SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI PROJEKT

Dohvat slika na temelju crteža

Juraj Šušnjara

Voditelj: *doc. dr. sc. Marin Šilić*

Zagreb, siječanj, 2017.

Sadržaj

1. Opis problema

Dohvaćanje slika (engl. Image Retrieval) je sustav za pretragu i pregledavanje slika iz velikih baza digitalnih slika. Tradicionalne metode najčešće koriste dodavanje metapodataka kao što su naslovi, ključne riječi ili opisi slika tako da dohvat može raditi na temelju tih anotacija. Ručno anotiranje slika je vremenski zahtjevno i skupo pa su se s vremenom pojavile učinkovitije metode. Razvoj računalnog vida omogućio je izbjegavanje tekstualnih opisa. Umjesto toga moguće je analizirati sami sadržaj slike (teksturu, boje, oblike itd.). Postoje razne metode analize sadržaja slike te pronalaska onih koji odgovaraju korisnikovom upitu.

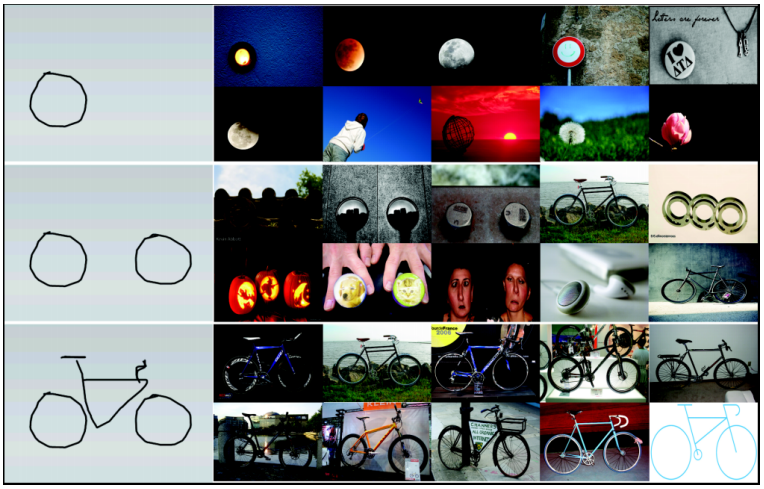
Cilj dohvaćanja fotografija iz crteža je omogućavanje korisnicima da što bolje izraze ono što žele. Neke stvari teže je opisati riječima dok ih crtežom možemo preciznije definirati. Na primjer želimo li pretražiti fotografije aviona u određenom položaju s određenim omjerom veličine trupa i krila i određenim logom zgodnije će biti napraviti crtež nego to sve opisati riječima. Na slici 1 pokazana je ova ideja. U prvom retku nacrtan je samo krug te model dohvaća slike mjeseca i ostalih okruglastih stvari. U zadnjem retku crtež bicikla dovršen je do kraja i vidimo kako model uspješno dohvaća tražene slike.

Dohvaćanje slika pomoću crteža težak je problem zbog toga što zahtjeva usporedbu između dvije različite domene (slike i crteži) koje je individualno teško za shvatiti te imaju poprilično različit izgled. Uobičajeni pristupi obuhvaćaju ručno oblikovane značajke koje se fokusiraju na rubove ili gradijente što nije primjenjivo na sve domene. Glavni razlog kompleksnosti ovog problema je što ljudi nisu dobri slikari. Prosječna osoba će crtež izraditi lijeno i istaknuti samo najosnovnije značajke. Ljudima takve crteže nije teško prepoznati ali naučiti računalo da radi tako nešto mnogo je teže. Razvoj konvolucijskih mreža omogućio je kvalitetno razumijevanje crteža i slika. Zasebno kategoriziranje jednih i drugih moguće je dovesti gotovo do ljudskih mogućnosti. Na taj način izrađeni su modeli koji dohvaćaju na temelju kategorizacije. Na primjer za crtež mačke dohvaćaju se sve slike mačaka bez obzira na sličnost mačaka na crtežu i skici. Takvo dohvaćanje smatra se uspješnim kod tog pristupa. Bitno je samo kategorizirati crtež. Jedan od radova koji se bavi tom tematikom opisan je u [6]. Kako bi se napravio kvalitetan dohvat slika na temelju crteža potrebno je osim kategorija uzeti u obzir i finije detalje koji se nalaze na crtežu. Moguće je napraviti među-domensku reprezentaciju za crteže i slike pomoću koje se dohvaćaju točne kategorije i fini detalji. Kako bi se to ostvarilo treba skupiti veliku količinu označenih crteža i pripadnih slika. To je napravljeno u [7]. U tom radu pokazano je sljedeće:

