# Explicación Detallada del Script de Inferencia - StrainNet

## Descripción General

Este documento explica el funcionamiento detallado del script de inferencia desarrollado para predecir mapas de desplazamiento utilizando el modelo StrainNet. Estos desplazamientos son esenciales para la reconstrucción de información de fase en aplicaciones como imágenes de contraste por moteo de rayos X.

## 1. Propósito General del Código

El objetivo principal del script es procesar un conjunto de pares de imágenes (Ref y Def) y generar mapas de desplazamiento (𝑢𝑥, 𝑢𝑦) que describen cómo se ha deformado el patrón en la imagen Def en comparación con la imagen Ref. Estos desplazamientos son fundamentales en aplicaciones donde las diferencias en los patrones permiten inferir información sobre las propiedades físicas de una muestra.

## 2. Argumentos de Entrada

El script acepta varios parámetros configurables desde la línea de comandos mediante la biblioteca argparse. Estos incluyen:

* Arquitectura del modelo (--arch): Permite seleccionar entre diferentes variantes de StrainNet (StrainNet\_f, StrainNet\_h, StrainNet\_l).  
  Carpeta de datos (data): Ruta que contiene los archivos de imágenes Ref y Def.  
  Modelo preentrenado (--pretrained): Archivo .pth que contiene los pesos del modelo previamente entrenado.  
  Carpeta de salida (--output): Directorio donde se guardarán los resultados.  
  Extensiones de archivo (--img-exts): Define las extensiones de archivo a procesar (por defecto .csv).

## 3. Flujo General del Script

El script sigue un flujo de trabajo estructurado para lograr el objetivo de inferencia:

### 3.1. Inicialización

Configura las rutas de entrada y salida, asegurando que existan directorios donde se guardarán los resultados. Se identifican los pares de imágenes Ref y Def dentro de la carpeta de datos, asociando cada archivo Ref con su correspondiente archivo Def.

### 3.2. Configuración del Dispositivo

Detecta automáticamente si hay una GPU disponible (torch.cuda.is\_available()) y la utiliza si está presente, optimizando el rendimiento del modelo durante la inferencia. Si no hay GPU disponible, se ejecuta en la CPU.

### 3.3. Carga del Modelo Preentrenado

El modelo preentrenado se carga desde el archivo proporcionado (.pth) y se inicializa en modo de evaluación (desactivando dropout y batch normalization).

### 3.4. Procesamiento de Pares de Imágenes

Cada par Ref-Def se carga desde los archivos CSV correspondientes, se normaliza y se convierte a tensores. Las imágenes Ref y Def se concatenan para formar un tensor de entrada.

### 3.5. Inferencia

El modelo predice los mapas de desplazamiento en las direcciones 𝑥 e 𝑦. Si es necesario, la salida se ajusta al tamaño de entrada mediante interpolación bilineal.

### 3.6. Desnormalización y Guardado

Los mapas de desplazamiento se desnormalizan y se guardan como archivos CSV, con nombres derivados de los archivos Ref-Def procesados.

## 4. Manejo de Errores y Advertencias

El script incluye múltiples niveles de manejo de errores para garantizar una ejecución robusta. Esto incluye advertencias si no se encuentran archivos Def correspondientes, errores en la carga del modelo y errores durante el procesamiento de pares específicos.

## 5. Resultados y Salida

Por cada par de imágenes procesado, el script genera dos mapas de desplazamiento:

* disp\_x.csv: Mapa de desplazamiento en la dirección horizontal (𝑢𝑥).  
  disp\_y.csv: Mapa de desplazamiento en la dirección vertical (𝑢𝑦).

Estos resultados son fundamentales para reconstruir las deformaciones sufridas por los patrones de moteo.

## 6. Aplicaciones Prácticas

Este flujo de trabajo es particularmente útil en aplicaciones donde se necesita mapear pequeños desplazamientos entre patrones, como:

* Reconstrucción de fase en imágenes de rayos X.  
  Monitorización de deformaciones en materiales.  
  Análisis de estructuras internas en tejidos blandos.

# Explicación General del Código MultiscaleLoss

El código multiscaleloss está diseñado para calcular errores en mapas de flujo entre una predicción generada por un modelo y un flujo objetivo, aplicando evaluaciones a múltiples escalas. Esto es especialmente útil en tareas como la estimación de flujo óptico o desplazamientos en imágenes, donde las resoluciones varían a lo largo de diferentes etapas de un modelo.

## Propósito Principal

El propósito general del código es medir la discrepancia entre las predicciones de un modelo y un conjunto de valores objetivo. Utiliza métricas específicas como el Error de Punto Final (EPE) y su extensión multiescala para evaluar la calidad de las predicciones.

## Funciones Principales

### EPE (Error de Punto Final):

* Calcula la distancia euclidiana entre vectores de flujo predicho y objetivo en cada píxel.  
  Soporta datos dispersos (sparse), ignorando regiones inválidas en el flujo objetivo.  
  Devuelve un valor promedio o sumado según el parámetro mean.

### sparse\_max\_pool:

* Aplica un max pooling adaptativo a flujos dispersos.  
  Maneja de manera separada valores positivos y negativos para mantener la integridad de los datos.

### multiscaleEPE:

* Extiende la métrica EPE a múltiples escalas de resolución.  
  Combina los errores en varias resoluciones ponderando cada escala con pesos definidos (por defecto, valores tomados de literatura previa).  
  Evalúa cómo un modelo maneja la predicción de flujos en diferentes niveles de detalle.

### realEPE:

* Interpola las predicciones del modelo a la resolución del flujo objetivo.  
  Calcula el error EPE en la resolución original, útil para comparar directamente con los datos de referencia.

## Uso Típico

Este conjunto de funciones se emplea en tareas de entrenamiento y evaluación de modelos para problemas como:

* Estimación de flujo óptico.  
  Reconstrucción de desplazamientos en imágenes deformadas.  
  Medición de errores en predicciones a diferentes niveles de resolución (entrenamiento multiescala).

## Ventajas del Enfoque Multiescala

### Robustez en la Evaluación:

Considera el rendimiento del modelo en diferentes escalas, lo que permite identificar problemas específicos en la reconstrucción de detalles finos o patrones globales.

### Compatibilidad con Datos Complejos:

Puede manejar datos dispersos, lo que es útil en aplicaciones donde algunas regiones del flujo objetivo no están definidas.

### Flexibilidad:

Permite ajustar pesos de escala según las necesidades del modelo o el dominio de aplicación.