SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Gabrijel Kovač

ISPITIVANJE MOĆI KONVOLUCIJSKIH NEURONSKIH MREŽA ZA KLASIFIKACIJU SLIKA U DUBOKOM UČENJU

PROJEKT

UVOD U UMJETNU INTELIGENCIJU

Varaždin, 2025.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Gabrijel Kovač

Matični broj: 0016156274

Studij: Informacijski i poslovni sustavi

ISPITIVANJE MOĆI KONVOLUCIJSKIH NEURONSKIH MREŽA ZA KLASIFIKACIJU SLIKA U DUBOKOM UČENJU

PROJEKT

Mentor:

izv. prof. dr. sc. Dijana Oreški

Varaždin, siječanj 2025.

Gabrijel Kovač

Izjava o izvornosti

Izjavljujem da je ovaj projekt izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI Radovi

Sažetak

Ovaj rad bavi se ispitivanjem moći konvolucijskih neuronskih mreža (CNN) za klasifikaciju slika, koristeći MNIST dataset kao osnovu za analizu. CNN modeli predstavljaju ključni alat u dubokom učenju, poznati po svojoj sposobnosti automatskog učenja značajki iz vizualnih podataka. Teorijski dio rada temelji se na analizi arhitekture CNN-a, koja uključuje konvolucijske slojeve, pooling slojeve i potpuno povezane slojeve. Rad pokazuje da implementirani CNN model postiže visoku točnost u klasifikaciji slika, čime se potvrđuje učinkovitost ovih mreža u jednostavnim zadacima klasifikacije. Zaključno, rad naglašava važnost CNN modela u analizi slika, dok MNIST dataset služi za razumijevanje složenijih problema i mogućnosti primjene u stvarnim scenarijima.

Ključne riječi: Konvolucijske neuronske mreže (CNN), klasifikacija slika, MNIST dataset, duboko učenje, konvolucijski slojevi, pooling slojevi, potpuno povezani slojevi, funkcija gubitka

Sadržaj

1. Uvod

Duboko učenje, a posebno konvolucijske neuronske mreže (engl. Convolutional Neural Networks, CNN), posljednjih je godina doživjelo izniman rast popularnosti. Glavna prednost konvolucijskih mreža je u mogućnosti automatskog učenja značajki (features) iz slikovnih podataka, za razliku od tradicionalnih metoda koje su se oslanjale na ručno dizajniranje. Upravo ta sposobnost adaptivnog učenja značajki omogućuje mnogo efikasniju i precizniju klasifikaciju slika, što se pokazalo iznimno korisnim u raznim domenama kao kod medicine (npr. dijagnosticiranje tumora s medicinskih slika) i automatizirane vožnje (prepoznavanje objekata na cesti).

MNIST skup podataka (Modified National Institute of Standards and Technology), koji se sastoji od rukom pisanih znamenki (0–9), najčešće se koristi kao uvodni primjer u područje dubokog učenja i klasifikacije slika. Iako je riječ o relativno jednostavnom problemu, MNIST je postao standard koji omogućuje usporedbu različitih modela dubokog učenja. Zbog svoje jednostavnosti pruža idealno okruženje za uvod u konvolucijske neuronske mreže. U ovom radu cilj je ispitati moć konvolucijskih neuronskih mreža pri klasifikaciji rukom pisanih znamenki.

2. Konvolucijske neuronske mreže

U ovome dijelu ćemo detaljno razraditi teorijsku pozadina konvolucijskih neuronskih mreža i klasifikacije slika u kontekstu dubokoga učenja. Obuhvaćene su osnovne definicije te ključni elementi arhitekture konvolucijskih mreža. Također se objašnjavaju koraci u procesu učenja modela i poboljšanju njegove točnosti na primjere MNIST skupa podataka.

