{

2

"nbformat": 4,

3

"nbformat\_minor": 0,

4

"metadata": {

5

"colab": {

6

"provenance": [],

7

"collapsed\_sections": [

8

"yRmwsk\_zfh1U"

9

]

10

},

11

"kernelspec": {

12

"name": "python3",

13

"display\_name": "Python 3"

14

},

15

"language\_info": {

16

"name": "python"

17

}

18

},

19

"cells": [

20

{

21

"cell\_type": "markdown",

22

"metadata": {

23

"id": "5UKKDaCfUNCG"

24

},

25

"source": [

26

"#Importing Keras libraries"

27

]

28

},

29

{

30

"cell\_type": "code",

31

"execution\_count": 73,

32

"metadata": {

33

"id": "wNS3p3egZKNm"

34

},

35

"outputs": [],

36

"source": [

37

"import keras\n"

38

]

39

},

40

{

41

"cell\_type": "markdown",

42

"source": [

43

"# Importing ImageDataGenerator from Keras"

44

],

45

"metadata": {

46

"id": "AucFhyi34cV6"

47

}

48

},

49

{

50

"cell\_type": "code",

51

"source": [

52

"from matplotlib import pyplot as plt\n",

53

"from keras.preprocessing.image import ImageDataGenerator"

54

],

55

"metadata": {

56

"id": "hrFaF\_FQZTiR"

57

},

58

"execution\_count": 74,

59

"outputs": []

60

},

61

{

62

"cell\_type": "markdown",

63

"source": [

64

"#Defining the Parameters"

65

],

66

"metadata": {

67

"id": "Kg-ab3M34wCA"

68

}

69

},

70

{

71

"cell\_type": "code",

72

"source": [

73

"train\_datagen=ImageDataGenerator(rescale=1./255,shear\_range=0.2,rotation\_range=180,zoom\_range=0.2,horizontal\_flip=True)\n",

74

"test\_datagen=ImageDataGenerator(rescale=1./255,shear\_range=0.2,rotation\_range=180,zoom\_range=0.2,horizontal\_flip=True)"

75

],

76

"metadata": {

77

"id": "cic8YJ5mZZ8V"

78

},

79

"execution\_count": 75,

80

"outputs": []

81

},

82

{

83

"cell\_type": "markdown",

84

"source": [

85

"#Applying ImageDataGenerator functionality to train dataset"

86

],

87

"metadata": {

88

"id": "4S\_FJTSk41js"

89

}

90

},

91

{

92

"cell\_type": "code",

93

"source": [

94

"from google.colab import drive\n",

95

"drive.mount('/content/drive')"

96

],

97

"metadata": {

98

"colab": {

99

"base\_uri": "https://localhost:8080/"

100

},

101

"id": "rICUi3ArZbrS",

102

"outputId": "5678ce92-2165-4414-93a7-7304e1b0ef62"

103

},

104

"execution\_count": 76,

105

"outputs": [

106

{

107

"output\_type": "stream",

108

"name": "stdout",

109

"text": [

110

"Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount(\"/content/drive\", force\_remount=True).\n"

111

]

112

}

113

]

114

},

115

{

116

"cell\_type": "code",

117

"source": [

118

"x\_train=train\_datagen.flow\_from\_directory('/content/drive/MyDrive/Dataset/Dataset/train\_set',target\_size=(64,64),batch\_size=32,class\_mode='binary')"

119

],

120

"metadata": {

121

"colab": {

122

"base\_uri": "https://localhost:8080/"

123

},

124

"id": "4MpwTUTzaJgn",

125

"outputId": "1c0cbc6d-1975-4ee4-be77-bdc21d55860d"

126

},

127

"execution\_count": 77,

128

"outputs": [

129

{

130

"output\_type": "stream",

131

"name": "stdout",

132

"text": [

133

"Found 436 images belonging to 2 classes.\n"

134

]

135

}

136

]