- Skupljanje velikog broje crteža stvarnih slika koje su izradili ispitanici. Baza sadrži

75471 crtež od 12500 slika koje pokrivaju 125 kategorija.

- Pristup navedenoj problematici pomoću dubokog učenja. Naučen je jedinstven prostor značajki za slike i crteže što istovremeno omogućava dohvaćanje slika pomoću crteža, crteža pomoću slika, slika pomoću slika i crteža pomoću crteža.
- Kako naučeni zajednički prostor vodi ka finom i značajno boljem dohvaćanju slika na temelju crteža od postojećih rješenja.



Slika 1: Dohvat slike na temelju crteža

2. Izrada modela

Baza crteža izrađena je tako da su korisnicima dane stvarne slike na temelju kojih su oni izradili jednostavne crteže kao što je prikazano na slici 2. Sam proces detaljno je opisan na [7]. Takva baza podataka korištena je kako bi se istrenirala zajednička reprezentacija crteža i slika tako da udaljenosti u prostoru značajki odgovaraju strukturalnoj i semantičkoj sličnosti između tih crteža i slika.

Slika 2: Crteži izrađeni na temelju slika

Duboke mreže koje su pokazale najbolje rezultate zahtjevale su kompleksne načine treniranja iz dva razloga:

1. Bez obzira što je baza crteža iz [7] najveća označena takva baza, duboke mreže sadrže desetke milijuna parametara koje treba naučiti pa zahtjevaju ogromnu količinu podataka. Zbog toga je potrebno napraviti predtreniranje modela.

2. Promatraju se dvije odvojene razine sličnosti - sličnost između crteža i točno one slike iz koje je ispitanik izradio taj crtež te sličnost na razini kategorije između crteža i svih slika u kategoriji tog crteža.

Kombinacijom obje forme kod nadziranog treniranja povećava performanse ali i kompleksnost. Ispituju se tri različite funkcije gubitka za treniranje mreža: sijamski gubitak, trostruki gubitak i klasifikacijski gubitak.

- **Sijamska mreža**. Par mreža istovremeno primaju par slika, svaka jednu sliku na svoj ulaz. Te slike su ili slične ili različite (tako su klasificirane). Sijamske mreže koriste kontrastnu funkciju gubitka:

$$L = l * d(S, I+) + (1 - l) * max(0, m - d(S, I-))$$

gdje je S crtež, $I+$ je slika istog objekta kao na crtežu, $I-$ slika različitog objekta od onog na crtežu, $d()$ euklidska udaljenost, m je margina, a $l \in \{0, 1\}$ je 1 za pozitivne, 0 za negativne parove. Tako će negativni parovi biti razdvojeni do maksimalne udaljenosti zadane marginom, a pozitivni parovi će biti približeni u prostoru svojstava.

- **Trostruka mreža**. Trostruke mreže su slične sijamskim. Razlika je što se koristi treniranje u formi "ulaz a bi trebao biti bliži ulazu b od ulaza c". Funkcija gubitka je tada:

$$L = max(0, m + d(S, I+) - d(S, I-))$$

  Tako možemo izraziti finije odnose od sijamskog gubitka koji nam samo govori trebaju li parovi točaka biti blizu ili daleko. Sijamska mreža može biti konceptualizirana kao dvije mreže, a trostruka kao tri dok je to zapravo samo jedna mreža koja se iznova koristi za sve ulaze. Za ulaze crteža i slika potrebno je trenirati mreže s odvojenim težinama jer su ulazi koje obrađuju različiti. Težine se mogu i dijeliti ukoliko možemo domene prebacivati jedne u drugu prije provlačenja kroz mreže (npr. sliku pretvoriti u crtež prije prolaska kroz mrežu).

