Machine Learning Engineer Nanodegree

Capstone Project - `bazema_pokemon`

Baptiste Azéma
February 21st, 2021

I. Definition

Project Overview

Pokémon are creatures that inhabit the fictional Pokémon World. Each creature has unique design and skills. We will focus on the Generation One featuring the original 151 fictional species of creatures introduced in the 1996 Game-Boy games "Pokémon".

As our world is becoming more and more bizarre, a future were Pokémon and Humans coexist might be possible. We might need some tool to identify if a photo shows a Pokémon, a human, and which Pokémon it looks like.

Problem Statement

This project aims to classify images as Pokémon species. When the input image is a Pokémon, the algorithm will respond the Pokémon name. When the input image is a human face or anything else, the algorithm will respond if it's human and which Pokémon it looks like.

Inspired by udacity's CNN Project: Dog Breed Classifier

Metrics

In this project we will do 3 different image classification. To measure the performance of each model/decision tree we will use the accuracy.

> Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition:
>
> $$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$
>
> For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

> $$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
> Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.
>
> Source: *https://developers.google.com/machine-learning/crash-course/classification/accuracy*

We can also use a confusion matrix, or a graphic showing the repartition of true positives per class.

II. Analysis

Data exploration

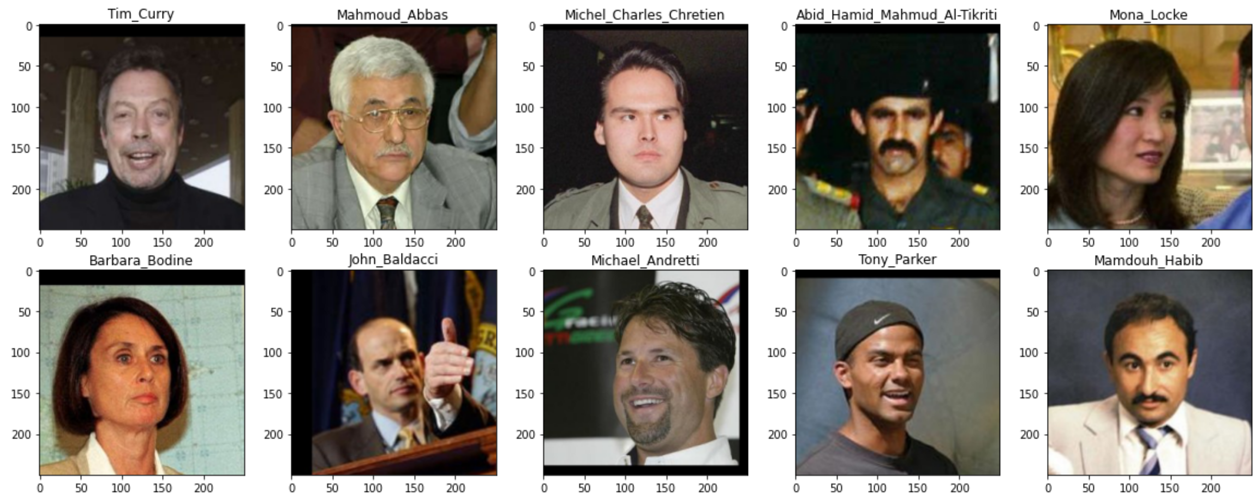
We are going to use the following datasets:

- Labeled Faces in the Wild Home. This dataset will be used to evaluate a human face detector.
- Pokemon Generation One. This dataset will be used to train and evaluate a Pokémon detector and a classifier of Pokémon species.
- CIFAR-100. This dataset will be used to train the Pokémon detector, providing examples not representing Pokémon.

lfw - Labeled Faces in the Wild Home

There are 13233 total human images. One face in each picture and each face is different.

We can see that we have every human races represented in the dataset.



