UNIVERZITET U NIŠU
ELEKTRONSKI FAKULTET
Katedra za računarstvo

Tehnički izveštaj

Detekcija voća i povrća

Duboko učenje

Student
Lazar Đorđević, br ind. 1413

Mentor:
prof. dr. Aleksandar Milosavljević

Niš, 2025

Sadržaj

Uvod

Cilj ovog projekta je da se napravi model za klasifikaciju voća i povrća na skupu Fruit-360-dataset[Fruit-360 repository]. Za određivanje uspešnosti modela koristiće se primarno rezultati rada autora [Mureșan, Oltean 2018]. U nastavku će najpre biti opisan skup podataka, zatim će biti dat kratak pregled korišćenih tehnologija i arhitekture projekta. Nakon toga će biti prikazana struktura modela koji su testirani i upoređivanje rezultata.

Podaci

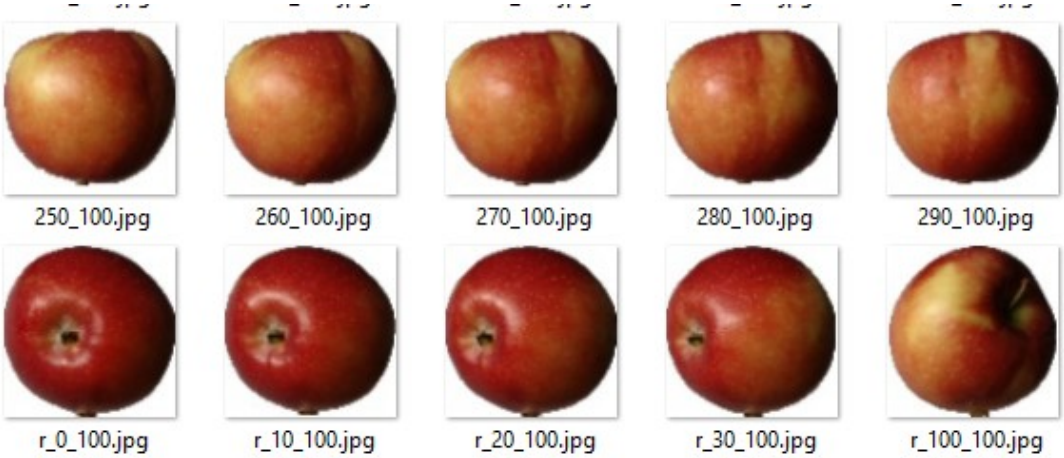
Originalni skup podataka zbog praktičnih razloga nije sadržao dovoljan broj podataka. Autori su vremenom dodavali nove klase i slike. Originalno su slike imale pozadinu i bile u različitim veličinama i klase nisu imale preveliki broj slika. Verzija skupa koji je ovde korišćen je skaliran na 100*100 piksela i ima 131-u klasu. Pozadina je flood-fill algoritmom popunjena belom bojom (nemaju svi pikseli intenzitet 255 i neki se razlikuju, ali su u takvom opsegu da je golim okom teško primetiti razliku).

U skorije vreme je objavljeno više radova koji su za treniranje modela koristili ovaj skup, ali nisu svi koristili istu verziju skupa. U originalnom radu je korišćen skup sa 120 klasa, a najnoviji skup sadrži preko 200 klasa. Skup podataka koji je korišćen u ovom projektu je dosta sličan skupu u originalnom radu.

Slike su podeljene u dva foldera Test i Training koji sadrže 22.688 i 67.692 fajla respektivno. Sve slike su raspoređene po folderima u zavisnosti od klase kojoj pripadaju, u svakom folderu je 200-700 za trening i preko 100 za test. Sve slike su imenovane na osnovu ugla pod kojim su snimljene i rotaciji po nekoj osi[1]. Da bi se ubrzalo vreme treniranja, iz skupa su izdvojene slike koje dovoljno dobro predstavljaju plod pod određenim uglom. Izdvojena su dva skupa. Oba skupa su pravljena na isti način, tako što su slike izdvojene da se sačuva svaki n-ti ugao. Pošto u osnovnom skupu nema slika pod svim uglovima (npr. postoje slike pod uglovima 1-20, ali između 20 i 30 stepeni može da ima jedna ili pak nijedna slika) izdvojene su slike tako da u prvom opsegu od 0 do n se izdvaja jedna slika, u opsegu od (n+1) do (2n) druga, ... i tako redom za svaki n-ti ugao i svaku osu rotacije. Ovo je rađeno samo za Training folder, a za evaluaciju i poređenje modela sa originalnim modelom autora skupa, korišćen je ceo Test folder. U nastavku skupovi *sN5* i *sN10* će se odnositi na skup koji ima sačuvan svaki 5-ti ugao, odnosno 10-ti ugao.

1 ime bez prefiksa označava uspravan plod, prefiksi r i r2 označavaju da je plod rotiran po 2. ili 3. osi

Na osnovu prethodno spomenutih skupova će biti trenirani modeli različitih arhitektura, sa različitim hiperparametrima, a najuspešniji će biti evaluirani na originalnom test skupu.



Slika 1: Primer slika iz sN5

Treniranje je uglavnom rađeno na *sn10,* pri čemu je vršena podela na trening i validacione podatke, u zavisnosti od potreba modela, uglavnom standardno 80/20%. Primećeno je da se na nekim modelima dešava overfitting na validacionim podacima, u tim slučajevima je osim drugih tehnika korišćena drugačija podela, u ekstremnim slučajevima je do trećine podataka korišćeno za validaciju, ali ne više od toga.

