Supplementary figures

Supplemental figure 1. Optimized Selection of Network Algorithms and Python Libraries through ChatGPT Assistance
We choose CNN for network algorithm, and we import requirent libraries, what should we do next in python i terms of import data, preprocessing images(resizing, normalizing, applying data augmentation and train-test split(%70 training, %15 test, %15 validation)

Assuming your images are stored in a directory and you have a CSV file with labels:

```python
new_size = (128, 128)
images_list = []
labels = []
subfolder_names = ['class1', 'class2', 'class3']  # Replace with your subfolder names

for i, folder_name in enumerate(subfolder_names):
    folder_path = os.path.join('d:\ismaill', folder_name)
    image_files = os.listdir(folder_path)
    images = []
    for file_name in image_files:
        file_path = os.path.join(folder_path, file_name)
        image = cv2.imread(file_path)
        resized_image = cv2.resize(image, new_size)
        images.append(resized_image)
        labels.append(i)
    images_list.append(np.array(images))

all_images = np.concatenate(images_list, axis=0)

all_images_scaled = all_images / 255.0

X_train, X_temp, y_train, y_temp = train_test_split(all_images_scaled, labels)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.15)

y_train = np.array(y_train)
y_val = np.array(y_val)
y_test = np.array(y_test)
```

Supplemental figure 2. ChatGPT’s Guidance on Data Import, Preprocessing, and Train-Test Split
Evaluate the model's performance using various metrics such as accuracy, precision, recall, and F1-score. Calculate the 95% confidence intervals for each metric to determine the range of possible values. Additionally, generate ROC curves and AUC values for each thyroid pathology. Plot loss and accuracy curves to visualize the training process and check for overfitting or underfitting. Finally, utilize the Grad-CAM visualization technique to examine the areas of the image that the model used to make its predictions.

Certainly! Here’s an example of how you can evaluate the model’s performance using various metrics, calculate confidence intervals, generate ROC curves and AUC values, plot loss and accuracy curves, and utilize Grad-CAM for visualization:

```python
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten

# Define the ChatGPT-assisted deep learning model
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=10, validation_data=(X_val, y_val))

# Evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred.round())
precision = precision_score(y_test, y_pred.round())
recall = recall_score(y_test, y_pred.round())
f1 = f1_score(y_test, y_pred.round())

# Plotting loss and accuracy
plt.figure(figsize=(10, 6))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()

# Compute 95% confidence intervals for each metric
bootstrap_scores = []
for _ in range(100):
    indices = np.random.choice(len(y_true), len(y_true))
    score = metrics(true, indices, y_pred)
    bootstrap_scores.append(score)
lower = np.percentile(bootstrap_scores, 2.5)
upper = np.percentile(bootstrap_scores, 97.5)

# Create Grad-CAM visualizations
gradcam = GradCam(model)
gradcam = gradcam(image)
plt.imshow(gradcam)
plt.show()

# Define a function to return the score of the desired class
def score_for_level(model, image):
    m = model.predict(image)
    if m[0][0] > 0.5:
        return 1
    else:
        return 0

Supplemental figure 4. Evaluating and Visualizing Model Performance through ChatGPT’s Code Suggestions