- **Klasifikacijska mreža**. Kako bi uspješno izgradili model potrebno je uzet u obzir semantiku (kategoriju objekata) i fine detalje (poza, oblik, kut slike itd.). Ako korisnik nacrta konja ne želimo dohvaćati fotografije zebre ili krave. Zbog toga se uvodi i klasifikacijski gubitak pri treniranju modela. Koristi se *softmax* gubitak preko 125 kategorija.

2.1. Treniranje

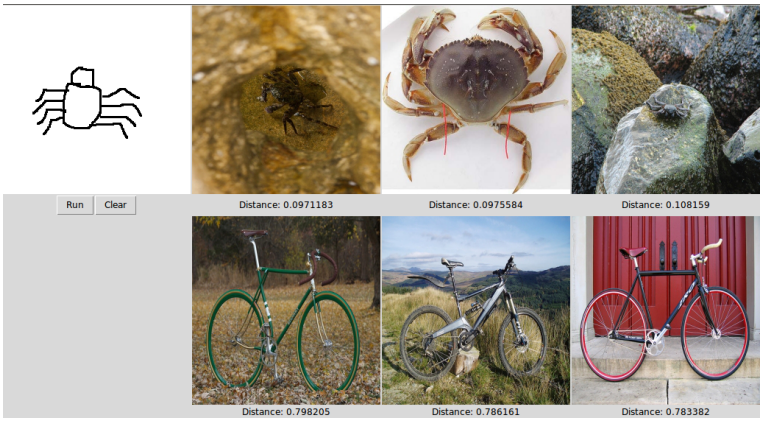
Za početak je potrebno istrenirati pod mreže kako bi znale kategorizirati crteže i slike (predtreniranje). Mreže neovisno uče težine prikladne za svaku domenu. Nakon toga duboke mreže trenirane su kako bi crteže i slike prebacile u zajednički 1024-dimenzionalni prostor. Za to se koriste sijamski, trostruki i klasifikacijski gubitak. Prvo se trenira sa sijamskim gubitkom. Ulazi su parovi crtež-slika. Mreža se trenira s velikom marginom što znači da su svi parovi iz iste kategorije označeni kao slični (gotovo isto kako klasifikacijski gubitak). Nakon toga koristi se mala margina pa se sličnim smatraju samo parovi koji su ispitanici eksplicitno nacrtali. Dakle, crtež i slika zeca će biti razmaknuti u prostoru značajki ukoliko crtež nije izrađen na temelju te slike zeca. Nakon sijamskog trenira se s trostrukim gubitkom. Sada su ulazi u obliku tripleta $(S, I+, I-)$ što odgovara crtežu, odgovarajućoj slici i neodgovarajućoj slici. $I+$ i $I-$ uzorkuju se samo iz kategorije crteža (npr. crtež zebre s odgovarajućom i neodgovarajućom slikom

zebre) zbog toga što se istovremeno već koristi klasifikacijski gubitak koji će razlikovati crteže od slika različite kategorije. Kod mreže s trostrukim gubitkom obje grane slika dijele težine. Rezultat toga je da će sijamska i trostruka mreža imati jedan set težina za domenu crteža i jednu za domenu slika.

Konačna mreža sastoji se od dvije pod mreže čija je zadaća preslikavanje ulaza crteža i slika u zajednički 1024-dimenzionalan prostor značajki.

