Факултет инжењерских наука Универзитета у Крагујевцу

Анђела С. Благојевић

Имплементација и примена 1Д конволуционих неуронских мрежа

завршни рад

Крагујевац, 2019.

Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Рачунарска техника и софтверско инжењерство
Ниво студија: Основне академске студије
Предмет: Вештачка интелигенција
Број индекса: 558/2015

Анђела С. Благојевић

Имплементација и примена 1Д конволуционих неуронских мрежа

завршни рад

Комисија за преглед и одбрану:

1. **Др Весна Ранковић - ментор**
2. _____
3. _____

Датум одбране:_____

Оцена:_____

У оквиру овог завршног рада кандидат треба да

Опише архитектуру тј. пет основних градивних компоненти (слојева) основне конволуционе неуронске мреже: улазни слој, слој конволуције, слој сажимања, слој активационе функције и потпуно повезани слој. Применити 1Д конволуциону неуронску мрежу у моделирању временске серије података.

Препоручена литература:

[1] S. Pattanayak, Pro Deep Learning with Tensorflow, Apress, 2017.
[2] F. Chollet, Deep Learning with Python, Shelter Island, New York, Manning Publications, 2018.
[3] M. Kachuee, S. Fazeli, M. Sarrafzadeh, ECG Heartbeat Classification: A Deep Transferable, IEEE International Conference on Healthcare Informatics (ICHI), 2018.

Крагујевац, 14.05.2019.

Ментор:
Др Весна Ранковић, редовни професор
_____

Садржај

Резиме

У овом завршном раду обрађен је модел конволуционих неуронских мрежа. У првом делу дат је увод у класичне неуронске мреже, како би се боље разумели основни појмови код дубоких мрежа. У другом делу рада описана је математичка конволуција по којој су конволуционе мреже и добиле име, као и архитектура ових мрежа. Описани су детаљно сви слојеви који се налазе у конволуционим неуронским мрежама. Након теоријских основа, следи имплементација једног модела поменутих мрежа на ЕКГ сигналу. Модел је развијен у *Python*-у, помоћу библиотека *Keras* и *TensorFlow*. Након што је модел истрениран, приказане су све карактеристике модела, као и резултати у виду тачности и функције губитка. Такође, пошто је скуп података небалансиран, приказани су и параметри осетљивост, прецизност и F1 оцена.

Кључне речи: дубоке неуронске мреже, конволуционе неуронске мреже, ЕКГ, конволуција, програмски језик *Python*

Summary

In this paper, a model of convolutional neural networks is given. The first part provides an introduction to classic neural networks in order to better understand the basic concepts of deep networks. The second part of the paper describes the mathematical convolution by which the convolution neural networks were named and architecture of these networks. All layers found in convolutional neural networks are described in detail. After theoretical foundations, the implementation of one model of the mentioned networks is given with ECG signal. The model was developed in Python, using the Keras and TensorFlow libraries. After the model has been trained, all the characteristics of the model are shown, as well as the results in terms of accuracy and loss function. Also, since the data set is unbalanced, the sensitivity, precision and F1 score are also displayed.

Key words: deep neural networks, convolutional neural networks, ECG, convolution, programming language Python

1 Увод

Дубоко учење је подручје вештачке интелигенције о чијем значају говоре задивљујући резултати неуронских мрежа на бројним задацима. Независно од тога да ли се ради о обради слика, текста, звука, видео снимака или су у питању сигнали попут електроенцефалограма (ЕЕГ) или електрокардиограма (ЕКГ), неуронске мреже постају све чешћи одабир за решавање проблема. Конкретно, ови алгоритми су све чешће примењивани у медицини, о чему сведоче и бројна истраживања попут [1 – 3]. Захваљујући развоју ових области, и сама медицина је напредовала. Рачунари решавају бројне проблеме, које сам човек не може. Рачунари успевају да увиде неке абнормалности, које људско око није у стању.

У овом раду биће разматран сигнал електрокардиограма, односно ЕКГ. Овај сигнал, због свог значаја, је чест избор при истраживању. Циљ овог рада је увидети да ли је за исти сигнал као у истраживању [3] могуће добити приближно добре резултате са једноставнијим алгоритмом. За решавање проблема изабране су 1Д конволуционе неуронске мреже, и циљ је формирати што једноставнији модел који ће дати задовољавајуће резултате. Да би се конволуционе неуронске мреже разумеле у потпуности, неопходно је прво почети са класичним неуронским мрежама које су описане на самом почетку рада. Врло сажето и сликовито објашњен је појам неурона и неуронске мреже. Циљ овог поглавља је увођење основних појмова попут активационих функција, градијентног спуста и функције губитка.

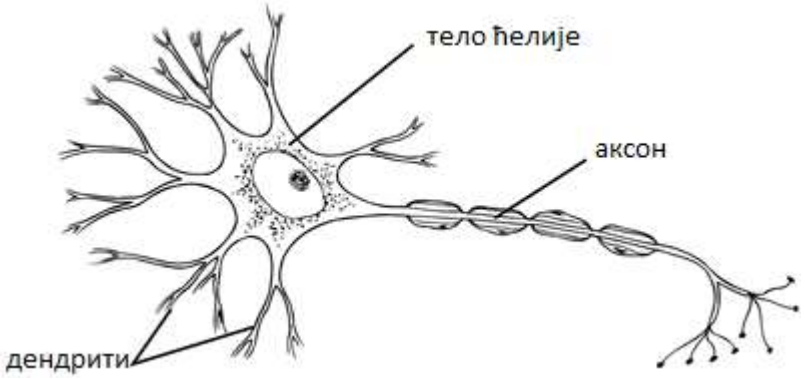
У следећем поглављу дат је детаљан опис конволуционих неуронских мрежа. Првенствено, објашњена је математичка конволуција која представља суштину ових неуронских мрежа. Како се практични део овог рада заснива на анализи једнодимензионалног сигнала, засебно је представљена и конволуција на овим сигналима. Затим је детаљно објашњен сваки слој поменутих мрежа, ради бољег разумевања и ради припреме за практични део рада.

Последње поглавље се односи на већ поменути практични део. Конволуциона неуронска мрежа која има задатак да класификује ЕКГ сигнал у једну од 5 различитих категорија аритмија имплементирана је у програмском језику *Python,* помоћу библиотека које су специјализоване за дубоко учење и представљају неке од најновијих и најоптимизованијих технологија за рад у овој области.

2 Увод у неуронске мреже

2.1 Основни модел неуронске мреже

Дубоко учење као грана вештачке интелигенције развило се моделирањем првих вештачких неурона 1940-их. Неуронске мреже су повезане мреже, састављене од градивних јединица, распоређених по слојевима, које називамо вештачким неуронима. Идеја модела вештачке мреже је опонашање рада биолошких неурона мозга. Неурони мозга међусобно су повезани синапсама и у интеракцији су са хиљадама неурона који их окружују. Циљ је опонашати биолошко учење које се темељи на преносу информација између неурона. Сваки биолошки неурон од суседних неурона прима улазне сигнале помоћу дендрита. Излазни, модификовани сигнал потом се шаље осталим неуронима преко аксона, који су својим завршецима повезани с дендритима других неурона [4].



Слика 2.1. Биолошки неурон.

2.2 Неурон

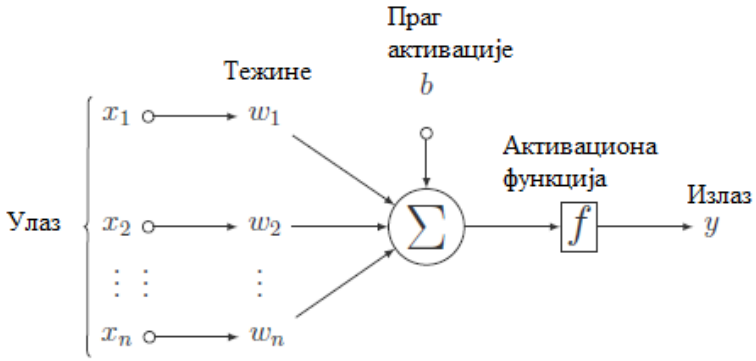
Неуронске мреже садрже неуроне који су мотивисани биолошким неуронима, али уз одређене модификације. Први математички модел неурона предложили су *McCulloch* и *Pitts* [5]. Како су многи детаљи у вези неурона, па и самог људског мозга још увек непознати, вештачки неурон се увек представља једноставним математичким моделом.

Дефиниција 2.2.1: Нека је $n \in \mathbb{N}$. Функцију $\mathbb{N}^f : \mathbb{R}^n \to \mathbb{R}$ дефинисану са

$$\mathbb{N}^f(x_1, \ldots, x_n) = f\left(\sum_{i=1}^{n} x_i w_i - b\right) \tag{2.1}$$

називамо McCulloch-Pitts неурон. Реалне бројеве $w_1, w_2, \ldots, w_n$ зовемо **тежине**, реалан број b **праг активације**, а функцију $f : \mathbb{R} \to \mathbb{R}$ **активациона функција**.

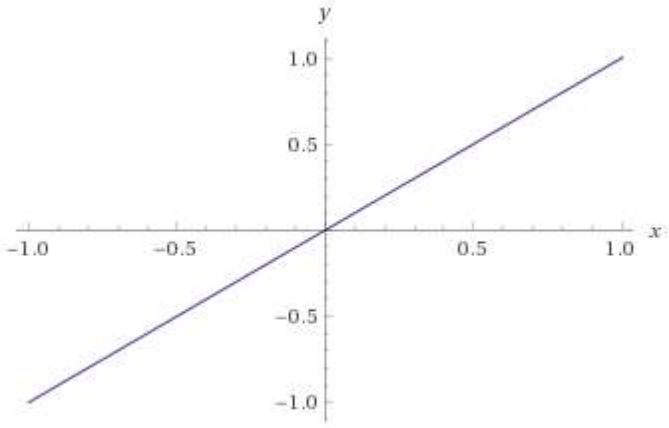
Тежине можемо представити као квантификацију важности сваке од компоненти улазног вектора. Оне представљају параметре које мрежа може сама научити. Сума унутар функције f је заправо скаларни производ вектора $(x_1, \ldots, x_n)$ и $(w_1, \ldots, w_n)$. Улога активационе функције подстакнута је природом рада биолошког неурона где се сигнал неурона прослеђује следећим неуронима ако и само ако је улазни надражај тог неурона довољно снажан. У оригиналној дефиницији овог модела неурона коришћена је функција сигнум, док се данас, због начина учења, користе разне диференцијабилне функције. Приметно је да је вредност прага активације баш она вредност за коју се догађа промена вредности функције сигнум. На слици 2.2 је графички приказ неурона из претходне дефиниције.



