{

"nbformat": 4,

"nbformat\_minor": 0,

"metadata": {

"colab": {

"provenance": []

},

"kernelspec": {

"name": "python3",

"display\_name": "Python 3"

},

"language\_info": {

"name": "python"

}

},

"cells": [

{

"cell\_type": "markdown",

"source": [

"# Task-1: - Perform following operations on image\n"

],

"metadata": {

"id": "W75VXEmlF3B6"

}

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "BTL1xkboA3eH"

},

"outputs": [],

"source": [

"import cv2\n",

"import numpy as np\n",

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

]

},

{

"cell\_type": "code",

"source": [

"image\_path = 'Lena\_Image.png'\n",

"gray\_image = cv2.imread(image\_path)\n",

"image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)"

],

"metadata": {

"id": "mAJ9arHLCVU\_"

},

"execution\_count": null,

"outputs": []

},

{

"source": [

"resized\_linear = cv2.resize(image, (200, 200), interpolation=cv2.INTER\_LINEAR)\n",

"cv2\_imshow(resized\_linear)\n"

],

"cell\_type": "code",

"metadata": {

"colab": {

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

"height": 217

},

"id": "4iBXW-6mDG5R",

"outputId": "e493e386-9a40-43cf-9dd5-e3bc23c77282"

},

"execution\_count": null,

"outputs": [

{

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

"data": {

"text/plain": [

"<PIL.Image.Image image mode=L size=200x200>"

],

"image/png": "\n",

"image/jpeg": "\n"

},

"metadata": {}

}

]

},

{

"cell\_type": "code",

"source": [

"resized\_nearest = cv2.resize(image, (400, 400), interpolation=cv2.INTER\_NEAREST)\n",

"cv2\_imshow(resized\_nearest)"

],

"metadata": {

"colab": {

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

"height": 417

},

"id": "\_yLH3e9RD8Vd",

"outputId": "c61537e8-afa9-46c5-d067-69e581fdd311"

},

"execution\_count": null,

"outputs": [

{

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

"data": {

"text/plain": [

"<PIL.Image.Image image mode=L size=400x400>"

],

"image/png": "\n",

"image/jpeg": "\n"

},

"metadata": {}

}

]

},

{

"cell\_type": "code",

"source": [

"resized\_cubic= cv2.resize(image, (600, 600), interpolation=cv2.INTER\_CUBIC)\n",

"cv2\_imshow(resized\_cubic)"

],

"metadata": {

"colab": {

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

"height": 617

},

"id": "aLz1jYWpEFzN",

"outputId": "b9e9d771-6448-4587-e31b-af1c0a678e37"

},

"execution\_count": null,

"outputs": [

{

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

"data": {

"text/plain": [

"<PIL.Image.Image image mode=L size=600x600>"

],

"image/png": "\n",

"image/jpeg": "\n"

},

"metadata": {}

}

]

},

{

"cell\_type": "code",

"source": [

"box\_blur= cv2.blur(image, (5, 5))\n",

"cv2\_imshow(box\_blur)"

],

"metadata": {

"colab": {

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

"height": 529

},

"id": "rJOvqwKXE3jO",

"outputId": "4b7128f3-fd21-4f10-a9dc-7043934fba69"

},

"execution\_count": null,

"outputs": [

{

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

"data": {

"text/plain": [

"<PIL.Image.Image image mode=L size=512x512>"

],

"image/png": "\n",

"image/jpeg": "\n"

},

"metadata": {}

}

]

},

{

"cell\_type": "code",

"source": [

"gaussian\_blur= cv2.GaussianBlur(image, (5, 5), 0)\n",

"cv2\_imshow(gaussian\_blur)"

],

"metadata": {

"colab": {

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

"height": 529

},

"id": "GAfDAnjCE82w",

"outputId": "4e849917-1e1d-4545-f024-b718adc54839"

},

"execution\_count": null,

"outputs": [

{

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

"data": {

"text/plain": [

"<PIL.Image.Image image mode=L size=512x512>"

],

"image/png": "\n",

"image/jpeg": "\n"

},

"metadata": {}

}

]

},

{

"source": [

"adaptive\_blur = cv2.bilateralFilter(image, 9, 75, 75)\n",

"cv2\_imshow(adaptive\_blur)"

],

"cell\_type": "code",

"metadata": {

"colab": {

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

"height": 529

},

"id": "dkFTtkIPFOwq",

"outputId": "fd140b6e-912c-4965-b338-d3652df4c1a3"

},

"execution\_count": null,

"outputs": [

{

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

"data": {

"text/plain": [

"<PIL.Image.Image image mode=L size=512x512>"

],

"image/png": "\n",

"image/jpeg": "\n"

},

"metadata": {}

}

]

},

{

"cell\_type": "markdown",

"source": [

"# Task-2: - Apply Machine Learning Algorithm and find the model accuracy based on K fold Cross Validation with (80-20 train-test split). \n"

],

"metadata": {

"id": "Nc1jR0akGBNC"

}

},

{

"cell\_type": "code",

"source": [

"import numpy as np\n",

"import pandas as pd\n",

"from sklearn import metrics\n",

"from sklearn.model\_selection import train\_test\_split\n",

"from sklearn.naive\_bayes import GaussianNB\n",

"from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier\n",

"from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix, roc\_auc\_score, roc\_curve\n",

"from keras.datasets import mnist\n",

"import matplotlib.pyplot as plt\n",

"\n",

"# Load the MNIST dataset\n",

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

"\n",

"# Preprocess the data\n",

"x\_train = x\_train.reshape(x\_train.shape[0], -1) / 255.0\n",

"x\_test = x\_test.reshape(x\_test.shape[0], -1) / 255.0\n",

"\n",

"# Select two algorithms to implement (Naive Bayes and Random Forest for example)\n",

