```
import nltk
import json
import pickle
import random
import keras
from google.colab import files
uploaded = files.upload()
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()


import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD


words=[]
classes = []
documents = []
ignore_words = ['?', '!']
# The file was uploaded to the current working directory, not the parent
# Use 'intents.json' instead of '../intents.json'
data_file = open('intents.json').read()
intents = json.loads(data_file)

# ... (rest of your code remains the same) ...
for intent in intents['intents']:
    for pattern in intent['patterns']:

        #tokenize each word
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        #add documents in the corpus
        documents.append((w, intent['tag']))

        # add to our classes list
        if intent['tag'] not in classes:
            classes.append(intent['tag'])

# lemmaztize and lower each word and remove duplicates
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]
words = sorted(list(set(words)))
# sort classes
classes = sorted(list(set(classes)))
# documents = combination between patterns and intents
print (len(documents), "documents")
# classes = intents
print (len(classes), "classes", classes)
# words = all words, vocabulary
print (len(words), "unique lemmatized words", words)


pickle.dump(words,open('words.pkl','wb'))
pickle.dump(classes,open('classes.pkl','wb'))

# create our training data
training = []
# create an empty array for our output
output_empty = [0] * len(classes)
# training set, bag of words for each sentence
for doc in documents:
    # initialize our bag of words
    bag = []
    # list of tokenized words for the pattern
    pattern_words = doc[0]
    # lemmatize each word - create base word, in attempt to represent related words
    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]
    # create our bag of words array with 1, if word match found in current pattern
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)
```

```
        # output is a '0' for each tag and '1' for current tag (for each pattern)
        output_row = list(output_empty)
        output_row[classes.index(doc[1])] = 1

        training.append([bag, output_row])

    #Before converting to numpy array, ensure all bags have the same length
    bag_len = len(words)  # Get the expected length of the bag of words
    for i in range(len(training)):
        if len(training[i][0]) != bag_len:  # If bag length is incorrect
            diff = bag_len - len(training[i][0])  # Calculate the difference
            training[i][0].extend([0] * diff)  # Pad with zeros to match expected length

    # shuffle our features and turn into np.array
    random.shuffle(training)
    training = np.array(training, object) # The change is here
    # create train and test lists. X - patterns, Y - intents
    train_x = list(training[:,0])
    train_y = list(training[:,1])
    print("Training data created")

    # Create model - 3 layers. First layer 128 neurons, second layer 64 neurons and 3rd output layer contains number of neurons
    # equal to number of intents to predict output intent with softmax
    model = Sequential()
    model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(len(train_y[0]), activation='softmax'))

    # Compile model. Stochastic gradient descent with Nesterov accelerated gradient gives good results for this model
    sgd = SGD(learning_rate=0.01, decay=1e-6, momentum=0.9, nesterov=True) # Changed 'lr' to 'learning_rate'
    model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

    #fitting and saving the model
    hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1)
    model.save('chatbot_model.h5', hist)

    print("model created")
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving intents.json to intents (3).json
47 documents
9 classes ['adverse_drug', 'blood_pressure', 'blood_pressure_search', 'goodbye', 'greeting', 'hospital_search', 'options', 'pharmacy_s
88 unique lemmatized words ["'s", ',', 'a', 'adverse', 'all', 'anyone', 'are', 'awesome', 'be', 'behavior', 'blood', 'by', 'bye', 'car
Training data created
Epoch 1/200
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.0849 - loss: 2.2566
Epoch 2/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.3206 - loss: 2.0565
Epoch 3/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.3456 - loss: 2.0673
Epoch 4/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5033 - loss: 1.9786
Epoch 5/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.3520 - loss: 1.9954
Epoch 6/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.4099 - loss: 1.8008
Epoch 7/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.3820 - loss: 1.7391
Epoch 8/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5571 - loss: 1.5572
Epoch 9/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5681 - loss: 1.5088
Epoch 10/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6083 - loss: 1.3152
Epoch 11/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6729 - loss: 1.1731
Epoch 12/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5281 - loss: 1.2605
Epoch 13/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6889 - loss: 1.1632
Epoch 14/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6713 - loss: 0.9231
Epoch 15/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6875 - loss: 0.9013
Epoch 16/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9108 - loss: 0.7109
Epoch 17/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6457 - loss: 0.8759
Epoch 18/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7897 - loss: 0.6427
Epoch 19/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7694 - loss: 0.6164
Epoch 20/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7824 - loss: 0.6506
Epoch 21/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8629 - loss: 0.5207
Epoch 22/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9227 - loss: 0.3881
```

```
import nltk
import json
import pickle
import random
import keras
from google.colab import files
uploaded = files.upload()  # File upload line remains here

# ... (other imports and code)

# The file was uploaded to the current working directory, not the parent
# Use 'intents.json' instead of '../intents.json'
data_file = open('intents.json').read()
intents = json.loads(data_file)

# Print the loaded data
print("Uploaded Data:")  # Add this line to print the data
print(json.dumps(intents, indent=4))  # Format the JSON for better readability

# ... (rest of your code)
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9826 - loss: 0.2083
Epoch 35/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9765 - loss: 0.2580
Epoch 36/200
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9941 - loss: 0.2016
Epoch 38/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9699 - loss: 0.2104
Epoch 39/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0817
Epoch 40/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9420 - loss: 0.2187
Epoch 41/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9642 - loss: 0.1890
Epoch 42/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9642 - loss: 0.1643
Epoch 43/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9826 - loss: 0.1193
Epoch 44/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8839 - loss: 0.3227
Epoch 45/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.1250
Epoch 46/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8894 - loss: 0.2636
Epoch 47/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9742 - loss: 0.1372
Epoch 48/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9892 - loss: 0.1089
Epoch 49/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9629 - loss: 0.1192
Epoch 50/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0536
Epoch 51/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0584
Epoch 52/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9767 - loss: 0.1094
Epoch 53/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9447 - loss: 0.2052
Epoch 54/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9629 - loss: 0.1643
Epoch 55/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0436
Epoch 56/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9892 - loss: 0.0638
Epoch 57/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9089 - loss: 0.1467
Epoch 58/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9371 - loss: 0.1959
Epoch 59/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.1494
Epoch 60/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0344
Epoch 61/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9326 - loss: 0.1931
Epoch 62/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0719
Epoch 63/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9780 - loss: 0.0984
Epoch 64/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9826 - loss: 0.0860
Epoch 65/200
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9699 - loss: 0.0816
Epoch 66/200
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving intents.json to intents (1).json

Uploaded Data:


```
{
  "intents": [
    {
      "tag": "greeting",
      "patterns": [
        "Hi there",
        "How are you",
        "Is anyone there?",
        "Hey",
        "Hola",
        "Hello",
        "Good day"
      ],
      "responses": [
        "Hello, thanks for asking",
        "Good to see you again",
        "Hi there, how can I help?"
      ],
      "context": [
        ""
      ]
    },
    {
      "tag": "goodbye",
      "patterns": [
        "Bye",
        "See you later",
        "Goodbye",
        "Nice chatting to you, bye",
        "Till next time"
      ],
      "responses": [
        "See you!",
        "Have a nice day",
        "Bye! Come back again soon."
      ],
      "context": [
        ""
      ]
    },
    {
      "tag": "thanks",
      "patterns": [
        "Thanks",
        "Thank you",
        "That's helpful",
        "Awesome, thanks",
        "Thanks for helping me"
      ],
      "responses": [
        "Happy to help!",
        "Any time!",
        "My pleasure"
```


```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np

from keras.models import load_model
# Make sure 'chatbot_model.h5' exists in the current directory, if not, run the training cell first
model = load_model('chatbot_model.h5')
import json
import random
intents = json.loads(open('intents.json').read())
words = pickle.load(open('words.pkl','rb'))
classes = pickle.load(open('classes.pkl','rb'))


def clean_up_sentence(sentence):
    # tokenize the pattern - split words into array
    sentence_words = nltk.word_tokenize(sentence)
    # stem each word - create short form for word
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    return sentence_words

# return bag of words array: 0 or 1 for each word in the bag that exists in the sentence
```

```
def bow(sentence, words, show_details=True):
    # tokenize the pattern
    sentence_words = clean_up_sentence(sentence)
    # bag of words - matrix of N words, vocabulary matrix
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                # assign 1 if current word is in the vocabulary position
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)
    return(np.array(bag))

def predict_class(sentence, model):
    # filter out predictions below a threshold
    p = bow(sentence, words,show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
    # sort by strength of probability
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append({"intent": classes[r[0]], "probability": str(r[1])})
    return return_list

def getResponse(ints, intents_json):
    tag = ints[0]['intent']
    list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if(i['tag']== tag):
            result = random.choice(i['responses'])
            break
    return result

def chatbot_response(msg):
    ints = predict_class(msg, model)
    res = getResponse(ints, intents)
    return res