Слика 2.2. Математички модел неурона.

2.3 Активационе функције

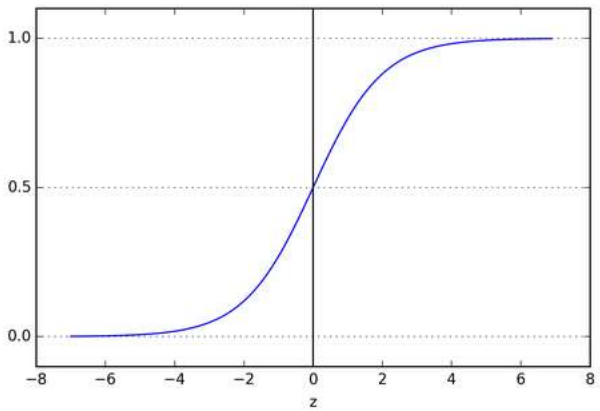
Уколико излазна вредност неурона линеарно зависи од његових улазних података, онда је реч о линеарној активационој функцији. Линеарна функција представљена је графички на слици 2.3.



Слика 2.3. График линеарне функције.

Нека је пред нама проблем бинарне класификације, где је улаз потребно сврстати у једну од две класе. У овом случају линеарна активациона функција често није најбоље решење, ради постизања бољих резултата, често се за активационе функције бирају нелинеарне функције. Најпозантији пример активационе функције за проблем бинарне класификације је сигмоидна функција:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$



Слика 2.4. График сигмоидне функције.

Поред сигмоидне функције, најчешћи одабир је функција тангенс хиперболички:

$$th(x) = \frac{e^{2x} - 1}{e^{2x} + 1}. \tag{2.3}$$

Ове две функције су повезане следећом линеарном трансформацијом:

$$th(x) = 2\sigma(2x) - 1. \tag{2.4}$$

То значи да свака функција која је израчуната са функцијом активације тангенс хиперболички може бити израчуната са сигмоидном функцијом активације. Зато сматрамо да су ове две функције еквивалентне као функције активације.

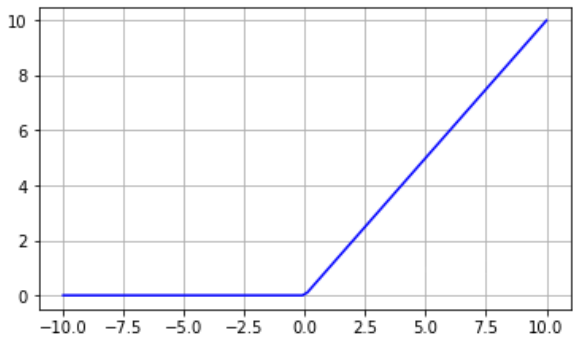
Ако изузмемо бинарну класификацију, сигмоидна активациона функција се све ређе користи и постаје застарела. Показано је да приликом учења неуронске мреже, информације које путују мрежом почињу да се губе, па је учење мреже отежано. Такође, излазне вредности сигмоидне функције нису центриране око исходишта, што је такође мана при учењу мреже.

За случај када бисмо желели решити класификациони проблем са више различитих излазних вредности онда бисмо користили *softmax* активациону функцију [6]. Дефиниција ове функције дата је са:

$$softmax(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \qquad \text{за } i = 1, \dots, K \text{ и } z = (z_1, \dots, z_K) \in \mathbb{R}^K. \tag{2.5}$$

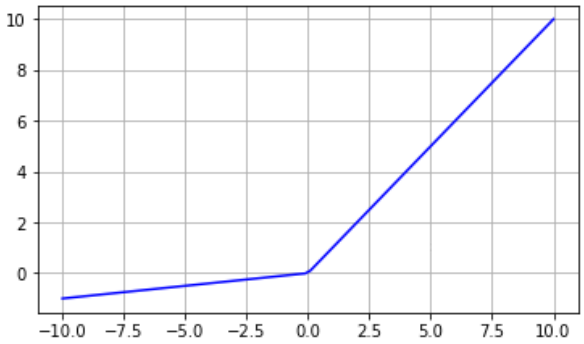
Једна од функција која се веома често користи код скривених слојева мреже, о којима ће у наставку бити говора, је *ReLU* (engl. Rectifed Linear Unit). Дефиниција ове функције дата је са:

$$ReLU(x) = \max\{0, x\}. \tag{2.6}$$



Слика 2.5. ReLU активациона функција [7].

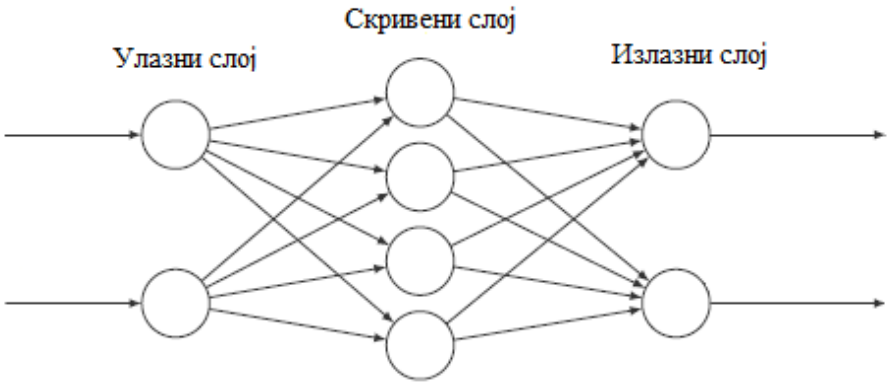
На слици 2.5 видимо да функција ReLU користи тачно нулу као границу одлуке. У пракси показује много боље резултате од сигмоидне функције, управо због своје "скоро" линеарне форме. За разлику од других нелинеарних функција активације, није је тешко израчунати, па је алгоритам бржи. Међутим у пракси се највише користи побољшана верзија функције која се зове пропусни *ReLU* (engl. leaky ReLU) и приказана је на слици 2.6.



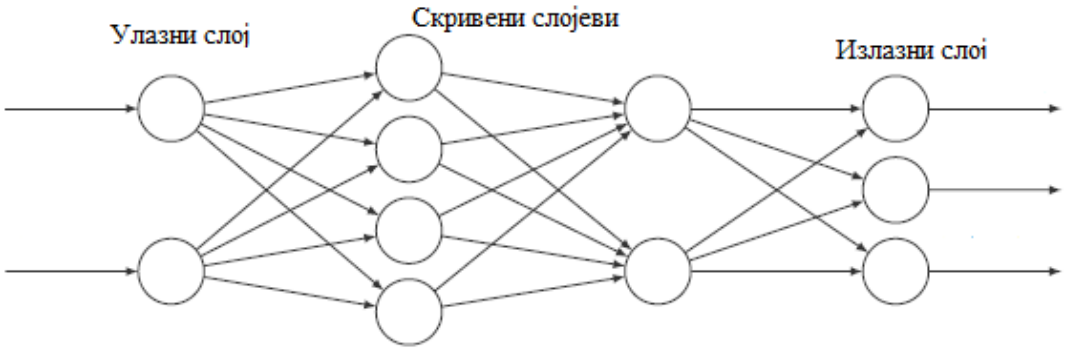
Слика 2.6. График функције пропусни ReLU [7].

2.4 Неуронска мрежа

Објаснимо сада терминологију неуронских мрежа. Нека је дата мрежа као на слици 2.7. Леви слој назива се улазни слој са улазним неуронима, док се десни десни назива излазни слој, слој са излазним неуронима. Средњи слој се назива скривени слој. Мрежа на слици 2.7 има само један скривени слој, али неуронске мреже могу имати више скривених слојева, и то су такозване дубоке неуронске мреже, као што је приказано на слици 2.8. Оваква неуронска мрежа позната је као вишеслојни перцептрон (engl. multilayer perceptron).



Слика 2.7. Неуронска мрежа са једним скривеним слојем.



Слика 2.8. Вишеслојни перцептрон са два скривена слоја.

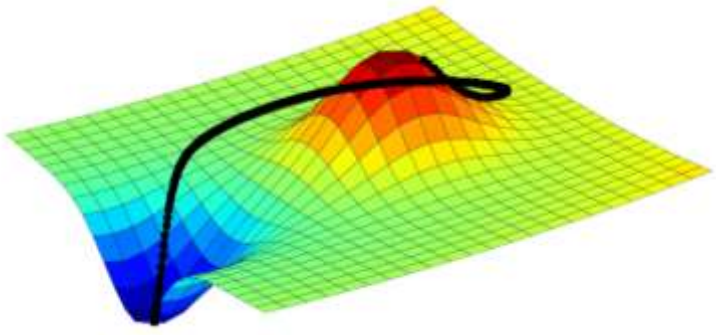
2.5 Учење неуронске мреже

Сада када смо упућени у терминологију неурона и неуронске мреже, потребно је савладати одређивање прагова активације и тежинских коефицијената за неуроне. Тај процес се назива учење, а кључну улогу код учења игра итеративни алгоритам градијентни спуст.

Дефиниција 2.5.1: Нека је $f : \mathbb{R}^n \to \mathbb{R}$ диференцијабилна функција, $x \in \mathbb{R}^n$ и $\alpha > 0$. Итеративни алгоритам дефинисан са

$$\begin{aligned} \theta_0 &= x, \\ \theta_{m+1} &= \theta_m - \alpha \nabla f(\theta_m), \quad m \in \mathbb{N}, \end{aligned} \tag{2.7}$$

називамо градијентни спуст (engl. gradient descent). Позитиван реалан број α називамо брзина учења (engl. learning rate). Смер најстрмијег пада у тачки θ_mозначен је са $\nabla f(\theta_m)$ и у сваком кораку алгоритма приближавамо се (локалном) минимуну.



Слика 2.9. Путања градијентног спуста ка локалном минимуну [8].

При овом алгоритму ради се минимизација функције губитка, о којој ће бити речи у наставку рада, на целом тренинг скупу. То је често превише споро јер се ради са великим скуповима података. Овај проблем решава идеја стохастичког градијентног спуста [9]. При овом алгоритму у свакој итерацији користи се само један пример из тренинг скупа. То омогућава брзу, али често непрецизну конвергенцију функције губитка. Зато се у пракси најчешће користи неки подскуп (engl. mini-batch) који омогућава брзо рачунање и прецизну конвергенцију. Данас су популарне разне варијације стохастичког градијентног спуста као што су на пример *AdaGrad*, *Adam* и *RMSProp*.

