

Task 2 (10 points)

Describe the convolutional neural network. The answer should be comprehensive. (5 points)

The practical example of application in cybersecurity with data and the python code are mandatory. (5 points)

Place your answer into the task2.pdf file. The source code and the data should be included in the resulting pdf file. No additional files are required.

Convolutional Neural Network (CNN):

A Convolutional Neural Network (CNN) is a type of deep neural network that is primarily used for analyzing visual imagery. It is composed of multiple layers of neurons that automatically and adaptively learn spatial hierarchies of features from the input data. CNNs are designed to recognize patterns directly from pixel images with minimal preprocessing.

Key Components:

Convolutional Layers:

Convolutional layers are the building blocks of CNNs. They consist of filters (also known as kernels) that slide over the input data, performing element-wise multiplication and summing up the results to produce feature maps.

Each filter captures different features such as edges, textures, or patterns. As the network learns, these filters are adjusted to recognize more complex features.

Pooling Layers:

Pooling layers follow convolutional layers and down-sample the feature maps. They reduce the spatial dimensions (width and height) of the input, thus decreasing the computational complexity and controlling overfitting.

Max pooling and average pooling are the most common pooling techniques. Max pooling retains the maximum value from each patch of the feature map, while average pooling calculates the average.

Activation Functions:

Activation functions like ReLU (Rectified Linear Unit) are applied after each convolutional and pooling layer to introduce non-linearity into the network, allowing it to learn complex patterns and relationships in the data.

Fully Connected Layers:

Fully connected layers are typically placed at the end of the network. They take the high-level features from the convolutional layers and transform them into class scores.

These layers perform classification or regression tasks based on the learned features.

additional details on Convolutional Neural Networks (CNNs):

Architecture:

- CNNs consist of multiple convolutional layers, each followed by a non-linear activation function and a pooling layer.
- These layers are stacked to form the convolutional base of the network.

- The convolutional base extracts increasingly complex and abstract features from the input image.
- The output of the convolutional base is then flattened and fed into one or more fully connected layers, which perform the final classification or regression task.

Training Process:

- CNNs are trained using backpropagation, similar to feed-forward neural networks.
- During training, the input images and their corresponding labels are fed into the network.
- The output of the network is compared to the true labels, and the error is calculated using a loss function (e.g., cross-entropy for classification, mean squared error for regression).
- The error is then backpropagated through the network, and the weights of the convolutional filters and fully connected layers are updated using an optimization algorithm (e.g., stochastic gradient descent).

Advantages and Applications:

- CNNs are particularly well-suited for image and video data due to their ability to capture spatial and temporal dependencies.
- They have achieved state-of-the-art performance in various computer vision tasks, such as image classification, object detection, semantic segmentation, and facial recognition.
- CNNs are also used in natural language processing, speech recognition, and other domains where the input data has a grid-like structure.

Advanced CNN Architectures:

- Over the years, numerous CNN architectures have been developed, such as AlexNet, VGGNet, GoogLeNet, ResNet, and DenseNet, each with its own unique design and performance characteristics.
- These architectures incorporate various techniques, such as skip connections, residual blocks, and dense connections, to improve the training and performance of CNNs on complex tasks.

CNNs have revolutionized the field of computer vision and have played a pivotal role in the success of deep learning in various domains. Their ability to learn hierarchical representations from raw data has made them an essential tool in many real-world applications.

Code for create randome data:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Generating synthetic data for malware detection
np.random.seed(42)

# Generate 25 data points
n = 25

# Generate synthetic image data (random pixel values representing grayscale images)
```

```

image_size = 32 # Assuming images are 32x32 pixels
num_channels = 1 # Grayscale images
X_images = np.random.randint(0, 256, size=(n, image_size, image_size, num_channels),
dtype=np.uint8)

# Generate labels (0: benign, 1: malicious)
y = np.random.randint(0, 2, n)

# Flatten images
X_images_flat = X_images.reshape(n, -1) # Flatten each image to a 1D array

# Save flattened images and labels to CSV with headers
data = {'image': X_images_flat.tolist(), 'label': y}
df = pd.DataFrame(data)
df.to_csv('malware_data.csv', index=False)

```

data:

image label

