Nathan Amsellem
Valentin Porchet
Enzo Hummady

Poké Deep - Sprint 1

1. Projet initial

Le projet que nous souhaitions mener consiste à répliquer l'expérimentation d'OpenAI sur Minecraft mais dans un environnement différent. Dans sa publication Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos, l'équipe de recherche apprend un agent à réaliser une suite d'actions dans un ordre précis pour atteindre un objectif prédéterminé. La différence par rapport à de l'apprentissage par renforcement réside dans l'étape dite de préapprentissage. En utilisant des données vidéos (2000 heures) avec les inputs correspondants issues de joueurs expérimentés, un premier modèle d'apprentissage supervisé est entraîné pour reconnaître quelles actions de clavier et souris correspondent à quelles images. Ce modèle permet ensuite de débloquer un gigantesque volume de données. En effet, en utilisant des vidéos Youtube que l'on annote grâce au modèle, OpenAI obtient 70 000 heures de vidéos annotées. Cette approche semi-supervisée est nécessaire afin de cloner le comportement observé dans ces vidéos. A la fin de cette étape, l'agent est capable de "jouer" à Minecraft en effectuant les actions de base mais il ne sait pas encore quoi faire. La dernière phase consiste à entraîner par renforcement cet agent pour atteindre l'objectif désiré.

Pour répliquer ces résultats sur le jeu Pokémon Emeraude, nous avons commencé par l'acquisition des données: d'une part de nos propres moyens pour avoir le dataset annoté, d'autre part sur internet pour les vidéos de gameplay non annotées. Les vidéos sont capturées en 540x360px à 15fps en couleurs. Elles sont ensuites transformées en 128x128px à 15fps en noir et blanc (contre 128x128 à 20fps en noir et blanc pour Minecraft). A chaque image, nous avons une ligne dans un fichier csv qui contient la touche de clavier utilisée (ou None à défaut). Avec cette méthode, nous avons capturé plus de 3 heures de vidéos, soit près de 200 000 images.

L'étape suivante consiste logiquement à créer le modèle d'annotation vidéo. En reprenant la description de celui utilisé par OpenAI, nous en avons reproduit une version allégée (cf Annexes) avec <5 000 000 de poids (contre ~500 000 000). Malheureusement, malgré de nombreux tests sur différentes architectures de modèles (plus simples comme des mélanges de CNN/RNN ou plus compliqués avec plus de paramètres) mais nous nous sommes heurtés à plusieurs problèmes.

Premièrement, des "Out Of Memory" errors à cause de la mémoire disponible dans nos GPU (3070 laptop 8Go). Sur ce point nous n'avons pas de solutions à part travailler sur des plus gros GPU. Deuxièmement, l'entraînement se concentre uniquement sur la classe majoritaire au point de ne plus avoir besoin de données pour prédire cette classe. C'est-à-dire que quelque soit la donnée d'entrée (image complètement blanche, complètement noire, ou n'importe quelle image du jeu), le modèle retourne le même vecteur de prédictions, généralement au millionième près. Pour tenter de contrer ce problème nous avons tenté de rééquilibrer les données mais sans succès.

Nathan Amsellem
Valentin Porchet
Enzo Hummady

Dans l'état actuel, une epoch prend en moyenne 10 minutes, ce qui ralentit considérablement les expérimentations lorsqu'on lance 10 ou même 5 epochs pour voir si un changement a un impact.

A titre de comparaison, OpenAI dit avoir entraîné leur modèle sur un cluster de 32 A100 pendant 4 jours pour faire 20 epochs sur les 70 000 heures de vidéos.

2. Projet souhaité :

Au vu des problématiques soulevées par le 1er projet et après une courte entrevue avec vous, nous avons pensé à mettre en place un reinforcement learning plutôt basique autour de l'API poke env qui permet de manipuler les actions de base dans pokémon et qui propose aussi des solution de renforcement orientées autour de gymnasium d'open ai.

Cependant, dans la mesure où l'apprentissage par renforcement est quelque chose de totalement nouveau pour la majorité du groupe et que le cours n'est pas directement orienté vers le renforcement, nous avons pensé qu'il serait assez complexe de rendre quelque chose de correct et de fonctionnel pour la fin de session.

Après consultation des sujets proposés dans la matière traitement d'images et vision par ordinateurs nous avons voulu changer de direction pour notre projet. En effet, le projet propose de travailler avec des réseaux antagonistes génératifs afin de générer de nouvelles images. Nous souhaitons donc aujourd'hui réaliser un projet plus complet et plus robuste qui répondra aux attentes des deux matières avec un travail plus approfondi que si nous avions mené nos deux projets séparément.

Le but de ce projet serait de générer des images, des caractéristiques, et noms de pokémon à partir d'un bruit aléatoire, en utilisant d'un GAN. Le GAN sera composé d'un générateur et d'un discriminateur, qui seront entraînés en compétition sur un ensemble de données contenant plus de 800 espèces de pokémon.

Nous souhaitons donc à présent développer cette solution autour de 2 datasets créés par Valentin.

