Sveučilište u Rijeci – Odjel za informatiku

Diplomski studij informatike

Filip Čulina, Ivan Modrić, Đino Prenc

Android aplikacija za prepoznavanje marke i modela automobila

Seminarski rad

Mentori: izv. prof. dr. sc. Marina Ivašić Kos

Kristina Host, mag. inf.

Rijeka, veljača 2022

Sadržaj

1. Uvod

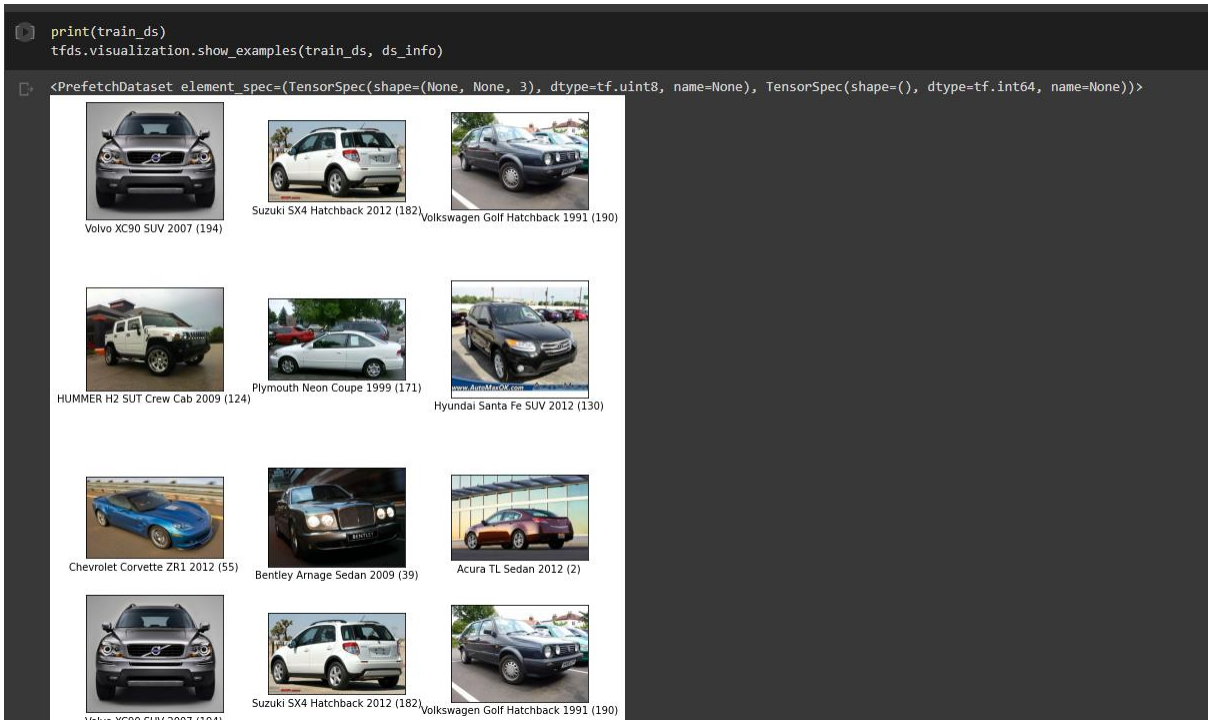
Jedno od neizostavnih područja računalne znanosti je strojno učenje koje postaje sve popularnije i doživljava veliki industrijski rast. Korištenjem resursa grafičkih kartica omogućuje se obrada velikih količina podataka u jako kratkom periodu što omogućuje „pametno učenje" i davanje preciznih odgovora na mnoge probleme i pitanja. Velikim svjetskim kompanijama stalo je do prikupljanja čim više informacija o vlastitim korisnicima, a kod tog procesa znatno im pomažu algoritmi stojnog učenja. Veliki napredak kod računalnih komponenti uvelike je pridonio razvoju strojnog učenja što je ovo područje znanosti populariziralo posebice u zadnjih desetak godina. Primjerice, treniranje neuronskih mreža danas je moguće izvesti na gotovo prosječnom stolnom ili prijenosnom računalu u svega nekoliko minuta, što je uvelike pridonijelo razvoju područja strojnog učenja. Nije potrebno previše isticati strjeloviti razvoj pametnih mobilnih uređaja koji su danas dostupni svima, stalno i svugdje stoga je i samo područje izrade mobilnih aplikacija doživjelo pravi procvat u zadnjih nekoliko godina. Stoga prava čarolija nastaje spajanjem mobilne aplikacije i algoritma pametnog učenja, a u našem slučaju android aplikacije za prepoznavanje marke i modela automobila.