```

[61, 197, 42, 40, 196, 207, 233, 186, 209, 160, 147, 242, 143, 142, 34, 99, 135, 53, 80, 196, 88, 207,
206, 200, 95, 120, 174, 68, 125, 115, 46, 126, 89, 247, 62, 151, 243, 202, 77, 46, 85, 228, 176, 184,
158, 89, 25, 188, 234, 167, 247, 240, 137, 129, 140, 64, 183, 218, 178, 225, 120, 190, 165, 125, 75,
163, 227, 131, 244, 37, 86, 19, 91, 162, 79, 198, 69, 130, 44, 158, 250, 183, 232, 143, 147, 167, 117,
170, 132, 0, 126, 141, 22, 203, 240, 96, 136, 186, 95, 29, 57, 13, 85, 244, 186, 142, 74, 32, 220, 136,
201, 223, 207, 131, 143, 135, 181, 157, 59, 181, 45, 28, 212, 68, 21, 231, 177, 182, 191, 172, 98, 5, 7,
102, 157, 64, 122, 253, 11, 185, 19, 11, 37, 168, 178, 10, 81, 8, 77, 165, 189, 50, 38, 16, 100, 152, 36,
101, 221, 36, 177, 154, 8, 138, 161, 32, 221, 255, 74, 61, 4, 248, 117, 7, 84, 159, 103, 100, 245, 123,
160, 56, 85, 110, 30, 47, 72, 16, 122, 38, 197, 73, 53, 108, 59, 141, 171, 135, 154, 154, 33, 20, 82, 215,
241, 40, 178, 12, 198, 131, 49, 255, 121, 3, 61, 24, 112, 248, 42, 110, 245, 75, 105, 8, 61, 25, 255, 81,
22, 152, 201, 127, 45, 110, 26, 232, 55, 91, 193, 153, 134, 62, 245, 48, 117, 172, 254, 15, 216, 191,
140, 157, 188, 42, 162, 184, 130, 230, 220, 103, 250, 43, 251, 58, 165, 95, 132, 10, 211, 92, 105, 175,
161, 229, 231, 88, 112, 156, 93, 251, 43, 4, 230, 67, 127, 80, 236, 176, 107, 171, 163, 175, 157, 240,
24, 226, 16, 239, 227, 242, 58, 35, 189, 242, 137, 189, 158, 68, 98, 183, 155, 124, 83, 59, 188, 207,
255, 145, 203, 179, 145, 15, 44, 14, 44, 168, 26, 218, 107, 76, 194, 137, 218, 184, 74, 91, 186, 145, 15,
18, 61, 239, 234, 66, 58, 201, 28, 239, 176, 72, 84, 242, 227, 152, 47, 196, 190, 6, 239, 185, 214, 103,
111, 168, 159, 22, 61, 56, 137, 223, 234, 210, 10, 33, 247, 114, 101, 219, 195, 165, 69, 67, 228, 203,
122, 189, 12, 248, 232, 81, 12, 140, 189, 242, 197, 182, 174, 66, 99, 166, 33, 109, 182, 236, 227, 243,
246, 192, 100, 193, 98, 132, 144, 254, 59, 159, 231, 238, 247, 29, 103, 163, 66, 138, 181, 167, 54, 135,
203, 184, 230, 191, 212, 188, 122, 32, 39, 146, 65, 186, 18, 167, 173, 151, 66, 81, 63, 161, 81, 170,
132, 123, 180, 40, 27, 92, 242, 25, 97, 159, 63, 77, 227, 103, 82, 112, 146, 2, 46, 169, 110, 248, 5, 17,
225, 49, 15, 83, 197, 28, 199, 212, 199, 125, 239, 188, 188, 68, 145, 51, 84, 243, 179, 10, 14, 126, 175,
51, 52, 98, 170, 234, 51, 214, 38, 165, 163, 108, 255, 165, 230, 41, 176, 82, 246, 182, 112, 127, 230,
70, 139, 41, 146, 154, 236, 38, 216, 16, 137, 108, 134, 76, 70, 115, 187, 180, 109, 67, 43, 105, 54, 75,
251, 60, 128, 44, 187, 55, 48, 48, 128, 99, 163, 46, 77, 247, 20, 107, 150, 113, 36, 65, 113, 33, 115, 38,