3. Opis razvijenog proizvoda

Za potrebe ovog projekta izrađena je jednostavna desktop aplikacija u *Pythonu*. Aplikacija se sastoji od prozora s praznim platnom za crtanje. Korisnik na tom platnu treba napraviti crtež slike koje želi dohvatiti. Pritiskom na gumb dohvaćaju se najsličnije i najrazličitije slike te se prikazuju udaljenosti tih slika od crteža. Udaljenosti su numeričke vrijednosti u intervalu $[0, 1]$. Računa se kosinusova udaljenost između vektora značajki crteža i slika. Na slici 3 prikazan je primjer dohvaćanja tri najsličnije i tri najrazličitije slike na temelju crteža.



Slika 3: Dohvaćanje slika preko izrađene aplikacije

Kako bi se ostvarila takva funkcionalnost preuzeta je istrenirana mreža koja na ulaz prima ili crtež ili sliku te na temelju toga računa njihovu reprezentaciju u prostoru značajki (1024-dimenzionalni vektor). Nadalje, preuzeta je *Sketchy* baza crteža. Mreža i baza mogu se skinuti s `http://sketchy.eye.gatech.edu` što je web stranica od proizvoda opisanog u [7]. Baza se sastoji od stvarnih slika objekata iz 125 različitih kategorija (JPG format). Svaka kategorija sadrži 100 slika te je sveukupno 12500 slika u bazi. To su slike koje se dohvaćaju na temelju danog crteža. U bazi se nalazi i po nekoliko crteža za svaku od slika. Preuzeta mreža istrenirana je na toj bazi. Sve slike iz baze potrebno je prebaciti u prostor značajki kako bi kasnije dohvaćanje trajalo što kraće. Dakle prije pokretanja aplikacije iterira se po svim slikama, pomoću preuzete istrenirane mreže računaju se značajke te se parovi (putanja do slike, vektor značajki) serijaliziraju i pohranjuju lokalno. Kada korisnik pokrene aplikaciju učitava se serijalizirani objekt koji predstavlja opisane parove te se vektori značajki iz parova indeksiraju pomoću algoritma najbližih susjeda [3]. Na taj način omogućava se učinkovit dohvat k najbližih slika. Nakon što korisnik napravi crtež i pritisne gumb, taj se crtež provlači kroz mrežu i računaju se njegove značajke. Na temelju tih značajki dohvaća se k najbližih vektora značajki iz prethodno opisanog indeksa. S obzirom da su lokalno spremljeni

parovi (vektor-putanja do slike) lako se dohvati k putanja do najbližih i najdaljih slika i prikažu se korisniku.

4. Tehničke značajke

Preuzeta istrenirana neuronska mreža pohranjena je kao *.caffemodel* te je za računanje značajki korišten **Caffe** radni okvir [1]. Takva neuronska mreža uz težine sadrži nazive slojeva, neurona, ulaza i izlaza te se može koristiti istovremeno za crteže i slike. To se radi tako da su s `https://github.com/janesjanes/sketchy` preuzete *.prototxt* datoteke s kojima se prije prolaska ulaznih podataka kroz mrežu može definirati koje slojeve treba koristiti. U tim datotekama opisana je struktura mreže. Učitavanje i inicijalizacija mreže prikazana je sljedećim kodom:

```
# Učitavanje modela za crteže iz preuzete mreže i .prototxt datoteke
sketch_net = caffe.Net(sketch_model_prototxt, caffemodel, caffe.TEST)
# Učitavanje modela za slike iz preuzete mreže i .prototxt datoteke
img_net = caffe.Net(image_model_prototxt, caffemodel, caffe.TEST)
# Definicija izlaznog sloja za crteže
output_layer_sketch = config.get('Network', 'output_layer_sketch')
# Definicija izlaznog sloja za slike
output_layer_image = config.get('Network', 'output_layer_image')
```

Potrebno je još definirati transformer koji će usrednjiti ulazne podatke, definirati kanale boja koji se koriste i preoblikovati podatke u oblik nad kojim je mreža trenirana. To je prikazano sljedećim kodom:

```
transformer = caffe.io.Transformer(
        {'data':np.shape(self.sketch_net.blobs['data'].data)})
transformer.set_mean('data', np.array([104, 117, 123]))
transformer.set_transpose('data', (2, 0, 1))
transformer.set_channel_swap('data', (2, 1, 0))
transformer.set_raw_scale('data', 255.0)
```

