Aprendizaje Automático
Memoria de Práctica

Detector de bloques de Minecraft

Antón Concheiro Fernández
Roi Santos Ríos
Adriano Miranda Seoane
Iago Fernández Garrido
Jorge Rivadulla Brey

ÍNDICE

1- Introducción

En este proyecto, realizaremos un prototipo capaz de reconocer distintos tipos de bloques del minecraft. Para esto, desarrollaremos una red neuronal especializada en clasificación, que indique el tipo de bloque según una imagen que le pasemos. Este es un problema que consideramos sencillo, y su principal propósito es ser un pequeño ejercicio de prácticas.

En la vida real las redes neuronales de clasificación de imágenes son de inmensa utilidad en un amplio abanico de campos, por lo que este pequeño ejemplo puede servirnos como tarea de aprendizaje, para familiarizarnos con este campo de la Computación.

2-Descripción del problema

El problema, pese a parecer sencillo, posee sus complicaciones. Para simplificarlo, hemos usado las siguientes restricciones:

Hemos decidido solo aceptar capturas de bloques cuando es de día en el juego, o cuando haya luz suficiente (excluimos capturas en cuevas oscuras o zonas de poca visibilidad).
De no ser así, los parámetros que extraemos de las imágenes de los bloques, la media y desviación típica de cada canal RGB se verán afectados por las fotos oscuras o de noche, complicando mucho más la clasificación.

Las imágenes que analizará el sistema se compondrán solamente del bloque en cuestión, no habrá otros bloques distintos que interfieran en la imagen. Habrá varias perspectivas desde las que se pueda ver el bloque: de frente, de lado, a lo lejos etc.

La base de datos que utilizamos en este problema consta de una colección de carpetas, una por cada bloque, llenas de imágenes de estos, variando los ángulos y las distancias.

De estas imágenes, sacaremos la media y desviación típica de los canales de RGB para construir las matrices con 6 columnas de datos.

Las imágenes que utilizamos para construir la base de datos las sacamos nosotros manualmente del juego. También nos ayudamos de la Wiki del juego (citada en la bibliografía).
Fue un proceso algo lento, pero de esta manera pudimos obtener las mejores imágenes posibles.

Como ya especificamos en las restricciones, nos hemos decidido por utilizar las imágenes de bloques siempre de día, y estando estos aislados, sin otros bloques de por medio que puedan interferir. Por ahora será así, pero un buen objetivo sería analizar varios bloques en una misma imagen.

Las principales propiedades que utilizaremos serán la media y desviación típica de cada canal RGB de las imágenes, y formamos una matriz de Nx6, siendo N el número de imágenes que tengamos por bloque. Cada fila representa una imagen de un bloque en concreto, y las columnas son las medias de R, G y B y sus desviaciones típicas

	μ R	μ G	μ B	σ R	σ G	σ B
Imagen1						
...						

3- Análisis Bibliográfico

En el ámbito a tratar , minecraft, hemos encontrado una cantidad bastante sorprendente de trabajos con inteligencia artificial, ninguna directamente igual a la nuestra. El más destacable el proyecto Malmo, una plataforma de Microsoft dedicada al estudio de la IA multitarea construida sobre el videojuego de minecraft, debido a que el juego ofrece unas posibilidades casi ilimitadas para sus jugadores, es una de las mejores plataformas para que una IA desarrolle capacidades de “planeación, razonamiento, lenguaje natural y aprendizaje”.

También queremos destacar que toda la versión java de minecraft es de código abierto por lo que hay mucho trabajo de código libre tanto en contenido para el propio juego como en inteligencias artificiales el cual llevó a la creación del concurso anual Generative Design, un concurso que consiste en usar una inteligencia artificial para generar ciudades o pueblos de la forma más realista posible en minecraft.

Cabe destacar que este es solo uno de los proyectos de este estilo que hemos encontrado durante nuestra búsqueda de información. Estos proyectos no están siempre desarrollados por programadores expertos, sino que en muchos casos nos hemos sorprendido de las técnicas que, programadores novatos, han utilizado para entrenar a sus inteligencias artificiales, ya que pese a no ser las más convencionales se han probado como muy eficientes.

4-Desarrollo

Para desarrollar este proyecto, hemos decidido afrontar el problema enfocándonos en el color de las imágenes. De esta manera podemos hacer que nuestro sistema inteligente asocie los tipos de bloques a unos colores determinados. Para esta primera aproximación elegimos unos bloques con colores bastante contrastados, para facilitar el reconocimiento.

Las características que extraemos, ya mencionadas anteriormente, consideramos que son las más apropiadas para un reconocimiento mediante el color. No nos centramos en formas ni en disposiciones dentro de las imágenes, ya que todos los bloques tienen la misma forma.

Los datos no requieren de ningún tratamiento de normalización, puesto que los valores de los canales RGB ya vienen normalizados entre 0 y 1. Además la media y desv. típica de estos también estará entre esos valores.