2.6 Функција губитка

Нека је дат скуп за учење модела, односно скуп података које је потребно класификовати у неку од K датих класа. Сваком i-том примеру означеном са $x^{(i)}$ који припада скупу података додељујемо ознаку $y^{(i)} \in \{1, \dots, K\}$ која означава којој класи $x^{(i)}$ припада. Након тренирања мреже, потребно је утврдити колико се добијени резултат разликује од стварне класе $y^{(i)}$ којој податак $x^{(i)}$ припада. Функција губитка мери одступање излазног резултата класификације од стварне вредности $y^{(i)}$ придружене податку $x^{(i)}$. Губитак ће бити већи ако је објекат погрешно класификован. Такође, важно је напоменути да избор функције губитка зависи од врсте улазних података за класификацију. Једна од функција губитка која је најчешћи избор је унакрсна ентропија (engl. cross entropy). Унакрсна ентропија представља функцију коју примењујемо на вектору резултата чије су вредности нормализоване. Унакрсна ентропија представља се следећом једначином:

$$L(y, p) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{\text{K}} y_{i,j}\log\left(p_{i,j}\right), \tag{2.8}$$

где је, као што је већ речено, K број класа, а N број узорака у скупу. Вредност y представља стварну вредност класе, док је вредност p излазни резултат класификације, односно процењена вредност.

3 Конволуционе неуронске мреже

Конволуционе неуронске мреже су посебан тип неуронских мрежа за процесирање неструктурираних података, посебно су значајне за рад са сликама, текстом, звуком, говором и једнодимензионалним сигналима. Погодније су за коришћење од потпуно повезаних неуронских мрежа зато што су потпуно повезане неуронске мреже веома временски захтевне. Број тежина с којима треба радити значајно се повећава с повећањем броја података или величине мреже, Додатно, у случају великог броја неурона или слојева, може доћи до прекомерног тренирања (engl. overfitting) уколико не постоји велика количина података у тренинг скупу. Конволуционе неуронске мреже су настале као покушај решавања тих проблема уз побољшање резултата, а идеја за модел је добијена кроз спознавање начина на који људски мозак функционише. *Hubel* и *Wiesel*, су својим радом за који су добили Нобелову награду 1981. године показали утицај ћелија видног кортекса мачке на разне подражаје. Конкретно, *Hubel* и *Wiesel* откривају да мождани неурони који су близу једни другима активирају се приликом гледања линија под једним углом, док друга група неурона се мења када променимо угао. Такође, неке групе неурона препознају ивице, без обзира на угао под којим се налазе. Њихов рад је био инспирација за моделирање конволуционе неуронске мреже. Име су добиле по математичкој операцији конволуцији, коју ове неуронске мреже примењују барем у једном истоименом слоју.

3.1 Конволуција

Конволуција је операција на две функције са реалним доменима, која на излазу даје измењену верзију једне од те две оригиналне функције. Измењени сигнал може имати боља својства од оригиналног сигнала за неки одређени задатак. За операцију конволуције се углавном користи ознака *. Знање о линеарним временски инваријантним системима и линеарним инваријантним системима са померајем помаже бољем разумевању конволуције на сигналима. Ове системе ћемо размотрити у наставку. Користићемо скраћенице LTI (engl. linear time invariant system) и LSI (engl. linear shift invariant system) [10].

Ако се за улазни сигнал система $x(t)$ производи излазни сигнал $y(t)$ неком трансформацијом $f(t)$. Онда излазни сигнал можемо записати у облику

$$y(t) = f\big(x(t)\big). \tag{3.1}$$

За систем кажемо да је линеаран ако важе следећа два својства:

- Скалирање: $$f\big(\alpha x(t)\big) = \alpha f\big(x(t)\big) \tag{3.2}$$
- Суперпозиција: $$f\big(\alpha x_1(t) + \beta x_2(t)\big) = \alpha f\big(x_1(t)\big) + \beta f\big(x_2(t)\big) \tag{3.3}$$

Слично, за систем кажемо да је временски инваријантан, ако важи:

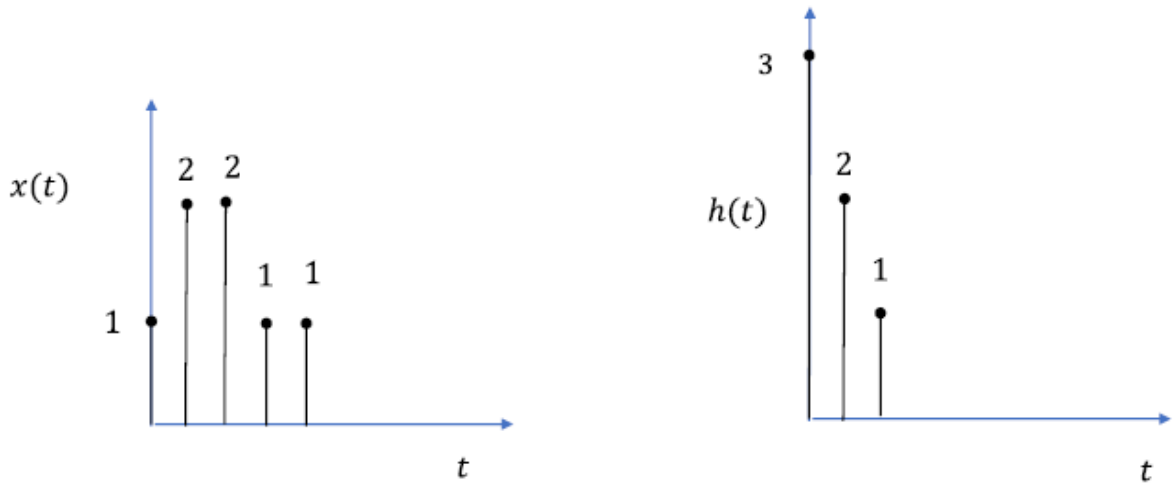
$$f\big(x(t-\tau)\big) = y(t-\tau). \tag{3.4}$$

3.1.1 Конволуција на једнодимензионалним сигналима

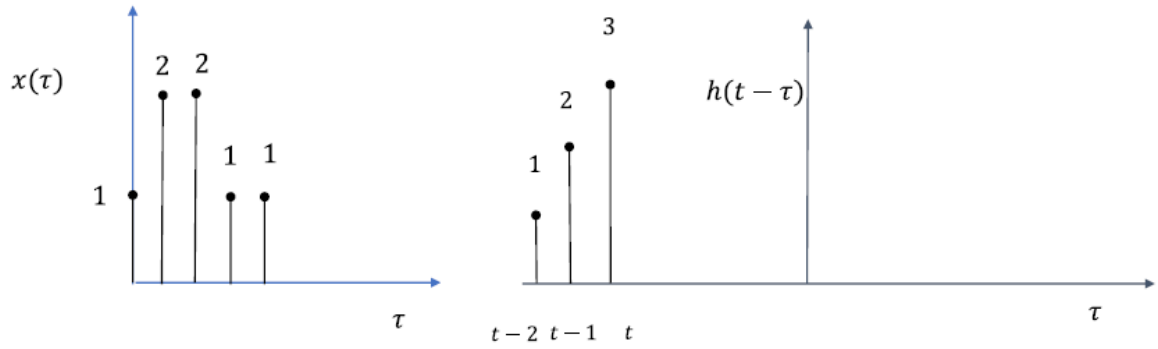
За систем кажемо да је временски дискретан ако на њему делују функције трансформације дефинисане на дискретном скупу времена $\mathbb{N} \cup \{0\}$. Ове функције називамо временски дискретним функцијама. Такође, за систем кажемо да је временски непрекидан ако на њему делују функције дефинисане на скупу $\mathbb{R}$ или на неком непрекидном подскупу истоименог скупа. За такве функције кажемо да су временски непрекидне функције. Конволуција мери степен преклапања између једне функције и помереног облика друге функције. Нека су x и h временски дискретне функције. Конволуцију функција x и h дефинишемо на следећи начин:

$$y(t) = x(t) * h(t) = \sum_{\tau=-\infty}^{+\infty} x(\tau)h(t-\tau). \tag{3.5}$$

На слици 3.1 приказане су две временски дискретне функције x и h у зависности од времена t.



Слика 3.1. Улазни дискретни сигнали [10].



Слика 3.2. Функције за рачунање конволуције [10].

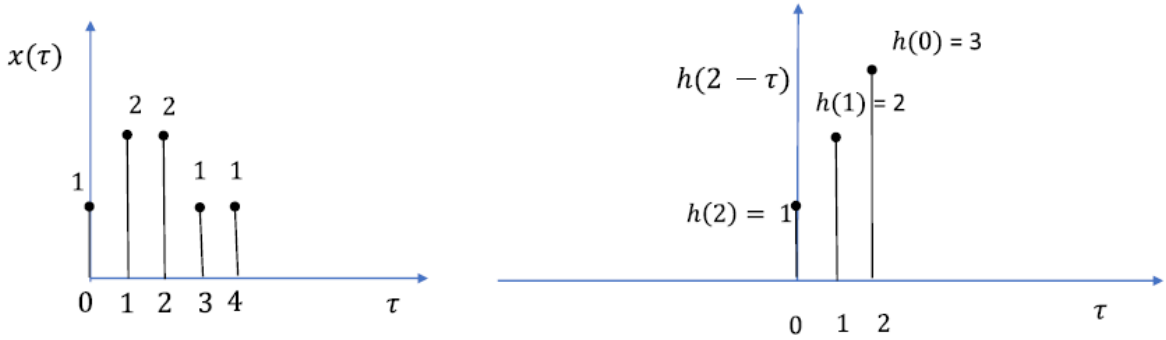
Функцију *h(t-τ)* је потребно израчунати за различите вредности времена *t* померањем дуж хоризонталне осе као што је приказано на слици 3.2. За сваку вредност *t* рачуна се сума $\sum_{\tau=-\infty}^{+\infty} x(\tau)h(t-\tau)$. Можемо уочити:

- Када је *t* = -1 тежине су дате са $h(-1-\tau)$ али нема преклапања са функцијом $x(\tau)$ па је сума једнака 0.

- Када је $t = 0$ тежине су дате са $h(-\tau)$ и једини елемент функције $x(\tau)$ који се преклапа са тежинама је $x(\tau = 0)$, са преклапајућом тежином $h(0)$, па је сума $x(0) * h(0) = 1 * 3$.

