Факултет инжењерских наука Универзитета у Крагујевцу

Анализа и пројектовање алгоритама

Студент:
Лазар Спасенић 431/2021

Предметни наставник:
Проф. Владимир Миловановић

Крагујевац, 2022

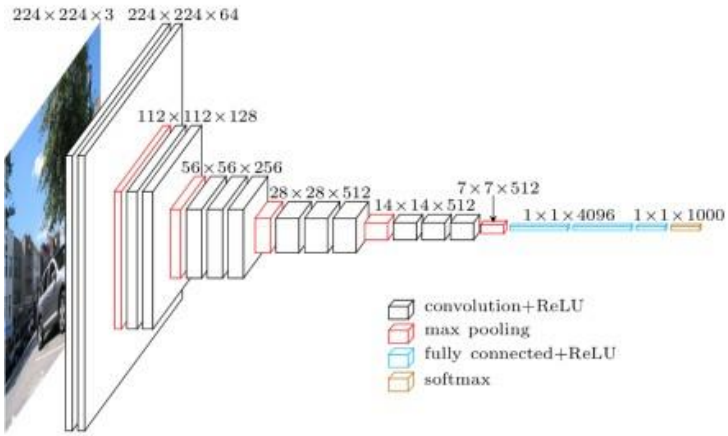
Садржај

Увод 3
Улазни подаци 4
Учитавање скупа података CIFAR-10 у Keras 4
Конволуциона неуронска мрежа за CIFAR -10 4
Закључак 9
Литература 10

Увод

Тема овог пројекта јесте да се развије и процени модел дубоког учења за препознавање објекара у Keras-у коришћењем VGG мреже. Учитаваће се CIFAR-10 класификациони подаци и обрађивати. За имплементацију ове конволуционе мреже коришћен је програмски језик python и његова библиотека keras, намењена креирању неуронских мрежа и имплементацији алгоритама дубоког учења.

VGG16 је један од бољих архитектонских модела. Најјединственија ствар у вези са VGG16 је то што су се уместо великог броја хиперпараметара фокусира на то да има конволуционе слојеве 3x3 са 1 кораком и увек се користи исти padding и maxpool слој од 2x2 са 2 корака.



Слика 1. Архитектура VGG16 мреже

Улазни подаци

Скуп података CIFAR-10 састоји се од 60.000 фотографија подељених у 10 класа. Класе укључују уобичајене предмете као што су авиони, аутомобили, птице, мачке и тако даље. Скуп података је подељен на стандардам начин, где се 50.000 слика користи за обуку модела, а преосталих 10.000 за процену његовог учинка. Фотографије су у боји са црвеним, зеленим и плавим компонентама, али су мале величине 32 x 32 пиксела.

Учитавање скупа података CIFAR-10 у Keras

Скуп података CIFAR-10 се веома једноставно учитава у Keras.

Keras има могућност да аутоматски преузима стандардне скупове података као што је CIFAR-10 и складишти их у директоријум ~/.keras/datasets помоћу функције cifar10.load_data().

Свака слика је представљена као тродимензионална матрица, са димензијама за црвену, зелену и плаву боју, као и са ширинином и висинином.

Конволуциона неуронска мрежа за CIFAR -10

Проблем CIFAR-10 најбоље се решава коришћењем Конволуционе неуронске мреже. На почетку је потребно дефинисати све класе и функције које ће се користити у овом задатку.

```
# Simple CNN model for the CIFAR-10 Dataset
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from tensorflow.keras import optimizers
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
import matplotlib.pyplot as plt
```

Слика 2. Дефинисање класа

Након дефинисања класа потребно је позвати функцију која ће учитавати CIFAR-10 скуп података.

```
# load data
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

Слика 3. Учитавање CIFAR-10 скупа

Вредности пиксела су у опсегу од 0 до 255 за сваки од црвених, зелених и плавих канала. Боље решење ће се добити ако се ради са нормализованим подацима. Улазне податке лако можемо да нормализујемо на опсег од 0 до 1 тако што ћемо сваку вредност поделити са 255. Подаци који се учитавају морају бити у облику целих бројева, тако да их морамо средити пре учитавања.

```
# normalize inputs from 0-255 to 0.0-1.0
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train = X_train / 255.0
X_test = X_test / 255.0
```

Слика 4. Нормализација улазних података

Излазне променљиве су дефинисане у векторском облику са вредностима од 0 до 1 за сваку класу. За лакше коришћење алгоритма потребно их је превести у бинарну матрицу. Пошто постоји 10 класа за овај проблем, може се очекивати да ће бинарна матрица имати ширину од 10.

