Basic program Tanh and Relu

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

!nvidia-smi

Sat May 22 20:13:14 2021

+-----------------------------------------------------------------------------+

| NVIDIA-SMI 465.19.01    Driver Version: 460.32.03    CUDA Version: 11.2     |

|-------------------------------+----------------------+----------------------+

| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |

| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |

|                               |                      |               MIG M. |

|===============================+======================+======================|

|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |

| N/A   44C    P8     9W /  70W |      0MiB / 15109MiB |      0%      Default |

|                               |                      |                  N/A |

+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+

| Processes:                                                                  |

|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |

|        ID   ID                                                   Usage      |

|=============================================================================|

|  No running processes found                                                 |

+-----------------------------------------------------------------------------+

import keras

from keras.models import Sequential

from keras.layers import Convolution2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense

from keras import regularizers

from keras.layers import Dropout

from keras.callbacks import ModelCheckpoint

import pandas\_profiling

# import tensorflow as tf

# #cpu - gpu configuration

# config = tf.ConfigProto( device\_count = {'GPU': 1 , 'CPU': 56} ) #max: 1 gpu, 56 cpu

# sess = tf.Session(config=config)

# keras.backend.set\_session(sess)

from tensorflow.python.client import device\_lib

print(device\_lib.list\_local\_devices())

[name: "/device:CPU:0"

device\_type: "CPU"

memory\_limit: 268435456

locality {

}

incarnation: 13882271212528950541

, name: "/device:GPU:0"

device\_type: "GPU"

memory\_limit: 11145797952

locality {

  bus\_id: 1

  links {

  }

}

incarnation: 1720511234218446309

physical\_device\_desc: "device: 0, name: Tesla K80, pci bus id: 0000:00:04.0, compute capability: 3.7"

]

from keras.preprocessing.image import ImageDataGenerator

from PIL import Image

train\_datagen = ImageDataGenerator(rescale=1./255,

                                  shear\_range=0.2,

                                  zoom\_range=0.2,

                                  horizontal\_flip=True)

test\_datagen = ImageDataGenerator(rescale=1./255)

training\_set = train\_datagen.flow\_from\_directory('/content/drive/MyDrive/Dataset',

                                                 target\_size=(256, 256),

                                                 batch\_size= 32,

                                                 class\_mode='categorical')

Found 720 images belonging to 4 classes.

test\_set = test\_datagen.flow\_from\_directory('/content/drive/MyDrive/test',

                                                        target\_size = (256, 256),

                                                        batch\_size = 32,

                                                        class\_mode = 'categorical')

Found 180 images belonging to 4 classes.

Model with Tanh

classifier = Sequential()

# Layer1

classifier.add(Convolution2D(128, (3, 3), input\_shape = (256, 256,3), activation = 'tanh'))

# Layer 2

classifier.add(Convolution2D(64, (3, 3), activation = 'tanh'))

# Pooling Layer

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

#Adding another layer

classifier.add(Convolution2D(32, (3, 3), activation = 'tanh'))

#Pooling

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

#Adding another layer

classifier.add(Convolution2D(32, (3, 3), activation = 'tanh'))

#Pooling

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

#Step 3- Flattening

classifier.add(Flatten())

#Step 4- Full connection

classifier.add(Dense(units = 128, activation = 'tanh'))

#For the output step

classifier.add(Dense(units = 4, activation = 'softmax'))

classifier.add(Dropout(0.01))

classifier.compile(optimizer = 'adam',loss = 'categorical\_crossentropy', metrics = ['accuracy'])

plot\_compare\_1 = classifier.fit\_generator(training\_set,

                    steps\_per\_epoch=int(720/32),

                    epochs = 32,

                    validation\_data=test\_set,

                    validation\_steps=int(180/32),)

/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit\_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

  warnings.warn('`Model.fit\_generator` is deprecated and '

Epoch 1/32

22/22 [==============================] - 15s 673ms/step - loss: 4.8847 - accuracy: 0.2689 - val\_loss: 2.7293 - val\_accuracy: 0.2375

