SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

STROJOVÉ VIDENIE A VÝPOČTOVÁ INTELIGENCIA
ZADANIE 1

2021 **ADAM MÓRICZ**

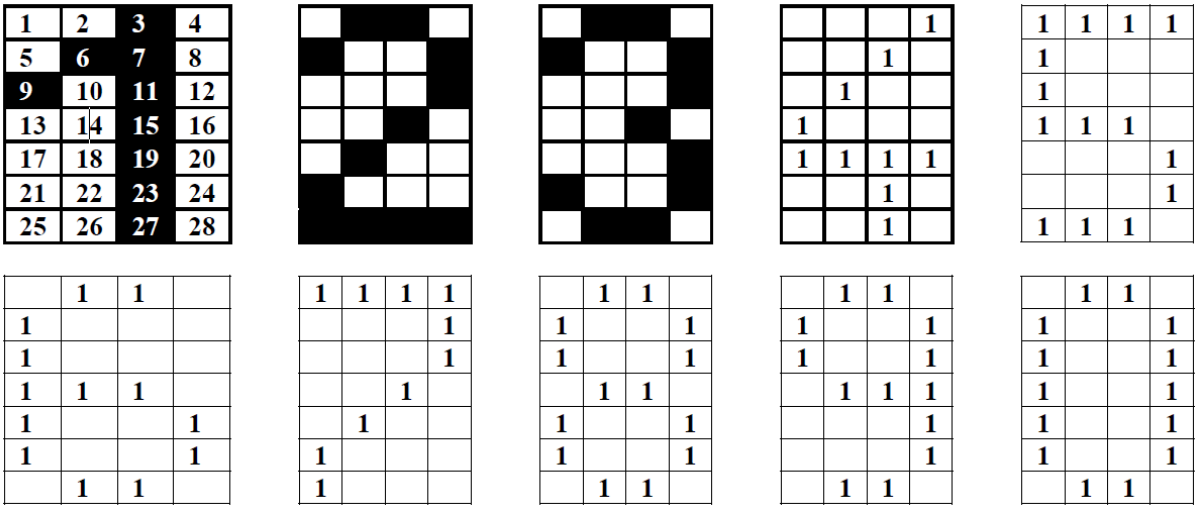
Zadanie

Našou úlohou bolo vytvoriť neurónovú sieť (ďalej len NS), ktorá bude rozpoznávať číslice od 0 po 9. Vstupom pre neurónovú sieť 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]



Po natrénovaní bude sieť schopná rozpoznávať všetky naučené znaky.

Úloha:

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

Kód si vieme rozdeliť na 4 časti. Prvá časť obsahuje *dataset*, ktorý je .csv súbor podobne ako na cvičeniach. Prvých 28 stĺpcov sú čísla 1 alebo 0 ktoré nám označuje, že ktoré pole je zafarbené. Posledný stĺpec nám označuje, že o ktoré číslo sa jedná na vstupných dátach od 0 po 9. Dataset obsahuje našich 10 čísiel v „čistej“ forme, t.j. v dátach sa nenachádza žiaden šum ani porušenie.

