



## Groupe D :

Alexis Gosselin  
Gabriel Rochaix--Yamamoto  
Issam Alouane  
Jonas Lavaur  
Justin Jourdant

# RAPPORT DE PROJET : Quel est le type de ce Pokémon ?!



Apprentissage Profond - IMM  
Département Sciences du Numérique - 2A  
Année 2022-2023

# Sommaire

I) Description du sujet choisi.....	3
II) Méthodologie d'acquisition et d'annotation des données.....	6
III) Méthodologie de partitionnement des images.....	7
IV) Estimation de la complexité du problème et description des résultats attendus.....	8
V) Script de chargement des données.....	9
VI) Sélection d'images de la base de données.....	11

# I) Description du sujet choisi

L'objectif de ce projet est, comme son nom l'indique, de pouvoir déterminer le type d'un pokémon à partir d'une image de celui-ci. Pokémons est une franchise créée par Satoshi Tajiri en 1996, notamment célèbre pour ses jeux vidéos, dans des séries éditées par Nintendo. La licence a par la suite été adaptée en dessin animé, en cartes à jouer et à collectionner, en mangas, en film... Il existe même des trains pokémons !



Pour pouvoir trier ces pokémons par type, nous allons utiliser un algorithme d'apprentissage profond en utilisant une base de données constituée d'un grand nombre d'images de pokémons labellisés par type. Les types dans pokémon correspondent en quelque sorte à des "éléments" auxquels sont rattachés les pokémons. Voici la liste des types présents dans cet univers :

POKÉMON TYPE SYMBOLS

NORMAL	FIRE	WATER	ELECTRIC	GRASS	ICE
FIGHTING	POISON	GROUND	FLYING	PSYCHIC	BUG
ROCK	GHOST	DRAGON	DARK	STEEL	FAIRY

Nous avons décidé de nous concentrer sur les quatres premières générations de pokémon pour pouvoir multiplier les images d'un même pokémon (cartes, figurine, sprite de jeu, images de dessins animés etc.)



Voici Darkrai, un pokémon de la quatrième génération (Diamant et Perle) de type ténèbre. Cette première image est une photo d'une figurine à son effigie. Ci-dessous, se trouvent deux autres images pour ce même pokémon qui l'illustre sous d'autres formats (modèle 3D et carte).



Ce pokémon est assez intéressant car sa couleur noir est un assez bon indicateur qu'il est de type ténèbre et il est assez évident pour un humain de le classer dans ce type. On aurait cependant éventuellement pu penser que Darkrai était un type spectre (voir un type poison ou psy).



Le pokémon représenté ci-dessus (tiré du dessin animé éponyme) est le plutôt célèbre Dracaufeu de type feu et vol. On voit ici facilement qu'il peut être rattaché au type feu entre celle sur sa queue et celles qu'il crache. De la même façon, ses ailes le rattachent au type vol de façon plutôt évidente. Cependant le type dragon aurait également pu lui être rattaché car il ressemble quand même assez à l'image du dragon que l'on peut avoir dans notre imaginaire européen.

Ce pokémon lève aussi une seconde question : celle des pokémons double types, c'est-à-dire un pokémon associé à deux types au lieu d'un pour une majorité. Ce problème de classement sera abordé par la suite et influencera également les résultats que l'on pourra attendre de notre algorithme.

## III) Méthodologie d'acquisition et d'annotation des données

Pour l'instant, nous nous sommes limités aux quatre premières générations et nous n'avons pas fini d'ajouter tous les pokémons. Nous continuerons d'en ajouter au fur et à mesure pour compléter la base de données.

Pour la génération 1, les environ 530 images ont été acquises une par une à la main (via Google Images) et regroupées et triées dans des dossiers correspondant au type auquel le pokémon est rattaché. Les images ont été choisies pour représenter les pokémons dans des contextes les plus variés possibles (modèle 3D, représentation dans l'animé, dessin, peluche,...).

De plus, pour les générations 2 et 3, la même méthodologie a été appliquée. Enfin, les images de la 4ème génération étaient prises d'un archive zip contenant les images de toutes les générations et triées à la main pour mettre chaque pokémon dans sa classe correspondante.

Pour l'annotation des données, nous avons donc choisi de regrouper les images des pokémons dans des dossiers dépendant de leur type : Normal, Feu, Eau, Électrique, Plante, Glace, Combat, Poison, Sol, Vol, Psy, Insecte, Roche, Spectre, Dragon, Ténèbres, Acier, Fée.

Enfin, pour les pokémons multi-types nous avons décidé de mettre le pokémon seulement une fois dans le type qui nous semble le plus pertinent. Cependant dans ce dossier de type, nous avons créé un sous-dossier pour les distinguer. Par exemple, Florizarre est de type plante/poison, nous l'avons mis dans le dossier pour les pokémons de types plante (car cela nous semblait le plus logique) et dans ce dossier nous avons donc créé un sous-dossier poison pour "ranger" et où seront également placé les autres pokémons du double type plante-poison.

