Cloud Computing

Projet - Présentation

Laurent Bossart

Département de Sciences Informatiques
Université de Mons

27 Janvier 2023

Aperçu

Introduction

Figure 1: Objectif du projet.

Cahier des charges

- Permettre à un utilisateur de s'inscrire / se connecter sur la plateforme.
- Permettre à un utilisateur de charger une image.
- Appliquer le modèle de classification d'images sur l'image.
- Afficher les résultats du modèle de classification.
- Optimisier / compresser le modèle en définissant différents paramètres.
- Héberger l'application sur une ressource Cloud / Edge.

Choix techniques - Django

- Proposé pour le projet
- De plus en plus utilisé
- Utilisé pour de gros services (YouTube, Instagram, Spotify, ...)[1]
- Sécurisé [1]
- Documentation complète et à jour
- Rapide à prendre en mains [2]
- Permet l'implémentation rapide d'un *back end* et d'un *front end*

Figure 2: Django

Choix techniques - Templates[3]

- Learning curve fluide
- Utilisation simple en combinaison avec Bootstrap
- Facilité pour trouver des exemples en ligne

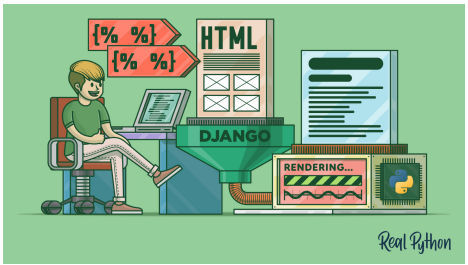

Figure 3: Templates

Choix techniques - SAAS

- Objectif du projet
- Pas besoin d'installer l'application directement sur nos appareils
- Pas besoin d'investir dans de l'hardware

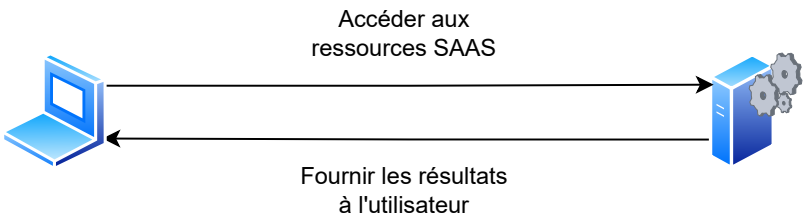

Figure 4: SAAS

Front end - Views

Listing 1: Exemple de *view*

```
@login_required
def image_upload_view(request):
    directory = Path(__file__).resolve().parent.parent
    if request.method == 'POST':
        form = ImageForm(request.POST, request.FILES)
        if form.is_valid():
            form.save()
            img_obj = form.instance
            # ============= machine learning
            model = load_model(os.path.join(directory, "Model_Suspect_Detection.h5"))

            input_dim = 224
            classes = [
                'appareil photo', 'arme', 'autre', 'baton', 'couteau', 'drone', 'gilet jaune', 'grenade',
                'personne'
            ]

            image_to_test = os.path.join(directory, "media", str(img_obj.image))

            img = PIL.Image.open(image_to_test).convert('RGB')
            x = tf.keras.utils.img_to_array(img, data_format='channels_last')
            x = tf.keras.preprocessing.image.smart_resize(x, size=(input_dim, input_dim))
            x = np.expand_dims(x, axis=0)
            start = time.perf_counter()
            prediction = model.predict(x, batch_size=64)[0]
            end = time.perf_counter()

            [...]
```

Front end - Views

Listing 2: Suite exemple de *view*

```
        [...]

        # ==========
        class_final = classes_translated[class_final]

        os.remove(image_to_test)
        Image.objects.all().delete()
        model_size = os.path.getsize(os.path.join(directory, "Model_Suspect_Detection.h5"))

        return render(request, 'upload.html', {'form': form,
                                               'class_name': class_final,
                                               'probability': probability,
                                               'elapsed': elapsed,
                                               'model_size': model_size,
                                               })
else:
    form = ImageForm()
return render(request, 'upload.html', {'form': form})
```

Front end - Templates

Listing 3: Exemple de *template*

```
{% extends "base.html" %}

{% block content %}
    <form method="post" enctype="multipart/form-data">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit" class="btn btn-primary btn-block">Upload</button>
    </form>

    {% if class_name %}
        <div class="alert alert-success">
            <h3>Successfully uploaded!</h3>
            <p>There is a probability of <strong>{{ probability }}%</strong> that you uploaded a picture
            from the category "<strong>{{ class_name }}"</strong>.</p>
            <p>The model took <strong>{{ elapsed }} seconds</strong> to execute and weights <strong>
            {{ model_size }} bytes</strong>.</p>
        </div>
    {% endif %}
{% endblock %}
```

Front end - Bootstrap

Listing 4: Exemple d'utilisation *bootstrap*

```
<html lang="en">
<head>
    <title>SAAS</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.1/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
    <style>

[...]
        {% else %}
            {% if '/accounts/login/' == request.path or '/accounts/signup/' == request.path %}
                {% block login %}
                {% endblock %}
            {% else %}
                <h1>Welcome</h1>
                <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, [...]</p>
            {% endif %}
        {% endif %}
[...]
```