#Creating GUI with tkinter
import tkinter
from tkinter import *


def send():
    msg = EntryBox.get("1.0",'end-1c').strip()
    EntryBox.delete("0.0",END)

    if msg != '':
        ChatLog.config(state=NORMAL)
        ChatLog.insert(END, "You: " + msg + '\n\n')
        ChatLog.config(foreground="#442265", font=("Verdana", 12 ))

        res = chatbot_response(msg)
        ChatLog.insert(END, "Bot: " + res + '\n\n')

        ChatLog.config(state=DISABLED)
        ChatLog.yview(END)

!pip install ipywidgets
import ipywidgets as widgets
from IPython.display import display

# ... (rest of your code including imports and functions)

# Create the UI elements
chat_history = widgets.Output()
user_input = widgets.Text(placeholder="Enter your message here")
send_button = widgets.Button(description="Send")

# Function to handle sending messages
def on_send_button_clicked(b):
    global user_message
```

```
    user_message = user_input.value

    with chat_history:
        display(widgets.HTML(f"<b>You:</b> {user_message}"))  # Display user message

    bot_response = chatbot_response(user_message)
    with chat_history:
        display(widgets.HTML(f"<b>Bot:</b> {bot_response}"))  # Display bot response

    user_input.value = ""  # Clear the input field

# Connect the send button to the event handler
send_button.on_click(on_send_button_clicked)

# Display the UI elements
display(chat_history)
display(user_input)
display(send_button)
```

```
Epoch 132/200
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you t
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0132
Epoch 133/200
Requirement already satisfied: ipywidgets in /usr/local/lib/python3.11/dist-packages (7.7.1)
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9760 - loss: 0.0411
Epoch 134/200
Requirement already satisfied: ipython-genutils~=0.2.0 in /usr/local/lib/python3.11/dist-packages (from ipywidgets) (0.2.0)
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0249
Epoch 135/200
Requirement already satisfied: widgetsnbextension~=3.6.0 in /usr/local/lib/python3.11/dist-packages (from ipywidgets) (3.6.10)
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0427
Epoch 136/200
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from ipywidgets) (3.0.14)
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9918 - loss: 0.0184
Epoch 137/200
Requirement already satisfied: jupyter-client>=6.1.12 in /usr/local/lib/python3.11/dist-packages (from ipykernel>=4.5.1->ipywidgets) (6.
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9780 - loss: 0.0625
Epoch 138/200
Requirement already satisfied: nest-asyncio in /usr/local/lib/python3.11/dist-packages (from ipykernel>=4.5.1->ipywidgets) (1.6.0)
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0108
Epoch 139/200
Requirement already satisfied: psutil in /usr/local/lib/python3.11/dist-packages (from ipykernel>=4.5.1->ipywidgets) (5.9.5)
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9623 - loss: 0.0525
Epoch 140/200
Requirement already satisfied: ipython-genutils~=0.2.0 in /usr/local/lib/python3.11/dist-packages (from ipywidgets) (0.2.0)
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0047
Epoch 141/200
Requirement already satisfied: decorator in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets) (4.4.2)
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9720 - loss: 0.0753
Epoch 142/200
Requirement already satisfied: pickleshare in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets) (0.7.5)
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9447 - loss: 0.1028
Epoch 143/200
Requirement already satisfied: pygments in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets) (2.18.0)
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0137
Epoch 144/200
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets) (4.9.0)
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0172
Epoch 145/200
Requirement already satisfied: parso<0.9.0,>=0.8.4 in /usr/local/lib/python3.11/dist-packages (from jedi>=0.16->ipython>=4.0.0->ipywidge
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9720 - loss: 0.0837
Epoch 146/200
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.11/dist-packages (from jupyter-client>=6.1.12->ipykernel>=
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0438
Epoch 147/200
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0258
Epoch 148/200
Requirement already satisfied: nbconvert>=5 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0-
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9720 - loss: 0.0534
Epoch 149/200
Requirement already satisfied: terminado>=0.8.3 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0240
Epoch 150/200
Requirement already satisfied: nbclassic>=0.4.7 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9623 - loss: 0.0778
Epoch 151/200
Requirement already satisfied: wcwidth in /usr/local/lib/python3.11/dist-packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ip
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0325
Epoch 152/200
Requirement already satisfied: notebook-shim>=0.2.3 in /usr/local/lib/python3.11/dist-packages (from nbclassic>=0.4.7->notebook>=4.4.1->
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9908 - loss: 0.0238
```