Konvolucijske neuronske mreže (engl. Convolutional Neural Networks, CNN) osmišljene su za obradu podataka organiziranih u obliku slika. Za razliku od klasičnih ANN-a koje očekuju podatke u obliku vektora, konvolucijske mreže koriste operatorski pristup 'konvolucija' za izdvajanje prostora i dubine značajki iz slika, što ih čini posebno pogodnima za zadatke poput klasifikacije, detekcije i segmentacije.

Elementi CNN arhitekture:

1. **Konvolucijski sloj** (engl. Convolutional layer):

   - U ovom sloju mreža uči skup filtera (kernels) koji se pomiču po ulaznim slikama kako bi izdvojili značajke i vrše operaciju konvolucije
   - Ova operacija množi vrijednosti piksela slike s vrijednostima unutar filtera. Rezultat se zbraja a suma predstavlja jednu vrijednost na feature map-i
   - Slojevi konvolucije omogućuju izdvajanje značajki (npr. rubova, kutova, oblika) koje se kombinacijom u dubljim slojevima pretvaraju u složenije karakteristike (npr. dijelovi objekata)

2. **Aktivacijske funkcije** (engl. Activation function):

   - Primjenjuje se na rezultate konvolucije
   - Najčešće korištene funkcije aktivacije u CNN-ima su ReLU (engl. Rectified Linear Unit)
   - o ReLU postavlja sve negativne vrijednosti na nulu, čime se povećava nelinearnost modela

3. **Pooling sloj** (engl. Pooling layer):

   - Smanjuje prostornu dimenziju izlaza iz konvolucijskih slojeva (npr. max pooling, average pooling)
   - Funkcija max pooling uzima maksimalnu vrijednost unutar malog područja, čime se postiže smanjenje broja parametara
   - Smanjenjem dimenzije slike smanjuje se i računalna složenost te rizik od overfitting-a

4. **Potpuno povezani sloj** (engl. Fully Connected layer, FC):

   - nakon nekoliko konvolucijskih i pooling slojeva, izdvojene značajke se spajaju u jedan ili više slojeva koji su povezani
   - o Ti slojevi tipično služe kao klasifikatori koji razdvajaju ulaz u ciljne kategorije (npr. znamenke 0–9)

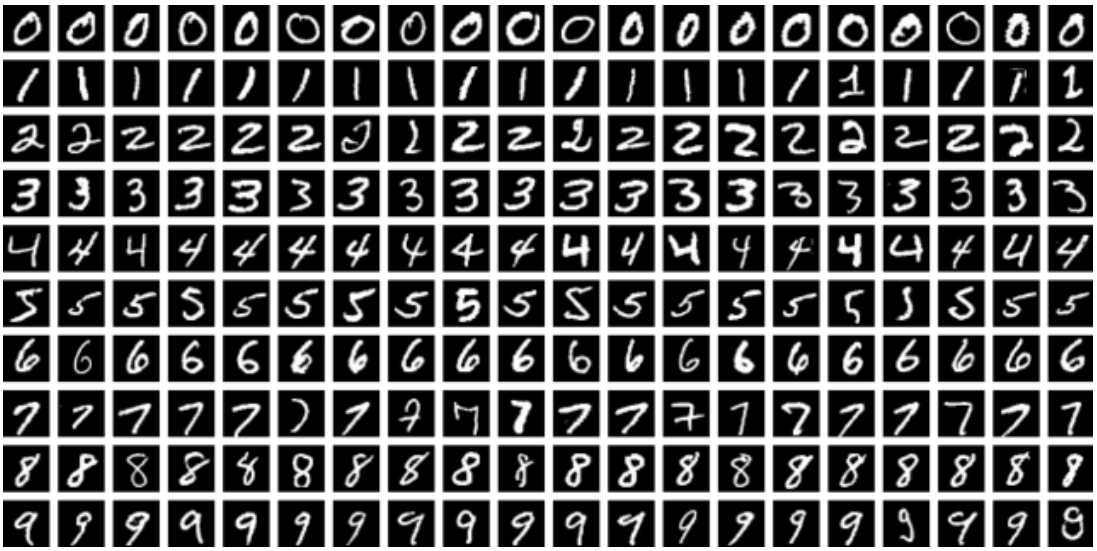
5. **Izlazni sloj:**

   - Za klasifikaciju slika, zadnji sloj obično sadrži onoliko neurona koliko je klasa koje mreža treba predviđati
   - Softmax funkcija često se koristi kako bi se dobile vjerojatnosti pripadanja svakoj klasi

