MANUAL DE USUARIO

Generación del conjunto de datos

El script de MATLAB "createDataset.m" permite crear automáticamente los conjuntos de datos de entrenamiento (80%) y test (20%) y con un factor de reducción de las imágenes de 2.5. Solo es necesario ejecutarlo sobre el directorio donde se encuentra el conjunto de datos anotados con la aplicación Turbot del Grupo de Aplicaciones Multimedia y Acústica.

Generación de mapas de densidad

El *script* para generar los mapas de densidad es "gen_density_maps.py". Genera dos archivos, uno para el conjunto de entrenamiento y otro para el de test, en formato de archivo comprimido de Numpy (extensión npz). Los argumentos que recibe como entrada mediante línea de comandos son los siguientes:

- **dataset_rootdir**: ruta donde se encuentra el conjunto de datos sobre el que generar los mapas de densidad.
- **sigma_fixed**: Determina si se utiliza una desviación estándar fija para el *kernel* gaussiano o una estrategia de geometría adaptativa (ver apartado **¡Error! No se encuentra el origen de la referencia.**). Toma el valor "True" para usar sigma fija y "False" en caso contrario.
- **sigma**: si se ha optado por usar una desviación estándar fija para el *kernel* gaussiano se indica aquí el valor a usar.
- **knn**: en caso de usar geometría adaptative, número de vecinos más cercanos para calcular la distancia media.
- **max_knn_avg_dist**: en caso de usar geometría adaptative, valor máximo de la distancia media. En caso de que el valor calculado sea superior se sustituirá por el indicado.
- **sigma_coef**: en caso de usar geometría adaptative, valor del coeficiente por el que se multiplica la distancia media para obtener el valor de desviación estándar para el *kernel* gaussiano.
- **sqr_side**: Limitación en pixels de la amplitude del *kernel* gaussiano, de tal forma que solo afectará la función a [-sqr_side/2, +sqr_side/2] pixels en torno a cada punto etiquetado.
- **num_proc**: número de proceso en paralelo para ejecutar el algoritmo de generación.

Ejemplo de uso

```
gen_density_maps.py --dataset_rootdir=./Turbots/ --sigma_fixed True --sigma 4
```

Entrenamiento del modelo

El *script* para entrenar el modelo es "train.py". El script genera los archivos con el modelo en la carpeta "./checkpoints". Además, genera archivos compatibles con Tensorboard para analizar la evolución del entrenamiento. Los argumentos que recibe como entrada mediante línea de comandos son los siguientes:

- **dataset_rootdir**: ruta donde se encuentra el conjunto de datos sobre el que generar los mapas de densidad.
- **densmaps_gt_npz**: ruta a los archivos de mapas de densidad en la forma "densitymap_*.npz" donde el asterisco sustituye a las palabras *train* y *test*. Ver apartado **¡Error! No se encuentra el origen de la referencia.** para información sobre el formato de los mapas de densidad.
- **num_intervals**: valor de C_{Max} para el contador.

Ejemplo de uso

```
train.py --dataset_rootdir=./Turbots/ --dataset_rootdir=./Turbots/densitymap_*.npz
--num_intervals 4
```

Además, el archivo "config_train_val_test.yaml" contiene otros parámetros sobre la configuración del modelo ver apartado **¡Error! No se encuentra el origen de la referencia.** para más información sobre los mismos.

Evaluación del modelo

El *script* para entrenar el modelo es "evaluate.py". A partir de un modelo ya entrenado calculas las métricas de error MAE, RMSE y MAPE tanto para los conjuntos de entrenamiento como de test.

Los argumentos que recibe como entrada mediante línea de comandos son los siguientes:

- **dataset_rootdir**: ruta donde se encuentra el conjunto de datos sobre el que generar los mapas de densidad.
- **densmaps_gt_npz**: ruta a los archivos de mapas de densidad en la forma "densitymap_*.npz" donde el asterisco sustituye a las palabras *train* y *test*. Ver apartado **¡Error! No se encuentra el origen de la referencia.** para información sobre el formato de los mapas de densidad.
- **num_intervals**: valor de C_{Max} para el contador.
- **trained_ckpt_for_inference**: ubicación del archivo con el modelo a evaluar.
- **visualize**: "True" para que genere imágenes con la visualización de la predicción para el conjunto de test. "False" en caso contrario y por defecto. Las imágenes se almacenan en el directorio "visualized_test_predictions".

Ejemplo de uso

```
evaluate.py --dataset_rootdir=./Turbots/ --dataset_rootdir=./Turbots/densitymap_*.npz
--num_intervals 4 --trained_ckpt_for_inference=./checkpoints/epoch_0200.pth --visualized
True
```

Además, el archivo "config_train_val_test.yaml" contiene otros parámetros sobre la configuración del modelo ver apartado **¡Error! No se encuentra el origen de la referencia.** para más información sobre los mismos.

Inferencia de nuevas imágenes

Existen dos *scripts* para la inferencia de nuevas imágenes con un modelo ya entrenado "inferenceCPU.py" y "inference.py". La única diferencia es que el primero utiliza solo la capacidad de computación de la CPU, y el segundo aprovecha la de la GPU mediante CUDA.

Los argumentos que recibe como entrada mediante línea de comandos son los siguientes:

- **num_intervals**: valor de C_{Max} para el contador.
- **trained_ckpt_for_inference**: ubicación del archivo con el modelo a evaluar.
- **imgs_for_inference_dir**: ruta al directorio que contiene las imágenes para la inferencia.

Ejemplo de uso

```
inference.py --num_intervals 4 --trained_ckpt_for_inference=./checkpoints/epoch_0200.pth
--imgs_for_inference_dir=./images/
```

Además, el archivo "config_train_val_test.yaml" contiene otros parámetros sobre la configuración del modelo ver apartado **¡Error! No se encuentra el origen de la referencia.** para más información sobre los mismos.