137

},

138

{

139

"cell\_type": "markdown",

140

"source": [

141

"#Applying ImageDataGenerator functionality to test dataset"

142

],

143

"metadata": {

144

"id": "go37fe2649lq"

145

}

146

},

147

{

148

"cell\_type": "code",

149

"source": [

150

"x\_test=test\_datagen.flow\_from\_directory('/content/drive/MyDrive/Dataset/Dataset/test\_set',target\_size=(64,64),batch\_size=32,class\_mode='binary')"

151

],

152

"metadata": {

153

"colab": {

154

"base\_uri": "https://localhost:8080/"

155

},

156

"id": "7PiQqUKvc6LD",

157

"outputId": "dfdfec12-7e5c-43c9-dea0-2468c3bdab15"

158

},

159

"execution\_count": 78,

160

"outputs": [

161

{

162

"output\_type": "stream",

163

"name": "stdout",

164

"text": [

165

"Found 121 images belonging to 2 classes.\n"

166

]

167

}

168

]

169

},

170

{

171

"cell\_type": "markdown",

172

"source": [

173

"#Importing Model Building Libraries"

174

],

175

"metadata": {

176

"id": "XhiFshCA5B7R"

177

}

178

},

179

{

180

"cell\_type": "code",

181

"source": [

182

"#to define the linear Initialisation import sequential\n",

183

"from keras.models import Sequential\n",

184

"#to add layers import Dense\n",

185

"from keras.layers import Dense\n",

186

"#to create Convolutional kernel import convolution2D\n",

187

"from keras.layers import Convolution2D\n",

188

"#import Maxpooling layer \n",

189

"from keras.layers import MaxPooling2D\n",

190

"#import flatten layer\n",

191

"from keras.layers import Flatten\n",

192

"import warnings\n",

193

"warnings.filterwarnings('ignore')"

194

],

195

"metadata": {

196

"id": "sv3Lk\_XBdH0n"

197

},

198

"execution\_count": 79,

199

"outputs": []

200

},

201

{

202

"cell\_type": "markdown",

203

"source": [

204

"#Initializing the model"

205

],

206

"metadata": {

207

"id": "iE6UPUhl5HE1"

208

}

209

},

210

{

211

"cell\_type": "code",

212

"source": [

213

"model = Sequential()"

214

],

215

"metadata": {

216

"id": "\_KGq9RuYdMF3"

217

},

218

"execution\_count": 80,

219

"outputs": []

220

},

221

{

222

"cell\_type": "markdown",

223

"source": [

224

"#Adding CNN Layers"

225

],

226

"metadata": {

227

"id": "LZmIgzgS5OoA"

228

}

229

},

230

{

231

"cell\_type": "code",

232

"source": [

233

"model.add(Convolution2D(32,(3,3),input\_shape=(64,64,3),activation='relu'))\n",

234

"#add maxpooling layers\n",

235

"model.add(MaxPooling2D(pool\_size=(2,2)))\n",

236

"#add faltten layer\n",

237

"model.add(Flatten())"

238

],

239

"metadata": {

240

"id": "dHrl0v0DdPrV"

241

},

242

"execution\_count": 81,

243

"outputs": []

244

},

245

{

246

"cell\_type": "markdown",

247

"source": [

248

"#Add Dense layers"

249

],

250

"metadata": {

251

"id": "q7HikkUD5Q6R"

252

}

253

},

254

{

255

"cell\_type": "code",

256

"source": [

257

"#add hidden layers\n",

258

"model.add(Dense(150,activation='relu'))\n",

259

"#add output layer\n",

260

"model.add(Dense(1,activation='sigmoid'))"

261

],

262

"metadata": {

263

"id": "m0NxRsREdSY6"

264

},

265

"execution\_count": 82,

266

"outputs": []

267

},

268

{

269

"cell\_type": "markdown",

270

"source": [

271

"#configuring the learning process"

272

],

273

"metadata": {

274

"id": "31h-Q72f5bWP"