2.1. MNIST dataset

MNIST (Modified National Institute of Standards and Technology) jedan je od najpoznatijih i najčešće korištenih skupova podataka u području strojnog učenja i dubokog učenja, a služi za klasifikaciju rukom pisanih znamenki (0–9).

Skup se sastoji od rukom pisanih znamenki (0–9). Svaka je slika veličine 28×28 piksela, pri čemu su vrijednosti piksela obično sivi tonovi. Ukupan broj slika u skupu je 70.000, od čega je 60.000 namijenjeno za treniranje, a 10.000 za testiranje.



Slika 1: Primjer MNIST dataset-a; preuzeto iz [1]

3. Opis implementacije

3.1. Biblioteke

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
```

Isječak koda 1: Primjer isječka koda

Prvi dio koda uključuje uvoz ključnih biblioteka potrebnih za implementaciju sustava:

- **NumPy:** Koristi se za manipulaciju numeričkim podacima.
- **Matplotlib i Seaborn:** Biblioteke za vizualizaciju podataka i rezultata
- **TensorFlow:** Glavna biblioteka za duboko učenje, koja omogućava konstrukciju i treniranje CNN modela.

3.2. Učitavanje MNIST dataseta

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

Isječak koda 2: Primjer isječka koda

‘tf.keras.datasets.mnist.load_data()‘ učitava MNIST dataset koji sadrži 60,000 slika za treniranje i 10,000 slika za testiranje. Slike su u formatu 28x28 piksela te svaka slika pripada jednoj od 10 kategorija (0-9 brojeva).

‘x_train‘ predstavlja trening skup slika tj. ulazne podatke za treniranje modela.

‘y_train‘ predstavlja skup oznaka tj. pripadajuće kategorije slika.

‘x_test‘ je testni skup slika koji se koristi za evaluaciju modela.

‘y_test‘ je testni skup oznaka.

3.3. Provjera dimenzija podataka

```
print(x_train.shape)
print(x_test.shape)
```

Isječak koda 3: Primjer isječka koda

Shape atribut vraća dimenzije niza te time provjeravamo da li podaci imaju očekivani oblik.

Rezultat:

- Trening skup: (60000, 28, 28) – 60000 slika dimenzija 28x28
- Test skup: (10000, 28, 28) – 10000 slika dimenzija 28x28

3.3.1. Vizualizacija prvih 5 slika

```
for i in range(5):
    plt.imshow(x_train[i], cmap='gray')
    plt.show()
```

Isječak koda 4: Primjer isječka koda

For petlja prolazi kroz prvih 5 slika u skupu za treniranje '(x_train)'. Funkcija 'imshow()' prikazuje 2D sliku kao grafikon. S obzirom da su slike u nijansama sive boje, 'cmap='gray" postavlja prikaz boja slike u sive.

3.3.2. Ispis oznaka za prvih 5 slika

```
for i in range(5):
    print(y_train[i])
```

Isječak koda 5: Primjer isječka koda

Nakon što smo vizualno prikazali prvih 5 slika, sada pomoću for petlje ispisujemo oznake za prvih 5 slika '(y_train)' kako bi smo provjerili da li oznake odgovaraju vrijednostima iz prikazanih slika.

3.4. Normalizacija podataka

