به نام خداوند بخشنده مهربان

مرضیه علیخانی
سید سعید حمیدی آزاد
بهمن ۱۴۰۲

Final Project Report
Image Processing course
Dr.Sabeti

فهرست مطالب

مقدمه

اختلال شناختی خفیف[1] (MCI) باعث کاهش عملکرد شناختی میشود که ممکن است شامل حافظه، زبان یا تفکر انتقادی باشد. به طور کلی اختلال شناختی خفیف شامل مشکلات حافظه اس. مشکلاتی مثل فراموش کردن حوادث اخیر. همچنین می تواند به عنوان مشکل با کلمات یا مشکل در حین گفتگو کردن نیز ظاهر شود. مشکلات تمرکز،توجه، و یا حل مسئله از دیگر علائم MCI هستند.

[1] https://www.alz.org/alzheimers-dementia/what-is-dementia/related_conditions/mild-cognitive-impairment

اختلال شناختی خفیف باعث تغییرات شناختی می شود که به اندازه کافی جدی است. زیرا توسط شخص مبتلا ، اعضای خانواده و دوستان فرد مبتلا قابل تشخیص است اما توانایی فرد در انجام امور روزمره را تحت تأثیر قرار نمی دهد. به طور تقریبی ۱۵ تا ۲۰ درصد افراد بالای سن ۶۵ سال دچار اختلال شناختی می شوند.

علائم و نشانه های اختلال شناختی خفیف

متخصصان در حوزه مغز و اعصاب، اختلال شناختی را به دو گروه گسترده تقسیم می‌کنند:

- اختلال amnestic که در آن از بین رفتن حافظه (آلزایمر) علامت رایج آن است.
- اختلال non amnestic که در آن مناطق دیگر شناختی مانند توانایی تصمیم گیری درست ، ترتیب مراحل لازم برای انجام یک کار پیچیده یا درک بصری تحت تأثیر قرار می‌گیرند.

دیگر نشانه های MCI

- گم کردن وسایل
- از یاد بردن رفتن به رویدادها و قرار ملاقات ها
- مشکل داشتن برای پیدا کردن کلمه های یک جمله در مقایسه با دیگر افراد هم سن و سال
- مشکلات حرکتی و مشکلات در تشخیص بوها نیز با MCI مرتبط میباشد.

چگونگی تشخیص بیماری MCI

تشخیص اختلال شناختی خفیف نیازمند یک تشخیص بالینی است که توسط متخصص مغز و اعصاب در مورد دلیل علائم فرد مشخص میشود. برای تشخیص MCI پزشک متخصص به وظایف مختصر عصبی شناختی، مثل معاینه مختصر حالت روانی می پردازد. در این روند از بیمار خواسته می‌شود تاریخ فعلی و روز هفته را اعلام کند و لیستی از موارد را به خاطر بسپارد.

همچنین متخصصان زمان تصمیم گیری در مورد تشخیص وجود این بیماریِ، مصاحبه‌های خانوادگی و غربالگری را برای بیماری‌های روانی همزمان (همبودی) مانند افسردگی نیز انجام می‌دهند.

با توجه به ماهیت پیچیده این بیماری، درمان برای هر فرد متناسب با شرایط خود بیمار خواهد بود. محققان دریافته اند که یک سوم افراد مبتلا به اختلال MCI به صورت قابل توجهی بهبود خواهند یافت. زیرا عموما به این دلیل که یک عامل برگشت پذیر مسئول مشکلات شناختی فرد است. عواملی مانند:

- آپنه خواب و سایر اختلالات خواب
- اختلالات خلقی (مانند اضطراب و افسردگی)
- کمبود ویتامین B12 و سایر مواد مغذی
- و استفاده از داروهایی که باعث عوارض جانبی شناختی می شوند

طبق تحقیقات انجام شده همان گونه که در بیماری آلزایمر دلایل ژنتیکی دیده می شود، عوامل ژنتیکی در افرادی که به MCI مبتلا هستند نیز نقش مهمی دارند. مطالعات در حال انجام است تا مشخص شود که چرا بعضی از افراد مبتلا به MCI به آلزایمر پیشرفته مبتلا می شوند ولی برخی دیگر خیر.