45, 228, 23, 178, 48, 95, 54, 241, 76, 105, 24, 98, 157, 6, 134, 119, 45, 37, 126, 138, 198, 56, 233, 128,
57, 134, 208, 244, 204, 199, 92, 106, 132, 246, 93, 56, 128, 51, 70, 73, 145, 155, 224, 8, 127, 20, 198,
67, 191, 251, 121, 195, 117, 121, 23, 207, 82, 215, 10, 88, 141, 2, 72, 13, 53, 9, 11, 60, 179, 21, 34, 224,

```

140, 168, 190, 145, 192, 206, 195, 141, 105, 73, 227, 70, 127, 197, 219, 12, 67, 143, 184, 137, 70, 234, 140, 42, 131, 71, 22, 134, 193, 218, 82, 146, 79, 89, 233, 151, 228, 174, 228, 200, 84, 236, 166, 175, 101, 13, 87, 145, 230, 52, 79, 196, 104, 200, 96, 157, 197, 223, 139, 101, 115, 216, 33, 21, 204, 7, 198, 165, 144, 204, 74, 175, 67, 23, 158, 174, 43, 130, 220, 251, 203, 218, 127, 116, 159, 59, 189, 30, 108, 161, 140, 139, 103, 216, 30, 222, 74, 191, 129, 87, 39, 186, 102, 101, 156, 53, 14, 42, 19, 251, 56, 77, 126, 137, 74, 120, 219, 39, 33, 102, 186, 188, 102, 39, 197, 57, 195, 168, 151, 160, 74, 103, 56, 54, 28, 64, 193, 120, 193, 226, 227, 202, 252, 99, 180, 245, 15, 40, 217, 47, 166, 165, 24, 173, 76, 158, 16, 17, 36, 91, 98, 232, 80, 174, 161, 255, 148, 61, 210, 186, 171, 107, 90, 6, 123, 86, 71, 188, 18, 142, 210, 213, 198, 19, 124, 20, 101, 174, 136, 93, 135, 223, 5, 92, 76, 28, 116, 215, 228, 24, 190, 219, 193, 155, 251, 141, 40, 237, 139, 59, 253, 77, 209, 230, 202, 202, 145, 29, 215, 251, 26, 237, 197, 175, 123, 83, 60, 66, 84, 2, 25, 158, 110, 56, 70, 54, 78, 19, 248, 175, 223, 236, 102, 130, 23, 64, 210, 15, 51, 222, 69, 65, 88, 171, 106, 61, 235, 127, 156, 206, 101, 227, 106, 59, 180, 15, 102, 193, 190, 117, 193, 142, 237, 182, 18, 151, 98, 155, 67, 233, 151, 159, 185, 134, 65, 48, 69, 69, 118, 248, 112, 78, 18, 206, 56, 143, 23, 143, 164, 208, 162, 247, 60, 230, 29, 86, 91, 230, 194, 68, 29, 155, 222, 132, 198, 233, 172, 126, 200, 255, 176, 250, 152, 180, 11, 22, 224, 206, 68, 178, 126, 148, 35, 143, 192, 160, 6, 172, 114, 106, 182, 128, 5, 79, 157, 61, 17, 202, 249, 36, 9, 188, 92, 190, 91, 64, 43, 113, 138, 208, 17, 192, 224, 225, 135, 85, 192, 217, 57, 81, 76, 181, 150, 133, 100, 171, 212, 123, 83, 113, 237, 149, 192, 31, 107, 75] 0

[51, 196, 44, 2, 194, 126, 58, 66, 209, 73, 209, 194, 232, 31, 161, 6, 205, 26, 136, 182, 30, 163, 3, 73, 218, 241, 196, 93, 144, 249, 208, 141, 182, 53, 141, 13, 109, 239, 104, 0, 92, 66, 67, 193, 163, 11, 129, 72, 228, 91, 134, 244, 153, 27, 164, 225, 52, 147, 122, 61, 144, 90, 249, 151, 85, 130, 210, 82, 252, 229, 156, 155, 152, 92, 14, 47, 1, 226, 66, 103, 246, 21, 178, 147, 85, 71, 45, 