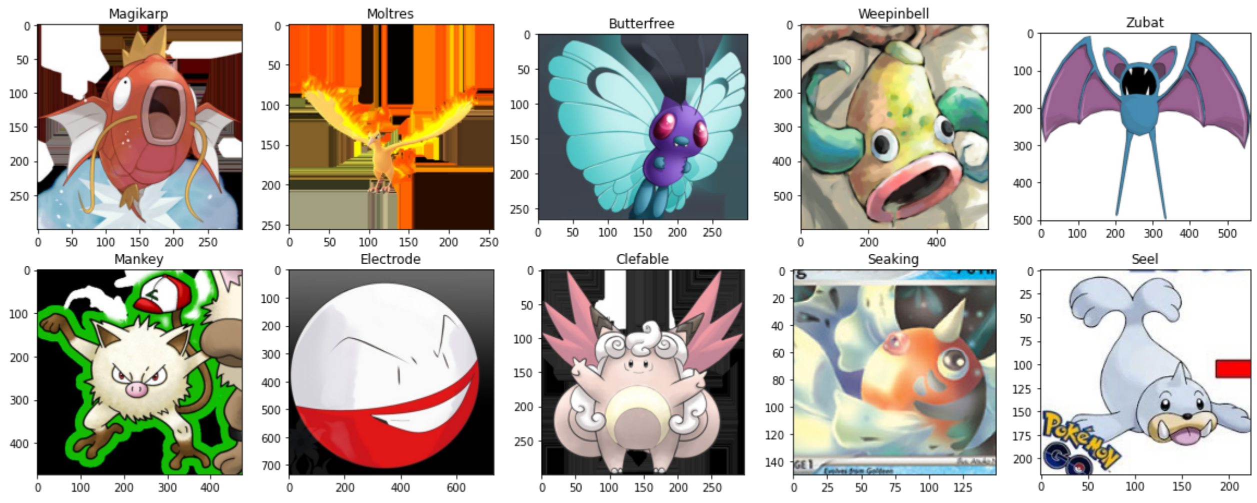
We won't use the labels of the dataset as we only want to detect if a human is present or not in the picture.

Pokémon

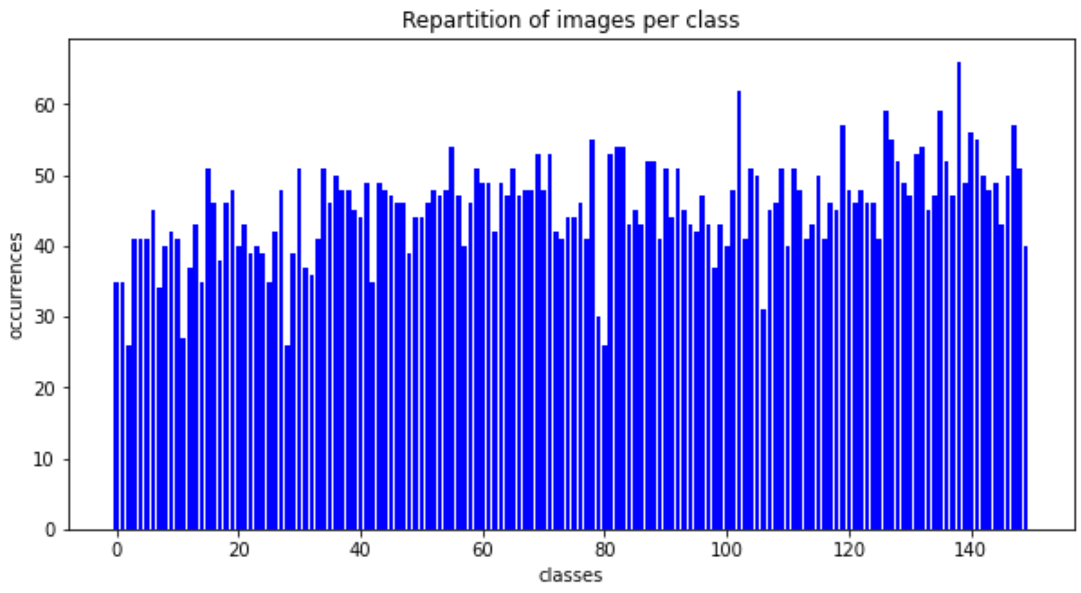
There are 6837 total Pokémon images. One Pokémon in each picture.

Number of Pokémon species in the dataset: 150. We miss 1 Pokémon in our dataset (Gen 1 is about 151 Pokémon), but we will ignore this issue

We have multiple representations of every Pokémon. This will enable us to recognize a Pokémon specie even for new pictures.



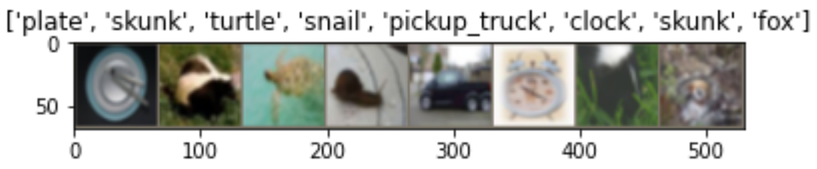
Repartition of images per class



Standard deviation is about 7.
The repartition between classes is not perfect, but it will be ok.

