Conversation opened. 2 messages. 1 message unread.

Skip to content

Using Gmail with screen readers

1 of 785

(no subject)

Inbox

Rebekka

AttachmentsThu, 3 Nov, 16:06 (7 days ago)

Rebekka

Attachments

14:21 (0 minutes ago)

to me

---------- Forwarded message ---------

From: Rebekka <rebekka1771@gmail.com>

Date: Thu, 3 Nov 2022 at 16:05

Subject:

To: <shruthinathan83@gmail.com>

One attachment

• Scanned by Gmail

{

"cells": [

{

"cell\_type": "markdown",

"id": "1bc6e315",

"metadata": {},

"source": [

"# Sprint 1\n",

"## Team ID: PNT2022-TMID26748\n",

"\n",

"### Importing required libraries"

]

},

{

"cell\_type": "code",

"execution\_count": 1,

"id": "f1caaada",

"metadata": {},

"outputs": [],

"source": [

"import numpy\n",

"import tensorflow"

]

},

{

"cell\_type": "code",

"execution\_count": 2,

"id": "d41cb711",

"metadata": {},

"outputs": [],

"source": [

"from tensorflow.keras.datasets import mnist \n",

"from tensorflow.keras.models import Sequential# stack for layers\n",

"from tensorflow.keras import layers#input,middle and output layers forcnn structure"

]

},

{

"cell\_type": "code",

"execution\_count": 3,

"id": "ffe518fa",

"metadata": {},

"outputs": [],

"source": [

"from tensorflow.keras.layers import Dense,Flatten#dense and flatten layers\n",

"from tensorflow.keras.layers import Conv2D#convolutional layers\n",

"from tensorflow import keras#library for building neural networks built on tensorflow"

]

},

{

"cell\_type": "code",

"execution\_count": 4,

"id": "759e5138",

"metadata": {},

"outputs": [],

"source": [

"from tensorflow.keras.optimizers import Adam#optimizers\n",

"from keras.utils import np\_utils"

]

},

{

"cell\_type": "markdown",

"id": "69a85859",

"metadata": {},

"source": [

"### Loading the data"

]

},

{

"cell\_type": "code",

"execution\_count": 5,

"id": "ec37769d",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz\n",

"11490434/11490434 [==============================] - 6s 1us/step\n"

]

}

],

"source": [

"(x\_train,y\_train),(x\_test,y\_test)=mnist.load\_data()"

]

},

{

"cell\_type": "markdown",

"id": "96d7186c",

"metadata": {},

"source": [

" Data is splitted for train and test in -70:30 ratio"

]

},

{

"cell\_type": "code",

"execution\_count": 6,

"id": "828f81dd",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"(60000, 28, 28)\n",

"(60000,)\n"

]

}

],

"source": [

"print(x\_train.shape)\n",

"print(y\_train.shape)"

]

},

{

"cell\_type": "markdown",

"id": "ed0523cc",

"metadata": {},

"source": [

"Training Dataset- 60000 images, Testing dataset- 10000 images"

]

},

{

"cell\_type": "markdown",

"id": "75be0be2",

"metadata": {},

"source": [

"### Analyzing the data"

]

},

{

"cell\_type": "code",

"execution\_count": 7,

"id": "a0767d75",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"array([[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0, 51, 159, 253, 159, 50, 0, 0, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 48, 238, 252, 252, 252, 237, 0, 0, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 54, 227, 253, 252, 239, 233, 252, 57, 6, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 60,\n",

" 224, 252, 253, 252, 202, 84, 252, 253, 122, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 163, 252,\n",

" 252, 252, 253, 252, 252, 96, 189, 253, 167, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 51, 238, 253,\n",

" 253, 190, 114, 253, 228, 47, 79, 255, 168, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 48, 238, 252, 252,\n",

" 179, 12, 75, 121, 21, 0, 0, 253, 243, 50, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 0, 38, 165, 253, 233, 208,\n",

" 84, 0, 0, 0, 0, 0, 0, 253, 252, 165, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 7, 178, 252, 240, 71, 19,\n",

" 28, 0, 0, 0, 0, 0, 0, 253, 252, 195, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 57, 252, 252, 63, 0, 0,\n",

" 0, 0, 0, 0, 0, 0, 0, 253, 252, 195, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 198, 253, 190, 0, 0, 0,\n",

" 0, 0, 0, 0, 0, 0, 0, 255, 253, 196, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 76, 246, 252, 112, 0, 0, 0,\n",

