Správa o realizácii projektu - Umelé nnurónové siete*

Šimon Vinkler
STU Bratislava
Fakulta Informatiky a Informačných technológií STU v Bratislave
xvinklers@stuba.sk

1. decembra 2024

*3. zadanie v predmete Umelá inteligencia akad. rok 2024/2025

Obsah

1 Znenie zadania

Projekt sa skladá z dvoch zadaní. Cieľom prvej časti je vytvoriť doprednú neurónovú sieť (viacvrstvový perceptrón, MLP) na klasifikáciu ručne písaných číslic z datasetu MNIST a vyhodnotiť výkonnosť modelu pomocou troch optimalizačných algoritmov: SGD, SGD s momentom a Adam. Cieľom druhej časti je implementovať algoritmus backpropagation bez použitia autograd knižníc na trénovanie jednoduchej doprednej neurónovej siete na riešenie logických problémov AND, OR a XOR.

2 Prvá časť

Ako už bolo zmienené, prvá časť sa bude zaoberať identifikáciou ručne písaných čísel z datasetu MNIST.

2.1 Príprava Dát

Príprava dát sa skladá z dvoch častí:

1. Načítanie datasetu:
   - Dataset MNIST bude načítaný z torchvision.datasets.
   - Dáta budú rozdelené na trénovaciu (60 000) a testovaciu množinu (10 000).

```
train_dataset = datasets.MNIST(root='./data',
    train=True, transform=transform, download=
    True)
test_dataset = datasets.MNIST(root='./data',
    train=False, transform=transform, download=
    True)
```

2. Predspracovanie dát:
   - **Normalizácia:** Prevedenie hodnôt pixelov na rozsah 0 až 1.

```
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])
```

   - Cieľové hodnoty budú spracované ako celé čísla (bez one-hot encodingu, použijeme CrossEntropyLoss).

```
criterion = nn.CrossEntropyLoss()
```

2.2 Architektúra modelu

Model neurónovej siete sa skladá zo 4 vrstiev - jednej vstupnej, dvoch skrytých a jednej výstupnej.

1. **Vstupná vrstva** sa skladá z celkovo (28 x 28 =) 784 vstupných neurónov, pričom každý neurón reprezentuje jeden pixel

2. **Prvá skrytá vrstva** obsahuje celkom 128 neurónov

3. **Druhá skrytá vrstva** obsahuje 64 neurónov

4. **Výstupná vrstva** obsahuje presne 10 neurónov, každý presne pre jednu číslicu (rozsah 0-9) s využitím LogSoftmax.

2.3 Konfigurácia hyperparametrov

Prvá konfigurácia nie je úplne optimalizovaná, Learning rate pre SGD je stále príliš vysoký a je potrebné ho znížiť. Learning rate pre Adam si vyžaduje taktiež korekciu. Momentum by mohlo byť tiež mierne upravené a zvýšené. Počet epôch je dostatočný. Model spĺňa minimálne požadované výsledky - mierne nad 97% pre každú optimalizačnú metódu, avšak napríklad u SGD s momentom vykazuje známky pretrénovania - mierne klesá úspešnosť.

Tabuľka 1: Hyperparameters of the MNIST Classifier Model, 1. configuration

Parameter	**Value**
Batch size	64
Number of epochs	25
Learning rate (SGD)	0.08
Learning rate (Adam)	0.002
Optimizers	SGD, SGD + Momentum, Adam
Momentum (SGD)	0.6
Loss function	CrossEntropyLoss
Activation functions	ReLU, LogSoftmax

Druhá konfigurácia je úspešnejšia, má plynulejší priebeh, avšak na grafoch je možné spozorovať pretrénovanie po približne 22. epoche. Veľmi zjavné je pretrénovanie u optimalizácie Adam, kde začína po 20. generácii stúpať test loss hodnota a klesať presnosť. pre ešte plynulejší priebeh a lepšiu presnosť by bolo vhodné znížiť hodnotu learning rate pre SGD a zvýšiť momentum.

