**Факултет инжењерских наука Универзитета у Крагујевцу**

|  |  |
| --- | --- |
|  |  |

**Назив студијског програма: Рачунарска техника и софтверско инжењерство**

**Ниво студија: Основне академске студије**

**Модул: Програмски преводиоци**

**Предмет: Програмски преводиоци**

**Број индекса: 579/2015**

**Ленка В. Милић**

**Предвиђање биљних болести биљке малине употребом конволуција у дубоким неуронским мрежама**

**Дипломски рад**

|  |  |
| --- | --- |
| **Комисија за преглед и одбрану:**   1. **Др. Владимир Миловановић -** **ментор** 2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 3. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Датум одбране:\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Оцена:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

У оквиру овог завршног рада кандидат треба да

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Препоручена литература:

[1] ...............................................

[2] ...............................................

[3] ...............................................

...

Крагујевац, датум ментор:

Владимир Миловановић Др.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Резиме*

Рад је посвећен малим фармерима на подручју централне и западне Србије који на својим парцелама поседују узгоје биљке малине. „Ова биљка је изложена патогенима као што су вируси, бактерије и гљивице због чега су потребне адекватне мере како би биљка била заштићена или излечена. Биљне болести узроковане патогенима узрокују значајне губитке приноса заседа у Србији која је на свету једна од највећих произвођача и извозника овог воћа (1).” Стога je сваки допринос овој пољопривредној грани од јаке важности. Сврха овог рада је приказ примене дубоког учења у класификацији биљних болести малине према сликама заражених листова, стабљика и плодова. Техником дубоког учења креира се неуронска мрежа чији се параметри тренирају до оптималних вредности. Резултат неуронске мреже на основу једног улазног податка је предвиђена биљна болест малине, односно здрава малина. Овакав закључак може бити искоришћен зарад примене одговарајућих мера као што је орезивање, коришћење хемијских препарата (и слично) како би се спречиo даљe развијање биљне болести.

*Кључне речи*

*Неуронска мрежа, параметри, мини серија улазних и излазних података, конволуција, потпуно повезана мрежа, пропагација, активациона функција, линеарна функција трошка параметара, трошак параметара, тренирање параметара.*

*Summary*

The paper is dedicated to small farmers in central and western Serbia who own raspberry plants on their plots. „This plant is exposed to pathogens such as viruses, bacteria and fungi, so adequate measures are needed to protect or cure the plant. Plant diseases caused by pathogens cause significant losses of yields in Serbia, which is one of the largest producers and exporters of this fruit in the world (1).” Therefore, any contribution to this agricultural branch is of great importance. The purpose of this paper is to demonstrate the application of deep learning in the classification of raspberry plant diseases according to images of infected leaves, stems and fruits. A deep learning technique creates a neural network whose parameters are trained to optimal values. The result of the neural network, based on one input, is the predicted herbal raspberry disease, that is, healthy plant raspberry. Such a conclusion can be used to implement appropriate measures such as pruning, the use of chemical preparations (and the like) to prevent the further development of plant disease.

*Key words*

*Neural network, parameters, mini-series of input and output data, convolution, fully connected network, propagation, activation function, linear parameter cost function, parameter cost, parameter training.*

Садржај

**1. Увод**……………………………………………………………………………………………….**3**

**2. Потребни софтвери и библиотеке**………………………………………………………………**4**

**3. Обрада и припрема података**…………………………………………………………………..**6**

**4. Архитектура неуронске мреже**…………………………………………………………………**9**

4. 1. Активационе функције и функција предвиђања…………………………………………...**11**

4. 2. Параметри неуронске мреже………………………………………………………………**13**

**5. Модел неуронске мреже**………………………………………………………………………**14**

5. 1. Иницијализација параметара / Дефинисање параметара………………………………...**14**

5. 2. Иницијализација оптимизатора алгоритма Адам…………………………………………**16**

5. 3. Петља кроз број епоха……………………………………………………………………...**18**

5. 3. 1. Избор мини-серија улазних и излазних података………………………………………**18**

5. 3. 2. Петља за сваку мини-серију података………………………………………………….**19**

5. 3. 3. Пропагација унапред…………………………………………………………………..**22**

5. 3. 3. 1. Конволуцијска мрежа………………………………………………………………**22**

5. 3. 3. 2. Креирање 2D улаза у потпуно повезану мрежу……………………………………...**27**

5. 3. 3. 3. Потпуно повезана мрежа…………………………………………………………..**27**

5. 3. 4. Рачунање трошка параметара………………………………………………………….**30**

5. 3. 5. Пропагација уназад…………………………………………………………………….**30**

5. 3. 5. 1. Потпуно повезана мрежа………………………………………...………………...**31**

5. 3. 5. 2. Конволуцијска мрежа……………………………………………………………...**34**

5. 3. 6. Update-овање параметара……………………………………………………………..**36**

**6. Функција модела неуронске мреже**………………………………………………………….**38**

**7. Тренирање и резултати неуронске мреже**……………………………………………………**40**

**8. Закључак**………………………………………………………………………………………...**43**

**9. Литература**………………………………………………………………………………………**44**

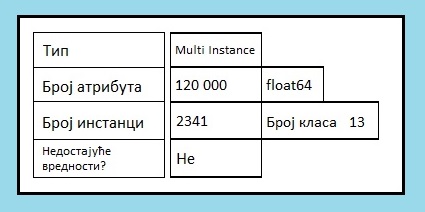
1. Увод

Задатак рада је да обухвати тринаест стања биљке малине од којих прво стање описује здраву биљку малине, а сва остала упућују на неку специфичну болест, вирус или стање које је узроковано малинином мушицом:

* Здрава малина
* Антракоза малине (*Anthracnose,* енгл*.*)
* Бактериозни рак - (*Agrobacterium tumefaciens*, лат.)
* Сива трулеж (*Gray mold*, енгл.)
* Бактериозна пламењача (*Erwinia amylovora,* лат*.*)
* Стање малине узроковано малинином мушицом
* Наранџаста рђа (*Orange rust,* енгл*.*)
* Пламењача корена малине (*Phytophthora,* енгл*.*)
* Љубичаста пегавост / Сушење изданка малине (*Didymella applanata* / *Cane blight,* енгл.)
* Вирус патуљасти жбун – RBDV (*Raspberry Bushy dwarf virus,* енгл*.*)
* Вирус тачкастог листа (*Leaf spot virus*, енгл.)
* Мозаик вирус (*Mosaic virus,* енгл*.*)
* Вирус коју узрокује коврџање листа малине (*Leaf curl virus,* енгл*.*)

Улазне параметре у мрежи (*атрибути*, слика 1), чине вредности пиксела слика заражених листова, стабљика и плодова. Укупан број атрибута је 120 хиљада за сваку од инстанци на улазу. Слике су тродимензионе, RGB слике облика (200, 200, 3).

Број инстанци који дефинише укупан број слика у проблему *предвиђања биљних болести малине* је 2341.

Излаз неуронске мреже су бинарне вредности према чему је једна од вредности излазног вектора 1 и дефинише установљену биљну болест малине или здраву биљку малине, а преосталих 12 вредности на излазу су нуле и негирају постојање преосталих стања. Класификација одређеног стања регулисана је нумерички и односи се на позицију неурона у последњем слоју неуронске мреже.

*Слика 1 :*

*Опис скупа података*

2. Потребни софтвери и библиотеке

*Софтвер:*

- Anaconda (платформа за Python науку о подацима)

*Библиотеке:*

- numpy

- scikit-learn

- matplotlib

- PIL

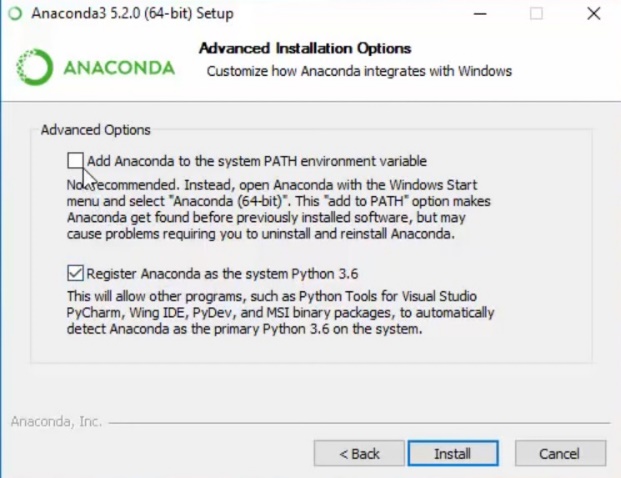
- math

- io

- jupyter notebook

Техника дубоког учења (или хијерархијског учења) базира се на учењу репрезентације података. Величина улазног скупа података неуронске мреже може достићи велике размере при чему време извршавања програма треба остати ефективно. Управо из тог разлога за проблем *предвиђања биљних болести малине* коришћена је Python библиотека *numpy*. Ова библиотека служи за брзе компутације између структура података које садрже велики број елемената. Улазни скуп података неуронске мреже представљен је као једна таква структура, односно као *n*-димензиони *numpy* вектор.

Компутације се на нивоу векторских структура имплементирају помоћу векторизације и *broadcast*-инга коју омогућава ова библиотека. Уметност векторизације у Python-у омогућава процесуирање *n*-димензионих *numpy* вектора без коришћења *foor* петљи. Коришћење споре *foor* петље у области дубоког учења је јако непожељно јер успорава извршавање програма односно тренирање параметара неуронске мреже. *Numpy* операције над *n*-димензионим *numpy* векторима омогућавају и до 300 пута мање време које је потребно за извршавања програма од оног времена извршавања када би се уместо операција овебиблиотеке користила *foor* петља зарад истог резултата.

Проблем *предвиђања биљних болести малине* имплементиран је коришћењем Anaconda 2019 платформе. Ова платформа се користи за развијање, тестирање и тренинг и омогућава брзо преузимање python библиотека, анализу података, визуализацију резултата, итд.

На следећем веб линку могуће је преузети једну од Anaconda дистрибуција :

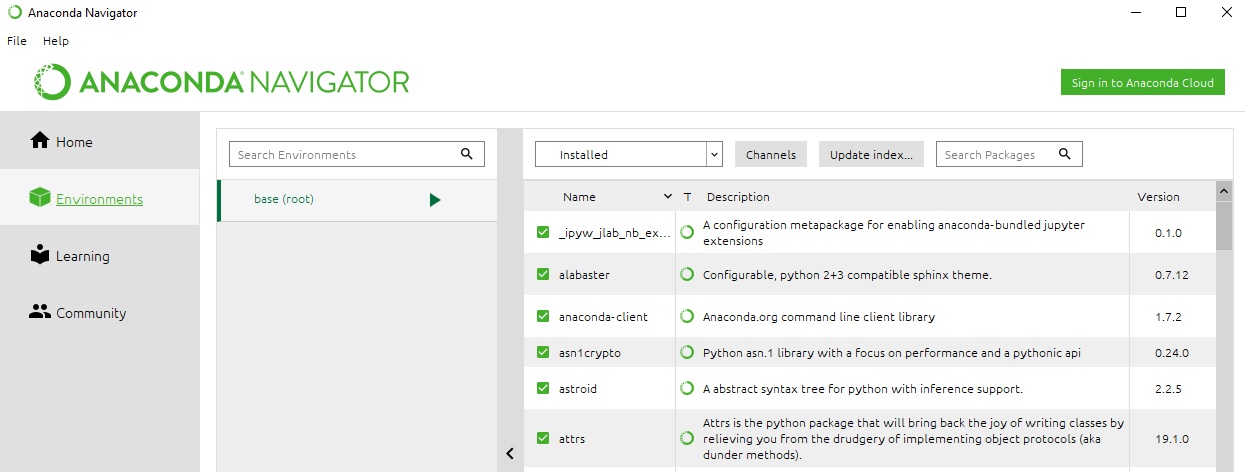
<https://www.anaconda.com/distribution/>

У току инсталације квадрат поред описа *«Add Anaconda to the system PATH environment variable»* сене селектује уколико на рачунару већ постоји инсталиран Python (слика 2).