CIFAR-100

There are 50000 total CIFAR images. Number of classes: 100. One object or animal in each picture.


['plate', 'skunk', 'turtle', 'snail', 'pickup_truck', 'clock', 'skunk', 'fox']
0
50
0
100
200
300
400
500

Standard deviation is 0.
The repartition between classes is perfect.

With this dataset we have many exemples of objects, animal, plants, vehicles, etc...
We will use it to detect if an input image is a Pokémon or something else, thus we won't use the provided labels.

Algorithms and Techniques

Our algorithm need to take 3 decisions:

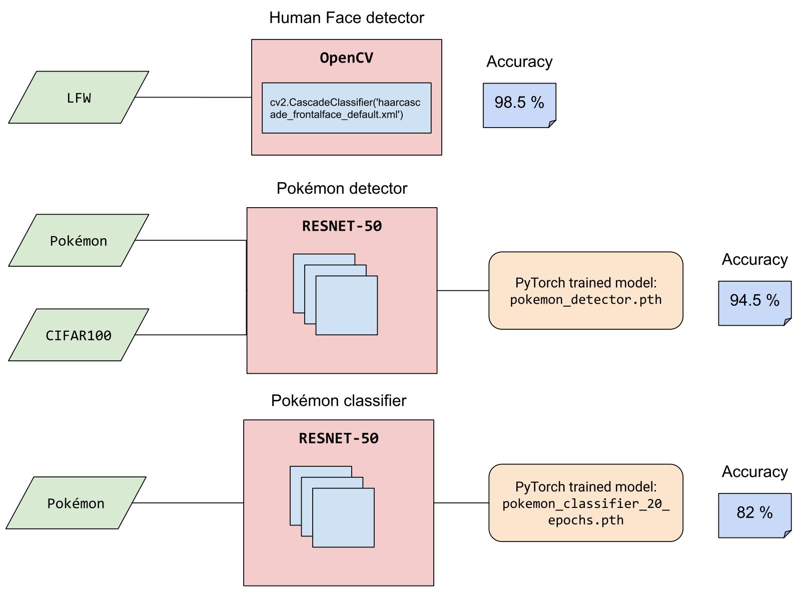
- is it human ?
- is it a Pokémon ?
- which Pokémon looked like this ?

The first item will be answered with OpenCV. The second and third items will be answered using the machine learning library PyTorch and reinforcement learning in order to create 2 models.