Tabuľka 2: Hyperparameters of the MNIST Classifier Model, 2. configuration

Parameter	**Value**
Batch size	64
Number of epochs	25
Learning rate (SGD)	0.04
Learning rate (Adam)	0.001
Optimizers	SGD, SGD + Momentum, Adam
Momentum (SGD)	0.6
Loss function	CrossEntropyLoss
Activation functions	ReLU, LogSoftmax

Tretia konfigurácia vykazovala lepšie výsledky pre SGD s momentom, avšak samotné SGD malo v porovnaní s predchádzajúcimi konfiguráciami vyššiu kumulatívnu chybu. Tento jav bol zapríčinený znížením hodnoty learning rate na 0.01 pri ponechaní rovnakého počtu epôch. Model nemal dostatočný priestor na postačujúce natrénovanie.

Tabuľka 3: Hyperparameters of the MNIST Classifier Model, 3. configuration

Parameter	**Value**
Batch size	64
Number of epochs	25
Learning rate (SGD)	0.01
Learning rate (Adam)	0.001
Optimizers	SGD, SGD + Momentum, Adam
Momentum (SGD)	0.6
Loss function	CrossEntropyLoss
Activation functions	ReLU, LogSoftmax

Štvrtá konfigurácia vykazovala prijateľné a stabilné výsledky. Optimalizátor Adam by síce uvítal nižší počet epôch no ostatné optimalizátory pracujú spoľahlivo. Táto konfigurácia reprezentuje finálny výsledok pokusov. Malou korekciou prešla hodnota learning rate u SGD a tiež hodnota momenta. Najstabilnejší bol optimalizátor SGD.

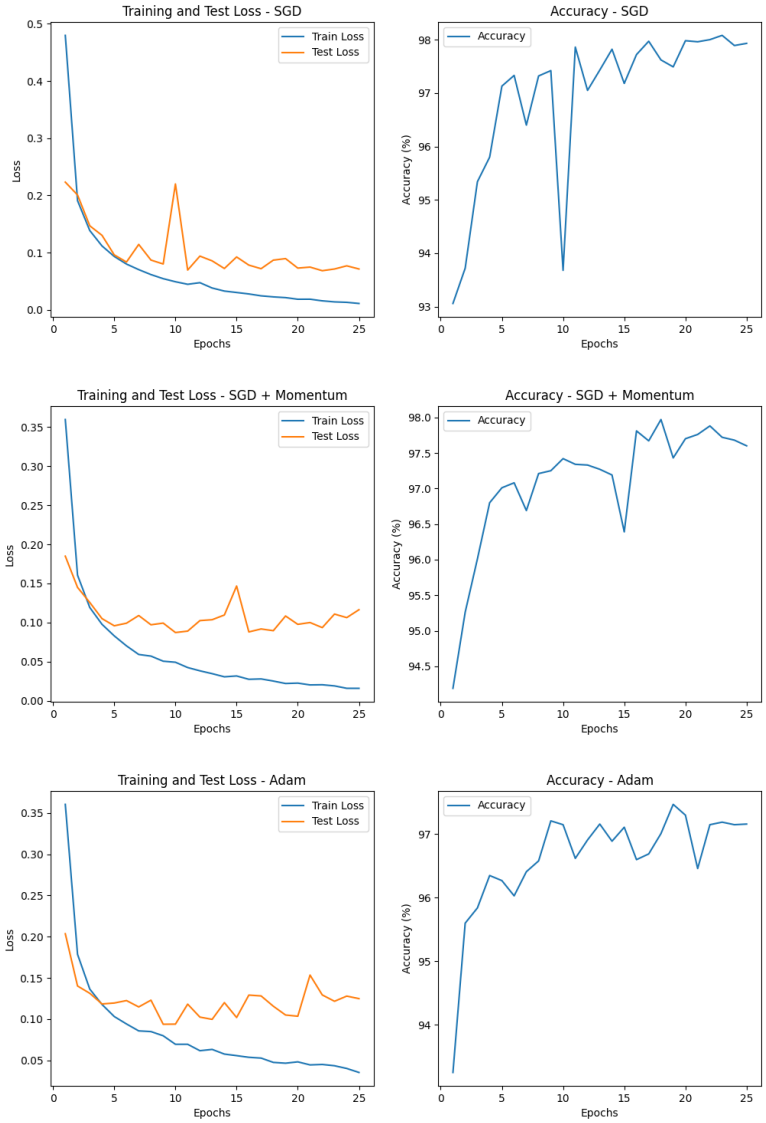
Tabuľka 4: Hyperparameters of the MNIST Classifier Model, 3. configuration

