Deep Learning pour la Vision Numérique
Retour d'expériences en Épidémiologie

Utilisation du Deep Learning dans le cadre de projets autour de la maladie de Lyme

Jocelyn DE GOËR (UMR EPIA)

Webinaire IMOTEP - 01 AVRIL 2021

UMR EPIA

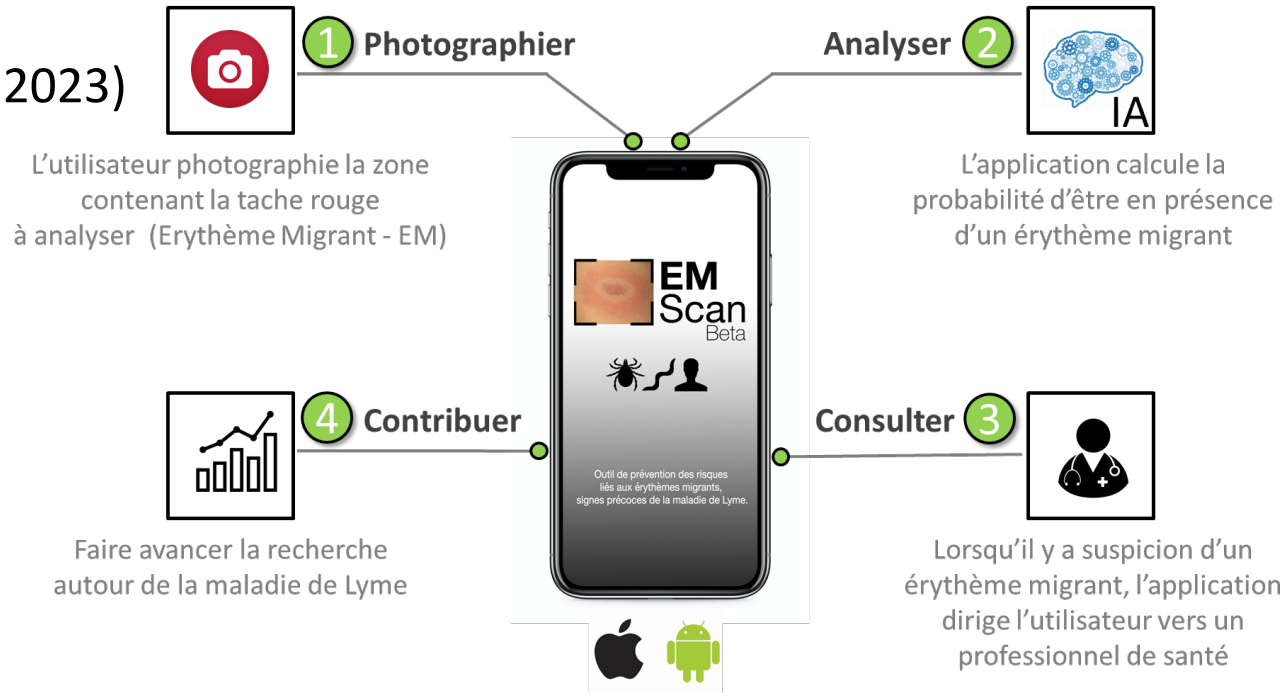
- ✓ **Démarrage en 2019 dans l'UMR EPIA d'une thématique de recherche autour du Deep Learning pour l'Épidémiologie**
  - ✓ Évaluer cette famille de méthodes
  - ✓ Voir ce que le DeepLearning peut apporter dans les problématiques en Épidémiologie
  - ✓ Collaboration avec le LIMOS (UMR CNRS, UCA, EMSE)
- ✓ **Thématique « tiques et maladies à tiques »**
  - ✓ **Deux projets de recherche**
    - ✓ 2019 : **Projet DAPPEM** (Développement d'une APPlication de détection des Erythèmes Migrants)
      - ✓ Financement : 200k€ (Fond FEDER de la région ARA) et 30k€ (MSA du Puys de Dôme)

    - ✓ 2020 : **Projet DCLIC** (Deep Convolutional Learning for *Ixodidae* Characterization
      - ✓ Financement : 10 k€ (Appel à projet « Créativité Scientifique 2020 » du Dpt. SA)
      - ✓ Partenaires : UMR EPIA, UMR ASTRES, UMR BioEpar