Le premier dataset est un dataset d'images de pokémon qui à été créé à partir du site https://pokemondb.net/ et les images qui convenaient le mieux à cette étude ont été récupérées à partir de plugins d'enregistrement d'images sur une URL donnée en parcourant la base sur les 900 Pokémons qui constituaient l'univers du jeu au moment où il a été créé. Il compte en tout plus de 14 000 images de Pokémon. Ce dataset n'est pas équilibré ce qui veut dire que certains pokémons possèdent plus d'images que d'autres, les subsets : [train, test] seront donc non équilibrés. Une data augmentation avec une rotation des images et une inversion selon l'axe vertical a aussi été pratiquée afin d'augmenter le nombre de données d'entrées.

Nathan Amsellem
Valentin Porchet
Enzo Hoummady

Image dataset pokémon : Palkia “2.png”

Le second dataset est un dataset qui référence les caractéristiques associées à chaque pokémon.Ce dataset recense les mêmes pokémons que le dataset d’images. Celui ci nous permettra d’associer les caractéristiques suivantes à chacune de nos images :

- Le Numéro du pokémon, dans l’ordre de sa sortie : **No**
- Le nom du pokémon : **Name**
- Le type principal du pokémon : **Type1**
- Le type secondaire du pokémon : **Type2**
- Le cumul des statistiques du pokémon : **Total**
- Le nombre de points de vie du pokémon : **HP**
- Le nombre de points d’attaque de base du pokémon : **Attack**
- Le nombre de points de défense de base du pokémon : **Defense**
- Le nombre de points d’attaque du pokémon sur le type qu’il contre : **Sp. Atk**
- Le nombre de points de défense du pokémon sur le type qu’il contre : **Sp. Def**
- Le nombre de points de vitesse du pokémon : **Speed**
- La génération dans laquelle est apparu le pokémon : **Generation**
- Le pokémon est-il légendaire : **Legendary**
- La couleur du pokémon : **Color**
- La taille du pokémon : **Height**
- Le poids du pokémon : **Weight**

No	Name	Type1	Type2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed
1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45
2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60
3	Venusaur	Grass	Poison	525	80	82	83	100	100	80
4	Charmander	Fire		309	39	52	43	60	50	65
5	Charmeleon	Fire		405	58	64	58	80	65	80
6	Charizard	Fire	Flying	534	78	84	78	109	85	100

PokeDataset.csv

Nous ne savons pas encore quelles sont les caractéristiques qui pourront nous être utiles pour la réalisation du GAN. En revanche, nous pouvons être certain que le type du pokémon pourra entrer en ligne de compte en plus des images pour diriger les prompts.

De plus, toutes les images ne possèdent pas le même format. Nous avons donc fait le choix au niveau du pré-process des images de toutes les reformater en 128x128, ce qui paraissait être un compromis équitable de remise à l'échelle de nos données. Les images de ce

Nathan Amsellem
Valentin Porchet
Enzo Hummady

dataset au format png n'ayant pas de fond, nous avons du remettre un fond blanc afin d'avoir des images traitables. Une phase vérification directe à partir de visualisation, soit dans les dossiers, soit au niveau du code a aussi eu lieu afin de confirmer la conformité des données et détecter certaines erreurs (dans la récupération des données notamment).

Ceci permettrait d'avoir une solution de deep learning "à étages" et modulable :
On peut imaginer des solutions variées et qui répondent à certaines problématiques du jeu avec par exemple :

- Générer aléatoirement un pokémon [image/asset, caractéristiques, nom] et ses évolutions.
- Générer un pokémon à partir de son/ses types.
- Générer un pokémon à partir de son nom
- Générer un pokémon aléatoirement

Toutes ces réalisations pourront être envisageables à partir du moment où nous jugerons les précédentes comme fonctionnelles.

Nathan Amsellem
Valentin Porchet
Enzo Hoummady
Annexes

```
    input_shape = (128, 128, 1)
    num_classes = 9

    input_shape = (FRAMES_PER_BATCH, ) + input_shape

    inputs = Input(shape=input_shape)
    x = layers.Conv3D(filters=16, kernel_size=(5, 1, 1),
padding='same', activation=None)(inputs)###
    x = layer_norm()(x)

    x = resnet_stack(x, 8, 2)###
    x = resnet_stack(x, 16, 2)###
    x = resnet_stack(x, 16, 2)###

    x = layer_norm()(x)

    x = layers.Reshape((FRAMES_PER_BATCH, -1))(x)

    x = frame_wise_dense(x, 32)###
    x = frame_wise_dense(x, 512)###

    num_heads = 4###
    head_dim = 16###
    x = multihead_attention(x, num_heads, head_dim)

    x = layers.Add()([x, x])
    x = layers.LayerNormalization()(x)

    x = frame_wise_dense(x, 2048)###
    x = frame_wise_dense(x, 512)###

    outputs = layers.Dense(num_classes, activation='softmax')(x)

    model = Model(inputs, outputs)
    model.summary(expand_nested=False)
    return model
```

(Chaque ‘#’ correspond à une division par 2 du paramètre par rapport à l’implémentation décrite par OpenAI)