*Слика 2:*

*Инсталација Anaconda 2019 дистрибуције*

Након инсталације софтвера у Windows старт менију избором Anaconda navigator-а приказује се прозор као на слици 3 и дифолт окружење *base (root).*



*Слика 3 :*

*Преглед окружења Anaconda navigator*

Библиотеке је могуће инсталирати у оквиру окружења Anaconda navigator (преглед инсталираних библиотека: Environments → base(root) окружење или виртуално окружење уколико је креирано → Installed ) или у Anaconda конзоли.

Инсталирање пакета у оквиру Anaconda конзоле :

* conda install -c anaconda numpy
* conda install -c anaconda scikit-learn
* conda update scikit-learn
* conda install -c conda-forge matplotlib
* conda install -c anaconda pil
* conda install -c anaconda jupyter

Радно окружење јупитерова свеска (jupyter notebook, енгл.) је веб апликација отвореног кода која омогућава интерактиван код, једначине, визуализацију и наративан текст на једном месту.

Покретање радног окружења, у оквиру локалног сервера на рачунару, у Anaconda конзоли :

* jupyter notebook

Потребне библиотеке су учитане на почетку свеске као на слици 4

Покретање програма *“raspberries\_proj.ipynb”* се извршава :

* кликом на опцију Run (*једна по једна линија*) или
* Kernel → Restart and Run All (*све линије*)

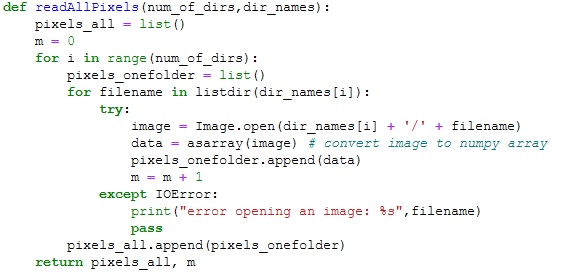


*Слика 4 :*

*Учитавање свих библиотека у оквиру јупитерове свеске и подешавање параметара за штампање графика*

3. Обрада и припрема података

Улазни подаци неуронске мреже су вредности пиксела слика у опсегу од 0 до 255. Преузимање вредности пиксела слика из свих под-директоријума директоријума *„pics”* омогућава функција *readAllPixels* која преузима укупан број под-директоријума, *num\_of\_dirs* и имена свих под-директоријума дефинисаних у листи *dir\_names* (слика 5). Ова функција враћа укупан број успешно учитаних слика *m* и листу *pixels* која садржи тачно тринаест под-листа. Ових тринаест под-листа појединачно садрже вредности пиксела слика које се односе на одређену биљну болест малине (слика 6).



*Слика 5 :*

*Учитавање вредности пиксела и пребројавање укупног броја слика*



*Слика 6:*

*Складиштење вредности пиксела и укупног броја слика m*

*Numpy* вектор *x* садржи све улазне податке у облику вектора димензије (m, 200, 200, 3). Излазни подаци представљени су вектором *y* облика (m, num\_of\_dirs) и креирани су према распореду вредности пиксела у листи *pixels* (слика 7).

Излазни вектор за скупове вредности пиксела који се класифицирају као

прво стање биљке малине (стање здраве малине) је :

[1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.],

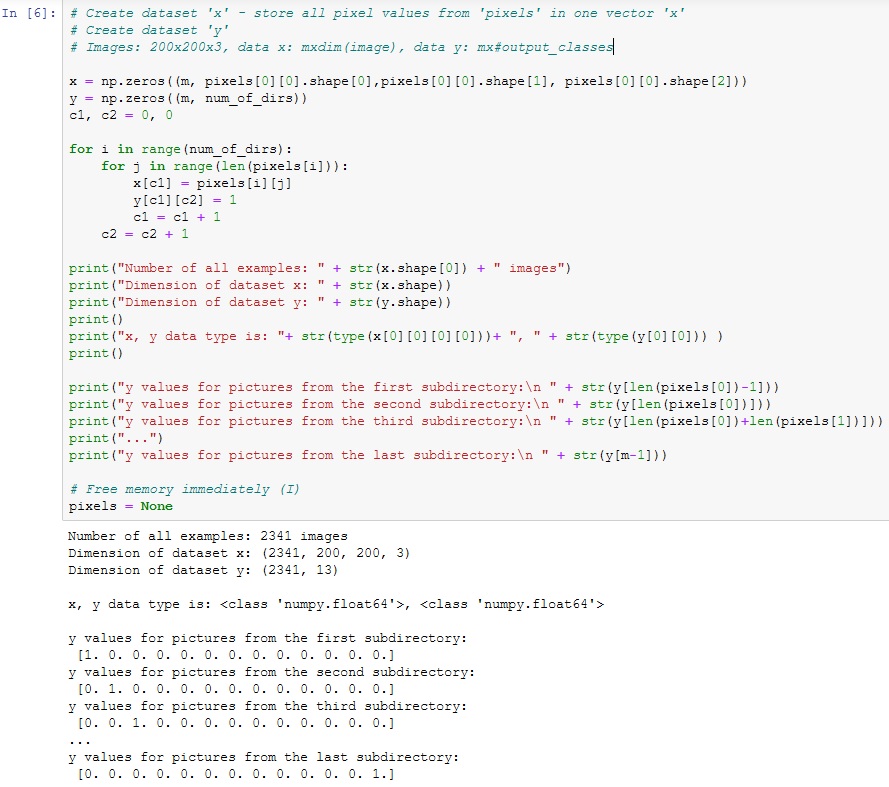
друго стање биљке малине (биљна болест Антракоза) је :

[0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.],

...

последње, тринаесто стање биљке малине (вирус коврџања листа малине ) је :

[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]



*Слика 7 :*

*Креирање numpy вектора за сет свих података, x и y*

За тренинг параметара неуронске мреже из скупа *x* и *y* изабрано је 80 посто података, а за тестирање параметара неуронске мреже изабрано је преосталих 20 посто. Функција *train\_test\_split* која омогућава раздвајање података на тренинг и тест скуп налази се у пакету *sklearn.model\_selection* (слика 4).

Тренинг подаци су *numpy* вектори *x\_train* и *y\_train.*

Тест подаци су *numpy* вектори *x\_test* и *y\_test*.

Преглед броја тренинг/тест примера, димензија ових вектора и укупног броја улазних података мреже налазe се на слици 8.



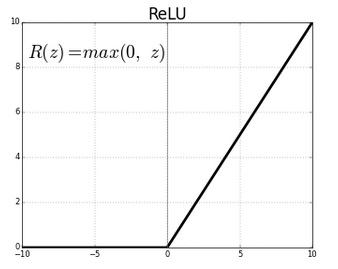
*Слика 8 :*

*Преглед података који ће се користити за тренинг и тест неуронске мреже*

4. Архитектура неуронске мреже

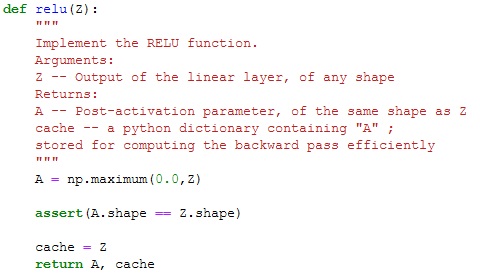
Модел неуронске мреже чине два конволуцијска слоја (*Conv -> ReLu -> MaxPooL*), десет потпуно повезана слоја (*Fc*) и један излазни слој.

Активациона функција која је коришћена у скривеним слојевима је ***ReLu*** *(Rectified linear unit)* нелинеарна функција (слика 9 и 10). У излазном слоју активациона функција је ***Softmax*** нелинеарна функција (слика 11 и 12).



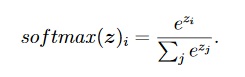
*Слика 9 :*

*ReLu активациона функција*



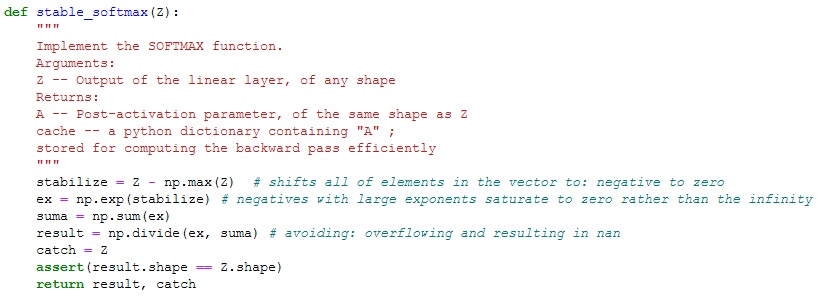
*Слика 10 :*

*Имплементација ReLu активационе функције скривених слојева*



*Слика 11 :*

*Softmax активациона функција*



*Слика 12 :*

*Имплементација Softmax активационе функције излазног слоја*

*Зашто су избор управо ове активационе функције слојева?*

4. 1. Активационе функције и функција предвиђања

У скривеним слојевима ***ReLu*** функција има бољу перформансу конвергенције у односу на сигмоидалну функцију скривених слојева, .

Такође временски је ефективнија јер само одређује максимум вредност између улазне вредности *X* и нуле. Главна предност функције ***ReLu*** је мања вероватноћа да дође до нестајања градијената (нула вредности градијената) или експлодирања градијената (јако велике вредности градијената) у једном од слојева неуронске мреже.

Линеарна компутација пре нелинеарне активације у једном слоју израчунава се према формули:

„ Међутим уколико је активациона функција линеарна , а вектор *b* изједначен са нула вектором онда je предвиђање на излазу неуронске мреже

једнако следећем изразу:

где је нумеричка вредност последњег слоја и

је или активациона вредност

*\* \* X* је или активациона вредност , итд.

