### Importing all the libraries

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
from tensorflow.keras.preprocessing import image
import numpy as np
import matplotlib.pyplot as plt
```

### Load and preprocess the data

```
# loading the cifar10 dataset


(x_train, y_train), (x_test, y_test) = datasets.cifar10.load_data()
```

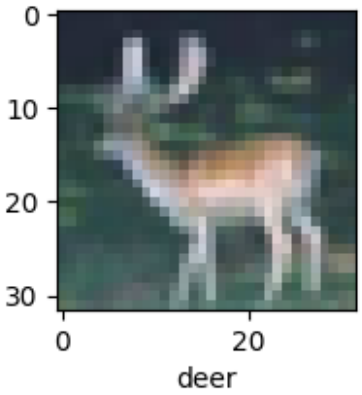
```
    Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
    170498071/170498071 [==============================] - 2s 0us/step
```

```
# converting data into one dimentional array


y_train = y_train.reshape(-1,)
y_train[:5]
```

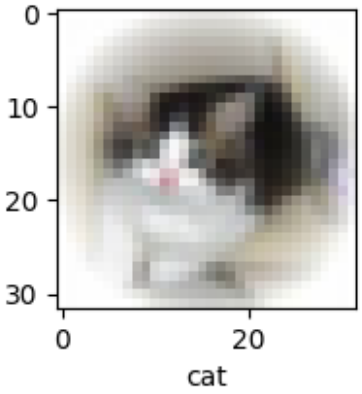
```
    array([6, 9, 9, 4, 1], dtype=uint8)
```

```
y_test = y_test.reshape(-1,)


# Only keep samples that represent animals (classes 2, 3, 4, 5, 6)

animal_indices_train = np.where((y_train >= 2) & (y_train <= 6))[0]
x_train = x_train[animal_indices_train]
y_train = y_train[animal_indices_train] - 2  # Adjust labels to be in range [0, 4]

animal_indices_test = np.where((y_test >= 2) & (y_test <= 6))[0]
x_test = x_test[animal_indices_test]
y_test = y_test[animal_indices_test] - 2  # Adjust labels to be in range [0, 4]
```

```
# defining target classes

classes = ["bird", "cat", "deer", "dog", "frog"]


def plot_sample(x, y, index):
    plt.figure(figsize=(15, 2))
    plt.imshow(x[index])
    plt.xlabel(classes[y[index]])


plot_sample(x_train, y_train, 70)
```




```
plot_sample(x_train, y_train, 100)
```




```
# Scaling the pixel values to a range between 0 and 1
```

```
x_train = x_train / 255.0
x_test = x_test / 255.0
```

### Define/Built the model

```
cnn = models.Sequential([
    layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(5, activation='softmax')  # Adjusted to 5 output classes
])
```

### Compile the model

```
cnn.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

### Train the model

```
cnn.fit(x_train, y_train, epochs=20)
```

```
Epoch 1/20
782/782 [==============================] - 32s 39ms/step - loss: 1.3004 - accuracy: 0.4491
Epoch 2/20
782/782 [==============================] - 32s 40ms/step - loss: 1.0425 - accuracy: 0.5871
Epoch 3/20
782/782 [==============================] - 29s 38ms/step - loss: 0.9381 - accuracy: 0.6356
Epoch 4/20
782/782 [==============================] - 32s 41ms/step - loss: 0.8621 - accuracy: 0.6664
Epoch 5/20
782/782 [==============================] - 30s 39ms/step - loss: 0.8022 - accuracy: 0.6964
Epoch 6/20
782/782 [==============================] - 30s 39ms/step - loss: 0.7458 - accuracy: 0.7168
Epoch 7/20
```

```
782/782 [==============================] - 29s 38ms/step - loss: 0.6997 - accuracy: 0.7358
Epoch 8/20
782/782 [==============================] - 29s 38ms/step - loss: 0.6485 - accuracy: 0.7564
Epoch 9/20
782/782 [==============================] - 32s 41ms/step - loss: 0.5970 - accuracy: 0.7788
Epoch 10/20
782/782 [==============================] - 29s 38ms/step - loss: 0.5614 - accuracy: 0.7911
Epoch 11/20
782/782 [==============================] - 29s 37ms/step - loss: 0.5132 - accuracy: 0.8098
Epoch 12/20
782/782 [==============================] - 29s 38ms/step - loss: 0.4788 - accuracy: 0.8225
Epoch 13/20
782/782 [==============================] - 31s 39ms/step - loss: 0.4411 - accuracy: 0.8358
Epoch 14/20
782/782 [==============================] - 30s 38ms/step - loss: 0.4026 - accuracy: 0.8508
Epoch 15/20
782/782 [==============================] - 29s 38ms/step - loss: 0.3687 - accuracy: 0.8648
Epoch 16/20
782/782 [==============================] - 30s 38ms/step - loss: 0.3384 - accuracy: 0.8772
Epoch 17/20
782/782 [==============================] - 30s 39ms/step - loss: 0.3089 - accuracy: 0.8890
Epoch 18/20
782/782 [==============================] - 31s 40ms/step - loss: 0.2779 - accuracy: 0.8986
Epoch 19/20
782/782 [==============================] - 30s 38ms/step - loss: 0.2470 - accuracy: 0.9119
Epoch 20/20
782/782 [==============================] - 33s 43ms/step - loss: 0.2267 - accuracy: 0.9189
<keras.src.callbacks.History at 0x7f7435a5fe20>
```