U radu [Mureșan, Oltean 2018] je kao cilj navedeno kreiranje modela koji može da pravi finu razliku između plodova, što podrazumeva razlikovanje više vrsta sličnih plodova, koje se ne razlikuju samo po sorti već i po tome u kakvom su stanju, odnosno kog su kvaliteta. Primer toga su klase *Apple Golden1-3* koje su iste sorte, ali je 1. počela da se kvari, a 2. i 3. su u relativno

Slika 2: S leva na desno primer klasa Apple Golden 1, 2 i 3

dobrom stanju. Slike na trening i test skupovima su iz tog razloga dosta prečišćene, da osvetljenje i pozadina ne bi uticali na donošenje odluka mreže. Naravno to znači da za mnoge praktične primene modeli neće biti relevantni. Zato će modeli biti isključivo trenirani na originalnim slikama, bez data augmentation

tehnika. Pošto su slike originalno u BGR formatu, jedina modifikacija je konvertovanje u RGB radi lakše analize slika.

Pokazalo se da data augmentation daje neznatne prednosti čak i na ovom prečišćenom test skupu, ali će u nastavku biti demonstrirano da znatno veći pomak može da se napravi poboljšanjem arhitekture same mreže, nego korišćenjem data augmentation[2] tehnika i proširivanjem skupa podataka[3]. Razlog za to je što je ova verzija skupa dosta balansirana i na trening i na test delu.

2 Primenjivanjem hue/saturation, rotiranjem i kombinovanjem HSL i grayscale verzije slike postignuta je 95,23% tačnost, dok je ista mreža trenirana sa RGB slikama postigla 94,43% [Mureșan, Oltean 2018]

3 Model u nastavku je postigao 96,28% nakon treniranja na *sn10*, nakon treniranja na celom trening skupu isti model je postigao 97,02%

Struktura projekta, tehnologije i biblioteke

Klasifikacija slika je jedan od klasičnih problema kojim se bavi duboko učenje. Za rešavanje datog problema korišćene su osnovne tehnike dubokog učenja. Korišćeni su modeli konvolucionih(u nastavku CNN) i višeslojnih perceptron mreža(u nastavku MLP). Primenjene tehnike obuhvataju pooling (max, average), dropout, batch normalization.

Za implementaciju ovih modela korišćena je Tensorflow biblioteka[Tf] u Python-u. Korišćena je verzija 2.10.1 [Tf-doc] zbog kompatibilnosti sa Windows-om. Tensorflow je u novijim verzijama prihvatio Keras API i prilikom instalacije automatski instalira verziju 2.10 Keras biblioteke[Keras-doc]. Keras je biblioteka koja podržava različite back-end biblioteke za mašinsko učenje. Što se tiče korišćenja, u kodu je Tensorflow korišćen preko Keras API-ja za kreiranje i rad sa modelima.

U ostatku koda, koji uglavnom obuhvata prikazivanje rezultata, izdvajanje i prosleđivanje podataka, uglavnom su korišćene osnovne funkcionalnosti Python-a, za konverziju u RGB korišćen je OpenCV, a za shuffle je korišćena numpy biblioteka.

Bitni delovi projekta su dostupni na Github repozitorijumu [jubilant-spoon].

Na repozitorijumu [Fruit-360 repository] ranije je bio dostupan kod koji sadrži verziju modela iz rada [Mureșan, Oltean 2018]. Radi lakšeg upoređivanja rezultata, taj kod je smešten u folderu *Fruit360_Org* u repozitorijumu [jubilant-spoon]. Taj model je služio u demonstrativne svrhe i po default-u je bio podešen da se trenira 25 epoha. Rezultat je bio 94,31% za metriku "sparse-categorical-acuracy". U originalnom radu model je postigao rezultat 95,23% [Mureșan, Oltean 2018]. Ovi rezultati će biti korišćeni za evaluaciju modela.

Glavne skripte za kreiranje, treniranje i podešavanje modela se nalaze u folderu *source/classification*. Korišćena je struktura koda sa sajta predmeta[DLv05]. Skripta *prepareData.py* je korišćena za redukciju osnovnog skupa podataka (sN5/10), prosleđivanjem parametra *angle*, koji je po defaultu 1, što označava da se sve slike kopiraju, što je korišćeno za dobijanje krajnjeg rezultata.

Skripta *model.py* sadrži strukture modela. Svaka metoda predstavlja tip modela, da bi se ubrzalo upoređivanje, a slojevi se dodaju sukcesivno preko kera API-ja.

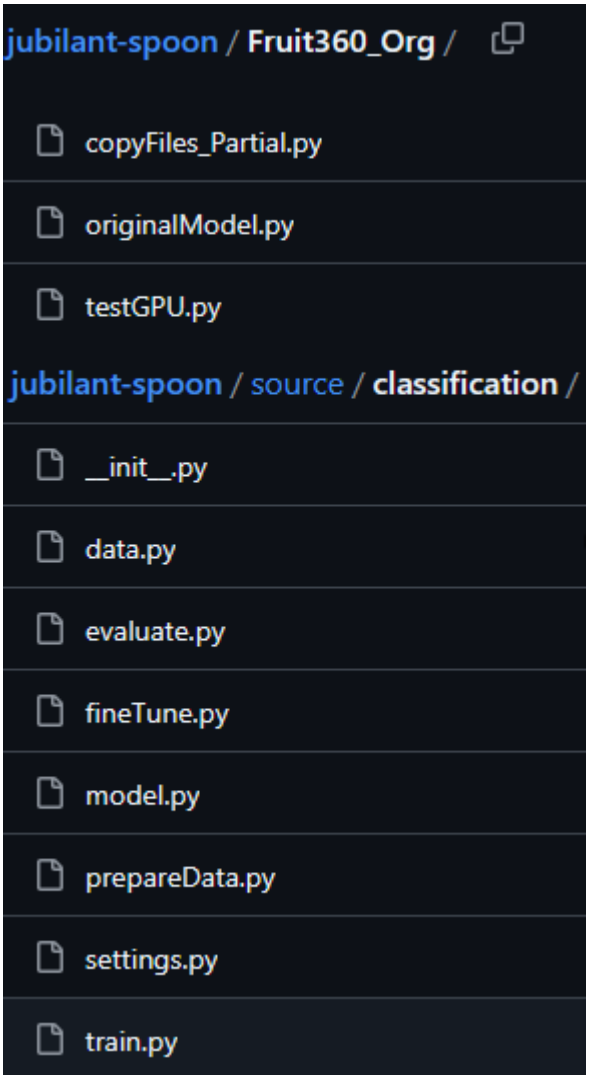
Za treniranje modela koristi se skripta *train.py*, i ona generiše sve potrebne informacije u vezi sa treniranjem (logovanje, prikaz loss funkcije i metrika kroz epohe, ...). Pri treniranju uvek je korišćena metrika *sparse_categorical_accuracy*, a za loss funkciju sparse_categorical_crossentropy. I metrika i loss su implementirani u Keras biblioteci. Rezultat je fajl koji sadrži informacije o treniranom modelu. Taj fajl koriste skripte fineTune i evaluate za pristup modelu. Prilikom treniranja korišćen je early stopping za 10 epoha a za redukovanje learning rate-a se po potrebi čeka najmanje 5 epoha ako ne dođe do promene. Za optimizator je na početku uglavnom korišćen RMSprop, a kasnije Nadam, da bi se ubrzala konvergencija. Ostali hiperparametri se uglavnom preuzimaju iz *settings.py*, ili iz samog modela. Za većinu modela je rađeno samo treniranje, evaluacija i fineTune su rađeni ako trening prođe kako treba (povećana tačnost na validacionom skupu, smanjenje overfitting-a i sl.).

