Wykonany kod

**Obraz zawierający tekst

Opis wygenerowany automatycznieZadanie 1**

from keras.models import Sequential

from keras.layers import \*

import numpy as np

m = Sequential()

m.add(Dense(2,input\_dim=2, activation='tanh'))

m.add(Dense(12,input\_dim=2, activation='tanh'))

#m.add(Activation('tanh'))

m.add(Dense(1,activation='sigmoid'))

#m.add(Activation('sigmoid'))

X = np.array([[0,0],[0,1],[1,0],[1,1]],'float32')

Y = np.array([[0],[1],[1],[0]],'float32')

Yand=np.array([[0],[0],[0],[1]],'float32')

Yor=np.array([[0],[1],[1],[1]],'float32')

Ynor=np.array([[1],[0],[0],[0]],'float32')

Ynand=np.array([[1],[1],[1],[1]],'float32')

m.compile(optimizer='adam',loss='binary\_crossentropy')

m.fit(X,Y,batch\_size=1,epochs=200,verbose=0)

print(m.predict(X))

print("Bramka AND")

m.fit(X,Yand,batch\_size=1,epochs=200,verbose=0)

print(m.predict(X))

print("Bramka OR")

m.fit(X,Yor,batch\_size=1,epochs=200,verbose=0)

print(m.predict(X))

print("Bramka NOR")

m.fit(X,Ynor,batch\_size=1,epochs=200,verbose=0)

print(m.predict(X))

print("Bramka NAND")

m.fit(X,Ynand,batch\_size=1,epochs=200,verbose=0)

print(m.predict(X))

**Zadanie 3**

Pytanie a)

Gdy model zawiera mniej warstw i neuronów, trenowanie takie sieci przebiega szybciej. Im bardziej rozbudowana architektura czyli więcej ukrytych warstw i neuronów tym trenowanie trwa dłużej.

Pytanie b)

W zależności od wybranej architektury i optymalizatora możemy uzyskać satysfakcjonujące wyniki. Dla porównywanych danych były to optymalizatory „adam” i „adadelta”.

Pytanie c)

Ponieważ dopasowujemy na poziomie zero jedynkowym, co oznacza, że nie ma stanów pośrednich dla poszczególnych dopasowani. Powoduje to ograniczenia w idealnym dopasowaniu do wyniku wzorcowego.

Pytanie d)

Zmiany optymalizatorów wpływają na wyniki otrzymywanych symulacji. Na przeprowadzonych symulacjach „adadelta” otrzymywał lepsze wyniki i generował mniej strat niż optymalizator „ftrl”