ПРИМЕНА ПРОГРАМСКОГ ЈЕЗИКА PYTHON У УЧЕЊУ ВЕШТАЧКИХ НЕУРОНСКИХ МРЕЖА

Перица Штрбац[1] Павле Штрбац [2] Милош Пејановић [3] Стефан Пејановић [4] Вукман Корађ [5]

Резиме: У оквиру предмета Програмирање у интегрисаним технологијама предвиђена је лекција у којој се учи примена програмског језика *Python* за решавање задатака из области вештачких неуронских мрежа. Претходне лекције предмета обучавају студенте да користе *Python* за решавање програмских задатака који се односе на базе података, нити, метапрограмирање, мрежно програмирање, математичка израчунавања и обраду слике. У нашем случају за примену програмског језика *Python* за учење вештачких неуронских мрежа коришћене су библиотека *TensorFlow* и апликативни програмски интерфејс *Keras*. У овом раду је моделовање на највишем нивоу апстракције урађено помоћу Надграђених Петри-мрежа. Циљ овог рада је да се прикаже један приступ обучавања студената за решавање задатака који се односе на примену вештачких неуронских мрежа коришћењем програмског језика *Python* те *TensorFlow* и *Keras*. Дат је практичан пример који се показује студентима и који се односи на решавање проблема класификације над одабраним скупом за тренирање и тестирање који се испоручује уз *Keras*. Након предавања студенти у оквиру вежби у двочланим тимовима раде задатак који је сличан задатку показаном на предавању.

Кључне речи: *Python*, вештачке неуронске мреже, *TensorFlow*, *Keras*, Надграђене Петри-мреже

USAGE OF PYTHON PROGRAMMING LANGUAGE IN LEARNING ARTIFICIAL NEURAL NETWORKS

Abstract: One lesson in the course "Programming in integrated technologies" refers to solving tasks in the field of artificial neural networks using the Python programming language. Previous course lessons train students to use Python to solve program tasks related to databases, threads, metaprogramming, network programming, mathematical calculations, and image processing. In our case, the TensorFlow library and Keras application programming interface were used to apply the Python programming language for learning artificial neural networks. In this paper, modeling at the highest level of abstraction is done using Upgraded Petri nets. This paper's aim is to present an approach to training students to solve tasks related to the application of artificial neural networks using the Python programming language and the TensorFlow and Keras. A practical example is given to the students and which refers to solving the problem of classification over the set that is delivered with Keras. After the lecture, students within the exercises in two-member teams do a task that is similar to the task shown in the lecture.

Key words: Python, artifical neural networks, TensorFlow, Keras, Upgraded Petri-nets

1. УВОД

Вештачке неуронске мреже припадају области вештачке интелигенције која се све више тражи на ИТ тржишту за решавање проблема класификације, категоризације и регресије. На Академији техничко-уметничких струковних студија на одсеку Висока школа електротехнике и рачунарства у Београду у оквиру предмета Програмирање у интегрисаним технологијама и Меко рачунарство слуша се тема о вештачким

[1] Професор, АТУСС – Одсек Висока школа електротехнике и рачунарства, Војводе Степе 283, Београд, pericas@viser.edu.rs:

[2] Дипл.инж, TMS d.o.o, Milutina Milankovića 11B, Београд, strbacpavle98@gmail.com:

[3] Предавач, АТУСС – Одсек Висока школа електротехнике и рачунарства, Војводе Степе 283, Београд, pejanovicm@viser.edu.rs:

[4] Дипл. инж, NewCo. – Научно-технолошки парк, Вељка Дугошевића 54, Београд., stefanpejanovic604@gmail.com:

[5] Асистент, АТУСС – Одсек Висока школа електротехнике и рачунарства, Војводе Степе 283, Београд, vkorac@viser.edu.rs:

неуронским мрежама. Интегрисане технологије које се користе за ову тему су: *Python*, *TensorFlow* и *Keras*.

Програмски језик *Python* је основни алат наведених предмета, а овде је коришћен због одличних и популарних модула програмске подршке за вештачке неуронске мреже. *TensorFlow* се у оквиру лекције користи као најнижа ниво за дистрибуирано извршавање програма на процесору или гафичкој картици [1]. *Keras* је употребљен као апликативни програмски интерфејс за креирање модела вештачке неуронске мреже који обухвата дефинисање улаза, слојева, излаза, активационе функције, лос функције те оптимизатора вештачке неуронске мреже и као спрега ка *TensorFlow* [2].

Лекција о вештачким неуронским мрежама се за потребна додатна знања ослања на стечено знање из претходних лекција у оквиру наведеног предмета које се првенствено односе на: рад са датотекама, графички кориснички интерфејс, математичка израчунавања и обраду слике.

Идеја је да студенти у двочланим тимовима на предавањима о вештачким неуронским мрежама ураде практичан пример креирања модела за дати проблем, тренирање вештачке неуронске мреже на скупу за тренинг те проверу тачности на тестном скупу те врше модификацију модела вештачке неуронске мреже у циљу њеног побољшања. Студенти примењујући знања из претходних лекција приказују и анализирају добијене резултате нумерички и графички. Након предавања, на припадним вежбама исти двочлани тимови добијају нов проблем који би требало да реше у оквиру вежби по узору на знање усвојено на предавању.

У овом раду је коришћена оригинална класа Петри-мрежа, Надграђене Петри-мреже (*Upgraded Petri-Nets - UPN*) за графовски приказ модела [3]. За креирање *UPN* модела коришћен је оригинални програмски пакет *PeM* (*Petri-net Manager*).

2. НАДГРАЂЕНЕ ПЕТРИ-МРЕЖЕ

Надграђене Петри-мреже чији су детаљи дати у [3] намењене су моделовању, симулацији и анализи комплексних система, посебно где су потребни паралелизам и синхронизација. Овде се даје само формална дефиниција. Надграђена Петри-мрежа је уређена деветорка: $C = (P, T, F, B, \mu, \theta, TF, TFL, PAF)$ где су:

$P=\{p_1, p_2, ..., p_n\}, n>0$ – коначан непразан скуп места p_i;
$T=\{t_1, t_2, t_3, ...,t_m\}, m>0$ – коначан непразан скуп прелаза t_j;
$F : T\times P \rightarrow N_0$ – улазна функција;
$B : T\times P \rightarrow N_0$ – излазна функција;
$\mu : P \rightarrow N_0$ – функција маркирања:
$\theta : T\times N_0 \rightarrow [0,1]$ – временска функција;
$TF: T\rightarrow A$ – функција прелаза;
$TFL: T \rightarrow N_0$ – функција нивоа паљења прелаза;
$PAF: P \rightarrow (x, y)$ – функција атрибута места;
$N_0 = \{0,1,2,...\}$ – скуп природних бројева проширен елементом нула;
A – скуп функција које се могу доделити прелазу
x – атрибут места p_i који моделује вредност операнда у датој функцији прелаза t_j где вреди $F(t_j, p_i)>0$;
y – атрибут места p_i који моделује редни број операнда у датој функцији прелаза t_j где вреди $F(t_j, p_i)>0$;

Ако прелазу t није додељена функција прелаза онда се *UPN* понаша као изворна Петри-мрежа.

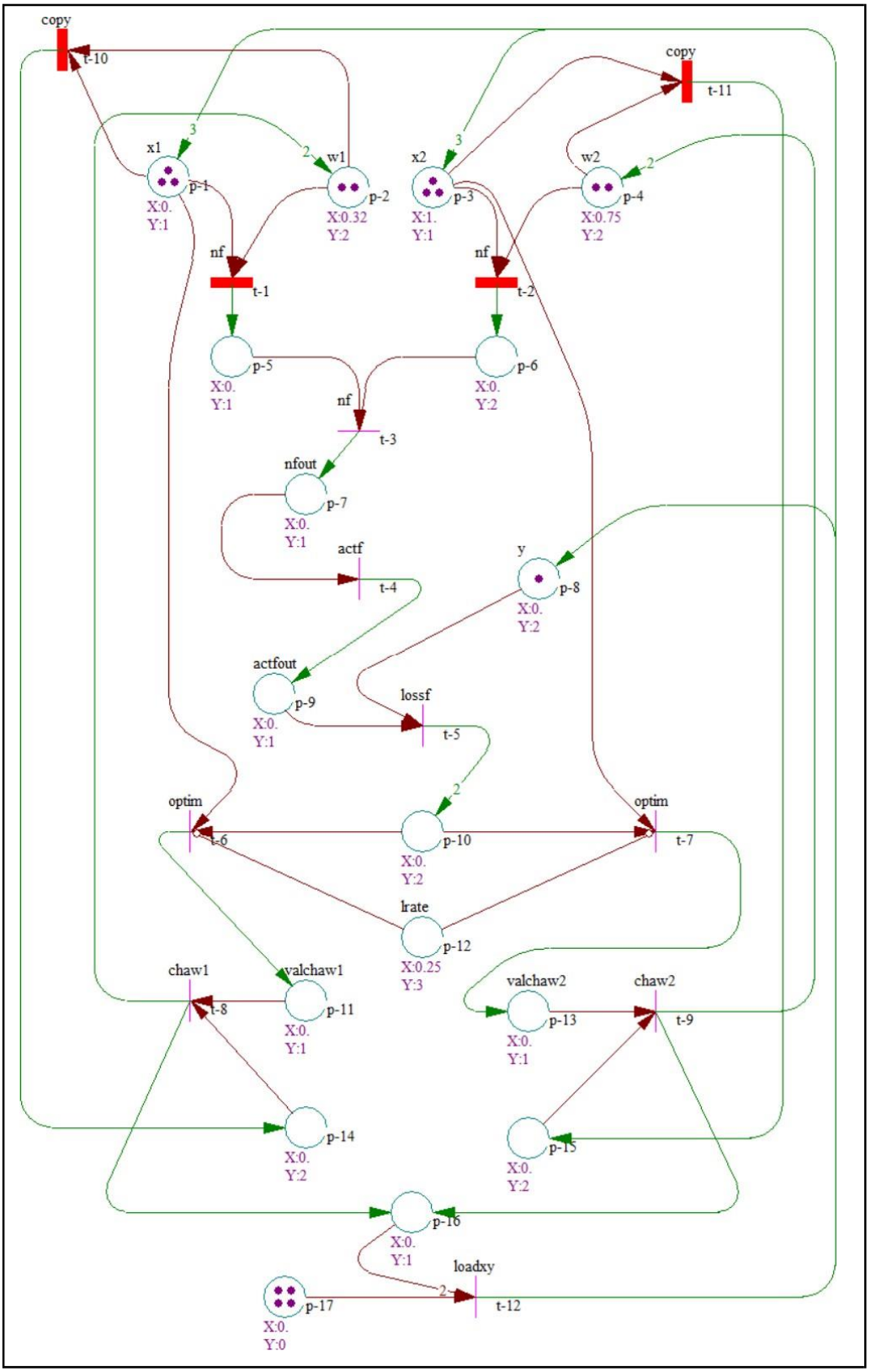
3. ВЕШТАЧКА НЕУРОНСКА МРЕЖА

Вештачке неуронске мреже развијене су по узору на биолошке неуронске мреже [4]. Погодне су за решавање проблема када су подаци неизвесни и непотпуни. У општем случају неурон има *n* улаза, дакле, вектор *X* представљен н-торком вредности $(x_1, x_2, ..., x_{n-1}, x_n)$ при чему је сваки улаз x_i пондерован тежинским фактором w_i. Комбиновање улаза и пондера ради функција мреже над чијим резултатом се примењује активациона фунцкија која даје као резултат вредност излаза неурона. Над излазном вредности неурона и очекиваној вредности *y* за дате вредности улаза делује функција губитка или функција грешке (*loss function*). Идеја је да се излаз ове функције користи са датим коефицијентом учења вештачке неуронске мреже (*learning rate*) и припадном вредности улаза x_i да би се израчунала корекција припадног тежинског фактора w_i. Након овога се врши корекција тежинског фактора, учитава нови улаз и припадни очекивани излаз и иде у следећи циклус обучавања неуронске мреже. У овом итеративном поступку вештачка неуронска мрежа „учи“ над тренинг скупом улаза и очекиваних излаза. Након учења над тренинг скупом следи провера квалитета вештачке неуронске мреже што се провере њени резултати над тестним скупом улаза и очекиваних излаза. Тестни скуп и тренинг скуп су дисјунктни. Побољшавање квалитета вештачке неуронске мреже иде променом параметара који учествију у циклусу рачунања (функција мреже, активациона функција, фунцкија губитка, тип мреже, број слојева мреже, начин повезивања неурона, коефицијент обучавања, оптимизатор) те поновним учењем над тренинг скупом и провером над тест скупом док се не дође до задовољавајуће коректности вештачке неуронске мреже.

3.1. UPN модел вештачког неурона

На слици 1 дат је *UPN* модел вештачког неурона као основног елемента вештачке неуронске мреже. Почетно означавање модела је $\mu_0(3,2,3,2,0,0,0,1,0,0,0,0,0,0,0,0,4)$. Места *x1* и *x2* моделују вектор улаза, док места *w1, w2* моделују припадне тежинске факторе. Прелази *nf* моделују функцију мреже. Место *nfout* моделује излазну вредност мрежне фунцкије које представља улаз у прелаз *actf* којим је моделована активациона фунцкија. Место *actfout* (*p-9*) моделује резултат активационе фунцкије. Сада су места *actfout* и *y* улази прелаза *lossf* (*t-5*) који моделује рачунање вредности грешке. Следи комбиновање вредности грешке (место *p-10*), датог улаза (места *p-1, p-2*) и коефицијента обучавања вештачке неуронске мреже (*lrate*, место *p-12*) да би се израчунале корекције за тежинске факторе *w1, w2* (респективно места *valchaw1 p-11* и *valchaw2 p-13*). За синхронизацију *UPN* модела коришћени су прелази *t-10, t-11* који служе за копирање вредности *w1, w2* које су потребне за рачунање нових коригованих вредности *w1* и *w2*. Прелази *chaw1* і *chaw2* врше корекцију тежинских фактора *w1* и *w2*. Да би наставило са учење над новим вектором *X* и припадном очекиваном излазу извршена је синхронизација местом *p-16* које моделује да је извршена корекција тежинских фактора и да може да се учита нови вектор *X* и припадни очекивани излаз *y*. Место *p-17* моделује број улазних узорака (вектора *X*). Паљењем прелаза *t-12* учитавају се нови *X* и припадни *y*. След паралеленог паљења прелаза је као што следи: *{t-1, t-2, t-10, t-11}, {t-3}, {t-4}, {t-5}, {t-6, t-7}, {t-8, t-9}, {t-12}*. Паљење више од једног прелаза моделује паралелизам. Овај циклус се понавља онолико пута колико је маркирање места *p-17*. Свуда где је излазна функција $B(t_i,p_j)>1$ моделован је паралелизам. Припадни след означавања за дати *UPN* модел је: $\mu(3,2,3,2,0,0,0,1,0,0,0,0,0,0,0,0,4)$, $\mu(1,0,1,0,1,1,0,1,0,0,0,0,0,1,1,0,4)$, $\mu(1,0,1,0,0,0,1,1,0,0,0,0,0,1,1,0,4)$, $\mu(1,0,1,0,0,0,0,1,1,0,0,0,0,1,1,0,4)$, $\mu(1,0,1,0,0,0,0,0,0,2,0,0,0,1,1,0,4)$, $\mu(0,2,0,2,0,0,0,0,0,0,1,0,1,1,1,0,4)$, $\mu(0,2,0,2,0,0,0,0,0,0,0,0,0,0,0,2,4)$, $\mu(0,2,0,2,0,0,0,0,0,0,0,0,0,0,0,2,4)$, $\mu(3,2,3,2,0,0,0,1,0,0,0,0,0,0,0,0,0,3)$. Оваквих итерација има онолико колико

је вредност *μ(p-17).* На крају сваког итеративног циклуса смањује се вредност *μ(p-17)* и то траје док се не дође до мртвог чвора *μ(3,2,3,2,0,0,0,1,0,0,0,0,0,0,0,0,0)* који моделује да је вештачка неуронска мрежа обрадила све улазне векторе (све слике) и да је једна епоха обучавања вештачке неуронске мреже завршена.



Слика 1 – Модел вештсчког неурона

3.2. Функција мреже

Студентима се приказују одабране функције мреже које делују над улазима и тежинским факторима. Овде су дате две функције мреже: линеарна и производ степеновања (формуле 1 и 2 респективно).

$$u = f_m(X, W) = \sum_{i=1}^{N} w_i x_i + \theta \quad (1)$$

$$u = f_m(X, W) = \mathrm{G}\, x_i^{w_i} + \theta \quad (2)$$

Величина θ представља *bias* (праг) и уводи се да би се решио проблем ако су сви улази једнаки 0 (практично улаз који је стално једнак 1 и који има припадни w_0).

3.3. Активациона функција

Активациона функција као улаз прихвата резултат функције мреже и даје излазну вредност неурона за дати улаз. Излазна вредност неурона је најчешће нормализована у интервалима [0,1] ili [-1,1]. Студентима се даје приказ одабраних активационих функција: линеарна, униполарни сигмоид, биполарни сигмоид, *relu* (*rectified linear activation unit*) и *softmax* које су дате формулама 3, 4, 5, 6 и 7, респективно.

$$f_a(u) = u \quad (3)$$

$$f_a(u) = \frac{1}{1 + e^{-u}} \quad (4)$$

$$f_a(u) = \frac{2}{1 + e^{-2u}} - 1 \quad (5)$$

$$f_a(u) = \begin{cases} u, & u > 0 \\ 0, & u \leq 0 \end{cases} \quad (6)$$

$$f_a(U)_i = \frac{e^{u_i}}{\sum_{j=0}^{n-1} e^{u_j}} \quad (7)$$

Вектор U представља вектор чија је дужина n и који представља улаз за функцију *softmax*.

4. ВЕШТАЧКА НЕУРОНСКА МРЕЖА ЗА КЛАСИФИКАЦИЈУ

Да би се реализовала вештачка неуронска мрежа коришћењем програмског језика *Python* студенти инсталирају као што следи: *Python* 64-bit v.3.8.5, *TensorFlow v.2.7.0.* и *Keras v.2.7.0*. Након инсталације потребно је креирати модел вештачке неуронске мреже. Задатак је да се модел креира коришћењем готових тренинг и тест скупова који се испоручују уз *Keras*. Одабрано је да студенти креирају вештачку неуронску мрежу која решава проблем класификације ручно написаних цифара од 0 до 9 [5]. Уз *Keras* се испоручује *MNIST (Modified National Institute of Standards and Technology database)* скуп података који садржи: скуп за тренирање који има 60000 слика ручно написаних цифара за тренирање са припадним вектором тачних вредности и скуп за тестирање који има 10000 слика ручно написаних цифара са припадним вектором тачних вредности. Слике су резолуције 28×28 пиксела (28·28=784 пиксела). Код приказан на слици 2 учитава наведени скуп слика и креира два пара *(train_images, train_labels)* и *(test_images, test_labels)*. Пар (train_images, train_labels) представља тренинг пар кога чине тродимензионална матрица величине 60000×28×28 и припадни вектор тачних вредности

дужине 60000. Пар *(test_images, test_labels)* представља тест пар кога чине тродимензионална матрица величине 10000×28×28 и припадни вектор тачних вредности дужине 10000. Оно што је нама потребно је улазни вектор буде димензија 1×784 (примена методе *reshape*) што значи да један улаз вештачке неуронске мреже прихвата вредност једног пиксела на датој слици те да те вредности пиксела буду нормализоване од 0 до 1 (примена дељења са 255.0). Ово се уради и за тренинг скуп и за тест скуп. Следеће је да код креира архитектуру вештачке неуронске мреже која се састоји од 2 потпуно повезана слоја (*Dense*) где први слој има 784 улаза и 512 излаза који представљају улаз у следећи слој који има 10 излаза. Први слој користи активациону функцију *relu*, док други слој користи активациону функцију *softmax*. Овде ће излаз фунцкије *softmax* бити вектор од 10 вероватноћа чија је сума свих елемената 1. Ове вероватноће на индексу вектора од 0 до 9 представљају резултат рада мреже у смислу колико улазна слика личи на цифру редом од 0 до 9. За компајлирање модела постављене су следеће вредности: оптимизаторска фунцкија је *RMSprop* (са коефицијентом обучавања вештачке неуронске мреже једнаким 0.01), функција губитка је *sparse categorical crossentropy* и функција мерења је *sparse categorical accuracy*. Оптимизатор *RMSprop* (*root mean square propagation*) служи за постизање минималне вредности функције губитка. За функцију губитка је коришћена *sparse categorical crossentropy* функција (пошто имамо 10 цифара) која делује над предикцијом (цифра протумачена од вештачке неуронске мреже на основу улаза *X*) и тачном цифром датом у припадној *y* вредности (лабели). Функција мерења тачности *sparse categorical accuracy* мери број погодака где предикција одговара припадној тачној вредности.

```
from tensorflow.keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_images = train_images.reshape(train_images.shape[0],-1)
train_images = train_images / 255.0
test_images = test_images.reshape(test_images.shape[0],-1)
test_images = test_images / 255.0
from tensorflow import keras
from tensorflow.keras import layers
model = keras.Sequential([
    layers.Dense(512, input_shape=(784,), activation="relu"),
    layers.Dense(10, activation="softmax")
])
from tensorflow.keras import optimizers
from tensorflow.keras import losses
from tensorflow.keras import metrics
model.compile(optimizer=optimizers.RMSprop(learning_rate=0.005),
              loss=losses.sparse_categorical_crossentropy,
              metrics=[metrics.sparse_categorical_accuracy])
history = model.fit(train_images, train_labels, epochs=10, batch_size=512,
                    validation_data=(test_images, test_labels))
print(history.history.keys()) # da studenti vide dostupne kljuceve
# za odvojenu validaciju (nenavodjenjem validation_data) i ispis vrednosti
# greske i tacnosti kod testiranja koristiti sledece 2 linije koda
#test_loss, test_accuracy = model.evaluate(test_images, test_labels)
#print("test_loss is {}\ntest_accuracy is {}".format(test_loss, test_accuracy))
```

Слика 2 – Python код вештачке неуронске мреже за класификацију

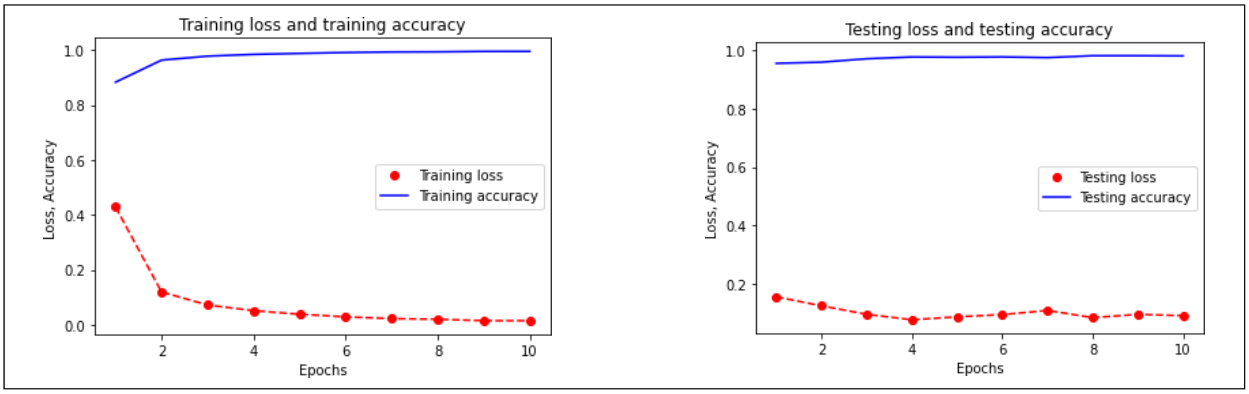
Након овога следи обучавање вештачке неуронске мреже где се методи *fit* прослеђују редом: матрица слика за тренинг димензија 60000×784, припадни вектор тачних одговора чија је дужина 60000, 10 епоха обучавања, узимање по 512 слика за обучавање (што

тражи мање меморије за обраду) и тако све до 60000 слика, навођењем података за валидацију (тестирање) да се уради над тест скупом. Метода *fit* враћа објекат који садржи речник *hystory* који ће према коду на слици 2 имати следеће кључеве: *loss*, *sparse_categorical_accuracy*, *val_loss* и *val_sparse_categorical_accuracy*. Евалуација над тест скупом може се урадити и посебно методом *evaluate* која враћа грешку губитка и тачност деловања вештачке неуронске мреже над тест скупом.

Сада је идеја да студенти коришћењем знања из раније лекције која обрађује *matplotlib* сликовито прикажу вредности 4 кључа речника *history*, дакле, грешке и тачности након сваке епохе при обучавању и при тестирању вештачке неуронске мреже, респективно (слике 3 и 4). Увидом у добијене податке, студенти експериментишу мењањем кода на слици 2 (мењање параметара мреже) те стартовањем тог кода и кода на слици 3. На основу добијених података (нумеричких и графичких) циљ студента је да постигне тражену тачност вештачке неуронске мреже.

```
import matplotlib.pyplot as plt
loss_values = history.history["loss"]
accuracy_values = history.history["sparse_categorical_accuracy"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "r--")
plt.plot(epochs, loss_values, "ro", label="Training loss")
plt.plot(epochs, accuracy_values, "b", label="Training accuracy")
plt.title("Training loss and training accuracy")
plt.xlabel("Epochs");plt.ylabel("Loss, Accuracy ")
plt.legend();plt.show()
plt.clf()
loss_values = history.history["val_loss"]
accuracy_values = history.history["val_sparse_categorical_accuracy"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "r--")
plt.plot(epochs, loss_values, "ro", label="Testing loss")
plt.plot(epochs, accuracy_values, "b", label="Testing accuracy")
plt.title("Testing loss and testing accuracy")
plt.xlabel("Epochs");plt.ylabel("Loss, Accuracy")
plt.legend(); plt.show()
```

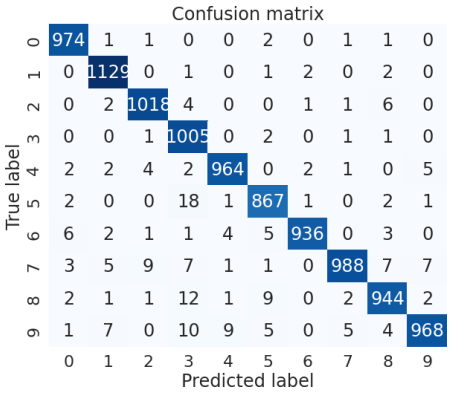
Слика 3 – Python код за приказ тачности вештачке неуронске мреже након сваке епохе