173, 132, 175, 163, 39, 3, 111, 159, 48, 163, 134, 47, 91, 5, 100, 153, 209, 243, 113, 111, 178, 148, 86, 223, 29, 204, 194, 99, 143, 117, 180, 239, 167, 228, 53, 239, 98, 246, 143, 85, 87, 88, 122, 149, 107, 153, 2, 93, 191, 182, 127, 230, 143, 166, 94, 189, 140, 211, 89, 128, 1, 160, 104, 205, 115, 13, 57, 25, 20, 55, 68, 95, 43, 227, 32, 154, 225, 80, 187, 6, 163, 148, 224, 239, 19, 45, 132, 138, 249, 143, 20, 241, 180, 237, 190, 90, 211, 15, 159, 135, 235, 119, 178, 92, 15, 83, 133, 75, 69, 156, 145, 80, 96, 219, 134, 216, 94, 7, 2, 243, 236, 87, 213, 164, 145, 130, 121, 199, 128, 216, 173, 183, 225, 18, 41, 128, 169, 96, 185, 32, 241, 65, 28, 160, 233, 78, 4, 84, 189, 85, 2, 177, 223, 123, 182, 44, 22, 201, 76, 119, 128, 209, 175, 77, 171, 149, 150, 9, 112, 45, 170, 9, 105, 112, 24, 246, 86, 90, 220, 127, 120, 140, 168, 17, 184, 32, 253, 148, 12, 132, 247, 119, 73, 217, 114, 233, 48, 76, 162, 248, 63, 154, 25, 95, 202, 120, 49, 51, 91, 127, 165, 54, 116, 77, 247, 93, 45, 151, 188, 57, 77, 66, 95, 243, 177, 238, 18, 79, 98, 114, 242, 193, 157, 62, 237, 185, 218, 112, 237, 157, 123, 93, 102, 190, 191, 226, 99, 236, 99, 47, 118, 182, 69, 190, 82, 184, 232, 205, 153, 20, 30, 11, 168, 116, 7, 155, 244, 130, 64, 171, 62, 228, 196, 220, 177, 182, 235, 170, 139, 227, 230, 144, 1, 227, 148, 39, 75, 68, 114, 176, 40, 146, 136, 84, 13, 250, 140, 56, 64, 173, 152, 86, 95, 200, 241, 83, 7, 180, 93, 191, 209, 87, 54, 202, 144, 158, 206, 184, 142, 39, 133, 120, 22, 246, 133, 225, 82, 135, 121, 201, 48, 227, 29, 73, 5, 38, 66, 106, 105, 10, 33, 123, 83, 143, 209, 197, 153, 111, 137, 99, 57, 76, 230, 215, 16, 195, 239, 203, 187, 44, 200, 42, 42, 240, 78, 118, 11, 3, 50, 22, 64, 154, 222, 154, 40, 53, 22, 185, 196, 155, 195, 248, 104, 254, 154, 236, 162, 112, 63, 36, 195, 157, 14, 133, 160, 77, 9, 64, 81, 62, 172, 133, 169, 18, 26, 177, 225, 201, 135, 21, 148, 126, 186, 104, 83, 172, 169, 87, 142, 29, 51, 152, 110, 21, 151, 124, 156, 11, 95, 78, 60, 54, 108, 89, 91, 111, 128, 199, 175, 108, 51, 50, 72, 184, 185, 215, 126, 66, 78, 175, 184, 252, 28, 101, 171, 69, 134, 252, 160, 7, 104, 197, 211, 224, 166, 245, 188, 45, 124, 226, 50, 140, 199, 105, 164, 115, 138, 192, 76, 101, 196, 59, 21, 128, 209, 45, 33, 2, 249, 35, 26, 243, 43, 185, 231, 44, 73, 203, 10, 206, 206, 165, 79, 52, 18, 10, 123, 96, 177, 11, 249, 124, 88, 169, 140, 77, 142, 241, 139, 72, 190, 38, 66, 20, 179, 198, 21, 40, 112, 138, 254, 121, 105, 35, 149, 90, 209, 51, 6, 110, 35, 30, 167, 160, 140, 32, 151, 111, 170, 140, 212, 6, 238, 145, 236, 80, 134, 168, 45, 167, 242, 57, 17, 174, 163, 183, 240, 189, 17, 151, 22, 135, 86, 200, 130, 215, 116, 127, 104, 40, 232, 30, 149, 20, 156, 54, 223, 93, 150, 252, 220, 128, 65, 232, 239, 220, 21, 9, 47, 54,

