

Fakultet inženjerskih nauka
Univerziteta u Kragujevcu

Tema:

*Primena konvolucione neuronske mreže AlexNet na skupu
podataka CIFAR-10*

student:
Nikola Mitrevski
400/2021

predmetni profesor:
dr Vesna Ranković
predmetni asistent:
Tijana Šušteršić

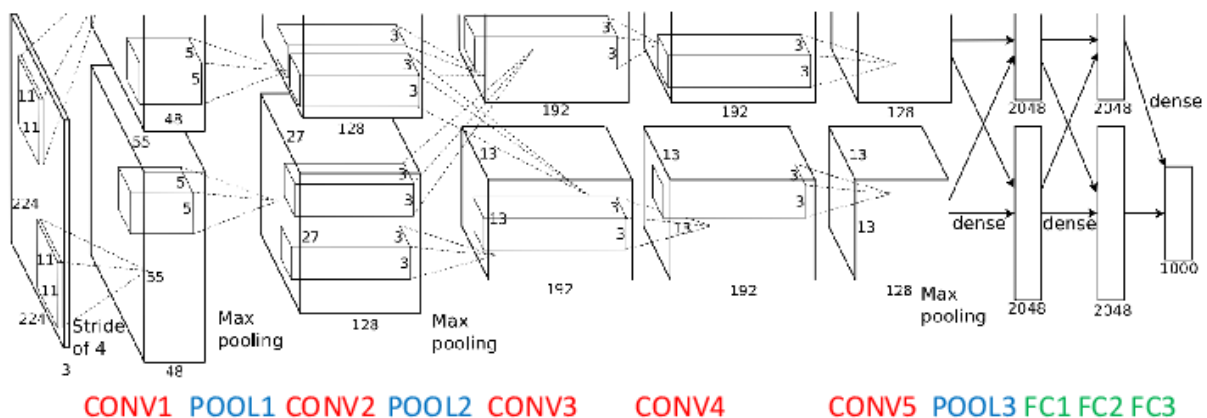
Kragujevac 2022.

Sadržaj:

1	Uvod	2
2	Skup podataka.....	3
2.1	Opis skupa podataka	3
2.2	Učitavanje skupa podataka i njegova transformacija	3
2.3	Pretprocesiranje, mešanje i grupisanje podataka	4
3	Implementacija AlexNet modela	5
4	Kompilacija AlexNet modela	7
5	Treniranje AlexNet modela	9
6	Vizuelizacija treniranja AlexNet modela kroz različite epohe.....	12
6.1	Vizuelizacija gubitaka AlexNet modela.....	12
6.2	Vizuelizacija tačnosti AlexNet modela.....	13
7	Ocena tačnosti AlexNet modela.....	14
8	Zaključak.....	15
9	Literatura	16

1 Uvod

AlexNet arhitektura (slika 1) je 2012 godine pobedila na izazovu klasifikacije velikog broja slika (Imagnet skup podataka – skup podataka sa skoro 14 miliona slika u hiljadu klasa) s tačnošću klasifikacije od 84.7%. Ona je predložena 2012 godine u istraživačkom radu pod nazivom “Imagenet Classification with Deep Convolution Neural Network” od strane Alex Krizhevsky i njegovih kolega. Ova arhitektura ima osam slojeva sa parametrima koji se mogu naučiti od kojih su pet slojeva konvolucije sa kombinacijom sažimanja po maksimumu i tri potpuno povezana sloja. U svim ovim slojevima se koriste ReLU aktivacione funkcije osim u izlaznom sloju. Alex Krizhevsky i njegove kolega u istraživačkom radu navode da su otkrili da korišćenje ReLU aktivacionih funkcija ubrzava proces treninga za skoro šest puta. Takođe ova arhitektura se sastoji od dva stabla slojeva, jer je obučavana paralelno na dva grafička procesora. [1]