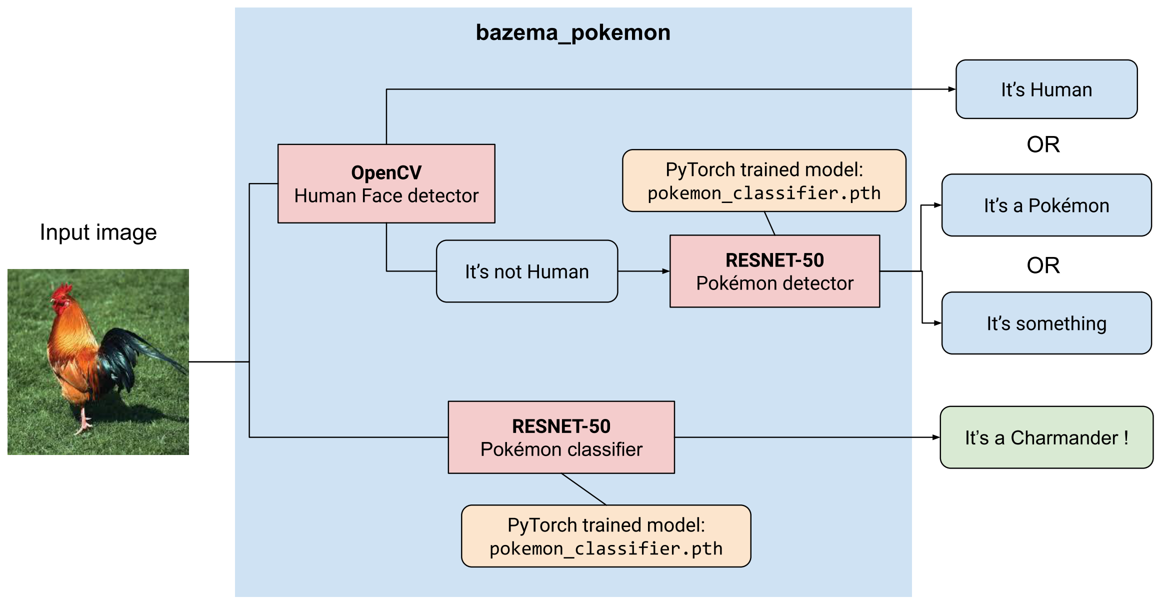
Technical environment

- OpenCV
- PyTorch
- CNN
- Transfer Learning
- Image classification
- AWS SageMaker

Training workflow


Human Face detector
OpenCV
LFW
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
Accuracy
98.5 %
Pokémon detector
RESNET-50
Pokémon
CIFAR100
PyTorch trained model: pokemon_detector.pth
Accuracy
94.5 %
Pokémon classifier
RESNET-50
Pokémon
PyTorch trained model: pokemon_classifier_20_epochs.pth
Accuracy
82 %

Prediction workflow


bazema_pokemon
Input image
OpenCV Human Face detector
It's Human
OR
PyTorch trained model: pokemon_classifier.pth
It's not Human
RESNET-50 Pokémon detector
It's a Pokémon
OR
It's something
RESNET-50 Pokémon classifier
It's a Charmander !
PyTorch trained model: pokemon_classifier.pth

Benchmark

In this project, we are not looking for the perfect model, but we will more focus on the ease of use of the final product. Thus, the threshold for the performance of our models will be an accuracy of 75%.

III. Methodology

As we have 3 models, we will address the following sections for each model: *Data processing*, *Implementation*, *Refinement*.

Detect human faces

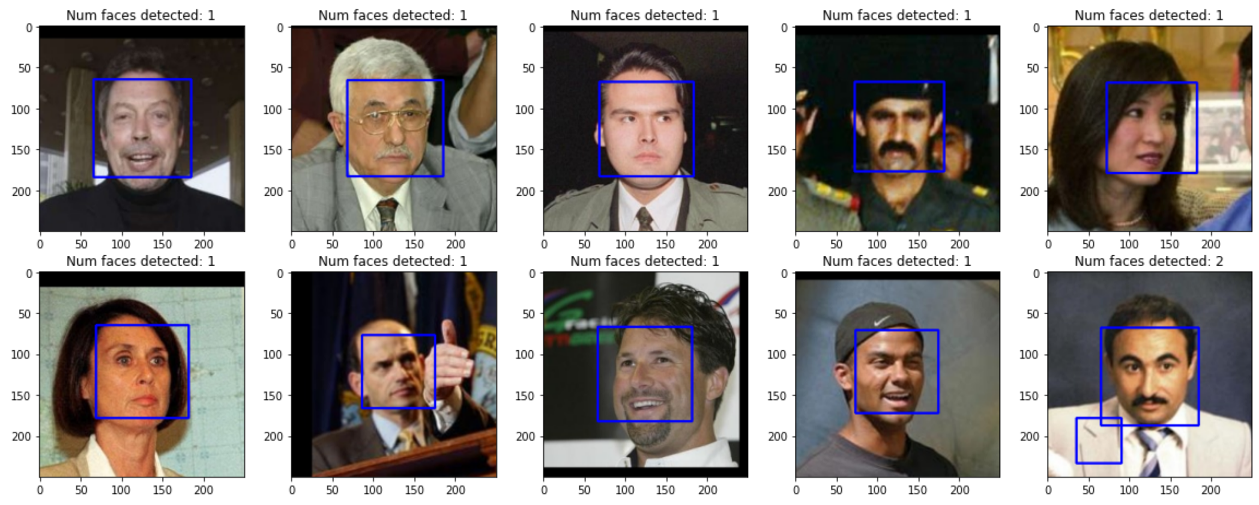
Data processing

There is no data preprocessing needed for this model.

Implementation

We use OpenCV to detect a human face, and more precisely a Haar-cascade of classifier which uses a decision tree to detect a human face in the image.

We use a pretrained cascade classifier named `haarcascade_frontalface_alt` provided by OpenCV.



As we can see, our tool detects well the faces with few false positives.

Refinement

As we used a pretrained model provided by OpenCV, the implementation was pretty straight forward, no refinement were needed.