علل و پیشگیری از اختلال شناختی خفیف

غالبا علت های اختلال شناختی خفیف هنوز به طور کامل شناخته نشده است. متخصصان معتقدند که بسیاری از موارد ناشی از تغییرات مغزی است که در مراحل اولیه بیماری آلزایمر یا سایر زوال عقل ها رخ می دهد. طبق تحقیقات زنان در هنگام قرارگیری در سن ۶۵ تا ۷۵ سال، بیشتر از نظر MCI در معرض خطر قرار می‌گیرند. محققان معتقدند ممکن است این امر ناشی از تغییرات هورمونی باشد که در دوران یائسگی رخ می‌دهد.

عوامل خطرساز مرتبط با MCI همان عوامل زوال عقل هستند:

افزایش سن ، سابقه خانوادگی آلزایمر یا زوال عقل دیگری و شرایطی که خطر بیماری های قلبی عروقی را افزایش می دهد.

درمانMCI

در حال حاضر هیچ داروی تایید شده توسط FDA جهت درمان اختلال شناختی خفیف وجود ندارد و شواهد محدودی مبنی بر مؤثر واقع شدن داروها یا مکمل‌ها وجود دارد. انتخاب سبک زندگی سالم مانند فعالیت بدنی منظم، یادگیری مداوم و تعامل اجتماعی مؤثر می باشد. داروهایی که برای معالجه علائم بیماری آلزایمر تأیید شده اند هیچ تأثیری در تأخیر یا جلوگیری از پیشرفت بیماری MCI ندارند.

از آنجا که MCI امکان دارد علائم اولیه بیماری آلزایمر باشد، حائز اهمیت است که هر ۶ تا ۱۲ ماه یک پزشک یا مرکز تخصصی فرد را چکاپ کند. برخی از راهکارها و روش های سالم زندگی می توانند سلامت کلی بیماررا بهبود بخشیده و در سلامت شناختی فرد نقش موثری داشته باشد. این راهکارها عبارتند از:

- ورزش منظم
- رژیم غذایی کم چربی و سرشار از میوه و سبزیجات
- بازی های فکری
- شرکت در فعالیت های اجتماعی

بنابراین میتوان گفت آسیب شناختی خفیف یا MCI به عبارتی به کاهش کارایی شناختی که از سطح عادی بیشتر است اما هنوز به حد کافی برای تشخیص اختلال شناختی مانند آلزایمر نرسیده است، گفته می‌شود.

در این حالت، شخص ممکن است با مشکلاتی مانند فراموشی سطحی و کاهش توجه مواجه شود، اما هنوز برای انجام فعالیت‌های روزمره خود دچار مشکل نمی باشد. علائم آسیب شناختی خفیف ممکن است توسط پزشک تشخیص داده شده و در صورت لزوم اقدام به درمان شوند.

هرچند که آسیب شناختی خفیف ممکن است نشانه اولیه بروز بیماری‌های شناختی مانند آلزایمر باشد، اما همه کسانی که با آسیب شناختی خفیف مواجه هستند به اختلال شناختی در آینده مبتلا نخواهند شد.

آلزایمر چیست؟

آلزایمر یک بیماری نوروتروفیک است که معمولاً در سنین بالای ۶۵ سال شروع می‌شود و با کاهش تدریجی عملکرد شناختی، مخصوصاً حافظه و تمرکز، همراه است. این بیماری باعث تغییرات در ساختار و عملکرد مغز می‌شود که به تدریج باعث اختلال در حافظه، تصویرسازی، فکر کردن و یادگیری می‌شود.

علائم این بیماری شامل فراموشی شدید و تدریجی، مشکل در صحبت کردن، از دست دادن تمرکز روزانه، مشکل در تصمیم‌گیری و برنامه‌ریزی، کاهش توانایی‌های حرکتی و حتی تغییر در رفتار اجتماعی می‌شود.