```
# one hot encode outputs
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]
```

Слика 5. Свођење излазних података на бинарну матрицу

Користиће се структура са два конволуциона слоја праћена максималним удруживањем и изравнавањем мреже до потпуно повезаних слојева како би се направила предвиђања.

Архитектуру мреже можемо сумирати на следећи начин:

- Два улазна слоја конволуције, 64 мапе карактеристика величине 3x3, функција активације исправљача и ограничење тежине максималне норме је постављен на 3
- Max Pool слој величине 2x2
- Два слоја конволуције, 128 мапа карактеристика величине 3x3, функција активације исправљача и ограничење тежине максималне норме је постављен на 3
- Max Pool слој величине 2×2
- Три слоја конволуције, 256 мапа карактеристика величине 3x3, функција активације исправљача и ограничење тежине максималне норме је постављен на 3
- Max Pool слој величине 2×2

• Три слоја конволуције, 512 мапа карактеристика величине 3x3, функција активације исправљача и ограничење тежине максималне норме је постављен на 3
• Max Pool слој величине 2x2
• Три слоја конволуције, 512 мапа карактеристика величине 3x3, функција активације исправљача и ограничење тежине максималне норме је постављен на 3
• Max Pool слој величине 2x2
• Flatten слој
• Два потпуно повезана слоја са 4096 јединица и функцијом активације исправљача
• Потпуно повезан излазни слој са 2 јединице и softmax функцијом активације

```
# Create the model
model = Sequential()
model.add(Conv2D(input_shape = (32, 32, 3), filters = 64, kernel_size = (3, 3), padding = "same", activation = "relu"))
model.add(Conv2D(filters = 64, kernel_size = (3, 3), padding = "same", activation = "relu"))
model.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2)))
model.add(Conv2D(filters = 128, kernel_size = (3, 3), padding = "same", activation = "relu"))
model.add(Conv2D(filters = 128, kernel_size = (3, 3), padding = "same", activation = "relu"))
model.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2)))
model.add(Conv2D(filters = 256, kernel_size = (3, 3), padding = "same", activation = "relu"))
model.add(Conv2D(filters = 256, kernel_size = (3, 3), padding = "same", activation = "relu"))
model.add(Conv2D(filters = 256, kernel_size = (3, 3), padding = "same", activation = "relu"))
model.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2)))
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = "same", activation = "relu"))
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = "same", activation = "relu"))
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = "same", activation = "relu"))
model.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2)))
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = "same", activation = "relu"))
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = "same", activation = "relu"))
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = "same", activation = "relu"))
model.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2)))
model.add(Flatten())
model.add(Dense(units = 4096, activation = "relu"))
model.add(Dense(units = 4096, activation = "relu"))
model.add(Dense(units = 10, activation = "softmax"))
```

Слика 6. Архитектура мреже

Након креирања softmax слоја модел је коначно припремљен. Сада треба саставити модел.

Логаритамска функција губитка се користи са алгоритмом оптимизације стохастичког градијента спуштања који је конфигурисан са великим импулсом и опадање тежине почиње са стопом учења од 0,01. Ако обука много одскаче од епоха, онда треба смањити стопу учења како би достиг глобалне минимуме.

```
epochs = 5
lrate = 0.01
decay = lrate/epochs
sgd = optimizers.SGD(lr = lrate, momentum = 0.9, decay = decay, nesterov = False)
model.compile(loss = 'categorical_crossentropy', optimizer = sgd, metrics = ['accuracy'])
model.summary()
```

Слика 7. Логаритамска функција губитка

Овај модел може да се уклопи са 5 епоха и величином серије 32. Када је модел уклопљен, процењује се на скупу тестних података и штампа се тачност класификације.

```
# Fit the model
history = model.fit(X_train, y_train, validation_data = (X_test, y_test), epochs = epochs, batch_size = 32)
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose = 0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Слика 8. Прилагођавање модела

Покретање кода прво се сумира структура мреже која потврђује да је дизајн исправно имплементиран.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_13 (Conv2D)           (None, 32, 32, 64)        1792
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 32, 32, 64)        36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 16, 16, 128)       73856
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 16, 16, 128)       147584
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 8, 8, 256)         295168
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 8, 8, 256)         590080
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 8, 8, 256)         590080
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 4, 4, 256)         0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 4, 4, 512)         1180160
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 4, 4, 512)         2359808
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 4, 4, 512)         2359808
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 2, 2, 512)         0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 2, 2, 512)         2359808
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 2, 2, 512)         2359808
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 2, 2, 512)         2359808
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 1, 1, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 4096)              2101248
_________________________________________________________________
dense_4 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dense_5 (Dense)              (None, 10)                40970
=================================================================
Total params: 33,638,218
Trainable params: 33,638,218
Non-trainable params: 0
```

Слика 9. Сумирање структуре мреже

Тачност и губитак класификације се штампа након сваке епохе и на скуповима података за обуку и на скуповима података на тесту.

