Task 2

A convolutional neural network (CNN) is a type of artificial neural network (ANN) specifically designed for processing and analyzing structured grid-like data, such as images, audio, and time-series data. CNNs are inspired by the organization and functionality of the visual cortex in animals, particularly the receptive fields of neurons and the hierarchical processing of visual information.

CNNs consist of multiple layers, each serving a specific purpose in extracting and learning features from input data. The key layers in a typical CNN architecture include: convolutional layers, activation layers, pooling layers, fully connected layers, and dropout layers.

CNN architectures may also include other components such as normalization layers, skip connections, and residual connections, depending on the specific application and design requirements.

CNNs have revolutionized various fields such as computer vision, natural language processing, and speech recognition. They have been successfully applied to tasks such as image classification, object detection, facial recognition, medical image analysis, and autonomous driving, among others.

Untitled.ipynb | Untitled1.ipynb | malware2.csv | malware.csv

Delimiter: ,

	hash	GetProcAddress	ExitProcess	WriteFile	GetLastError	CloseHandle	FreeLibrary	Sleep
1	2e186e57548cd4c703a5d	1	1	1	1	1	1	1
2	aae939f25a8e0d63dd523	1	1	1	1	1	1	1
3	75e1c6d5f25e748663076	1	1	1	1	1	0	1
4	)ea61297ea624ae198067	1	1	1	1	0	0	1
5	7facf29172f32df6bc77f48	1	1	1	1	1	1	1
6	)e606e49da90fee2252f52	0	0	0	0	0	0	0
7	)9d807b3f5f4cc00b1e9b7	1	1	1	1	1	1	1
8	53a6b70c6951a3245dc4a	1	1	1	1	1	1	1
9	24164184460f8523ed7e2	1	1	1	1	1	1	1
10	1a14eb119982d6ec793d7	1	1	1	1	1	1	0
11	246a8707a1ac1fd1e2e36	1	1	1	1	1	1	1
12	c1fa36a97fdc732f51a637	1	1	1	1	1	0	1
13	5061647779 41faf953094	1	1	1	1	1	1	0
14	a698e38af5cca54bdfbdd0	1	1	1	1	1	0	1
15	a17054a9a65028a0a80f8	1	1	1	1	1	1	1
16	3a81926aef367530a2a9fc	1	1	1	1	1	1	1
17	e5af2d2834677173e454b	1	1	0	0	0	0	0
18	c5d5370e381042f973ea5	1	1	1	1	1	1	1
19	3a9fd5c22cad19b21dfb30	1	1	1	1	1	1	1
20	184f62f5734a0c1d140e89	1	1	1	1	1	1	1
21	28d22bf1dd003af2eb36d	1	1	1	1	1	1	1
22	1d5ba1ae0bd681b01bfec	1	1	1	1	1	0	1
23	d546b4b52e45ca031e5c5	1	1	1	1	1	1	1
24	cab26c02c8bec3e5989de	1	1	1	1	1	1	0
25	2d49221b28ed5765fb773	1	1	0	0	0	0	0
26	a42fb819f1cdbf2239c1db	1	1	1	1	1	0	1
27	2d387620d0c194ade6673	1	1	1	1	1	1	1
28	)d274f1c7376fb341b2848	0	0	0	0	0	0	0

Importing required libraries

import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

```
from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from keras.utils import to_categorical

# Load the dataset
data = pd.read_csv("malware2.csv")

# Preprocessing
X = data.drop(columns=['malware'])
y = data['malware']

# Encoding the target variable
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Reshaping the data for CNN input (assuming the features are already in the form of image-like matrices)
X_train = X_train.values.reshape(-1, 32, 32, 1)
X_test = X_test.values.reshape(-1, 32, 32, 1)

# Normalizing the pixel values
X_train = X_train / 255.0
X_test = X_test / 255.0

# One-hot encoding the target variable
y_train = to_categorical(y_train)
```

```
y_test = to_categorical(y_test)

# Creating the CNN model
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(32, 32, 1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(2, activation='softmax'))

# Compiling the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Training the model
model.fit(X_train, y_train, epochs=10, batch_size=128, validation_data=(X_test, y_test))

# Evaluating the model
loss, accuracy = model.evaluate(X_test, y_test)
print("Accuracy:", accuracy)
```