275

}

276

},

277

{

278

"cell\_type": "code",

279

"source": [

280

"model.compile(loss='binary\_crossentropy',optimizer=\"adam\",metrics=[\"accuracy\"])"

281

],

282

"metadata": {

283

"id": "5MFPL0qQdV0j"

284

},

285

"execution\_count": 83,

286

"outputs": []

287

},

288

{

289

"cell\_type": "markdown",

290

"source": [

291

"#Training the model"

292

],

293

"metadata": {

294

"id": "dCV8Tck05qOP"

295

}

296

},

297

{

298

"cell\_type": "code",

299

"source": [

300

"model.fit\_generator(x\_train,steps\_per\_epoch=14,epochs=10,validation\_data=x\_test,validation\_steps=4)"

301

],

302

"metadata": {

303

"colab": {

304

"base\_uri": "https://localhost:8080/"

305

},

306

"id": "sKKRck\_cdYoN",

307

"outputId": "fb45bfe0-8345-40fa-f306-18da8fd7de4a"

308

},

309

"execution\_count": 84,

310

"outputs": [

311

{

312

"output\_type": "stream",

313

"name": "stdout",

314

"text": [

315

"Epoch 1/10\n",

316

"14/14 [==============================] - 23s 2s/step - loss: 1.0287 - accuracy: 0.6261 - val\_loss: 0.3554 - val\_accuracy: 0.8430\n",

317

"Epoch 2/10\n",

318

"14/14 [==============================] - 22s 2s/step - loss: 0.3303 - accuracy: 0.8349 - val\_loss: 0.1432 - val\_accuracy: 0.9421\n",

319

"Epoch 3/10\n",

320

"14/14 [==============================] - 21s 1s/step - loss: 0.2342 - accuracy: 0.8991 - val\_loss: 0.1357 - val\_accuracy: 0.9339\n",

321

"Epoch 4/10\n",

322

"14/14 [==============================] - 20s 1s/step - loss: 0.2079 - accuracy: 0.8945 - val\_loss: 0.1010 - val\_accuracy: 0.9669\n",

323

"Epoch 5/10\n",

324

"14/14 [==============================] - 24s 2s/step - loss: 0.1700 - accuracy: 0.9335 - val\_loss: 0.0907 - val\_accuracy: 0.9504\n",

325

"Epoch 6/10\n",

326

"14/14 [==============================] - 21s 2s/step - loss: 0.1963 - accuracy: 0.9174 - val\_loss: 0.1715 - val\_accuracy: 0.9091\n",

327

"Epoch 7/10\n",

328

"14/14 [==============================] - 22s 1s/step - loss: 0.2057 - accuracy: 0.8991 - val\_loss: 0.1369 - val\_accuracy: 0.9256\n",

329

"Epoch 8/10\n",

330

"14/14 [==============================] - 22s 2s/step - loss: 0.1909 - accuracy: 0.9037 - val\_loss: 0.0811 - val\_accuracy: 0.9752\n",

331

"Epoch 9/10\n",

332

"14/14 [==============================] - 20s 1s/step - loss: 0.1515 - accuracy: 0.9404 - val\_loss: 0.0631 - val\_accuracy: 0.9835\n",

333

"Epoch 10/10\n",

334

"14/14 [==============================] - 23s 2s/step - loss: 0.1780 - accuracy: 0.9266 - val\_loss: 0.1486 - val\_accuracy: 0.9256\n"

335

]

336

},

337

{

338

"output\_type": "execute\_result",

339

"data": {

340

"text/plain": [

341

"<keras.callbacks.History at 0x7fd8057e4b10>"

342

]

343

},

344

"metadata": {},

345

"execution\_count": 84

346

}

347

]

348

},

349

{

350

"cell\_type": "markdown",

351

"source": [

352

"#Save the model"

353

],

354

"metadata": {

355

"id": "NRbjg9k75xq3"