```
x_train = tf.keras.utils.normalize(x_train, axis=1)
x_test = tf.keras.utils.normalize(x_test, axis=1)
```

Isječak koda 6: Primjer isječka koda

U ovom djelu provodi se normalizacija ulaznih slika kako bi se njihove vrijednosti piksela smanjile s originalnog raspona od 0 do 255 na raspon [0, 1]. Ovo se izvodi pomoću funkcije tf.keras.utils.normalize, gdje se specificira da se normalizacija vrši po svakom redu slike (axis=1). Normalizacija poboljšava stabilnost i brzinu treniranja neuronske mreže.

3.4.1. Reshape podataka

Nakon normalizacije, slike se prilagođavaju obliku koji je kompatibilan s konvolucijskim neuronskim mrežama. Korištenjem funkcije reshape podaci iz trening skupa '(x_train)' i testnog skupa '(x_test)' transformiraju se iz oblika (60000, 28, 28) i (10000, 28, 28) u (60000, 28, 28, 1) i (10000, 28, 28, 1). Ova dodatna dimenzija označava da se radi o slikama u sivim tonovima, odnosno slikama s jednim kanalom.

```
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
```

Isječak koda 7: Primjer isječka koda

3.5. Arhitektura CNN-a

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

Isječak koda 8: Primjer isječka koda

Ovaj dio koda definira arhitekturu konvolucijske neuronske mreže (CNN) pomoću TensorFlow i Keras biblioteka. Prvo se uvoze potrebni moduli, uključujući Sequential za izgradnju modela kao linearnog niza slojeva te slojevi poput Conv2D, MaxPooling2D, Dense i Flatten. Model se inicijalizira pomoću Sequential() klase, čime se kreira prazan model u koji će se redom dodavati slojevi. Prvi sloj koji se dodaje je konvolucijski sloj (Conv2D), koji koristi 32 filtera veličine 3x3, ReLU aktivacijsku funkciju i oblik ulaznih podataka (28, 28, 1), gdje su 28x28 dimenzije slike, a 1 označava jedan kanal (sivi tonovi). Nakon konvolucijskog sloja dodaje se pooling sloj (MaxPooling2D), koji smanjuje dimenzionalnost izlaza za faktor 2 pomoću maks-pooling operacije. Zatim se dodaje drugi par konvolucijskog i pooling sloja. Ovaj konvolucijski sloj ima 64 filtera veličine 3x3 i koristi istu ReLU aktivacijsku funkciju za nelinearnost. Pooling sloj ponovo smanjuje dimenzionalnost izlaza za faktor 2. Nakon konvolucijskih i pooling slojeva, višedimenzionalni izlaz se pretvara u jednodimenzionalni niz pomoću sloja Flatten. Ovaj niz postaje ulaz za potpuno povezane slojeve (Dense). Prvi potpuno povezani sloj ima 128 neurona i koristi ReLU aktivaciju, dok je posljednji sloj izlazni sloj s 10 neurona (za 10 klasa MNIST dataseta) i softmax aktivacijom. Funkcija softmax osigurava da izlaz predstavlja vjerojatnost za svaku klasu.

3.6. Kompilacija modela

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
↪  metrics=['accuracy'])
```

Isječak koda 9: Primjer isječka koda

U ovom djelu koda konfiguriramo model kako bi bio spreman za proces treniranja, definirajući tri komponente: optimizer, funkciju gubitka i metrike koje će se pratiti tijekom treniranja. Parametar optimizer='adam' postavlja Adam algoritam kao optimizator. Adam je popularan algoritam za optimizaciju koji prilagođava stope učenja za svaku težinu u modelu. Parametar 'loss='sparse_categorical_crossentropy" specificira funkciju gubitka koja se koristi za treniranje modela. Ova funkcija se koristi za klasifikacijske zadatke s više klasa, gdje su oznake dani kao cijeli brojevi. Izračunava pogrešku između stvarnih oznaka i predviđanja modela, uzimajući u obzir vjerojatnosti predviđene za svaku klasu. Parametar metrics=['accuracy'] određuje metrike koje će se koristiti za praćenje performansi modela tijekom treniranja i evaluacije. Točnost (accuracy) mjeri postotak ispravnih predikcija.