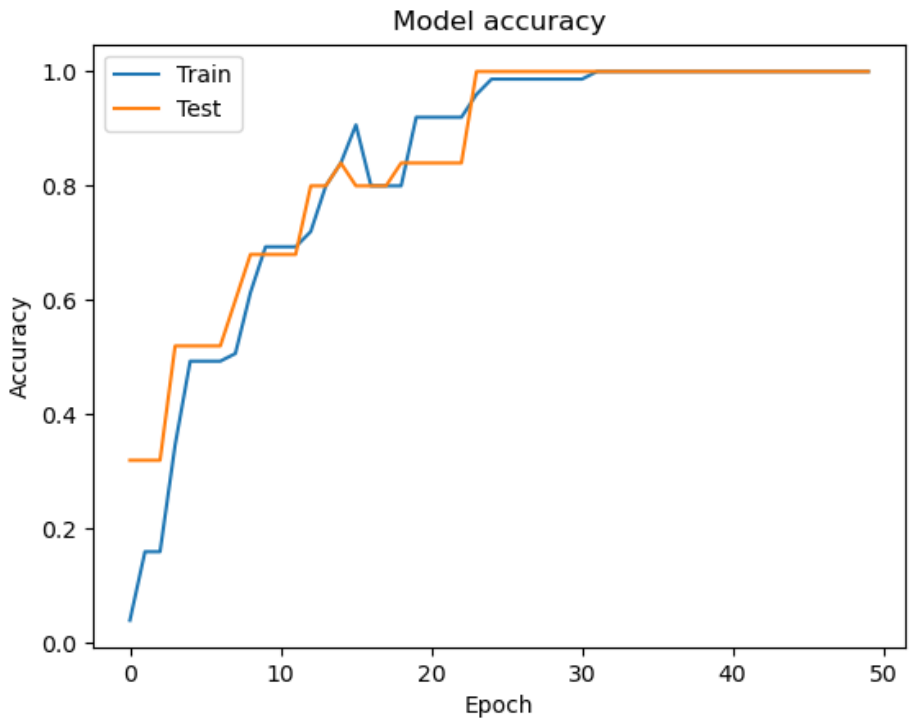
Každé číslo sa musí v datasete nachádzať viackrát, lebo z datasetu sa vyberie 25% dát ako testovací dataset. Dáta sa načítajú do premennej X a do y sa načíta posledný stĺpec, ktorý je naša závislá premenná. Po rozdelení dát, musíme zmeniť *y_train* a *y_test* z 1D poľa na typ *numpy.array* aby sme vedeli s ňou pracovať v NS.

```
from sklearn.metrics import confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.python.keras.layers.core import Flatten
tf.__version__

dataset = pd.read_csv('input2.csv',delimiter=';',header=None)
X=dataset.iloc[:,:28].values
y=dataset.iloc[:,28].values

X_train, X_test, y_train, y_test = train_test_split(
    X, y,test_size=0.25, random_state=0)

sc=StandardScaler()
X=sc.fit_transform(X)
y_train = np.array(y_train).astype('float32').reshape((-1, 1))
y_test = np.array(y_test).astype('float32').reshape((-1, 1))
```

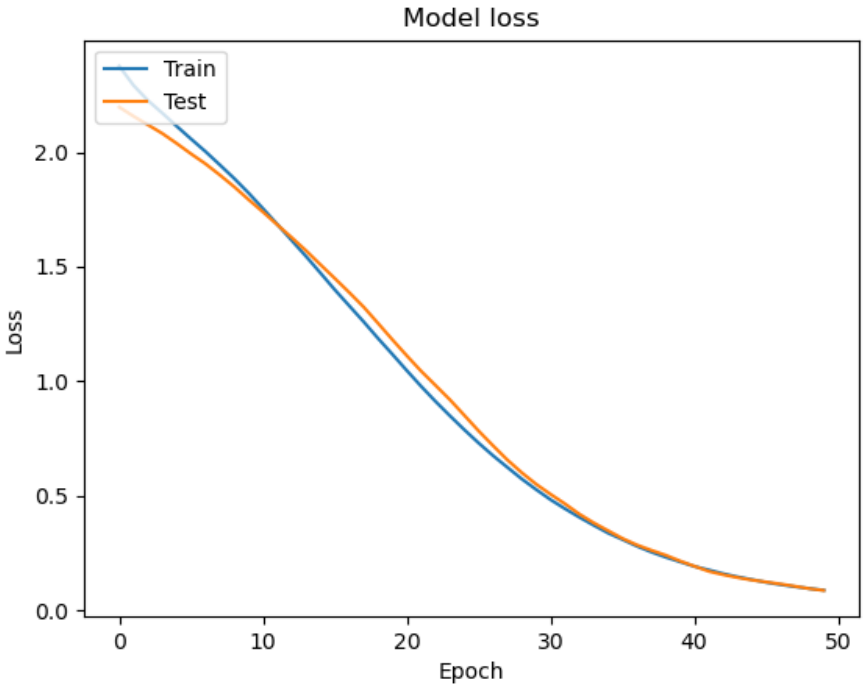
V druhej časti máme vytvorenie NS pod menom *ann*. Počet vstupných neurónov je 28 keďže máme 28 stĺpcov na vstupe. Prvá vrstva obsahuje 32 neurónov, prvá skrytá vrstva má 48 neurónov a na výstupe máme 10 neurónov, označujúcich čísla od 0 po 9. Potom sa NS kompiluje a natrénuje pomocou zvoleného algoritmu učenia, stratovej funkcie, veľkosťou dávky a poctu epochou.

```
ann = tf.keras.models.Sequential()

ann.add(tf.keras.layers.Dense(units=32, activation='relu', input_dim=28))

ann.add(tf.keras.layers.Dense(units=48, activation='relu'))

#ann.add(tf.keras.layers.Dense(units=32, activation='relu'))

ann.add(tf.keras.layers.Dense(units=10,activation='softmax'))

ann.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

ann1 = ann.fit(X_train, y_train, validation_data=(X_test, y_test),
epochs=50,verbose=1)
```

Tretia časť kódu sa venuje vykresleniu grafov, ako sa vytrénovala naša NS. Kód je taký istý ako na cvičeniach.



