

H A C K A T H O N

# Détection de Défauts Industriels avec IA

Computer Vision • Machine Learning • Deep Learning • CBIR • VLM

4 Jours • Du Traitement d'Image à l'IA Générative

Steve Ataky, Ph.D. Eng.

# Programme du Hackathon - 4 Jours



1

JOUR 1

## Fondamentaux

Traitements d'image,  
extraction de  
caractéristiques, ML vs DL,  
métriques



2

JOUR 2

## Optimisation

Hyperparamètres, Grid  
Search, interface Streamlit



3

JOUR 3

## CBIR

Recherche par similarité,  
métriques de distance,  
base de signatures



4

JOUR 4

## VLM & GenAI

Vision Language Models,  
description automatique,  
rapport

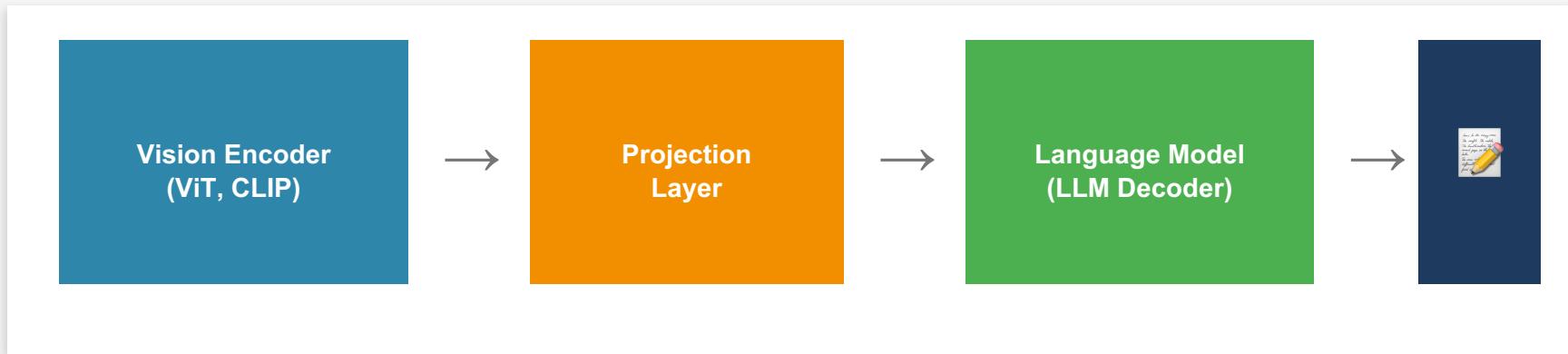
JOUR 4

# Vision Language Models & IA Générative

Description automatique des défauts avec VLM

# Vision Language Models (VLM) - Comment ça marche?

Un VLM combine vision (images) et langage (texte) dans un seul modèle



## 1. Vision Encoder

Convertit l'image en embeddings visuels (tokens)

## 2. Projection Layer

Aligne l'espace visuel avec l'espace textuel du LLM

## 3. Language Model

Génère du texte conditionné sur les embeddings visuels

# Modèles VLM Populaires

Modèle	Organisation	Caractéristiques
BLIP / BLIP-2	Salesforce	Léger, bon pour captioning, open source
LLaVA	Microsoft	Instruction-tuned, très performant
GPT-4 Vision	OpenAI	SOTA mais API payante
Claude 3	Anthropic	Excellent raisonnement, API
Qwen-VL	Alibaba	Multilingue, open source
Moondream	Community	Ultra-léger (~1.8B), local

 Pour ce hackathon: BLIP-2 via Transformers (gratuit, local, facile à intégrer)

Alternative légère: Template-based descriptions (règles basées sur la prédiction)

# Utilisation de BLIP pour Description d'Image

```
from transformers import BlipProcessor, BlipForConditionalGeneration
from PIL import Image

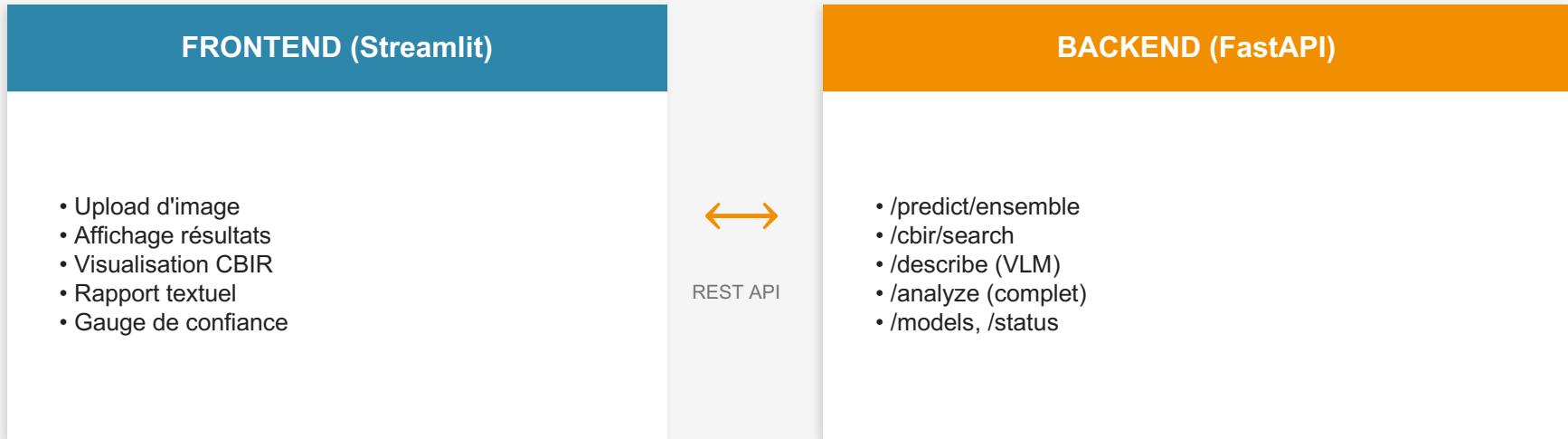
# Charger le modèle BLIP
processor = BlipProcessor.from_pretrained('Salesforce/blip-image-captioning-base')
model = BlipForConditionalGeneration.from_pretrained(...)

# Générer une description
image = Image.open('defect.jpg')
inputs = processor(image, return_tensors='pt')
output = model.generate(**inputs, max_length=50)
caption = processor.decode(output[0], skip_special_tokens=True)

# Avec prompt conditionnel
prompt = 'This industrial component shows'
inputs = processor(image, prompt, return_tensors='pt')
```

💡 Le prompt conditionnel guide la génération vers le contexte industriel

# Architecture Complète - Backend + Frontend



## COUCHE MODÈLES



# Conseils pour le Hackathon

1

## Commencer simple

Baseline fonctionnel avant optimisation

2

## Tester fréquemment

Valider chaque composant isolément

3

## Documenter le code

Commentaires clairs pour l'équipe

4

## Versionner

Git commits réguliers

5

## Métriques d'abord

Définir le succès avant de coder

6

## GPU si dispo

Colab/Kaggle pour l'entraînement

# Questions?

Bonne chance pour le Hackathon! 

Ressources: GitHub, Documentation, Équipe support