**Presentación sobre CE aplicada al transporte físico**

Buenos días. Somos Vadim y Willow. El tema de nuestra presentación será CE aplicada al transporte físico. Quisiera empezar con el contenido de la presentación.

Como primer punto voy a hablar sobre el algoritmo que se llama Population Based Training o entrenamiento basado en la población.

De segundo, W va a presentar os el algoritmo genético para la optimización del control del tráfico.

Empezamos con PBT y las empresas que desarrollaban este algoritmo. **Waymo**, antes conocida como **Google self-driving car project,** es una empresa que desarrolla los vehículos autónomos.

**DeepMind** es una compañía de inteligencia artificial inglesa. Originalmente el algoritmo fue desarrollado por DeepMind para mejorar los videojuegos.

DeepMind y Waymo se unían para encontrar un proceso más eficiente para la capacitación y la optimización de los algoritmos de autoaprendizaje de la empresa para la conducción autónoma.

**Descripción téchnica del problema:**

Generalmente hay dos procesos separados básicos en el aprendizaje profundo: el **entrenamiento de modelo** y el **tuneo de hiperparámetros**.

Los investigadores se preguntaban si sería posible combinar estos dos procesos separados para lograr un rendimiento mejor y más rápido. Es decir, mientras los modelos están entrenando, se adaptan los hiperparamentros. De este modo se formaba la idea del entrenamiento basado en la población.

¡Pero primero veamos qué significa el tuneo de hiperparametros! Algunos ejemplos para un tuneo de hiperparametros: un programa de tasa de aprendizaje (ingl. learning rate) o un programa de conexiones celulares de arquitectura neuronal (ingl. Neural Architecture Cell Connections)

Existen dos modelos comunes para el tuneo de hiperparámetros: búsqueda aleatoria (ingl. Random/Grid Search Optimisation) y ajuste manual (ingl. Sequential Optimisation).

**Con el ajuste manual (Sequential Optimisation**), los investigadores deben adivinar los mejores hiperparámetros, entrenar a sus modelos con ellos y luego evaluar el rendimiento. Esto se hace una y otra vez, hasta que el investigador esté contento con el rendimiento de la red. Aunque esto puede resultar en un mejor rendimiento, la desventaja es que esto lleva mucho tiempo. Hay también la posibilidad de automatizar este proceso, como la optimización bayesiana, pero todavía lleva mucho tiempo.

**En la búsqueda aleatoria (Random/ Grid Search Optimisation),** se aplica muchos programas de hiperparámetros aleatorios sobre múltiples tipos de hiperparámetros para entrenar diferentes redes de forma independiente y en paralelo. Entonces se puede determinar el mejor modelo de rendimiento. La desventaja es que este modelo necesita muchos recursos de cómputo a causa de muchos procesos ejecutados en paralelo.

**PBT se puede ver como un híbrido de ambos modelos**! El entrenamiento basado en la población PBT de redes neuronales permite a los trabajadores (ingl. Workers) **explotar** los resultados parciales de otros trabajadores y **explorar** nuevos hiperparámetros a medida que avanza el entrenamiento. Echemos un vistazo al algoritmo ahora.

(El método, se llama Train() y obtiene un parámetro de populación inicial. Después hay un bucle-For que tiene que ser en paralelo. Por ejemplo: aquí podemos ejecutar hasta el número de experimentos. Y también hay un bucle regular while(¡numeroDePasos). El entrenamiento dura **T** pasos en total (T está depende del dominio). La función **step() (Si nuestro modelo es una red neuronal, generalmente optimizamos los pesos Gama de manera iterativa, por ejemplo, utilizando un descenso de gradiente estocástico para la función objetivo Q.)** realiza un paso de optimización, y la función de evaluación **eval()** evalúa el modelo en el conjunto de validación y devuelve la métrica de evaluación. )

Aquí hay tres funciones específicas de PBT: **ready(), exploit()** y **explore().**

El método **ready()** toma el paso de tiempo actual t, y devuelve un bool si la ejecución del entrenamiento debe compararse con los otros agentes (Workers) en este paso de tiempo o no.

Si ready() devuelve True, el rendimiento del modelo se compara con el resto de la población, y si se cumple un criterio, se reemplaza con una copia de un modelo con mejor rendimiento y sus hiperparámetros con la ayuda de **exploit()-**función: por ejemplo con Bynary Tournament.

La función **explore()** se llama si se reemplaza el modelo y el hiperparámetro con uno diferente de la población. (Reemplaza los hiperparámetros y el modelo copiados por una versión cambiada.)

Al final se actualizará la población y después devolvemos el mejor modelo.

**Conclusión:**

Como conclusión puede decir que PBT ahorra tiempo y recursos.

Además, se tiene hiperparámetros dinámicos. Y al final hay ninguna intervención en la construcción del modelo.

A la derecha un GIF que demuestra la función de PBT. Y ahora Willow le mostrará los algoritmos genéticos para la optimización del control del tráfico.