Detect Pokémon

In order to detect Pokémon among images, we need to train a model with images of Pokémon and images of other classes.

Data processing

We split our datasets to training and testing. We use all the Pokémon dataset but just a random subset of the CIFAR100 dataset as we need to have the same mount of exemples for both classes.

dataset	total	training	testing
Pokémon	6837	5797	1023
Cifar100	6800	5800	1000

We use the following image augmentation :

- RandomResizedCrop(size=64)
- RandomHorizontalFlip()
- RandomRotation(degrees=15)
- Normalize(mean*nums=[0.485, 0.456, 0.406], std*nums=[0.229, 0.224, 0.225])

The data prepareted for training is as follows:

['Other', 'Other', 'Other', 'Pokemon', 'Other', 'Other', 'Other', 'Other', 'Other', 'Pokemon', 'Other', 'Other', 'Pokemon', 'Other', 'Other', 'Other']

Implementation

We will use transfer learning in order to speed up the training process.

Our problem is a classification with 2 classes: "Pokémon", "Other".

- "Pokémon" dataset represents the class "Pokémon"
- "CIFAR100" dataset represents the class "Other"

This is an image classification problem. Convolutional neural network are a good choice here. We use a pre-trained Resnet50 provided by PyTorch to speed-up the learning phase.

Refinement

I first had troubles to build the training dataset. The Pokémon dataset is about 6800 images and the CIFAR100 is about 50000 images. Simply concatenating the 2 datasets produced a 56800 images dataset but it was unbalanced. I used the `RandomSampler` function provided by Pytorch to select a subset of images in the CIFAR100 dataset.

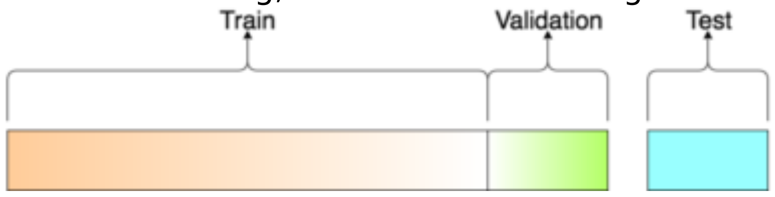
At first I tried to train on 20 epochs, but the model was overfitting. I reduced the number of epochs to 3 and the result was already good.

Pokémon classifier

Data processing

We use the "Pokémon" dataset here. Each species is a class, i.e. we have 150 classes.

We split our datasets to training, validation and testing.



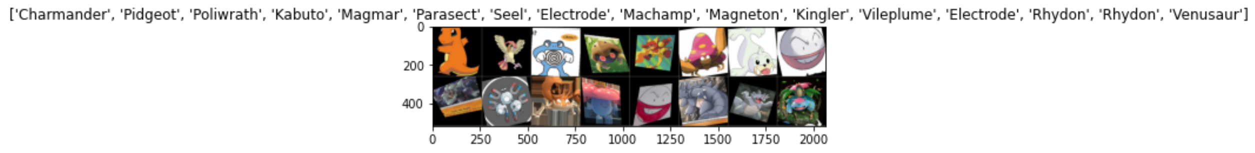
- train set for actually training the model
- validation set to compute metric during the training
- test set to compute accuracy after training on new data

dataset	total	training	validation	testing
Pokémon	6837	5456	1376	1024

We use the following image augmentation :

- RandomResizedCrop(size=300)
- RandomHorizontalFlip()
- RandomCrop(size=256)
- RandomPerspective()
- RandomRotation(degrees=15)
- Normalize(mean*nums=[0.485, 0.456, 0.406], std*nums=[0.229, 0.224, 0.225])

The data prepared for training is as follows:


['Charmander', 'Pidgeot', 'Poliwrath', 'Kabuto', 'Magmar', 'Parasect', 'Seel', 'Electrode', 'Machamp', 'Magneton', 'Kingler', 'Vileplume', 'Electrode', 'Rhydon', 'Rhydon', 'Venusaur']
0
200
400
0
250
500
750
1000
1250
1500
1750
2000

We use colab and the provided GPU for training (Nvidia Tesla T4)

Implementation

In order to classify Pokémon images, we need to train a model with images of Pokémon and images of other classes.
We will use transfer learning in order to speed up the training process.

Our problem is a classification with 150 classes, each for per Pokémon species.

- "Pokemon" dataset

This is an image classification problem. Convolutional neural network are a good choice here.