104, 187, 90, 38, 115, 129, 137, 124, 173, 229, 140, 160, 193, 14, 65, 11, 58, 2, 103, 76, 203, 42, 115, 27, 154, 158, 233, 157, 126, 197, 240, 139, 178, 66, 45, 14, 229, 55, 196, 42, 23, 62, 235, 133, 68, 202, 20, 94, 127, 14, 255, 28, 105, 108, 54, 89, 67, 115, 114, 10, 230, 91, 98, 113, 185, 137, 20, 113, 229, 165, 181, 134, 48, 46, 223, 138, 175, 56, 180, 105, 228, 49, 38, 230, 219, 241, 43, 82, 110, 70, 216, 185, 22, 142, 83, 176, 50, 145, 79, 112, 221, 81, 50, 34, 112, 115, 37, 20, 82, 104, 110, 163, 104, 87, 34, 167, 45, 1, 162, 205, 109, 103, 87, 71, 61, 83, 88, 231, 149, 225, 251, 255, 106, 42, 164, 107, 133, 153, 30, 78, 64, 70, 203, 87, 38, 165, 183, 16, 82, 93, 202, 201, 16, 41, 243, 141, 34, 132, 248, 186, 186, 138, 61, 210, 75, 64, 164, 73, 143, 33, 49, 129, 146, 119, 121, 231, 91, 0, 83, 29, 113, 210, 137, 240, 206, 129, 143, 194, 88, 212, 204, 87, 99, 147, 124, 23, 191, 60, 94, 196, 203, 214, 17, 189, 228, 148, 8, 70, 78, 87, 240, 188, 193, 67, 253, 165, 18, 126, 255, 253, 84, 139, 121, 193, 238, 21, 195, 147, 190, 147, 77, 1, 101, 81, 44, 232, 38, 128, 230, 237, 202, 29, 133, 92, 33, 125, 253, 117, 71, 96, 3, 123, 88, 23, 47, 120, 165, 13, 149, 197, 206, 35, 77, 145, 223, 196, 173, 16, 63, 202, 46, 169, 10, 23, 96, 49, 16, 187, 139, 164, 212, 94, 183, 183, 33, 54, 225, 66, 146, 204, 181, 214, 219, 207, 68, 131, 155, 23, 159, 63, 162, 204, 174, 237, 133, 151, 202, 209, 64, 181, 153, 221, 223, 25, 222, 39, 142, 193, 114, 30, 11, 79, 117, 15, 135, 165, 193, 210, 66, 237, 166, 202, 167, 64, 249, 49, 224] 0

[240, 45, 178, 59, 74, 186, 153, 192, 178, 131, 73, 221, 162, 100, 132, 4, 42, 199, 50, 79, 28, 134, 90, 132, 164, 156, 178, 177, 80, 19, 146, 135, 185, 190, 32, 4, 53, 54, 160, 102, 38, 117, 102, 95, 41, 248, 33, 76, 155, 167, 9, 90, 28, 133, 133, 174, 170, 203, 33, 139, 130, 193, 49, 53, 243, 215, 158, 18, 137, 228, 135, 204, 186, 12, 70, 202, 233, 215, 147, 205, 141, 123, 192, 55, 203, 146, 78, 106, 174, 89, 82, 173, 41, 154, 150, 72, 93, 79, 159, 205, 159, 61, 172, 72, 31, 43, 215, 143, 169, 113, 119, 123, 159, 133, 170, 189, 25, 88, 54, 184, 114, 173, 111, 169, 128, 209, 222, 91, 55, 0, 4, 91, 252, 204, 43, 156, 121, 90, 17, 207, 105, 128, 234, 97, 129, 140, 113, 75, 80, 59, 245, 77, 157, 2, 59, 146, 119, 49, 93, 6, 230, 98, 210, 190, 198, 131, 0, 1, 51, 65, 3, 97, 90, 72, 42, 58, 136, 45, 130, 77, 21, 62, 