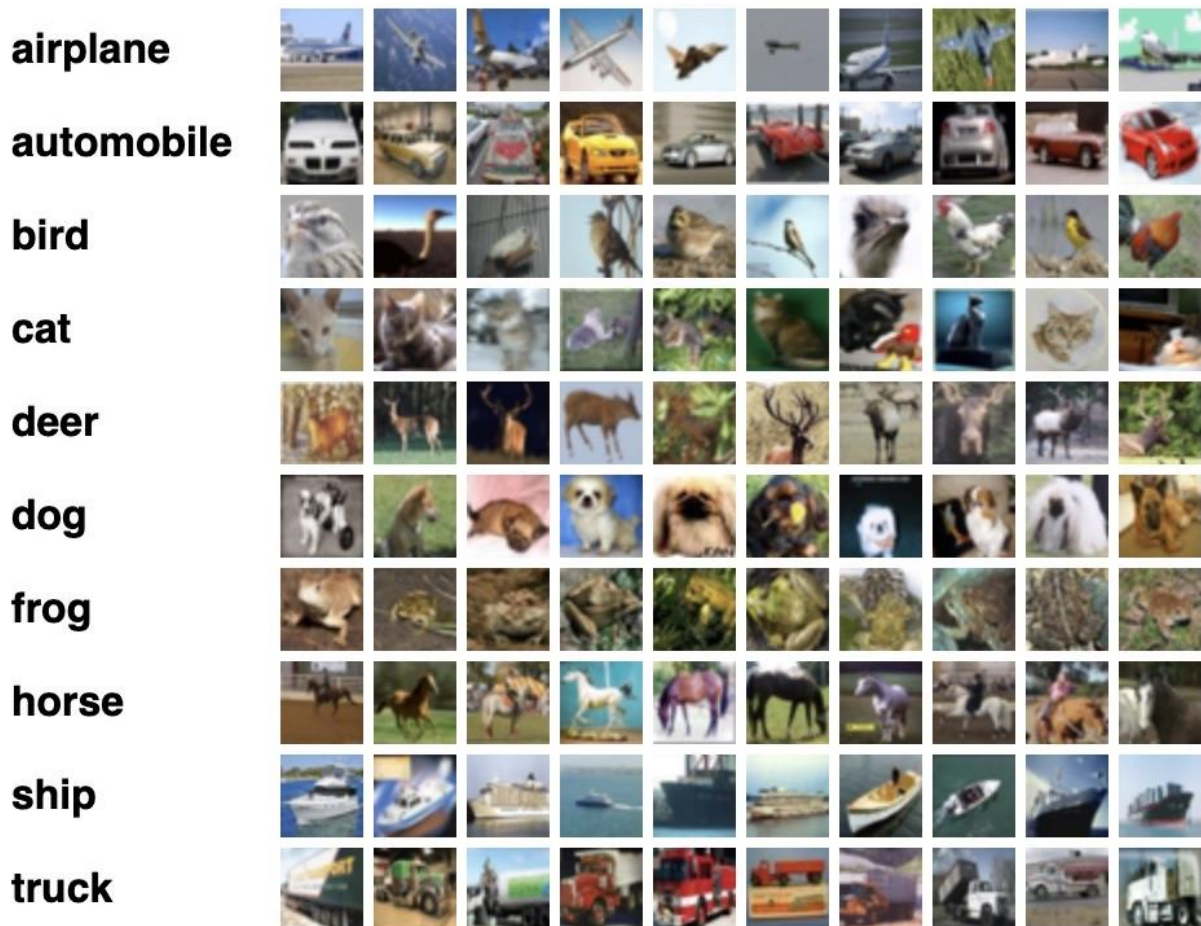


Slika 1 Prikaz AlexNet arhitekture

2 Skup podataka

2.1 Opis skupa podataka

CIFAR-10 (Canadian Institute for Advanced Research, 10 classes) skup podata (slika 2) je skup podataka koji se sastoji od 60000 slika. Svaka slika u ovom skupu podataka je predstavljena veličinom od 32x32 piksela (širina x visina) i dubinom kanala veličine 3. [2]



Slika 2 Prikaz CIFAR-10 skupa podataka

2.2 Učitavanje skupa podataka i njegova transformacija

Za učitavanje CIFAR-10 skupa podataka koristi se modul `tensorflow.keras.datasets`.

Sledeća linija koda koristi se za učitavanje trening i test skupa podataka:

```
(train_images, train_labels), (test_images, test_labels) =  
keras.datasets.cifar10.load_data()
```

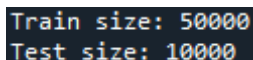
Nakon učitavanja trening i test skupa podataka, potrebno je uraditi odgovarajuću transformaciju podataka da bi mogli da se koriste sa TensorFlow-om. TensorFlow

obezbeđuje skup funkcija i operacija koje omogućavaju lakšu manipulaciju i modifikaciju podataka.

