Asignatura	Datos del alumno	Fecha
Herramientas para la Computación en la Nube Dirigidas a Inteligencia Artificial	Apellidos: Ortega de Mues	01/29/2024
	Nombre: Mariano	

Actividad 1. Laboratorio: *Fine-tuning* de modelos

1. La función `torch.cuda.is_available()` permite establecer si existe soporte para GPU compatible con CUDA en la maquina donde se ejecuta el notebook, devolviendo true o false. En el notebook empleado si es true establece como device a emplear `cuda:0` que corresponde con el primer dispositivo GPU disponible. La función se complementa con la función `torch.cuda.device_count()` que devuelve el numero de total de dispositivos compatibles CUDA. En caso de emplearamos GPU como Compute type devolvería true. CUDA es el acrónimo de (Compute Unified Device Architecture). Es una plataforma que permite la ejecución en paralelo de cálculos generales no solo relacionados con gráficos empleando los llamados CUDA cores. Esta arquitectura es especialmente interesante para el entrenamiento de redes neuronales.

2. La variable le `images_population` es un diccionario con keys los nombres de las subcarpetas de la images (seran los nombres de las clases a reconocer) y el total de imagenes disponibles para cada clase:

   {'Cloudy': 300, 'Rain': 215, 'Shine': 253, 'Sunrise': 357, 'Multi-class Weather Dataset': 0}

   La variable `le` proporciona un diccionario con el encoding para las etiquetas de los tipos de imágenes, el encoding, necesario ya que el modelo empleado no admite variables categóricas, es:

   {'Cloudy': 0, 'Multi-class Weather Dataset': 1, 'Rain': 2, 'Shine': 3, 'Sunrise': 4}

Asignatura	Datos del alumno	Fecha
Herramientas para la Computación en la Nube Dirigidas a Inteligencia Artificial	Apellidos: Ortega de Mues Nombre: Mariano	01/29/2024

3. La clase WeatherDataset hereda de torch.tuils.Dataset (issubclass=True) y es empleado para definir la estructura de datos personalizada para el problema de clasificación que estamos modelando. El metodo __len()__ permite conocer la longitud del dataset. En nuestro caso la carga del data set se realiza ppor primera vez tras haber establecido el set de test en 20% para los datos de entrenamiento, motivo por el cual el tamaño del dataset de train es 900 (1125-(20%1125)).

4. Mediante el método Compose encadenamos dos trasformaciones los datos que representan cada una de nuestras imágenes:
   Transforms.Resize(): redimensionamos la imagen a 230x230.
   ToTensor(): se crea un tensor con los datos de las images, adicionalmente los datos de las imagenes son rescalados de 0-255 a [0.0,1.0] con precision float32.

5. Una vez los datos estan en el dataset, el dataloader permite inyectarlos en el modelo de forma sencilla, habilitando tambien transformaciones previas como la definicion de lotes o el empleo de multiples workers.

6. `RandomRotation()`:esta transformación nos permite rotar de forma aleatoria las imagenes en un rango de grados (en este caso –30, 30).
   `RandomResizedCrop()`: esta transformación muy util para simular distintas escalas de imagen y con ello la posición de objetos en la imagen, recorta de forma aleatoria una sección de la imagen y la redimensiona al tamaño establecido. En nuestro caso estamos empleando 230.
   `RandomFlip()`: esta transformación permite girar 180 grados una imagen de forma aleatoria (50% de probabilidad).

   Con estas transformaciones estamos tratando de crear un dataset de train con mayor diversidad (variabilidad) de forma que el modelo pueda generalizar mejor

y evitemos el overfiting. Estas transformaciones son parte de las herramientas de data augmentation. No las aplicamos a test o validación para no alterar la medición del rendimiento del modelo.

7. Con este parametro estamos indicando que comenzamos con el modelo ya entrenado. Estamos haciendo transfer learning ya que el modelo de Pytorch [2] esta entrenado previamente con el set ImageNet [1]. Con esto reducimos el tiempo y recursos para reentrenarlo con nuestro nuevo set de imagenes. El uso de pretrained esta deprecado, ahora se debe usar:

```
weights=DenseNet161_Weights.IMAGENET1K_V1
```

8. Para que se realice un checkpoint se deben producir dos condiciones:

1.- Se debe ejecutar el metodo train_model con el parámetro model_checkpoint!=0.
2.- La funcion de perdida calculada en la última validación de entrenamiento debe ser menor que la anterior.

9. Porque no es necesario calcular gradientes durante esta fase, así ahorramos uso de memoria y logramos un rendimiento mucho mayor en la evaluación del modelo.

10. Indican que para la clase 'Sunrise' el modelo clasifico correctamente 22 casos (VP=22) y hubo 2 casos incorrectos (FN=2).

Referencias:

Asignatura	Datos del alumno	Fecha
Herramientas para la Computación en la Nube Dirigidas a Inteligencia Artificial	Apellidos: Ortega de Mues	01/29/2024
	Nombre: Mariano	

[1] ImageNet Dataset: Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In 2009 IEEE Conference on Computer Vision and Pattern Recognition (pp. 248–255). IEEE. Retrieved from https://www.image-net.org/

[2] PyTorch Documentation - Pretrained models in torchvision: PyTorch Team. (n.d.). Torchvision Models Documentation. Retrieved from https://pytorch.org/vision/stable/models.html