Уколико је сваки вектор за из опсега, изједначена са вектором

|  |  |
| --- | --- |
| *0.5* | *0* |
| *0* | *0.5* |

важи следеће :

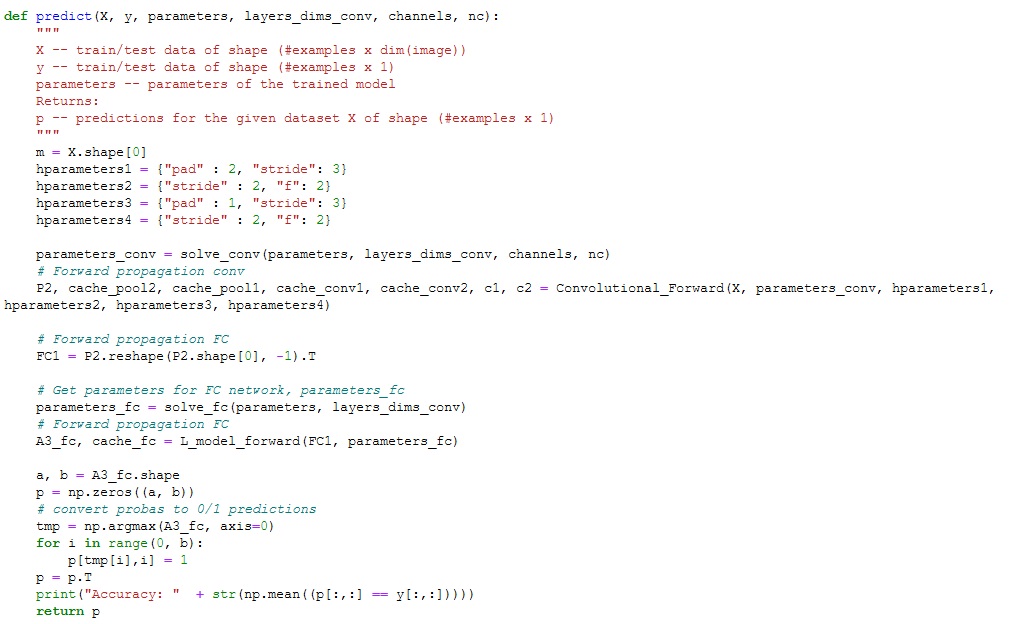
(2) ”

Израз доказује чињеницу да ће предвиђање имати степен опадања.

Уколико су параметри < (јединична матрица) за дубоке неуронске мреже вредност активационе функције ће се експоненцијално смањивати, односно расти за 0.5 → 1.5 ( степен раста и > ). У првом случају вредности активационих функција ће тежити нестајању тј. нули, а у другом случају експлодирању тј. бесконачно. Самим тим и њихови градијенти ће бити мањи, односно већи, како учење напредује дубље унутар неуронске мреже. Како вредности ових параметара зависе од дубине мреже на иницијализацију вектора у слоју треба да утиче број неурона у слоју . Стога уколико је варијанса параметара грубо око нуле, дефинисана изразом  *,* где је број неурона у слоју, а варијанса вредности ***ReLu*** активационе функције подразумевано 1, онда ће и предвиђањебити у том опсегу. На овај начин могуће је редуковање проблема нестајања/експлодирања градијента, али не и искорење овог проблема. У поглављу *5.1* имплементирана је иницијализација параметара за потпуно повезану мрежу на начин који је представљен овде.

У логистичком регресијском моделу ***Softmax*** функција се користи за мулти-класификацију вредности излазног слоја неуронске мреже. Збир вероватноћа вредности на које је примењен ***Softmax*** је један. Максимална вредност у колони излазног вектора облика (m, 13), која припада опсегу (0, 1), за једну инстанцу на улазу представља предвиђену биљну болест за ту инстанцу и биће претворена у вредност 1.

Упоређивањем вредности које је неуронска мрежа предвидела за тренинг/тест скуп података и реалних вредности излазног скупа одређује се прецизност неуронске мреже на овим скуповима. Предвиђање у неуронској мрежи имплементира се као један пролазак кроз све слојеве мреже (слика 13), док је тренинг заснован на узастопним проласцима у циљу оптимизовања параметара слојева неуронске мреже. Тренинг параметара прати и назадан пролазак (пропагација уназад) о коме ће речи бити касније.



*Слика 13 :*

*Предвиђање на тренинг и тест скупу података и одређивање тачности мреже на ова два скупа*

4. 2. Параметри неуронске мреже

Слојеви неуронске мреже садрже велики број параметара који учествују у креирању комплексне границе одлучивања. Граница одлучивања класификује улазне податке према излазима система тј. стањима биљке малине. Параметри неуронске мреже који припадају слоју преузимају информације од слоја и учествују у линераној компутацији и нелинеарној активацији према чему граница одлучивања добија комплекснији изглед.

Број слојева у конволуцијском делу неуронске мреже је два према чему постоје два филтера при операцији конволуције. Филтери у конволуцијском делу неуронске мреже представљају *numpy* *n*-димензионе векторе. Овакви вектори постоје и у потпуно повезаном делу неуронске мреже. Димензије филтера при операцији конволуције су редом (5, 5, 3, 4) и (3, 3, 4, 8). Прве три димензије се односе на ширину, висину и дубину и филтера, а трећа димензија на то колико је оваквих филтера употребљено у конволуцијској операцији.

Број слојева у потпуно повезаном делу мреже као и број параметара који припадају овим слојевима (без излазног слоја) је десет. Број неурона у скривеним слојевима потпуно повезане мреже је редом 447, 774, 1094, 631, 315, 181, 127, 73, 36, 20. Њихове димензије зависе од броја неурона у суседним слојевима. Због овакве конфигурације мреже параметар који припада првом скривеном слоју је облика (447, 200), други је облика (774, 447), трећи је облика (1094, 774), … , и последњи је облика (20, 36). Излаз конволуцијске мреже је излаз последњег *pooling* слоја димензије (m, 5, 5, 8). Овај 4-димензиони вектор обликован је у вектор димензије (200, m) и он представља улазни вектор у потпуно повезану мрежу. На овај начин је установљен број улазних веза које припадају првом параметру потпуно повезане мреже (→200). Унутар конволуцијске мреже параметри су 4-димензиони, док у потпуно повезаној мрежи 2-димензиони, због чега је било потребно обликовање ових вектора при преласку са конволуцијске на потпуно повезану мрежу.

Параметар последњег излазног слоја је облика (13,20) и једино његове вредности учествују у ***Softmax*** активационој функцији.

5. Модел неуронске мреже

*Генерална методологија мреже :*

1. Иницијализација параметара / Дефинисање параметара
2. Иницијализација оптимизатора Адам алгоритма
3. Петља кроз број епоха

3. 1. Избор мини-серија улазних и излазних података

3. 2. Петља за сваку мини-серију података

3. 3. Пропагација унапред

3. 3. 1. Конволуцијска мрежа

3. 3. 2. Креирање 2D улаза у потпуно повезану мрежу

3. 3. 3. Потпуно повезана мрежа

3. 4. Рачунање трошка параметара

3. 5. Пропагација уназад

3. 5. 1. Потпуно повезана мрежа

3. 5. 2. Конволуцијска мрежа

3. 6. Update-овање параметара (користећи параметре и градијенте из пропагације уназад)

4. Штампање трошка параметара при свакој епохи

5. Коришћење истренираних параметара за предвиђање

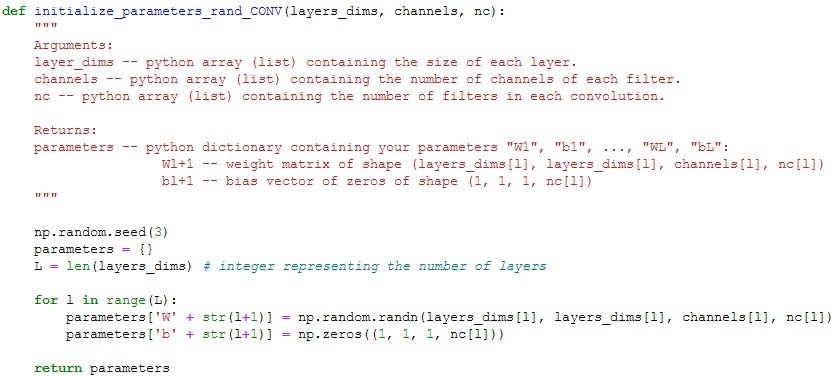
5. 1. Иницијализација параметара / Дефинисање параметара

На слици 14 и 15 приказане су функције иницијализације параметара за конволуцијски део и потпуно повезани део неуронске мреже.

Слика 14 приказује рандом иницијализацију параметара конволуцијске мреже. Слика 15 приказују *«he»* иницијализацију параметара која се показала најбољом за потпуно повезане, дубоке мреже. Ова иницијализациона техника омогућава иницијализацију параметара тако да вредности параметара у једном слоју зависе од броја неурона претходног слоја. Разлог томе је што величина мреже (број неурона у скривеним слојевима) утиче на експлодирање/нестајање градијената у одређеном слоју тј. када градијенти постану нула или јако велики.

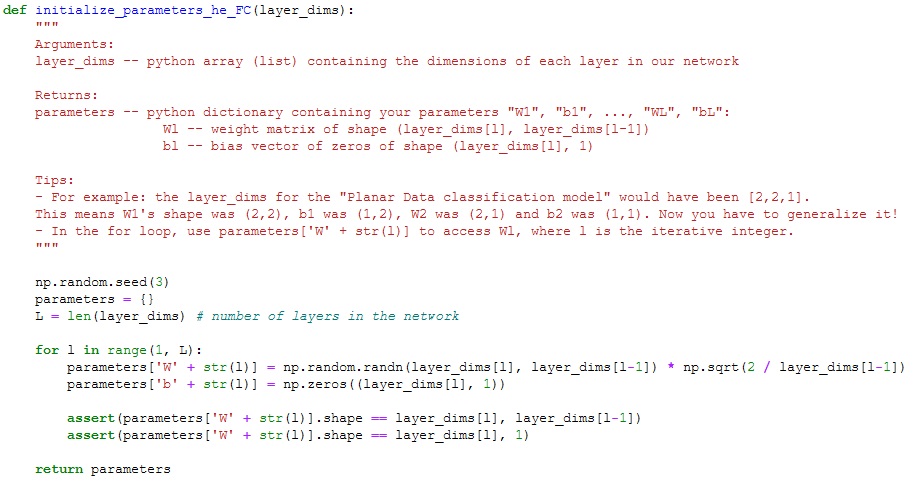
Иницијализациона техника *«he»* је имплементирана као рандом техника садодатним чланом који износи

, а који представља варијансу вредности параметара (поглавље 4.1)



*Слика 14 :*

*Random иницијализација параметара конволуцијске мреже*

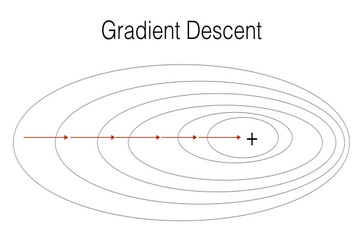


*Слика 15 :*

*«he» иницијализација параметара потпуно повезане мреже*

5. 2. Иницијализација оптимизатора Адам алгоритма

Оптимизатори су компоненте алгоритма који налази локални минимум функције трошка параметара (алгоритам „Градијентног спуста”, слика 16). Оптимизатори служе да убрзају алгоритам односно учење параметара неуронске мреже. „Адам” оптимизациони алгоритам је настао спајањем „Момент”алгоритма (*Momentum*, енгл.) и алгоритма „Kвадратни корен средњег квадрата” (*Root Mean Square prop*, енгл.) који редукују осцилације функције трошка параметара. Осцилације трошка параметара се јављају приликом процесуирања серија података (у односу на целе скупове података) у току једне епохе (поглавље 5.3).