Sada je moguće na temelju ulaznog crteža ili slike izračunati vektor značajki u 1024-dimenzionalnom prostoru. Ispod prikazana *get_feature* funkcija na ulaz prima putanju do slike ili crteža za koje želimo izračunati značajke i *boolean* varijablu koja označava radi li se o slikama ili crtežima. U zadnjem dijelu funkcije koristi se *numpy* kako bi se preoblikovao izlaz iz mreže u *1x1024* vektor.

```
def get_feature(self, full_path, sketch=False):
    layer = self.output_layer_sketch if sketch else self.output_layer_image
    net = self.sketch_net if sketch else self.img_net
    inp = (self.transformer.preprocess('data', caffe.io.load_image(full_path)))
    sketch_in = np.reshape([inp], np.shape(self.sketch_net.blobs['data'].data))
    query = net.forward(data=sketch_in)
    query = np.copy(query[layer])
    query = np.reshape(query, [np.shape(query)[1]])
    return query
```

Nakon što je definirana mreža i računanje značajki pomoću mreže potrebno je napraviti indeks značajki od baze slika koja se pretražuje prilikom korisnikova upita. Kao što je već navedeno koristi se baza iz [7]. Funkcija *create_features_idx* za argumente prima model, korijenski direktorij baze slika i zastavicu koja označava radi li se o slikama ili crtežima. Izlaz iz funkcije je rječnik koji sadrži liste vektora značajki i putanja do slika iz kojih su te značajke izračunate.

```
def create_features_idx(model, root_dir, sketch=False):
    image_paths = []
    features = []
    print 'Calculating features'
    counter = 0
    for dirpath, dirnames, filenames in walk(root_dir):
        for img_path in filenames:
            counter += 1
            print 'Image', counter, '->', img_path
            full_path = dirpath + '/' + img_path
            feature = model.get_feature(full_path, sketch=sketch)
            image_paths.append(full_path)
            features.append(feature)
    return {'features': features, 'image_paths': image_paths}
```

Pomoću liste izračunatih vektora značajki izrađuje se indeks k najbližih susjeda. Koristi se implementacija iz *sklearn* paketa. Koristeći indeks može se jednostavno tražiti da se za neki ulazni vektori izračuna sličnost sa svakim vektorom u indeksu i vrati k najbližih kao što je prikazano sljedećim kodom:

```
# index['features'] je lista svih vektora značajki iz baze slika
# query je vektor značajki za koji želimo pronaći najsličnije iz baze
nbrs = NearestNeighbors(n_neighbors=np.size(self.index['features'], 0),
            algorithm='brute', metric='cosine').
                fit(self.index['features'])
distances, indices = nbrs.kneighbors([query])
```

Za izradu korisničkog sučelja korišten je *Tkinter*, standardni *Python*-ov GUI paket.

4.1. Struktura projekta

Projekt se sastoji od sljedećih *Python* skripti:

- **model.py** - Sadrži API za pristup modelu (računanje vektora značajki ulaznih slika).
- **index.py** - Sadrži API za dohvaćanje k najsličnijih i najrazličitijih slika na temelju korisnikova crteža. Pokretanjem ove skripte računaju se vektori značajki na temelju baze slika koje trebaju biti lokalno pohranjene u direktoriju. Vektori značajki i putanje do svake slike serijaliziraju se u *Python* rječnik koji se sprema lokalno na disk kako kasnije nebi bilo potrebno ponovno računati sve značajke. Na taj način znatno se ubrzava proces dohvaćanja rezultata.
- **main.py** - Glavni program koji učitava model i serijalizirani indeks te pokreće desktop aplikaciju u kojoj korisnik može praviti crteže i dohvaćati rezultate.
- **config.py** - Sadrži konfiguracijski objekt koji iz konfiguracijske datoteke učitava sve potrebne parametre. Parametri su opisani niže.

