

```

{
  "cells": [
    {
      "cell_type": "markdown",
      "metadata": {},
      "source": [
        "PNT2022TMID21192 Testing The Model."
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "rifIriR8gKIO"
      },
      "source": [
        "##IMPORT LIBRARIES"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 156,
      "metadata": {
        "id": "Olr6hhKBgJIU"
      },
      "outputs": [],
      "source": [
        "from tensorflow.keras.models import Sequential\n",
        "from tensorflow.keras.layers import Dense, Conv2D, Flatten,\nDropout, MaxPooling2D\n",
        "from tensorflow.keras.preprocessing.image import\nImageDataGenerator\n",
        "import numpy as np\n",
        "import matplotlib.pyplot as plt\n",
        "import cv2"
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "kB1D7TDkKn3P"
      },
      "source": [
        "##unzip the file\n",
        "\n"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 1,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "ecgFltU1IIE4",
        "outputId": "43551853-e064-4597-9320-dfde90b59d8c"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [

```

"\u001b[1;30;43mStreaming output truncated to the last 5000
lines.\u001b[0m\n",

" extracting: Dataset/training_set/G/1225.png \n",
" extracting: Dataset/training_set/G/1226.png \n",
" extracting: Dataset/training_set/G/1227.png \n",
" extracting: Dataset/training_set/G/1228.png \n",
" extracting: Dataset/training_set/G/1229.png \n",
" inflating: Dataset/training_set/G/123.png \n",
" extracting: Dataset/training_set/G/1230.png \n",
" extracting: Dataset/training_set/G/1231.png \n",
" extracting: Dataset/training_set/G/1232.png \n",
" inflating: Dataset/training_set/G/1233.png \n",
" inflating: Dataset/training_set/G/1234.png \n",
" inflating: Dataset/training_set/G/1235.png \n",
" inflating: Dataset/training_set/G/1236.png \n",
" inflating: Dataset/training_set/G/1237.png \n",
" inflating: Dataset/training_set/G/1238.png \n",
" inflating: Dataset/training_set/G/1239.png \n",
" inflating: Dataset/training_set/G/124.png \n",
" inflating: Dataset/training_set/G/1240.png \n",
" inflating: Dataset/training_set/G/1241.png \n",
" inflating: Dataset/training_set/G/1242.png \n",
" inflating: Dataset/training_set/G/1243.png \n",
" inflating: Dataset/training_set/G/1244.png \n",
" inflating: Dataset/training_set/G/1245.png \n",
" extracting: Dataset/training_set/G/1246.png \n",
" inflating: Dataset/training_set/G/1247.png \n",
" inflating: Dataset/training_set/G/1248.png \n",
" inflating: Dataset/training_set/G/1249.png \n",
" inflating: Dataset/training_set/G/125.png \n",
" inflating: Dataset/training_set/G/1250.png \n",
" inflating: Dataset/training_set/G/1251.png \n",
" inflating: Dataset/training_set/G/1252.png \n",
" inflating: Dataset/training_set/G/1253.png \n",
" inflating: Dataset/training_set/G/1254.png \n",
" inflating: Dataset/training_set/G/1255.png \n",
" inflating: Dataset/training_set/G/1256.png \n",
" inflating: Dataset/training_set/G/1257.png \n",
" inflating: Dataset/training_set/G/1258.png \n",
" inflating: Dataset/training_set/G/1259.png \n",
" inflating: Dataset/training_set/G/126.png \n",
" inflating: Dataset/training_set/G/1260.png \n",
" inflating: Dataset/training_set/G/1261.png \n",
" extracting: Dataset/training_set/G/1262.png \n",
" inflating: Dataset/training_set/G/1263.png \n",
" inflating: Dataset/training_set/G/1264.png \n",
" inflating: Dataset/training_set/G/1265.png \n",
" inflating: Dataset/training_set/G/1266.png \n",
" inflating: Dataset/training_set/G/1267.png \n",
" extracting: Dataset/training_set/G/1268.png \n",
" inflating: Dataset/training_set/G/1269.png \n",
" inflating: Dataset/training_set/G/127.png \n",
" inflating: Dataset/training_set/G/1270.png \n",
" inflating: Dataset/training_set/G/1271.png \n",
" inflating: Dataset/training_set/G/1272.png \n",
" inflating: Dataset/training_set/G/1273.png \n",
" inflating: Dataset/training_set/G/1274.png \n",
" inflating: Dataset/training_set/G/1275.png \n",
" inflating: Dataset/training_set/G/1276.png \n",
" inflating: Dataset/training_set/G/1277.png \n",
" inflating: Dataset/training_set/G/1278.png \n",

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```

    ]
  },
  {
    "source": [
      "!unzip '/content/drive/MyDrive/IBMPROJECT/conversation engine for
deaf and dumb.zip'"
    ]
  },
  {
    "cell_type": "markdown",
    "metadata": {
      "id": "thhFCF-2KhrP"
    },
    "source": [
      "##DATA AUGMENTATION"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": 35,
    "metadata": {
      "id": "1VGhefJYE2a7"
    },
    "outputs": [],
    "source": [
      "from keras.preprocessing.image import ImageDataGenerator\n",
      "train_datagen=ImageDataGenerator(rescale = 1./255,
shear_range=0.2,
zoom_range=0.2,horizontal_flip=True,vertical_flip=False)\n",
      "test_datagen = ImageDataGenerator(rescale=1./255)"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": 53,
    "metadata": {
      "colab": {
        "base_uri": "https://localhost:8080/"
      },
      "id": "EYejW9eYHfpT",
      "outputId": "e027dfe2-939b-4b73-e518-7dd3ddc22b5d"
    },
    "outputs": [
      {
        "name": "stdout",
        "output_type": "stream",
        "text": [
          "Found 15750 images belonging to 9 classes.\n"
        ]
      }
    ],
    "source": [
      "x_train =
train_datagen.flow_from_directory(\"/content/Dataset/training_set\",
target_size=(64,64),batch_size=100,\n",
      "
class_mode='categorical', color_mode =\"grayscale\")"
    ]
  },
  {
    "cell_type": "code",

```

```

"execution_count": 54,
"metadata": {
  "colab": {
    "base_uri": "https://localhost:8080/"
  },
  "id": "ET_fSsUKHi2P",
  "outputId": "98bccfde-365f-4c31-823d-2087ebd3a27c"
},
"outputs": [
  {
    "name": "stdout",
    "output_type": "stream",
    "text": [
      "Found 2250 images belonging to 9 classes.\n"
    ]
  }
],
"source": [
  "x_test =
test_datagen.flow_from_directory(\"/content/Dataset/test_set\",
target_size=(64,64),batch_size=100,\n",
  "                                     class_mode='categorical',
color_mode =\"grayscale\")"
]
},
{
  "cell_type": "code",
  "execution_count": 55,
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "VmWN1IALKcPk",
    "outputId": "94e013e6-6ede-434e-e605-10fcb8d27f4d"
  },
  "outputs": [
    {
      "data": {
        "text/plain": [
          "158"
        ]
      },
      "execution_count": 55,
      "metadata": {},
      "output_type": "execute_result"
    }
  ],
  "source": [
    "len(x_train)"
  ]
},
{
  "cell_type": "code",
  "execution_count": 56,
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "7IFFqi_yOzsn",
    "outputId": "fbb3d793-aa57-46f4-c757-ebce1e7a6ab5"
  },
  "outputs": [

```

```

"outputs": [
  {
    "data": {
      "text/plain": [
        "23"
      ]
    },
    "execution_count": 56,
    "metadata": {},
    "output_type": "execute_result"
  }
],
"source": [
  "len(x_test)"
]
},
{
  "cell_type": "code",
  "execution_count": 57,
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "jnMxokV6Q1f7",
    "outputId": "bef23bde-5e6a-4b08-e455-7c1cc7a5e201"
  },
  "outputs": [
    {
      "data": {
        "text/plain": [
          "{ 'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}"
        ]
      },
      "execution_count": 57,
      "metadata": {},
      "output_type": "execute_result"
    }
  ],
  "source": [
    "x_train.class_indices"
  ]
},
{
  "cell_type": "markdown",
  "metadata": {
    "id": "XfbAOQT7Kc77"
  },
  "source": [
    "##MODEL BUILDING"
  ]
},
{
  "cell_type": "code",
  "execution_count": 40,
  "metadata": {
    "id": "hkBXqkwzHmvU"
  },
  "outputs": [],
  "source": [
    "from keras.models import Sequential\n",

```

```

        "from keras.layers import Dense\n",
        "from keras.layers import Convolution2D\n",
        "from tensorflow.keras.layers import Conv2D, MaxPooling2D\n",
        "from keras.layers import Dropout\n",
        "from keras.layers import Flatten"
    ]
},
{
    "cell_type": "code",
    "execution_count": 101,
    "metadata": {
        "id": "V5US28ApHnTW"
    },
    "outputs": [],
    "source": [
        "#Creating the model\n",
        "model=Sequential()\n",
        "#Adding the layers\n",
        "model.add(Convolution2D(32, (3,3), input_shape=(64,64,1),\nactivation = 'relu'))\n",
        "model.add(MaxPooling2D(pool_size=(2,2)))\n",
        "model.add(Flatten())\n",
        "\n",
        "#adding hidden layers\n",
        "model.add(Dense(400, activation='relu'))\n",
        "model.add(Dense(200, activation='relu'))\n",
        "model.add(Dense(100, activation='relu'))\n",
        "\n",
        "#Adding the output layer\n",
        "model.add(Dense(9, activation='softmax'))"
    ]
},
{
    "cell_type": "code",
    "execution_count": 102,
    "metadata": {
        "id": "S84mvff6H53r"
    },
    "outputs": [],
    "source": [
        "model.compile(loss='categorical_crossentropy', optimizer='adam',\nmetrics=['accuracy'])"
    ]
},
{
    "cell_type": "code",
    "execution_count": 157,
    "metadata": {
        "colab": {
            "base_uri": "https://localhost:8080/"
        },
        "id": "X5kQdiZog8Cz",
        "outputId": "021af953-da34-461a-a62c-4553bb7dc851"
    },
    "outputs": [
        {
            "name": "stdout",
            "output_type": "stream",
            "text": [
                "Epoch 1/10\n"
            ]
        }
    ]
}

```

```
    },
    {
        "name": "stderr",
        "output_type": "stream",
        "text": [
            "/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.\n",
            "\\"\\\\"Entry point for launching an IPython kernel.\n"
        ]
    },
    {
        "name": "stdout",
        "output_type": "stream",
        "text": [
            "30/30 [=====] - ETA: 0s - loss: 0.0083 - accuracy: 0.9957"
        ]
    },
    {
        "name": "stderr",
        "output_type": "stream",
        "text": [
            "WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batches (in this case, 50 batches). You may need to use the repeat() function when building your dataset.\n"
        ]
    },
    {
        "name": "stdout",
        "output_type": "stream",
        "text": [
            "\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\b\\r30/30 [=====] - 18s 587ms/step - loss: 0.0083 - accuracy: 0.9957 - val_loss: 0.2910 - val_accuracy: 0.9693\n",
            "Epoch 2/10\n",
            "30/30 [=====] - 12s 402ms/step - loss: 0.0081 - accuracy: 0.9980\n",
            "Epoch 3/10\n",
            "30/30 [=====] - 12s 400ms/step - loss: 0.0102 - accuracy: 0.9963\n",
            "Epoch 4/10\n",
            "30/30 [=====] - 12s 402ms/step - loss: 0.0049 - accuracy: 0.9993\n",
            "Epoch 5/10\n",
            "30/30 [=====] - 12s 402ms/step - loss: 0.0030 - accuracy: 0.9997\n",
            "Epoch 6/10\n",
            "30/30 [=====] - 12s 394ms/step - loss: 0.0019 - accuracy: 0.9997\n",
            "Epoch 7/10\n",
            "30/30 [=====] - 12s 401ms/step - loss: 0.0081 - accuracy: 0.9973\n",
            "Epoch 8/10\n",
            "30/30 [=====] - 12s 402ms/step - loss: 0.0124 - accuracy: 0.9960\n",
```



```

        "Epoch 9/10\n",
        "30/30 [=====] - 12s 401ms/step -
loss: 0.0070 - accuracy: 0.9987\n",
        "Epoch 10/10\n",
        "30/30 [=====] - 12s 399ms/step -
loss: 0.0089 - accuracy: 0.9973\n"
    ]
},
{
    "data": {
        "text/plain": [
            "<keras.callbacks.History at 0x7f8d4bd41cd0>"
        ]
    },
    "execution_count": 157,
    "metadata": {},
    "output_type": "execute_result"
},
{
    "source": [
        "model.fit_generator(x_train, steps_per_epoch=30, epochs=10,
validation_data=x_test, validation_steps=50)"
    ]
},
{
    "cell_type": "code",
    "execution_count": 77,
    "metadata": {
        "id": "tDXeRBOcIDoX"
    },
    "outputs": [],
    "source": [
        "model.save('Real_time.h5')"
    ]
},
{
    "cell_type": "markdown",
    "metadata": {
        "id": "1dnUiQGCLkHc"
    },
    "source": [
        "##TEST THE MODEL\n",
        "\n",
        "\n"
    ]
},
{
    "cell_type": "code",
    "execution_count": 104,
    "metadata": {
        "id": "eWzt2n82H50m"
    },
    "outputs": [],
    "source": [
        "from tensorflow.keras.models import load_model\n",
        "from tensorflow.keras.preprocessing import image\n",
        "import numpy as np\n",
        "import cv2"
    ]
},
{

```

```

"cell_type": "code",
"execution_count": 105,
"metadata": {
  "id": "JTC-UdX0LtYA"
},
"outputs": [],
"source": [
  "model = load_model('/content/Real_time.h5')"
]
},
{
  "cell_type": "code",
  "execution_count": 151,
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/",
      "height": 117
    },
    "id": "TKWIEDoHL1aT",
    "outputId": "ea8f3473-clb2-4eca-e7be-0f9fdaf04120"
  },
  "outputs": [
    {
      "data": {
        "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAGQAAABkCAIAAAD/gAIDAAAC4U1EQVR4nO3cvUrzUBgH8Oe8vE
OK0EJw8AN0k4YM2UQyZXTo4Ngr8BKKl1AnL8ALaKFD6SDBNSCiizrpFXTpls khmHc4r7FEanryd
Z7U/2+KtT15eHj+8aRvIQAAAAAAAAAAAAAAAAAAAAAAAAAA4LcTmc+4uLiQB7e3t0T09PRUbUVr
SwpbNhwOqzvjmY5jvP8/FzdiazfX1NROfn56Wv/Kf0FX+jfr8fN9n15WXpPcFkKdjYZg0GA8M
wDMMocc2/qa+73e7V1RURnZ6elniazbcXklWF9NYhjmMtdVTn/f2diFqtVvG1viZL/hApviI38s
r1+vpafCnEUMFXDDdyrJYJkX1v17FCaq2Pj4+CK7L19vZGRJZ15V4BMVSQnszj4+OHhwctpdSjS
BgxWQrSzXp8fBRCCCHOzs7CMAzDUETz1Ynj2Pd93/dzvDZ9u5OYzWadTmf5EblV2dvba7fbOc7E
x9bWVr4XIoYKcm097u/v5cHJyUkZxdRN9WJfdJ/2M3lp2N3ddRyn0hPlc3NzQ0S9Xm/N5yOGCqg
drO9Go1Fy7LruwFBzQV8t34Y627WMj53o/JTtcy37RFDBXqaxe29M8/zPM/LfJqeGLLqVMoPlz
DEUIGeZgVBEASB11Nn6vf7q76FyWoI13W1fLKfaVXBOvdZxPtKblmWfCc6gRgqQLNWmkwmqUcQw
2zJzguTpUBzs0zTNE1Tbw2ZdnZ25IHmGER8wyiTiBgqYDFZ1IThIkyWEi7NiqIoiiLdVWTgEkOJ
eRi5TFYj8GpW8d83qxTT4njmkddkMcd0siRu88V6soQQ4/F4PB7rLuQ/ls3ihnUME0zy2IzJ2t/
f110CUVOaxUQzmjWfz8Un27Zt255Op/WX0YxmQZnu7u40fvjaMI7j1NAsxFBBM/ZZ6+h2u/T5lw
2wlsPDQ8SQhc2JYWJ7e5uIFotF6StvYlMSqf+AcXR0REQvLy+5F0QMFfwDzs0UMA7yqYwAAAAAS
UVORK5CYII=",
        "text/plain": [
          "<PIL.Image.Image image mode=RGB size=100x100 at
0x7F8D5DAA7050>"
        ]
      },
      "execution_count": 151,
      "metadata": {},
      "output_type": "execute_result"
    }
  ],
  "source": [
    "img =
image.load_img('/content/Dataset/test_set/H/107.png',target_size =
(100,100))\n",
    "img"
  ]
},
{
  "cell_type": "code",
  "execution_count": 149,
  "metadata": {

```

```

      "id": "jflfQxF0VZqe"
    },
    "outputs": [],
    "source": [
      "from skimage.transform import resize\n",
      "def detect(frame):\n",
      "    img=image.img_to_array(frame)\n",
      "    img = resize(img, (64,64,1))\n",
      "    img = np.expand_dims(img,axis=0)\n",
      "    pred=np.argmax(model.predict(img))\n",
      "    op=['A','B','C','D','E','F','G','H','I']\n",
      "    print(\"THE PREDICTED LETTER IS \",op[pred])"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": 150,
    "metadata": {
      "colab": {
        "base_uri": "https://localhost:8080/"
      },
      "id": "xAEhpLN0XTQJ",
      "outputId": "b4723c45-c267-423d-ab62-1234e0bdd02f"
    },
    "outputs": [
      {
        "name": "stdout",
        "output_type": "stream",
        "text": [
          "1/1 [=====] - 0s 28ms/step\n",
          "THE PREDICTED LETTER IS  H\n"
        ]
      }
    ],
    "source": [
      "img=image.load_img(\"/content/Dataset/test_set/H/107.png\")\n",
      "detect(img)"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": 155,
    "metadata": {
      "colab": {
        "base_uri": "https://localhost:8080/"
      },
      "id": "bY4Rq7oaXdHt",
      "outputId": "20d402bd-9d31-4940-d1f9-0102f65aa916"
    },
    "outputs": [
      {
        "name": "stdout",
        "output_type": "stream",
        "text": [
          "1/1 [=====] - 0s 26ms/step\n",
          "THE PREDICTED LETTER IS  A\n"
        ]
      }
    ],
    "source": [
      "img = image.load_img('/content/Dataset/test_set/A/110.png')\n",

```

```

        "pred=detect(img)"
    ]
},
{
    "cell_type": "code",
    "execution_count": 158,
    "metadata": {
        "colab": {
            "base_uri": "https://localhost:8080/"
        },
        "id": "JcI8nejybLG8",
        "outputId": "30fc86bc-d9a3-4632-8aff-7beeb962e01f"
    },
    "outputs": [
        {
            "name": "stdout",
            "output_type": "stream",
            "text": [
                "1/1 [=====] - 0s 30ms/step\n",
                "THE PREDICTED LETTER IS  E\n"
            ]
        }
    ],
    "source": [
        "img=image.load_img('/content/Dataset/test_set/E/111.png')\n",
        "detect(img)"
    ]
}
],
"metadata": {
    "colab": {
        "provenance": []
    },
    "kernelspec": {
        "display_name": "Python 3",
        "name": "python3"
    },
    "language_info": {
        "name": "python"
    }
},
"nbformat": 4,
"nbformat_minor": 0
}

```