Projet DAPPEM

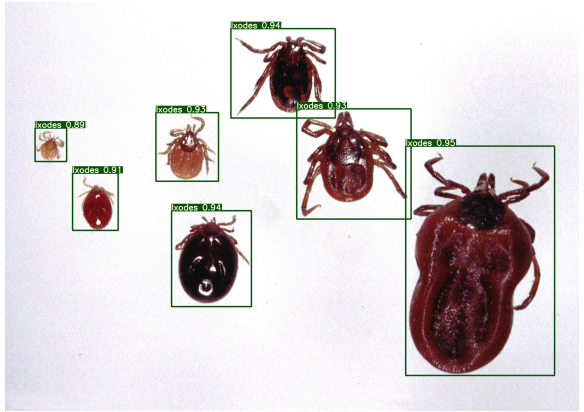
- **Objectif :**
  - Développer un réseau de neurones capable d'identifier des érythèmes migrants à partir d'un photo
  - Intégrer ce réseau de neurones à une application pour smartphone Android / iOS
  - **Public visé** : médecins et population générale

- **Thèse de Doctorat en informatique**
  - Sk Imran HOSSAIN - (Mars 2020 – Mars 2023)
  - Co-encadrement :
    - E. MEPHU-NGUIFO (LIMOS)
    - J. DE GOËR (INRAE EPIA)

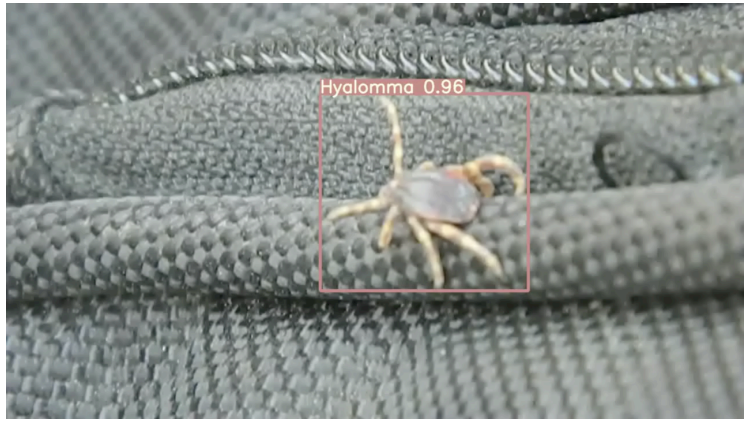


Projet DCLIC

- ✓ **Objectifs :**
  - ✓ Développer un réseau de neurone capable de localiser en temps réel différents genres de tiques (ixodes ricinus, hyalomma, Dermacentor, rhipicephalus) à partir d'une photo ou un flux vidéo
  - ✓ **Quelle précision** :
    - ✓ Localisation précise d'une ou plusieurs tiques
    - ✓ Possibilité de comptage automatique



Détection et identification
à partir de photos



Détection et identification
en temps réel à partir d'un flux vidéo

Mise en œuvre du DeepLearning

- **Quatre étapes principales**
  - Constitution du jeu de données (dataset) d'entrainement
  - Annotation du jeu de données
  - Création, entraînement et évaluation du modèle
  - Intégration au sein d'une application

Mise en œuvre du DeepLearning

Constitution du jeu d'entrainement | Annotation | Entraînement du modèle | Intégration au sein d'une application

- ✓ **Constituer un dataset d'images représentatives de la problématique que l'on souhaite traiter**
  - ✓ Plusieurs centaines d'images sont nécessaires
- ✓ **Utiliser un dataset existant :**
  - ✓ Kaggle : https://www.kaggle.com/datasets
  - ✓ COCO : https://cocodataset.org/
  - ✓ Google : https://opensource.google/projects/open-images-dataset
  - ✓ ImageNet : http://www.image-net.org/update-mar-11-2021.php
- ✓ **Avoir un nombre d'images suffisamment élevé**
  - ✓ **DAPPEM** : 1672 images (52% sont des EM, 48% sont d'autres pathologies de la peau)
    - ✓ 10% : Partenaires du projet, 90% : Internet (Image Scrapping)
  - ✓ **DCLIC** : 1636 images : 391 : Ixodes, 253 : Dermacentor, 170 : Hyalomma, 50 : Rhipicephalus, 772 : autres
    - ✓ 20% : Partenaires du projet, 80% : Internet