Parameter	**Value**
Batch size	64
Number of epochs	25
Learning rate (SGD)	0.02
Learning rate (Adam)	0.002
Optimizers	SGD, SGD + Momentum, Adam
Momentum (SGD)	0.8
Loss function	CrossEntropyLoss
Activation functions	ReLU, LogSoftmax

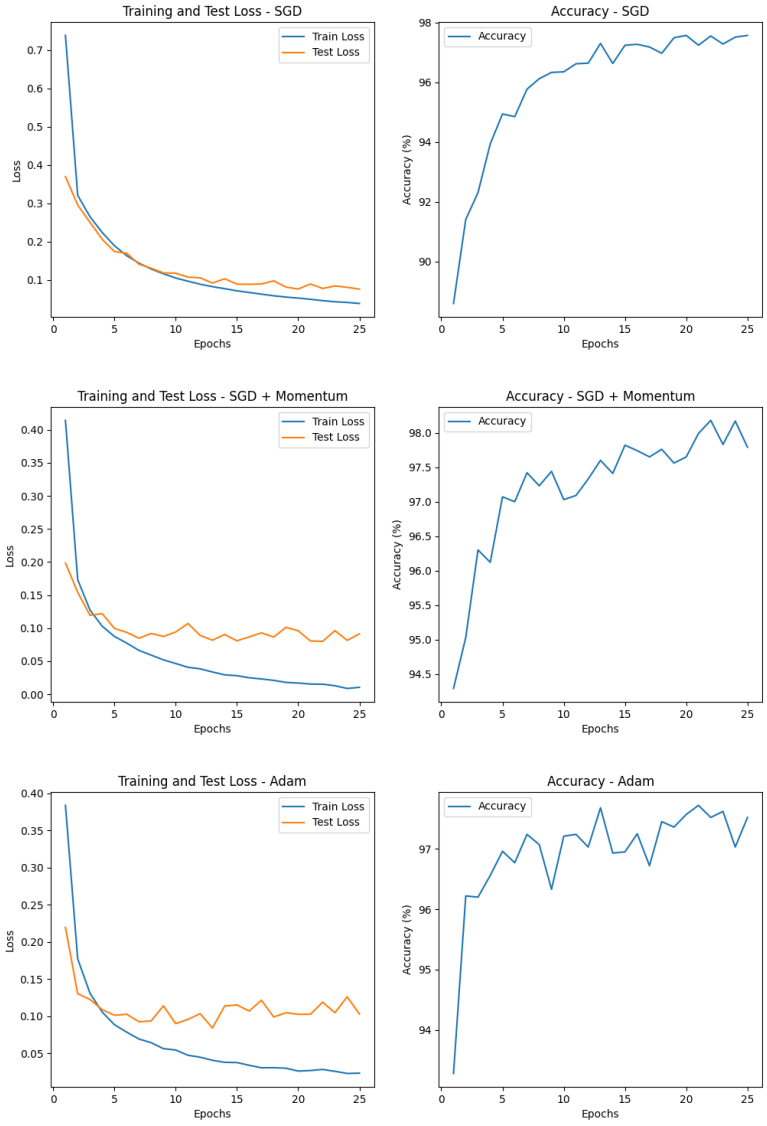
Piata konfigurácia je čisto test úspešnoti pre veľkosť dávky 32. Táto konfigurácia nedosahovala podobnú úspešnosť ako 4. konfigurácia a bola menej presná.