Sledeće linije koda vrše opisanu transformaciju:

```
train_ds = tf.data.Dataset.from_tensor_slices((train_images,train_labels))
test_ds = tf.data.Dataset.from_tensor_slices((test_images,test_labels))
```

Ukupan broj elemenata u trening i test skupu podataka prikazan je na slici 3.



```
Train size: 50000
Test size: 10000
```

Slika 3 Ukupan broj elemenata u trening i test skupu podataka

2.3 Pretprocesiranje, mešanje i grupisanje podataka

Nad trening i test skupom podataka, sprovedene su tri operacije:

1. pretprocesiranje podataka;
2. mešanje podataka;
3. grupisanje podataka.

Sledeće linije koda sprovode tri navede operacije nad skupovima podataka:

```
train_ds=(train_ds
    .map(process_image)
    .shuffle(buffer_size=train_ds_size)
    .batch(batch_size=32,drop_remainder=True)
)

test_ds=(test_ds
    .map(process_image)
    .shuffle(buffer_size=test_ds_size)
    .batch(batch_size=32,drop_remainder=True)
)
```

Parametar `batch_size` je hiperparametar koji se koristi za grupisanje podataka. Ovaj hiperparametra je jednak broju ulaza koji se propuštaju kroz mrežu, pre nego što se uradi back propagation. Vrednost ovog hiperparametra se kreće u opsegu od 1 do veličine trening/test skupa podataka. Što je manja vrednost ovog hiperparametra, to je veće vreme obuke, ali je zato i tačnost veća. [3]

3 Implementacija AlexNet modela

Sledeće linije koda definišu strukturu AlexNet modela:

```
model = keras.models.Sequential([
    keras.layers.Conv2D(filters=128, kernel_size=(11,11), strides=(4,4),
activation='relu', input_shape=(64,64,3)),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(2,2)),
    keras.layers.Conv2D(filters=256, kernel_size=(5,5), strides=(1,1),
activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(3,3)),
    keras.layers.Conv2D(filters=256, kernel_size=(3,3), strides=(1,1),
activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=256, kernel_size=(1,1), strides=(1,1),
activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=256, kernel_size=(1,1), strides=(1,1),
activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(2,2)),
    keras.layers.Flatten(),
    keras.layers.Dense(1024, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(1024, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(numOfClasses, activation='softmax')
])
```

Conv2D sloj predstavlja sloj konvolucije i njegova namena je da napravi mapu svojstava od učitane slike. [4]

BatchNormalization sloj se koristi za normalizaciju i standardizaciju ulaznih piksela. Normalizacija piksela znači da se vrednost svakog piksela deli brojem 255, tj. vrednost

svakog piksela se svodi s opsega 0-255 na opseg od 0-1. Standardizacija piksela znači da se od trenutne vrednosti piksela oduzima srednja vrednost piksela, a zatim se nova vrednost piksela deli vrednošću standardne devijacije piksela. [5]

MaxPool2D sloj predstavlja sloj sažimanja i njegova namena je da smanji dimenziju mape svojstava. [6]

Flatten sloj se koristi za pretvaranje podataka u jednodimenzionalni niz. [7]

Dense sloj predstavlja potpuno povezan sloj i njegova namena je da iskoristi dobijene podatke kako bi izvršio klasifikaciju. [8]

4 Kompilacija AlexNet modela

Kompilacija modela predstavlja konfigurisanje modela pre njegove obuke.

Sledeće linije koda vrše kompilaciju AlexNet modela:

```
model.compile(  
    loss='sparse_categorical_crossentropy',  
    optimizer=tf.optimizers.SGD(lr=0.001),  
    metrics=['accuracy']  
)
```

Parametar loss predstavlja funkciju gubitaka (troškova). Koristi se za izračunavanje vrednosti poznate kao gubitak. Što je gubitak manji, to su performanse modela bolje. Jedna od najkorišćenijih funkcija gubitaka zove se unakrsna entropija (cross entropy).

Parametar optimizer koristi se za optimizaciju modela. Dva najkorišćenija algoritma za optimizaciju modela su Stochastic Gradient Descent (SGD) optimizator i Adam optimizator. Jedan od parametara koji se najčešće prosleđuje navedenim optimizatorima je hiperparametar koji se zove stopa učenja (learning rate - lr). Ovaj hiperparametar „govori“ koliko mreža brzo uči i od njega zavisi gubitak. Ako je vrednost ovog hiperparametara previsoka, težine veza i pragovi aktivacije će se podešavati na svakom prolazu, dok ako je vrednost preniska, prilagođavanje će biti veoma sporo. Bilo koji uslov od gornja dva može dovesti do toga da mreža nikada ne konvergira. [9]

Neki od načina za pronalaženje najbolje vrednosti za stopu učenja su:

1. Korišćenje konstantne vrednosti za stopu učenja.
2. Smanjivanje vrednosti za stopu učenja tokom vremena. Na primer, za prvih 10 epoha može se koristiti vrednost 0.001 za stopu učenja. U epohi 11 spušta se vrednost za stopu učenja na 0.0001. U epohi 20 pada na 0.00001. Razlog za smanjivanje brzine učenja tokom vremena je taj što prilagođavanja težina postaju manje promenljiva kako mreža uči. To je kao kad se osoba šiša. Frizer prvo odseče dugu kosu. Zatim vrši neka podešavanja i seče kraće dužine. Na kraju pravi vrlo male rezove oko ušiju za finalnu frizuru.
3. Uzimanje visoke vrednosti za stopu učenja od 0.1, a zatim uzimanje niske vrednosti za stopu učenja od $1e-7$. Stope učenja obično variraju između 0.1 i $1e-7$, ali najčešće nisu ove krajnje vrednosti, nego neka između. Zatim se može

pokušati sa vrednostima 0.01 i 1e-6, itd. Na kraju se upoređuju rezultati da bi se pronašla najbolja vrednost za stopu učenja.

Parametar metrics predstavlja funkcije koje se koriste za procenu performansi modela. Ove funkcije su slične funkcijama gubitaka, osim što se rezultati procena ne koriste pri obučavanju modela.

Na slici 4 je prikazana struktura kreiranog AlexNet modela.

Model: "sequential"		
Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 14, 14, 128)	46592
batch_normalization (Batch Normalization)	(None, 14, 14, 128)	512
max_pooling2d (MaxPooling2D)	(None, 7, 7, 128)	0
conv2d_1 (Conv2D)	(None, 7, 7, 256)	819456
batch_normalization_1 (Batch Normalization)	(None, 7, 7, 256)	1024
max_pooling2d_1 (MaxPooling2D)	(None, 2, 2, 256)	0
conv2d_2 (Conv2D)	(None, 2, 2, 256)	590080
batch_normalization_2 (Batch Normalization)	(None, 2, 2, 256)	1024
conv2d_3 (Conv2D)	(None, 2, 2, 256)	65792
batch_normalization_3 (Batch Normalization)	(None, 2, 2, 256)	1024
conv2d_4 (Conv2D)	(None, 2, 2, 256)	65792
batch_normalization_4 (Batch Normalization)	(None, 2, 2, 256)	1024
max_pooling2d_2 (MaxPooling2D)	(None, 1, 1, 256)	0
flatten (Flatten)	(None, 256)	0
dense (Dense)	(None, 1024)	263168
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 1024)	1049600
dropout_1 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 10)	10250
=====		
Total params: 2,915,338		
Trainable params: 2,913,034		
Non-trainable params: 2,304		

Slika 4 Struktura AlexNet modela

5 Treniranje AlexNet modela

Sledeće linije koda se koriste za treniranje AlexNet modela:

```
history=model.fit(
    train_ds,
    epochs=50,
    validation_data=test_ds,
    validation_freq=1
)
```

Parametar epochs je hiperparametar koji „govori“ koliko će se puta proći kroz ceo skup podataka za obuku. Vrednost ovog hiperparametra se kreće u opsegu od 1 do beskonačno.

Na slikama 5, 6, 7, 8 i 9 su prikazane epohe treniranja AlexNet modela.

```
Epoch 1/50
C:\Users\Master4\AppData\Roaming\Python\Python38\site-packages\keras
\optimizer_v2\gradient_descent.py:102: UserWarning: The `lr` argument is deprecated, use
`learning_rate` instead.
  super(SGD, self).__init__(name, **kwargs)
1562/1562 [=====] - 521s 331ms/step - loss: 2.1080 - accuracy: 0.2555 -
val_loss: 1.6129 - val_accuracy: 0.4199
Epoch 2/50
1562/1562 [=====] - 532s 340ms/step - loss: 1.7136 - accuracy: 0.3768 -
val_loss: 1.4365 - val_accuracy: 0.4785
Epoch 3/50
1562/1562 [=====] - 545s 348ms/step - loss: 1.5429 - accuracy: 0.4426 -
val_loss: 1.3307 - val_accuracy: 0.5204
Epoch 4/50
1562/1562 [=====] - 540s 345ms/step - loss: 1.4111 - accuracy: 0.4917 -
val_loss: 1.2465 - val_accuracy: 0.5502
Epoch 5/50
1562/1562 [=====] - 576s 368ms/step - loss: 1.3063 - accuracy: 0.5330 -
val_loss: 1.1693 - val_accuracy: 0.5830
Epoch 6/50
1562/1562 [=====] - 592s 378ms/step - loss: 1.2230 - accuracy: 0.5636 -
val_loss: 1.1192 - val_accuracy: 0.6030
Epoch 7/50
1562/1562 [=====] - 636s 406ms/step - loss: 1.1472 - accuracy: 0.5939 -
val_loss: 1.0788 - val_accuracy: 0.6197
Epoch 8/50
1562/1562 [=====] - 608s 388ms/step - loss: 1.0767 - accuracy: 0.6207 -
val_loss: 1.0606 - val_accuracy: 0.6249
Epoch 9/50
1562/1562 [=====] - 601s 384ms/step - loss: 1.0138 - accuracy: 0.6438 -
val_loss: 1.0041 - val_accuracy: 0.6499
Epoch 10/50
1562/1562 [=====] - 624s 398ms/step - loss: 0.9523 - accuracy: 0.6653 -
val_loss: 0.9849 - val_accuracy: 0.6613
```