10, 104, 33, 179, 151, 117, 109, 64, 236, 217, 48, 14, 102, 209, 128, 118, 105, 204, 72, 90, 15, 238, 83, 241, 100, 68, 52, 150, 138, 139, 210, 51, 154, 62, 99, 217, 85, 169, 123, 199, 49, 58, 167, 240, 60, 231, 44, 66, 216, 86, 195, 93, 58, 25, 136, 243, 137, 127, 104, 128, 105, 59, 255, 135, 76, 112, 149, 120, 122, 181, 145, 192, 224, 55, 203, 234, 224, 210, 105, 67, 101, 33, 42, 65, 102, 219, 166, 76, 56, 3, 200, 118, 68, 106, 133, 34, 106, 39, 210, 250, 222, 22, 47, 60, 146, 180, 125, 114, 18, 112, 238, 24, 95, 62, 6, 108, 17, 59, 207, 73, 246, 119, 196, 217, 139, 74, 15, 131, 95, 151, 128, 83, 179, 197, 114, 1, 8, 227, 36, 190, 85, 19, 230, 1, 140, 227, 210, 92, 220, 211, 245, 103, 232, 190, 164, 184, 123, 109, 124, 12, 0, 9, 80, 165, 151, 152, 191, 208, 55, 147, 217, 114, 243, 22, 213, 136, 93, 201, 61, 42, 93, 86, 71, 253, 108, 4, 188, 163, 246, 22, 124, 33, 184, 140, 160, 186, 39, 246, 65, 110, 150, 219, 29, 179, 168, 206, 238, 192, 94, 121, 19, 126, 170, 117, 187, 26, 13, 1, 81, 4, 62, 57, 106, 197, 209, 2, 248, 233, 246, 238, 201, 168, 168, 106, 151, 123, 55, 88, 247, 165, 174, 39, 131, 70, 54, 41, 36, 112, 25, 170, 131, 138, 43, 144, 76, 222, 115, 181, 88, 46, 155, 241, 204, 135, 137, 45, 145, 5, 251, 0, 252, 234, 118, 227, 250, 191, 245, 60, 87, 223, 32, 113, 193, 26, 62, 240, 175, 140, 145, 36, 32, 74, 204, 43, 57, 254, 165, 195, 210, 133, 253, 28, 200, 196, 59, 202, 27, 202, 247, 145, 34, 222, 168, 163, 248, 203, 161, 220, 230, 183, 124, 218, 124, 222, 35, 122, 137, 121, 196, 188, 64, 195, 155, 254, 41, 60, 216, 225, 84, 50, 222, 173, 136, 73, 19, 111, 187, 146, 246, 18, 110, 134, 98, 191, 14, 63, 42, 254, 226, 183, 239, 255, 184, 193, 107, 104, 68, 23, 149, 185, 104, 54, 147, 73, 249, 37, 59, 29, 149, 43, 209, 142, 63, 137, 176, 132, 40, 26, 231, 77, 214, 108, 92, 150, 70, 20, 34, 22, 55, 18, 153, 44, 150, 155, 93, 0, 235, 16, 128, 246, 24, 229, 23, 115, 218, 203, 242, 44, 104, 174, 60, 214, 133, 111, 116, 45, 118, 63, 131, 146, 161, 100, 72, 234, 190, 12, 204, 255, 8, 180, 23, 119, 59, 200, 110, 73, 124, 27, 45, 16, 48, 231, 203, 71, 224, 208, 34, 153, 50, 87, 229, 40, 187, 74, 17, 154, 69, 18, 0, 42, 36, 31, 131, 16, 30, 253, 1, 73, 225, 59, 247, 248, 168, 144, 247, 202, 61, 140, 69, 139, 92, 173, 223, 231, 185, 197, 78, 143, 13, 80, 159, 249, 163, 72, 38, 133, 163, 85, 119, 59, 244, 234, 88, 156, 78, 183, 156, 225, 7, 38, 169, 67, 153, 134, 8, 84, 35, 166, 136, 73, 47, 30, 227, 28, 84, 30, 231, 130, 27, 244, 185, 222, 195, 246, 130, 206, 189, 209, 215, 162, 207, 236, 195, 12, 213, 205, 43, 42, 203, 178, 120, 154, 207, 107, 9, 110, 50, 3, 137, 206, 103, 184, 137, 73, 229, 186, 43, 40,