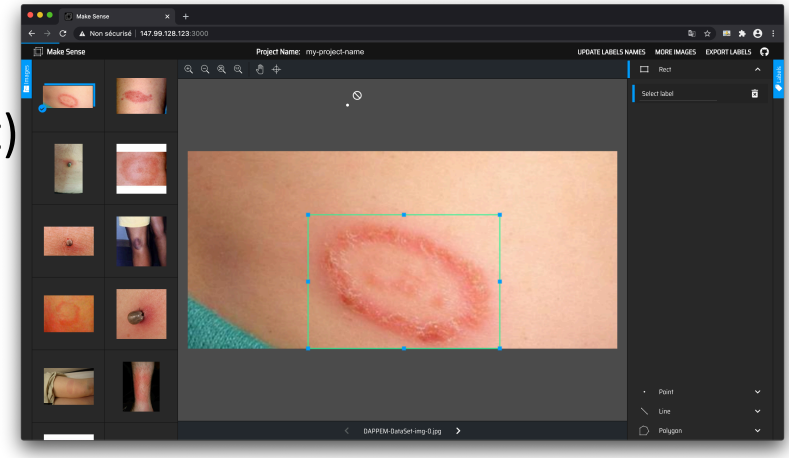
- ✓ **Image Scrapping :**
  - ✓ Plugin Chrome : Fatkun

Mise en œuvre du DeepLearning

Constitution du jeu d'entrainement | **Annotation** | Entraînement du modèle | Intégration au sein d'une application

- ✓ **Organiser les images par classe – avis d'experts**
  - ✓ DAPPEM : 52% sont des EM, 48% sont d'autres pathologies de la peau
    - ✓ Développement d'un outil permettant la classification des photos (N. DAMBRINE)
  - ✓ DCLIC : 391 : Ixodes, 253 : Dermacentor, 170 : Hyalomma, 50 : Rhipicephalus

- ✓ **Annotation :**
  - ✓ Délimiter manuellement, les objets que doit « apprendre » à reconnaître le modèle
  - ✓ Outils :
    - ✓ MakeSens : https://www.makesense.ai (OpenSource)
    - ✓ Microsoft VOTT : https://github.com/microsoft/VoTT (gratuit)
    - ✓ LabelImg : https://github.com/tzutalin/labelImg (gratuit)
    - ✓ RoboFlow : https://roboflow.com (payant)
  - ✓ Étape manuelle est fastidieuse



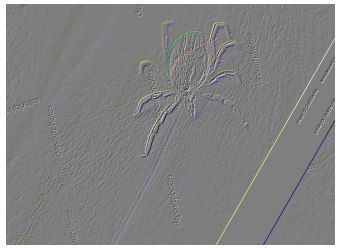
Mise en œuvre du DeepLearning

Constitution du jeu d'entrainement → **Annotation** → Entraînement du modèle → Intégration au sein d'une application

- ✓ **Augmentation de données**
  - ✓ Générer artificiellement des images à partir des images du jeu de données
  - ✓ Utilisation de différentes techniques :
    - ✓ Rotation, contraste, luminosité, contours, netteté, flou…

Image originale

Rotation + contraste



Rotation + contours

Rotation + luminausité

  - ✓ **Outils :**
    - ✓ Rotational Augmentation : https://github.com/whynotw/rotational-data-augmentation-yolo
    - ✓ Librairie Python : imageAug : https://pypi.org/project/ImageAug/

Mise en œuvre du DeepLearning

Constitution du jeu d'entrainement → Annotation → **Entraînement du modèle** → Intégration au sein d'une application

- ✓ **Partage du dataset original (hors images augmentées) :**
  - ✓ 70% pour l'entraînement + les images augmentées
  - ✓ 20% pour la validation
  - ✓ 10% pour les tests + dataset background

- ✓ **Sélection d'une architecture de réseau de neurones**
  - ✓ Construire sa propre architecture : utilisation académique ou de recherche
  - ✓ Utilisation d'une architecture pré-entraînée pour la spécialiser : fine-tuning
    - ✓ Réentraîner un réseau généraliste existant à partir de données spécifiques
    - ✓ Redéfinir les classes de sortie
  - ✓ Utilisation de techniques de Transfert Learning
    - ✓ Utiliser les connaissances d'un réseau déjà existant : freezer certaines couches
      - ✓ DAPPEM : Utilisation d'un modèle de détection des mélanomes

Mise en œuvre du DeepLearning

Constitution du jeu d'entrainement → Annotation → **Entraînement du modèle** → Intégration au sein d'une application

✓ **Principales architectures pour l'analyse d'image disponible pour la communauté :**

