

# NAÏVE BEES: DEEP LEARNING WITH IMAGES

## Descripción guiada del proyecto


¿Puede una máquina distinguir entre una abeja melífera y un abejorro? Ser capaz de identificar especies de abejas a partir de imágenes, aunque es un desafío, permitiría a los investigadores recopilar datos de campo de manera más rápida y efectiva. En este proyecto, creará un modelo de aprendizaje profundo simple que puede detectar automáticamente abejas melíferas y abejorros, luego cargará un modelo previamente entrenado para su evaluación. Utilizará keras , scikit-learn , scikit-image y numpy , entre otras bibliotecas populares de Python.

Este proyecto es la tercera parte de una serie de proyectos que analizan el trabajo con datos de imágenes, la construcción de clasificadores utilizando técnicas tradicionales y el aprovechamiento del poder del aprendizaje profundo para la visión por computadora.

Los requisitos previos recomendados para este proyecto son Aprendizaje profundo avanzado con Keras en Python , Introducción a la visualización de datos con Python , Naïve Bees: carga y procesamiento de imágenes , y Naïve Bees: predice especies a partir de imágenes .

### Tareas del proyecto

1. Importar bibliotecas de Python
2. Cargar etiquetas de imagen
3. Examinar valores RGB en una matriz de imagen
4. Normalizar los datos de la imagen
5. Dividir en conjuntos de entrenamiento, prueba y evaluación
6. Construcción de modelos (parte i)
7. Construcción de modelos (parte ii)
8. Compilar y entrenar modelo
9. Cargar modelo y puntuación previamente entrenados
10. Visualice el historial de entrenamiento del modelo
11. Genera predicciones



My Account

My Progress

My Bookmarks

For Business

Career Tracks

Skill Tracks

Courses

Practice

Projects

Assessments

Live Events


PREMIUM PROJECT

Naïve Bees: Deep Learning with Images

Build a deep learning model that can automatically detect honey bees and bumble bees in images.

Start Project

11 tasks | 2503 participants | 1500 XP



Guided Project Description

Can a machine distinguish between a honey bee and a bumble bee? Being able to identify bee species from images, while challenging, would allow researchers to more quickly and effectively collect field data. In this

TECHNOLOGY  
Python

PROJECT INSTRUCTIONS

Task 11: Instructions

probability greater than 0.5 were assigned class 1, a bumble bee.

Transfer learning was explored in a future Naïve Bees DataCamp project and will be linked here when available.

+1500 XP

Congratulations, you passed all project tasks!

Rate this project to finish...

★★★★★

Finish

Download Notebook

PROJECT: NAÏVE BEES: DEEP LEARNING WITH IMAGES

GUIDED

jupyter notebook (unsaved changes)

File Edit View Insert Cell Kernel Help

Trusted Python 3

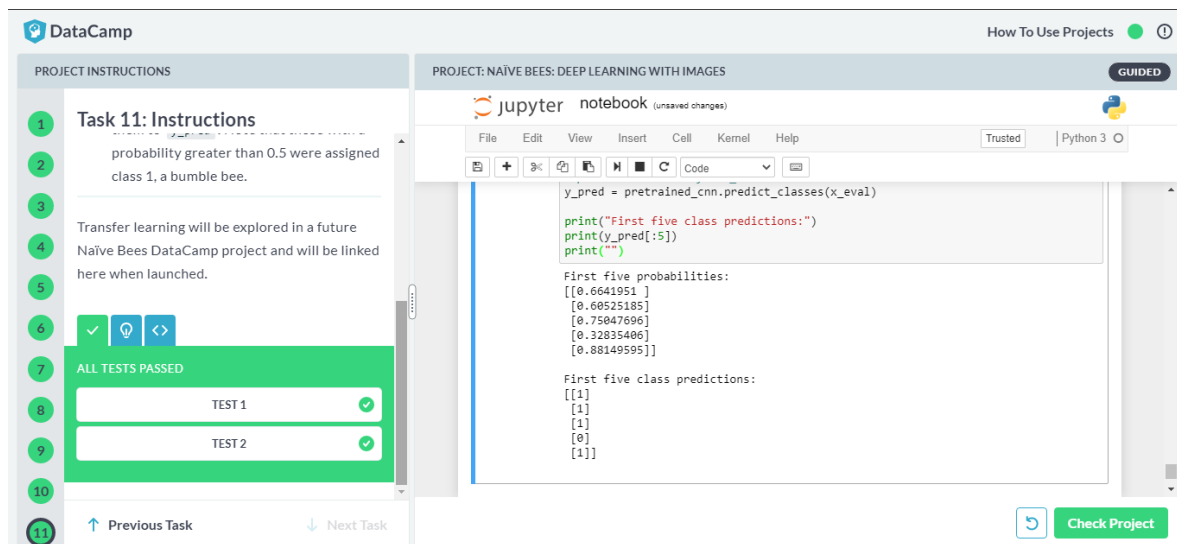
```
y_pred = pretrained_cnn.predict_classes(x_eval)

print("First five class predictions:")
print(y_pred[:5])
print("")

First five probabilities:
[[0.6641951 ]
 [0.60525185]
 [0.75047696]
 [0.32835406]
 [0.88149595]]

First five class predictions:
[[1]
 [1]
 [1]
 [0]
 [1]]
```

Check Project



## Tarea 1 : Instrucciones

Importe las bibliotecas de Python con las que trabajará.