169, 145, 158, 188, 100, 132, 216, 3, 28, 22, 19, 116, 126, 250, 13, 80, 81, 115, 65, 253, 9, 150, 144, 150, 105, 196, 3, 14, 166, 134, 102, 35, 209, 71, 226, 90, 0, 91, 22, 109, 38, 207, 93, 123, 20, 241, 100, 141, 64, 92, 205, 128, 14, 103, 80, 153, 131, 43, 225, 80, 187, 216, 63, 70, 60, 125, 175, 186, 147, 87, 104, 102, 137, 108, 239, 223, 86, 61, 65, 199, 104, 232, 79, 240, 33, 114, 243, 240, 123, 153, 26, 206, 109, 48, 164, 220, 175, 159, 191, 217, 32, 226, 116, 144, 97, 137, 240, 140, 230, 242, 8, 105, 60, 105, 109, 211, 9, 165, 60, 85, 181, 110, 178, 77, 83, 39, 253, 31, 64, 180, 2, 102, 76, 241, 70, 27, 0, 54, 33, 81, 132, 52, 56, 176, 79, 178, 166, 119, 14, 227, 129, 66, 240, 190, 27, 73, 27, 130, 49, 65, 222, 42, 44, 246, 22, 182, 137, 255, 239, 149, 22, 80, 108, 234, 27, 48, 106, 114, 25, 25, 144, 62, 231, 37, 112, 125, 221, 134, 97, 111, 67, 25, 140, 11, 158, 14, 141, 31, 110, 148, 138, 142, 76, 28, 27, 175, 187, 160, 41, 231, 41, 56, 223, 147, 127, 93, 98, 168, 223, 118, 60, 217, 16, 137, 122, 136, 233, 110, 63, 141, 212, 201, 30, 155, 184, 204, 80, 70, 111, 209, 238, 53, 78, 189, 200, 185, 57, 245, 9, 50, 35, 109, 7, 58, 3, 211, 148, 163, 109, 85, 204, 114, 153, 144] 1

Code:

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the data from the CSV file
data = pd.read_csv('malware_data.csv')

# Separate the features (images) and labels
X = np.array([row for row in data['image'].apply(lambda x: np.array(eval(x)))])
y = data['label'].values

# Determine the dimensions of your input images
IMG_HEIGHT = X.shape[1] # Assuming each image is a 1D array
IMG_WIDTH = 1 # For 1D arrays, the width is 1
IMG_CHANNELS = 1 # For grayscale images, use 1 channel

# Split 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)

# Reshape the input data to match the expected format for CNN
X_train = X_train.reshape(-1, IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS)
X_test = X_test.reshape(-1, IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS)

# CNN model architecture
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS)))
```

```

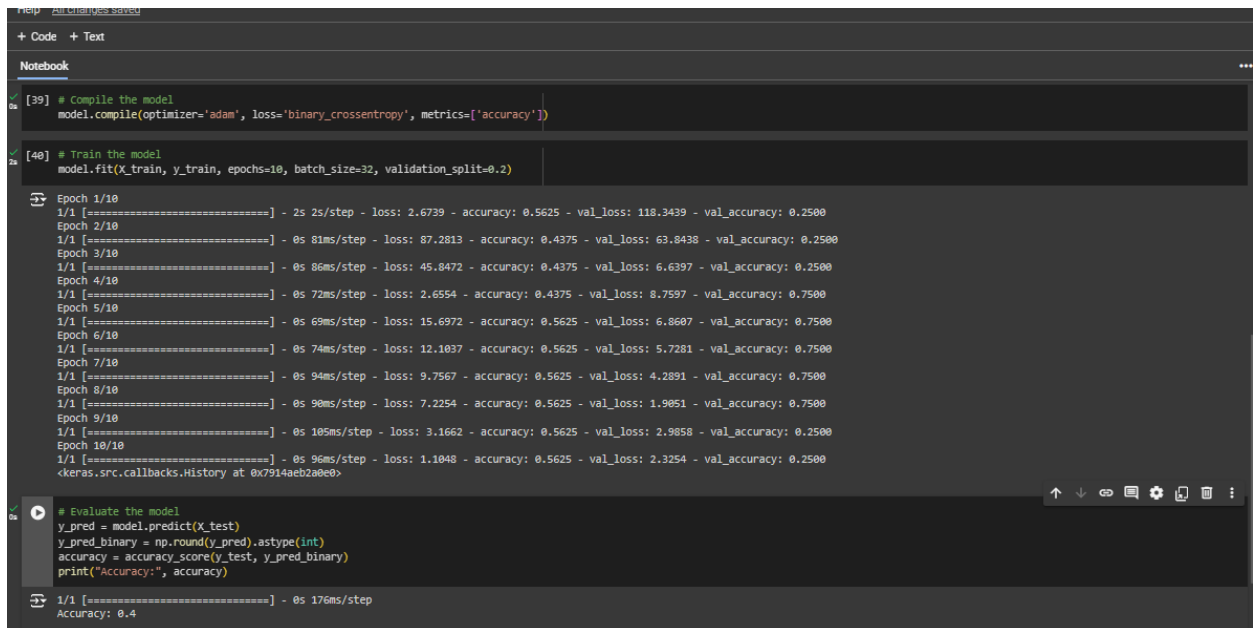
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

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

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)

# Evaluate the model
y_pred = model.predict(X_test)
y_pred_binary = np.round(y_pred).astype(int)
accuracy = accuracy_score(y_test, y_pred_binary)
print("Accuracy:", accuracy)