*Слика 16 :*

*Алгоритам Градијентног спуста (Gradient Descent, енгл.)*

*Момент алгоритам*

Овај алгоритам је настао спајањем алгоритма „Мини серије Градијентног спуста” (*Mini-Batch Gradient Descent*, енгл., поглавље 5.3) и алгоритма „Eкспоненцијално отежано усредњавање” (*Exponentially weighted averages*, енгл.). Идеја је редуковати осцилације трошка параметара које алгоритам „Мини серије Градијентног спуста” прави у циљу достизања оптималне вредности функције трошка параметара. На нову вредност функције трошка параметара утиче оптимизатор *𝑣* при чему су вредности разлика хоризонталних корака близу нули. Разлике по вертикалној оси остају непромењене.

„Интуиција: параметар *𝑣*  се сматра као *«*брзина*»* лопте која се котрља низбрдо ка оптималном трошку и развија брзину (и нагиб - момент) према смеру градијента тј. «нагиба брда*»* (2).”

*Алгоритам Квадратни корен средњег квадрата*

Алгоритам који такође убрзава градијентни спуст, јер успорава учење у вертикалном смеру, а убрзава у хоризонталном смеру, употребљава се због недостатка „Момент” алгоритма који не редукује разлике у вертикалним корацима. Када ова два алгоритма раде заједно (→Адам) учење је брже за било коју структуру неуронске мреже.

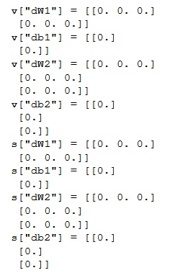
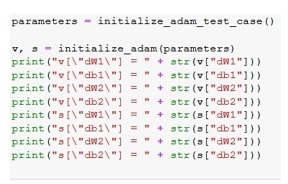
*Адам алгоритам*

„Како ради?

1. Рачуна експоненцијално отежане просеке претходних градијената и складишти их у променљиву *𝑣*  (складиштење пре корекције која користи одступање ), и израчунава *𝑣𝑐𝑜𝑟𝑟𝑒𝑐𝑡𝑒𝑑*  (са корекцијом која користи одступање)
2. Рачуна експоненцијално отежане просеке квадрата претходних градијената и складишти их у променљиву *s*  (складиштење пре корекције са одступањем ), и израчунава *s𝑐𝑜𝑟𝑟𝑒𝑐𝑡𝑒𝑑*  (са корекцијом која користи одступање )
3. Update-ује параметре у правцу заснованом на комбиновању информација из 1. и 2

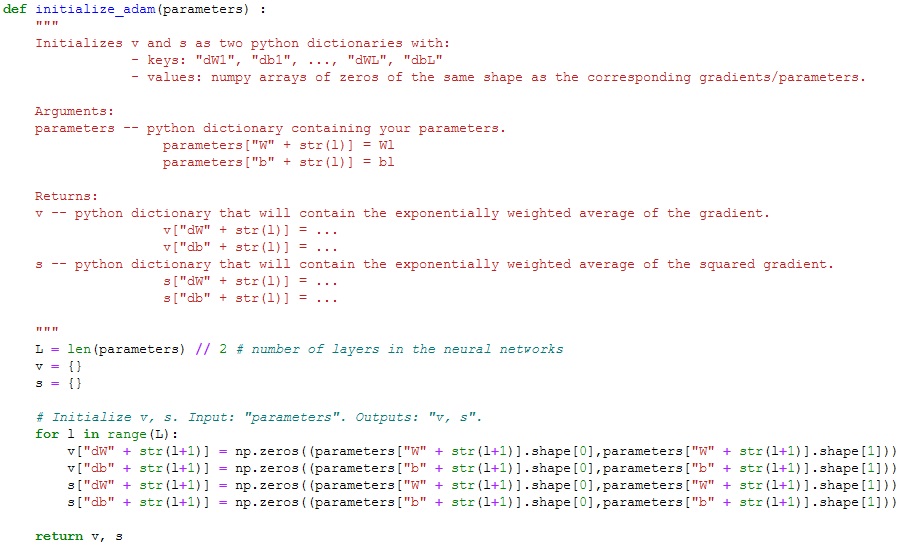
(2).”

Иницијализација Адам оптимизатора *𝑣*  и *s*  је нула-иницијализација *numpy* вектора који припадају Python-овом речнику оптимизатора. Вредности оптимизатора приказане су кроз пример (слика 17), а имплементација иницијализације ових речника приказана је на слици 18.

**

*Слика 17 :*

*Иницијализација оптимизатора 𝑣 и s*



*Слика 18 :*

*Иницијализација Адам оптимизатора*

5. 3. Петља кроз број епоха

Епоха је време које је потребно да неуронска мрежа процесуира цео сет улазних података било да су ону процесуирани одједном (*«hole batch»*) или у серијама (*«mini batch»*)*.* Пролазак кроз епохе омогућен је неизбежном *foor* петљом.

5. 3. 1. Избор мини-серија улазних и излазних података

Мини-серија улазних и излазних података односи се на серије парова улазних и излазних тренинг података *minibatch\_X и minibatch\_Y* (слика 20).

*Мини-серије Градијентног спуста алгоритам:*

Разлика алгоритма Мини-серија Градијентног спуста у односу на стандардни алгоритам Градијентног спуста је што алгоритам Градијентног спуста даје резултат вредности трошка параметара (*cost,* слика 20) тек након што процесуира цео сет података.

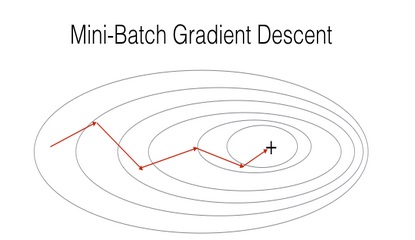
Уколико је *m* укупан број тренинг примера, односно цео сет тренинг података, а *num\_iter* број итерација, односно број епоха, овај алгоритам ће након једне итерације процесуирати цео сет тренинг података од *m* примера и направити мали корак у оптимизацији трошка (трошак ће бити оптимизован само једном).

Да би оптимизовао укупан трошак овај алгоритам ће процесуирати цео сет тренинг података (*«hole batch»*) онолико пута колики је дефинисани, максимални број итерација.

5. 3. 2. Петља за сваку мини-серију података

Уколико је *k+1* једнако укупном броју тренинг података подељеним са величином серије података, да би оптимизовао укупан трошак овај алгоритам ће процесуирати серије тренинг података *k+1* пута и понављаће ово онолико пута колики је дефинисани, максимални број итерација. Алгоритам садржи додатну *foor* петљу за процесуирање свих мини-серија тренинг података (слика 20).

Функција трошка алгоритма Градијентног спуста у времену стално опада. Функција трошка Мини-серија Градијентног спуста у времену није константно опадајућа функција, а разлог томе је што приликом рачунања функције трошка сваки пут алгоритам ради са другачијим сетом података. Трошак се update-ује након процесуирања сваке мини-серије, а не након процесуирања целог сета тренинг података. Из овог разлога функција трошка параметара садржи шум (слика 19).

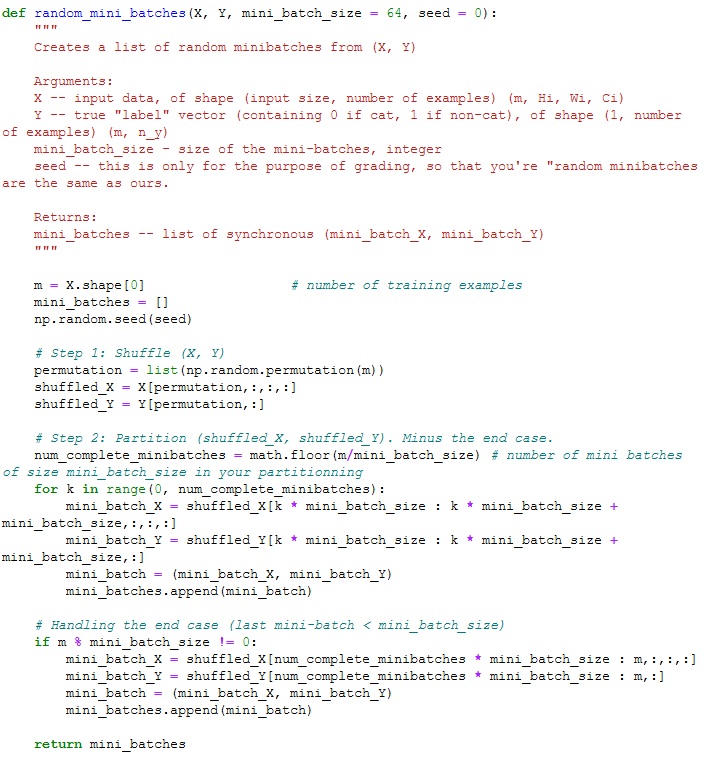


*Слика 19:*

*Алгоритам Мини-серије Градијентног спуста (Mini-batch Gradient Descent, енгл.)*

За поделу скупа тренинг података, *X* и *Y* на мини-серије података креирана је функција *random\_mini\_batches.*

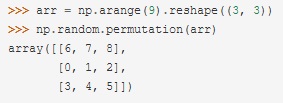
Вектори *X* и *Y* су вектори димензија редом (m, Hi, Wi, Ci) и (m, n\_y), где је *m* укупан број тренинг инстанци, *(Hi, Wi, Ci)* - димензија сваке инстанце и *n\_y* број излазних вредности за сваку од инстанци(слика 20).



*Слика 20 :*

*Имплементација функције која креира random серије тренинг података*

Како би се на рандом начин из датог скупа података изабрале мини-серије тренинг података ова функција креира листу *permutation*. Листа садржи пермутоване, целе бројеве из опсега (0, m). Пермутација је обезбеђена *numpy* функцијом *np.random.permutation* чија је примена приказана на слици 21. Векторизација у Python-у је омогућила избегавање петљи и пермутацију података у само једном реду.



*Слика 21 :*

*Приказ функције np.random.permutation*

Пермутоване вредности се односе на њима одговарајући тренинг пример. Структуре које дефинишу пермутоване податке су *shuffled*\_X и *shuffled*\_Y (слика 20).

У односу на изабрану величину мини-серију података (колико тренинг примера садржи једна серија) креира се број који описује колико таквих серија постоји. Уколико се испостави да укупна количина тренинг података *m* није дељива са величином једне мини-серије *mini\_batch\_size* онда ће све мини-серије (сем последње) имати исти број парова података, а последња серија онолико парова података колико је потребно да би се допунио број до укупног броја тренинг података (слика 22).



*Слика 22*

*Серије података тренинг примера*

5. 3. 3. Пропагација унапред

Како функционише пропагација унапред заправо тако функционише и неуронска мрежа када треба да креира предвиђање за одређени улазни податак.

Активационе функције које се користе у слојевима неуронске мреже у области неуронских мрежа не треба да буду линеарне.

*“Композиција сваке две линеарне функције је линеарна функција”*

Повећање дубине неуронске мреже тј. броја скривених слојева једна је од метода за увећање комплексности границе одлучивања. Уколико су као активационе функције изабране линеарне функције неуронска мрежа ће израчунати линеарну активациону функцију улаза и понашаће се као да нема скривене слојеве.

За границу одлучивања ово значи да се неће добро прилагодити подацима јер ће имати линеаран облик. Да би се добро прилагодила улазним подацима граница одлучивања може бити јако комплексна.