2. Motivacija

Mnogi ljudi gledajući neki automobil često se pitaju o kojem automobilu je zapravo riječ. Kako život u neznanju može biti nezgodan, motivirani ovim problemom odlučili smo kreirati aplikaciju kojom se može fotografirati neki automobil ili pak odabrati postojeću fotografiju iz galerije te brzo i bezbolno dobiti informacije o traženome automobilu. Aplikacija je prilagođena za android uređaje, a brzinu i fluidnost prepoznavanja automobila osigurali smo tako da se samo prepoznavanje slike događa na udaljenom serveru. Iako ovakav pristup ubrzava vrijeme izvođenja, kod kompleksnih modela mana je što aplikacija treba biti povezana s internetom kada korisnik želi pokrenuti prepoznavanje. Za izgradnju aplikacije koristili smo Android Studio i programski jezik Java. Server s kojim aplikacija komunicira baziran je na Python Flask biblioteci, a sam model konvolucijske mreže napravljen je pomoću Tensorflow tj. Keras biblioteke.

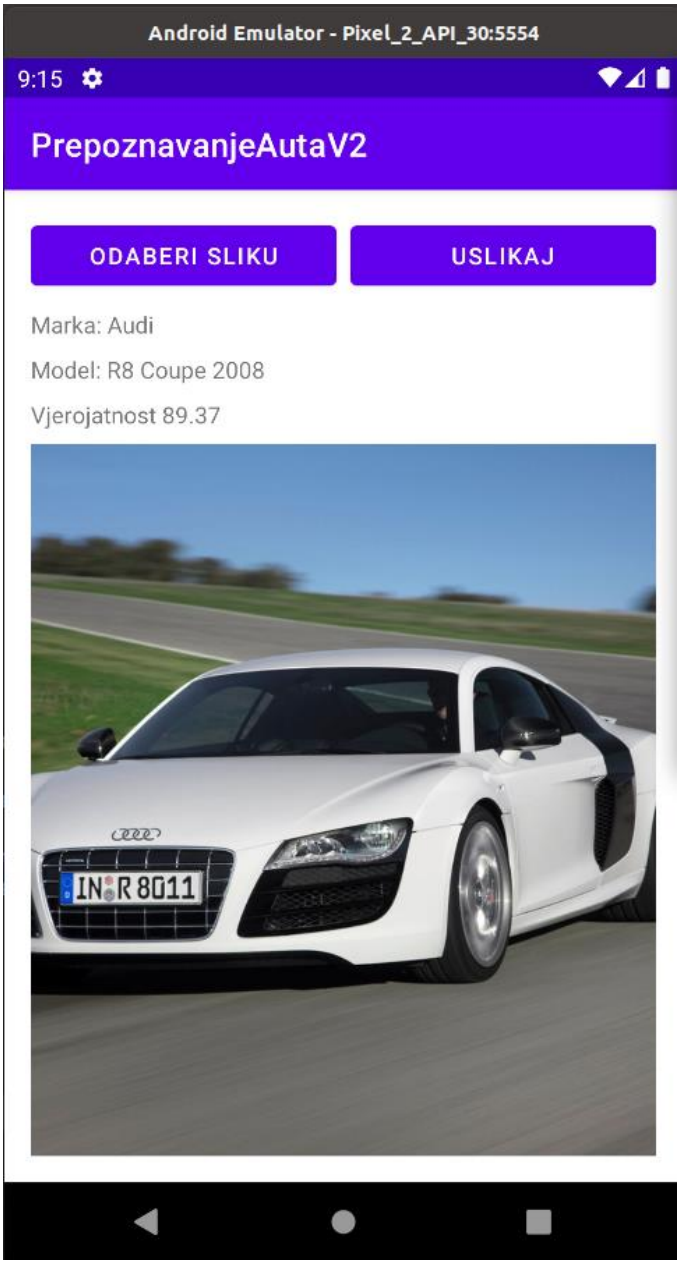
3. Podaci korišteni u konvolucijskoj mreži