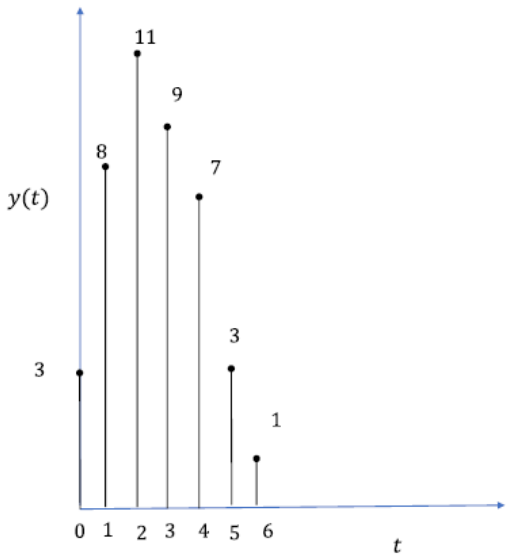
- Када је $t = 1$ тежине су дате са $h(1-\tau)$. Елементи $x(0)$ и $x(1)$ се преклапају са тежинама $h(1)$ и $h(0)$, респективно. Из тога следи да је сума једнака $x(0) * h(1) + x(1) * h(0) = 1 * 2 + 2 * 3 = 8$.

- Када је *t* = 2 тежине су дате са $h(2-\tau)$. Елементи $x(0)$, $x(1)$ и $x(2)$ се преклапају са тежинама $h(2)$, $h(1)$ и $h(0)$, респективно. Из тога следи да је сума једнака $x(0) * h(2) + x(1) * h(1) + x(2) * h(0) = 1 * 1 + 2 * 2 + 2 * 3 = 11$. Преклапање функција за вредност *t = 2* приказано је на слици 3.3.



Слика 3.3. Преклапање функција у тачки t=2 [10].

Функције које улазе у конволуцију дају излазну функцију, односно дискретну функцију *y(t)*, која је приказана на слици 3.4.



Слика 3.4. Излазни сигнал. Резултат конволуције [10].

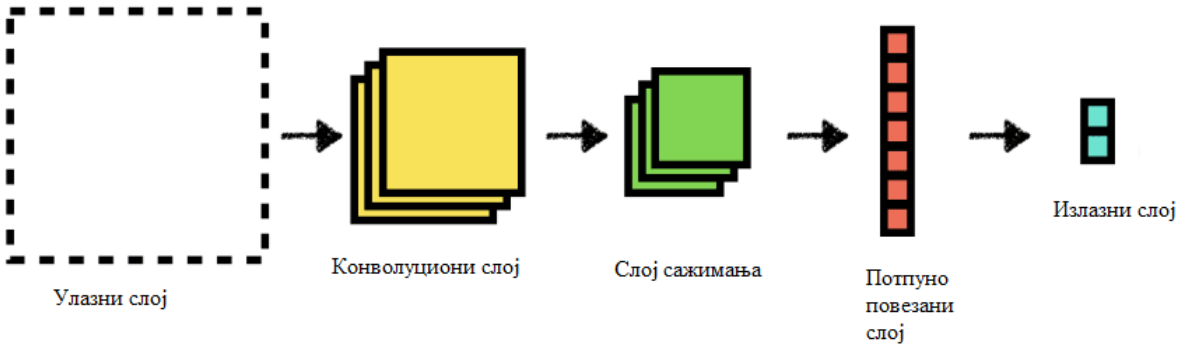
Сличан поступак понављамо и уколико се ради о временски непрекидним сигналима. Ако имамо две временски непрекидне функције $x(t)$ и $h(t)$. Тада конволуцију те две функције дефинишемо на следећи начин:

$$y(t) = x(t) * h(t) = \int_{\tau=-\infty}^{\infty} x(\tau)h(t-\tau)d\tau. \tag{3.6}$$

3.2 Архитектура конволуционе неуронске мреже

У овом делу биће описане основне компоненте конволуционе неуронске мреже као и њихова улога. На улазу конволуционе неуронске мреже може бити једнодимензионални сигнал, који приказујемо као низ који карактерише дужина његових сегмената, односно временске периоде.

Конволуциону неуронску мрежу можемо приказати као низ слојева, при чему сваки слој трансформише резултат претходног слоја уз помоћ одређених функција и производи излаз који се назива активациона мапа. Основне компоненте конволуционе неуронске мреже су: **улазни слој, конволуциони слој, слој сажимања, активационе функције** и **потпуно повезани слој**.



Слика 3.5. Слојеви конволуционе неуронске мреже.

При моделирању конволуционе неуронске мреже, најчешће се појављују три основне фазе. У првој фази се одвијају конволуције, што за резултат има активационе мапе. У другој фази, свака активациона мапа пролази кроз трансформацију неке од активационих функција. У трећој фази користе се функције сажимања како би се модификовао активациони слој.

3.2.1 Улазни слој

Неопходни део класичних и конволуционих неуронских мрежа представља улазни слој мреже. Конкретно код једнодимензионалних конволуционих неуронских мрежа на улаз се доводи једнодимензионални сигнал. Сваки оригинални сигнал након очитавања са сензора неопходно је претпроцесирати и припремити за улазни слој неуронске мреже, који је углавном у векторском облику. О обради сигнала и претпроцесирању биће речи у наставку.

3.2.2 Конволуциони слој

Конволуциони слој је основна компонента конволуционих неуронских мрежа, он обавља већи део трансформације података. Састоји се од два дела. У првом делу налази се улазна активациона мапа која улази у конволуцију са одређеним бројем филтера који могу препознати различите карактеристике. На улаз првог конволуционог слоја мреже се доводи улазни слој, односно на улазу се налази вектор одређених димензија. Након тога, у другом делу конволуционог слоја, на добијени резултат делује се неком од активационих функција. **Хиперпараметри** који се дефинишу у сваком конволуционом слоју су **број филтера d**, **величина филтера f**, **допуњавање p** и **корак s**.

Број филтера

Сваки конволуциони слој има унапред одређен број филтера, који чувају вредности тежина W. Те вредности представљају параметре модела па их мрежа може научити.

Величина филтера

Величина филтера дефинише се дужином вектора. На пример филтер дужине 9 садржи исто толико тежина. Тежине свих филтера у конволуционом слоју су иницијализоване на произвољне вредности пре почетка учења мреже. Нека је на улазу конволуционог слоја сигнал који је представљен вектором дужине n и нека је d број филтера. Сигнал улази у конволуцију са филтером дужине f. Уколико сваки филтер конволуирамо са улазним сигналом, па на резултат делујемо активационом функцијом, као крајњи резултат добија се активациона мапа која има димезије $(n - f + 1) \times d$.

Допуњавање

Допуњавање (engl. Padding) представља допуњавање нулама оригиналне активационе мапе око ивица. Осим што се допуњавањем може управљати димензијама активационе мапе, допуњавање доприноси да тежине које се налазе по ободу више учествују у преносу информација јер улазе више пута у конволуцију са филтером [10].

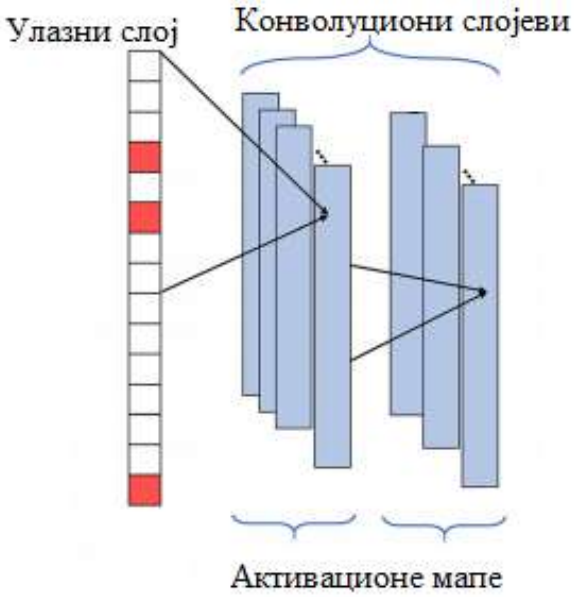
Две најчешће врсте допуњавања на основу димензија активационе мапе су:

- Валидно (engl. Valid padding) – активациону мапу не допуњујемо нулама, па се димензије излазне активационе мапе мењају
- Непромењено (engl. Same padding) – активациону мапу допуњујемо нулама, па се димензије мапе не мењају

Корак

Корак (engl. Stride) представља број за који се померамо при конволуцији активационе мапе и филтера. Подразумевана вредност је 1 и у том случају се вредности не прескачу. Уколико при конволуцији укључимо допуњавање p и корак s, као резултат се добија активациона мапа димензија

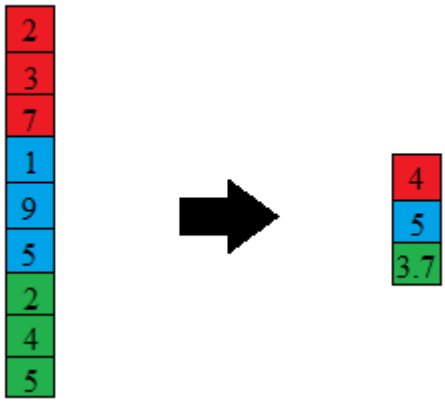
$$\frac{n + 2p - f}{s} \times d.$$



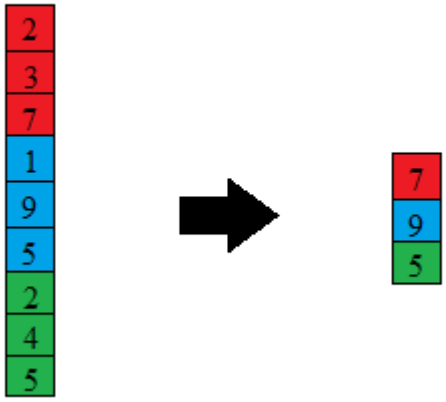
Слика 3.6. Приказ активационих мапа у конволуционим слојевима.

3.2.3 Слој сажимања

Слој сажимања (engl. pooling layer) представља примену функција сажимања на активационим мапама из претходног слоја, како би се умањиле димензије мапа. Два најчешћа типа сажимања су сажимање просечном вредношћу (engl. average-pooling), приказано на слици 3.7 и сажимање максималном вредношћу (engl. max-pooling) приказано на слици 3.8. На датим примерима у слоју сажимања користи се филтер величине 3, филтер ће обухватити три тежине и у случају сажимања максималном вредношћу, излаз ће бити максимална вредност од могуће три. У супротном, код сажимања просечном вредношћу, нова мапа која настаје у овом слоју биће испуњена просечним вредностима одређеним филтером [11].