Ово је још један разлог због чега су за активационе функције изабране нелинеарне функције ***ReLu*** и ***Softmax***. Ове функције се врло добро могу прилагодити било ком сету података.

5. 3. 3. 1. Конволуцијска мрежа

Пропагација унапред кроз конволуцијску мрежу односи се на математику операцију конволуције између улазног вектора у конволуцијски слој и филтера тог слоја (слика 24). Резултат конволуције неуронске мреже је вектор чији је параметар који се односи на висину и ширину појединачне слике исти као параметар висине и ширине слике која припада улазном вектору. Трећи параметар резултујућег вектора односи се на дубину ове запремине и једнак је броју филтера који су употребљени у конволуцијској операцији. Такође број канала који дефинише дубину филтера se мора слагати са бројем канала улазног вектора.

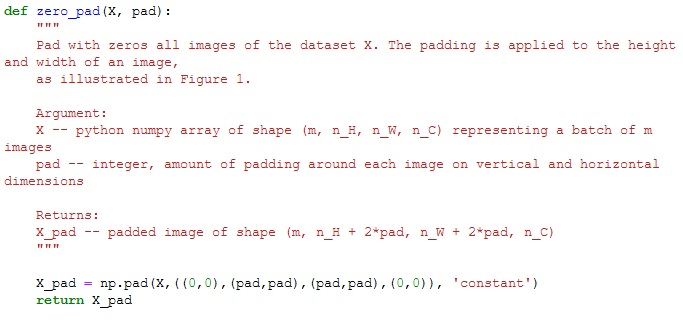
*Метод нула-подметања* (*zero-padding*)

Нула-подметање слике је техника која обезбеђује да се при конволуцији сви пиксели слике користе равномеран број пута. Неравномеран број коришћења пиксела је последица тога што се за конволуцију вредности пиксела једне слике користи вишеструка конволуција.

Једна конволуција над целом сликом могућа је уколико би филтер био истих димензија као цела слика. Тада би филтер имао параметар за сваки пиксел слике. На тај начин сваки пиксел слике у операцији конволуције би се користио само једном са одговарајућим параметром филтера.

Вишеструка конволуција односи се на ситуацију када је улазни сет пиксела целе слике подељен на делове чије су димензије (обавезно) једнаке димензији филтера. Скупови оваквих пиксела појединачно учествују у конволуцији са истим филтером у једном слоју неуронске мреже. Разлог неравномерног коришћења пиксела у конволуцији је последица тога што пиксели који припадају једном скупу могу припадати и другом скупу пиксела (исти пиксели учествују у више конволуција). Такође они пиксели који се налазе угловима слике могу се наћи само у једном скупу пиксела (пиксели по ивицама слике учествују у само једној конволуцији). Ова појава се односи на начин како су пиксели раздвајани у посебне скупове (слика 25).

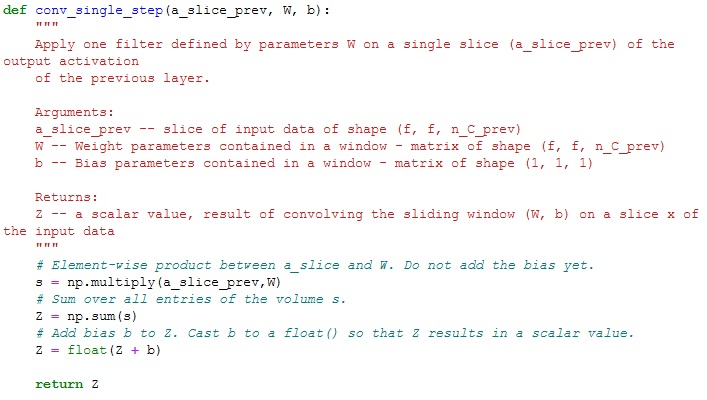
Метод *нула-подметања* омогућава додавање оквира нула-пиксела (по ивицама слике). На тај начин омогућено је равномерно коришћење пиксела слика у конволуцијским операцијама. У односу на величину параметра ове методе *pad* зависи са колико оквира нула је слика проширена (слика 23). Све слике су проширене тако да након конволуције параметар висине и ширине резултујуће слике остане исти као одговарајући параметар слике пре конволуције (*«same* техника*»*).



*Слика 23:*

*Метод Нула-подметања (zero-padding)*

Један корак конволуцијске операције приказан је на слици 24 и представља множење вредности пиксела једног дела слике и вредности елемената одабраног филтера. Множење матрица се односи на “*element -wise*“ матрично множење. Резултат једне конволуције је сума свих производа тј скаларна вредност.



*Слика 24 :*

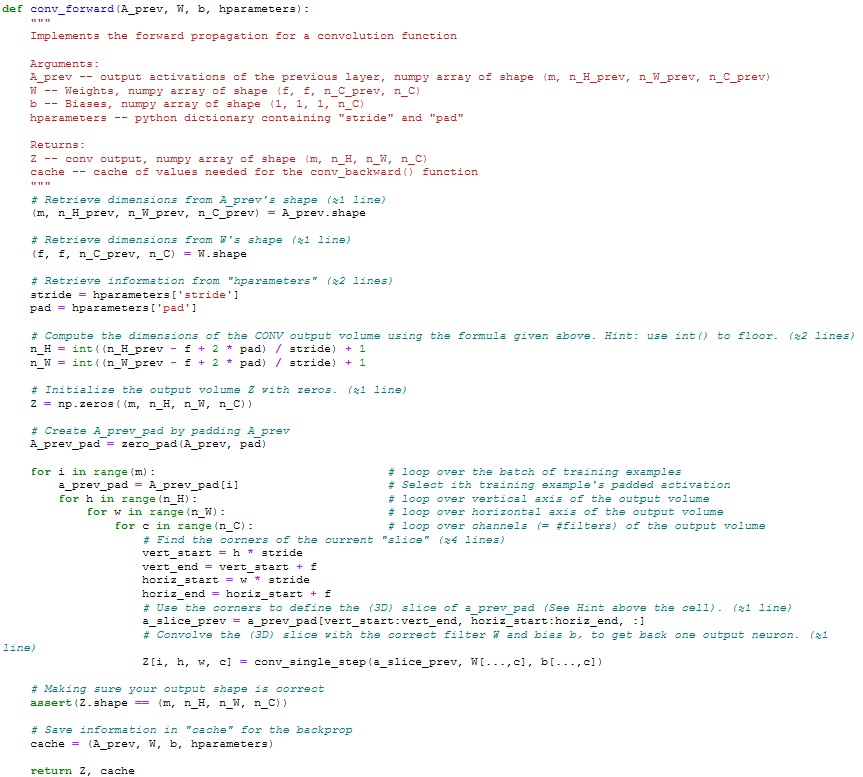
*Конволуцијска операција дела слике и филтера из одговарајућег конволуцијској слоја*

Функција која омогућава пропагацију унапред кроз један слој конволуцијске мреже прати следеће кораке:

1. Одређивање висине/ширине дела излазног вектора
2. Нула-подметање вектора
3. Петља за сваку мини-серију тренинг података
4. Селекција једног тренинг податка
5. Петља кроз број пиксела по вертикали и хоризонтали и број канала резултујућег вектора након конволуције
6. Израчунавање вредности позиције за сва четири краја тренутног дела слике
7. Креирање 3D дела слике на основу позиција сва четири краја дела слике
8. Конволуција 3D дела слике и одговарајућег филтера

Колико ће пиксела бити прескочено (на десно или доле) у креирању следећег скупа пиксела зависи од параметра ове методе *stride*. Ово је 2. параметар разматране методе.

Пропагација унапред кроз конволуцијску мрежу приказана је на слици 25.



*Слика 25 :*

*Пропагација унапред кроз конволуцијски слој*

*Pooling техника*

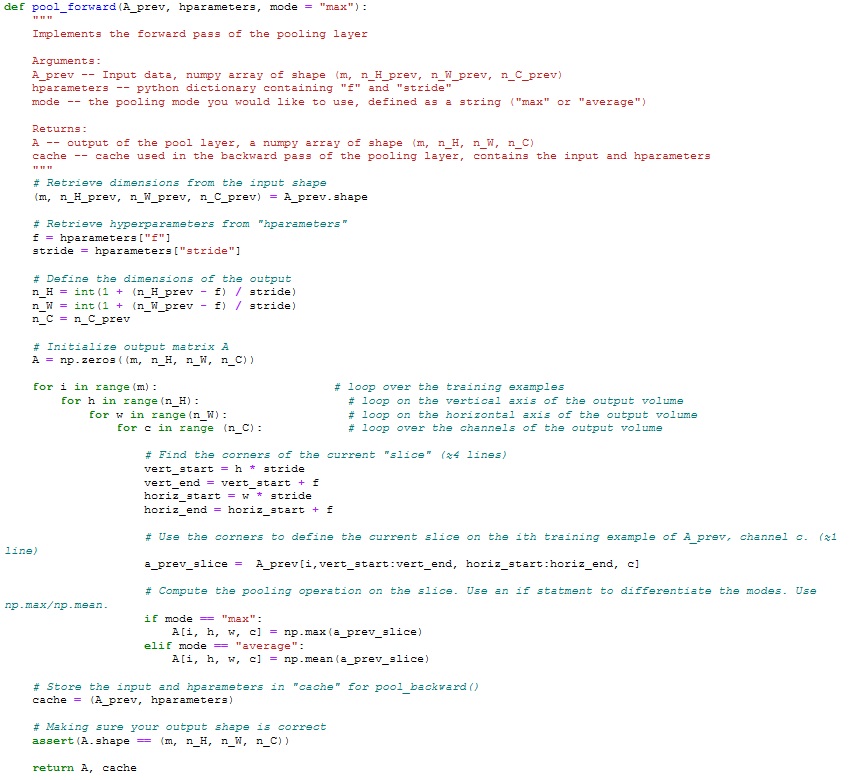
Ова техника се користи када је потребно смањење димензија вектора унутар неуронске мреже. Заправо вектор над којим се врши *pooling* техника је резултујући вектор конволуције.

Слојем неуронске мреже се сматра онај слој који у себи садржи параметре чије вредности могу да се тренирају до оптималних вредности. *Pooling* *техника* не пружа ову могућност тренирања. Она не представља посебан слој, већ се операција придружује претходном конволуцијском слоју. Стога се сматра да једном слоју конволуцијске мреже припада операција конволуције и операција коју описује *pooling* *техника*.

Оба конволуцијска слоја неуронске мреже користе *max-pooling* *технику* са параметрима *f* = *#филтера* = 2 и *s* = *stride* = 2 (постоји и *average-pooling*).

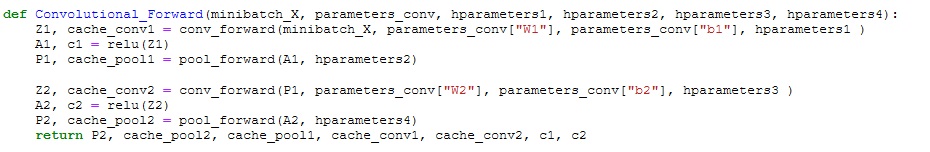
Овакви параметри осигуравају да ће прве две димензије улазног вектора, тј. висина и ширина, бити дупло мање након операције *max-pooling-а*. Такође дубина улазног вектора након *max-pooling-а* остаје иста за *stride* = 2. Пропагација унапред за *pooling* *технику* имплементирана је на сличан начин као и пропагација при конволуцији.