### Evaluate the model

```
cnn.evaluate(x_test, y_test)
```

```
157/157 [==============================] - 3s 17ms/step - loss: 1.3632 - accuracy: 0.6694
[1.3631736040115356, 0.6693999767303467]
```

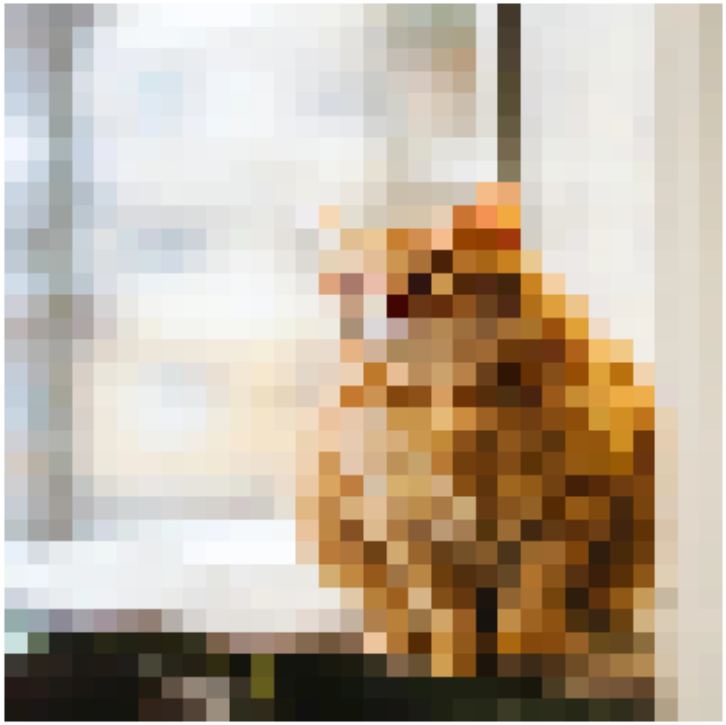
### Function for Classification

```
def classify_animal_image(model, image_path):
    # Load the image
    img = image.load_img(image_path, target_size=(32, 32))

    # Convert the image to a numpy array
    img_array = image.img_to_array(img)

    # Expand dimensions to match the model input shape
    img_array = np.expand_dims(img_array, axis=0)

    # Preprocess the image
    img_array = img_array / 255.0

    # Predict the class probabilities
    predictions = model.predict(img_array)

    # Get the predicted class index
    predicted_class_index = np.argmax(predictions)

    # Map the class index to the corresponding animal class
    animal_classes = ["bird", "cat", "deer", "dog", "frog"]
    predicted_animal_class = animal_classes[predicted_class_index]

    # Display the image
    plt.imshow(img)
    plt.axis('off')
    plt.show()

    return predicted_animal_class, predictions[0][predicted_class_index]
```

**Example usage:**

```
# Path to the input image
image_path = '/content/pexels-photo-1170986.webp'


# Classify the input image
predicted_class, confidence = classify_animal_image(cnn, image_path)
```

```
1/1 [==============================] - 0s 22ms/step
```



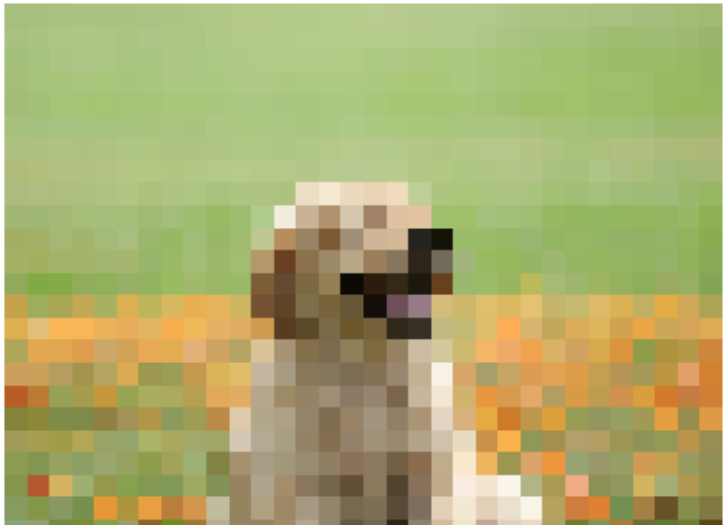
```
print("Predicted class:", predicted_class)
print("Confidence:", confidence)
```

```
Predicted class: cat
Confidence: 0.95283914
```

```
# Path to the input image
image_path = '/content/dog-puppy-on-garden-royalty-free-image-1586966191.jpg'
```

```
# Classify the input image
predicted_class, confidence = classify_animal_image(cnn, image_path)
```

```
1/1 [==============================] - 0s 20ms/step
```



```
print("Predicted class:", predicted_class)
print("Confidence:", confidence)
```

```
Predicted class: dog
Confidence: 0.98681086
```