Слика 3.7. Графички приказ сажимања просечном вредношћу.



Слика 3.8. Графички приказ сажимања максималном вредношћу.

3.2.4 Потпуно повезани слој

Потпуно повезани слој налази се на крају конволуционе неуронске мреже. Скраћено се назива FC слој (engl. fully connected layer). Састављен је од неурона који су у потпуности повезани са претходним слојем, као што је то случај код класичних неуронских мрежа. Сваки неурон повезан је са сваким елементом из активационе мапе претходног слоја. У пракси се користи неколико потпуно повезаних слојева на крају неуронске мреже. Након ових слојева следи резултат у векторском облику.

3.2.5 Слој искључивања

За разлику од претходних слојева, овај слој се користи као средство за побољшање генерализације неуронске мреже. Ова техника при сваком кораку тренинга насумично искључује излазне елементе вектора претходног слоја и прослеђује нове вредности следећем слоју мреже. Слој искључивања дефинише се формулом:

$$(dropout_p(x))_i = random(p) \cdot x_i. \tag{3.7}$$

Где је $p \in [0,1]$ функција која генерише насумичне реалне бројеве у истом сегменту.

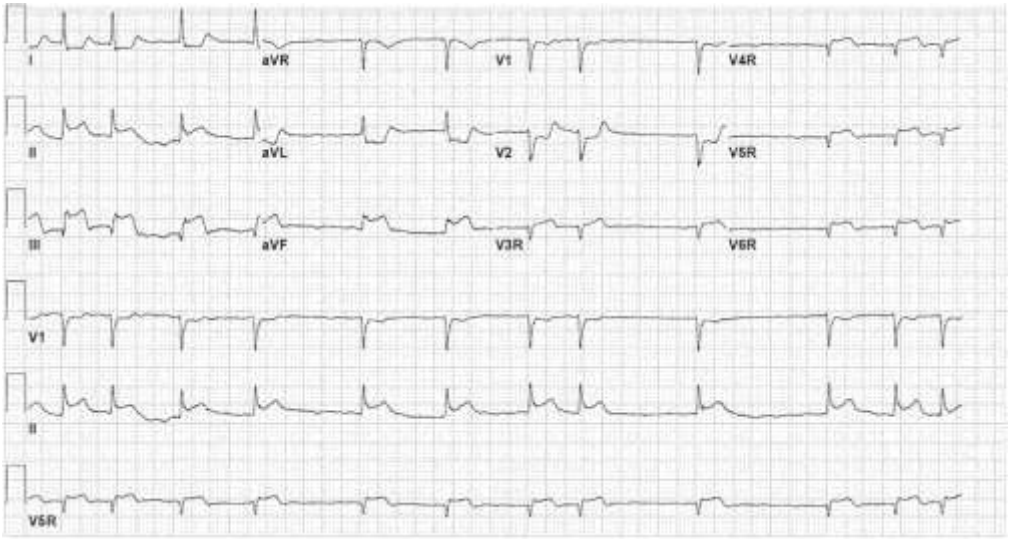
4 Имплементација конволуционих неуронских мрежа

4.1 Сигнали електрокардиограма (ЕКГ)

Кардиоваскуларне болести један су од главних узрока смрти на планети, на кардиоваскуларни проблем указују неправилни (аритмички) откуцаји срца. Сигнали ЕКГ широко се користе од стране лекара да надгледају и процењују здравље срца.

У конвенционалном 12-каналном ЕКГ-у, поставља се 10 електрода – на екстремитетима пацијента и на површини грудног коша. Укупне величине електричног потенцијала срца се затим мере из 12 различитих углова, а снима се у одређеном временском периоду, обично 10 секунди. На тај начин, укупна величина и смер електричне деполаризације срца ухваћени су у сваком тренутку током срчаног циклуса. График напона у односу на време које производи овај неинвазивни медицински поступак назива се **електрокардиограм**.

Током очитавања ЕКГ-а могуће је открити срчане абнормалности. Међутим, визуелни преглед и анализа ЕКГ сигнала од стране кардиолога је напорна и субјективна али била је кључна за откривање могуће срчане претње. Аутоматизовано надгледање, откривање и идентификација срчаних сигнала захтева тачну анализу у стварном времену, која је могућа помоћу алгоритама вештачке интелигенције [12].



Слика 4.1. Приказ електрокардиограма (ЕКГ) [13].

4.1.1 Скуп података

База података *MIT-BIH Arrhythmia* [14] садржи 48 двоканалних амбулантних снимака ЕКГ-а који трају пола сата, добијених од 47 испитаника који су проучавани у лабораторији за аритмију између 1975. и 1979. Двадесет и три снимка изабрана су насумично из скупа од 4000 24-часовних амбулантних снимака ЕКГ-а прикупљених у бостонској болници *Beth Israel*; преосталих 25 снимака изабрано је из истог скупа али тако да укључују мање уобичајене аритмије које не би биле добро представљене на малом случајном узорку.

Различити субјекти снимљени су на фреквенцији узорковања од 360 Hz. Сваки снимак означен је од стране минимално два кардиолога, такође сва неслагања су решена да би се добиле крајње ознаке за сваки снимак који је укључен у базу података. На основу поменутих ознака сигнали у бази могу се разврстати у 5 различитих категорија [3].

4.1.2 Претпроцесирање података

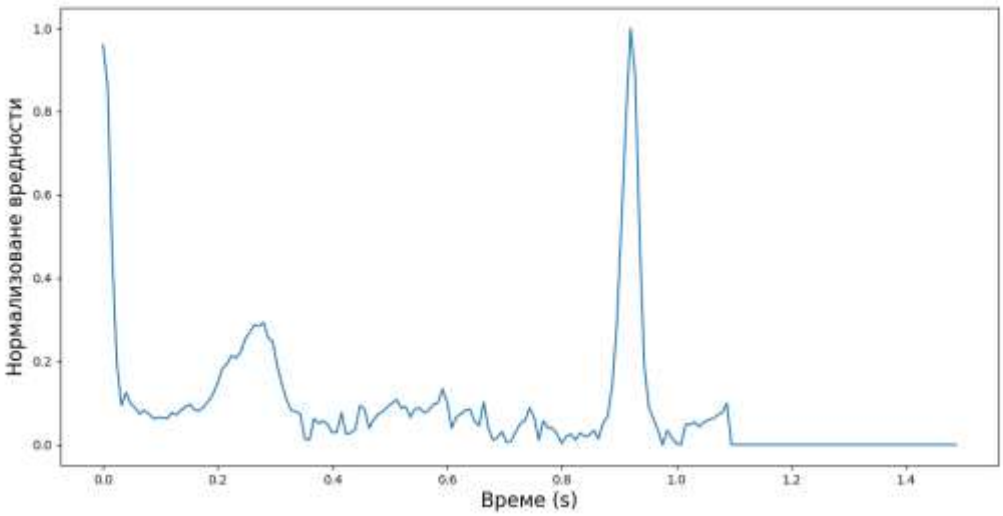
Улазни подаци за учење неуронских мрежа нису увек у облику у каквом бисмо желели да буду, зато је понекад неизбежно користити неку од метода претпроцесирања података. Желели бисмо да вредности улазних података буду нормализоване, то јест да буду у распону [0,1]. Многи модели дубоког учења захтевају улазне податке стандардних вредности, па их је потребно на неки начин скалирати. Метода претпроцесирања која је коришћена описана је у [3].

Предложена је једноставна, али ефикасна метода за издвајање откуцаја из ЕКГ сигнала. Кораци који су коришћени при претпроцесирању су следећи:

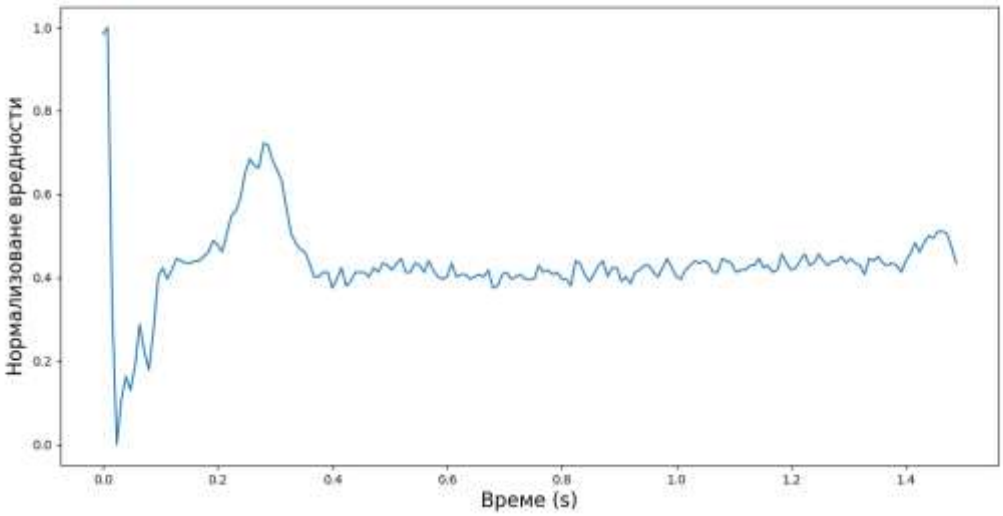
1. Дељење континуалног ЕКГ сигнала на прозоре од 10s и одабир одређеног прозора из сигнала
2. Нормализовање вредности амплитуде у распону [0,1]
3. Проналажење скупа свих локалних максимума на основу нуле функције првог извода
4. Формирање скупа ЕКГ сигнала тако да се ограниче нормализоване вредности локалних максимума на вредност 0.9
5. Налажење медијане између две вредности локалних максимума као номиналне вредности окуцаја (T)
6. За сваки локални максимум изабрати сигнал дужине 1.2T
7. Допуњавање сваког изабраног дела сигнала нулама, да би његова дужина била једнака унапред дефинисаној дужини

Предложена метода је једноставна и ефикасна у извлачењу једног откуцаја из сигнала са различитим морфологијама. Не користи се ниједан облик филтрирања или било која обрада којој је потребна било каква претпоставка о морфологији сигнала или о спектру.