Skripta *fineTune.py* funkcioniše isto, samo što ne kreira model nego ga čita iz fajla. U većini slučajeva efekat nije bio jasno vidljiv nakon ovog koraka, jer su početni modeli maltene uvek imali overfitting, ili na trening, ili na validacionom skupu. Ideja je bila da ukoliko se stane sa treniranjem zbog relativno malog broja zadatih epoha (na početku samo 50, kasnije 100) može da se desi earlyStopping u trenutku kad bi trebalo značajno da se smanji learningRate. Da se ne bi ponavljao ceo proces treniranja *fineTune* uglavnom nastavlja treniranje, ali sa dosta manjim learningRate-om. Ovde je uglavnom korišćen 10-1000 puta manji learningRate nego na treningu. *FineTune* je vrlo retko davao bolje rezultate na validacionom skupu, par puta čak i gore (što je verovatno posledica overfittinga na validacionom skupu, taj problem je imala većina modela na početku), ali je na test skupu često davao mala poboljšanja.

Za evaluaciju se koristi *evaluate.py*, ona funkcioniše slično kao prethodne dve. Nudi mogućnost prikaza grešaka.

Za pristup podacima se koristi *data.py*. U ovoj skripti se vrši podela na validacione i trening podatke, tako što se prvi podatak smešta u trening skup, sledeći u validacioni, a nakon toga svaki n-ti(uglavnom 5, ali u nekim slučajevima je korišćen i veći broj validacionih podataka da bi se smanjio overfitting) u validacioni.

Pristup hiperparametrima, putanjama na kojima se nalaze fajlovi podataka i rezultat su smešteni u *setting.py*.



Slika 3: Struktura projekta na repozitorijumu

Struktura modela neuronskih mreža

Zbog značajne redukcije podataka na trening skupu bilo je moguće za kraće vreme trenirati više modela. Zbog toga je tražena idealna arhitektura na sledeći način:

- najpre postaviti bitne slojeve (potpuno povezane(FC) u nastavku D-slojevi, i konvolucije u nastavku C-slojevi, Flatten, ulaz i izlaz)
- izvršiti inicijalni trening
- ako trening ne daje ni približno dobre rezultate značajno promeniti hiperparametre (learning rate, veličina filtera, broj neurona), ili izmeniti arhitekturu
- ako trening daje rezultate koji obećavaju (mogu da stignu između 70-90%, ali veoma lako stanu sa obučavanjem) dodati sekundarne slojeve (BatchNormalization(BN), MaxPool(MP), AveragePool(AvgP)) - ovde je cilj da se uoči da li se slojevi uklapaju u arhitekturu i koja su potencijalna mesta za dodavanje
- Nastaviti treniranje menjanjem hiperparametara u zavisnosti od problema koje ispoljava trenutni model (povećati regularizaciju, dodati dropout, posmatrati batch size u slučaju overfitting-a, u slučaju spore konvergencije pokušati drugi optimizator)
- Ukoliko je dostignut najbolji rezultat i svaki pokušaj podešavanja hiperparametara smanjuje rezultat modela dostignuta je granica trenutne arhitekture, treba dodati neki bitan sloj i nastaviti sa fineTuningom(pod pretpostavkom da taj jedan sloj ne menja bitno parametre mreže), ili preći na testiranje druge arhitekture

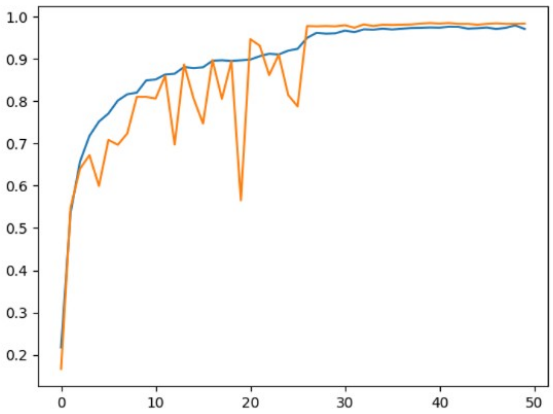
Zbog lakšeg opisa i naming konvencije koja je korišćena pri treniranju mreža na početku su date 4 generalne arhitekture koje su najviše testirane:

- D5 - MLP mreža sa 5 FC unutrašnjih slojeva
- D10 - mreža sa 10 FC slojeva
- Conv2_2 - konvoluciona mreža sa 2 Conv2D sloja i 2 FC sloja
- Conv2_3 - konvoluciona mreža sa 3 Conv2D sloja i 2 FC sloja

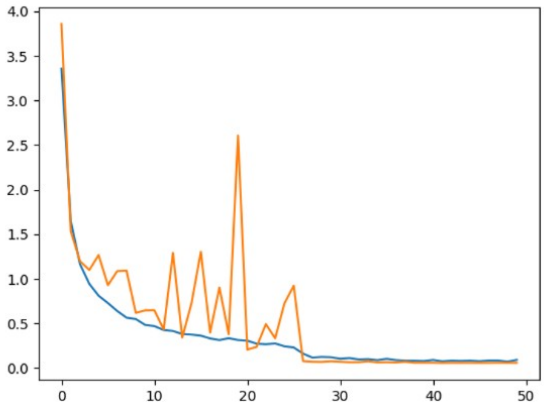
Svaka modifikacija dodavanjem drugih slojeva u neku od ovih arhitektura je bila naznačena izmenom imena modela na osnovu parametra koji je promenjen, ili imena sloja koji je dodat. Za pozivanje metoda za kreiranje modela se koristi naziv arhitekture.

Inicijalno testiranje

Prilikom prvog testiranja korišćene su plitke mreže do dubine 5 sa FC slojevima. Ono što je inicijalno moglo da se zaključi je da mreža može brzo da se trenira i da može da dostigne solidne rezultate na treningu i bez prevelikog menjanja hiperparametara.



Slika 4: Accuracy kroz epohe. Plava kriva predstavlja rezultat na trening skupu, a narandžasta na validacionom



Slika 4: loss na treningu(plavo) i validaciji(narandžasto)

Na slici 4 može da se vidi primer jednog od boljih pokušaja u inicijalnoj fazi. Ova mreža se sastoji od D-slojeva i par BN. Ispostavilo se da je neophodno dodati nekoliko BN, ali da ne postoji nikakav benefit od dodavanja na zadnji sloj, čak se tad i kvare performanse. Model je overfittovan na validacionom skupu. Uprošćavanjem modela se pogoršavaju rezultati, a jači model bi se još brže overfittovao. Odlučeno je da se usvoji arhitektura D5 i da se detaljnije ispita da li je moguće rešiti ovaj problem.

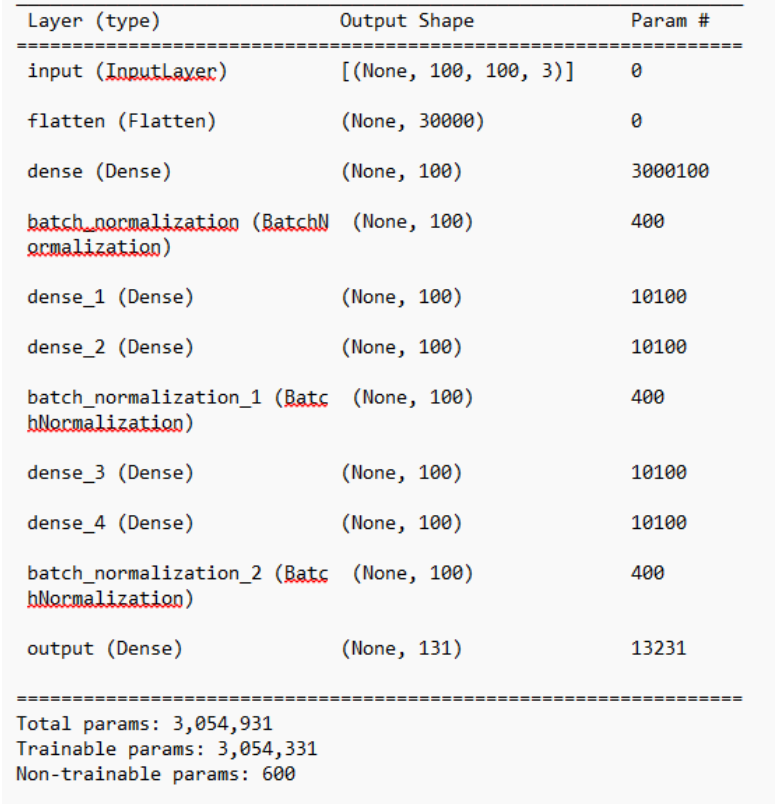
D5

Ova arhitektura je prva koja je detaljnije testirana. Zato možda postoji prostora za popravku rezultata, s obzirom da su neki zaključci izvedeni kasnije. Najverovatnije ne može da da najoptimalnije rezultate.

Najveći problem ove arhitekture je što ima ubedljivo najveći broj parametara u prvom sloju. Zaključeno je (a kasnije testiranjem i

na D10 potvrđeno) da se previše podataka gubi ukoliko ima manje od 30 neurona u prvom sloju. Sa 50 neurona mreža je u mogućnosti da postigne prosečne rezultate, a sa povećanjem neurona u prvom sloju rezultati postaju konzistentniji. Međutim već sa 100 neurona u prvom sloju se dobija 3 miliona parametara. Nakon toga treniranje počinje značajno da se usporava. Najviše su testirane verzije sa po 100 neurona, jer se činilo kao da daju najbolje rezultate. Prostim povećanjem neurona mreža se ili lakše overfittuje, ili počinje da gubi performanse.