Ako vidíme na prvom grafe, NS získala accuracy 100 percent už v 24. epochu, ale trénovanie sme nezastavili keďže náš loss sa ešte stále znižoval ako vidíme na nasledujúcom grafe.



Štvrtá a posledná časť kódu sa nám venuje, že ako dobre vie naša NS predikovať vstupy s čistými číslami, porušenými alebo zašumenými. Takisto ako na začiatku sa načítajú údaje z .csv súboru, s rozdielom, že sa posledný stĺpec, ktorý označuje že aké číslo označujú dáta v danom riadku, sa bude používať len na výpis do terminálu, že či NS uhádlo číslo správne. Prvých 10 riadkov v súbore sú čisté čísla od 0 po 9.

Ďalej vidíme jednu malú funkciu, ktorá nám vráti array naplnení s číslom 0. Toto budeme používať ako ukážku výstupu ako to mame definované v zadaní. Rozdielom že prvý prvok v poli označuje nulu, a nie posledný prvok ako to mame v zadaní.

V ďalšom odseku sa nám vypíše *presnosť* a *strata* nasej NS.

Na konci kódu sa začne predikcia dát. Vo for cykle si najprv dáme jednotku na index najväčšej presnosti ktorú nám vypľula NS, potom si presnosť prekonvertujeme z Numpyjovský float32 na Pythonovský float a nakoniec vypíšeme jednotlivé predikcie s ich presnosťou, vektorovou formou a aké číslo to bolo naozaj.

```
datasetPredict = pd.read_csv('inputPredict.csv', delimiter=';', header=None)
X_p = datasetPredict.iloc[:, :28].values
y_p = datasetPredict.iloc[:, 28].values
```

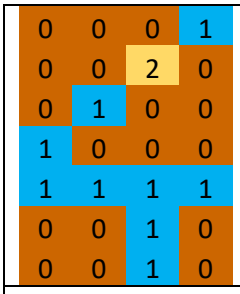
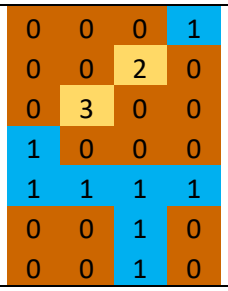
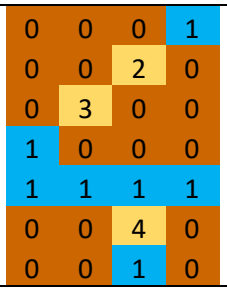
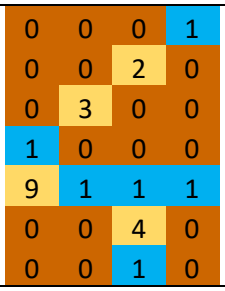
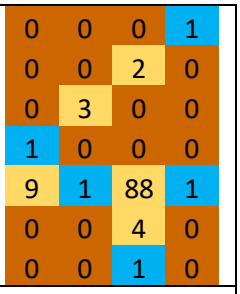
```
def resultVectorNulling():
    result = []
    for i in range(10):
        result.append(0)
    return result

loss,accuracy = ann.evaluate(X_test,y_test)
print(f'Loss is: {loss}')
print(f'Accuracy is: {accuracy}')

result = resultVectorNulling()
predict = ann.predict(X_p)

for i in range(0, len(predict)):
    result[np.argmax(predict[i])] = 1
    val = np.float32(predict[i][np.argmax(predict[i])])
    accuracy_number = val.item()
    if(i ==10):
        print('Zasumenie: ')
    elif(i ==15):
        print('Porusenie: ')
    print('The result is probably: {} with accuracy: {:.2f} in vector form: {}
and should be {} '.format(
        np.argmax(predict[i]), accuracy_number, np.array(result), y_p[i]))
    result=resultVectorNulling()
```

Rozpoznávanie porušených čísiel

Na túto úlohu bola použitá NS ktorú sme si práve vysvetlili, s vrstvami neurónov: 28,32,48,10 a aktivačnými funkciami v poradí relu,relu a softmax. S takimito parametrami vie NS uhádnuť čisté čísla s minimálne 70 percent presnosťou. Na otestovanie robustnosti siete pre zašumenie a porušenie číslic sme použili číslo 4.