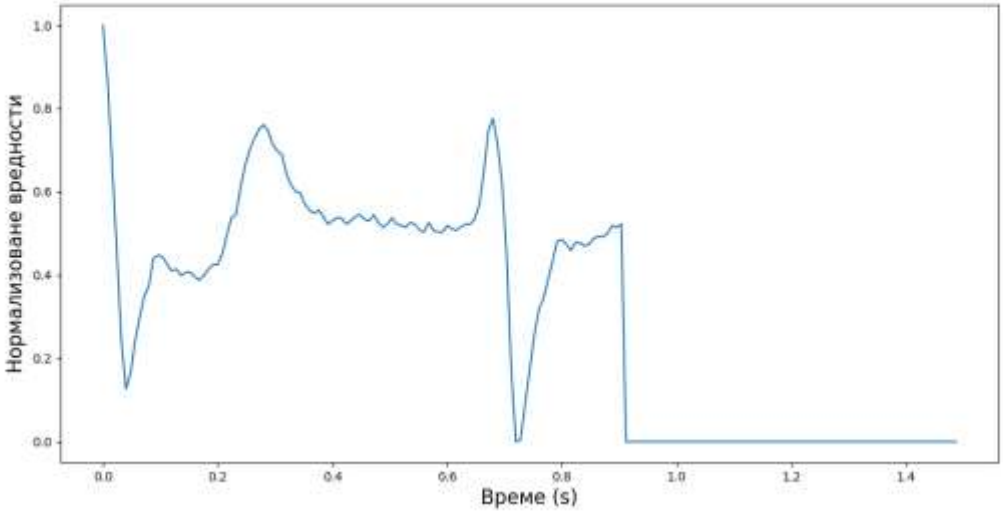
Такође, сви издвојени откуцаји имају идентичне дужине што је од суштинске важности за употребу обрађених делова. Као што је већ поменуто, сигнали се могу сврстати у пет различитих категорија. На сликама од 4.2 до 4.6 биће приказан по један претпроцесирани сигнали сваке категорије.



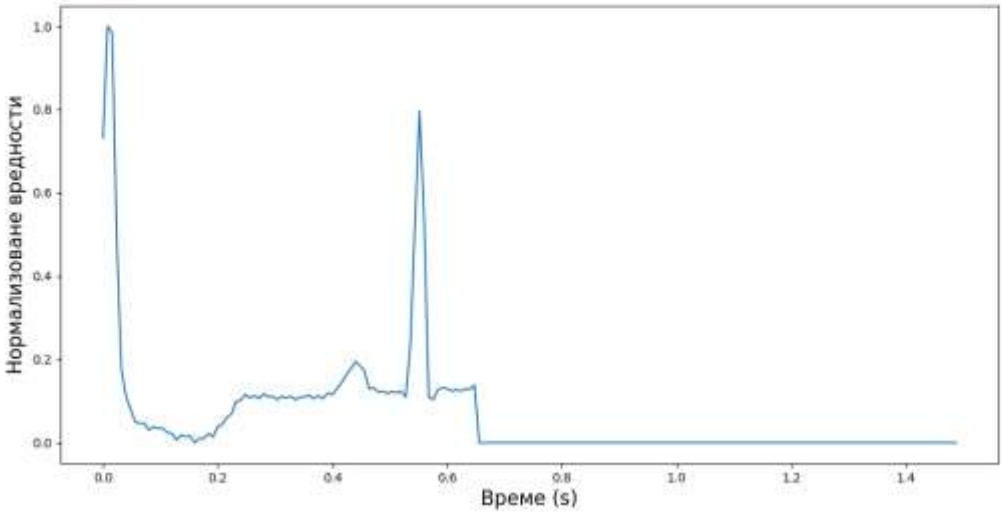
Слика 4.2. Претпроцесирани сигнал категорије N.



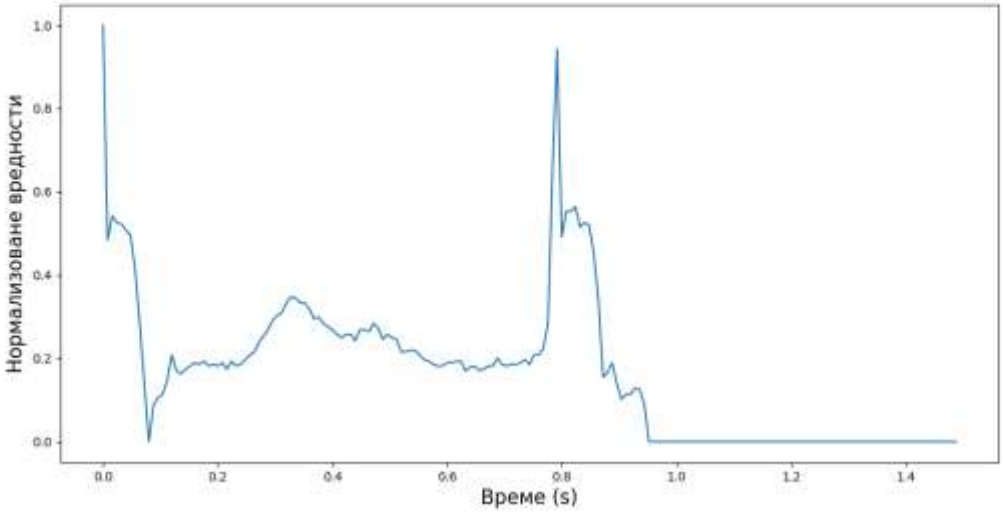
Слика 4.3. Претпроцесирани сигнал категорије S.



Слика 4.4. Претпроцесирани сигнал категорије V.



Слика 4.5. Претпроцесирани сигнал категорије F.



Слика 4.6. Претпроцесирани сигнал категорије Q.

4.2 Модел конволуционе неуронске мреже

За имплементацију решења коришћен је програмски језик *Python 3* [15], као и његова библиотека *Keras* [16]. *Keras* је библиотека отвореног кода, специјализована за неуронске мреже, која у позадини користи библиотеку, *TensorFlow* [17].

На улаз модела доводимо ЕКГ сигнал који је описан у претходном делу; улаз је вектор колона. Димензије се добијају следећим прорачуном: обрађени сигнал, издвојен из оригиналног сигнала ЕКГ-а, има временско трајање од 1496 ms, као и фреквенцију од 125Hz. Из приложеног можемо израчунати период који ће имати вредност 8 ms, односно сигнал ће бити подељен на 187 делова, па ће улаз имати димензије 187×1.

За креирање модела, као што је већ напоменуто, коришћена је библиотека *Keras*. У овој библиотеци постоје два модела неуронске мреже, секвенцијални (engl. sequential) и функционални (engl. functional). За решавање овог класификационог проблема, коришћен је секвенцијални модел, зато што је циљ употреба што једноставније конволуционе неуронске мреже, која ће притом дати задовољавајуће резултате. Функционални модели користе се за сложеније мреже, попут резидуалних (engl. ResNet)[1].

Модел је креиран методом пробе и грешке, односно на тај начин одређен је број конволуционих слојева, величине и број филтара, као и број потпуно повезаних слојева. Активациона функција у свим слојевима сем у последњем је *ReLU*, о којој је већ било речи. У последњем потпуно повезаном слоју, односно слоју класификације активациона функција је *softmax*, чија је дефиниција дата у одељку 2.3.

[1] Неуронске мреже које се састоје од тзв. резидуалних блокова, идеја ових мрежа је копирање већ научених тежина из нижих слојева и њихово додавање у слојеве за мапирање

Први конволуциони слој на улазу има вектор 187×1. У овом слоју користи се 16 филтара, величине 5. Допуњавање је валидно, што значи да се излазна активациона мапа смањује. Корак је постављен на подразумевану вредност 1. Излаз из овог слоја је вектор димензија 183×16. Број параметара, односно број тежинских коефицијената и прагова активација који се одређују током тренирања је 96. После конволуционог слоја следи слој сажимања максималном вредношћу. На улазу налази се вектор димензија 183×16. Величина филтра који се користи приликом сажимања је 2. Излаз из овог слоја је вектор димензија 91×16. Потом следи слој одбацивања са дефинисаном вредношћу функције p о којој је било речи у поглављу 3.2.5. Вредност ове функције постављена је на 0.1.

Други конволуциони слој на улазу има вектор 91×16. У овом слоју користи се 32 филтра, величине 3. Допуњавање је валидно, што значи да се излазна активациона мапа смањује. Корак је постављен на подразумевану вредност 1. Излаз из овог слоја је вектор димензија 89×32. Број параметара који се одређују током тренирања је 1568. После конволуционог слоја следи слој сажимања максималном вредношћу. На улазу налази се вектор димензија 89×32. Величина филтра који се користи приликом сажимања је 2. Излаз из овог слоја је вектор димензија 44×32. Потом следи слој одбацивања, који као и у претходном случају има вредност функције p постављену на 0.1.

Последњи, односно трећи конволуциони слој на улазу има вектор 44×32. У овом слоју користи се 64 филтра, величине 3. Допуњавање је валидно, што значи да се излазна активациона мапа смањује. Корак је постављен на подразумевану вредност 1. Излаз из овог слоја је вектор димензија 42×64. Број параметара, односно број тежинских коефицијената и прагова активација који се одређују током тренирања је 6208. После конволуционог слоја следи слој сажимања максималном вредношћу, али у овом случају користи се посебна функција *global max pooling*, која редукује димензије активационе мапе и припрема излазну активациону мапу за потпуно повезани слој. На улазу налази се вектор димензија 42×64. Величина филтра који се користи приликом сажимања је 2. Излаз из овог слоја је вектор димензија 64×1.

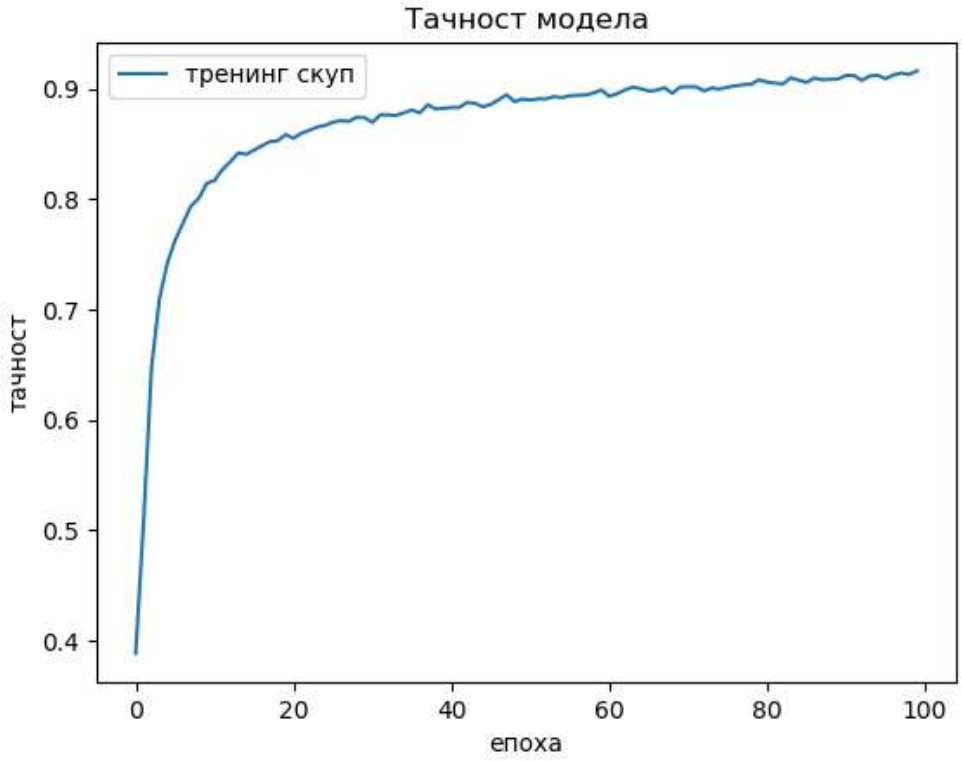
После конволуционог слоја следе потпуно повезани слојеви. У датом моделу постоје три потпуно повезана слоја. Први потпуно повезани слој има 32 неурона и укупан број тежинских коефицијената и прагова активација који се уче приликом тренирања у овом слоју је 2080. Други потпуно повезани слој има 16 неурона и број параметара које мрежа учи приликом тренирања је 528. И последњи потпуно повезани слој, уједно и излазни слој мреже има 5 неурона, односно 5 излаза. Као што је већ речено, мрежа је креирана како би решавала проблем вишекласне класификације са 5 могућих класа. У последњем слоју број параметара који мрежа учи је 85. Укупан број параметара који мрежа учи приликом тренирања износи 10565. На табели 4.1 дат је сажет приказ карактеристика описаног модела.

Табела 4.1. Коришћени модел конволуционе неуронске мреже.

Врста слоја	Својства	Улаз	Излаз	Параметри #
Конволуциони слој	$s = 1; f = 5; d = 16;$ активациона функција: $ReLU$	187×1	183×16	96
Слој сажимања	величина филтра сажимања $= 2$	183×16	91×16	
Слој искључивања	$p = 0.1$			
Конволуциони слој	$s = 1; f = 3; d = 32;$ активациона функција: $ReLU$	91×16	89×32	1568
Слој сажимања	величина филтра сажимања $= 2$	89×32	44×32	
Слој искључивања	$p = 0.1$			
Конволуциони слој	$s = 1; f = 3; d = 64;$ активациона функција: $ReLU$	44×32	42×64	6208
Слој сажимања	величина филтра сажимања $= 2$	42×64	64×1	
Потпуно повезани слој	активациона функција: $ReLU$	64×1	32×1	2080
Потпуно повезани слој	активациона функција: $ReLU$	32×1	16×1	528
Потпуно повезани слој	активациона функција: *softmax*	16×1	5×1	85

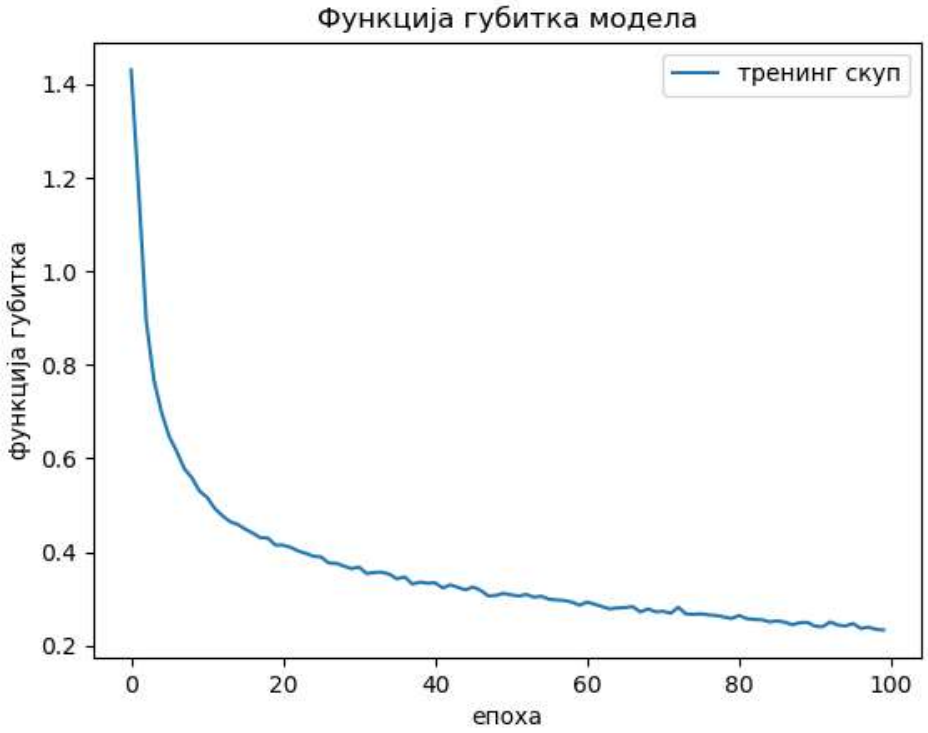
4.3 Тренирање и резултати модела

За тренирање модела коришћене су уграђене функције библиотеке *Keras*. Код за тренирање модела налази се у одељку 7.1. Првенствено се користи функција *compile* да се конфигурише модел за тренинг. Параметри поменуте функције су оптимизациони алгоритам *Adam*, који се базира на градијентном спусту и функција губитка категоричка унакрсна ентропија (engl. categorical cross entopy). Категоричка унакрсна ентропија је врста унакрсне ентропије приликом које сваки податак може припадати само једној класи. Функција која се користи за тренирање модела је *fit*. Параметри ове функције су величина подскупа који се користи приликом учења неуронске мреже (engl. batch size) и у овом случају има вредност 64. Други параметар је број епоха[2], који има вредност 100 и последњи параметар означава колико података ће бити издвојено из тренинг скупа за валидацију. У овом случају подела је 80% за тренинг, 20% за валидацију. Резултати тренирања представљени су на сликама 4.7 и 4.8.



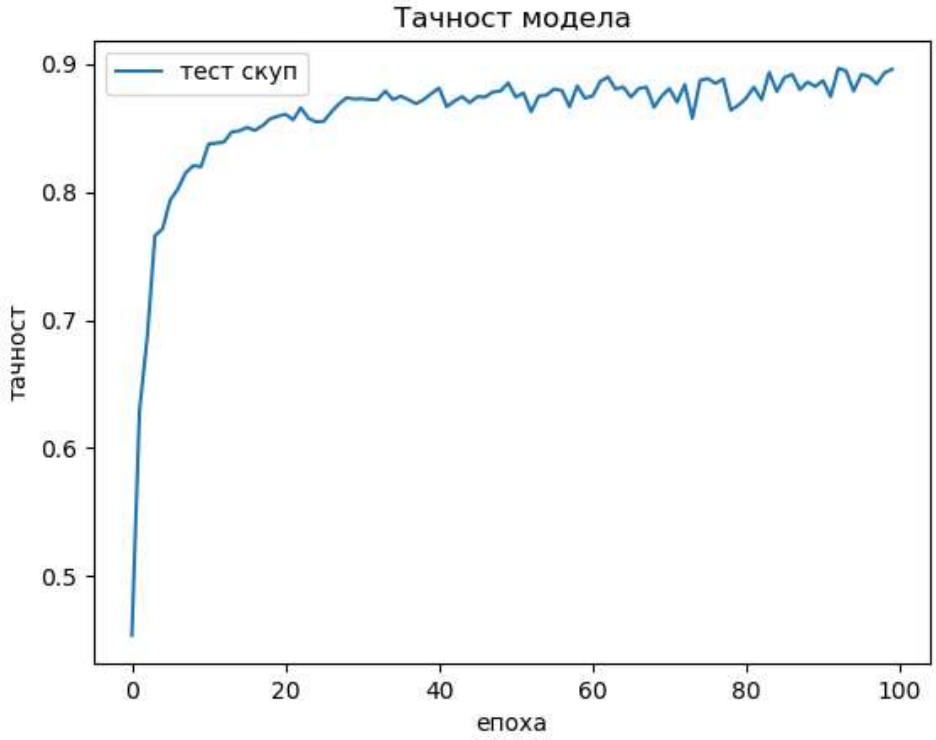
Слика 4.7. Тачност модела на тренинг скупу.

[2] Представља број који означава колико ће пута сви подаци из тренинг скупа проћи кроз неуронску мрежу.

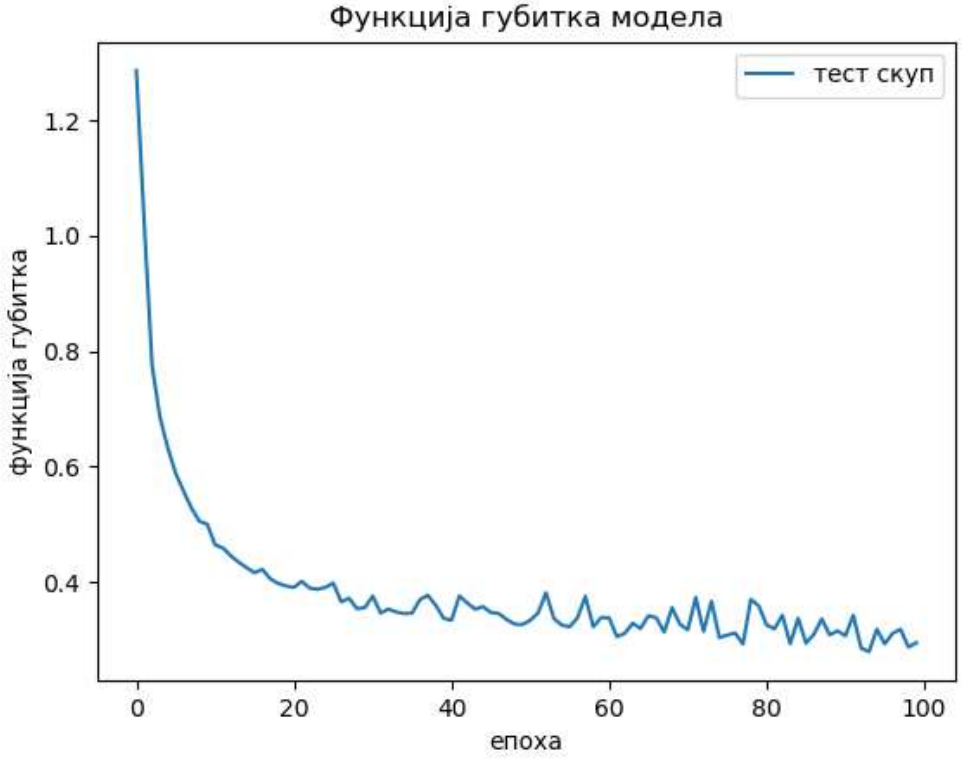


Слика 4.8. Функција губитка на тренинг скупу.

Након што је модел истрениран, потребно је тестирати га на тест скупу. Тачност модела и функција губитка на тест скупу приказане су на сликама 4.9 и 4.10.



Слика 4.9. Тачност модела на тест скупу.



Слика 4.10. Функција губитка на тест скупу.

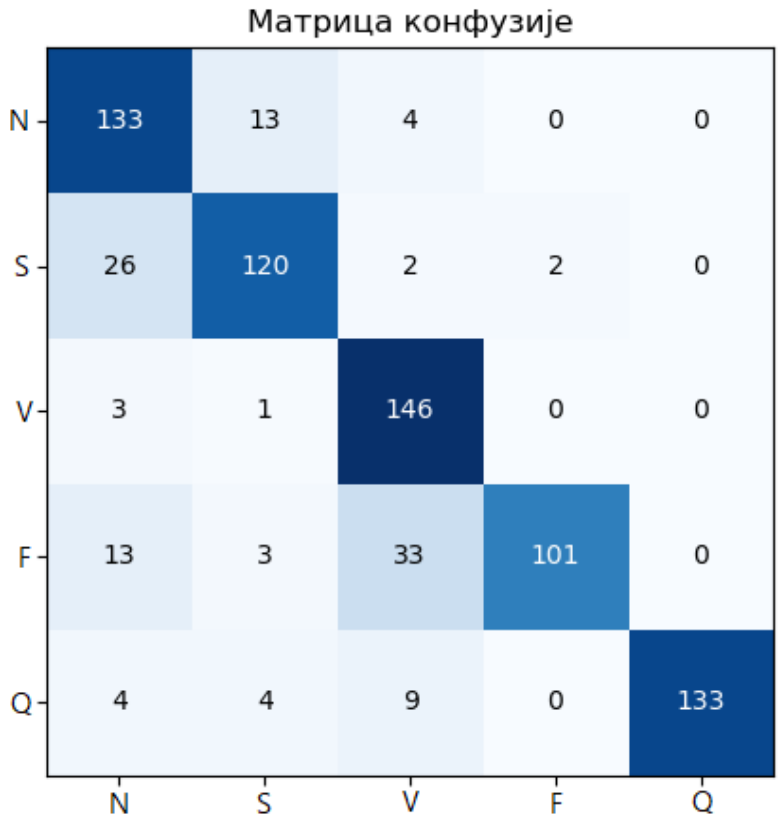
Како су подаци небалансирани, није довољно тумачити само тачност модела, већ је неопходно анализирати и прецизност (engl. precision), осетљивост (engl. recall) и F1 оцену (engl. F1 score). На табели 4.2 приказани су поменути параметри за сваку класу скупа података. Ове параметре могуће је израчунати помоћу матрице конфузије.[3] Матрица конфузије за дати скуп података представљена је на слици 4.11.

Прецизност се може израчунати уколико се број елемената који су тачно предвиђени као позитивна класа подели са укупним бројем елемената који су предвиђени као позитивна класа, што укључује и тачно и нетачно предвиђене елементе.

Осетљивост се може израчунати уколико се број елемената који су тачно превиђени као позитивна класа подели са збиром елемената који су предвиђени тачно као позитивна класа и елемената који су нетачно предвиђени као негативна класа.

$$F1 = 2 \cdot \frac{\text{прецизност} \cdot \text{ осетљивост}}{\text{прецизност} + \text{ осетљивост}}$$

[3] Специфична матрица која омогућава преглед перформанси алгоритма. Сваки ред матрице представља инстанце у предвиђеној класи, док свака колона представља инстанце у актуелној класи, или обрнуто.



Слика 4.11. Матрица конфузије.

Табела 4.2. Приказ осетљивости, прецизности и F1 оцене за свих 5 могућих класа.

	осетљивост	прецизност	F1 оцена
Класа *N*	0.91	0.72	0.80
Класа *S*	0.78	0.91	0.84
Класа *V*	0.94	0.82	0.87
Класа *F*	0.75	0.94	0.84
Класа *Q*	0.92	1.00	0.96

5 Закључак

Циљ овог рада био је креирати једноставан модел који ће успешно решавати проблем класификације сигнала. Притом је коришћена библиотека *Keras* која је специјалозована за тренирање дубоких неуронских мрежа. Било је потребно проучити како се користи библиотека, разумети методе претпроцесирања оригиналног сигнала, затим спровести поступак учења конволуционе неуронске мреже и описати добијене резултате. Како је истраживање спроведено по угледу на чланак [3] у којем се примењује резидуална конволуциона неуронска мрежа, главни циљ овог рада био је да се креира једноставнији модел за рад са истим скупом података. Као што видимо просечна тачност на једноставном моделу је 89%, што за пар процената ниже него на моделу са резидуалним блоковима, где је просечна тачност 93.4%. Међутим, главни проблем је небалансираност података, зато су испитани и параметри попут осетљивости, прецизности и F1 оцене. Просечна вредност осетљивости на тест скупу је 86%, просечна вредност прецизности је 87% и F1 оцена је 86.2%. Може се приметити да модел боље класификује класе *N, V* и *Q* зато што је број узорака сигнала који припадају овим категоријама у тренинг скупу далеко већи од броја узорака преостале две класе *S* и *F*. Током рада на развоју модела, покушавано је балансирање података, међутим услед недостатка оригиналних сигнала, небалансираност није искорењена. Из тог разлога, модел даје лошије резултате приликом класификације ове две класе.

6 Литература

[1] O. Yildirim , R. S. Tan, R. U. Acharya, An efficient compression of ECG signals using deep convolutional autoencoders, Cognitive Systems Research, vol. 52, July 2018., pp. 198-211.

[2] J. Zhicheng, G. Xinbo, W. Ying, L. Jie, X. Haojun, Deep Convolutional Neural Networks for mental load classification based on EEG data, Pattern Recognition, vol. 76, April 2018., pp. 582-595.

[3] M. Kachuee, S. Fazeli, M. Sarrafzadeh, ECG Heartbeat Classification: A Deep Transferable, 2018 IEEE International Conference on Healthcare Informatics (ICHI), July 2018.

[4] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, D. J. Inman, 1D Convolutional Neural Networks and Applications – A Survey, 2019.

[5] W. S. McCulloch, W. H. Pitts, A Logical Calculus of the Ideas Immanent, Bulletin of Mathematical Biophysics, vol. 5, 1990., pp. 99-115

[6] M. A. Nielsen, Neural Networks and Deep Learning, Determination Press, 2015. http://neuralnetworksanddeeplearning.com/ [Последњи приступ: 19. Септембар 2019.]

[7] A. Sharma, Learn OpenCV, 2017. https://www.learnopencv.com/understanding-activation-functions-in-deep-learning/ [Последњи приступ: 19. Септембар 2019.]

[8] O. Liu, Python tutorial on Linear Regression with Batch Gradient Descent, 2016 https://ozzieliu.com/2016/02/09/gradient-descent-tutorial/ [Последњи приступ: 19. Септембар 2019.]

[9] F. Chollet, Deep Learning with Python, Shelter Island, New York, Manning Publications, 2018., ISBN 9781617294433

[10] S. Pattanayak, Pro Deep Learning with Tensorflow, Apress, 2017., ISBN 978-1-4842-3096-1, pp. XXI, 398

[11] М. М. Дабовић, И. И. Тартаља, Дубоке конволуцијске неуронске мреже - концепти и актуелна истраживања, Зборник 61. Конференције за електронику, телекомуникације, рачунарство, аутоматику и нуклеарну технику, ETPAH 2017, Кладово, 05. до 08. јуна 2017., ISBN 978-86-7466-692-0, стр. VI1.1.1-6

[12] D. E. Becker, Fundamentals of Electrocardiography Interpretation, Anesthesia Progress, vol. 53, 2006., pp. 53-64

[13] T. L. Teigeler, K. A. Ellenbogen, S. K. Padala, The Right-Sided ECG for the Right Diagnosis, American Heart Association, vol. 138, Jul 2018., pp. 107-109

[14] PhysioNet https://www.physionet.org/content/mitdb/1.0.0/ [Последњи приступ: 19. Септембар 2019.]

[15] Python https://www.python.org [Последњи приступ: 17. Септембар 2019.]

[16] Keras Documentation https://keras.io [Последњи приступ: 17. Септембар 2019.]

[17] TensorFlow https://www.tensorflow.org [Последњи приступ: 17. Септембар 2019.]

7 Додатак

7.1 Део кода за тренирање модела конволуционе неуронске мреже

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from keras import activations
from keras.models import Sequential
from keras.layers import Dense, Dropout, Conv1D, MaxPooling1D, GlobalMaxPool1D,
df_train = pd.read_csv("/mitbih_train.csv", header=None)
df_train = df_train.sample(frac=1)
df_test = pd.read_csv(“ /mitbih_test.csv", header=None)
Y = np.array(df_train[187].values).astype(np.int8)
X = np.array(df_train[list(range(187))].values)[..., np.newaxis]
Y_test = np.array(df_test[187].values).astype(np.int8)
X_test = np.array(df_test[list(range(187))].values)[..., np.newaxis]
model = Sequential()
model.add(Conv1D(16, kernel_size=5, activation=activations.relu,
input_shape=(187, 1),   padding="valid"))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(rate=0.1))
model.add(Conv1D(32, kernel_size=3, activation=activations.relu, padding="valid"))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(rate=0.1))
model.add(Conv1D(64, kernel_size=3, activation=activations.relu, padding="valid"))
model.add(GlobalMaxPool1D())
model.add(Dense(32, activation=activations.relu))
model.add(Dense(16, activation=activations.relu))
model.add(Dense(5, activation=activations.softmax))
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['acc'])
history = model.fit(X, Y, batch_size = 64, epochs=100, validation_split=0.2)
pred_test = model.predict(X_test)
pred_test = np.argmax(pred_test, axis=-1)
```