Tabuľka 5: Hyperparameters of the MNIST Classifier Model, 3. configuration

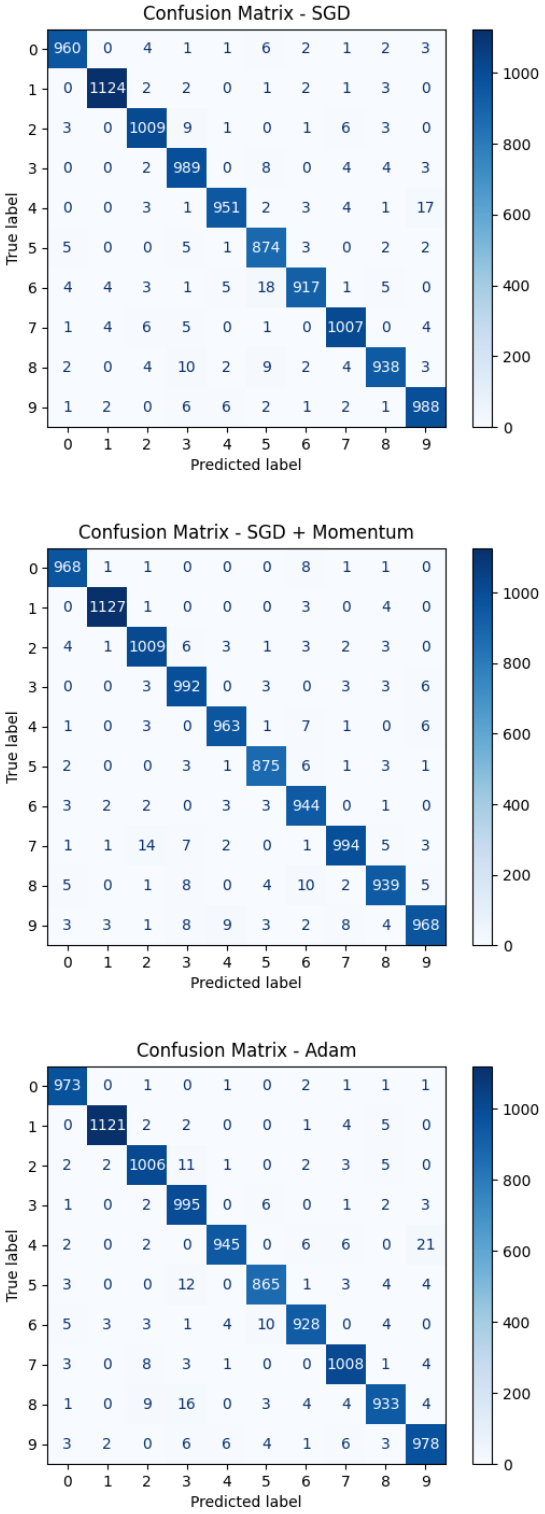
Parameter	Value
Batch size	32
Number of epochs	25
Learning rate (SGD)	0.02
Learning rate (Adam)	0.002
Optimizers	SGD, SGD + Momentum, Adam
Momentum (SGD)	0.8
Loss function	CrossEntropyLoss
Activation functions	ReLU, LogSoftmax



Obr. 1: 1. konfigurácia - grafy



Obr. 2: Najlepší model (4. konfigurácia) - confusion matrices



Obr. 3: Najlepší model (4. konfigurácia) - confusion matrices

2.4 Vyhodnotenie

Najúspešnejší model bol dosahoval stabilné a pomerne dobré výsledky. Najväčšie problémy mal však s rozpoznaním čísla 4, toto bolo zamieňané s číslom 9. Ďalej s číslom 6, toto bolo zamieňané s číslom 5. Najúspešnejšia architektúra bola 128 a 64 neurónov v dvoch skrytých vrstvách. Pri polovičnom počte sa model nezvládal v dostatočnej presnosti učiť pri počte epoch 25. Taktiež obptimálny batch size pre najlepší model bol 64. Vyhodnotenie jednotlivých optimalizátorov:

- **SGD:** Konvergencia je pomalšia, model vyžaduje viac epoch pre dosiahnutie primeranej presnosti, avšak, vychádza z pokusov najlepšie.
- **SGD + Momentum:** Rýchlejšia konvergencia ako čistý SGD, menšie kolísanie počas tréningu.
- **Adam:** Najlepšia presnosť a najrýchlejšia konvergencia, zvlášť pri menších rýchlostiach učenia.

3 Druhá časť

Ako už bolo zmienené, druhá časť sa bude zaoberať predikciou výslednej hodnoty logických operácií, nazýva sa taktiež ako XOR problém. Jedná sa o implementáciu plne funkčného algoritmu backpropagation bez použitia autograd knižníc ako napríklad PyTorch alebo TensorFlow. Implementovaný algoritmus je validovaný na multi-layer perceptrone (MLP) na XOR probléme. Model podporuje doprednú aj spätnú propagáciu výsledku, taktiež korekciu váh s a bez využitia momenta.

3.1 Rozbor modelu

Použitý prístup je modulárna neurónová sieť s vrstvami a aktivačnými funkciami. Každý modul je schopný vykonať forward a backward posun. Sieť je trénovaná za použitia MSE (= Mean Squared Error) loss funkcie. Kľúčové komponenty implementácie:

1. **Lineárna vrstva:** vykonáva výpočet hodnoty neurónu y = Wx + b
2. **Aktivačné funkcie:**
   - ReLU (Rectified Linear Unit): f(x) = max(0,x)
   - Sigmoid: f(x) = 1 / (1+e -x)
   - Tanh: f(x) = ((e x) - (e -x)) / ((e x) + (e -x))
3. **Loss funkcia:** Mean Squared Error (MSE)
4. **Backward Propagation:** Reťazovo založený výpočet gradientu na úpravu váh
5. **Trnénovacia konfigurácia:** Algoritmus Gradient Descent s a bez momenta

3.2 Detaily implementácie

Každý modul je navrhnutý tak, aby bol schopný vykonať spätné a dopredné posuny. Dopredný posun počita hodnotu neurónu Wx + b. Spätný posun ráta gradient/deriváciu pre korekciu váh a biasov. Aktivačné funkcie sú pri metóde forward aplikované na výpočty neurónov, pri metóde backward sa ráta ich derivácia - gradient. MSE loss funkcia ráta odchýlku od reálnej hodnoty pro forward a pre backward ráta deriváciu vzhľadom na predikovanú hodnotu. Úpravy váh sú počítané: W = W - learning_rate grad_w bez momenta, pri momente sa rieši rýchlosť nárastu, ktorá sa vypočíta ako: momentum predch. rýchlosť + learning_rate grad_w. Táto hodnota je nová rýchlosť gradientu, je odčítaná od prechádzajucej váhy.