Ono što nedostaje u analizi je slučaj kada zadnji sloj ima veći broj neurona od broja klasa, prvi sloj ima neznatno veći broj neurona od 100 a neuroni između 100 ili malo manje. Ti testovi nisu u dovoljnoj meri urađeni da bi se došlo do jasnog zaključka. Primećeno je da BN slojevi značajno doprinose stabilnosti rezultata. Sledeći primer je jedan od boljih rezultata ove arhitekture (slika 5):

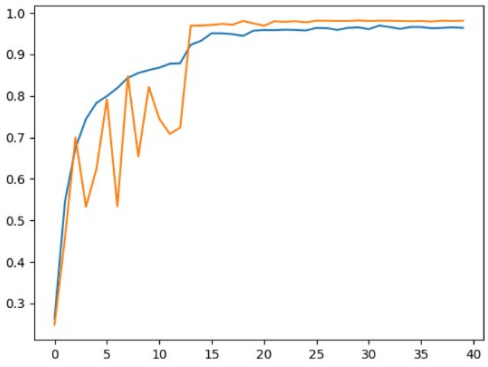


Layer (type)	Output Shape	Param #
input (InputLayer)	[(None, 100, 100, 3)]	0
flatten (Flatten)	(None, 30000)	0
dense (Dense)	(None, 100)	3000100
batch_normalization (BatchNormalization)	(None, 100)	400
dense_1 (Dense)	(None, 100)	10100
dense_2 (Dense)	(None, 100)	10100
batch_normalization_1 (BatchNormalization)	(None, 100)	400
dense_3 (Dense)	(None, 100)	10100
dense_4 (Dense)	(None, 100)	10100
batch_normalization_2 (BatchNormalization)	(None, 100)	400
output (Dense)	(None, 131)	13231

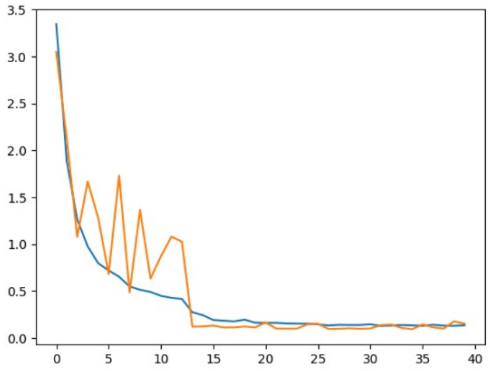
Total params: 3,054,931
Trainable params: 3,054,331
Non-trainable params: 600

Slika 5: D5 Mreža sa 3 Batch Normalization sloja (D5_3BN)

Ova mreža je na evaluaciji postigla 93,61%. To je u rangu mreže koja je u radu [Mureșan, Oltean 2018] bila označena kao neuspešna zbog overfitinga, što je i kod ove mreže slučaj (slika 6,7).

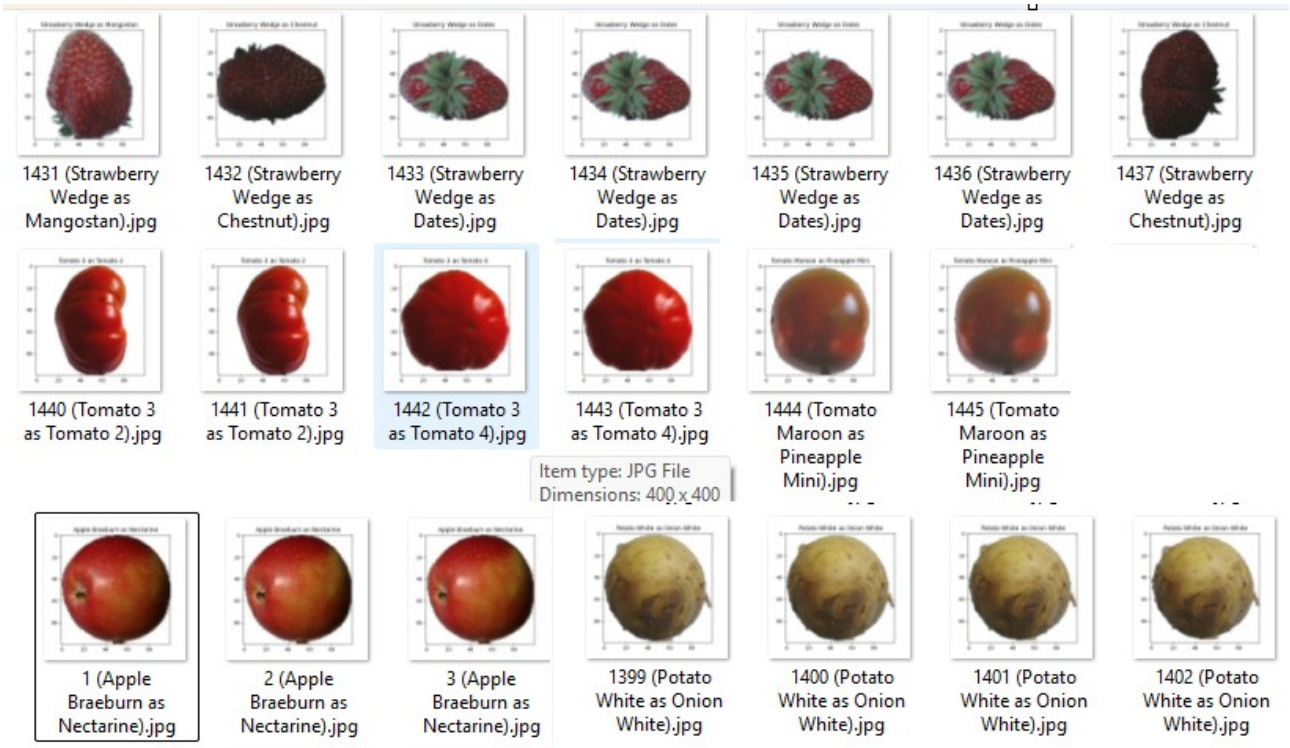


Slika 6: Accuracy u toku treniranja D5_BN5 (plavo - trening, validacija - narandžasto)



Slika 7: Loss na trening setu (plavo) i navalidacionom (narandžasto)