Epoch 2/32

22/22 [==============================] - 15s 692ms/step - loss: 2.9201 - accuracy: 0.2907 - val\_loss: 1.5898 - val\_accuracy: 0.2688

Epoch 3/32

22/22 [==============================] - 15s 679ms/step - loss: 1.4834 - accuracy: 0.3503 - val\_loss: 1.5691 - val\_accuracy: 0.2625

Epoch 4/32

22/22 [==============================] - 15s 667ms/step - loss: 1.4743 - accuracy: 0.3924 - val\_loss: 1.4634 - val\_accuracy: 0.2375

Epoch 5/32

22/22 [==============================] - 15s 670ms/step - loss: 1.4433 - accuracy: 0.3895 - val\_loss: 1.4498 - val\_accuracy: 0.2625

Epoch 6/32

22/22 [==============================] - 15s 665ms/step - loss: 1.4798 - accuracy: 0.3968 - val\_loss: 1.4722 - val\_accuracy: 0.2562

Epoch 7/32

22/22 [==============================] - 15s 668ms/step - loss: 1.5115 - accuracy: 0.3808 - val\_loss: 1.4228 - val\_accuracy: 0.2625

Epoch 8/32

22/22 [==============================] - 15s 671ms/step - loss: 1.3985 - accuracy: 0.3648 - val\_loss: 1.4475 - val\_accuracy: 0.2688

Epoch 9/32

22/22 [==============================] - 15s 673ms/step - loss: 1.5252 - accuracy: 0.3953 - val\_loss: 1.4550 - val\_accuracy: 0.2500

Epoch 10/32

22/22 [==============================] - 15s 670ms/step - loss: 1.4180 - accuracy: 0.3895 - val\_loss: 1.4613 - val\_accuracy: 0.2500

Epoch 11/32

22/22 [==============================] - 15s 673ms/step - loss: 1.5060 - accuracy: 0.3939 - val\_loss: 1.4523 - val\_accuracy: 0.2500

Epoch 12/32

plot\_compare\_1.history['val\_accuracy']

[0.23749999701976776,

 0.26875001192092896,

 0.26249998807907104,

 0.23749999701976776,

 0.26249998807907104,

 0.2562499940395355,

 0.26249998807907104,

 0.26875001192092896,

 0.25,

 0.25,

 0.25,

 0.2562499940395355,

 0.24375000596046448,

 0.24375000596046448,

 0.2562499940395355,

 0.2562499940395355,

 0.23749999701976776,

 0.25,

 0.2562499940395355,

 0.25,

plot\_compare\_1.history['accuracy']

[0.26889535784721375,

 0.2906976640224457,

 0.3502906858921051,

 0.39244186878204346,

 0.3895348906517029,

 0.3968023359775543,

 0.38081395626068115,

 0.364825576543808,

 0.39534884691238403,

 0.3895348906517029,

 0.39389535784721375,

 0.3866279125213623,

 0.38226744532585144,

 0.39244186878204346,

Model with 'ReLU' activation

classifier = Sequential()

#Creating the method for model

#Step 1- Convolution

classifier.add(Convolution2D(128, (3, 3), input\_shape = (256, 256,3), activation = 'relu'))

#adding another layer

classifier.add(Convolution2D(64, (3, 3), activation = 'relu'))

#Max pooling it

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

#Adding another layer

classifier.add(Convolution2D(32, (3, 3), activation = 'relu'))

#Pooling

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

#Adding another layer

classifier.add(Convolution2D(32, (3, 3), activation = 'relu'))

#Pooling

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

#Step 3- Flattening

classifier.add(Flatten())

#Step 4- Full connection

classifier.add(Dense(units = 128, activation = 'relu'))