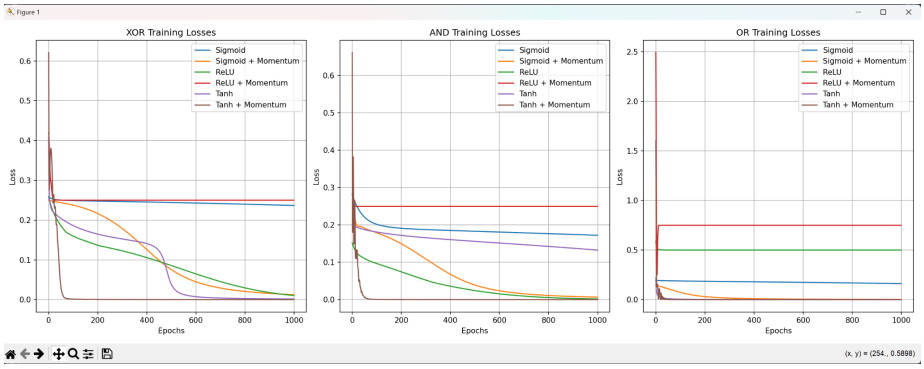
3.3 Pokusy

Bolo vykonaných viac ako 10 pokusov, do dokumentácie budú však priložené grafy len z 3, z toho jeden graf z viacvrstvej architektúry.

Prvá konfigurácia nie je úplne optimalizovaná, Learning rate pre XOR pri sigmoide konvergoval pomaly, vysledky pre sigmoid s momentom už boli pomerne postačujúce. AND a OR pri sigmoide vyžadovali ešte miernu korekciu learning rate a momenta.

Relu bez momenta vykazovalo pri každom pokuse pre XOR a miestami aj pre OR konštanté hodnoty loss funkcie iba s krátkou konvergenciou. AND mal väčšiu úspešnosť konvergencie.

Tanh funkcia mala dobrý priebeh už pri aktuálnej konfigurácii a nevyžadovala výrazné korekcie.

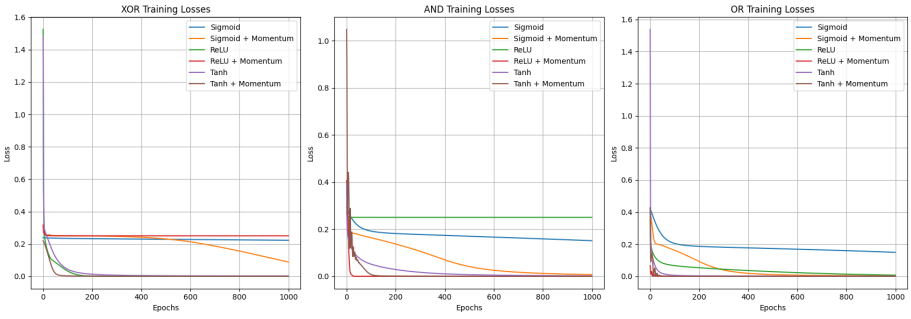


Obr. 4: Prvá konfigurácia

Gate Type	Activation Function	Learning Rate	Momentum
XOR	Sigmoid	0.10	0.0
XOR	Sigmoid + Momentum	0.06	0.9
XOR	ReLU	0.01	0.0
XOR	ReLU + Momentum	0.01	0.9
XOR	Tanh	0.10	0.0
XOR	Tanh + Momentum	0.10	0.9
AND	Sigmoid	0.05	0.0
AND	Sigmoid + Momentum	0.05	0.9
AND	ReLU	0.01	0.0
AND	ReLU + Momentum	0.01	0.9
AND	Tanh	0.10	0.0
AND	Tanh + Momentum	0.10	0.9
OR	Sigmoid	0.05	0.0
OR	Sigmoid + Momentum	0.05	0.9
OR	ReLU	0.10	0.0
OR	ReLU + Momentum	0.01	0.9
OR	Tanh	0.10	0.0
OR	Tanh + Momentum	0.10	0.9

Tabuľka 6: 1. zdokumentovaná konfigurácia

V poradí druhá zdokumentovaná konfigurácia (v realite finálny, najlepší dosiahnutý model) vykazovala pre sigmoidu a tanh stabilné hodnoty, najväčšiu presnosť mala funkcia tanh. Relu bez momenta väčšinou stagnovalo.