Kriva na treningu izgleda pravilno međutim zbog manjka validacionih podataka overfittuje na validacionom skupu i prerano prestaje sa treniranjem. Što se tiče analize grešaka D5, uglavnom je teško ustanoviti šta je mrežu navelo, međutim konkretno za D5_BN3 mogu da se uoče neke pravilnosti (slika 8):



Slika 8: Greške uočene kod D5_BN3 mreže - konzistentno meša određene klase iz određenih uglova

Naravno besmisleno je pretvarati se da tu postoji "viša "logika, ako model vidi istu jagodu kao tri različite klase, međutim jasno se vidi na ovom primeru da pod određenim uglom model daje konzistentno predviđanje. Takva konzistentnost nije zapažena kod

drugih modela ovog tipa arhitekture[4] koji su bili testirani i više je karakteristična za CNN. U tom smislu, iako je rađen fine tuning, moguće da bi ovaj model uz bolju raspodelu podataka mogao da postigne neki prosečan rezultat, s obzirom da je sa ovim parametrima rano završen trening, a konačan rezultat ulazi u kategoriju mreža koje su predstavljene u radu autora skupa podataka.

D10

Jedan faktor koji je uticao da se pređe na dublju mrežu je da nije intuitivno korigovati D5 bez pada performansi. Sa smanjivanjem broja neurona drastično padaju performanse. Povećavanje broja lako dovodi do eksplozije parametara. Prema tome i Dropout i L1 neće takvoj mreži doneti poboljšanje. Zaključeno je da proširivanje podataka iako može da da bolje performanse verovatno neće nadmašiti originalne rezultate.

Pokušano je sa dodavanjem slojeva. Jedan benefit bi kod te mreže trebalo da bude to što ako se uspešno redukuju dimenzije u početnim slojevima u krajnjim slojevima bi moglo da se doda više neurona bez rizika od eksplozije parametara. Sa većom dubinom i povećanjem broja neurona ima više izgleda da će regularizacija, batch_size i dropout imati efekta. Doduše te tehnike su imale efekta i kod D5, ali često negativne.

Očigledan previd kod pokušaja sa D10 je to što se nije uzelo u obzir da postoje očigledni limiti u redukciji početnih slojeva. Ni jedna mreža ove arhitekture nije prešla 90% na treningu. Ova arhitektura se mnogo sporije trenirala, bilo je dosta teže podesiti hiperparametre. Pokazalo se da je maltene neminovno da dođe do overfittinga na validacionom skupu, čak i sa proširivanjem skupa. Ono što je bilo zavaravajuće je što su neki parametri (batch_size, L1 i L2 kernel regularizacija i dropout) konzistentno davali mala poboljšanja. Time je stvoren privid da sa određenom kombinacijom parametara možda može da se dođe do zaključka šta presudno utiče na ponašanje mreže. Međutim svaki trening se završavao overfittingom i podešavanje parametara je manje-više samo odlagalo epohu u kojoj se overfitting dešava.

Ono što je testirano posebno na ovoj mreži je uticaj položaja Batch Normalization sloja. Pokušane su dve varijante, da se stavi

4 Naravno misli se na konkretnu primenu na ovom skupu a ne da model dobro rešava generalne probleme

pre ili posle aktivacije. Zaključak je da na ovoj arhitekturi to nije presudno uticalo na performanse.

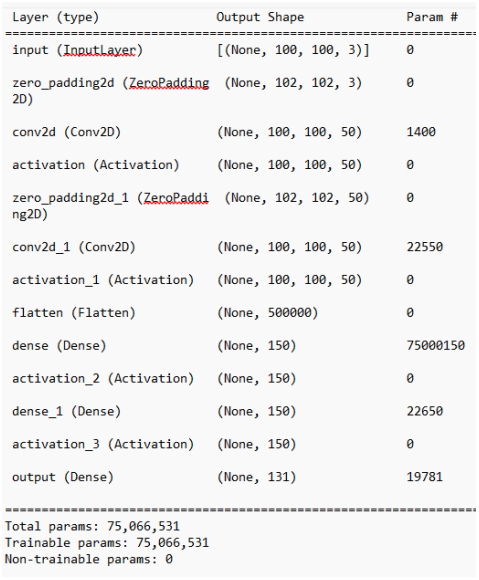
Eksperimentisano je u većoj meri sa aktivacijama nego nad prethodnim arhitekturama.

Iako ova arhitektura nije dala praktične rezultate, benefit je bio što je veća mreža dala više mogućnosti za eksperimentisanje sa parametrima i skupom, što je pomoglo u donošenju zaključaka za naknadne testove.

Conv2_2

Nakon neuspešnog eksperimenta sa D10 odlučeno je da se potpuno promeni arhitektura. U originalnom radu arhitekture počinju sa jako malo filtera u prvoj konvoluciji (uglavnom 8). U kasnijim slojevima broj filtera se povećava, ali u stepenima dvojke (16-64). Zatim slede dva ogromna sloja FC slojeva, prvi ima 1024 neurona[Mureșan, Oltean 2018].

