**PROYECTO:**

Implementación mediante Streamlit (inicialmente, posteriormente se hará una página más atractiva) de una página web donde el usuario pueda subir una imagen y mediante un modelo que entrenaremos esta será clasificada en una de las siguientes clases.

clases = ["Bedroom", "Coast", "Forest", "Highway", "Industrial",

          "Inside city", "Kitchen", "Living room", "Mountain", "Office",

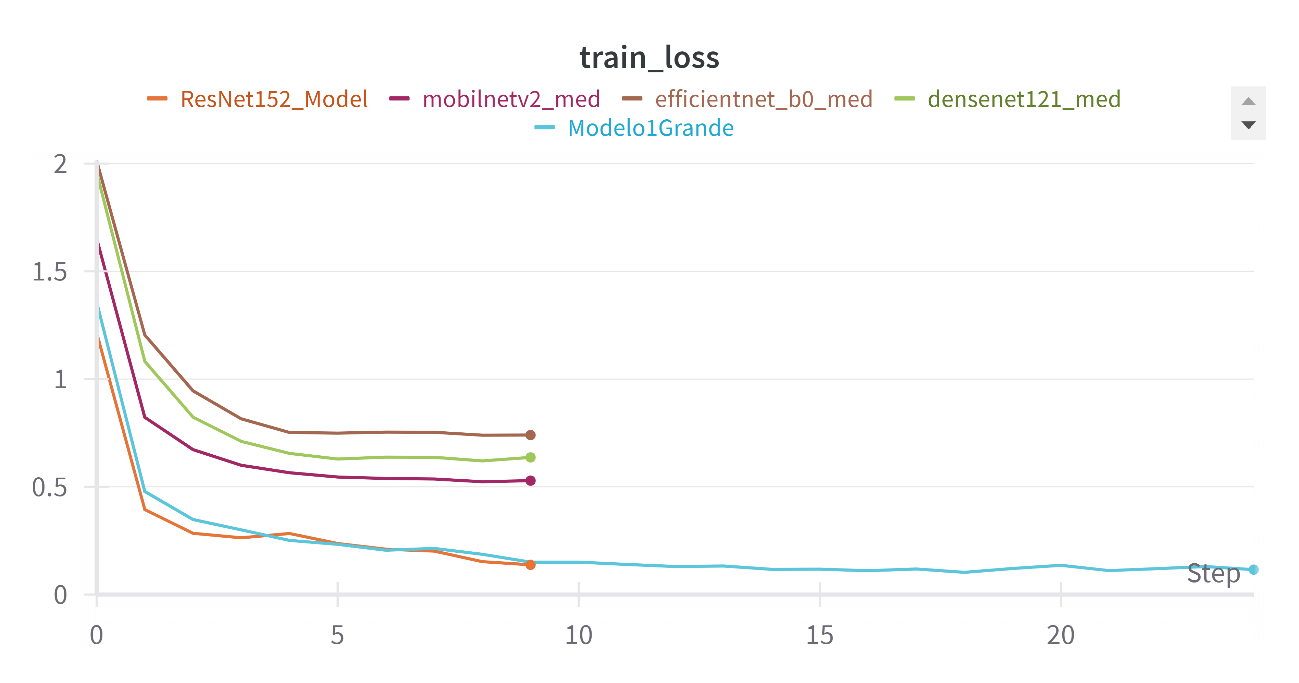
          "Open country", "Store", "Street", "Suburb", "Tall building"]

Partimos de una gran cantidad de modelos preentrenados:  
  
['alexnet', 'convnext\_base', 'convnext\_large', 'convnext\_small', 'convnext\_tiny', 'densenet121', 'densenet161', 'densenet169', 'densenet201', 'efficientnet\_b0', …]

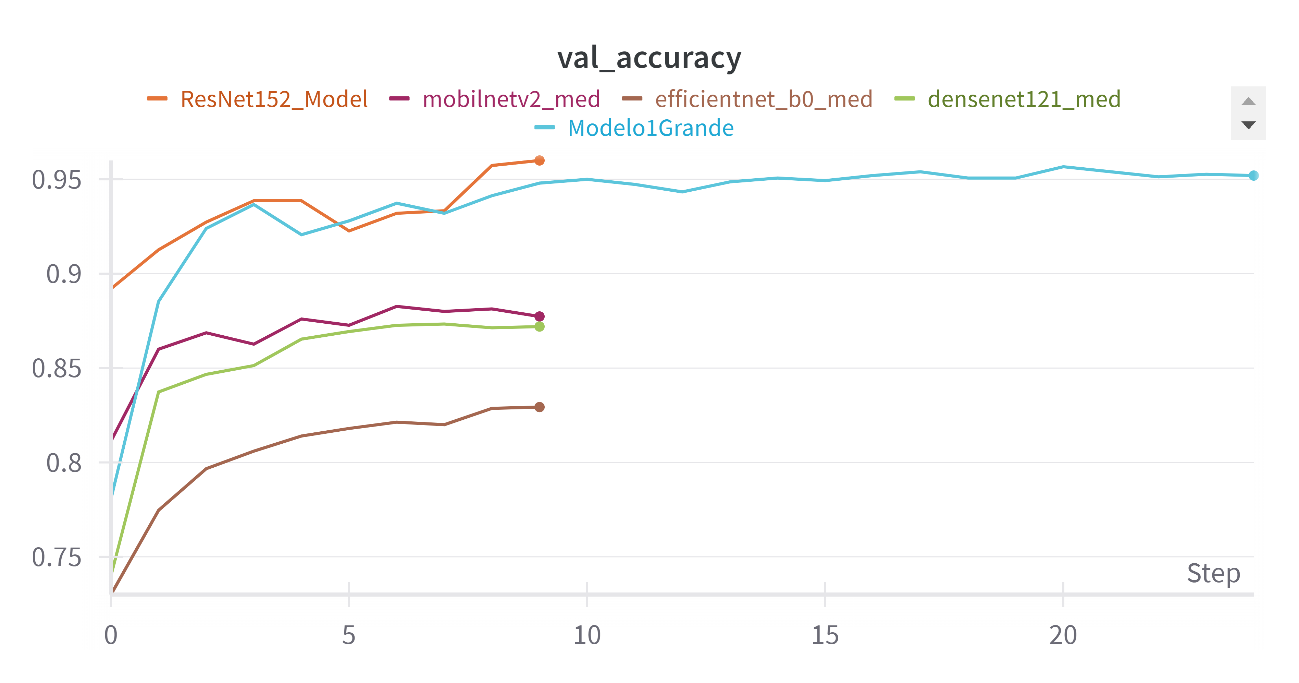
Tras hacer unas primeras comprobaciones y entrenar modelos a partir de unos ya entrenados de las siguientes familias:

ResNet, DenseNet, EfficientNet, ShuffleNet, MobilNet, EfficientNet omprobamos que la familia de modelos que mejor desempeño nos ofrece es ResnNet.

Estos son los resultados encontrados y que hemos registrado en Weight and Biases, os he dado acceso para posibles comprobaciones. <https://wandb.ai/sergom2/ML2-img-class>



Gráfico, Gráfico de líneas

Descripción generada automáticamente

Vemos como modelos como efficientnet no parecen ser los mejores para este propósito, ResNet tiene gran desempeño, algo que era de esperar debido a su complejidad.

Tras estas comprobaciones vamos a dedicar nuestro esfuerzo computacional en entrenar un modelo a partir del ResNet50 con 25 epochs y 10 capas descongeladas, además de algunas transformaciones que haremos a las imágenes.

Todo el código está en el repositorio <https://github.com/sergom2/Practicas_DeepLearning_2024> más concretamente en <https://github.com/sergom2/Practicas_DeepLearning_2024/tree/main/streamlit/session1> al cual tenéis acceso.

Texto

Descripción generada automáticamente

El código de entrenamiento se encuentra en *entrenamiento.ipynb*.

Tras más de 10 horas de entrenamiento conseguimos un modelo que parece bastante preciso.



Pusimos en marcha este modelo en la página de stremalit, esta se puede probar con el archivo streaminicial.py usando streamlit run en la consola.

Tras realizar algunas pruebas nos damos cuenta de que cuando otorga probabilidad a una clase cercana a 1 suele acertar, pero cuando duda (otorga probabilidad a la clase menos a 0.9) se equivoca en ocasiones.

Para solventar esto decidimos entrenar otro modelo a partir de una ResNet más compleja, optamos por la ResNet152, esto nos permitirá conocer características más concretas.

Texto

Descripción generada automáticamente

Usamos el mismo learning\_rate y número de capas descongeladas que anteriormente, sin embargo reducimos el número de epochs debido a la complejidad del modelo.

El código del entrenamiento del modelo se encuentra en *modelosgrandes.ipynb.*

Las transformaciones que haremos a las imágenes siguen siendo las mismas.



Vemos como este modelo también debería tener una gran capacidad de clasificación, pero haciendo pruebas nos damos cuenta de que en ocasiones no clasifica bien, sin embargo, en aquellas ocasiones en que el modelo entrenado a partir de un ResNet50 duda, este modelo consigue clasificar bien las imágenes debido a su mayor grado de complejidad y detectar patrones más complicados.

**SOLUCIONES:**

Visto esto nos decantamos por dar 3 posibles soluciones:

* Despliegue de página con predicción mediante el modelo entrenado con ResNet50. *streaminicial.py*
* Despliegue de página con posibilidad de elección de el modelo que queremos usar, hay 3 de ejemplo pero se podrían poner todos los necesarios. *streammulti.py*
* La que creemos mejor, despliegue de página que utiliza por defecto el modelo entrenado con ResNet50 y en caso que este “dude” en sus predicciones aplicación del modelo entrenado usando el ResNet152 para asegurar la predicción. *streamfinal.py*

Vamos a mostrar como funciona la solución que nosotros consideramos como óptima:

Página desplegada:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

En caso de elegir la opción en la que se puede elegir el modelo se verá así y el desplegable nos permitirá elegir el modelo a usar:

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Para la solución que nosotros hemos considerado la óptima, usando el modelo entrenado con Resnet50 y en caso de duda el modelo entrenado con ResNet152, siempre indicará la probabilidad que le da a cada clase y el modelo usado para ello:

En caso de que no dude el modelo entrenado con ResNet50:

Cocina con estantes blancos

Descripción generada automáticamente

En caso de duda del modelo entrenado con ResNet50:

Pantalla de computadora encendida sobre un escritorio

Descripción generada automáticamente

Cabe destacar que si el modelo entrenado con ResNet152 obtiene en las dos primeras posiciones de clases con más probabilidad, las mismas que el modelo entrenado con ResNet50, se mostrarán las probabilidades del modelo entrenado a partir del ResNet50.

**FUTUROS PASOS:**

En un futuro, tras la aprobación del proyecto:

* Se considerará entrenar de nuevo modelos a partir de ResNet reduciendo el leraning\_rate y con un número mayor de capas descongeladas, quizás se añadan más epochs, aunque no se espera que un aumento de estas últimas de una mejora considerable.
* Se considerará aumentar la cantidad de imágenes que se usan para el entrenamiento, preferiblemente de manera natural, si no fuera posible se hará de manera artificial mediante más transformaciones.
* También se desplegará esta aplicación en una nueva página, mucho más cuidada y llamativa y con la identidad de marca del cliente.
* Es posible que se añada la posibilidad de que el propio usuario de feed-back de las predicciones para futuras mejoras de los modelos.