Операција која се користи у техници *max-pooling-*а је селекција максималног елемента у скупу пиксела једног дела слике (слика 26).



*Слика 26 :*

*Пропагација унапред и pooling техника*

**

*Слика 27 :*

*Функција која омогућава пропагацију унапред кроз два конволуцијска слоја (и pooling техника)*

5. 3. 3. 2. Креирање 2D улаза у потпуно повезану мрежу

Излаз последњег конволуцијској слоја *P2* је излазни вектор након последње примене *pooling* *технике*. Овај вектор је димензије (m, 5, 5, 8), где је *m* величина једне мини-серије података. Параметри потпуно повезане мреже су 2-димензиони вектори, стога је овај 4-димензиони вектор обликован у вектор димензије (200, m) :

*FC1 = P2.reshape(P2.shape[0], -1).T* (слика 39)

5. 3. 3. 3. Потпуно повезана мрежа

У једном неурону слоја потпуно повезане неуронске мреже се најпре извршава линеарна компутација *z = w\*a + b* као на слици 30*.* Линеарном комутацијом обезбеђује се зависност резултата мреже од параметара потпуно повезане мреже и улазног скупа података. Различите вредности параметара *w* и *b* резултирају другачијом границом одлучивања. Најоптималније вредности параметара *w* и *b* креирају границу одлучивања која се најбоље прилагођава улазним подацима и управо ове параметре алгоритам Градијентног спуста покушава да оптимизује.

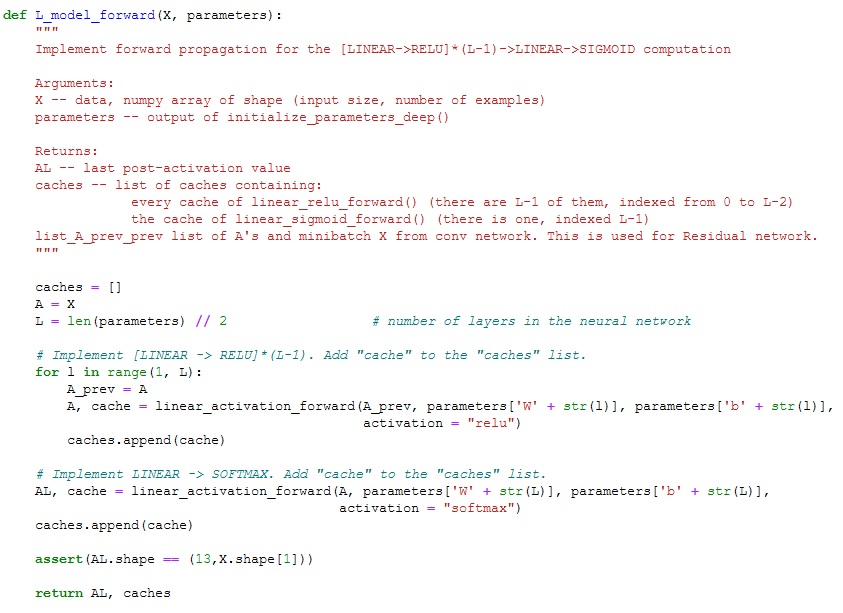
За линеарну једначину чији резултат зависи од параметaра *w* и *b* може се искористити интуиција описана једначином *l = h1\*m + h2* или

„ Дужина растегљивог канапа *l(→z)* о који је обешена лоптица масе *m(→a)*, за *m* = 0 увек је нека константна вредност *h2*(*→b*). Дужина растегљивог канапа *l(→z)* зависи од масе *m(→a)* и градијента *h1(→w),* што значи да за свако *m(→a)* дужина канапа

*l(→z)* се повећава h1(*→w*) пута. “

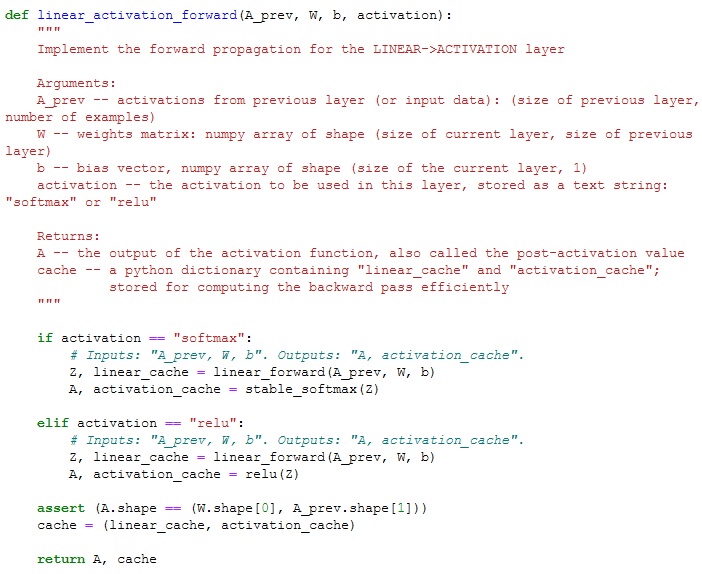
Линеарна репрезентација се затим шаље активационим функцијама ***ReLu*** или ***sigmoid.*** Активационим функцијама се обезбеђује нелинеарност модела неуронске мреже да би се избегла последица да скривени слојеви да немају утицај на креирање границе одлучивања. Пропагација унапред кроз потпуно повезану мрежу имплементирана је помоћу функција са слика 28, 29 и 30.

Имплементација активационе функције ***Softmax***, која је коришћена у пропагацији кроз потпуно повезану мрежу, изведена је на начин да одговара реалним подацима које неуронске мрежа процесуира. Тиме је обезбеђено да подаци не достигну *null* вредности (слика 12) и да не дође до прекорачења опсега вредности које припадају float64 типу података.



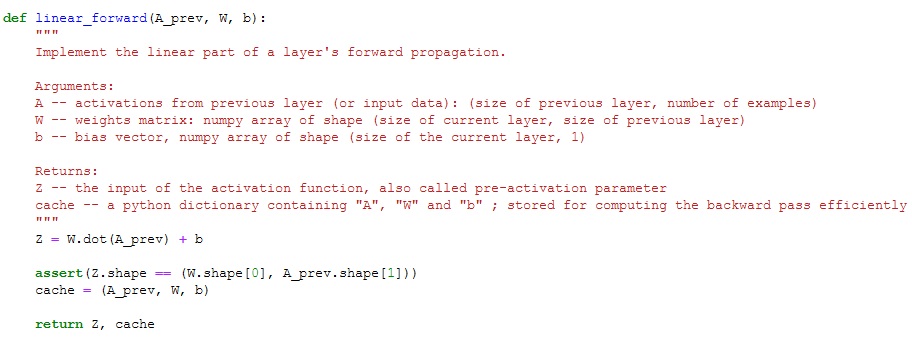
*Слика 28 :*

*Функција која омогућава пролаз кроз свих L слојева потпуно повезане мреже*



*Слика 29 :*

*Функција која омогућава избор активационе функције у слојевима и рачунање вредности активационе функције у скривеним слојевима и на излазу*



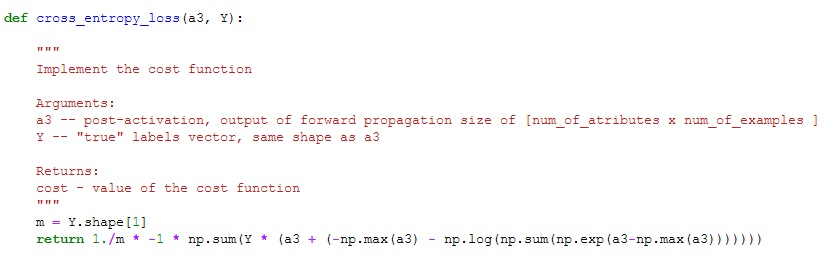
*Слика 30:*

*Функција која израчунава вредност линеарне функције пре активацијe*

5. 3. 4. Рачунање трошка параметара

Функција трошка параметра се користи за процену успешности модела. Она је мера колико је модел погрешан у погледу његове способности да процени однос између улазних и излазних података.

У односу на очекиване вредности излазног вектора избран је модел који примењује логистичку регресију. Логистичка регресија униоси зависну променљиву која се представља бинарним вредностима. То значи да би исход могао бити у било којем од два облика (0 или 1). Обзиром на вишеструки излаз (који зависи од броја стања биљке малине) примењена је ***Softmax*** функција. Функција трошка за проблем логистичке регресије са ***Softmax*** функцијом се назива и *“Cross-validation” или“Cross-entropy-Loss”* функција. Функција којом се израчунава трошак логистичке регресије приказана је на слици 31.



*Слика 31 :*

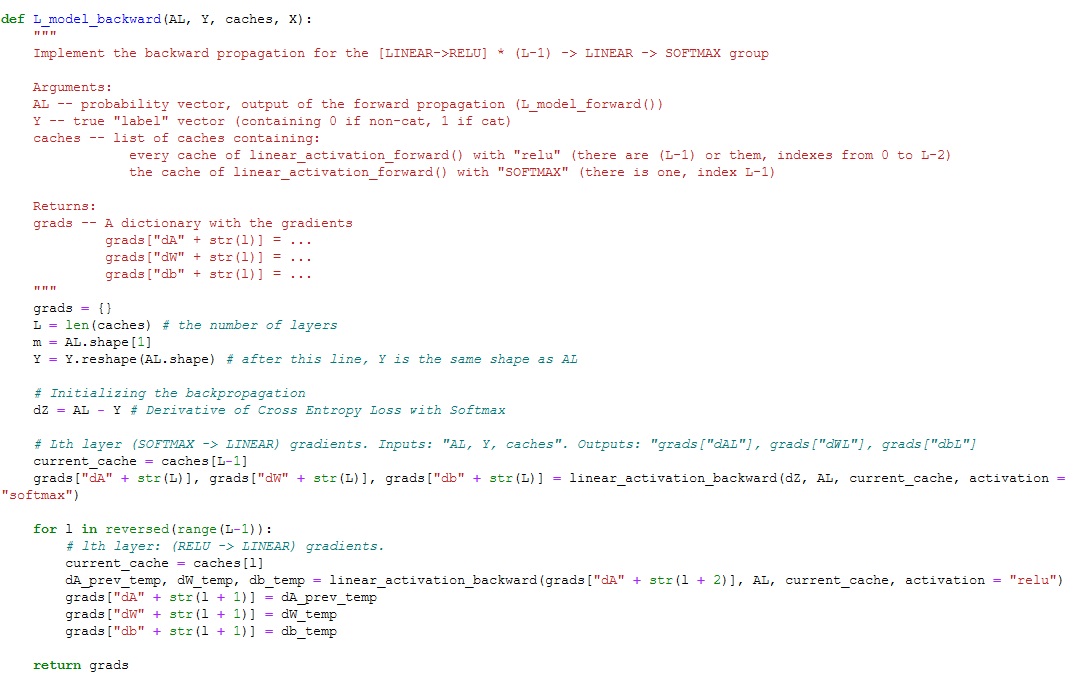
*Функција трошка параметара логистичке регресије сa Softmax функцијом*

5. 3. 5. Пропагација уназад

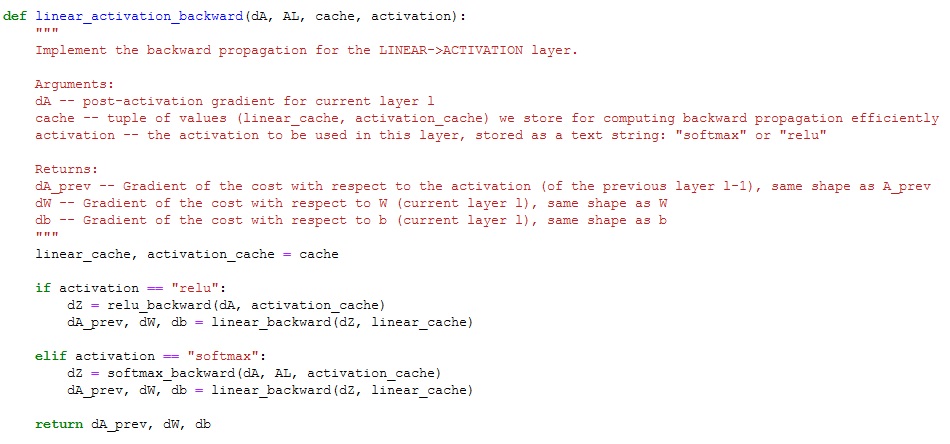
Пропагација уназад се користи за израчунавање градијената свих параметара у мрежи. Градијенти параметара су потребни како би се извршио update свих параметара у току оптимизације трошка параметара када се укључи један од алгоритма за оптимизацију.