- Importar `keras`, la biblioteca de aprendizaje profundo que utilizará.
- Importe la función `Sequential` del `models` módulo de `keras`. Este es el tipo de modelo que utilizará.
- Importar las funciones `Dense`, `Dropout`, `Flatten`, `Conv2D`, `MaxPooling2D` desde el `layers` módulo de `keras`. Estos formarán las diferentes capas de su red neuronal convolucional.

## Tarea 2 : Instrucciones

Cargue el DataFrame de etiquetas y nombres de imágenes, explore el conjunto de datos y luego asigne etiquetas de imagen a `y`.

- Usando la `read_csv` función de pandas, cargue el `labels.csv` archivo que vive en la `datasets` carpeta. Asegúrese de configurarlo `index_col=0` para que los nombres de las imágenes se carguen como índice.
- Imprima los recuentos de valores de `genus` en el `labels` DataFrame para mostrar que el conjunto de datos está perfectamente equilibrado entre las dos clases.
- Asigne los `genus` valores de columna (la matriz de etiquetas de imagen) a `y`.

## Tarea 3 : Instrucciones

Cargue la primera imagen de su DataFrame y explore su forma y valores RGB.

- Cargue la primera imagen del índice de DataFrame de etiquetas usando `io.imread`, la función de carga de imagen de scikit-image y asígnela a `example_image`.
- Muestre `example_image` usando `plt.imshow()`.
- Imprima la forma de `example_image` para ver que tiene 50 por 50 píxeles y tiene 3 canales.
- Imprima los valores R, G y B para el píxel superior izquierdo del `example_image`. Recuerde que la imagen tiene forma (X, Y, Z) y que las coordenadas X e Y de este píxel son (0, 0).

## Tarea 4 : Instrucciones

Normalice cada característica (es decir, canal) de cada imagen iterando sobre los canales de cada imagen en un bucle for. Luego, apile las matrices resultantes en una sola matriz y asígnela a X.

- Asignar el `StandardScaler()` objeto a `ss`.
- Dentro del bucle for que recorre cada canal de una imagen, llame `ss.fit_transform` a cada canal.
- Se utiliza `np.array()` para apilar `image_list` (una lista de matrices de imágenes normalizadas) en una matriz que contiene todas las imágenes del conjunto de datos.
- Imprima la forma de `X`.

## Tarea 5 : Instrucciones

Divida los datos en conjuntos de entrenamiento, prueba y evaluación.

- Úselo `train_test_split` para dividir el 20% de los datos `X` y `y` en conjuntos de validación estableciendo `test_size` igual a 0.2.

- Dividir los datos restantes (`x_interim` y `y_interim`) en `x_train`, `x_test`, `y_train`, `y_test` de manera que 40% de los datos entra en el conjunto de prueba. Asegúrese de configurarlo `random_state=52` para garantizar resultados consistentes.
- Imprime la forma de `x_train`.
- Imprima el número de muestras en formato `x_train`.

## Tarea 6 : Instrucciones

Especifique el número de clases en el modelo y luego cree las dos primeras capas de la red neuronal.

- Establecer `num_classes` igual a 1. Dado que su modelo intenta predecir si una imagen es de un abejorro o una abeja melífera (es decir, no un abejorro), puede enmarcar esto como un problema de clasificación binaria.
- Definir `model` como `Sequential()` para inicializar el modelo.
- Ya hay una capa convolucional 2D en el código con 32 filtros. Agregue otra [capa convolucional 2D](#) (`Conv2D`), esta vez con 64 filtros. Se debe tener el mismo `kernel_size` y `activation` especificaciones como la primera capa convolucional 2D.

La forma de entrada en la primera capa se refiere a las dimensiones de las imágenes que se pasan. No es necesario especificar la forma de entrada después de la primera capa, ya que las siguientes capas pueden hacer inferencias de forma automática

## Tarea 7 : Instrucciones

Termine de construir la CNN y luego imprima el resumen del modelo.

