UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN

Informe: Tarea #3

Informe sobre Transformers y Stacked AutoEncoders
Universidad Central de Venezuela por el
Estudiante. Rafael Eduardo Contreras.

Profesor: Fernando Crema

Caracas, jul 15 de 2025

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN

ÍNDICE GENERAL

Dataset

BRISC 2025 - Link

BRISC es un conjunto de datos de resonancia magnética de alta calidad, anotado por expertos, diseñado para la segmentación y clasificación de tumores cerebrales. En esta tarea solo utilizamos la parte de clasificación para tumores.

El dataset de clasificación contiene imágenes en distintos cortes, estos vienen siendo cortes axial, coronal y sagital. La tarea encomendada es crear distintos modelos de clasificación para los tipos de tumores presentes, siendo este glioma, meningioma, pituitaria y no tumor.

Dataloader

Hemos creado un Dataloader para poder manejar y transformar los datos de forma sencilla y eficiente. Hacemos distintas transformaciones a las imágenes al usarlas, entre ellas: un resize para que nuestros modelos no sean tan grandes, transformación a tensor y normalización. Además este objeto carga todas las imágenes a memoria para acelerar el proceso de entrenamiento.

Modelos

ViT custom

El primer modelo trabajado con el dataset es el Visual Transformer hecho en el notebook de esta tarea. Este modelo consta de varias clases:

- Residual y PreNorm: La primera es una conexión residual dada una función pasada como argumento, ejecuta y = f(x) + x. La segunda normaliza los valores y luego ejecuta una función pasada, ejecuta f(norm(x)).
- FeedForward: Una red que dadas la dimensión de entrada y salida ejecuta una capa lineal, la función GELU y la capa lineal de salida.
- Attention: La capa de atención. Esta divide al input en tres matrices usando una red lineal cuya salida es 3 veces la dimensión de entrada. Luego separa el output en tres matrices Q, K y V. A estas se les aplica el proceso explicado en el anexo A. Hay que tomar en cuenta que este modelo ejecuta tantos procesos de atención como heads se especifiquen en los argumentos. Finalmente, esta capa concatena las cabezas y ejecuta la proyección final, esto consiste en una capa lineal con la misma dimensión de salida que de entrada.
- Transformer: Este modelo ejecuta varias capas (pueden ser especificadas) de lo mostrado en el anexo B.
- ViT: Finalmente tenemos la clase que representa el modelo final. Aqui agregamos algunas definiciones que consideramos pertinentes para entenderlo:
    - Patch: Un patch es un pequeño fragmento de una imagen. En lugar de procesar la imagen completa de una vez, el ViT la divide en una cuadrícula de estos parches.
    - Embedding: Cada uno de estos parches se "aplana" en un vector largo y plano (canales incluidos).

    - CLS: Token de Clasificación. El CLS es un token especial que se añade al principio de la secuencia de parches de la imagen. Por cada imagen existe 1 CLS que es la representación final de la imagen para luego usarse en la clasificación.

  El proceso de inferencia del ViT es mas complejo que el de los módulos anteriores. Primero hacemos las transformaciones respectivas a la imagen para obtener sus patches, a estos los convertimos en embeddings con las dimensiones adecuadas usando una capa lineal.

  Luego de esto, agregamos los tokens CLS a cada imagen en el batch. Este input pasa por la capa de transformer y al obtener el resultado separamos la imagen del token CLS para ser usado en la capa de clasificación.

Este modelo es entrenado usando el dataset descrito arriba, no hacemos énfasis en diferenciar los distintos cortes de las imágenes, solo nos enfocamos en la tarea de clasificación.

Huggingface ViT

Para el ViT de huggingface usamos paquetes para descargar el modelo preentrenado. Este es el google/vit-base-patch16-224. El 16-224 se refiere al tamaño del patch (16x16) y la resolución de las imágenes para él fine tuning (224x224). Aquí no tenemos acceso a la arquitectura específica del modelo, así que asumimos es bastante similar a la explicada en el ViT de nosotros.

En esta ocasión entrenamos el modelo descongelando los gradientes. Debido a que sin descongelar los mismos el modelo tiene un bajo performance, decidimos descongelarlos. Gracias a esto, los resultados de clasificación luego del entrenamiento sobrepasaron los de los modelos anteriores.

Stacked Autoencoder

Para el stacked autoencoder el modelo es más simple que el de ViT, aunque el modelo es mas grande y tarda más en entrenarse. Hice varios cambios al inicio, relacionado a las dimensiones de las capas y las dimensiones iniciales de las imágenes, esto para reducir el tamaño y facilitar el entrenamiento. Finalmente usamos las computadoras ofrecidas por la escuela para poder entrenar el modelo sin dificultad.

El stacked autoencoder viene definido por dos módulos (clases) principales:

- AutoEncoderLayer: Este modulo es bastante simple, solo es necesario especificar 3 cosas, las dimensiones de entrada/salida y si estamos en proceso de entrenamiento. Cada una de estas capas es entrenada individualmente, su objetivo es aprender de manera no supervisada usando la entrada como referencia.
    - Durante el proceso de inferencia el input solo pasa a traves del encoder. Este es una capa lineal con una función ReLU.
    - Durante el proceso de aprendizaje la salida del encoder pasa por el decoder. La perdida viene dada por la diferencia que existe entre la entrada de la capa actual y la salida del decoder.

    - Este proceso de entrenamiento se repite en cada una de las capas. Congelando los pesos de las capas anteriores.
- StackedAutoEncoderClassifier: Este módulo utiliza las capas de encoders anteriores y usa la salida final de menor dimensión para clasificar la imagen usando una capa de clasificación lineal. El proceso de entrenamiento es el mismo que el de un clasificador estándar, solo que los pesos de los encoders estarán congelados.

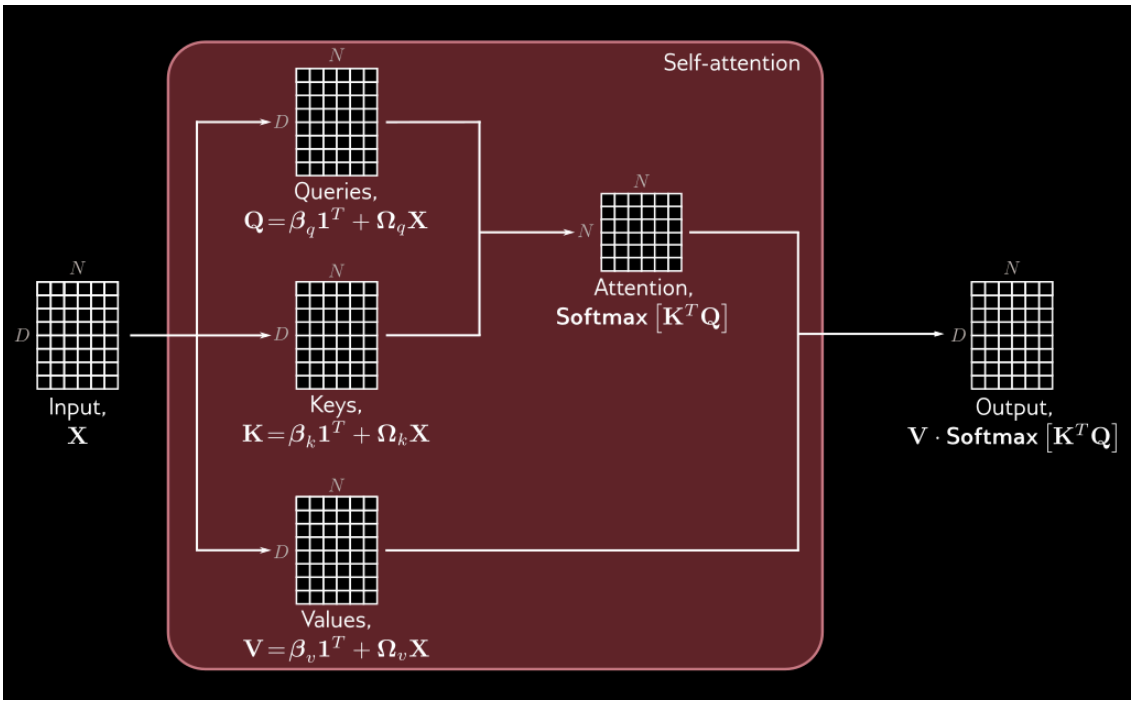
Resultados

Sobre la precisión con el dataset de prueba durante el entrenamiento. El mejor modelo fue el que fue sacado de huggingface, nuestro modelo se aproxima en estabilidad y precisión durante el entrenamiento. Hay momentos en el entrenamiento donde el Stacked autoencoder tiene mejor precisión que nuestro modelo; sin embargo, su precisión varía mucho y no parece estabilizarse durante las épocas. Más información en el anexo C.

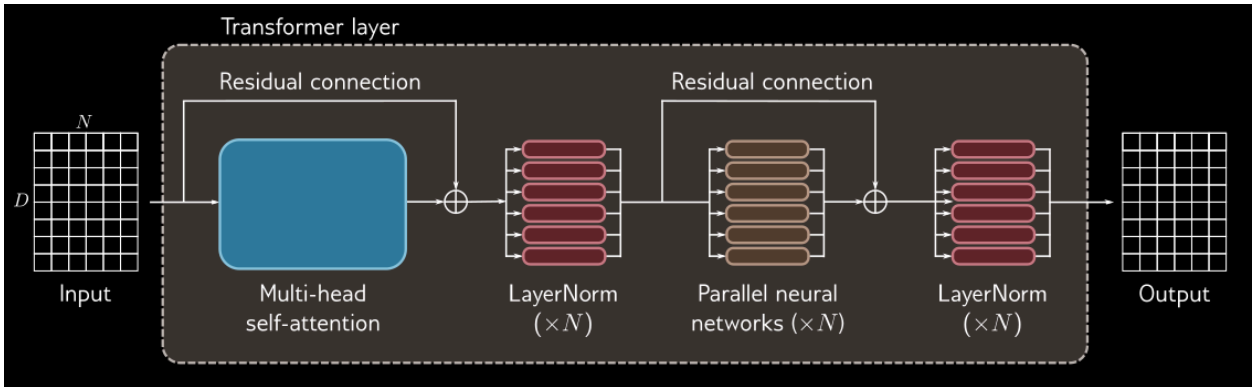
Sobre la perdida durante el entrenamiento. El modelo que llega a una menor perdida más rápido (en menos épocas) también es el de huggingface. El SA(stacked autoencoder) le sigue y luego nuestro modelo. La gráfica muestra que la perdida del SA también se vuelve menos suave en las últimas épocas, al igual que con la precisión. Para más información ver el anexo D.

Sobre las gráficas custom, decidimos hacer curvas ROC para nuestro ViT. Seguimos los ejemplos dados en la documentación de wandb para esto. Los resultados en general reflejan el accuracy final de todos los modelos. El mejor modelo es el de huggingface, seguido por nuestro ViT casero y finalizando con el stacked autoencoder.

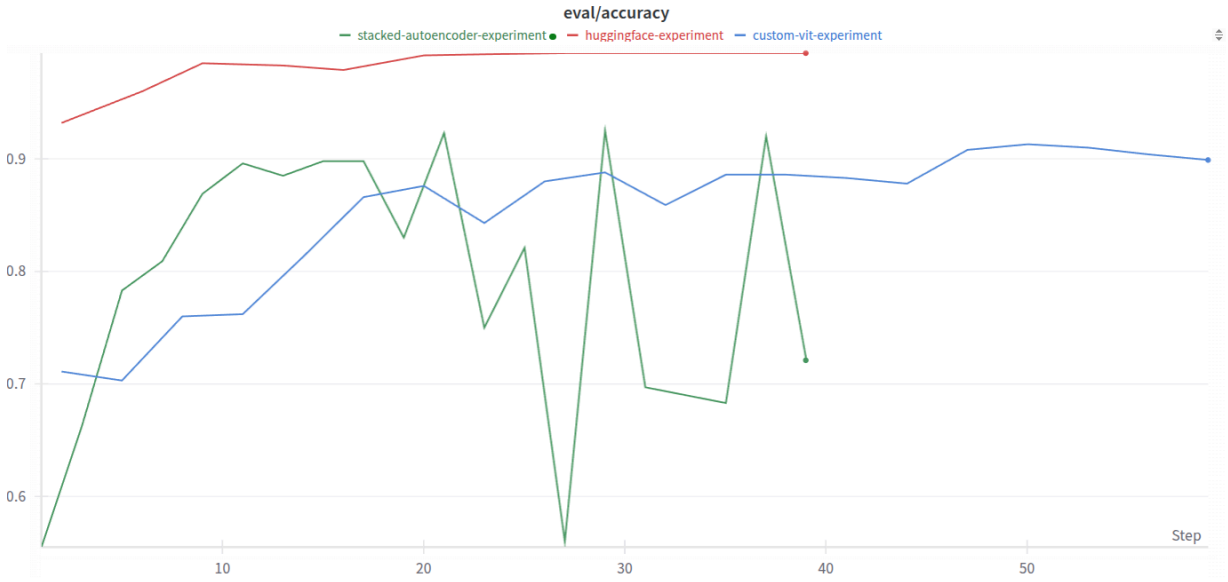
ANEXOS



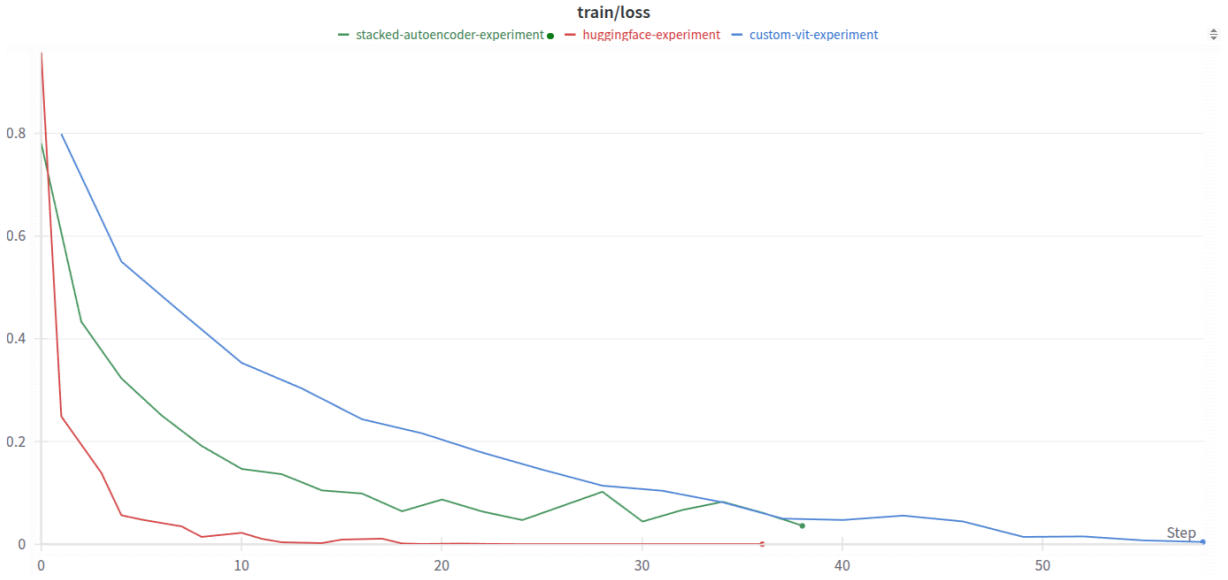
Anexo A: Self-Attention



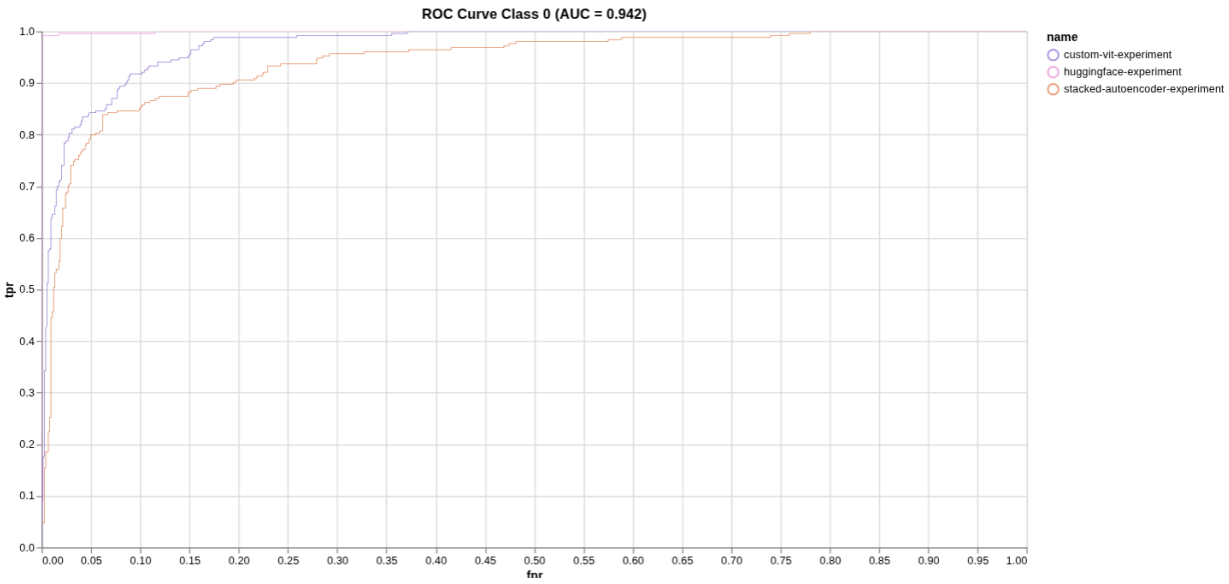
Anexo B: Capa Transformer



Anexo C: evaluación de precisión durante entrenamiento



Anexo D: perdida durante entrenamiento



Anexo E: Grafico custom, curva ROC para la clase 0.