### **III) Méthodologie de partitionnement des images**

Nous avons mis toutes nos images dans le dossier global et nous avons ensuite écrit un script python qui permet de répartir toutes les images dans les dossiers train, validation ou test selon un certain pourcentage qu'on choisit. Nous avons choisi une répartition 80/10/10. Le script essaie de bien respecter cette répartition par type, pour éviter le cas où on ne teste pas une classe parce qu'on a mis toutes les images de la classe dans train ou validation. Le script est dans le git : partition.py

## **IV) Estimation de la complexité du problème et description des résultats attendus**

Avec une base de données contenant plus de 1500 images, la complexité de ce projet sera modérée à élevée en raison de la taille (relativement grande) de la base de données, le temps d'entraînement sera alors un peu élevé et nécessitera donc des ressources informatiques suffisantes pour avoir un temps raisonnable. Nous prévoyons d'utiliser un modèle de deep learning avec une architecture convective pour extraire les caractéristiques clés des images. Le modèle sera entraîné sur les images de la base de données avec un algorithme de descente de gradient stochastique (SGD) pour minimiser la fonction de coût. Nous prévoyons de tester différents modèles et hyperparamètres pour trouver la meilleure combinaison pour notre problème de classification. Les résultats attendus sont une classification précise des images de Pokémons inconnus avec une précision de classification supérieure à 90 %.

## V) Script de chargement des données

La différence principale de notre base de données avec l'exemple donné sur moodle est que nous avons des sous-types dans des sous-dossiers. Le script à lancer est scriptLoad.py et les fonctions auxiliaires sont dans loadimages.py

Pour rendre plus lisible et modulable le code nous avons codé des fonctions auxiliaires :

- load\_image qui permet d'enregistrer l'image à l'index courant et la classe courante
- load\_sub\_type qui teste pour chaque item du répertoire, s'il s'agit d'un fichier (image) ou d'un sous-dossier. S'il s'agit d'une image, la fonction appelle load\_image, s'il s'agit d'un sous-dossier, la fonction s'appelle récursivement. Cela permettrait donc théoriquement également d'avoir des sous-sous-type. Pour l'instant load\_sub\_type ne change pas l'index de la classe car au départ nous allons seulement considérer les types principaux des pokémons, nous verrons plus tard lorsque l'on considérera les types secondaires.
- count\_images qui compte récursivement toutes les images dans tous les dossiers ce qui permet d'initialiser les x\_train et y\_train à la bonne taille.

```
23 ▼ def load_image(path, image_size, x, y, current_index, idx_class):  
24     #<0xa0>Ouverture de l'image  
25     img = Image.open(path)  
26     #<0xa0>Conversion de l'image en RGB  
27     img = img.convert('RGB')  
28     #<0xa0>Redimensionnement de l'image et écriture dans la variable de retour x  
29     img = img.resize((image_size,image_size))  
30  
31  
32     x[current_index] = np.asarray(img)  
33     # Écriture du label associé dans la variable de retour y  
34     y[current_index] = idx_class  
35     return current_index+1  
36  
37 ▼ def load_sub_type(type_path, image_size, x, y, current_index, idx_class):  
38     dirs = os.listdir(type_path)  
39  
40     for item in dirs:  
41         itemPath = os.path.join(type_path, item)  
42  
43         if os.path.isdir(itemPath):  
44             current_index = load_sub_type(itemPath, image_size, x, y, current_index, idx_class)  
45  
46         elif os.path.isfile(itemPath):  
47             current_index = load_image(itemPath, image_size, x, y, current_index, idx_class)  
48         else:  
49             print("Unknown : " + itemPath)  
50  
51     return current_index  
  
9 ▼ def count_images(path):  
10    dirs = os.listdir( path )  
11  
12    count = 0  
13    for item in dirs:  
14        itemPath = os.path.join(path, item)  
15  
16        if os.path.isdir(itemPath):  
17            count += count_images(itemPath + "/")  
18        elif os.path.isfile(itemPath):  
19            count +=1  
20  
21    return count
```

Dans load\_data il reste donc plus qu'à parcourir les dossiers en appelant load\_images ou load\_sub\_type :

```
def load_data(data_path, classes, dataset='train', image_size=128):

    nb_images = 0;
    for i in range(len(classes)):
        #dirs = sorted(os.listdir(data_path + dataset + '/' + classes[i]) +"/")
        type_path = data_path + dataset + "/" + classes[i] +"/"
        nb_images += count_images(type_path)

    x = np.zeros((nb_images, image_size, image_size, 3))
    y = np.zeros((nb_images, 1))

    current_index = 0

    for idx_class in range(len(classes)):
        type_path = data_path + dataset + "/" + classes[idx_class] +"/"
        dirs = sorted(os.listdir(type_path))

        #nb_images_type = count_images(type_path)

        for item in dirs:
            #item = dirs[idx_img]
            itemPath = os.path.join(type_path, item)

            if os.path.isfile(itemPath):
                current_index = load_image(itemPath, image_size, x, y, current_index, idx_class)
            elif os.path.isdir(itemPath):
                current_index = load_sub_type(itemPath, image_size, x, y, current_index, idx_class)

    return x, y
```

## VI) Sélection d'images de la base de données

Voici une sélection d'images issues de notre base de données avec des explications de comment les trouver.

Par exemple, dans le dossier spectre on peut trouver une sélection d'images de pokémon de ce type comme :



feuforeve\_3d  
(477 qui correspond au numéro du pokémon Noctunoir dans le pokédex)



feuforeve\_dessin



ou encore 477

On a également des sous-dossiers dans le dossier des pokémons spectres comme dragon, ténèbres, vol ou poison. Par exemple dans le sous-dossier poison on peut retrouver :



ectoplasma\_3d



ectoplasma\_dessin

ou encore fantominus



Dans le dossier électrique par exemple on peut trouver :



pharamp\_dessin



raichu\_figurine

Tandis que dans le dossier sol on peut trouver :



ossatueur\_dessin



sablette\_photo



472

Enfin dans le type psy par exemple on peut trouver :



mew\_art



alakazam\_3d2



mentali\_fanart

Ce qui est intéressant au travers de ces exemples est de voir que nous sommes instinctivement capables de classer les Pokémons dans les types correspondants que ce soit par leur couleur ou par certains attributs.