Ovde je pokušano sa malo drugačijim pristupom. Koristeći zaključke prethodnih eksperimenata 100 neurona bi trebalo da mogu dovoljno dobro(za početak) da obrade sliku. Problem se u stvari javlja u najvećoj meri zbog toga što se mreža obučava kratko zbog overfittinga. Jedan od glavnih problem D10 je što početni slojevi moraju da budu manji od D5 i pored toga što su kasniji slojevi "jači" i treba da imaju redundansu, oni samo "pamte napamet" to malo podataka što imaju. Konvoluciona mreža će svakako smanjiti dimenzije same slike, ako se koristi stride i pool, ako se sa tim podacima stigne do kasnijih slojeva verovatno će se opet desiti isto. Zato je pokušano sa plitkom konvolucijom koja ima više filtera. Zbog lakšeg prikaza demonstriraćemo prostu mrežu koja ne koristi ni stride ni pool, da bi se videlo koliko maksimalno parametara može da ima ova mreža (slika 9):



Layer (type)	Output Shape	Param #
input (InputLayer)	[(None, 100, 100, 3)]	0
zero_padding2d (ZeroPadding2D)	(None, 102, 102, 3)	0
conv2d (Conv2D)	(None, 100, 100, 50)	1400
activation (Activation)	(None, 100, 100, 50)	0
zero_padding2d_1 (ZeroPadding2D)	(None, 102, 102, 50)	0
conv2d_1 (Conv2D)	(None, 100, 100, 50)	22550
activation_1 (Activation)	(None, 100, 100, 50)	0
flatten (Flatten)	(None, 500000)	0
dense (Dense)	(None, 150)	75000150
activation_2 (Activation)	(None, 150)	0
dense_1 (Dense)	(None, 150)	22650
activation_3 (Activation)	(None, 150)	0
output (Dense)	(None, 131)	19781

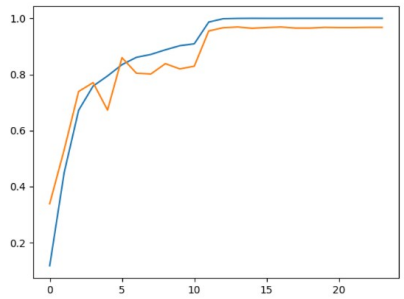
Total params: 75,066,531
Trainable params: 75,066,531
Non-trainable params: 0

Slika 9: Conv2_2_Empty

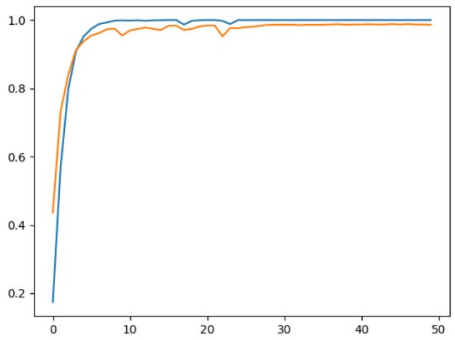
Iz ovoga se vidi potencijalni problem ove mreže. Stride je 1 u obe konvolucije filteri su 3*3. Jasno je da sa ovom veličinom slike ne može da se radi toliko fino analiziranje slike. Neophodno je da stride bude barem 2, pošto ima samo dve konvolucije, slika će i dalje biti dovoljno velika da može da se uradi MaxPool (korišćen je stride 2 veličina 2*2). Za optimizaciju je maltene tokom celog testiranja konvolucija bio korišćen Nadam. Vrlo brzo je model stigao do 90% na testu. Međutim tu su se već pokazale limitacije. Pokušano je sa dodavanjem Batch Normalization i regularizacije, međutim to nije uzrokovalo značjnim poboljšanjem performansi. Model je na validaciji pokazivao mnogo bolje rezultate nego kod prethodnih arhitektura, ali je bio problem jer se na treningu overfittuje. Doduše predviđanja su dosta konzistentna. I rezultati na validaciji su obećavajući, što znači da je rešen problem podele validacionih podataka.

Analizom rezultata došlo se do zaključka da model zapravo pravi najbolju moguću apstrakciju i da prosto nije pogodan za rešavanje nekih klasa. Zato je odlučeno da se doda još jedan sloj konvolucije. Pre toga je rađen ekperiment sa, batch size-om dropout i L2/L1 regularizacijom. Zaključeno je da osnovni model ima problem sa overfittingom (slika 10), ali da L1 regularizacija (slika 12) maltene nema nikakav efekat, dok Batch Normalization (slika 11). Doduše to je maltene izjednačilo validacionu i trening krivu, što dovodi u opasnost da i na validaciji dođe do overfittinga. Da bi se smanjila šansa da dođe do overffitinga uvedena je L2 regularizacija (slika 12). Dupliranje Batch-size je uticao da model malo uspori treniranje da bi u kasnijoj fazi mogao

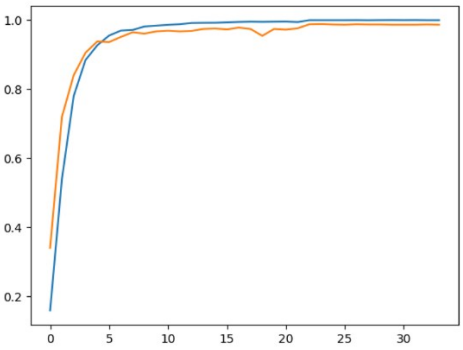
da smanji learning rate. Na ovoj arhitekturi nije bilo značajnog pomaka u krajnjem rezultatu, ali se vidi da za razliku od L1 ima efekta, pa je odlučeno da se nastavi sa testiranjem L2 i na drugoj arhitekturi. U nastavku prikaz accuracy metrike pri treniranju (slike 10-14):



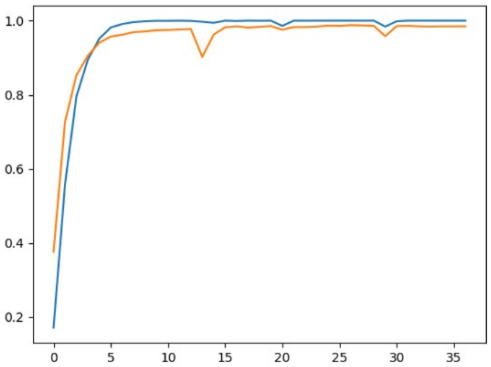
Slika 11: Konvolucija sa MaxPool



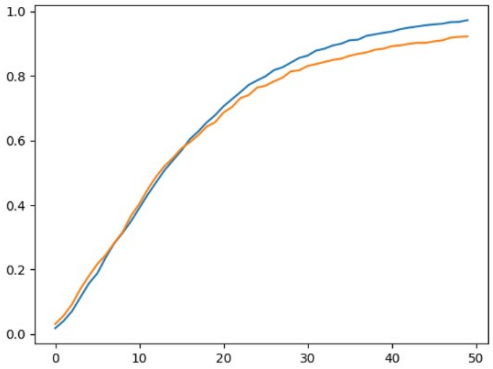
Slika 10: Dodavanje BN



Slika 13: Dodavanje L1 regularizacije



Slika 12: Dodavanje L2, za razliku od L1 utiče na oba seta, kod L1 smanjenje validacije nije jasno da li je uticaj regularizacije