Слика 3 – Излаз извршавања Python кода са слике 3

Следеће је да студенти виде матрицу конфузије (погодака и промашаја предикције вештачке неуронске мреже) [6]. Припадни код и изглед матрице конфузије дати су на слици 4. Нпр. цифра 0 је тачно препозната 974 пута, два пута погрешно као цифра 5 и по 1 пут погрешно као цифре 2, 3, 7 и 8.

```
from tensorflow.keras import utils
import numpy as np
from sklearn.metrics import confusion_matrix
import seaborn
y_prediction = model.predict(test_images)
y_prediction_classes = np.argmax(y_prediction, axis=1)
confusion_mtx = confusion_matrix(test_labels, y_prediction_classes)
fig, ax = plt.subplots(figsize=(12, 8))
seaborn.set(font_scale=2.0)
ax = seaborn.heatmap(confusion_mtx, annot=True, fmt='d',
                     ax=ax, cmap="Blues")
ax.set_xlabel('Predicted label'); ax.set_ylabel('True label')
ax.set_title('Confusion matrix')
```



Слика 4 – Матрица конфузије, Python код лево и изглед матрице десно

5. ЗАКЉУЧАК

Рад описује један приступ увођења студената у рад са вештачким неуронским мрежама коришћењем програмског језука *Python* и припадних библиотека *TensorFlow* и *Keras*. Модел вештачког неурона је дат у Надграђеним Петри-мрежама да би се показао ниво паралелизма који се може искористити у раду са вештачким неуронским мрежама коришћењем *Nvidia* графичке картице. Студенти проласком кроз: фунцкије мреже, активационе фунцкије, учитавање и подешавање скупа података за тренирање и тестирање вештачке неуронске мреже, пројектовања слојева мреже, оптимизаторе, коефицијент обучавања, фунцкије губитка, постављање параметара за обуку, графички приказ грешке губитка и тачности након сваке епохе при тренирању и тестирању те приказ матрице конфузије стижу до нивоа да самостално пројектују, обуче, анализирају и побољшају вештачку неуронску мрежу за конкретан проблем класификације. Предвиђено је да студенти на основу конкретне вештачке неуронске мреже урађене на предавању пројектују на припадним вежбама нову вештачку неуронску мрежу за други проблем класификације и да на основу података који се тичу грешке губитка и тачности при тренирању и тестирању те матрице конфузије ураде побољшавање вештачке неуронске мреже док не дођу до тражене тачности.

6. ЛИТЕРАТУРА

[1] Janardhanan, P.S. (2020). *Project repositories for machine learning with TensorFlow*. Elsevier Procedia Computer Science Volume 171, 2020, 188-196.

[2] Hiwarkar, K., Ranjan, R., Ringnekar, V., Rinwa, H., Singh, R. (2021). *Image Captioning using Keras,* International Journal of Scientific Research in Engineering and Management (IJSREM), Volume: 05, Issue: 06, 1-4.

[3] Štrbac, P., Milovanović, G.V. (2013). *Upgraded Petri net model and analysis of adaptive and static arithmetic coding*, Elsevier: Mathematical and Computer Modelling Vol. 58, 1548–1562.

[4] Пејановић, М., Корач, В., Штрбац, П., Пејановић, С. (2020). *Примена NEUROPH радног оквира у учењу вештачких неуронских мрежа*, 7. Конференција са међународним учешћем - Управљање знањем и информатика, Врњачка бања, 17-24

[5] Chollet, F. (2018). *Deep Learning with Python.* Manning Publications Co.

[6] Hasnain, M., Fermi, M. P., Ghani, I., Imran, M., Alzahrani, M. Y., Budiarto R. (2020). *Evaluating Trust Prediction and Confusion Matrix.* IEEE Access Volume 8, 90847-90861