356

}

357

},

358

{

359

"cell\_type": "code",

360

"source": [

361

"model.save(\"/content/drive/MyDrive/Dataset/forest.h5\",save\_format=\"h5\")"

362

],

363

"metadata": {

364

"id": "0R8H0LRVfdS0"

365

},

366

"execution\_count": 85,

367

"outputs": []

368

},

369

{

370

"cell\_type": "markdown",

371

"source": [

372

"#Predictions"

373

],

374

"metadata": {

375

"id": "5qczWh5I53Ug"

376

}

377

},

378

{

379

"cell\_type": "code",

380

"source": [

381

"#import load model from keras.model\n",

382

"from keras.models import load\_model\n",

383

"#import image from keras\n",

384

"from tensorflow.keras.preprocessing import image\n",

385

"import numpy as np\n",

386

"#import cv2\n",

387

"import cv2\n",

388

"#load the saved model\n",

389

"model=load\_model(\"/content/drive/MyDrive/Dataset/forest.h5\")\n",

390

"img=image.load\_img('/content/drive/MyDrive/Dataset/Dataset/test\_set/with fire/Forest\_Fire (3).jpg')\n",

391

"x=image.img\_to\_array(img)\n",

392

"res=cv2.resize(x,dsize=(64,64),interpolation=cv2.INTER\_CUBIC)\n",

393

"#expand the image shape\n",

394

"x=np.expand\_dims(res,axis=0)"

395

],

396

"metadata": {

397

"id": "yRmwsk\_zfh1U"

398

},

399

"execution\_count": 86,

400

"outputs": []

401

},

402

{

403

"cell\_type": "code",

404

"source": [

405

"pred=model.predict(x)\n",

406

"pred = int(pred[0][0])\n",

407

"pred\n",

408

"int(pred)"

409

],

410

"metadata": {

411

"colab": {

412

"base\_uri": "https://localhost:8080/"

413

},

414

"id": "ctUBhhzZhAa2",

415

"outputId": "91995bd9-d8c7-4721-9a67-7397d028f408"

416

},

417

"execution\_count": 87,

418

"outputs": [

419

{

420

"output\_type": "stream",

421

"name": "stdout",

422

"text": [

423

"1/1 [==============================] - 0s 77ms/step\n"

424

]

425

},

426

{

427

"output\_type": "execute\_result",

428

"data": {

429

"text/plain": [

430

"1"

431

]

432

},

433

"metadata": {},

434

"execution\_count": 87

435

}

436

]

437

},

438

{

439

"cell\_type": "code",

440

"source": [

441

"pip install twilio"

442

],

443

"metadata": {

444

"colab": {

445

"base\_uri": "https://localhost:8080/"

446

},

447

"id": "8oPnai91hErS",

448

"outputId": "6e0baf75-1373-47bc-f06e-1bcf9bc1edd8"

449

},

450

"execution\_count": 88,

451

"outputs": [

452

{

453

"output\_type": "stream",

454

"name": "stdout",

455

"text": [

456

"Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/\n",

457

"Requirement already satisfied: twilio in /usr/local/lib/python3.7/dist-packages (7.15.3)\n",

458

"Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-packages (from twilio) (2022.6)\n",

459

"Requirement already satisfied: PyJWT<3.0.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from twilio) (2.6.0)\n",

460

"Requirement already satisfied: requests>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from twilio) (2.23.0)\n",

461

"Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (1.24.3)\n",

462

"Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (2022.9.24)\n",

463

"Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (2.10)\n",

464

"Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (3.0.4)\n"

465

]

466

}

467

]

468

},

469

{

470

"cell\_type": "code",

471

"source": [

472

"from twilio.rest import Client"

473

],

474

"metadata": {

475

"id": "Dbp68LQIhMiz"

476

},

477

"execution\_count": 90,

478

"outputs": []