" 0, 0, 0, 0, 0, 0, 0, 253, 252, 148, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 85, 252, 230, 25, 0, 0, 0,\n",

" 0, 0, 0, 0, 0, 7, 135, 253, 186, 12, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 85, 252, 223, 0, 0, 0, 0,\n",

" 0, 0, 0, 0, 7, 131, 252, 225, 71, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 85, 252, 145, 0, 0, 0, 0,\n",

" 0, 0, 0, 48, 165, 252, 173, 0, 0, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 86, 253, 225, 0, 0, 0, 0,\n",

" 0, 0, 114, 238, 253, 162, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 85, 252, 249, 146, 48, 29, 85,\n",

" 178, 225, 253, 223, 167, 56, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 85, 252, 252, 252, 229, 215, 252,\n",

" 252, 252, 196, 130, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 28, 199, 252, 252, 253, 252, 252,\n",

" 233, 145, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 25, 128, 252, 253, 252, 141,\n",

" 37, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0],\n",

" [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\n",

" 0, 0]], dtype=uint8)"

]

},

"execution\_count": 7,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"x\_train[1]"

]

},

{

"cell\_type": "code",

"execution\_count": 8,

"id": "81deb472",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"3"

]

},

"execution\_count": 8,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"y\_train[10]"

]

},

{

"cell\_type": "markdown",

"id": "1ee51dfe",

"metadata": {},

"source": [

"The image in 10th position of dataset is 3 here "

]

},

{

"cell\_type": "code",

"execution\_count": 9,

"id": "e4ddab45",

"metadata": {},

"outputs": [],

"source": [

"import matplotlib.pyplot as plt #used for data visualization"

]

},

{

"cell\_type": "code",

"execution\_count": 10,

"id": "ac518156",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"<matplotlib.image.AxesImage at 0x28eff03ee50>"

]

},

"execution\_count": 10,

"metadata": {},

"output\_type": "execute\_result"

},

{

"data": {

"image/png": "",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"plt.imshow(x\_train[10]) #plotting the 10th image"

]

},

{

"cell\_type": "markdown",

"id": "e392c8c6",

"metadata": {},

"source": [

"### Reshaping the data"

]

},

{

"cell\_type": "markdown",

"id": "af8bcc89",

"metadata": {},

"source": [

"As we are using Deep learning neural network, the input for this network to get trained on should be of higher dimensional. Our dataset is having three-dimensional images so we have to reshape them too higher dimensions"

]

},

{

"cell\_type": "code",

"execution\_count": 11,

"id": "7dece652",

"metadata": {},

"outputs": [],

"source": [

"#CNN expected format: (batch,height,width,channel)\n",

"x\_train=x\_train.reshape(60000,28,28,1).astype('float32')\n",

"x\_test=x\_test.reshape(10000,28,28,1).astype('float32')"

]

},

{

"cell\_type": "markdown",

"id": "84fac0f9",

"metadata": {},

"source": [

"### Applying One Hot Encoding"

]

},

{

"cell\_type": "markdown",

"id": "16240262",

"metadata": {},

"source": [

"We are applying One Hot encoding for y\_train set. It converts the numerical values to classes, where 0-9 are seperate classes. If the value is 5, then class 5 is '1', else it is '0'."

]

},

{

"cell\_type": "code",

"execution\_count": 12,

"id": "d3318a0d",

"metadata": {},

"outputs": [],

"source": [

"no\_of\_classes=10\n",

"y\_train=np\_utils.to\_categorical(y\_train,no\_of\_classes) #converts output to binary format\n",

"y\_test=np\_utils.to\_categorical(y\_test,no\_of\_classes)"

]

},

{

"cell\_type": "code",

"execution\_count": 13,

"id": "397a4181",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"array([0., 0., 0., 1., 0., 0., 0., 0., 0., 0.], dtype=float32)"

]

},

"execution\_count": 13,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"y\_train[10]"

]

},

{

"cell\_type": "markdown",

"id": "b9d47caf",

"metadata": {},

"source": [

"We can see that label 3 value is 1"

]

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3 (ipykernel)",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.9.12"

}

},

"nbformat": 4,

"nbformat\_minor": 5

}

BIN +46.2 KB

Project development phase/Sprint- 1/Data\_Preprocessing (sprint-1) (1).docx

Binary file not shown.

761

sprint 1.txt

Displaying sprint 1.txt.