Projekt se sastoji od sljedećih datoteka:

- **triplet_googlenet_finegrain_final.caffemodel** Istrenirani model. [2]
- **imagedeploy.prototxt** Opisana struktura mreže za obradu slika. [4]
- **sketchdeploy.prototxt** Opisana struktura mreže za obradu crteža. [5]
- **config.cfg** Konfiguracijska datoteka (primjer na slici 4) u kojoj se zadaje:
  - *caffemodel* = putanja do modela
  - *sketch_prototxt* = putanja do .prototxt datoteka za slike

- *image_prototxt* = putanja do .prototxt datoteka za crteže
- *output_layer_sketch* = naziv izlaznog sloja za crteže
- *output_layer_image* = naziv izlaznog sloja za slike
- *image_root_dir* = direktorij u kojem se nalazi baza slika
- *sketch_root_dir* = direktorij u kojem se nalazi baza crteža (ukoliko želimo pretraživati crteže umjesto slika)
- *pickle_img_idx* = datoteka u koju se pohranjuje serijalizirani rječnik koji sadrži vektore značajki i putanje do odgovarajućih datoteka iz baze iz kojih su ti vektori izračunati
- *query_sketch* = datoteka u koju se sprema korisnikov crtež koji služi kao upit za dohvaćanje slika

```
[Network]
caffemodel = model/triplet_googlenet_finegrain_final.caffemodel
sketch_prototxt = model/sketchdeploy.prototxt
image_prototxt = model/imagedeploy.prototxt
output_layer_sketch = pool5/7x7_s1_s
output_layer_image = pool5/7x7_s1_p

[Index]
image_root_dir = /home/juraj/projekt/sketch/rendered_256x256/256x256/photo/tx_000000000000
sketch_root_dir = /home/juraj/projekt/sketch/rendered_256x256/256x256/sketch/tx_000000000000
pickle_img_idx = img_idx.pkl

[Canvas]
query_sketch = query.png
```

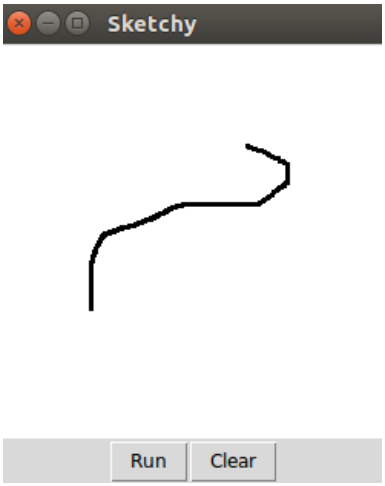
Slika 4: Primjer konfiguracijske datoteke