```
Epoch 1/5
1563/1563 [==============================] - 11508s 7s/step - loss: 2.3031 - accuracy: 0.0978 - val_loss: 2.3027 - val_accuracy: 0.1000
Epoch 2/5
1563/1563 [==============================] - 11414s 7s/step - loss: 2.3028 - accuracy: 0.0983 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 3/5
1563/1563 [==============================] - 11448s 7s/step - loss: 2.3027 - accuracy: 0.0980 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 4/5
1563/1563 [==============================] - 11401s 7s/step - loss: 2.3027 - accuracy: 0.0979 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 5/5
1563/1563 [==============================] - 11434s 7s/step - loss: 2.3027 - accuracy: 0.0982 - val_loss: 2.3026 - val_accuracy: 0.1000
Accuracy: 10.00%
```

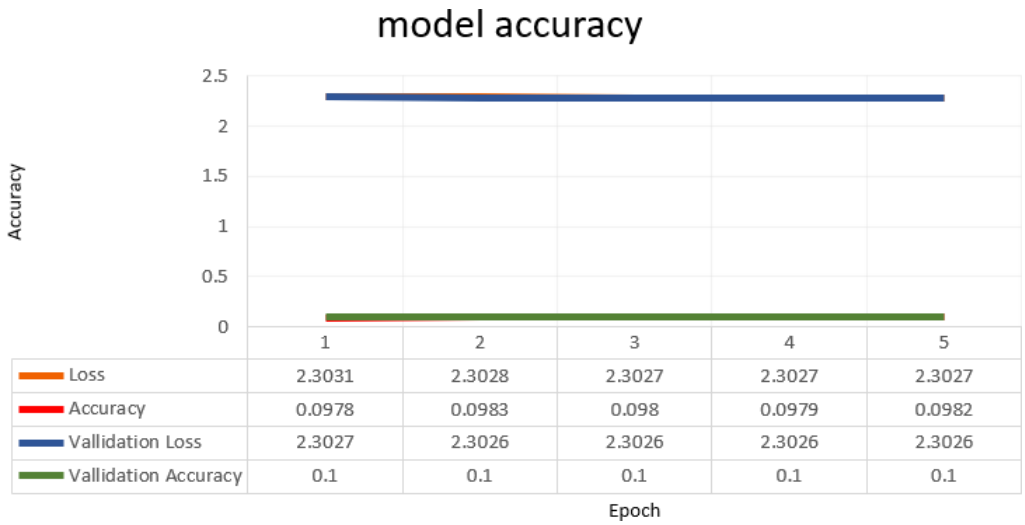
Слика 10. Класификација сваке епохе

Модел је оцењен на тест сету и постиже тачност од 10,00%.

```
plt.plot(history.history["accuracy"], color = 'red')
plt.plot(history.history['val_accuracy'], color = 'green')
plt.plot(history.history['loss'], color = 'orange')
plt.plot(history.history['val_loss'], color = 'blue')
plt.title("model accuracy")
plt.ylabel("Accuracy")
plt.xlabel("Epoch")
plt.legend(["Accuracy", "Validation Accuracy", "loss", "Validation Loss"])
plt.show()
```

Слика 11. Визуелизација тачности и губитака

Када се модел обучи, може се визуализовати тачност обуке/валидације и губици. Све тачности обуке/валидације и губици се чувају у history и одатле ће се визуализовати.



Слика 12. Модел тачности

Закључак

Проблем аутоматске класификације фотографија објеката је тежак због скоро бесконачног броја пермутација објеката, позиција, осветљења и тако даље. Модел је обучен за веома мали број епоха, 5, а уобичајено је да се велике конволуционе неуронске мреже обучавају за стотине или хиљаде епоха. Смањењем броја епоха и слојева то јест смањењем квалитета слика, знатно краће траје обука мреже.

Мрежа се додатно може унапредити постављањем већих димензија слике на улазу у мрежу. Може се повећати и дубина мреже која може укључити више мапа карактеристика на улазу и мање агресивно удруживање, али то повећава дужину трајања тренирања мреже.

Литература

[1] - Object Classification with CNNs using the Keras Deep Learning Library (machinelearningmastery.com) - 31.05.2022.
[2] - https://www.analyticsvidhya.com/blog/2020/02/learn-image-classification-cnn-convolutional- neural-networks-3-datasets/ - 31.05.2022.
[3] - https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c - 02.06.2022