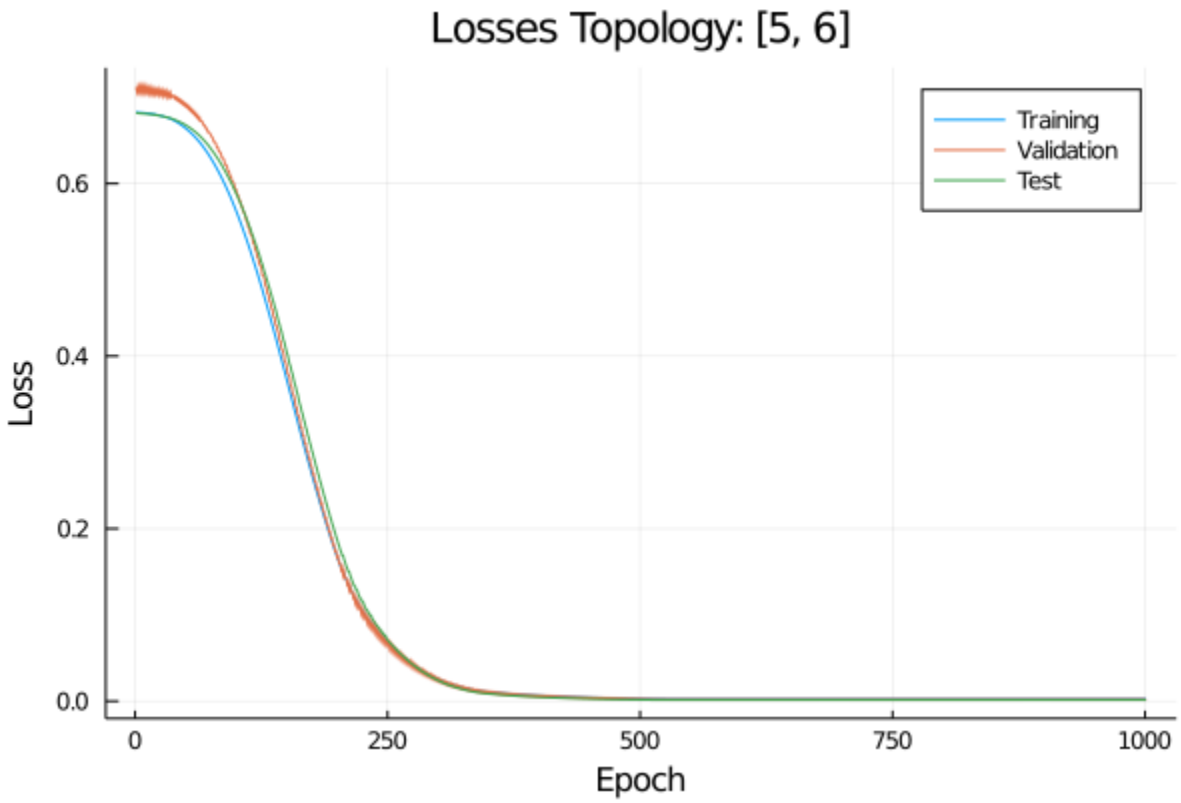
Para clasificar los bloques, utilizamos tantas neuronas de salida como “clases” tengamos (una clase será un bloque, y algunos bloques especiales, que no todas sus caras sean iguales, tendrán una clase por cada cara distinta). Entonces nuestros resultados son el grado de seguridad que tiene la RRNN acerca del bloque en cuestión. En otras palabras, si la neurona de salida madera devuelve un valor de 0.98, se puede confirmar que el bloque es madera.

Para la topología, introducimos un array con diferentes topologías para en un bucle seleccionar la mejor según el loss del conjunto de validación. Probamos con una y dos capas ocultas y vimos que con dos capas ocultas se entrenaba mucho más rápido.
Para separar los conjuntos de entrenamiento, validación y test usamos one-hot-encoding. Entrenamos a la red un máximo de 1000 epochs y repetimos todo el proceso un número arbitrario de veces, para seleccionar la mejor red, al ser aleatorio el proceso de división de conjuntos.
El entrenamiento se para prematuramente si el conjunto de validación no mejora su loss en 100 epochs, se estableció este valor porque un número más bajo hacía parar la mitad de los entrenamientos antes de tiempo y devolvía una RRNN con bastante loss.

Para el entrenamiento, el algoritmo ADAM con un ratio de aprendizaje de 0.01 se encontró como la mejor solución; ratios más altos y más bajos hacían que la red se entrenase bastante mal, parando prematuramente en la mayoría de los entrenamientos.

Las primeras iteraciones no funcionaban correctamente, ya que por la falta de patrones, al consistir el conjunto de validación un 20%, al igual que el de test, el loss en muchos entrenamientos era demasiado alto.