5. Upute za korištenje

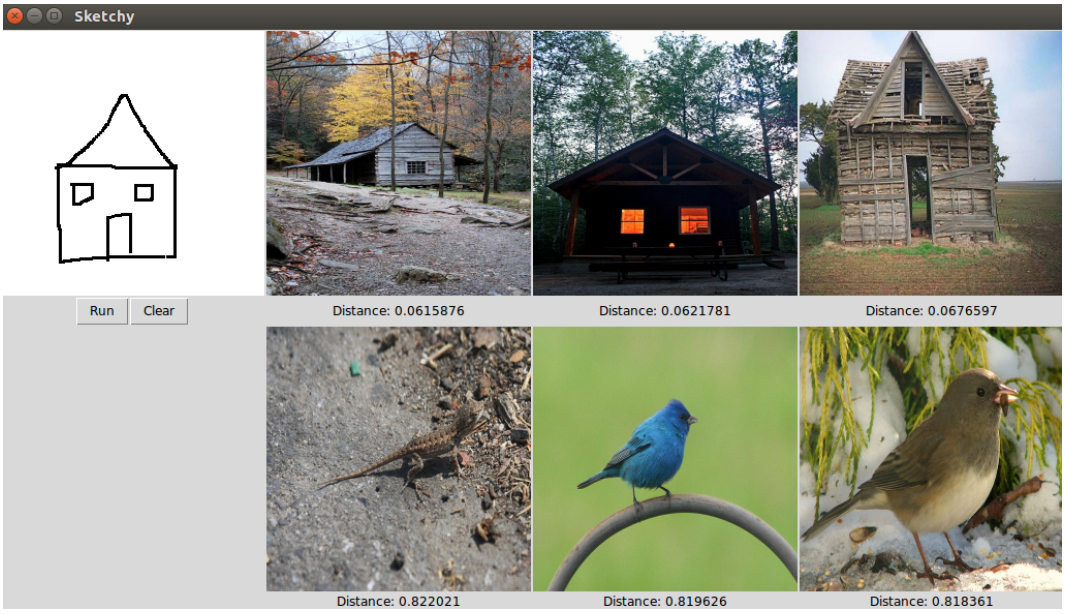
Kako bi se aplikacija mogla koristiti potrebno je prvo preuzeti određene datoteke i podesiti konfiguracijsku datoteku kako je opisano u prethodnom poglavlju. Nakon toga treba pokrenuti skriptu *index.py* koja će pretvoriti sve slike iz baze u vektore i spremiti serijalizirani rječni kao datoteku na disku. Prilikom pokretanja skripte kao argument predati konfiguracijsku datoteku. S obzirom da je u bazi 12500 slika izvođenje ove skripte može potrajati i do nekoliko sati. Kada je ta skripta izvršena može se pokrenuti skripta *main.py* koja pokreće glavni program i prikazuje korisniku prozor za crtanje. Skripti treba predati istu konfiguracijsku datoteku. Čitav proces preglednije je prikazan ispod:

1. Preuzeti potrebne datoteke.
2. Podesiti konfiguracijsku datoteku.
3. Pokrenuti *index.py*: *python index.py -config config.cfg*
4. Pokrenuti *main.py*: *python main.py -config config.cfg*

Kada se otvori glavni prozor korisniku se prikaže prazno platno na kojem može napraviti svoj crtež (slika 5). Pritiskom na gumb *Clear* briše se sve što je do sada nacrtano. Pritiskom na gumb *Run* pokreće se dohvaćanje najsličnijih i najrazličitijih slika od korisnikovog crteža (slika 6).



Slika 5: Početni prozor aplikacije



Slika 6: Prikaz rezultata

6. Zaključak

Eksponencijalni razvoj dubokog učenja promijenio je način na koji gledamo slike. Omogućio nam je da radimo izrazito kompleksne analize različitih značajki. U ovom radu obrađeno je kategoriziranje crteža i određivanje značajki koje su zajedničke slikama i crtežima. To uvelike proširuje opcije prilikom pretraživanja slika na Internetu. Korisniku je nekad jednostavnije nacrtati nešto nego to izraziti riječima. Ovakav dohvat slike savršen je za pametne uređaje i tablete. Koristeći zaslone osjetljive na dodir veoma je lako i vremenski brzo napraviti crtež. Zbog toga bi za budući rad bilo zanimljivo napraviti Android ili iOS aplikaciju pomoću koje bi korisnici crtali i pretraživali neku bazu slika. S obzirom da tako nešto komercijalno još ne postoji bilo bi dobro i promatrati koliko bi to korisnicima bilo učinkovito i zabavno za svakodnevnu uporabu.

7. Literatura

[1] Caffe. *http://caffe.berkeleyvision.org*.

[2] caffemodel. *https://goo.gl/cqXm7F*.

[3] k Nearest Neighbors. *http://www.scholarpedia.org/article/K-nearest_neighbor*.

[4] Image prototxt. *https://github.com/janesjanes/sketchy/blob/master/models/triplet_googlenet/Triplet_googlenet_imagedeploy.prototxt*.

[5] Sketch prototxt. *https://github.com/janesjanes/sketchy/blob/master/models/triplet_googlenet/Triplet_googlenet_sketchdeploy.prototxt*.

[6] John Collomosse Rui Hu. *A Performance Evaluation of Gradient Field HOG Descriptor for Sketch Based Image Retrieval*. 2013.

[7] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: Learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (proceedings of SIGGRAPH)*, 2016.