5. 3. 5. 1. Потпуно повезана мрежа

Функције које омогућавају рачунање градијената параметара кроз потпуно повезану мрежу налазе се на сликама 32, 33, 34. За рачунање градијената потпуно повезане мреже коришћени су и градијенти функција ***ReLu*** и ***Softmax*** (слика 35 и 36) и градијент функције трошка параметара *dZ* одређен као на слици 32.

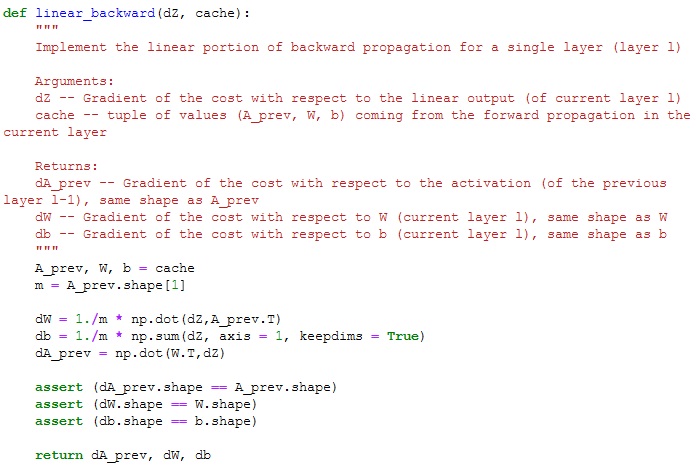
*Слика 32 :*

*Функција која израчунава градијент активационе функције и градијент параметара на излазу и омогућава пролаз кроз свих L-1 слојева мреже*

**

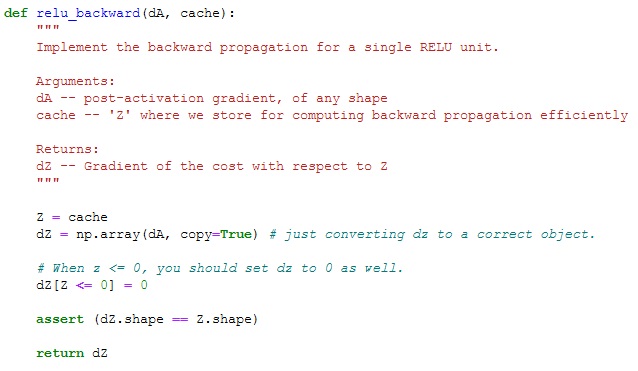
*Слика 33 :*

*Функција која омогућава подешавање активационе функције у скривеним слојевима и рачунање градијената активационих функција*

**

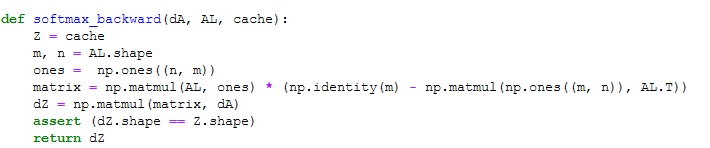
*Слика 34 :*

*Функција која израчунава градијент активационе функције претходног слоја помоћу линеарних компутација*

**

*Слика 35 :*

*Функција која израчунава градијент ReLu активационе функције*

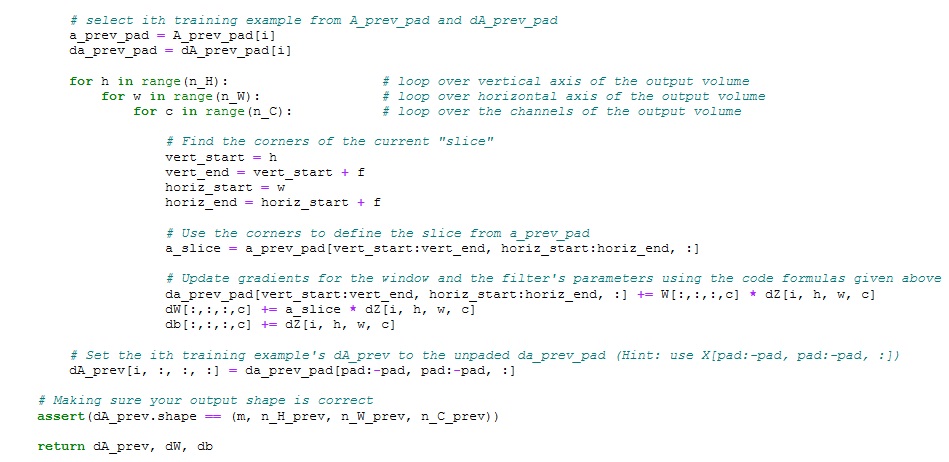
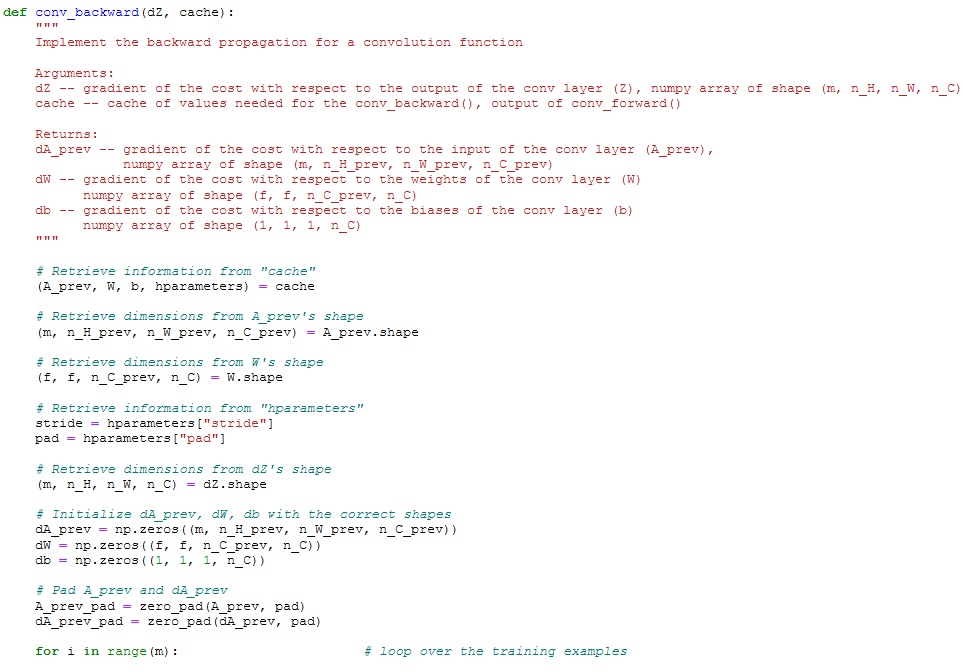
**

*Слика 36 :*

*Функција која израчунава градијент softmax активационе функције*

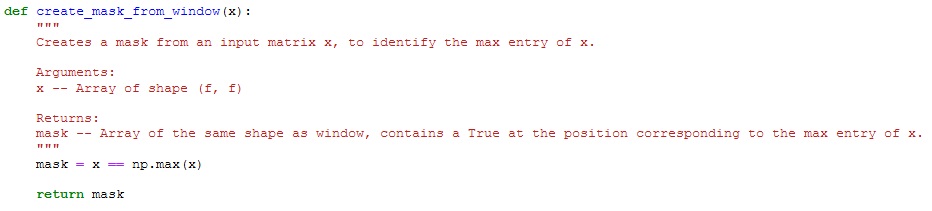
5. 3. 5. 2. Конволуцијска мрежа

Пропагација уназад је имплементирана као на сликама 37, 38 и 39.



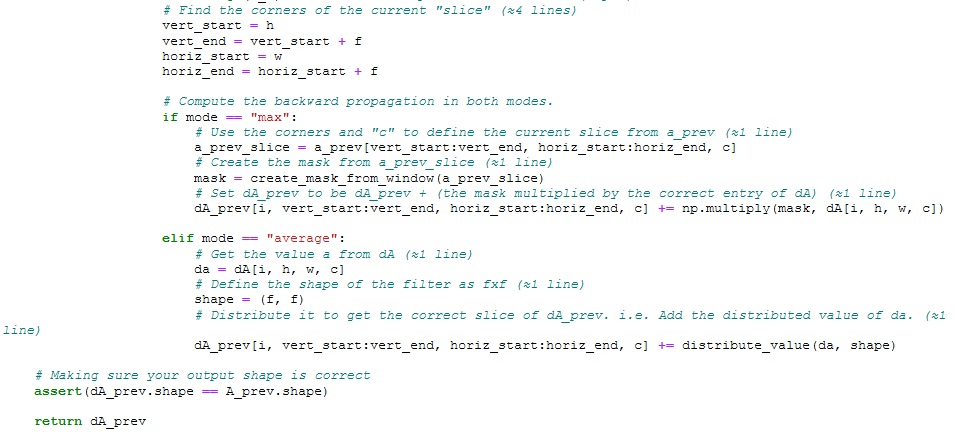
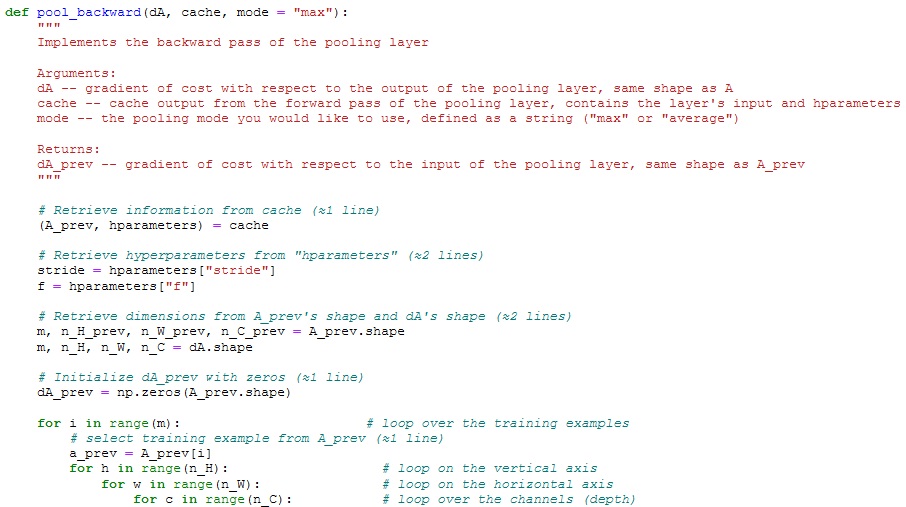
*Слика 37 :*

*Функција која израчунава градијент параметара филтера и градијент улазних података у једном слоју конволуцијске мреже*



*Слика 38 :*

*Помоћна функција за пропагацију уназад кроз pooling слој која прати максимум вредност параметара слоја након технике pooling-а*



*Слика 39 :*

*Функција која израчунава градијент улазних података pooling слоја*

5. 3. 6. Update-овање параметара

Update-овање параметара се врши како би се пронашле оптималне вредности за све параметре мреже који дефиниши границу одлучивања. Помоћу вредности најоптималнијих параметара трошак параметара (cost) је минималан.