```



```

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

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)

# Evaluate the model
y_pred = model.predict(X_test)
y_pred_binary = np.round(y_pred).astype(int)
accuracy = accuracy_score(y_test, y_pred_binary)
print("Accuracy:", accuracy)

```

Epoch 1/10
1/1 [=====] - 2s 2s/step - loss: 2.6739 - accuracy: 0.5625 - val_loss: 118.3439 - val_accuracy: 0.2500
Epoch 2/10
1/1 [=====] - 0s 81ms/step - loss: 87.2813 - accuracy: 0.4375 - val_loss: 63.8438 - val_accuracy: 0.2500
Epoch 3/10
1/1 [=====] - 0s 86ms/step - loss: 45.8472 - accuracy: 0.4375 - val_loss: 6.6397 - val_accuracy: 0.2500
Epoch 4/10
1/1 [=====] - 0s 72ms/step - loss: 2.6554 - accuracy: 0.4375 - val_loss: 8.7597 - val_accuracy: 0.7500
Epoch 5/10
1/1 [=====] - 0s 69ms/step - loss: 15.6972 - accuracy: 0.5625 - val_loss: 6.8687 - val_accuracy: 0.7500
Epoch 6/10
1/1 [=====] - 0s 74ms/step - loss: 12.1037 - accuracy: 0.5625 - val_loss: 5.7281 - val_accuracy: 0.7500
Epoch 7/10
1/1 [=====] - 0s 94ms/step - loss: 9.7567 - accuracy: 0.5625 - val_loss: 4.2891 - val_accuracy: 0.7500
Epoch 8/10
1/1 [=====] - 0s 90ms/step - loss: 7.2254 - accuracy: 0.5625 - val_loss: 1.9851 - val_accuracy: 0.7500
Epoch 9/10
1/1 [=====] - 0s 105ms/step - loss: 3.1662 - accuracy: 0.5625 - val_loss: 2.9858 - val_accuracy: 0.2500
Epoch 10/10
1/1 [=====] - 0s 96ms/step - loss: 1.1048 - accuracy: 0.5625 - val_loss: 2.3254 - val_accuracy: 0.2500
<keras.src.callbacks.History at 0x7914aeb2a0eb>

```

# Evaluate the model
y_pred = model.predict(X_test)
y_pred_binary = np.round(y_pred).astype(int)
accuracy = accuracy_score(y_test, y_pred_binary)
print("Accuracy:", accuracy)

```

1/1 [=====] - 0s 176ms/step
Accuracy: 0.4