|  |  |
| --- | --- |
| Unidad II | Actividad II – Transfer Learning - LORCA |
| Alumnos |  |

1. Detalles de la actividad.

* Leer el dataset de Skin Cancer y cargarlo como tensores.
* Realizar un Data Augmentation mediante ResizeCrop y HorizontalFlip.
* Usar el modelo predefinido y preentrenado de ResNet18 de PyTorch.
* Reemplazar la última capa por una capa Linear que tenga 2 clases a clasificar (benigno/maligno).
* Realizar un entrenamiento completo.
* Mostrar las gráficas de Loss y de Accuracy.
  1. Implementación del modelo.

Es importante tener el modelo implementado, previamente debemos descargar los pesos, para esta actividad en particular aplicaremos los conocimientos de Transfer Learning en “restnet18”, se revisó la documentación y existen más de una variante del modelo restNet, se optó por explorar una red con más capas. La extensión ***.pth*** es una extensión de archivo comúnmente utilizada en PyTorch para guardar los objetos serializados como los estados del modelo ***(state\_dict),*** optimizadores, configuraciones de entrenamiento y otros objetos. Este formato utiliza la biblioteca de serialización de ***PyTorch*** para guardar datos en un formato binario que puede ser cargado posteriormente en otro script o entorno de ***PyTorch***

A screen shot of a computer code

Description automatically generated*Imagen 01: Script que guarda los pesos en este caso restnet34.*

A screenshot of a computer program

Description automatically generated

A

B

*Imagen 02: Estructura inicial del modelo*

Importamos las librerías necesarias para nuestro modelo(A), posteriormente debemos redimensionar las imágenes que tendremos como entradas y convertirlos en un tensor. Posteriormente leemos nuestro ***dataset*** con las imágenes de ***test***, ***train*** y los parámetros necesarios como por ejemplo el ***batchsize***.

A computer screen with text and images

Description automatically generated

*Imagen 03: Carga de los pesos del Modelo restNet*

Debemos crea una instancia del modelo ResNet-34 en PyTorch sin cargar los pesos preentrenado y también precisar que el modelo se debe ejecutar en nuestra tarjeta gráfica.

A screen shot of a computer program

Description automatically generated

*Imagen 04: Número de épocas y cálculo de la función de perdida*

En este bloque de código entrenamos un modelo de red neuronal en PyTorch durante un número determinado de épocas. En cada época, se procesa el conjunto de datos de entrenamiento en minilotes, donde los datos de entrada y las etiquetas se transfieren a la GPU. Se realizan los siguientes pasos para cada minilote: reinicio de gradientes del optimizador, forward pass para obtener predicciones del modelo, cálculo de la pérdida, retropropagación para calcular los gradientes, y actualización de los pesos del modelo utilizando el optimizador.

Al final de cada época, se calculan y registran la pérdida y la precisión medias, las cuales se imprimen junto con el tiempo transcurrido. Este proceso ajusta iterativamente los pesos del modelo para mejorar su desempeño en la tarea específica del conjunto de datos de entrenamiento.

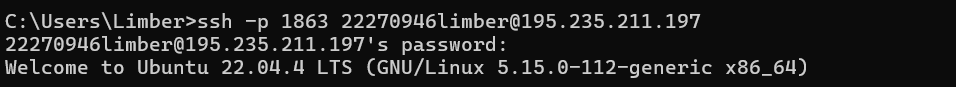
A computer screen shot of a program code

Description automatically generated

*Imagen 05: Resultado del entrenamiento.*

Ahora evalúamos el modelo de red neuronal utilizando el conjunto de datos de prueba después de cada época de entrenamiento. Se configura el modelo en modo de evaluación y se desactiva el cálculo de gradientes para optimizar el uso de memoria y velocidad. Luego, se itera sobre los minilotes del conjunto de prueba, realizando una pasada hacia adelante para obtener las predicciones del modelo y calcular la pérdida y la precisión. Las métricas de pérdida y precisión se acumulan y se calculan para todo el conjunto de prueba, almacenándose para su posterior análisis. Al final de cada época, se imprimen estas métricas junto con el tiempo transcurrido. Finalmente, se trazan gráficos de las pérdidas y precisiones de entrenamiento y prueba a lo largo de todas las épocas para visualizar el rendimiento del modelo.

1. Prueba del modelo en el clúster de LORCA
   1. Detalles.



*Imagen 06: Conexión al Clúster de Lorca.*

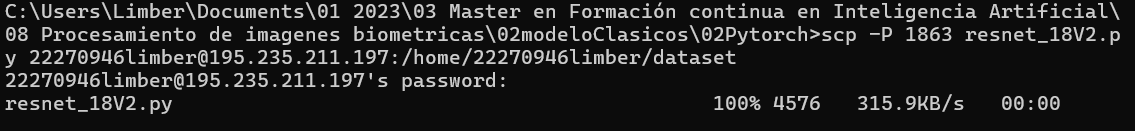
El nodo principal del Clúster es el único que tiene conexión a internet por consiguiente es importante conectarnos por ssh y transferir los archivos que se ejecutaron de manera local al Clúster.

A computer screen with white text

Description automatically generated

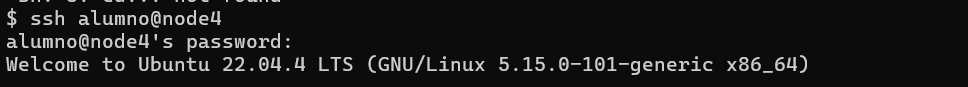
*Imagen 07: Pasar el data set al servidor remoto.*

Debemos pasar nuestro data ser al Nodo principal que es el único que tiene salida a internet para que posteriormente trabajemos con el nodo asignado a cada grupo



*Imagen 08: Copiar el script con el modelo.*

Finalmente copiamos el Script que contiene la lógica de nuestro modelo el cual ejecutaremos en el nodo asignado. Estos archivos inicialmente están en el nodo principal, estos pasos debemos repetirlo y tener todo nuestro entorno completo.



*Imagen 09: Conexión al node4.*

Nuestro grupo tiene asignado el nodo 4, nos conectamos a ello por ssh y preparar el entorno necesario.

A computer screen shot of a black screen

Description automatically generated

*Imagen 10: Pasar al node4 el data set.*

Pasamos el data set al node4, como se comentó al inicio de esta actividad este nodo no tiene salida a internet, por ello debemos pasar los ficheros necesarios para ejecutar el modelo.

A screenshot of a computer program

Description automatically generated

*Imagen 11: Descomprimir el dataset.*

Como ultimo paso descomprimimos nuestro conjunto de datos y luego configurar el Path de nuestro Script con la nueva ruta de nuestro Cluster.

A screenshot of a computer program

Description automatically generated

*Imagen 12: Entrenamiento del modelo en el Cluster.*

Es importante cambiar ciertos parámetros que se tiene en el Script inicial ya que la ejecución estaba en local, en la imagen 12 lo tenemos ejecutando en el nodo asignado al grupo.

1. Conclusiones.