با توجه به اینکه بیماری آلزایمر پیشرفتی است و در حال حاضر درمان دائمی و موثری برای آن وجود ندارد، تشخیص زودهنگام و کاهش عوامل خطر برای جلوگیری از بروز بیماری بسیار مهم است. همچنین درمان‌های پشتیبانی مانند روان‌درمانی و توانبخشی برای کمک به مدیریت علائم بیماری و بهبود کیفیت زندگی بیمار و خانواده بیمار استفاده می‌شود.

۵ تفاوت اصلی بین آسیب شناختی خفیف و آلزایمر چیست؟

آسیب شناختی خفیف و آلزایمر هر دو با کاهش توانایی شناختی همراه هستند، اما تفاوت‌هایی بین آن‌ها وجود دارد. برخی از تفاوت‌های اصلی بین آسیب شناختی خفیف و آلزایمر عبارتند از:

شدت علائم:

در آسیب شناختی خفیف، علائم شناختی کمتر شدید هستند و شخص هنوز می‌تواند به خوبی فعالیت‌های روزمره خود را انجام دهد، اما در آلزایمر، علائم شدیدتر و بیشتر است و باعث مشکلات جدی در فعالیت‌های روزمره شخص می‌شود.

سرعت پیشرفت بیماری:

در آسیب شناختی خفیف، پیشرفت بیماری به طور کلی کندتر اتفاق می‌افتد و شاید حتی تا سال‌ها طول بکشد، اما در آلزایمر، بیماری به صورت پیشرفتی تندتر پیش می‌رود و به مرور زمان علائم شدیدتر می‌شوند.

نوع علائم:

در آسیب شناختی خفیف، فراموشی سطحی و کاهش توجه رایج‌تر هستند، در حالی که در آلزایمر، فراموشی عمیق و کاهش شدید توانایی شناختی رخ می‌دهد.

تشخیص:

آسیب شناختی خفیف بر اساس امتیازات نمره‌ای در تست‌های شناختی تشخیص داده می‌شود، اما تشخیص آلزایمر بر اساس بررسی تاریخچه پزشکی، تست‌های شناختی و تصاویر مغزی صورت می‌گیرد.

احتمال بروز بیماری:

آسیب شناختی خفیف ممکن است نشانه اولیه بروز آلزایمر باشد، اما همه کسانی که آسیب شناختی خفیف را تجربه می‌کنند، به آلزایمر مبتلا نمی‌شوند. اما در بسیاری از موارد، آسیب شناختی خفیف به علت عوامل دیگری مانند بیماری عروقی مغزی، اختلالات در خواب، کمبود ویتامین‌ها، داروهای خاص یا عوامل استرس زا ایجاد می‌شود و با درمان این عوامل، علائم آسیب شناختی خفیف بهبود می‌یابد.

در کل، تشخیص و تفاوت‌های بین آلزایمر و آسیب شناختی خفیف بسیار حیاتی است، زیرا درمان و مدیریت بیماری‌هایی مانند آسیب شناختی خفیف و آلزایمر به شدت وابسته به شناسایی صحیح و به موقع آن‌ها است. به همین دلیل، اگر به هر نوع از علائم شناختی برخورد کرده‌اید، بهتر است به پزشک خود مراجعه کنید تا بتواند این علائم را ارزیابی کرده و درمان مناسب را تجویز کند.

آیا می توان گفت آسیب شناختی خفیف (MCI) همان آلزایمر درجه اول است؟

خیر، آلزایمر و آسیب شناختی خفیف دو بیماری متفاوت هستند و نمی‌توان آن‌ها را به عنوان یک بیماری در نظر گرفت. آسیب شناختی خفیف در واقع یک مرحله قبل از آلزایمر است که به تدریج می‌تواند به آلزایمر تبدیل شود.

آسیب شناختی خفیف شامل مشکلات شناختی و حافظه می‌شود که می‌تواند باعث اختلال در فعالیت‌های روزانه فرد شود، اما برخلاف آلزایمر، معمولاً توانایی فرد در اجرای فعالیت‌های پیچیده و خلاقانه تحت تأثیر قرار نمی‌گیرد و عموماً قابلیت یادگیری فرد در طول زمان باقی می‌ماند.