Zašumenie

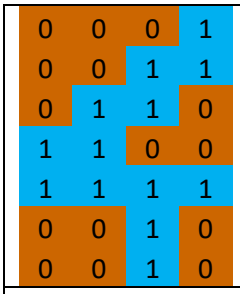
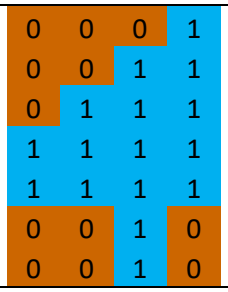
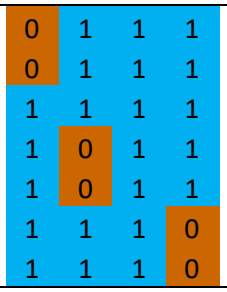
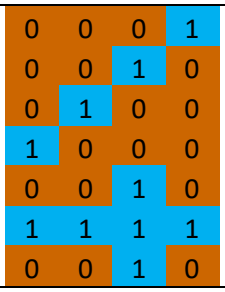
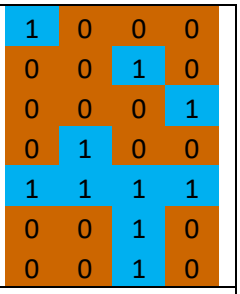
V nasledujúcich tabuľkách uvidíme ako uhádla naša NS číslo s akou presnosťou.

				
65%	66%	70%	17%	0%

Ako vidíme v tabuľke prvé tri zašumenia s najväčšou hodnotou 4, NS ešte stále vie uhádnuť že aké je to číslo, ale keď už tam dáme väčšie čísla ako 9 alebo 88 tak to naruší už rozpoznávanie. Pri predposlednom zašumení nám NS povedala že je to číslo 8 na 51% a pri poslednom zašumení že sa jedna o číslo 1 na 100%.

Porušenie

NS je trosku odolnejšia pri porušení číslic ako aj uvidíme v nasledujúcej tabuľke.

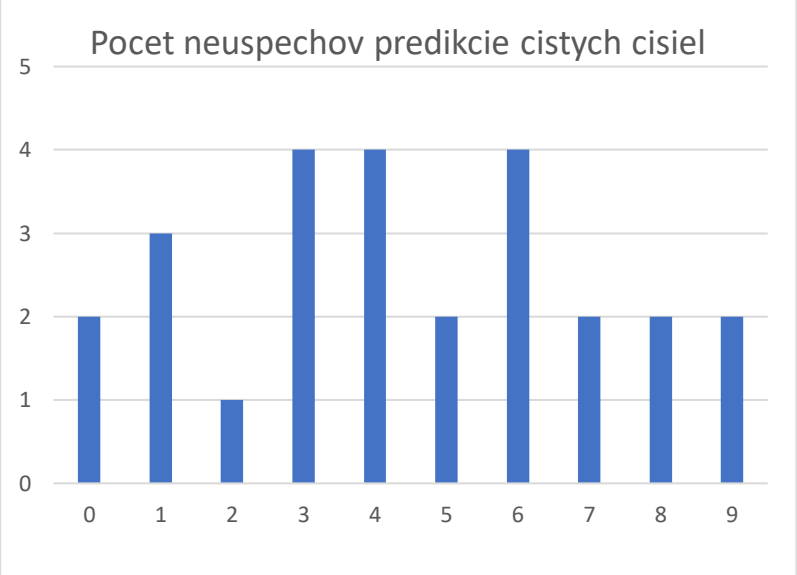
				
70%	49%	13%	21%	32%

Pri prvých troch porušení sme pridávali ďalšie jednotky do vstupu a NS aj tak vedela uhádnuť pri prvých dvoch prípadoch o aké číslo sa jedna, pričom aj s ľudským okom to vieme rozpoznať že je to číslo 4. Pri tretom prípade už ani mi neviem čo je to za číslo a ani NS to nevedelo rozpoznať na 4-ku. Posledné dva porušenia boli že sme si vymenili riadky alebo stĺpce. V našom prípade NS vedela rozpoznať obidve čísla že je to 4.