VGG	Classification	Very Deep Convolutional Networks for Large-Scale Image Recognition (arXiv 2014)
ResNet	Classification	Deep Residual Learning for Image Recognition (arXiv 2015)
Inception	Classification	Going Deeper with Convolutions (arXiv 2014)
SSD	Détection + localisation	SSD: Single Shot MultiBox Detector (arXiv 2016)
Yolo	Détection + localisation	YOLOv4: Optimal Speed and Accuracy of Object Detection (arXiv 2020)
MaskRCNN	Segmentation	Mask R-CNN (arXiv 2017)
Yolact	Segmentation	YOLACT: Real-time Instance Segmentation

Mise en œuvre du DeepLearning

Constitution du jeu d'entrainement → Annotation → **Entraînement du modèle** → Intégration au sein d'une application

✓ **Principales bibliothèques logicielles :**

TensorFlow Keras	Google	https://www.tensorflow.org/?hl=fr https://keras.io
PyTorch	Fabebook	https://pytorch.org
Darknet	University of Washington	https://pjreddie.com/darknet/
CoreML	Apple	https://developer.apple.com/documentation/coreml
Caffe	University of Berkley	https://caffe.berkeleyvision.org
MSCT	Microsoft Corp.	https://docs.microsoft.com/en-us/cognitive-toolkit/
ONNIX	Microsoft Corp.	https://onnx.ai

Mise en œuvre du DeepLearning

Constitution du jeu d'entrainement | Annotation | **Entraînement du modèle** | Intégration au sein d'une application

- ✓ **Caractéristiques du DATASET DCLIC :**
  - ✓ DATASET d'entraînement :
    - ✓ 864 images de tiques : Ixodes, Dermacentor, Hyalomma, Rhipicephalus
    - ✓ 772 images d'autres insectes : Halyomorpha, Tetranychidae, Philodromidae, Thomisidae
    - ✓ 19 632 images augmentées
  - ✓ DATASET de validation
    - ✓ 328 images de tiques et d'autres classes d'insectes

  DATASET de tests :
    - ✓ 164 images de tiques et d'autres classes d'insectes
    - ✓ 15 375 images du DATASET ArTaxOr – Kaggle
      - ✓ 2 278 Odonata
      - ✓ 2 107 Lepidoptera
      - ✓ 2 049 Hymenoptera
      - ✓ 2 388 Hemiptera
      - ✓ 2 030 Diptera
      - ✓ 2 111 : Coleoptera
      - ✓ 2 419 : Araneae

Mise en œuvre du DeepLearning

Constitution du jeu d'entrainement | Annotation | **Entraînement du modèle** | Intégration au sein d'une application

- **Environnement technique :**
  - **Environnement matériel :**
    - Station de travail : Dell Precision 5820 XCTO
      - CPU : Intel(R) Xeon(R) W-2245 CPU @ 3.90GHz
      - RAM : 32Go DDR4 266Mhz ECC
      - Disque dur : SSD PCIe NVMe M.2 de 2 To
      - Carte graphique : 2 x NVIDIA RTX 8000 48Go de VRAM
      - Prix : 9 700€ HT
  - **Environnement logiciel :**
    - Système d'exploitation : Ubuntu 18.04 LST, NVIDIA 460.56, CUDA 11.2
    - Système de conteneur : Docker CE 20.10.5
    - Python 3.8 + PyTorch 1.8
    - YoloV5.4 (https://github.com/ultralytics/yolov5)

Mise en œuvre du DeepLearning

Constitution du jeu d'entrainement | Annotation | **Entraînement du modèle** | Intégration au sein d'une application

- **Utilisation de ressources de calcul mutualisées**
  - Ressources de calcul des mésocentres
    - Mésocentre de l'UCA : https://mesocentre.uca.fr
    - IDRIS (CNRS) Jan-Zay : http://www.idris.fr
  - **Avantage :**
    - Accès à de grandes puissances de calcul
  - **Inconvénients :**
    - Administratifs et réglementaires
      - Procédures longues
      - Pas de labellisation pour les données de santé
    - Technique :
      - Problèmes de version de librairies
      - Pas de virtualisation / conteneurs

Mise en œuvre du DeepLearning

Constitution du jeu d'entrainement → Annotation → **Entraînement du modèle** → Intégration au sein d'une application