Slika 5 Treniranje AlexNet modela, prvih deset epoha

```

Epoch 11/50
1562/1562 [=====] - 627s 400ms/step - loss: 0.9023 - accuracy: 0.6831 -
val_loss: 0.9595 - val_accuracy: 0.6630
Epoch 12/50
1562/1562 [=====] - 624s 399ms/step - loss: 0.8530 - accuracy: 0.7005 -
val_loss: 0.9320 - val_accuracy: 0.6769
Epoch 13/50
1562/1562 [=====] - 609s 389ms/step - loss: 0.7989 - accuracy: 0.7219 -
val_loss: 0.9445 - val_accuracy: 0.6728
Epoch 14/50
1562/1562 [=====] - 624s 399ms/step - loss: 0.7556 - accuracy: 0.7366 -
val_loss: 0.9365 - val_accuracy: 0.6804
Epoch 15/50
1562/1562 [=====] - 633s 404ms/step - loss: 0.7138 - accuracy: 0.7519 -
val_loss: 0.9194 - val_accuracy: 0.6843
Epoch 16/50
1562/1562 [=====] - 632s 404ms/step - loss: 0.6713 - accuracy: 0.7662 -
val_loss: 0.9108 - val_accuracy: 0.6906
Epoch 17/50
1562/1562 [=====] - 642s 410ms/step - loss: 0.6239 - accuracy: 0.7811 -
val_loss: 0.9128 - val_accuracy: 0.6944
Epoch 18/50
1562/1562 [=====] - 647s 413ms/step - loss: 0.5821 - accuracy: 0.7981 -
val_loss: 0.9287 - val_accuracy: 0.6954
Epoch 19/50
1562/1562 [=====] - 625s 400ms/step - loss: 0.5469 - accuracy: 0.8100 -
val_loss: 0.9599 - val_accuracy: 0.6923
Epoch 20/50
1562/1562 [=====] - 609s 389ms/step - loss: 0.5065 - accuracy: 0.8230 -
val_loss: 0.9603 - val_accuracy: 0.6944

```

Slika 6 Treniranje AlexNet modela, drugih deset epoha

```

Epoch 21/50
1562/1562 [=====] - 609s 389ms/step - loss: 0.4634 - accuracy: 0.8385 -
val_loss: 0.9753 - val_accuracy: 0.6957
Epoch 22/50
1562/1562 [=====] - 610s 389ms/step - loss: 0.4444 - accuracy: 0.8459 -
val_loss: 0.9937 - val_accuracy: 0.6948
Epoch 23/50
1562/1562 [=====] - 618s 395ms/step - loss: 0.4068 - accuracy: 0.8568 -
val_loss: 1.0102 - val_accuracy: 0.7071
Epoch 24/50
1562/1562 [=====] - 675s 431ms/step - loss: 0.3742 - accuracy: 0.8699 -
val_loss: 1.0541 - val_accuracy: 0.6948
Epoch 25/50
1562/1562 [=====] - 644s 411ms/step - loss: 0.3500 - accuracy: 0.8781 -
val_loss: 1.0443 - val_accuracy: 0.6967
Epoch 26/50
1562/1562 [=====] - 618s 395ms/step - loss: 0.3244 - accuracy: 0.8876 -
val_loss: 1.1051 - val_accuracy: 0.6906
Epoch 27/50
1562/1562 [=====] - 633s 405ms/step - loss: 0.2971 - accuracy: 0.8968 -
val_loss: 1.0971 - val_accuracy: 0.7048
Epoch 28/50
1562/1562 [=====] - 629s 402ms/step - loss: 0.2796 - accuracy: 0.9023 -
val_loss: 1.1426 - val_accuracy: 0.6936
Epoch 29/50
1562/1562 [=====] - 618s 395ms/step - loss: 0.2520 - accuracy: 0.9119 -
val_loss: 1.2271 - val_accuracy: 0.6920
Epoch 30/50
1562/1562 [=====] - 648s 414ms/step - loss: 0.2351 - accuracy: 0.9185 -
val_loss: 1.1732 - val_accuracy: 0.6973