"\n",

"# 1. Naive Bayes Classifier\n",

"nb\_model = GaussianNB()\n",

"nb\_model.fit(x\_train, y\_train)\n",

"nb\_predictions = nb\_model.predict(x\_test)\n",

"\n",

"# Metrics for Naive Bayes\n",

"nb\_accuracy = accuracy\_score(y\_test, nb\_predictions)\n",

"nb\_precision = precision\_score(y\_test, nb\_predictions, average='weighted')\n",

"nb\_recall = recall\_score(y\_test, nb\_predictions, average='weighted')\n",

"nb\_f1 = f1\_score(y\_test, nb\_predictions, average='weighted')\n",

"nb\_cm = confusion\_matrix(y\_test, nb\_predictions)\n",

"\n",

"# 2. Random Forest Classifier\n",

"rf\_model = RandomForestClassifier(n\_estimators=100, random\_state=42)\n",

"rf\_model.fit(x\_train, y\_train)\n",

"rf\_predictions = rf\_model.predict(x\_test)\n",

"\n",

"# Metrics for Random Forest\n",

"rf\_accuracy = accuracy\_score(y\_test, rf\_predictions)\n",

"rf\_precision = precision\_score(y\_test, rf\_predictions, average='weighted')\n",

"rf\_recall = recall\_score(y\_test, rf\_predictions, average='weighted')\n",

"rf\_f1 = f1\_score(y\_test, rf\_predictions, average='weighted')\n",

"rf\_cm = confusion\_matrix(y\_test, rf\_predictions)\n",

"\n",

"# ROC and AUC for Random Forest\n",

"rf\_probs = rf\_model.predict\_proba(x\_test)\n",

"rf\_roc\_auc = roc\_auc\_score(y\_test, rf\_probs, multi\_class='ovr')\n",

"rf\_fpr, rf\_tpr, \_ = roc\_curve(y\_test, rf\_probs[:, 1], pos\_label=1)\n",

"\n",

"# Display Results\n",

"print(\"Naive Bayes Results:\")\n",

"print(f\"Accuracy: {nb\_accuracy}\")\n",

"print(f\"Precision: {nb\_precision}\")\n",

"print(f\"Recall: {nb\_recall}\")\n",

"print(f\"F1 Score: {nb\_f1}\")\n",

"print(f\"Confusion Matrix:\\n{nb\_cm}\")\n",

"\n",

"print(\"\\nRandom Forest Results:\")\n",

"print(f\"Accuracy: {rf\_accuracy}\")\n",

"print(f\"Precision: {rf\_precision}\")\n",

"print(f\"Recall: {rf\_recall}\")\n",

"print(f\"F1 Score: {rf\_f1}\")\n",

"print(f\"Confusion Matrix:\\n{rf\_cm}\")\n",

"print(f\"AUC: {rf\_roc\_auc}\")\n",

"\n",

"# Plot ROC Curve for Random Forest\n",

"plt.figure()\n",

"plt.plot(rf\_fpr, rf\_tpr, label=f\"Random Forest (AUC = {rf\_roc\_auc:.2f})\")\n",

"plt.xlabel(\"False Positive Rate\")\n",

"plt.ylabel(\"True Positive Rate\")\n",

"plt.title(\"ROC Curve\")\n",

"plt.legend(loc=\"lower right\")\n",

"plt.show()\n"

],

"metadata": {

"colab": {

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

},

"id": "WL-GTW3fGIpd",

"outputId": "6d506057-3160-486b-d7c0-38951042f6bd"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Naive Bayes Results:\n",

"Accuracy: 0.5558\n",

"Precision: 0.6917258486386416\n",

"Recall: 0.5558\n",

"F1 Score: 0.5170420327932064\n",

"Confusion Matrix:\n",

"[[ 870 0 3 5 2 5 31 1 35 28]\n",

" [ 0 1079 2 1 0 0 10 0 38 5]\n",

" [ 79 25 266 91 5 2 269 4 271 20]\n",

" [ 32 39 6 353 2 3 51 8 409 107]\n",

" [ 19 2 5 4 168 7 63 7 210 497]\n",

" [ 71 25 1 20 3 44 40 2 586 100]\n",

" [ 12 12 3 1 1 7 895 0 26 1]\n",

" [ 0 15 2 10 5 1 5 280 39 671]\n",

" [ 13 72 3 7 3 11 12 4 648 201]\n",

" [ 5 7 3 6 1 0 1 13 18 955]]\n",

"\n",

"Random Forest Results:\n",

"Accuracy: 0.9704\n",

"Precision: 0.9703956228285179\n",

"Recall: 0.9704\n",

"F1 Score: 0.9703722598097009\n",

"Confusion Matrix:\n",

"[[ 971 0 0 0 0 2 3 1 3 0]\n",

" [ 0 1127 2 2 0 1 2 0 1 0]\n",

" [ 6 0 1002 5 3 0 3 8 5 0]\n",

" [ 1 0 9 972 0 9 0 9 8 2]\n",

" [ 1 0 0 0 955 0 5 1 4 16]\n",

" [ 5 1 1 9 2 860 5 2 5 2]\n",

" [ 7 3 0 0 3 3 937 0 5 0]\n",

" [ 1 4 20 2 0 0 0 989 2 10]\n",

" [ 4 0 6 7 5 5 5 4 930 8]\n",

" [ 7 6 2 12 12 1 0 4 4 961]]\n",

"AUC: 0.9991475561626355\n"

]

},

{

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

"data": {

"text/plain": [

"<Figure size 640x480 with 1 Axes>"

],

"image/png": "\n"

},

"metadata": {}

}

]

}

]

}