- ✓ **Caractéristiques du modèle YoloV5 DCLIC**
  - ✓ Nombre de couches : 500
  - ✓ Nombre paramètres : 46 652 890
  - ✓ Taille des images en entrée : 640
  - ✓ Nombre d'epochs : 500
  - ✓ Taille des batchs : 16
  - ✓ Mémoire VRAM nécessaire : 22 Go
  - ✓ Mémoire RAM nécessaire : 16 Go
  - ✓ Utilisation GPU : 80% des deux NVIDIA RTX 8000
  - ✓ Utilisation CPU : 8 cœurs à 100%
  - ✓ Temps d'exécution : 45h
  - ✓ Consommation électrique totale : 27 Kw

Mise en œuvre du DeepLearning

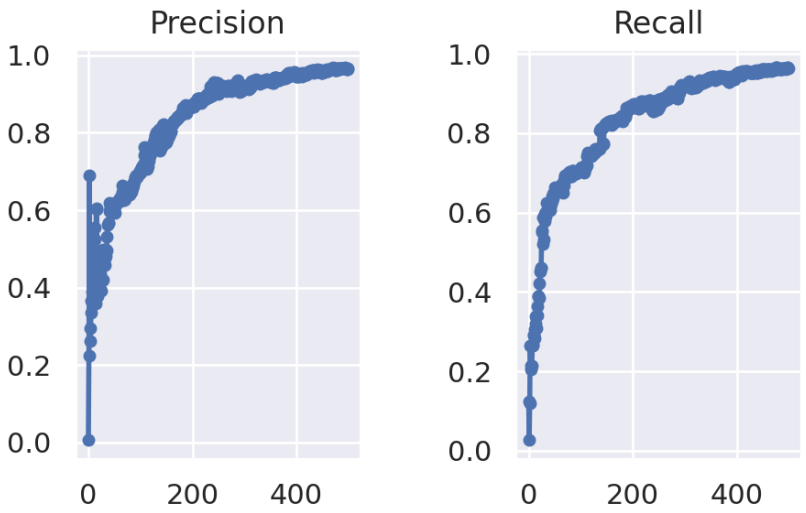
Constitution du jeu d'entrainement → Annotation → **Entraînement du modèle** → Intégration au sein d'une application

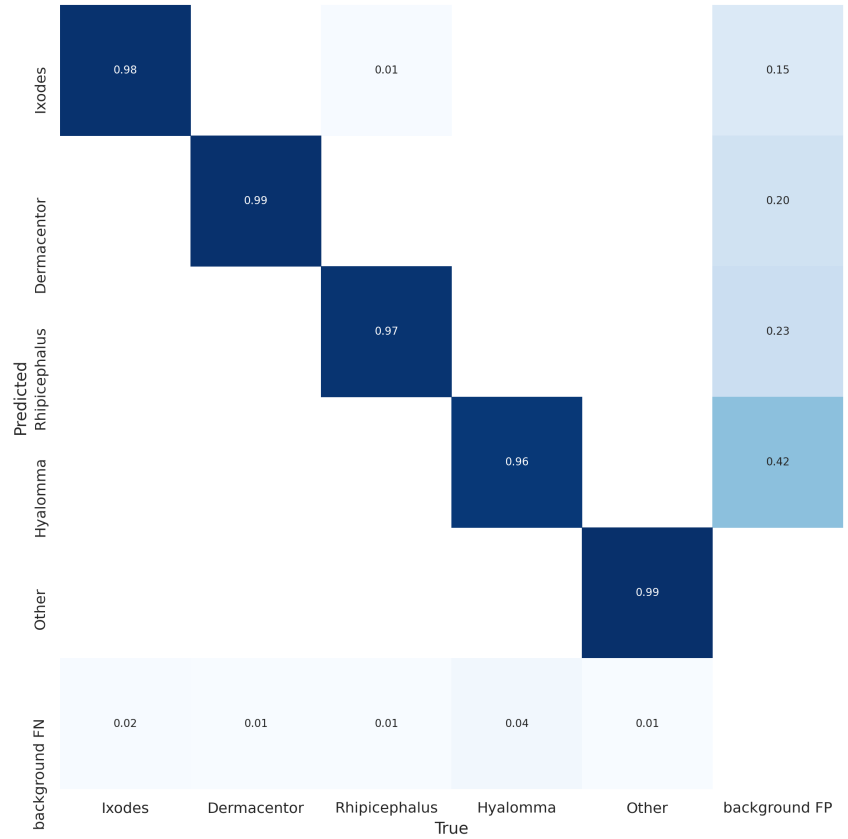
- ✓ **Résultats**
  - ✓ Precision : 0.967
  - ✓ Recall : 0.963
  - ✓ Accuracy : 0.968

$$\text{Precision} = \frac{VP}{VP+FP}$$

$$\text{Recall} = \frac{VP}{VP+FN}$$

$$\text{Accuracy} = \frac{VP+VN}{Total}$$





Matrice de confusion

Mise en œuvre du DeepLearning

Constitution du jeu d'entrainement → Annotation → Entraînement du modèle → **Intégration au sein d'une application**

