Nicolas Cari Rodriguez

SistemasInteligentes-CNN-RNN
Laboratorio

UCB - Cochabamba, 07 de Mayo del 2023

1. ConvNets

- ¿Cuál es la entrada y la salida del siguiente código?

Train_Y_one_hot = to_categorical(train Y)

Esto transforma las labels de nuestro código en hot encoding, en el archivo de código tenemos el siguiente ejemplo:
Original label: 9
After conversion to one-hot: [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]

Donde tenemos 9 labels y la novena es 1, las otras labels serán 0 para que sepamos en hot encoding a cual label nos estamos refiriendo.

- ¿Cuál es el rol de cada uno de estos parámetros?

  fashion_model.add(Conv2D(32, kernel_size=(3, 3), activation='linear', input_shape=(28,28,1), padding='same'))

  - Conv2D = capa de convolucion utilizada para procesar imágenes 2D.
  - 32 son la cantidad de filtros que tenemos.
  - kernel_size será nuestro tamaño del kernel que iremos recorriendo la imagen.
  - la activación linear es la activación por defecto donde la salida será la misma que la entrada, sin cambiar los valores en ella.
  - Input_shape es el tamaño de la imagen, es de 28x28 bits.
  - Padding=same hace que la salida y la entrada sean de la misma forma y no distintas.

- ¿Cuál es la función de la capa Conv2D, MaxPooling2D y Dense? ¿Qué hacen?

La capa conv2D es la capa de convolución, extrae características de la imagen con filtros que hace que destaquen las características mas importantes.

La capa MaxPooling2D reduce la dimensión creada por la capa conv2D, para que sea más fácil entrenar bajando la resolución de la imagen.

La capa Dense se utiliza para clasificar la imagen en distintas categorías, cada nodo de esta capa esta conectada a todos los nodos de la capa anterior, se utiliza después de hacer las capas de convolución antes de empezar la clasificación.

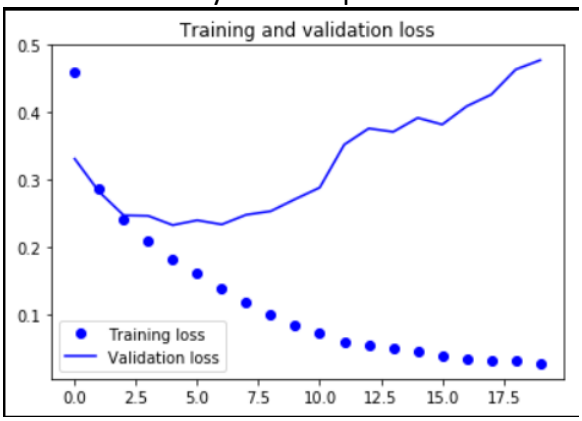
- ¿Cuál es la función de la capa LeakyReLU y Dropout en el segundo modelo? ¿Qué hacen?

LeakyRelU funciona como RelU pero los valores negativos que tienen salida cero cambian con una pendiente, así tienen un flujo de gradiente constante y no todos tienen el mismo valor.

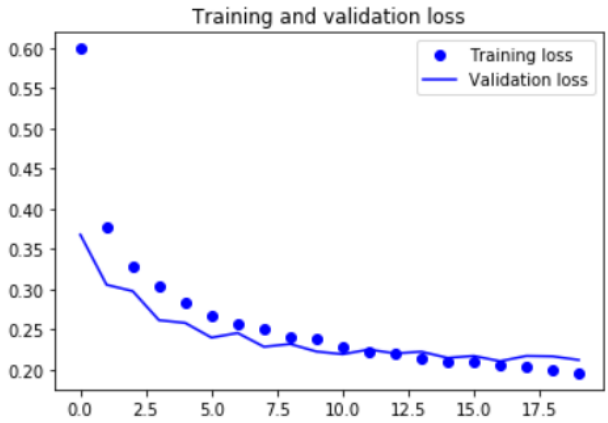
Dropout se utiliza para prevenir el sobreajuste del modelo. Le damos un porcentaje de las neuronas y las apaga con valor 0, así el modelo tiene que aprender patrones y que no dependa de una neurona todo el modelo.

- ¿Qué modelo está haciendo overfitting?