```

Slika 7 Treniranje AlexNet modela, trećih deset epoha

```

Epoch 31/50
1562/1562 [=====] - 660s 421ms/step - loss: 0.2194 - accuracy: 0.9242 -
val_loss: 1.2610 - val_accuracy: 0.6989
Epoch 32/50
1562/1562 [=====] - 699s 447ms/step - loss: 0.2026 - accuracy: 0.9301 -
val_loss: 1.2285 - val_accuracy: 0.6957
Epoch 33/50
1562/1562 [=====] - 678s 433ms/step - loss: 0.1884 - accuracy: 0.9353 -
val_loss: 1.2732 - val_accuracy: 0.7030
Epoch 34/50
1562/1562 [=====] - 656s 419ms/step - loss: 0.1642 - accuracy: 0.9437 -
val_loss: 1.3039 - val_accuracy: 0.6978
Epoch 35/50
1562/1562 [=====] - 670s 428ms/step - loss: 0.1549 - accuracy: 0.9463 -
val_loss: 1.3419 - val_accuracy: 0.6998
Epoch 36/50
1562/1562 [=====] - 666s 425ms/step - loss: 0.1521 - accuracy: 0.9470 -
val_loss: 1.3319 - val_accuracy: 0.7034
Epoch 37/50
1562/1562 [=====] - 665s 425ms/step - loss: 0.1379 - accuracy: 0.9526 -
val_loss: 1.3765 - val_accuracy: 0.7020
Epoch 38/50
1562/1562 [=====] - 719s 459ms/step - loss: 0.1277 - accuracy: 0.9563 -
val_loss: 1.4309 - val_accuracy: 0.6938
Epoch 39/50
1562/1562 [=====] - 682s 436ms/step - loss: 0.1167 - accuracy: 0.9602 -
val_loss: 1.4352 - val_accuracy: 0.6991
Epoch 40/50
1562/1562 [=====] - 719s 459ms/step - loss: 0.1095 - accuracy: 0.9630 -
val_loss: 1.4664 - val_accuracy: 0.6997

```

Slika 8 Treniranje AlexNet modela, četrtih deset epoha

```

Epoch 41/50
1562/1562 [=====] - 715s 457ms/step - loss: 0.1068 - accuracy: 0.9632 -
val_loss: 1.5276 - val_accuracy: 0.6959
Epoch 42/50
1562/1562 [=====] - 701s 448ms/step - loss: 0.0994 - accuracy: 0.9671 -
val_loss: 1.5019 - val_accuracy: 0.7036
Epoch 43/50
1562/1562 [=====] - 706s 451ms/step - loss: 0.1006 - accuracy: 0.9664 -
val_loss: 1.5043 - val_accuracy: 0.6961
Epoch 44/50
1562/1562 [=====] - 719s 459ms/step - loss: 0.0896 - accuracy: 0.9696 -
val_loss: 1.5413 - val_accuracy: 0.7013
Epoch 45/50
1562/1562 [=====] - 723s 462ms/step - loss: 0.0865 - accuracy: 0.9712 -
val_loss: 1.4982 - val_accuracy: 0.7040
Epoch 46/50
1562/1562 [=====] - 716s 457ms/step - loss: 0.0855 - accuracy: 0.9707 -
val_loss: 1.4952 - val_accuracy: 0.7081
Epoch 47/50
1562/1562 [=====] - 735s 469ms/step - loss: 0.0744 - accuracy: 0.9756 -
val_loss: 1.6095 - val_accuracy: 0.7003
Epoch 48/50
1562/1562 [=====] - 715s 457ms/step - loss: 0.0693 - accuracy: 0.9768 -
val_loss: 1.5391 - val_accuracy: 0.7073
Epoch 49/50
1562/1562 [=====] - 725s 463ms/step - loss: 0.0673 - accuracy: 0.9775 -
val_loss: 1.6173 - val_accuracy: 0.6991
Epoch 50/50
1562/1562 [=====] - 715s 457ms/step - loss: 0.0643 - accuracy: 0.9784 -
val_loss: 1.6034 - val_accuracy: 0.7028

