11 DE JUNIO DE 2021

RECONOCIMIENTO DE TEXTO EN ESCENA

APRENDIZAJE PROFUNDO

EDGAR GONZALEZ PAZ
FACULTAD DE ESTUDIOS SUPERIORES ACATLÁN
[Dirección de la compañía]

Introducción.

El siguiente reporte está basado en el artículo "Reconocimiento de texto en imágenes por medio de Redes Neuronales Convolucionales Recurrentes", de Baoguang Shi, Xiang Bai, Cong Yao.

Los retos en la visión computacional constituyen una oportunidad grande para proponer arquitecturas de redes neuronales que logren detectar los caracteres en la imagen y producir una secuencia de posibles letras mostradas en la imagen. Así pues, esta tarea difiere de la detección de objetos en la que estamos interesados en detectar categorías de objetos pues el problema de reconocer texto en imágenes implica una variación en la longitud de las secuencias.

Para lograr cumplir el objetivo, se han presentado varios enfoques como:

1. Detección y reconocimiento de caracteres empleando Redes Neuronales Profundas Convolucionales (DCNN). Desventaja: Construir un clasificador robusto de detección.

2. Reconocimiento por palabra, planteado como un problema de clasificación. Desventaja: No escalable a algunos idiomas por tratar múltiples clases.

El enfoque propuesto por Baoguang Shi, Xiang Bai y Cong Yao es una arquitectura que combina las redes neuronales Convolucionales y Recurrentes, toma ventajas de ambas, por mencionar algunas: aprende representaciones informativas sin necesidad binarización/segmentación/componente de localización, y de producir directamente etiquetas de secuencias y por lo tanto, tener un número menor de parámetros para el modelo.

Arquitectura.

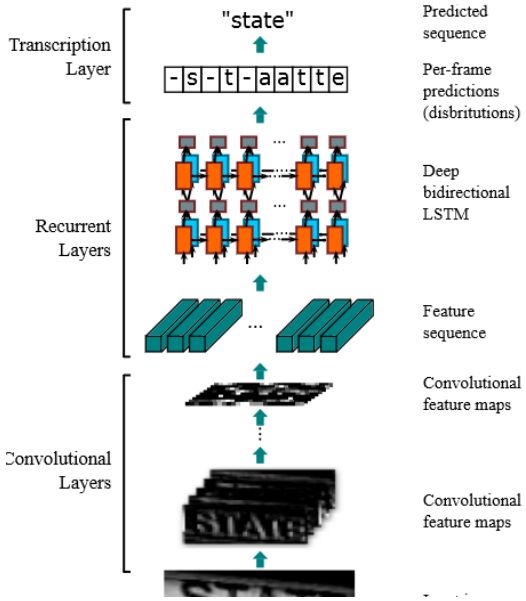

Figura 1. Arquitectura "Convolutional Recurrent Neural Network"

El esquema mostrado nos ayuda visualmente a explicar la arquitectura de esta red neuronal, en primer lugar, tenemos redes convolucionales (se desecha la parte completamente conectada y se aprovecha el max-pooling), su principal tarea es extraer secuencias de características de representación de la imagen que servirán para generar vectores de características secuenciales, mismas que sirven de entrada para la red recurrente bidireccional.

Antes de describir la siguiente capa, tanto en el artículo como en la implementación se hace uso de redes residuales, con el propósito de evitar el problema de gradiente desvaneciente. Además, se sabe que las redes ResNet tienen menores filtros y menor complejidad que las redes VGG. Por otra parte, es una buena propuesta comenzar por un modelo de transfer

Learning que ayuda al modelo con las características que ya aprendió previamente en otro conjunto de datos.

La siguiente capa tiene la tarea de predecir una etiqueta para cada frame en el vector de características. La principal ventaja es que puede retropropagar los errores de la entrada, logrando que se pueda entrenar junto con la capa convolucional en una red unificada. El problema del gradiente desvaneciente se ataca con el uso de LSTM, capturan dependencias a largo plazo, aún más si se combina/apila con otra que capture el contexto de la secuencia de la imagen en dirección contraria.

La transcripción de la predicción realizada por frame a una secuencia está condicionada a las predicciones de los demás frames y es realizada por medio de una transcripción basada en un léxico, el cual es aplicado en la presente práctica. Como función de pérdida para el entrenamiento de la red se utiliza el negativo de la log-verosimilitud de la probabilidad de cada etiqueta de secuencia condicionada a las predicciones de los demás frames, esta compara las verdaderas etiquetas de secuencia contra la secuencia producida por las redes recurrentes y convolucionales.

Reproducción de resultados del artículo

En el siguiente link, podemos encontrar el repositorio que contiene el código base que implementa una arquitectura Convolutional Recurrent Neural Network (CRNN). Además, se probará superar el performance del modelo presentado, modificando una parte de la capa convolucional y el tipo de optimizador, así como experimentar con distintos hiper parámetros. Como framework, se utilizará Pytorch, comenzando por un modelo pre entrenado conocido como res-net18, posteriormente se pasa a una capa convolucional, para pasar por dos unidades recurrentes con puertas bidireccionales y ser pasada al último por una capa de transcripción. El entrenamiento es realizado por medio de lotes de tamaño 8, en un total de 50 épocas. La función de pérdida es conocida como Clasificación temporal conexionista, el optimizador utilizado es el Descenso de Gradiente Estocástico.

Descripción del conjunto de datos

Contiene imágenes en un mismo formato y tamaño, fondo blanco con texto negro, son una combinación de números y letras en minúsculas. Existen perturbaciones en las imágenes, tales como: escritura no uniforme, líneas que atraviesan por completa o parcialmente el texto, distorsión en algunas zonas. Esto se traduce como un reto para el modelo para identificar patrones con ruido. El conjunto de entrenamiento tiene un total de 856 imágenes y el conjunto de validación 214 imágenes.

Figura 2. Muestra del dataset

Resultados

La arquitectura e hiperparámetros propuesta por el autor fueron modificadas para obtener mejores resultados, en primer lugar, la modificación propuesta para la arquitectura se dio principalmente en dos componentes: el modelo pre entrenado para la capa convolucional y el optimizador para la función de pérdida (Tabla 1).

Componente	Baseline	Propuesta
Modelo pre-entrenado	res_net18	res_net34
Optimizador	SGD	AdaDelta

Tabla 1. Modificaciones en la arquitectura

Además, también se propone probar con distintos valores de los hiper parámetros: la tasa de drop out y de aprendizaje (Tabla 2). Los experimentos derivados de la combinación de cambio de arquitectura con cambio en hiper parámetros dio un total de 36 experimentos. Para poder colectar aquellos experimentos con mejor desempeño, podríamos tomar en cuenta (como lo hace el autor del código) sólo la métrica de precisión (número de predicciones de texto acertadas entre el total de muestras), sin embargo, como extra se propone añadir el índice de Jaccard ¿por qué? La justificación es que en aquellas predicciones donde no coincide al 100% carácter por carácter y falla en sólo algunas palabras también debe ser tomado en cuenta. Es decir, no podemos pasar por alto que el modelo predice correctamente un porcentaje de los caracteres dentro de la imagen y descartar cuando erra en uno o más. El índice de Jaccard, utilizando dos listas (en este caso, etiquetas y predicciones) mide el grado de similitud entre estos dos conjuntos.

Hiper parámetro	Baseline	Propuesta 1	Propuesta 2
Tasa Drop out	0.1	0.5	0.75
Tasa de aprendizaje	0.02	0.05	0.09

Tabla 2. Hiper parámetros propuestos

De los más de 30 experimentos resultantes, se tomaron aquellos con mejores resultados (Tabla 1) logrados en la métrica de precisión para el conjunto de validación y luego sobre la métrica del índice. Así pues, se considera el modelo baseline para poder realizar un análisis de resultados.

Nombre	Modelo pre-entrenado	Optimizador	Tasa de aprendizaje	Valor Drop out	Precisión (entrenamiento	Precisión (validación)	Indice Jaccard (entrenamiento	Indice Jaccard (validación)	Tiempo de ejecución (s)
Experimento 1	res_net34	AdaDelta	0.09	0.1	98.36	88.32	96.78	79.08	273.89
Experimento 2	res_net34	AdaDelta	0.09	0.5	98.36	87.85	96.78	78.33	276.53
Experimento 3	res_net18	SGD	0.09	0.1	89.37	85.05	80.78	73.98	180.36
Experimento 4	res_net18	AdaDelta	0.09	0.5	96.73	84.58	93.67	73.28	205.02
Experimento 5	res_net18	AdaDelta	0.09	0.75	96.61	84.58	93.45	73.28	205.26
Baseline	res_net18	SGD	0.02	0.1	92.76	79.91	86.49	66.54	180.42

Tabla 3. Resultados experimentos y modelo baseline

*Función de pérdida

Las gráficas siguientes demuestran el promedio de la función de pérdida por cada batch, difieren en el eje de las x por los diferentes tamaños entre el conjunto de entrenamiento y validación. Se optó por presentar el contraste del experimento 1 versus el modelo baseline, tiene un comportamiento parecido en la manera que cae la función de pérdida abruptamente al inicio, se detiene un poco y se repite una caída igual de prolongada para después mantenerse oscilando, es aquí donde el tamaño de oscilaciones marca la diferencia, mientras que el baseline tiene un rango mayor de movimiento, en el experimento 1 al final se mantiene entre 0 y 1.2 aprox., en contraste con el propuesto por el autor del repositorio que está entre 0 y 1.5. Como conclusión, podemos decir que un incremento en el número de épocas probablemente no repercuta más en el performance de nuestros experimentos, es decir, incluso podemos obtener parámetros que de una época a otra afecten las predicciones.

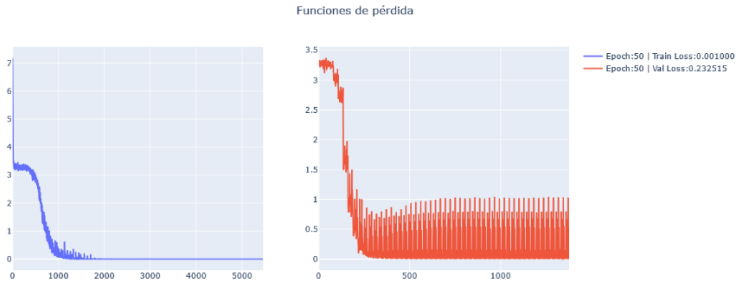

Figura 3. Función de pérdida en cada época (Experimento 1)

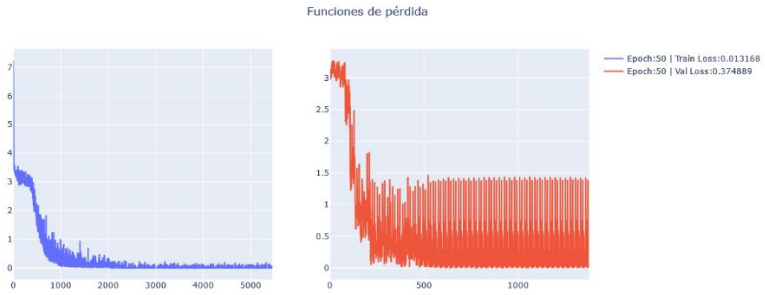

Figura 4. Función de pérdida en cada época (modelo baseline)

-Análisis sobre métricas de evaluación

El siguiente gráfico contiene los resultados resumidos de los mejores experimentos y el modelo baseline, el modelo con mejores métricas reportadas fue aquel que emplea el modelo res-net34 con un Optimizador AdaDelta (como observación, en los experimentos realizados en el artículo, mencionan que converge más rápido que el SGD, comparación que no podemos apreciar tan bien en la tabla porque la tasa de aprendizaje en el modelo pre entrenado y la tasa de drop-out son distintas aún con la misma tasa de aprendizaje). Visualmente, podemos apreciar que el modelo baseline fue superado por un poco menos de 10 puntos porcentuales del mejor). Como observación, podemos apreciar un poco de varianza en entre los resultados del modelo de entrenamiento y de validación, menos en el experimento 3, el cual se “cuela” entre los mejores modelos a pesar de que tiene las peores métricas de todos excepto en la precisión en el dataset de validación. El índice de Jaccard y la precisión aumentan de manera simultánea, sin embargo, esta métrica no se logró aumentar lo suficiente, las predicciones del experimento 1 las considero “de fiar”, pensemos en que dada una imagen generamos una predicción, sabemos que existirán algunas palabras que en el peor de los casos ignore o algunas otras falle. Se podría tener un diccionario de palabras recopilado por la experiencia del usuario y por medio de una métrica como “edit distance”, tratar de encontrar por medio de comparaciones a aquellas dentro del diccionario con un umbral definido y pueda ser al menos un poco más confiable nuestra predicción.

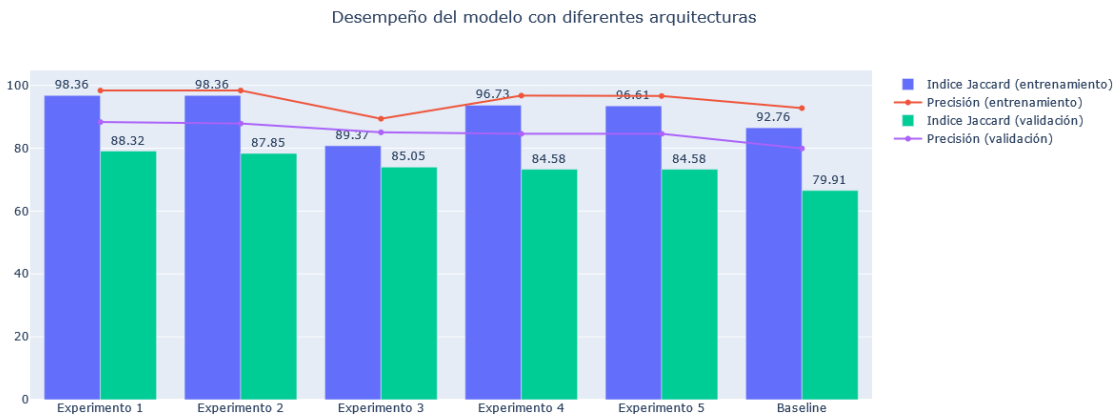

Figura 5. Desempeño de modelos

-Comparación de predicciones

En general, podemos concluir que los modelos logran reconocer los patrones generados por los caracteres en el dataset, ejemplo de ello es la impecable detección de una combinación de letras y números:

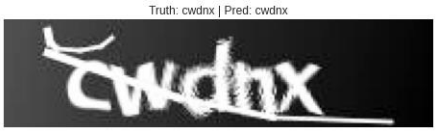

Figura 6. Predicción para imágen con etiqueta cwdnx

Figura 7. Ejemplo de etiqueta con mala predicción

En una comparación de resultados aleatoria entre los distintos modelos, se pudo lograr detectar que, en particular para este conjunto de datos, a los modelos les cuesta dar una buena predicción cuando en la imagen muestra encontramos caracteres repetidos seguidos y con alguna perturbación, ejemplo claro el siguiente en el cual ningún modelo pudo acertar.

Y para una comparación entre el mejor modelo y el baseline, se ilustra con las siguientes imágenes, donde se confirma que los modelos erran donde se presentan las condiciones enunciadas en el párrafo anterior. Por último, es también válido mencionar que está técnica conocida como data augmentation es realizada para que incremente nuestro dataset, como observación, en este dataset no se incluye las imágenes sin perturbaciones.

Figura 8. Experimento 1, predicción para gp22x

Figura 9. Modelo baseline, predicción para gp22x

La notebook de trabajo puede consultarse en el siguiente link.