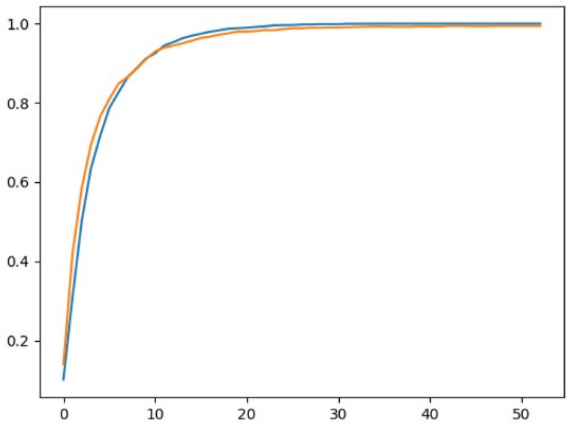
Slika 14: Efekat dupliranja veličine batch-a. Značajno smanjena šansa za overfitting na validacionom skupu (koji je bio glavnin problem MPL)

Conv2_3

Iz prethodnog primera se pokazalo da menjanje parametara modela ima dosta logičnije posledice, nego menjanje parametara MLP mreža na ovom skupu podataka. Iz tog razloga odlučeno je da neki parametri ostanu isti nakon dodavanja mreže. Primećene su neke greške kod padding-a. Sloj koji je dodat ima više filtera (80), veličine 3*3. Da se ne bi smanjila slika (s obzirom da je posle MaxPool već mala slika) stride je 1. Promena rezultata je bila vidljiva, međutim nije bila mnogo velika. Nakon popravljanja paddinga, promena aktivacije u selu (to je vidljivo popravilo rezultate u odnosu na leaky-relu) i dodavanja još jednog sloja BN, model je postigao 93,4%, što je začuđujuće i dalje slabije od najbolje D5 mreže. Ti rezultati nisu bili prihvatljivi.

Dodatnom analizom uočena su dva propusta. Prvi propust je korišćenje MaxPool na slici koja je jako male veličine. Logičnije je koristiti AveragePool, pogotovo sa stride 2 Max Pool može da ignoriše dosta feature-a. Drugi propust je da je mreža i dalje u zadnjim slojevima bila veličine MLP po broju neurona iako ukupno ima mnogo manje parametara i objektivno ima prostora za pojačanje.

Nakon otklanjanja prvog propusta mreža je nadmašila neke modele iz originalnog rada i sve modele ovog ekperimenta. U dva FC sloja je dodato 50 neurona (u oba je sada 150). Ovo je očigledno usporavalo i prošlu i ovu arhitekturu. Mreža se mnogo brže trenirala i postigla je skoro maksimalni skor na treningu, bez overfittinga (slika 15):



Slika 15: Zadnjih 30 epoha mreža polako povećava Accuracy

Rezultat evaluacije je preko 96,2% što premašuje originalni rezultat za skoro 1%, pri čemu treba napomenuti da je ovo trening sa znatno manjom količinom podataka i neuporedivo manjim brojem epoha u odnosu na originalni model. Trening na celom skupu je poboljšao model i podigao tačnost na nešto više od 97%.

Zaključak

Eksperimentalno je pokazano da je moguće napraviti light-weight mrežu koja ostvaruje bolje performanse od originalnog modela. Kako brzo napreduje ova grana sada je već proširen skup podataka i može se reći da ovi rezultati nisu toliko aktuelni. Resnet i VGG lako premašuju performanse mreža o kojima je bila diskusija, postižući maltene savršenu tačnost. Radovi na ovu temu sada već primenjuju transfer-learning sa drugih robustnijih modela na deo podataka sa [Fruit-360 repository] koji je vezan za otkrivanje instanci. Pokazalo se da jako uspešno može da se primeni mask ili faster-RCNN. Ovaj projekat je imao za cilj da demonstrira osnovne tehnike dubokog učenja, na datasetu koji u današnje vreme nije najpogodniji za treniranje u sve svrhe, ali je za demonstrativne svrhe jako pogodan.

Reference

Literatura

[Mureșan, Oltean 2018]: Horea Mureșan, Mihai Oltean, Fruit recognition from images using deep learning, Acta Univ. Sapientiae, Informatica, 2018, Vol. 10, Issue 1, pp 26-42
<DLv05>: Aleksandar Milosavljević, Data augmentation and Fine-tuning, , https://cs.elfak.ni.ac.rs/nastava/mod/url/view.php?id=9867, arhiva: , pristupljeno:
<Fruit-360 repository>: Mihai Oltean, , 2018, https://github.com/fruits-360/fruits-360-100x100, arhiva: , pristupljeno:
<jubilant-spoon>: Lazar Đorđević, Repozitorijum projekta, , https://github.com/djordjeviclazar/jubilant-spoon, arhiva: , pristupljeno:
<Keras-doc>: François Chollet, Keras Api documentation v 2, , https://keras.io/2/api/, arhiva: , pristupljeno:
<Tf-doc>: Google, Tensorflow API documentation v 2.10.1, , https://www.tensorflow.org/versions/r2.10/api_docs/python/tf, arhiva: , pristupljeno:
<Tf>: Google, Tensorflow, , https://www.tensorflow.org/, arhiva: , pristupljeno: