PNT2022TMID50371


```
{
 "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": 2,
   "metadata": {
    "id": "yhrkzc9ZpmOU"
   },
   "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)\n",
 "test_datagen=ImageDataGenerator(rescale=1./255)"
]
},
{
 "cell_type": "code",
 "source": [
 "x_train = train_datagen.flow_from_directory('/content/Dataset/training_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode=\"grayscale\")"
],
 "metadata": {
 "id": "911jlfgkNnqq",
 "colab": {
 "base_uri": "https://localhost:8080/"
 },
 "outputId": "44f120a6-86ad-47c5-c78a-b8329539d64c"
},
 "execution_count": 3,
 "outputs": [
 {
 "output_type": "stream",
 "name": "stdout",
```

```
        "text": [
          "Found 15750 images belonging to 9 classes.\n"
        ]
       }
     ]
    },
    {
     "cell_type": "code",
     "source": [
      "x_test =
test_datagen.flow_from_directory('/content/Dataset/test_set',target_size=(64,64),batch_size=300,class
_mode='categorical',color_mode=\"grayscale\")"
     ],
     "metadata": {
      "colab": {
       "base_uri": "https://localhost:8080/"
      },
      "id": "0jQeqRgJQ_k3",
      "outputId": "d87d023d-a162-431d-e065-6886122187a2"
     },
     "execution_count": 4,
     "outputs": [
      {
       "output_type": "stream",
       "name": "stdout",
       "text": [
```

```
        "Found 2250 images belonging to 9 classes.\n"
      ]
     }
    ]
   },
   {
    "cell_type": "code",
    "source": [
     "from keras.models import Sequential\n",
     "from keras.layers import Dense\n",
     "from keras.layers import Convolution2D\n",
     "from keras.layers import MaxPooling2D\n",
     "from keras.layers import Dropout\n",
     "from keras.layers import Flatten"
    ],
    "metadata": {
     "id": "hENrq9luV5CV"
    },
    "execution_count": 5,
    "outputs": []
   },
   {
    "cell_type": "code",
    "source": [
     "model = Sequential()"
```

```
 ],
 "metadata": {
  "id": "_Xog4wM-WHQL"
 },
 "execution_count": 6,
 "outputs": []
},
{
 "cell_type": "code",
 "source": [
  "model.add(Convolution2D(32,(3,3),input_shape=(64,64,1), activation='relu'))\n",
  "#no. of feature detectors, size of feature detector, image size, activation function"
 ],
 "metadata": {
  "id": "IxBznrvkXOiQ"
 },
 "execution_count": 7,
 "outputs": []
},
{
 "cell_type": "code",
 "source": [
  "model.add(MaxPooling2D(pool_size=(2,2)))"
 ],
 "metadata": {
```

```
    "id": "m9i6nyiiYAzH"
   },
   "execution_count": 8,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "model.add(Flatten())"
   ],
   "metadata": {
    "id": "YrEJW7pAYFA4"
   },
   "execution_count": 9,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "model.add(Dense(units=512, activation = 'relu'))"
   ],
   "metadata": {
    "id": "qIvMupXlYg8d"
   },
   "execution_count": 10,
```

```
  "outputs": []
 },
 {
  "cell_type": "code",
  "source": [
   "model.add(Dense(units=9,  activation = 'softmax'))"
  ],
  "metadata": {
   "id": "BSaehFfcY4iz"
  },
  "execution_count": 11,
  "outputs": []
 },
 {
  "cell_type": "code",
  "source": [
   "model.compile(loss='categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])"
  ],
  "metadata": {
   "id": "Dq7W6q62Y9RC"
  },
  "execution_count": 12,
  "outputs": []
 },
 {
```

```
"cell_type": "code",

"source": [

 "model.fit_generator(x_train,steps_per_epoch=24,epochs=10,validation_data = x_test, validation_steps= 40)\n",

 "#steps_per_epoch = no. of train images//batch size"

],

"metadata": {

 "colab": {

  "base_uri": "https://localhost:8080/"

 },

 "id": "T530ZkC6ZSOk",

 "outputId": "0adc68b9-1573-482b-87bd-fa1d56cb730f"

},

"execution_count": 13,

"outputs": [

 {

  "output_type": "stream",

  "name": "stderr",

  "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"

  ]

 },

 {

  "output_type": "stream",
```

