Reconocimiento de ASL con aprendizaje profundo

Construya una red neuronal convolucional para clasificar imágenes de letras del lenguaje de señas estadounidense.

Descripción guiada del proyecto

El lenguaje de señas americano (ASL) es el idioma principal utilizado por muchas personas sordas en América del Norte, y también lo utilizan las personas con problemas de audición y audición. El lenguaje es tan rico como los lenguajes hablados y emplea signos hechos con la mano, junto con gestos faciales y posturas corporales.

En este proyecto, entrenará una red neuronal convolucional para clasificar imágenes de letras ASL. Después de cargar, examinar y preprocesar los datos, entrenará la red y probará su rendimiento.

Los requisitos previos recomendados para este proyecto son Introducción al aprendizaje profundo en Python y Procesamiento de imágenes con Keras en Python .

Tareas del proyecto

1. Lenguaje de señas americano (ASL)

2. Visualiza los datos de entrenamiento
3. Examinar el conjunto de datos
4. Codificar los datos en un solo uso
5. Definir el modelo
6. Compila el modelo
7. Entrena el modelo
8. Prueba el modelo
9. Visualiza errors

Tarea 1 : Instrucciones

Primero, cargue el conjunto de datos usando un archivo auxiliar especial (`sign_language.py`).

- Ejecute la celda de código tal como está, sin modificaciones.

Tarea 2 : Instrucciones

Úselo `matplotlib`para visualizar algunas de las imágenes en el conjunto de datos de entrenamiento.

- Asignar `labels`a una lista de Python con tres elementos: `'A'`, `'B'`y `'C'`que corresponden a las cartas firmadas que aparecen en las imágenes.

Tarea 3 : Instrucciones

Cuente el número de apariciones de cada letra en el tren y pruebe los conjuntos de datos.

- Asignar la variable `num_B_train`a un número entero contando el número de veces que `1`aparece en`y_train.`
- Asignar la variable `num_C_train`a un número entero contando el número de veces que `2`aparece en`y_train.`
- Asignar la variable `num_B_test`a un número entero contando el número de veces que `1`aparece en`y_test.`
- Asignar la variable `num_C_test`a un número entero contando el número de veces que `2`aparece en`y_test.`

Tarea 4 : Instrucciones

Utilice la función Keras incorporada `to_categorical`para codificar los datos en un solo uso.

- Utilice el vector de clase `y_train`para asignar la variable `y_train_OH`a las etiquetas de entrenamiento one-hot.
- Utilice el vector de clase `y_test`para asignar la variable `y_test_OH`a las etiquetas de prueba one-hot.

Enlaces Útiles:

- **Documentación de la**`to_categorical` función de Keras

Tarea 5 : Instrucciones

Especifique una red neuronal convolucional en Keras.

- La primera capa convolucional de la red ya se ha proporcionado en el código. Agregue una capa de agrupación máxima (agrupación sobre ventanas de tamaño 4x4).
- Agregue otra capa convolucional (15 filtros, tamaño de kernel de 5, `same`relleno, `relu`activación).
- Agregue otra capa de agrupación máxima (agrupación sobre ventanas de tamaño 4x4).

Enlaces Útiles:

- **Documentación de** Keras `Conv2D`
- **Ejercicio de** agrupación de capas de Keras en el curso Redes neuronales convolucionales para procesamiento de imágenes

Tarea 6 : Instrucciones

Especifique el optimizador y la función de pérdida, junto con una métrica que se rastreará durante el entrenamiento.

- Compile el modelo con el `'rmsprop'` optimizador, `'categorical_crossentropy'` como función de pérdida y `'accuracy'` como métrica.

Enlaces Útiles:

- Compile un **ejercicio de** redes neuronales en el curso Redes neuronales convolucionales para procesamiento de imágenes

Tarea 7 : Instrucciones

Utilice el `.fit()` método del modelo para entrenar el modelo.

- Usa los datos de entrenamiento `x_train` y los datos de etiqueta `y_train_OH` para entrenar el modelo durante dos épocas. Debe alcanzar una precisión de validación final de al menos el 80%. (Tenga en cuenta que debe obtener buenos resultados si reserva el 20% de los datos de entrenamiento para usarlos como datos de validación y usa un tamaño de lote de 32).

Enlaces Útiles:

- **Ejercicio de** adaptación de una red neuronal a datos de ropa en el curso Redes neuronales convolucionales para procesamiento de imágenes

Tarea 8 : Instrucciones

Utilice el `.evaluate()` método del modelo para evaluar el modelo.

- Asignar `x` a los datos de prueba en `x_test` y asignar `y` a las etiquetas de prueba en `y_test_OH`.