- Agrega una `MaxPooling2D` capa con `pool_size=(2, 2)`.
- Agregue una segunda `Dropout` capa con una tasa de 0.5.
- Para la `Dense` capa final que genera predicciones, pase su especificado previamente `num_classes` como el primer parámetro. Establecer `sigmoid` como función de activación ya que se trata de un problema de clasificación binaria.
- Muestre el resumen del modelo usando `model.summary()`.

---

Las capas convolucionales comparten pesos entre píxeles, a diferencia de las capas completamente

## Tarea 8 : Instrucciones

Compile el modelo y entrenelo de forma simulada en un subconjunto de los datos durante cinco épocas.

- En la función de compilación, establezca `keras.losses.binary_crossentropy` como pérdida y establezca `accuracy` como métrica.
- En la función de ajuste que entrena el modelo, establezca épocas iguales a 5.

## Tarea 9 : Instrucciones

Cargue el modelo previamente entrenado y calcule la pérdida y la precisión del conjunto de reserva.

- Cargue el modelo preentrenado usando `load_model` desde el `keras.models` módulo y asígnelo a `pretrained_cnn`.
- Evalúe el modelo previamente entrenado en el conjunto reservado (`x_eval` y `y_eval`) usando el `.evaluate` método. Asigne la tupla resultante a `eval_score`.
- Imprima la pérdida para el conjunto de reserva.
- Imprima la precisión del conjunto reservado.

---

Puede ver que el modelo se generaliza bastante bien, ya que la precisión es similar para el conjunto de prueba y el conjunto de reserva: 0,66 para los datos que el modelo ha visto (el conjunto de prueba) y 0,65 para los datos que el modelo no ha visto (el conjunto de reserva).

*Nota: el kernel de este portátil puede morir si carga el modelo previamente entrenado demasiadas veces. En ese caso, guarde el cuaderno y vuelva a cargar el*

*proyecto. Si aún tiene problemas, siempre puede hacer clic en la flecha de reinicio del proyecto junto al botón "Comprobar proyecto" (advertencia: perderá todo su código después de hacer clic en este botón).*

## Tarea 10 : Instrucciones

Cargue el historial del modelo y trace la precisión de la validación y la pérdida durante el período de entrenamiento.

- Imprime las claves del `pretrained_cnn_history` diccionario. Esto muestra que puede acceder a las puntuaciones de pérdida y precisión para el conjunto de prueba y el conjunto de reserva.
- Trace la precisión de la validación con `pretrained_cnn_history['val_acc']`.
- Trace la pérdida de validación.

## Tarea 11 : Instrucciones

Finalmente, obtendrá las probabilidades y clases predichas para cada imagen en el conjunto de validación.

- Utilice el `predict` método de nuestro `pretrained_cnn` modelo para obtener las probabilidades `x_eval` y asignarlas `y_proba`.
- Imprima las primeras cinco probabilidades de la `y_proba` lista que acaba de crear.
- Utilice el `predict_classes` método de su `pretrained_cnn` modelo para obtener las clases predichas `x_eval` y asignarlas `y_pred`. Tenga en cuenta que a aquellos con una probabilidad superior a 0,5 se les asignó la clase 1, un abejorro.

---

El aprendizaje por transferencia se explorará en un futuro proyecto de Naïve Bees DataCamp y se vinculará aquí cuando se lance.

Learn R, Python & Data Science | (28) CENTRO MUSICAL POMAPA | Proyectos | Colors RGB

projects.datacamp.com/projects/555

Aplicaciones | Bookmarks | Jooble - Asistente c... | Kung Fu Panda 2 (2... | Esquema de un pro... | TARJETA DIGITAL | Cursos online | Ud... | MP3 Download | Ministerio de Trabaj...

DataCamp

Cómo Utilizar Proyectos

INSTRUCCIONES DEL PROYECTO

Tarea 6: Instrucciones

de entrada desde la primera capa, ya que las siguientes capas no hacen inferencias de forma automática.

+ 1500 XP

¡Felicitaciones, aprobó todas las tareas del proyecto!  
Califica este proyecto para terminar ...

Terminar

Descargar cuaderno

PROYECTO: ABEJAS INGENUAS: APRENDIZAJE PROFUNDO CON IMÁGENES

GUIADO

jupyter notebook (unsaved changes)

File Edit View Insert Cell Kernel Help Trusted Python 3

In [95]:

```
# predicted probabilities for x_eval
y_proba = pretrained_cnn.predict(x_eval)

print("First five probabilities:")
print(y_proba[:5])
print("")

# predicted classes for x_eval
y_pred = pretrained_cnn.predict_classes(x_eval)

print("First five class predictions:")
print(y_pred[:5])
print("")

First five probabilities:
[[0.6641951 ]
 [0.60525185]
 [0.75047696]
 [0.32835406]
```

Ver Proyecto

project (2).zip | project (1).zip | project.zip

Mostrar todo

Escribe aquí para buscar

03:45 9/09/2020