Zbog velikog broja postojećih modela automobila na cesti nije bilo moguće da naš mali tim fotografira svaki auto ili da pronađemo dovoljno velik broj uzoraka za svaki model automobila. Svjesni spomenutog problema odlučili smo pronaći i iskoristiti neki od gotovih skupova podataka (eng. *dataset*). S obzirom na popularnost automobila očekivali smo veliki broj različitih skupova podataka za današnje automobile, no ipak smo naišli na podosta manji broj nego što smo očekivali. Jedan od najpovoljnijih skupova podataka na koje smo naišli je Cars196 (http://ai.stanford.edu/~jkrause/cars/car_dataset.html). Radi se o 196 različitih modela automobila proizvedenih do 2012. godine. Navedeni skup podataka također sadrži 16 185 slika automobila koji su raspoređeni gotovo u omjeru 50:50, točnije postoji 8144 slika namijenjenih za treniranje te 8041 slika koje su predviđene za testiranje kreiranog modela. U dataset ulazi i devkit koji sadrži nazive, dimenzije te potrebne okvire za treniranje modela.



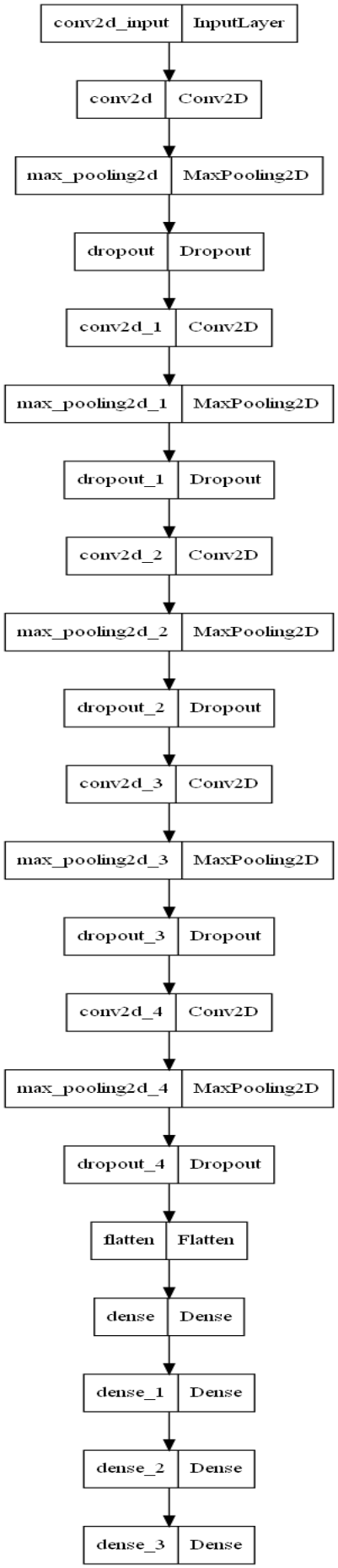
4. Android aplikacija

Izrađena aplikacija funkcionira na sljedeći način. Prilikom ulaska u aplikaciju korisniku je ponuđeno učitavanje fotografije iz galerije ili fotografije koja se može izravno kreirati (uslikati) kroz aplikaciju. Prilikom prvog korištenja bilo koje od navedenih funkcionalnosti potrebno je dati dozvolu za čitanje vanjske memorije, odnosno dozvolu za korištenje kamere uređaja. Nakon toga slika se šalje na AWS EC2 server na kojem je instaliran Python Flask okvir. Na server pristiže fotografija u formatu Base64 znakovnog niza gdje se potom vrši učitavanje modela koji sadrži konvolucijsku neuralnu mrežu fotografija automobila te se učitana fotografije validira prema spomenutom modelu nakon čega se rezultat validacije vraća u aplikaciju i prikazuje korisniku na sučelju. Korisniku se prikazuju tri podatka. Marka automobila, model automobila te vjerojatnost koja ukazuje koliko je model uopće siguran da se radi upravo o navedenoj marki i modelu automobila.


Android Emulator - Pixel_2_API_30:5554
9:15
PrepoznavanjeAutaV2
ODABERI SLIKU
USLIKAJ
Marka: Audi
Model: R8 Coupe 2008
Vjerojatnost 89.37
IN R 8011