479

},

480

{

481

"cell\_type": "code",

482

"source": [

483

"if pred==0:\n",

484

" print('Forest fire')\n",

485

" account\_sid='ACe5bf204e0b9f4730256b6e4d050f4356'\n",

486

" auth\_token='2da5727db583fbcc8b271bc7fa2fb9e6'\n",

487

" client=Client(account\_sid,auth\_token)\n",

488

" message=client.messages \\\n",

489

" .create(\n",

490

" body='forest fire is detected,stay alert',\n",

491

" #use twilio free number\n",

492

" from\_='+13465176134',\n",

493

" #to number\n",

494

" to='+918248192151')\n",

495

" print(message.sid)\n",

496

" print(\"Fire detected\")\n",

497

" print(\"SMS Sent!\")\n",

498

"elif pred==1:\n",

499

" print('No Fire')"

500

],

501

"metadata": {

502

"colab": {

503

"base\_uri": "https://localhost:8080/"

504

},

505

"id": "dm8qrmeEhRBA",

506

"outputId": "a47488ae-9911-4a55-b69d-c3f3fb0ce2ef"

507

},

508

"execution\_count": 91,

509

"outputs": [

510

{

511

"output\_type": "stream",

512

"name": "stdout",

513

"text": [

514

"No Fire\n"

515

]

516

}

517

]

518

},

519

{

520

"cell\_type": "markdown",

521

"source": [

522

"#Open cv for video processing"

523

],

524

"metadata": {

525

"id": "8zP7R6bw6IxY"

526

}

527

},

528

{

529

"cell\_type": "code",

530

"source": [

531

"pip install twilio"

532

],

533

"metadata": {

534

"colab": {

535

"base\_uri": "https://localhost:8080/"

536

},

537

"id": "ydwRNdzXjWbu",

538

"outputId": "281b1f71-8c3c-4e76-da9b-39de96e25f8f"

539

},

540

"execution\_count": 92,

541

"outputs": [

542

{

543

"output\_type": "stream",

544

"name": "stdout",

545

"text": [

546

"Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/\n",

547

"Requirement already satisfied: twilio in /usr/local/lib/python3.7/dist-packages (7.15.3)\n",

548

"Requirement already satisfied: requests>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from twilio) (2.23.0)\n",

549

"Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-packages (from twilio) (2022.6)\n",

550

"Requirement already satisfied: PyJWT<3.0.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from twilio) (2.6.0)\n",

551

"Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (3.0.4)\n",

552

"Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (2.10)\n",

553

"Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (1.24.3)\n",

554

"Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (2022.9.24)\n"

555

]

556

}

557

]

558

},

559

{

560

"cell\_type": "markdown",

561

"source": [

562

"#Creating An Account in Twilio Service"

563

],

564

"metadata": {

565

"id": "AcUiOHx160y9"

566

}

567

},

568

{

569

"cell\_type": "markdown",

570

"source": [

571

"#Sending Alert Message"

572

],

573

"metadata": {

574

"id": "cTmMgqhq64Yy"

575

}

576

},

577

{

578

"cell\_type": "code",

579

"source": [

580

"from logging import WARNING\n",

581

"#import opencv library\n",

582

"import cv2\n",

583

"#import numpy\n",

584

"import numpy as np\n",

585

"#import image function from keras\n",

586

"from keras.preprocessing import image\n",

587

"#import load\_model from keras\n",

588

"from keras.models import load\_model\n",

589

"#import client from twilio API\n",

590

"from twilio.rest import Client\n",

591

"#import playsound package"

592

],

593

"metadata": {

594

"id": "qk33uxDgjaL7"

595

},

596

"execution\_count": 93,

597

"outputs": []

598

},

599

{

600

"cell\_type": "code",

601

"source": [

602

"import cv2 \n",

603

"import numpy as np\n",

604

"from google.colab.patches import cv2\_imshow\n",

605

"from matplotlib import pyplot as plt\n",

606

"import librosa\n",

607

"from tensorflow.keras.preprocessing import image\n",

608

"from keras.models import load\_model\n",

609

"# Create a VideoCapture object and read from input file\n",

610

"# If the input is the camera, pass 0 instead of the video file name\n",

611

"cap = cv2.VideoCapture('/content/drive/MyDrive/Dataset/Dataset/pexels-arnav-kainthola-7543653.mp4')\n",