Nuestros resultados actuales son muy buenos, a pesar de que no empleamos una gran cantidad de patrones.
Esta es la gráfica del entrenamiento:

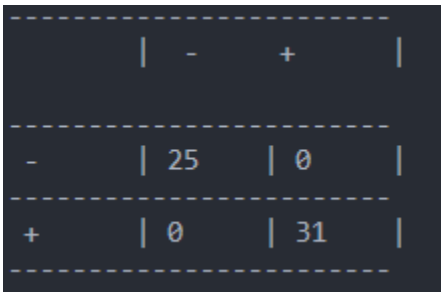


Más o menos, en todas las iteraciones vemos que el error de validación se reduce a 0 en torno a las 300 iteraciones de entrenamiento.

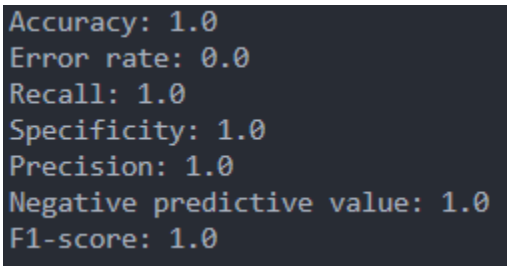
Hasta ahora, hemos creado un bucle en el que se aplican los pesos aleatorios, pero se van guardando los que mejores resultados dan, para luego aplicarlos a

las sucesivas iteraciones. Los peso los guardamos en un archivo “myweights.bson”.

Empleamos también las matrices de confusión para mejorar la visualización de los resultados. Como hasta ahora el proyecto se comporta como un clasificador binario, utilizamos un formato de matriz de confusión de ese tipo de clasificadores. Esta es de la gráfica anterior:



Como podemos observar, la tasa de falsos positivos y falsos negativos es 0, lo que nos quiere decir que es muy capaz de diferenciar entre los dos bloques iniciales que usamos.
Aquí tenemos en detalle las estadísticas obtenidas:



Durante todo el proceso hasta obtener estos buenos resultados, tuvimos muchos problemas. Hasta casi el final la matriz de confusión estaba mal implementada, y nos daba falsos positivos cuando no debía, e incluso solo llegaba a reconocer uno de los dos bloques. Por aquel entonces solo logramos un mísero y desdichado 51% de tasa de acierto.
Al final obtuvimos estos buenos resultados, pero bueno, cabe decir que se debe principalmente a que las fotos de ambos bloques tienen características muy distintas. Por ahora, para ser el comienzo, estamos bastante contentos, y creemos que el resto de la práctica evolucionará a buen ritmo.

5-Conclusiones

Como ya hemos mencionado anteriormente, el pronóstico de este trabajo es bueno. Esto se debe principalmente a la simpleza de esta primera iteración del desarrollo de la práctica, hasta ahora solo tiene que reconocer 2 bloques, por lo que actualmente se considera un problema de clasificación binaria.

El sistema que empleamos creemos que es el adecuado para este tipo de problemas, ya la propia definición del sistema de clasificación multiclase encaja perfectamente con el problema que planteamos para resolver.

Actualmente, el mayor bottleneck que tenemos es adquirir más imágenes para añadir bloques. Es una tarea bastante tediosa y consume bastante tiempo, además de que en el propio juego hay sobre unos 150 bloques.
Igualmente, nos centraremos en la realización de las funciones principales, ya que añadir más bloques no debería cambiar el funcionamiento principal del sistema.

6-Trabajo futuro

El primer enfoque futuro que le daremos será para ampliar el sistema, añadiendo muchos más bloques para que reconozca, convirtiéndolo así en un sistema de clasificación multiclase. Será excluyente, puesto que en el caso de los bloques, solo puede ser 1 a la vez, y nos afirmará el grado de seguridad que tiene acerca del bloque.

Aplicando el trabajo a un ámbito más general, podríamos emplear el sistema inteligente para cualquier tipo de clasificación que se base en los colores de las imágenes, por el hecho de que nuestro sistema inteligente trabaja exclusivamente con los canales RGB de estas.

Una posible expansión del trabajo sería que permita identificar varios elementos en una misma imagen, analizandola por zonas, en vez de al completo. Esto resultaría en un sistema bastante más complejo pero es una funcionalidad que consideramos muy útil, y aplicable en innumerables casos.

Por ahora, este proyecto es muy sencillito y lo hemos limitado bastante con las restricciones que impusimos, pero mismamente podríamos ampliarlo más añadiendo los bloques de noche, creándolos como clases nuevas, igual que haremos con los bloques que tienen caras diferentes.

7-Bibliografia

Para obtener la información y los conocimientos necesarios hemos buscado información de diversas fuentes entre las que se encuentra la wiki oficial de la propia aplicación (https://minecraft.gamepedia.com/Minecraft_Wiki), el libro Dive into Deep Learning cullos autores son Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola,
La información relevante del proyecto Malmo:
https://www.microsoft.com/en-us/research/project/project-malmo/
https://rincondelatecnologia.com/microsoft-busca-inteligencia-artificial-minecraft/
Del concurso Generative Design:
http://gendesignmc.engineering.nyu.edu/