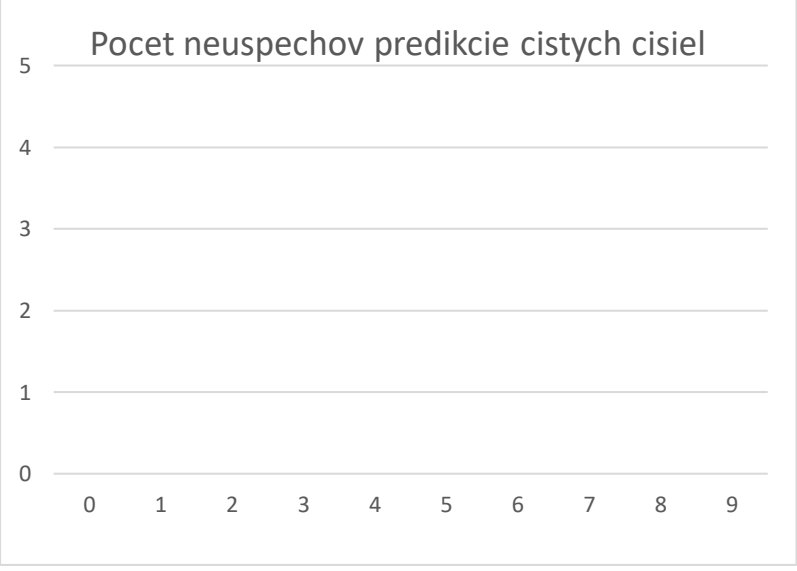
Pri viacero spúšťaní trénovania a predikcie sme si všimli že naše údaje sa veľmi nemenia. NS v drvivej väčšine prípadoch nerozpoznala veľmi zašumene a porušene čísla.

Úspešnosť NS s rôznym počtom vrstiev, neurónov a aktivačnými funkciami

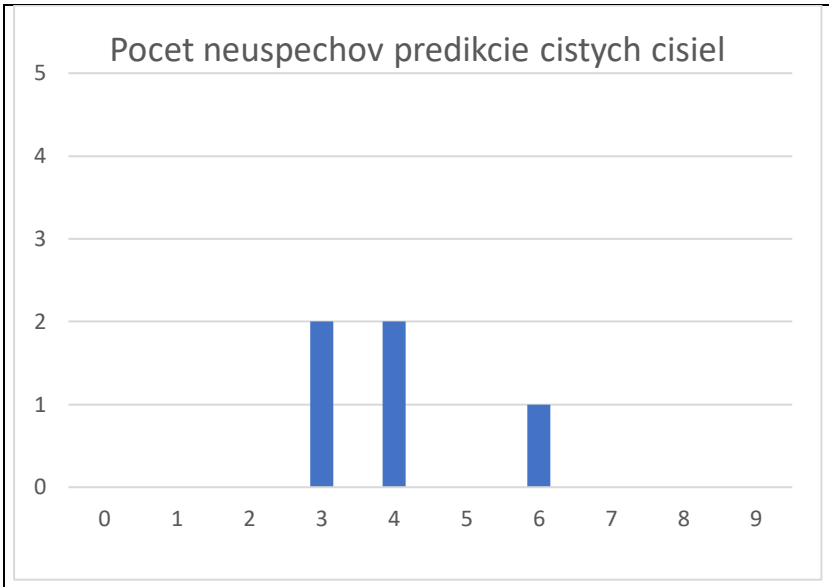
Vyskúšali sme rôzne aktivačne funkcie, počet neurónov a skrytých vrstiev. Nikde sme nemenili počet epochou, čo nám zostalo na 50. Vyskúšali sme takto 6 rôznych NS, pričom sme všetky znovu natrénovali 5-krat aby sme videli rozdiely medzi trénovaním a predikciou. Výsledky sú uvedene v grafoch a tabuľkách nižšie.