3.7. Treniranje modela

```
1 train = model.fit(x_train, y_train, epochs=3)
```

Isječak koda 10: Primjer isječka koda

U ovom djelu koda pokrećemo proces treniranja modela pomoću funkcije fit. Skup podataka za treniranje '(x_train)' i pripadajuće oznake '(y_train)' prolaze kroz model u tri epohe (epochs=3), što znači da se podaci tri puta obrađuju. Tijekom svakog prolaza, model izračunava gubitak na temelju svojih predikcija i stvarnih oznaka. Rezultati treniranja, uključujući gubitak i točnost za svaku epohu, pohranjuju se u varijablu train.

3.8. Evaluacija modela

```
1 loss, accurancy = model.evaluate(x_test, y_test)
2 print(loss)
3 print(accurancy)
```

Isječak koda 11: Primjer isječka koda

U ovom djelu koda evaluiramo model na testnom skupu podataka (x_test i y_test) kako bi se procijenile njegove performanse nakon treniranja. Funkcija model.evaluate() izračunava gubitak (loss) i točnost (accuracy) na temelju podataka koje model nije prethodno vidio.

3.9. Predikcija modela

U ovom djelu koda koristimo testni skup (x_test) za generiranje predikcija modela pomoću funkcije model.predict(). Najvjerojatnija klasa za svaki uzorak određuje se pomoću np.argmax(), a stvarne oznake (y_test) uspoređuju se s predikcijama kako bi se izradila 'Confusion matrix' (tf.math.confusion_matrix). Matrica se zatim vizualizira kao heatmap te nam omogućava prikaz točnih i pogrešnih predikcija modela za svaku klasu.

```
prediction = model.predict(x_test)
prediction_label = np.argmax(prediction, axis=1)

matrix = tf.math.confusion_matrix(y_test, prediction_label)

plt.figure(figsize=(10,8))
sns.heatmap(matrix, annot=True, fmt="d")
plt.xlabel("Predicted number")
plt.ylabel("Number")
plt.title("Confusion Matrix")
plt.show()
```

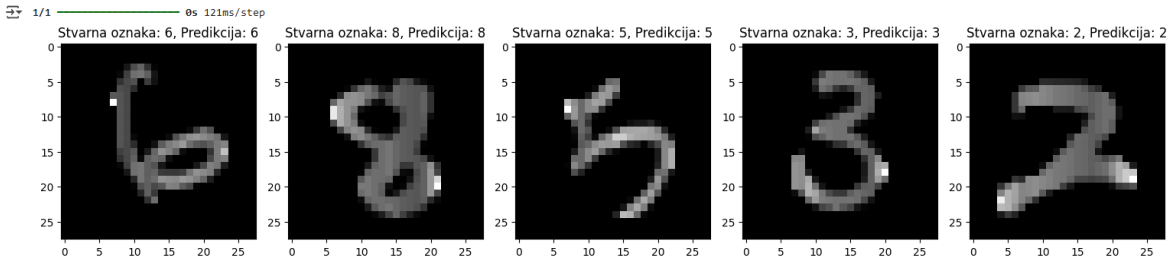
Isječak koda 12: Primjer isječka koda

4. Prikaz rada aplikacije

```
import random

random_index = np.random.choice(len(x_test), 5)

image = x_test[random_index]
label = y_test[random_index]

prediction = model.predict(image)
prediction_label = np.argmax(prediction, axis=1)

plt.figure(figsize=(15, 3))

for i in range(5):
    plt.subplot(1, 5, i+1)
    plt.imshow(image[i].squeeze(), cmap='gray')
    plt.title(f"Stvarna oznaka: {label[i]}, Predikcija: {prediction_label[i]}")

plt.tight_layout()
plt.show()
```

Isječak koda 13: Primjer isječka koda

U ovom djelu koda nasumično odabiremo 5 slika iz testnog skupa te ih prikazujemo zajedno s njihovim stvarnim oznakama i predikcijama modela. Prvo se generiraju indeksi nasumičnih uzoraka pomoću np.random.choice, a zatim se dohvaćaju slike (x_test) i pripadajuće oznake (y_test). Model predviđa klase za odabrane slike pomoću model.predict, a najvjerojatnije klase određuju se funkcijom np.argmax. Svaka slika prikazuje se u sivoj skali, dok se u naslovu za svaku sliku prikazuju informacije o stvarnoj oznaci i predikciji.