612

" \n",

613

"# Check if camera opened successfully\n",

614

"if (cap.isOpened()== False): \n",

615

" print(\"Error opening video stream or file\")\n",

616

" \n",

617

"# Read until video is completed\n",

618

"while(cap.isOpened()):\n",

619

" # Capture frame-by-frame\n",

620

" ret, frame = cap.read()\n",

621

" if ret == True:\n",

622

" \n",

623

" cv2\_imshow(frame)\n",

624

" x=image.img\_to\_array(frame)\n",

625

" res=cv2.resize(x,dsize=(64,64),interpolation=cv2.INTER\_CUBIC)\n",

626

" #expand the image shape\n",

627

" x=np.expand\_dims(res,axis=0)\n",

628

" model=load\_model(\"/content/drive/MyDrive/Dataset/forest.h5\")\n",

629

" pred=model.predict(x)\n",

630

" pred = int(pred[0][0])\n",

631

" pred\n",

632

" int(pred)\n",

633

" if pred==0:\n",

634

" print('Forest fire')\n",

635

" break\n",

636

" else:\n",

637

" print(\"no danger\")\n",

638

" break\n",

639

" \n",

640

"# When everything done, release the video capture object\n",

641

"cap.release()\n",

642

" \n",

643

"# Closes all the frames\n",

644

"cv2.destroyAllWindows()"

645

],

646

"metadata": {

647

"colab": {

648

"base\_uri": "https://localhost:8080/",

649

"height": 678

650

},

651

"id": "bKo9V1\_wjgCX",

652

"outputId": "3c5e8628-e731-4f64-b6c0-2176749281fb"

653

},

654

"execution\_count": 94,

655

"outputs": [

656

{

657

"output\_type": "display\_data",

658

"data": {

659

"text/plain": [

660

"<PIL.Image.Image image mode=RGB size=1920x1080 at 0x7FD805C18350>"

661

],

662

"image/png": "\n"

663

},

664

"metadata": {}

665

},

666

{

667

"output\_type": "stream",

668

"name": "stdout",

669

"text": [

670

"1/1 [==============================] - 0s 67ms/step\n",

671

"Forest fire\n"

672

]

673

}

674

]

675

},

676

{

677

"cell\_type": "code",

678

"source": [

679

"from twilio.rest import Client\n",

680

"if pred==0:\n",

681

" print('Forest fire')\n",

682

" from twilio.rest import Client\n",

683

" account\_sid='ACe5bf204e0b9f4730256b6e4d050f4356'\n",

684

" auth\_token='2da5727db583fbcc8b271bc7fa2fb9e6'\n",

685

" client=Client(account\_sid,auth\_token)\n",

686

" message=client.messages \\\n",

687

" .create(\n",

688

" body='forest fire is detected,stay alert',\n",

689

" #use twilio free number\n",

690

" from\_='+13465176134',\n",

691

" #to number\n",

692

" to='+918248192151')\n",

693

" print(message.sid)\n",

694

" print(\"Fire detected\")\n",

695

" print(\"SMS Sent!\")\n",

696

"elif pred==1:\n",

697

" print('No Fire')"

698

],

699

"metadata": {

700

"colab": {

701

"base\_uri": "https://localhost:8080/"

702

},

703

"id": "rJCsIA6cm8K7",

704

"outputId": "a5e57734-9754-47e3-e4b1-633a45f79906"

705

},

706

"execution\_count": 95,

707

"outputs": [

708

{

709

"output\_type": "stream",

710

"name": "stdout",

711

"text": [

712

"Forest fire\n",

713

"SM4c829a88e83e0902f4027553ddab0bf6\n",

714

"Fire detected\n",

715

"SMS Sent!\n"

716

]

717

}

718

]

719

}

720

]

721

}