```

Slika 9 Treniranje AlexNet modela, petih deset epoha

6 Vizuelizacija treniranja AlexNet modela kroz različite epohe

Sledeće linije koda vrše vizuelizaciju vrednosti gubitaka i tačnosti AlexNet modela prilikom njegove obuke kroz različite epohe:

```
f,ax=plt.subplots(2,1,figsize=(10,10))

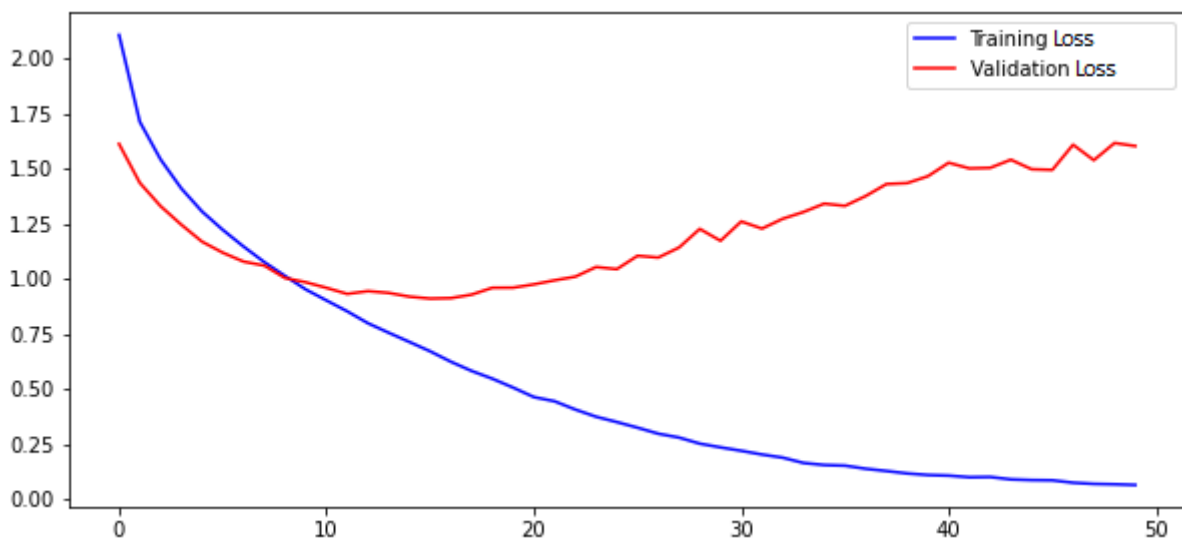
ax[0].plot(model.history.history['loss'],color='b',label='Training Loss')
ax[0].plot(model.history.history['val_loss'],color='r',label='Validation Loss')

ax[1].plot(model.history.history['accuracy'],color='b',label='Training Accuracy')
ax[1].plot(model.history.history['val_accuracy'],color='r',label='Validation Accuracy')

plt.legend()
```

6.1 Vizuelizacija gubitaka AlexNet modela

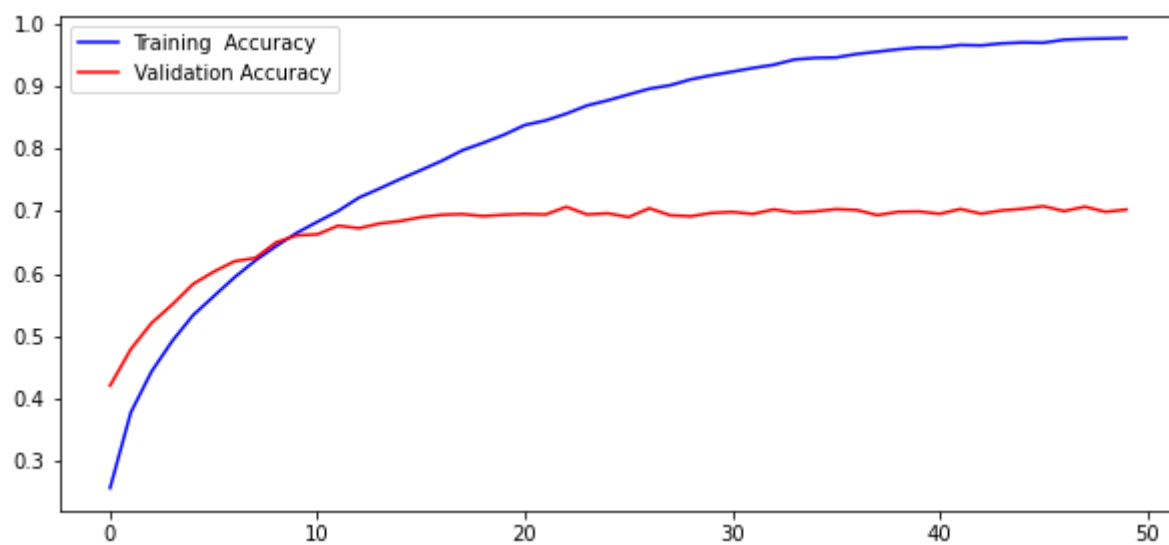
Na slici 10 su vizualizovani gubitci prilikom obuke modela.