*Градијент опадања*



Правило градијента опадања : , где је 𝛼 стопа учења.

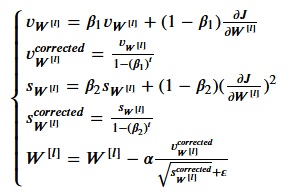
*Вредност стопе учења која је коришћена није константна вредност већ се смањује чиме је учење убрзано* (слика 41).

Како би update-овао параметре градијент опадања користи градијенте параметара који су претходно израчунати.

Уколико је градијент параметра позитиван број следи да ће вредност претходног параметра бити већа од вредности update-ованог параметра , па ће смер градијента опадања одржати правилан смер ка минимуму.

Уколико је градијент параметра негативан број следи да ће вредност претходног параметра бити мања од вредности update-ованог параметра па ће смер градијента опадања одржати правилан смер ка минимуму.

Стога за било који знак градијента параметара градијент опадања ће тежити да минимизује параметре .



*Адам алгоритам*

“Правило Адама за = 1,…,

, где је :

броји број корака Адам

алгоритма,

број слојева,

*𝛽*1 и *𝛽*2 су хипер параметри који контролишу експоненцијално отежане просеке,

стопа учења и

*𝜀* јако мали број да би се избегло дељење нулом

(2).”

Како би update-овао параметре алгоритам Адам користи градијенте параметара који су претходно израчунати (слика 40).



*Слика 40 :*

*Update-овање параметара користећи алгоритам Адам*

6. Функција модела неуронске мреже

За тренирање параметара неуронске мреже све функције које су претходно приказане у моделу неуронске мреже су повезане у оквиру функције модела са слике 41, 42 и 43.

Функција модела је дефинисана на следећи начин :

*model(X, Y, layers\_dims\_conv, channels, nc, layers\_dims\_fc, learning\_rate = 0.0007, mini\_batch\_size = 64, beta1 = 0.9, beta2 = 0.999, epsilon = 1e-8, num\_epochs = 100, print\_cost = True),* где је

* X - вредност свих улазних тренинг података
* Y - вредност свих излазних тренинг података
* layers\_dims\_conv - речник који садржи димензије филтера у конволуцијском делу мреже
* channels - број канала за сваки од филтера
* nc - број филтера за сваку од конволуција
* layers\_dims\_fc - речник који садржи димензије параметара у скривеним слојевима потпуно повезане мреже
* learning\_rate = 0.0007 - дифолт вредност стопе учења
* mini\_batch\_size = 64 дифолт вредност величине серије података
* beta1 = 0.9, beta2 = 0.999, epsilon = 1e-8 - дифолт вредности хипер параметара Адам алгоритма
* num\_epochs = 100 - дифолт вредност броја епоха
* print\_cost = False дифолт вредност параметра који одлучује о цртању функције трошка



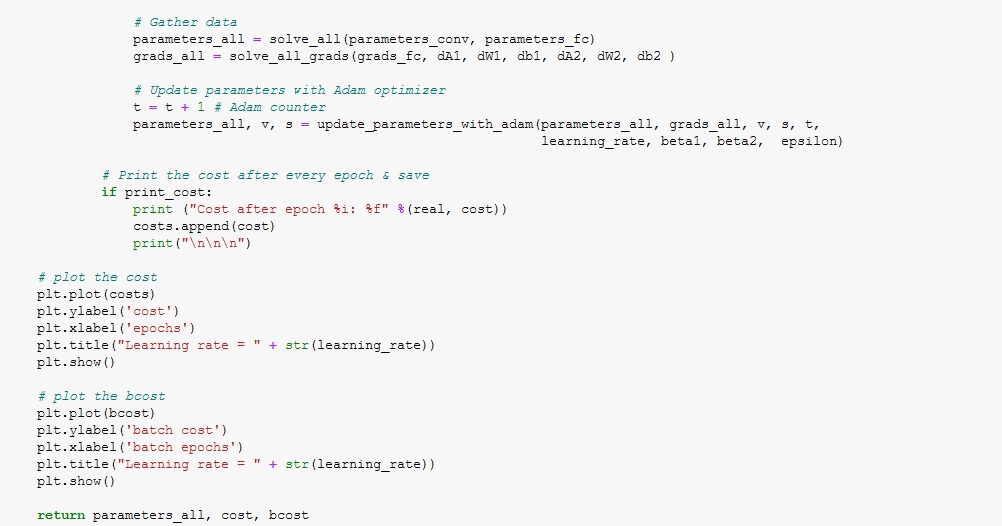
*Слика 41 :*

*Модел оптимизоване неуронске мреже, I део*



*Слика 42 :*

*Модел оптимизоване неуронске мреже, II део*

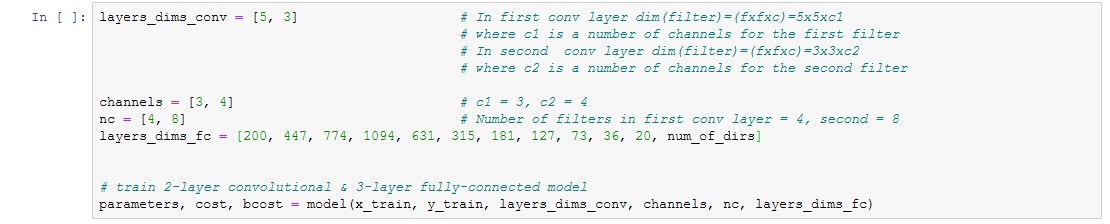
**

*Слика 43 :*

*Модел оптимизоване неуронске мреже, III део*

7. Тренирање и резултати неуронске мреже

За тренирање параметара неуронске мреже позива са функција модела којој су прослеђени одговарајући параметри који фигуришу унутар конволуцијске (*layers\_dims\_conv, channels, ns*) и потпуно повезане мреже (*layers\_dims\_fc*) (слика 44).

**

*Слика 44 :*

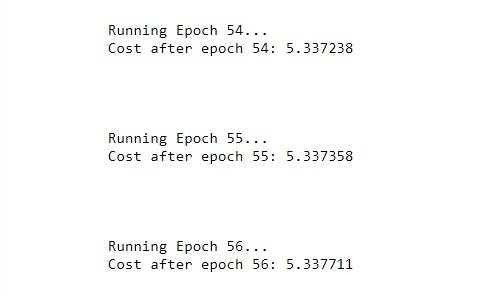
*Позив функције модела и тренирање параметара на тренинг скупу података*

Резултати неуронске мреже, односно вредности функције трошка параметра у току 10 епоха су приказани на сликама 45, 46 и 47.



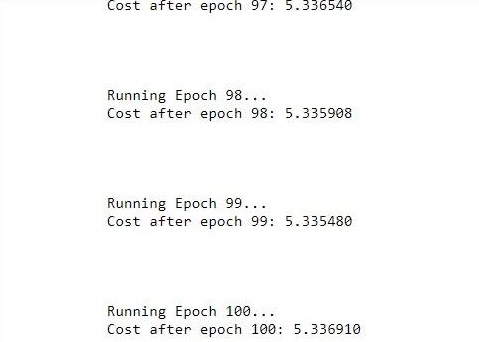
*Слика 45 :*

*Резултат после првих 4 епоха*

**

*Слика 46 :*

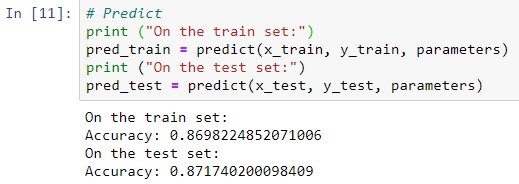
*Резултат после 56 епоха*

**

*Слика 47 :*

*Резултат после 100 епоха*

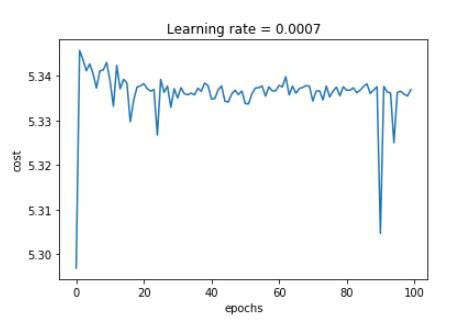
Тачност неуронске мреже на тренинг и тест скупу односно разлика предвиђених вредности излазног вектора у односу на стварне вредности излазног вектора представљена је на слици 48.

**

*Слика 48 :*

*Одређивање тачности алгоритма на тренинг и тест скупу*

Вредност функције трошка није константно опадајућа функција јер Адам алгоритам оптимизације у току епоха изабира серије улазних података на рандом начин тј. при рачунању трошка и update-овања параметара увек ради са различитим скупом серије података. Графички приказ представљен је на слици 49.

**

*Слика 49 :*

*Графички приказ вредности функције трошка у зависности од броја епоха*

8. Закључак

Грешка коју овај класификациони алгоритам прави на :

* тренинг скупу податка је : 0,130, грешка је 13%
* тест скупу података је : 0,128, грешка је 12,8%

Ова грешка се може сматрати грешком високог одступања/бијаса (*bias*, engl.). Тренинг сет даје објашњење о високом бијасу. Грешка услед бијаса је количина колико се предвиђање очекиваног модела разликује од праве вредности тренинг података. Начини на који се ова грешка може смањити су:

* већа мрежа / већи број слојева мреже
* дужа итерација / већи број епоха
* другачија архитектура мреже

9. Литература

1. М. Николић, М. Ивановић, С. Миленковић, Ј. Миливојевић, М. Милутиновић. The state and prospects of raspberry production in Serbia. Acta horticulturae, 2008; 777(777):243-250, доступно са:

<https://www.researchgate.net/publication/284707670_The_state_and_prospects_of_raspberry_production_in_Serbia>

2. A. Ng, Ванредни професор са Универзитета Станфорд и oснивач курсева deeplearning.ai и Coursera Inc., Deep Learning Specialization [Видео материјали], доступно са:

<https://www.coursera.org/specializations/deep-learning>