We use a pre-trained Resnet50 provided by PyTorch to speed-up the learning phase.

We use a pretrained resnet50 provided by PyTorch. We fix all layers expect the final layer.

Refinement

In my first experiments I used trained the model during too much epoch (40 epochs) and the model was overfitting. After few iterations I reduced it 5 epochs which gave me a good result without overfitting.

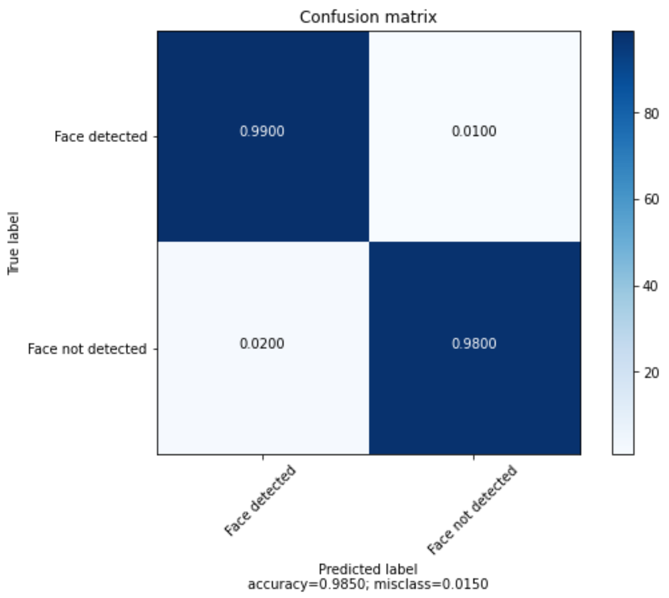
IV. Results

As we have 3 models, we will address the following sections for each model: *Model Evaluation and Validation*, *Justification*.

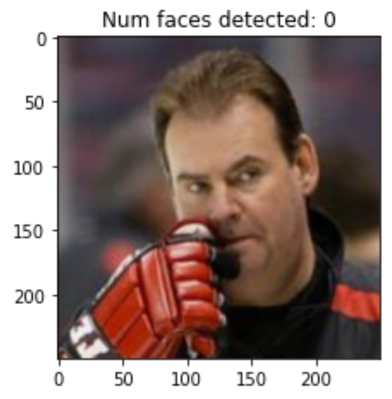
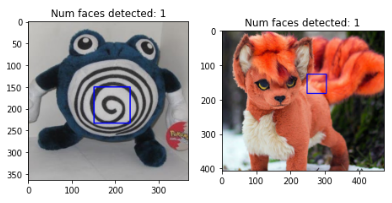
Detect human faces

Model Evaluation and Validation

To mesure the performance of this solution, we build a dataset of 200 images: 100 human faces and 100 images of Pokémon. The following confusion matrix shows the performance.



Exploration of errors

input	**interpretation**
	There is 1 false negative. The face is not well detected because of the hand covering part of the face
	These are 2 false positives.

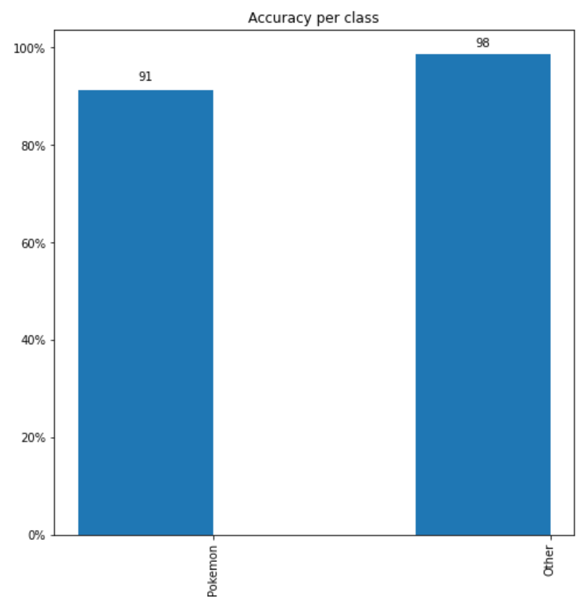
Justification

It works quite well ! It detects well when the image is a human face 99% of the time and 98% of the time when it's not a human face. We have very few False-Positives and False-Negatives.

Detect Pokémon

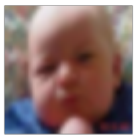
Model Evaluation and Validation

As shown in the graphic below, our accuracy per class computed on the test set is good: 91% for Pokémon and 98% for Others.