- ✓ **Puissance de calcul nécessaire**
  - ✓ Négligeable par rapport à la puissance nécessaire pour les phases d'entraînement
    - ✓ Fonctionne depuis un smartphone ou un RaspberryPI : YoloV3 @1-15 fps sur un RaspberryPI 4
    - ✓ Détection en temps réel : YoloV5 @20-60 fps sur un iPhone X
- ✓ **Taille des modèles :** Resnet50 (100 Mo), YoloV5l (100Mo), Mask RCNN (250 Mo)
- ✓ **Intégration au sein d'applications :**
  - ✓ Préparer les images à envoyer au modèle (format, taille…)
  - ✓ Récupérer les valeurs de sorties et les afficher (noms des classes, bounding box, masque de segmentation)
    - ✓ Librairies de traitement d'images (OpenCV)
- ✓ **Librairies d'interrogation des modèles :**
  - ✓ Tensorflow Lite
  - ✓ PyTorch
  - ✓ Swift CoreML
- ✓ **Conversion des modèles :**
  - ✓ ONNX (Open Neural Network Exchange) : https://onnx.ai

Intégration au sein d'une application

Constitution du jeu d'entrainement → Annotation → Entraînement du modèle → **Intégration au sein d'une application**

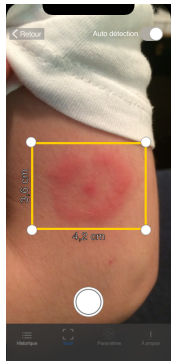
- ✓ **Prendre en compte le contexte**
  - ✓ Des confusions sont possibles :
    - ✓ Liées au contexte
    - ✓ Liées aux limites actuels des modèles

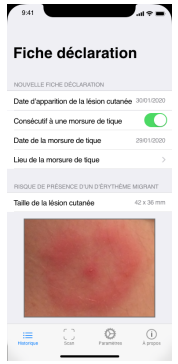
Arbre ou brocoli


Pomme ou chouette


Chiwawa ou muffin

- ✓ **Utilisation de deux modèles au sein de l'application DAPPEM**
  - ✓ YoloV5 : Détection de la peau humaine et des tâches rouges
  - ✓ Resnet50 : Probabilité que l'image contienne un érythème migrant, pondération via un questionnaire médical


Aide à la prise de photos



Fiche de déclaration

Conclusion

- **Conclusion**
  - Le DeepLearning a permis des améliorations significative dans les tâches de traitement et d'analyse d'images
  - Les techniques sont perfectibles, notamment au niveau de l'explicabilité des résultats
  - **La mise œuvre du DeepLearning nécessite :**
    - Quelques connaissances théoriques générale
    - De pouvoir constituer des dataset représentatifs et en quantité suffisante
    - Des connaissances techniques :
      - Langage Python
        - Gestion de données
        - Traitement d'images
    - Des ressources de calcul flexibles et suffisantes

Perspectives :

- **Perspectives**
  - Deux projets :
    - Ethologie des tiques dans un milieu contrôlé
      - Détection et quantification automatisée des déplacements en fonction de certaines zones
      - K. CHALVET, N. DAMBRINE et P. WONGNAK (Doctorant)
    - Extraction et mesure automatisée des caractéristiques morphologies des tiques
      - Mesure des rostres, boucliers
      - X. BAILLY, V. POUX, Y. FRANDO (Stagiaire École Central SupElec)

- **Perspective au sein du CATI IMOTEP**
  - Création d'un groupe de travail autour du DeepLearning
    - Accompagner les personnes ou équipes qui souhaitent utiliser ces familles de méthodes

Deep Learning pour la Vision Numérique
Retour d'expériences en Épidémiologie

Utilisation du Deep Learning dans le cadre de projets autour de la maladie de Lyme

Jocelyn DE GOËR (UMR EPIA)

Webinaire IMOTEP - 01 AVRIL 2021