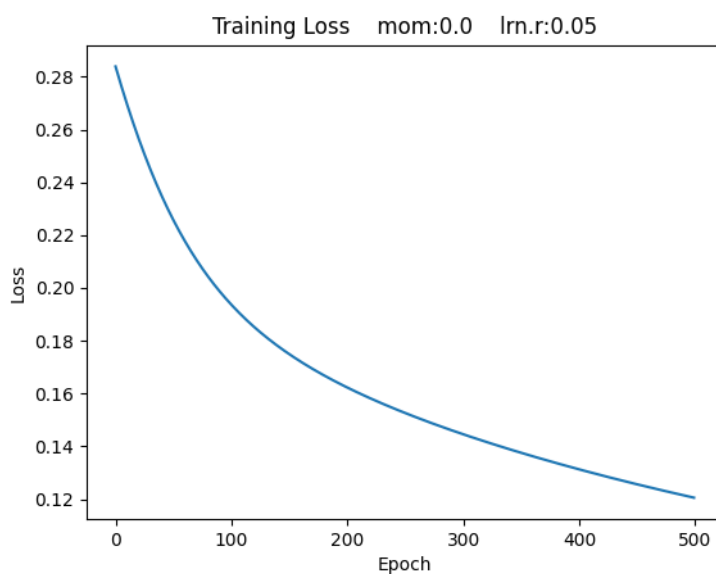
*1 skrytá vrstva*



*1 skrytá vrstva*



*2 skryté vrstvy*



*0 skrytých vrstev*

## Zhodnotenie

Implementoval som neuronovú sieť ktorú som následne trénoval na jednoduchých booleanovských operáciach. Neuronovú sieť som testoval s 0 až 2 skrytými vrstvami. Dostal som veľa rôznych výsledkov najväčší efekt malo momentum chyby ktoré veľmi urýchlilo trénovanie modelu.