Result examples

predicted: Other, ground truth: Other



predicted: Other, ground truth: Other

predicted: Pokemon, ground truth: Pokemon

predicted: Pokemon, ground truth: Pokemon

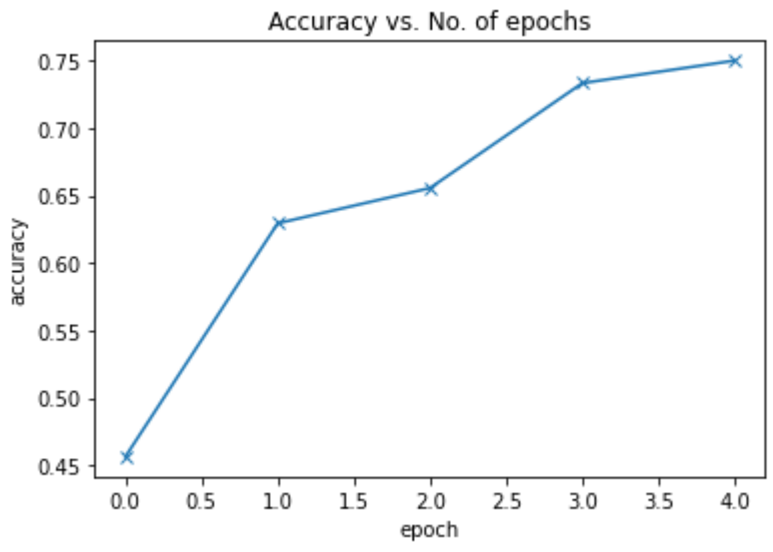
Justification

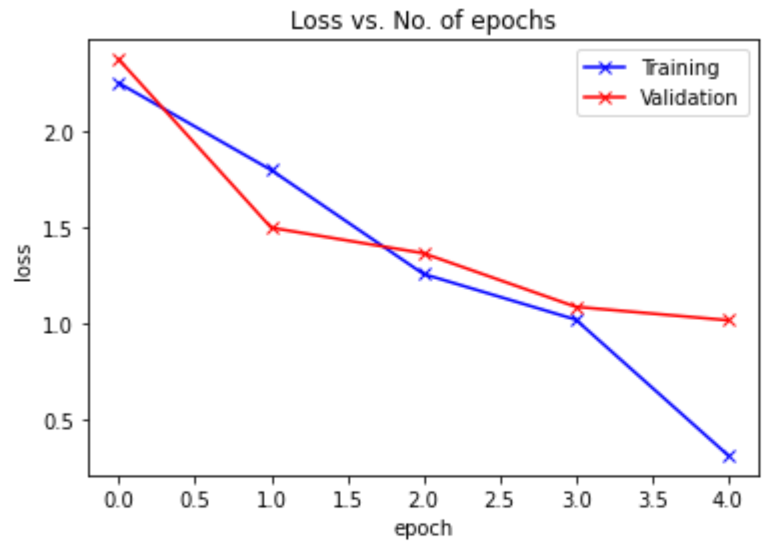
With an accuracy of 94% our final application should be able to detect if the image is a Pokémon or not.

Pokémon classifier

Model Evaluation and Validation

Performance on the validation set





Performance after training on the test set

Try to predict on new data:

predicted: Mankey, ground truth: Mankey

predicted: Persian, ground truth: Persian

predicted: Psyduck, ground truth: Psyduck

predicted: Hitmonlee, ground truth: Kangaskhan

Justification

> Accuracy of the network on the test images: 82 %

The accuracy of the model on new images is 82%. This is correct and we can now move to our final application.

V. Conclusion

Final application

Our final application is written in Python and uses:

- the OpenCV `haarcascade_frontalface_alt` pretrained classifier
- our `pokemon_detector.pth` model trained
- our `pokemon_classifier.pth` model trained

The process is, for an input image:

- take an image as input
- detect what's on the image
  - use Human detector (opencv)
  - use Pokémon detector (pytorch model)
- use the trained Pokémon classifier to give the name of the Pokémon (if it's a Pokémon), or the Pokémon it looks like (if it's a Human or something else)