#For the output step

classifier.add(Dense(units = 4, activation = 'softmax'))

classifier.add(Dropout(0.01))

classifier.compile(optimizer = 'adam',loss = 'categorical\_crossentropy', metrics = ['accuracy'])

plot\_compare = classifier.fit\_generator(training\_set,

                    steps\_per\_epoch=(720/32),

                    epochs = 32,

                    validation\_data=test\_set,

                    validation\_steps=(180/32),)

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit\_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

  warnings.warn('`Model.fit\_generator` is deprecated and '

Epoch 1/32

22/22 [==============================] - 282s 11s/step - loss: 1.5853 - accuracy: 0.3376 - val\_loss: 1.1267 - val\_accuracy: 0.5000

Epoch 2/32

22/22 [==============================] - 14s 633ms/step - loss: 1.3108 - accuracy: 0.4235 - val\_loss: 0.8796 - val\_accuracy: 0.5833

Epoch 3/32

22/22 [==============================] - 14s 615ms/step - loss: 0.9970 - accuracy: 0.5696 - val\_loss: 1.0787 - val\_accuracy: 0.5389

plot\_compare.history['val\_accuracy']

[0.550000011920929,

 0.49444442987442017,

 0.5111111402511597,

 0.5777778029441833,

 0.5777778029441833,

 0.5444444417953491,

 0.6277777552604675,

 0.6611111164093018,

 0.6555555462837219,

 0.699999988079071,

 0.6944444179534912,

 0.8444444537162781,

 0.800000011920929,

 0.7055555582046509

plot\_compare.history['accuracy']

[0.34166666865348816,

 0.4902777671813965,

 0.5236111283302307,

 0.5916666388511658,

 0.6583333611488342,

 0.6555555462837219,

 0.699999988079071,

 0.7361111044883728,

 0.7430555820465088,

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix()

len(training\_set[0])

**Plotting the validation accuracy between relu and tanh activation function**

import matplotlib.pyplot as pllt

%matplotlib inline

pllt.plot(plot\_compare.history['val\_accuracy'])

pllt.plot(plot\_compare\_1.history['val\_accuracy'])

pllt.title('Validation accuracy comparison of relu and tanh')

pllt.ylabel('Accuracy %')

pllt.xlabel('Number of epoch')

pllt.legend(['Relu', 'tanh'])

pllt.show()

**Plotting the loss between relu and tanh activation function**

import matplotlib.pyplot as pllt

%matplotlib inline

pllt.plot(plot\_compare.history['val\_loss'])

pllt.plot(plot\_compare\_1.history['val\_loss'])

pllt.title('Loss comparison of relu and tanh')

pllt.ylabel('Loss')

pllt.xlabel('Number of epoch')

pllt.legend(['Relu', 'tanh'])

pllt.show()

Plotting the accuracy between relu and tanh activation function

import matplotlib.pyplot as pllt

%matplotlib inline

pllt.plot(plot\_compare.history['accuracy'])

pllt.plot(plot\_compare\_1.history['accuracy'])

pllt.title('Accuracy comparison of relu and tanh')

pllt.ylabel('Accuracy %')

pllt.xlabel('Number of epoch')

pllt.legend(['Relu', 'tanh'])

pllt.show()

**Visualizing the impact of each layer on an input image**

import cv2

import matplotlib.pyplot as pllt

%matplotlib inline

h = cv2.imread('/content/drive/MyDrive/Dataset/large.cell.carcinoma/000173.png')

pllt.imshow(h)

h.shape

(299, 444, 3)

h = h.astype(np.float16)

test\_model = Sequential()

test\_model.add(Convolution2D(3, (3, 3), input\_shape = h.shape, activation = 'relu'))

h\_batch = np.expand\_dims(h, axis=0)

conv\_h = test\_model.predict(h\_batch)

def visualize\_h(test\_model,h):

    h\_batch = np.expand\_dims(h, axis=0)

    conv\_h = test\_model.predict(h\_batch)

    conv\_h = np.squeeze(conv\_h, axis=0)

    print (conv\_h.shape)

    pllt.imshow(conv\_h)