```
{
 "nbformat": 4,
 "nbformat_minor": 0,
 "metadata": {
  "colab": {
   "provenance": []
  },
  "kernelspec": {
   "name": "python3",
   "display_name": "Python 3"
  },
  "language_info": {
   "name": "python"
  }
 },
 "cells": [
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "S2YEqD2m-0Je"
   },
   "outputs": [],
   "source": [
    "import numpy as np\n",
    "import tensorflow #open source used for both ML and DL for computation\n",
    "from tensorflow.keras.datasets import disaster #disaster dataset\n",
    "from tensorflow.keras.models import Sequential #it is a plain stack of layers\n",
    "from tensorflow.keras import layers #A Layer consists of a tensor- in\n",
    "tensor-out computation function\n",
    "from tensorflow.keras.layers import Dense, Flatten #Dense-Dense Layer is the\n",
```

```
    "regular deeply connected\n",
    "#flatten-used for flattening the input or change the dimension\n",
    "from tensorflow.keras.layers import Conv2D #convoLutional Layer\n",
    "from keras.optimizers import Adam #optimizer\n",
    "from keras. utils import np_utils #used for one-hot encoding\n",
    "import matplotlib.pyplot as plt #used for data visualization\n",
    "(x_train, y_train), (x_test, y_test)=disaster.load_data() #splitting the\n",
    "disaster data\n",
    "#Reshaping to format which CNN expects (batch, height, width, channels)\n",
    "x_train=x_train.reshape (60000, 28, 28, 1).astype('float32')\n",
    "x_test=x_test.reshape (10000, 28, 28, 1).astype ('float32')\n",
    "#one hot encode\n",
    "number_of_classes = 10 #storing the no of classes in a variable\n",
    "y_train = np_utils.to_categorical (y_train, number_of_classes) #converts the\n",
    "output in binary format\n",
    "y_test = np_utils.to_categorical (y_test, number_of_classes)\n",
    "Downloading data from\n",
    "https://storage.googleapis.com/tensorflow/tf-keras-datasets/disaster.npz\n",
    "11490434/11490434 [==============================] - 1s 0us/step"
  ]
 },
 {
  "cell_type": "markdown",
  "source": [
   "ADD CNN LAYERS"
  ],
  "metadata": {
   "id": "ReSmzNWIAL4n"
  }
 },
 {
```

```
 "cell_type": "code",
 "source": [
  "#create model\n",
  "model=Sequential ()\n",
  "#adding model Layer\n",
  "model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1), activation='relu'))\n",
  "model.add(Conv2D(32, (3, 3), activation = 'relu'))\n",
  "#model.add(conv2D(32, (3,3), activation = 'relu))\n",
  "#flatten the dimension of the image\n",
  "model.add(Flatten())\n",
  "#output layer with 10 neurons\n",
  "model.add(Dense(number_of_classes,activation = 'softmax'))\n"
 ],
 "metadata": {
  "id": "9HCsWP4F_Mnh"
 },
 "execution_count": null,
 "outputs": []
},
{
 "cell_type": "markdown",
 "source": [
  "COMPILING THE MODEL"
 ],
 "metadata": {
  "id": "TxDLP63K_3-5"
 }
},
{
 "cell_type": "code",
 "source": [
```

```
   "#Compile model\n",
   "model.compile(loss= 'categorical_crossentropy', optimizer=\"Adam\",\n",
   "metrics=['accuracy'])\n"
  ],
  "metadata": {
   "id": "ogBYRAXc_qew"
  },
  "execution_count": null,
  "outputs": []
 },
 {
  "cell_type": "markdown",
  "source": [
   "TRAIN THE MODEL"
  ],
  "metadata": {
   "id": "6XRDw-xRA31C"
  }
 },
 {
  "cell_type": "code",
  "source": [
   "#fit the model\n",
   "model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=5,\n",
   "batch_size=32)\n",
   "Epoch 1/5\n",
   "1875/1875 [==============================] - 194s 103ms/step - loss: 0.1173 -\n",
   "accuracy: 0.9662 - val_loss: 0.0771 - val_accuracy: 0.9777\n",
   "Epoch 2/5\n",
   "1875/1875 [==============================] - 195s 104ms/step - loss: 0.0644 -\n",
   "accuracy: 0.9801 - val_loss: 0.0795 - val_accuracy: 0.9777\n",
```

```
    "Epoch 3/5\n",
    "1875/1875 [==============================] - 197s 105ms/step - loss: 0.0441 -\n",
    "accuracy: 0.9863 - val_loss: 0.1046 - val_accuracy: 0.9759\n",
    "Epoch 4/5\n",
    "1875/1875 [==============================] - 205s 110ms/step - loss: 0.0351 -\n",
    "accuracy: 0.9887 - val_loss: 0.0871 - val_accuracy: 0.9782\n",
    "Epoch 5/5\n",
    "1875/1875 [==============================] - 207s 110ms/step - loss: 0.0284 -\n",
    "accuracy: 0.9909 - val_loss: 0.1242 - val_accuracy: 0.9762\n",
    "<keras.callbacks.History at 0x7fab39e6e9d0>"
   ],
   "metadata": {
    "id": "LtW2nct__toO"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "markdown",
   "source": [
    "OBSERVING THE METRICS"
   ],
   "metadata": {