"name": "stdout",

"text": [

"Epoch 1/10\n",

"24/24 [==============================] - ETA: 0s - loss: 1.2714 - accuracy: 0.6219"

]

},

{

"output_type": "stream",

"name": "stderr",

"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, 40 batches). You may need to use the repeat() function when building your dataset.\n"

]

},

{

"output_type": "stream",

"name": "stdout",

"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\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\r24/24
[==============================] - 41s 2s/step - loss: 1.2714 - accuracy: 0.6219 - val_loss: 0.4031 - val_accuracy: 0.8982\n",

"Epoch 2/10\n",

"24/24 [==============================] - 33s 1s/step - loss: 0.2827 - accuracy: 0.9211\n",

"Epoch 3/10\n",

"24/24 [==============================] - 34s 1s/step - loss: 0.1448 - accuracy: 0.9615\n",

```
    "Epoch 4/10\n",
    "24/24 [==============================] - 32s 1s/step - loss: 0.0958 - accuracy: 0.9746\n",
    "Epoch 5/10\n",
    "24/24 [==============================] - 34s 1s/step - loss: 0.0679 - accuracy: 0.9826\n",
    "Epoch 6/10\n",
    "24/24 [==============================] - 32s 1s/step - loss: 0.0424 - accuracy: 0.9909\n",
    "Epoch 7/10\n",
    "24/24 [==============================] - 32s 1s/step - loss: 0.0373 - accuracy: 0.9908\n",
    "Epoch 8/10\n",
    "24/24 [==============================] - 33s 1s/step - loss: 0.0319 - accuracy: 0.9915\n",
    "Epoch 9/10\n",
    "24/24 [==============================] - 32s 1s/step - loss: 0.0235 - accuracy: 0.9940\n",
    "Epoch 10/10\n",
    "24/24 [==============================] - 32s 1s/step - loss: 0.0170 - accuracy: 0.9972\n"
   ]
  },
  {
   "output_type": "execute_result",
   "data": {
    "text/plain": [
     "<keras.callbacks.History at 0x7fe3bd2e8c90>"
    ]
   },
   "metadata": {},
   "execution_count": 13
```

```
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "model.save('aslpng1.h5')"
   ],
   "metadata": {
    "id": "tbD4YC8VZlIB"
   },
   "execution_count": 14,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "from keras.models import load_model\n",
    "import numpy as np\n",
    "import cv2"
   ],
   "metadata": {
    "id": "wBCEfO5qd0Gj"
   },
   "execution_count": 17,
```

```
  "outputs": []
 },
 {
  "cell_type": "code",
  "source": [
   "model=load_model('aslpng1.h5')"
  ],
  "metadata": {
   "id": "sZYDfTiuZmUU"
  },
  "execution_count": 18,
  "outputs": []
 },
 {
  "cell_type": "code",
  "source": [
   "from skimage.transform import resize\n",
   "def detect(frame):\n",
   "  img = resize(frame,(64,64,1))\n",
   "  img = np.expand_dims(img,axis=0)\n",
   "  if(np.max(img)>1):\n",
   "    img = img/255.0\n",
   "  prediction = model.predict(img)\n",
   "  print(prediction)\n",
   "  prediction = np.argmax(prediction,axis=1)\n",
```

```
    "  print(prediction)"
   ],
   "metadata": {
    "id": "05vEcPg2bJfW"
   },
   "execution_count": 25,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "frame=cv2.imread('/content/Dataset/test_set/G/1.png')\n",
    "data = detect(frame)"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "bdi1cEHebj35",
    "outputId": "147bc04d-4759-43cd-90ab-fd53ee38317e"
   },
   "execution_count": 26,
   "outputs": [
    {
     "output_type": "stream",
```

```
        "name": "stdout",
        "text": [
         "1/1 [==============================] - 0s 25ms/step\n",
         "[[2.9662006e-09 3.0511607e-09 5.7518361e-07 2.6636766e-09 7.6029876e-09\n",
         "  1.4324395e-08 9.9982303e-01 1.7639149e-04 1.6517550e-09]]\n",
         "[6]\n"
        ]
       }
     ]
    },
    {
     "cell_type": "code",
     "source": [],
     "metadata": {
      "id": "LQ1_aUY3b1qJ"
     },
     "execution_count": null,
     "outputs": []
    }
  ]
}
```