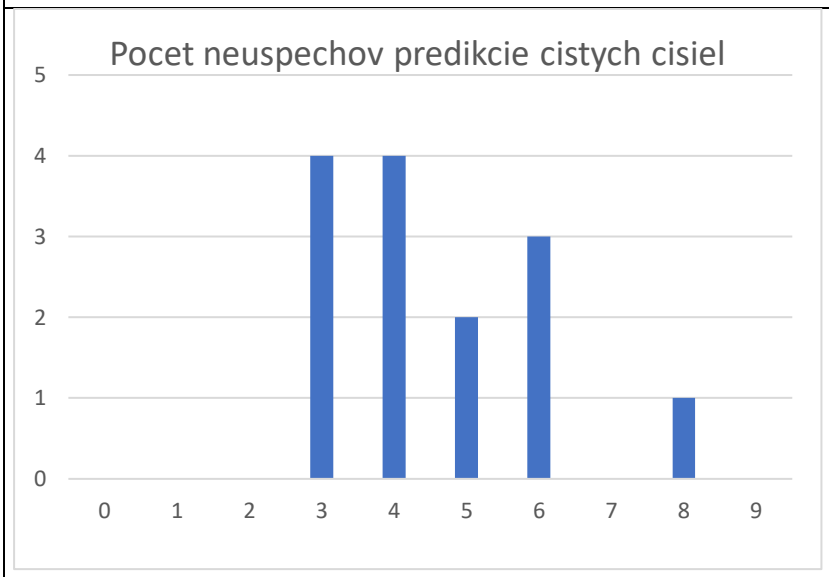
Vrstva	neurón	Funkcia
1	9	Relu
2	6	Tanh
3	15	Relu
4	10	softmax



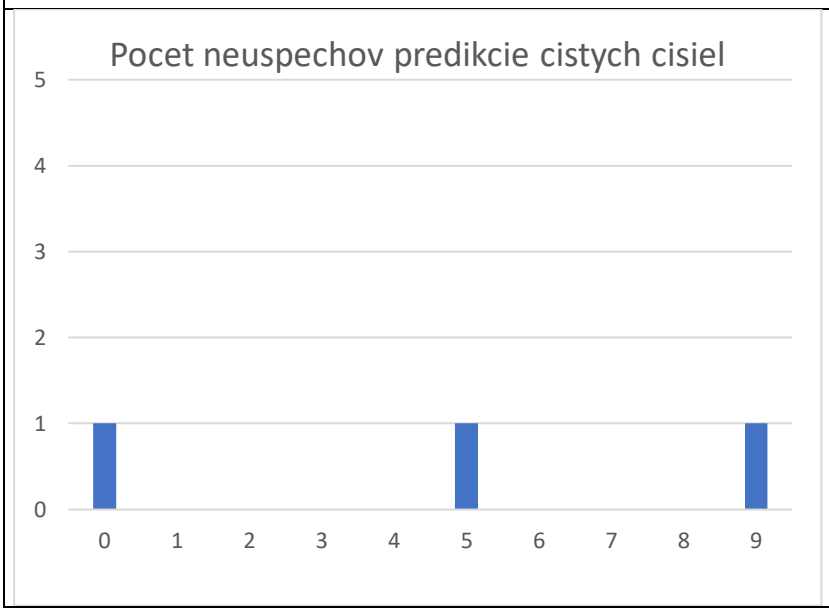
Vrstva	neurón	Funkcia
1	22	Tanh
2	42	Tanh
3	10	softmax



Vrstva	neurón	Funkcia
1	12	Tanh
2	14	Tanh
3	10	Relu
4	10	softmax



Vrstva	neurón	Funkcia
1	12	Tanh
2	14	Relu
3	10	Tanh
4	18	Relu
5	10	softmax



Vrstva	neurón	Funkcia
1	22	Tanh
2	18	Tanh
3	10	Tanh
4	15	Relu
5	25	Relu
6	10	softmax

Ako vidíme, v tabuliek keď použijeme viac skrytých vrstiev s viacerými neurónmi, NS ma oveľa menší omyl. Ale keď zvýšime počet neurónov, tak aj 1 skrytá vrstva nám postačí aby sa každé jedno čisté číslo uhádlo dobre ako aj vidíme v druhej tabuľke.

Záver

Celé zadanie bolo vypracované v Pythone 3.8.8. Na trénovanie bolo použité len čisté čísla keďže nebolo špecifikované že musíme trénovať aj s zašumenými číslami, ale v praxi by sa to tak robilo. Vytvorených bolo 6 rôznych NS, kde sme iba od komentovali riadky kódu a prepisovali aktivačne funkcie a počet neurónov v kóde. Najlepšiu NS ktorú sme našli s počtom neurónov, ktorú sme aj opisovali v kóde na začiatku, nám stále uhádla všetky čisté čísla pri predikcii. Pri testovaní ci sieť rozozná zašumené alebo prerušene dáta sme sa dozvedeli že až kým ani človek nevie rozoznať číslo, tak ani neurónová sieť nevie.