Para la detección de fauna en un entorno natural o salvaje como puede ser un bosque de nuestra zona, debemos de implantar la suficiente tecnología que certifique con seguridad la visión o detección de una especie de animal concreto. Para ello, es recomendable usar:

* Observación de animales mediante imágenes térmicas: útil para detectar aves y animales terrestres que se puedan camuflar con el entorno debido a su tamaño o color de pelaje sobre todo durante la noche.
* Observación de animales mediante imágenes pasivas: utilizando cámaras con una buena resolución que durante la luz del día puedan capturar diferentes imágenes en un rango de tiempo específico (por ejemplo, capturar imágenes cada 1 -2 minutos o videos de una duración de unos 40 segundos cada 4 o 5 minutos).

Hay que tener en cuenta que los dispositivos a implantar no deben de ser invasivos, para evitar alterar el comportamiento de la fauna.

Al ser un entorno a estudiar grande, recomendaría usar drones con este tipo de cámaras implantadas principalmente para la observación de animales terrestres, ya que, si necesitamos implantar cámaras estáticas en todo el entorno, el gasto de implantación sería bastante mayor.

Una vez obtenidas las imágenes, para implantar el programa de detección de dichas imágenes, usaríamos el lenguaje de programación Python con Deep Learning (subconjunto de Matchine Learning), que nos permite que a partir de una gran cantidad de datos y con un gran procesamiento de algoritmos, conseguimos que el programa reconozca las imágenes de manera automática.

El Deep Learning, se divide en tres capas:

* Input layer (capa de entrada): es la capa donde entran los datos, en la que hay redes neuronales que se activan o no dependiendo de los datos.
* Hidden layer (capa oculta): donde se activa el reconocimiento de algunas líneas o puntos en este caso de la imagen hasta decidir de qué se trata.
* Output layer (capa de salida): determina de que se trata una vez estudiados esos puntos o líneas.

Dentro del Deep learning, deberemos de usar la biblioteca TensorFlow, una de las más utilizadas para el aprendizaje automático.

Para empezar el proyecto, necesitamos:

1. Carga de datos mediante imágenes: con las que el programa pueda empezar a reconocer los distintos animales y poder entrenarle con la finalidad de que cada vez el programa sea más fiable en su detección.

Con la obtención de cantidad de fotos por cada especie que sea posible encontrarse en el entorno, teniendo en cuenta las distintas razas disponibles de cada especie.

Además, almacenamos los datos en tres carpetas distintas: entrenamiento, validación y prueba. En cada carpeta, se crean subcarpetas con el nombre distintivo de cada especie y de cada raza.

**TENGO QUE VER TODAVIA COMO OBTENGO ESAS IMÁGENES CON LAS QUE ENTRENO AL PROGRAMA.**

**ES IMPORTANTE QUE TODAS LAS IMÁGENES USADAS PARA EL ENTRENAMIENTO TENGAN EL MISMO TAMAÑO, YA QUE LA CANTIDAD DE NEURONAS DE ENTRADA EN LA RED VA A SER SIEMPRE LA MISMA (LIBRERÍA CV2)**

**EL TAMAÑO A ELEGIR ES IMPORTANTE, ASÍ COMO EL COLOR, PORQUE CUANTO MAS GRANDES SEAN LAS IMÁGENES Y MAS COLORES TENGA CADA IMAGEN, MAS TARDARÁ LA RED NEURONAL EN RECONOCER LAS IMÁGENES.**

**NECESITAREMOS CONVERTIR LAS IMÁGENES A DISTINTOS FORMATOS PARA REDUCIR EL TIEMPO DE RECONOCMIENTO DE CADA IMAGEN**

**LAS IMÁGENES VAN A TENER PIXELES QUE LAS NEURONAS VAN A ESTUDIAR. SE RECOMIENDA QUE LOS VALORES DE ESTOS PIXELES ESTÉN ENUMERADOS DEL 0 AL 1, Y NO DEL 1 AL 255 (NORMALIZACIÓN CON numpy).**

1. Definimos el tipo de red neuronal a usar: Usaremos la red neuronal convolucional, ya que son muy útiles para la clasificación de imágenes, extrayendo características relevantes de las imágenes, útiles para la clasificación y reconocimiento de imágenes.

Aunque podemos comprobar que red neuronal funciona mejor mediante la herramienta gráfica TensorBoard.

El modelo de datos a usar va a ser de aumento de datos (aumentativo), ya que trae un rendimiento mejor y evitas el sobreajuste de datos a la hora del entrenamiento.

1. Entrenamiento de la red neuronal mediante las imágenes del set:

De todas las imágenes disponibles en el set, dedicaremos una parte de ellas para el entrenamiento y otra parte para las pruebas, siendo las del entrenamiento la mayor parte de ellas, más o menos, el 85% para el entrenamiento y un 15% de ellas para las pruebas (el porcentaje variará dependiendo del volumen de imágenes del set.

El entrenamiento se realizará con el número de épocas más adecuado (número de iteraciones que se realizan en un modelo de aprendizaje automático).

1. Exportamos el modelo entrenado para poder usarlo
2. Inclusión del modelo al programa Python
3. En el programa Python, agregamos la cámara, un tag de video para obtener el video de la página web y un “canvas” para procesar las imágenes que vienen del video hacia el canvas.

Cargamos el modelo para poder hacer predicciones (tensorflowjs).

Redimensionar las dimensiones del canvas a las dimensiones de la imagen con la que el modelo ha entrenado (redimensionando la imagen).

Cuando tengamos un modelo, accedemos a TensorBoard ejecutando en el terminal:  
tensorboard --logdir=path/to/logs  
Y entrando en google poniendo esto:  
http://localhost:6006/

Para una mayor complejidad del proyecto del TFG, a la hora de realizarlo, haremos un apartado de usuarios, en los que en cada usuario se guardará los datos del usuario y de los animales que va encontrando, diciendo el animal que se ha encontrado y la posición GPS en la que el animal ha sido encontrado para cada animal y cada usuario.

El servidor de Django está en ejecución en <http://127.0.0.1:8000/>.

Para entrar en el mio tengo que poner: <http://127.0.0.1:8000/admin>

Y luego las credenciales

CÓDIGO DE APRENDIZAJE AUTOMÁTICO

<https://learn.microsoft.com/es-es/azure/machine-learning/tutorial-auto-train-image-models?view=azureml-api-2&tabs=cli>

VIDEO PROYECTO

<https://www.youtube.com/watch?v=DbwKbsCWPSg>

<https://www.youtube.com/watch?v=EAqb20_4Rdg>