Slika 10 Vizuelizacija gubitaka AlexNet modela

6.2 Vizuelizacija tačnosti AlexNet modela

Na slici 11 su vizualizovane tačnosti prilikom obuke modela.



Slika 11 Vizuelizacija tačnosti AlexNet modela

7 Ocena tačnosti AlexNet modela

Ocena tačnosti AlexNet modela prikazana je na slici 12.

A screenshot of a terminal window with a black background and white text. The text reads "Accuracy Score = 0.7081330418586731".

Accuracy Score = 0.7081330418586731

Slika 12 Ocena tačnosti AlexNet modela

8 Zaključak

AlexNet arhitektura ima osam slojeva sa parametrima koji se mogu naučiti od kojih su pet slojeva konvolucije sa kombinacijom sažimanja po maksimumu i tri potpuno povezana sloja. U svim ovim slojevima se koriste ReLU aktivacione funkcije osim u izlaznom sloju.

CIFAR-10 skup podata je skup podataka koji se sastoji od 60000 slika (raspoređene u 10 različitih klasa), gde je svaka slika predstavljena veličinom od 32x32 piksela (širina x visina) i dubinom kanala veličine 3.

Hiperparametri koji su podešavani da bi se dobile što bolje performanse AlexNet modela su:

1. `batch_size` – koristi se za grupisanje podataka unutar trening i test skupova;
2. `lr` (learning rate) – koristi se za podešavanje brzine učenja modela;
3. `epochs` – koristi se da se naznači koliko će se puta proći kroz ceo skup podataka za obuku.

Prosečna ocena tačnosti AlexNet modela sa ovako podešenim hiperparametrima: `batch_size=32`, `lr=0.001`, `epochs=50` iznosi 0.7081330418586731.

9 Literatura

- [1] „Introduction to The Architecture of Alexnet,“ [Na mreži]. Available: <https://www.analyticsvidhya.com/blog/2021/03/introduction-to-the-architecture-of-alexnet/>. [Poslednji pristup 03 04 2022].
- [2] „CIFAR10 small images classification dataset,“ Keras, [Na mreži]. Available: <https://keras.io/api/datasets/cifar10/>. [Poslednji pristup 03 04 2022].
- [3] „Difference Between a Batch and an Epoch in a Neural Network,“ Machine Learning Mastery, [Na mreži]. Available: <https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/>. [Poslednji pristup 03 04 2022].
- [4] „Conv2D layer,“ Keras, [Na mreži]. Available: https://keras.io/api/layers/convolution_layers/convolution2d/. [Poslednji pristup 03 04 2022].
- [5] „BatchNormalization layer,“ Keras, [Na mreži]. Available: https://keras.io/api/layers/normalization_layers/batch_normalization/. [Poslednji pristup 03 04 2022].
- [6] „MaxPooling2D layer,“ Keras, [Na mreži]. Available: https://keras.io/api/layers/pooling_layers/max_pooling2d/. [Poslednji pristup 03 04 2022].
- [7] „Flatten layer,“ Keras, [Na mreži]. Available: https://keras.io/api/layers/reshaping_layers/flatten/. [Poslednji pristup 03 04 2022].
- [8] „Dense layer,“ Keras, [Na mreži]. Available: https://keras.io/api/layers/core_layers/dense/. [Poslednji pristup 03 04 2022].
- [9] „Hyperparameters for Classifying Images with Convolutional Neural Networks – Part 1 – Learning Rate,“ MARK III SYSTEMS, [Na mreži]. Available: <https://www.markiiisys.com/blog/hyperparameters-for-classifying-images-with->

[convolutional-neural-networks-part-1-learning-rate/](#). [Poslednji pristup 03 04 2022].