Enlaces Útiles:

- **Documentación del** `.evaluate()` método Keras

Tarea 9 : Instrucciones

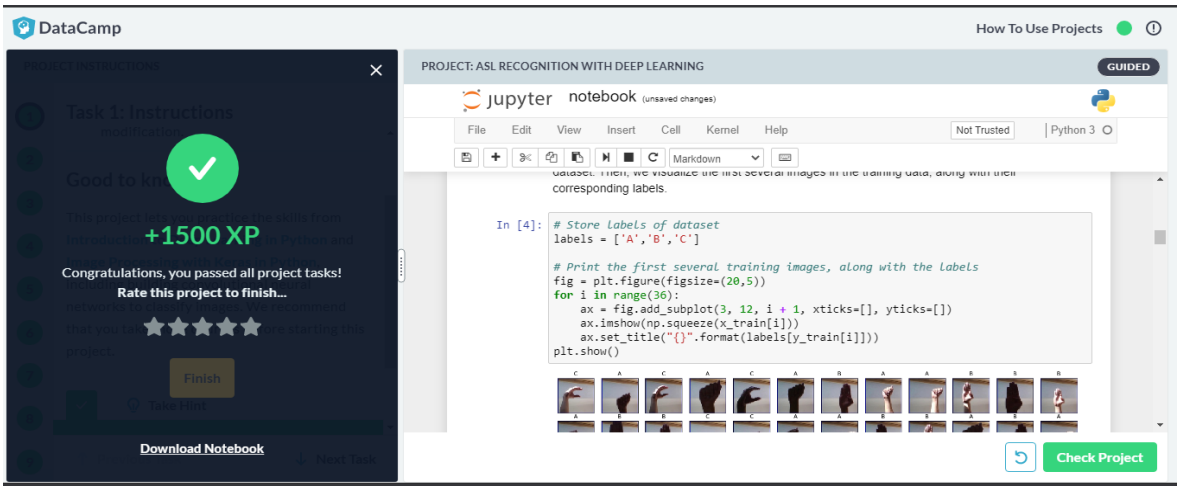
Visualice imágenes que fueron clasificadas incorrectamente por el modelo.

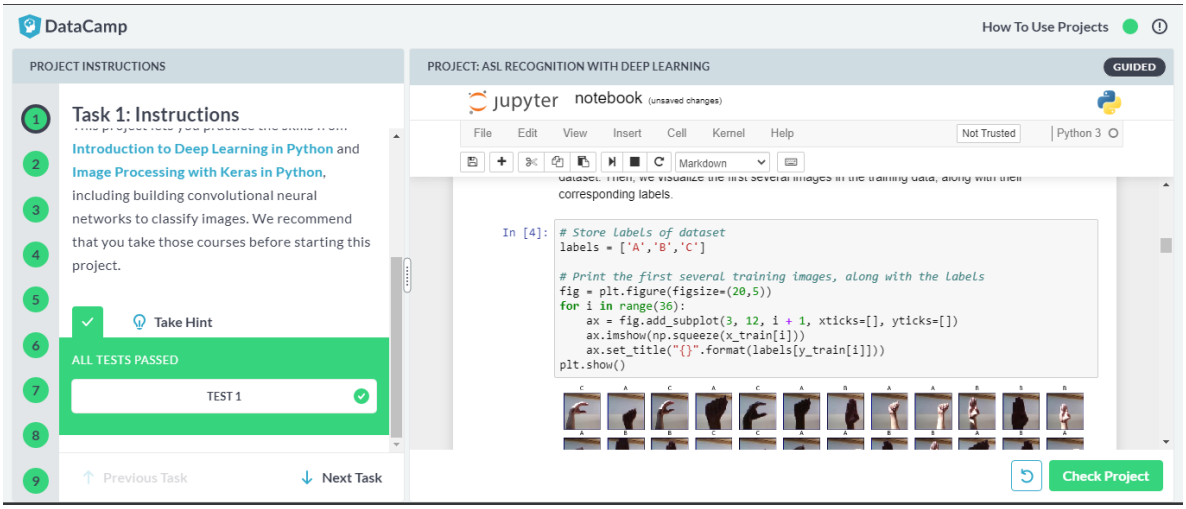
- Utilice el `.predict()` método del modelo para asignar `y_probs` a una matriz numerosa con una forma que `(600, 3)` contenga las probabilidades predichas del modelo de que cada imagen pertenezca a cada clase de imagen.
- Asignar `y_preds` a las etiquetas previstas del modelo para cada imagen en el conjunto de datos de prueba. Tenga en cuenta que `y_preds` debe ser un array numpy con forma `(600,)`, donde cada entrada es uno de `0`, `1` o `2`, lo que corresponde a `'A'`, `'B'` y `'C'`, respectivamente.
- Use las etiquetas de verdad del terreno para el conjunto de datos de prueba (`y_test`) y las etiquetas predichas del modelo (`y_preds`) para determinar qué imágenes se clasificaron incorrectamente. Asigne la variable `bad_test_idxs` a una matriz numérica unidimensional que contenga todos los índices correspondientes a imágenes que fueron clasificadas incorrectamente por el modelo.

Enlaces Útiles:

- Documentación del `.predict()` método Keras

Imágenes de concluir el Proyecto Guiado




DataCamp
How To Use Projects
PROJECT INSTRUCTIONS
PROJECT: ASL RECOGNITION WITH DEEP LEARNING
GUIDED
Task 1: Instructions
Introduction to Deep Learning in Python and
Image Processing with Keras in Python,
including building convolutional neural
networks to classify images. We recommend
that you take those courses before starting this
project.
Take Hint
ALL TESTS PASSED
TEST 1
Previous Task
Next Task
jupyter notebook (unsaved changes)
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3
Markdown
corresponding labels.
In [4]:
# Store labels of dataset
labels = ['A','B','C']
# Print the first several training images, along with the labels
fig = plt.figure(figsize=(20,5))
for i in range(36):
ax = fig.add_subplot(3, 12, i + 1, xticks=[], yticks=[])
ax.imshow(np.squeeze(x_train[i]))
ax.set_title("{}".format(labels[y_train[i]]))
plt.show()
Check Project