Slika 2: Primjer schreenshot koda

5. Kritički osvrt programskog rješenja

Programsko rješenje implementirano za temu "Ispitivanje moći konvolucijskih neuronskih mreža za klasifikaciju slika" pokazuje visoku praktičnost i jasnoću u implementaciji zadatka. Koristeći MNIST dataset i biblioteke poput TensorFlow-a i Keras-a, rješenje opisuje ključne aspekte razvoja i treniranja konvolucijske neuronske mreže (CNN).

Odabrana arhitektura (Conv2D, MaxPooling2D, Dense, itd.) su optimalni za rješavanje problema klasifikacije rukom pisanih brojeva. Proces treniranja, evaluacije i vizualizacije rezultata je jasno strukturiran, a funkcionalnosti poput konfuzijske matrice i prikaza predikcija dodatno doprinose razumijevanju performansi modela.

Programsko rješenje ima široku primjenu u područjima gdje je potrebno prepoznavanje i klasifikacija rukom pisanih brojeva. Konkretan primjer primjene je automatizacija prepoznavanja poštanskih brojeva na paketima u logističkim sustavima. Model može prepoznati rukom napisane poštanske brojeve, smanjujući potrebu za ručnim unosom, ubrzavajući obradu i smanjujući broj pogrešaka.

6. Zaključak

Konvolucijske neuronske mreže (CNN) predstavljaju snažan alat za klasifikaciju i prepoznavanje slika. Kroz implementaciju CNN modela jasno je demonstrirano kako ove mreže mogu automatski identificirati ključne značajke iz ulaznih podataka, poput rubova, oblika i tekstura, što rezultira visokom točnošću u klasifikacijskim zadacima. Rješenja temeljena na CNN-u imaju široku primjenu u različitim domenama. Na primjer, u prepoznavanju rukopisa, koriste se za digitalizaciju dokumenata, automatsko čitanje poštanskih brojeva ili popunjavanje obrazaca. U medicinskoj analizi, CNN-ovi pomažu u otkrivanju bolesti, poput dijagnostike tumora. U autonomnim sustavima, poput autonomnih vozila, koriste se za prepoznavanje prometnih znakova, pješaka i drugih objekata u okruženju.

Iako implementacija na MNIST datasetu služi kao početna točka za razumijevanje principa dubokog učenja, ovo rješenje pruža temelj za rješavanje složenijih problema. Model razvijen za MNIST može se prilagoditi za rad s većim i složenijim datasetima.

Popis literature

[1] Wikipedia contributors. „MNIST database.” [Online; pristupljeno: 30. prosinca 2024.], Wikipedia, The Free Encyclopedia. (2024.), adresa: https://en.wikipedia.org/wiki/MNIST_database (pogledano 30. 12. 2024.).

[2] J. Wu, *Introduction to Convolutional Neural Networks*, [Preuzeto: 30. prosinca 2024.], 2017.

[3] M. Dabović. „Duboke konvolucijske neuronske mreže – koncepti i aktuelna istraživanja.” [Online]. Dostupno: https://www.researchgate.net/publication/321709294. [Preuzeto: 2. siječnja 2025.] (12. 2017.), adresa: https://www.researchgate.net/publication/321709294.

[4] Google Colab. „Convolutional Neural Network Implementation.” [Online]. Available: https://colab.research.google.com/drive/1gC72Ww90qmqC8ODmMcP7ShQjQTwdbLsq. [Accessed: Jan. 24, 2025.] (1. 2025.), adresa: https://colab.research.google.com/drive/1gC72Ww90qmqC8ODmMcP7ShQjQTwdbLsq.

Popis slika

Popis isječaka koda