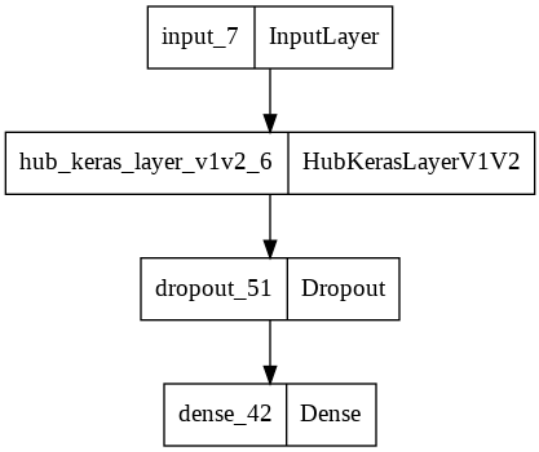
5. Rezultati modela i usporedba

Našu konvolucijsku mrežu započeli smo kopirajući arhitekturu korištenu na vježbama, ali smo odmah uočili slabu točnost. Jedan od načina kojim smo pokušali popraviti točnost bio je povećanjem broja epoha, no tada se situacija čak i pogoršala. Nakon 34 epohe točnost modela je počela opadati. Pokušali smo čak s 400 epoha u nadi da je pad točnosti s 34 epohe bila samo anomalija, ali nakon višesatnog treniranja točnost se i dalje nije popravila. Zatim smo krenuli istraživati različite metode i pristupe, no niti s jednim nismo bili zadovoljni. Jedna od preporuka bila je i implementacija posebnog prilagođenog sloja te istražujući kako ga implementirati shvatili smo da se radi o mnogo kompleksnijem pristupu nego što smo prvotno mislili.


conv2d_input InputLayer
conv2d Conv2D
max_pooling2d MaxPooling2D
dropout Dropout
conv2d_1 Conv2D
max_pooling2d_1 MaxPooling2D
dropout_1 Dropout
conv2d_2 Conv2D
max_pooling2d_2 MaxPooling2D
dropout_2 Dropout
conv2d_3 Conv2D
max_pooling2d_3 MaxPooling2D
dropout_3 Dropout
conv2d_4 Conv2D
max_pooling2d_4 MaxPooling2D
dropout_4 Dropout
flatten Flatten
dense Dense
dense_1 Dense
dense_2 Dense
dense_3 Dense

```
----------------------------------------------------------------
Epoch 1/30
2022-03-01 19:21:46.923043: I tensorflow/stream_executor/cuda/cuda_dnn.cc:368] Loaded cuDNN version 8200
100/100 [==============================] - 56s 532ms/step - loss: 5.2803 - accuracy: 0.0035 - val_loss: 5.2775 - val_accuracy: 0.0125
Epoch 2/30
100/100 [==============================] - 54s 542ms/step - loss: 5.2743 - accuracy: 0.0094 - val_loss: 5.2752 - val_accuracy: 0.0081
Epoch 3/30
100/100 [==============================] - 54s 542ms/step - loss: 5.2180 - accuracy: 0.0097 - val_loss: 5.1839 - val_accuracy: 0.0106
Epoch 4/30
100/100 [==============================] - 54s 536ms/step - loss: 5.1711 - accuracy: 0.0075 - val_loss: 5.1566 - val_accuracy: 0.0069
Epoch 5/30
100/100 [==============================] - 53s 531ms/step - loss: 5.1374 - accuracy: 0.0113 - val_loss: 5.1610 - val_accuracy: 0.0113
```

Nakon dugotrajnog testiranja korištenjem kompleksnijih slojeva proučili smo Lite Model Maker od Tensorflowa te smo se odlučili testirati model i s njime. Nakon testiranja utvrdili smo da je puno pouzdaniji od svega do tada korištenog. S njime smo uspjeli testiranjem na 5 epoha i batch sizeom postavljenim na 32 postići najveći postotak točnosti do sada od 31,15 %. Također postavljanjem batch sizea na 96 mislili smo da će se vrijeme smanjiti, a točnost povećati, no do toga nije došlo.




```
Total params: 3,664,100
Trainable params: 251,076
Non-trainable params: 3,413,024
_________________________________________________________________
None
Epoch 1/5
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descent.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(SGD, self).__init__(name, **kwargs)
254/254 [==============================] - 140s 540ms/step - loss: 5.0933 - accuracy: 0.0385
Epoch 2/5
254/254 [==============================] - 135s 531ms/step - loss: 4.4803 - accuracy: 0.1375
Epoch 3/5
254/254 [==============================] - 134s 528ms/step - loss: 4.0907 - accuracy: 0.2315
Epoch 4/5
254/254 [==============================] - 135s 532ms/step - loss: 3.8061 - accuracy: 0.3118
Epoch 5/5
254/254 [==============================] - 136s 535ms/step - loss: 3.5850 - accuracy: 0.3691
252/252 [==============================] - 136s 535ms/step - loss: 3.7198 - accuracy: 0.3115
```