    "id": "tj2NrU-TBMlZ"
   }
  },
  {
   "cell_type": "code",
   "source": [
    "# Final evaluation of the model\n",
    "metrics = model.evaluate(x_test, y_test, verbose=0)\n",
```

```
   "print(\"Metrics (Test loss & Test Accuracy): \")\n",
   "print(metrics)\n",
   "Metrics (Test loss & Test Accuracy):\n",
   "[0.12423925846815109, 0.9761999845504761]"
  ],
  "metadata": {
   "id": "jD7lks2eBT_M"
  },
  "execution_count": null,
  "outputs": []
 },
 {
  "cell_type": "markdown",
  "source": [
   "TEST THE MODEL"
  ],
  "metadata": {
   "id": "v-8GA-FFBbrl"
  }
 },
 {
  "cell_type": "code",
  "source": [
   "# Predicting the Output\n",
   "prediction=model.predict(x_test[:4])\n",
   "print(prediction)\n",
   "1/1 [==============================] - 0s 85ms/step\n",
   "[[6.25729828e-13 8.39843610e-19 3.52224987e-07 3.49750486e-08\n",
   "2.61816901e-21 4.89236403e-17 6.80994400e-23 9.99999642e-01\n",
   "1.00192285e-10 1.46840540e-09]\n",
   "[2.13350471e-09 7.24474439e-11 1.00000000e+00 1.42506189e-12\n",
```

```
    "1.07695855e-18 2.63603979e-20 9.05597333e-11 3.16722711e-12\n",
    "1.44268256e-12 2.35227114e-22]\n",
    "[5.82387694e-09 9.99992609e-01 2.25220695e-08 7.11702832e-15\n",
    "1.89918569e-06 1.03023368e-07 8.88878637e-10 1.41979017e-09\n",
    "5.35583422e-06 6.07789372e-13]\n",
    "[1.00000000e+00 1.34929596e-16 1.43765699e-14 2.60143985e-17\n",
    "2.02902851e-16 6.25009593e-13 1.38456402e-09 4.86662780e-15\n",
    "3.15356907e-11 1.25656317e-11]]\n",
    "import numpy as np\n",
    "print(np.argmax(prediction, axis=1)) #printing our Labels from first 4 images\n",
    "print(y_test[:4]) #printing the actual Labels\n",
    "[7 2 1 0]\n",
    "[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]\n",
    "[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]\n",
    "[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]\n",
    "[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]\n"
   ],
   "metadata": {
    "id": "w8GIPJd7Bh5H"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "markdown",
   "source": [
    "SAVE THE MODEL"
   ],
   "metadata": {
    "id": "_8R2oSXfBtBK"
   }
```

```
},
{
 "cell_type": "code",
 "source": [
  "from google.colab import drive\n",
  "drive.mount('/content/drive')\n",
  "Drive already mounted at /content/drive; to attempt to forcibly remount, call\n",
  "drive.mount(\"/content/drive\", force_remount=True).\n",
  "%cd /content/drive/MyDrive/DISASTER DATASET/dataset.zip\n",
  "/content/drive/MyDrive/DISASTER DATASET/dataset.zip\n",
  "# Save the model\n",
  "model.save('models/disaster.h5')\n",
  "\n"
 ],
 "metadata": {
  "id": "L-UDDRAiB0_j"
 },
 "execution_count": null,
 "outputs": []
},
{
 "cell_type": "markdown",
 "source": [
  "TEST WITH SAVED MODEL"
 ],
 "metadata": {
  "id": "oXzvoxKpCBcr"
 }
},
{
 "cell_type": "code",
```

```
"source": [
  "# Taking images as input and checking results\n",
  "# Importing the keras libraries and packages\n",
  "from tensorflow.keras.models import load_model\n",
  "model = load_model(r:'/content/drive/MyDrive/DISASTER DATASET/dataset.zip')\n",
  "from PIL import Image #used for manipulating image upload by the user.\n",
  "for index in range(0):\n",
  "img = Image.open(\"data/\" + str(index) + \".png\").convert('L') # convert\n",
  "image to monochrome\n",
  "img = img.resize((28,28)) # resizing of input image\n",
  "im2arr = np.array(img) #convert to image\n",
  "im2arr = im2arr.reshape(28,28,1) #reshaping according to our requirement\n",
  "#predicting the Test set results\n",
  "y_pred = model.predict(im2arr)\n",
  "print(y_pred)\n",
  "[[2.6514371e-08 9.9987853e-01 2.6678819e-09 5.0345729e-17 1.1578899e-04\n",
  "3.5318081e-13 1.3091828e-13 5.5813430e-06 1.7989708e-10 2.9838541e-09]]\n",
  "1/1 [==============================] - 0s 17ms/step\n",
  "[[3.7167319e-03 1.4217998e-03 8.2274787e-03 7.9998100e-01 1.1364324e-01\n",
  "2.0436779e-02 6.3992090e-08 3.5715982e-02 3.3161716e-05 1.6823808e-02]]\n",
  "1/1 [==============================] - 0s 16ms/step\n",
  "[[3.89392152e-02 1.03533266e-05 1.84860080e-01 2.39315932e-03\n",
  "2.41028378e-04 8.98893850e-05 1.02114845e-02 3.54044059e-05\n",
  "7.63198674e-01 2.07284338e-05]]\n",
  "index = ['Cyclone' , 'Earthquake' , 'Flood' , 'Wildfire']\n",
  "result = str(index[pred[0]])\n",
  "result\n",
  "'Cyclone'"
],
"metadata": {
  "id": "UAU9w4tZCNHl"
```

```
    },
    "execution_count": null,
    "outputs": []
   }
  ]
}
```