El primer modelo es que esta haciendo overfitting, podemos ver que mientras vamos entrenando nuestro accuracy no sube pero nuestro loss baja.



Mientras tanto en nuestro segundo modelo podemos ver que el loss sigue bajando:



Esto claramente significa que nuestro primer modelo esta sobreentrenado y nuestro segundo modelo no, porque el loss no aumenta como el primero.

2. Recurrent Neural Networks

- ¿Qué hace el proceso de tokenizar el corpus?

Lo que hace es convertir una secuencia de texto en secuencia de números. Por lo que en el código: tokenizer = Tokenizer(), tokenizer.fit_on_texts(corpus) agarra cada palabra y le da un identificador, así las palabras son números únicos y es mas fácil entrenarlo.

- ¿Para qué se añade un padding a la izquierda?

Las palabras no son todas del mismo tamaño, pero necesitamos una longitud misma de entrada para entrenarlo, entonces a las palabras mas cortas se les añade un padding a la izquierda para que sean del mismo tamaño que las palabras largas, para que no rompan su valor se les agrega 0 y seguirán siendo lo mismo: ej: 1355 a 001355

- ¿Qué significa este array después de tokenizar? ¿Por qué se repiten varias veces el mismo elemento?

[[18, 97],
 [18, 97, 7],
 [18, 97, 7, 40],
 [18, 97, 7, 40, 75]]

Esto significa que input_sequences empieza con 18, 97 que significa can't feels. Luego va aumentando palabras a esta oración hasta llegar a 18, 97, 7, 40, 75 que es: cant feels a no get.
Se repite el mismo elemento hasta llegar al tamaño de la token list que conseguimos en el for, por lo que si hacemos correr el código tendremos listas de distintos tamaños que se iran aumentando palabra a palabra uno por uno con un mínimo de 2 palabras por el token_list[:i+1]

que hay en el código.

- ¿Qué hace esta línea de código?

predictors, label = input_sequences[:, :-1], input_sequences[:, -1]

esto separa nuestras secuencias de entradas y la última palabra será la etiqueta, por lo que tenemos cada oración con nuestra secuencia de palabras y la última palabra será lo que tratara de predecir, con esto tenemos un label de la palabra que tendría que estar prediciendo y así vemos si el modelo lo hizo bien o no.

- ¿Qué hacen las capas de Embedding, Bidirectional, LSTM y sus diferentes parámetros?

model.add(Embedding(total_words, 50, input_length=max_sequence_len-1))

Embedding convierte los índices enteros de las palabras (su formato numérico) y lo convierte en un vector denso que sea mas fácil de entrenar después. Total words agarra todas las palabras y 50 es el tamaño que tendrá el vector para cada secuencia de palabras que tenga, el input length solo le dará el tamaño de cada secuencia menos 1 porque como vimos antes el ultimo es el tag que tiene que tratar de predecir.

model.add(Bidirectional(LSTM(150, return_sequences=True)))

Bidirectional permite que la información fluya de adelante hacia atrás como de atrás hacia adelante, por lo que eres capaz de entender utilizando el contexto antes de la palabra como el contexto delante de la palabra.

LSTM es long short-term memory que se utiliza para evitar el desvanecimiento del gradiente mientras vamos recorriendo mas capas, porque cada capa el gradiente se vuelve mas pequeño. 150 es el numero de unidades que será capaz de recordar y return_sequence devuelve todas las salidas, no solo la ultima.

- Explica en tus propias palabras cómo predice el modelo entrenado.

Utilizando make_lyrics("i'm watching", 10) Podemos ir paso a paso como el modelo entrena:
Como vimos en un anterior ejemplo se empieza con una secuencia como ser [[18, 97]] que será nuestra capa con los valores i’m y watching, después de eso intenta hacer exactamente lo que hacia para entrenar el modelo pero intenta predecir el siguiente numero a estas palabras, lo que tendríamos valores como [18, 97, 7] para esta siguiente palabra, sigue haciendo lo mismo hasta llegar a 10 valores siguientes y termina de la siguiente manera:
i'm watching tryna fuckin' scream but the words won't come out search.

Que nos muestra las otras 10 proximas palabras que predijo. Podemos cambiar las palabras iniciales y nos intentara predecir que seguiría a continuación.