Front end - View + Template + Bootstrap

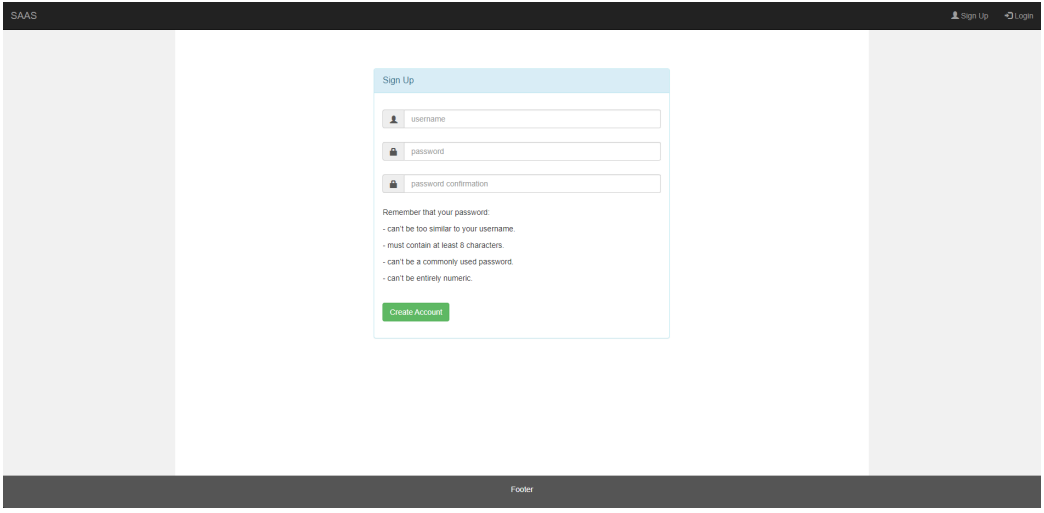

Figure 5: Rendu front end

Back end - Pruning

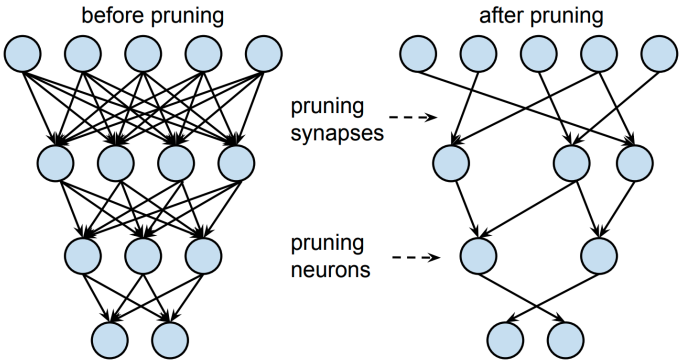

Figure 6: Pruning

Back end - Pourquoi le pruning ?

1. Coûts
2. Performances
3. Latence
4. Exigences en matière de mémoire

Figure 7: Pruning

Mise en ligne

Développement local

Développer l'application sur localement pour bénéficier de meilleures performances

Tester sur VM

Tester l'application sur la machine virtuelle utilisée pour les TPs du cours

Création du Dockerfile

Pour permettre le déploiement facile de l'application, création d'un Dockerfile

Création de l'image Docker

Créer l'image Docker de l'application en utilisant le Dockerfile

Mise en ligne sur Dockerhub

L'image Docker a été push sur Dockerhub pour la rendre facilement accessible

Figure 8: Étapes de la mise en ligne

Mise en ligne

Listing 5: Dockerfile

```
FROM python:3.10

COPY requirements.txt requirements.txt
RUN python3 -m pip install -r requirements.txt

COPY . code
WORKDIR /code/Project/backend/

EXPOSE 6660

ENTRYPOINT ["python3", "manage.py"]
CMD ["runserver", "0.0.0.0:6660"]
```

Démonstration

`http://195.154.19.214:8080`

Conclusion

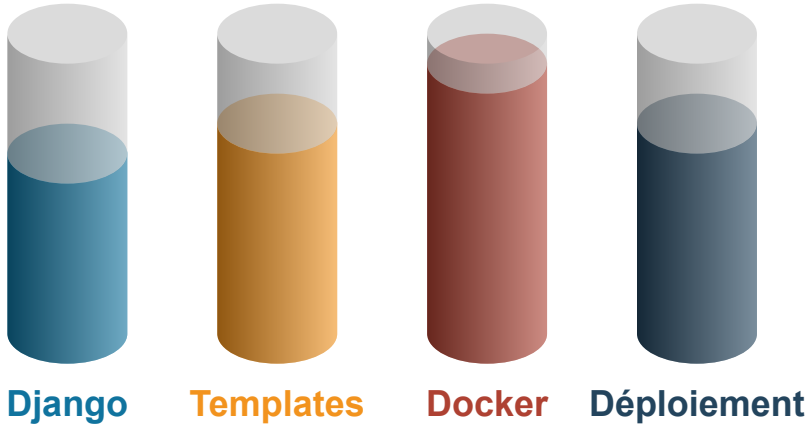

Figure 9: Conclusion

Références

[1] Dibbyyan Nat, Décembre 2021

https://inteliqoservices.com/the-best-front-end-framework-for-django/, consulté le 29/04/2022

[2] Django, 2022

https://www.djangoproject.com/, consulté le 29/04/2022

[3] Monocubed, Octobre 2021

https://www.monocubed.com/blog/why-use-react/, consulté le 29/04/2022

Merci pour votre attention

Figure 10: The Matrix