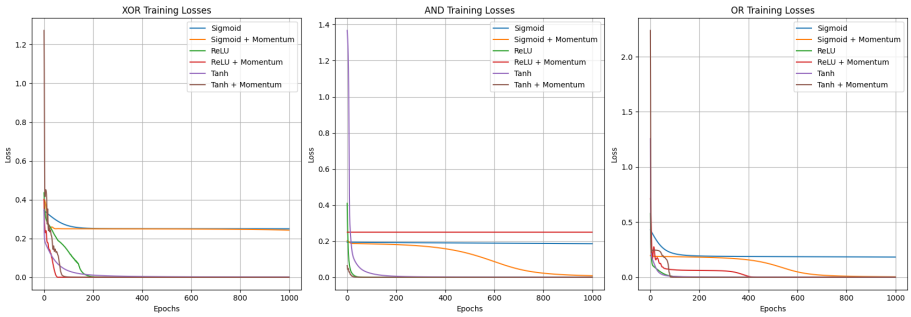


Obr. 5: 2. zdokumentovaná konfigurácia (finálny model)

Gate Type	Activation Function	Learning Rate	Momentum
XOR	Sigmoid	0.05	0.0
XOR	Sigmoid + Momentum	0.06	0.9
XOR	ReLU	0.05	0.0
XOR	ReLU + Momentum	0.05	0.9
XOR	Tanh	0.08	0.0
XOR	Tanh + Momentum	0.08	0.9
AND	Sigmoid	0.05	0.0
AND	Sigmoid + Momentum	0.05	0.9
AND	ReLU	0.05	0.0
AND	ReLU + Momentum	0.02	0.9
AND	Tanh	0.08	0.0
AND	Tanh + Momentum	0.08	0.9
OR	Sigmoid	0.05	0.0
OR	Sigmoid + Momentum	0.05	0.9
OR	ReLU	0.05	0.0
OR	ReLU + Momentum	0.02	0.9
OR	Tanh	0.08	0.0
OR	Tanh + Momentum	0.08	0.9

Tabuľka 7: 2. zdokumentovaná konfigurácia

Architektúra s 2 hidden layers nedosahovala dobré výsledky pre sigmoid. Ostatné funkcie maly oveľa lepšiu konvergenciu ako pri architektúre s jednou skrytou vrstvou. Bolo vykonaných viacero testov, avšak zdokumentovaný bude len jeden pre ilustráciu. (Tabuľka konfigurácií je zhodná s tabuľkou z finálneho modelu)



Obr. 6: Finálny model s 2 skrytými vrstvami, každá obsahovala 4 neuróny

3.4 Vyhodnotenie

Implementovaný algoritmus backpropagation bol schopnú úspešne natrénovať neurónovú sieť pre logické problémy AND, OR a XOR. Pri použití momenta mal finálny model najlepšiu konvergenciu v drvivej väčšine prípadov.

Najlepšie výsledky dosahovala funkcia tanh, ktorá najlepšie konvergovala či už s momentom alebo bez. Sigmoida a ReLU s momentom vykazovali taktiež pomerne obstojné výsledky pri porovnaní ich loss funkcií. Avšak, bez využitia momenta klesala ich úspešnosť pri viacerých pokusoch - vykazovali slabú konvergenciu až stagnáciu. Sigmoida bola však plynulejšia a stabilnejšia.

Viacvrstvá architektúra vykazovala najlepšie výsledky práve pre funkcie s vuyžitím momenta, sigmoida a ReLU bez momenta väčšinou stagnovali.

Počet epoch na ktorý bol finálny model trénovaný je 1000, avšak, tento počet by mohol byť o štvrtinu až polovicu nižší. Trénovanie a vykreslenie grafov loss funkcií prebehne do 5 sekúnd. [1]

[1] Zdroje:

- https://towardsdatascience.com/math-neural-network-from-scratch-in-python-d6da9f29ce65
- https://www.3blue1brown.com/topics/neural-networks
- korekcia štruktúry kódu a prípadný refactoring pri ponechaní pôvodnej logiky: ChatGPT, GitHub Copilot (Efektívnejší a čitatelnejší vývoj algoritmov)