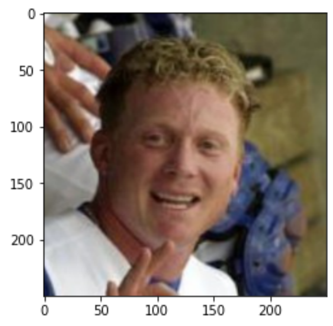
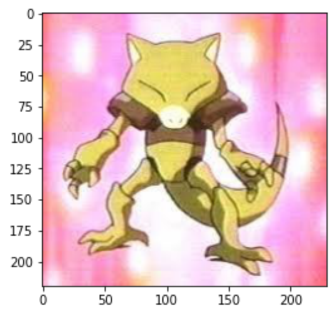
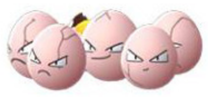
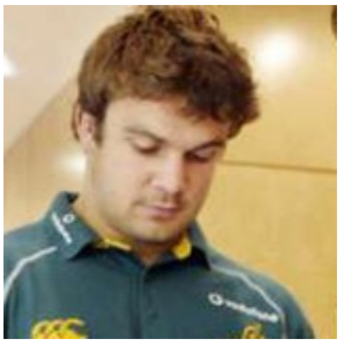
The usage is pretty simple, you can run it with Docker. You can check the prediction_workflow for the detailed process.

Exemple usage

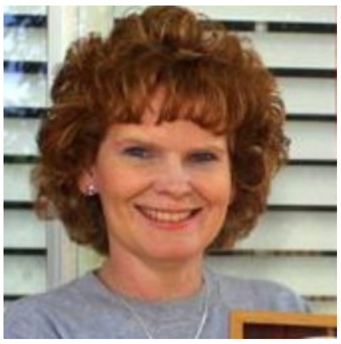
```
> docker run -v $PWD:/app bameza/bazema_pokemon:latest \
  --image_path pokemon.jpg
```

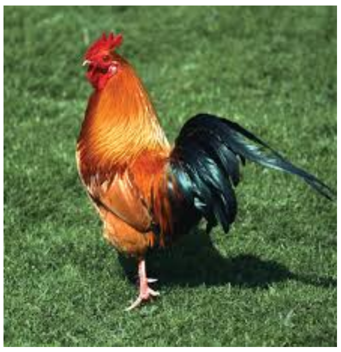
```
I guess it's a Pokemon: Abra !
Took 0:00:00.219980 to predict
```

Prediction on few images:

input	result
	It's a Human, it looks like the Pokemon Blastoise ! Took 0:00:00.202679 to predict
	It doesn't look like a Pokemon nor a Human, but it looks like Rattata ! Took 0:00:00.220836 to predict
	It's a Human, it looks like the Pokemon Exeggcute ! Took 0:00:00.274708 to predict
	It's a Human, it looks like the Pokemon Tauros ! Took 0:00:00.281719 to predict

input	result

	`It's a Human, it looks like the Pokemon Drowzee !`
	`Took 0:00:00.221947 to predict`

	`It doesn't look like a Pokemon nor a Human, but it looks like Spearow !`
	`Took 0:00:00.227700 to predict`

Reflection

It was great to perform all the steps needed to answer a machine learning problem: design the problem, look for relevant datasets, data exploration, modeling/prototyping, evaluate the model and iterate to find the best model/parameters, implement the final application embedding the models and make it available through the Docker Hub.

This project isn't perfect, the performance of the models can be improved... but we have a working Minimum Valuable Product, packaged in a simple docker image, which can be run anywhere !

Plus, it was really fun to work on this topic !

I learned a lot, especially how to use PyTorch to save a model and reuse it for the predictions.

Futur improvements

This project can be improved.. like a lot! Here is a list of possible improvements:

- benchmark more models
- better hyperparameter tuning
- automated training pipeline: train many models with many hyperparameter and automatically pick the best one.
- use AWS SageMaker for a more "production-ready" implementation

- more data augmentation
- add unit tests to have a more robust solution
- CI/CD pipeline to build/test/deploy

Resources used

- https://pytorch.org/tutorials
- https://towardsdatascience.com/how-to-train-an-image-classifier-in-pytorch-and-use-it-to-perform-basic-inference-on-single-images-99465a1e9bf5?gi=f8b2c41eddbc
- https://curiousily.com/posts/transfer-learning-for-image-classification-using-torchvision-pytorch-and-python/
- https://lionbridge.ai/articles/end-to-end-multiclass-image-classification-using-pytorch-and-transfer-learning/
- https://medium.com/@011ivand3r/pytorch-pokedex-70ad86f42568