با این حال، با توجه به مراحل تکاملی بیماری، در برخی موارد، آسیب شناختی خفیف ممکن است به آلزایمر تبدیل شود. در کل، تشخیص دقیق و مدیریت مناسب آسیب شناختی خفیف و آلزایمر بسیار حائز اهمیت است و در صورت مشاهده هرگونه نشانه‌ای از این بیماری‌ها، بهتر است به پزشک خود مراجعه کنید.

آیا آسیب شناختی خفیف (MCI) خطرناک است و احتمال ابتلا به آلزایمر را دارد؟

آسیب شناختی خفیف نیاز به توجه دارد اما معمولاً خطرناک نیست و می‌توان آن را بهبود داد. با این حال، احتمال تبدیل شدن آسیب شناختی خفیف به آلزایمر وجود دارد. بیمارانی که دچار آسیب شناختی خفیف هستند، به تدریج می‌توانند از این بیماری به آلزایمر مبتلا شوند، اما در بعضی موارد آن‌ها می‌توانند تا سال‌ها بدون پیشرفت بیشتر به زندگی خود ادامه دهند.

همچنین باید گفت که مشخص نیست که آسیب شناختی خفیف در همه موارد به آلزایمر تبدیل می‌شود و به همین دلیل برای برخی افراد، تغییر در شیوه زندگی و درمان زودهنگام می‌تواند کمک کننده باشد. به هر حال، هر گونه

نشـانه‌ای از مشـکلات شـناختی و حافظـه بـه همـراه علائـم دیگـر نظیـر تغییـرات شخصـیتی، تغییـرات روانـی، یـا تغییـرات رفتاری، نباید نادیده گرفته شود و نیاز به ارزیابی دقیق و درمان دارد.

روش های تصویربرداری از مغز برای پیش بینی آلزایمر در افراد مبتلا بهMCI

روش هـای تصـویربرداری از مغـز بـرای پـیش بینـی آلزایمـر در افـراد مبـتلا بـه MCI شـامل انـواع مختلفـی از روش هـا مـی باشد. در ادامه به برخی از این روش ها اشاره خواهیم کرد:

۱. PET scan: در ایـن روش، یـک مـاده رادیواکتیـو بـه بیمـار تزریـق مـی شـود و سـپس بـا اسـتفاده از دسـتگاه PET scan، تصـاویری از فعالیـت مغـز بیمـار گرفتـه مـی شـود. بـا تحلیـل ایـن تصـاویر، نشـان داده شـده اسـت کـه در بیماران MCI که در آینده به آلزایمر مبتلا می شوند، فعالیت در مناطق خاصی از مغز کاهش می یابد.
۲. MRI: در ایـن روش، بـا اسـتفاده از دسـتگاهMRI ، تصـاویر ۳ Dاز مغـز بیمـار گرفتـه مـی شـود. بـا تحلیـل ایـن تصـاویر، نشـان داده شـده اسـت کـه در بیمـاران MCI کـه در آینـده بـه آلزایمـر مبـتلا مـی شـوند، حجـم برخـی از مناطق مغز کاهش می یابد.
۳. rs-fMRI: در ایـن روش، بـا اسـتفاده از دسـتگاهfMRI ، تصـاویری از فعالیـت هـای مغـز بیمـار در حالـت اسـتراحت گرفته می شود. با تحلیل ایـن تصـاویر، نشـان داده شـده اسـت کـه در بیمـاران MCI کـه در آینـده بـه آلزایمـر مبـتلا می شوند، شبکه های عصبی مرتبط با حافظه و توجه کاهش می یابند.
٤. EEG: در ایـن روش، بـا اسـتفاده از الکترودهـایی کـه بـر روی سـر بیمـار قـرار مـی گیرنـد، فعالیـت هـای الکتریکـی مغز بیمار ثبت می شـود. بـا تحلیـل ایـن فعالیـت هـا، نشـان داده شـده اسـت کـه در بیمـاران MCI کـه در آینـده بـه تد ططآلزایمر مبتلا می شوند، فعالیت های الکتریکی مغز در برخی از مناطق کاهش می یابد.++

بـا توجـه بـه اینکـه هـر یـک از ایـن روش هـا دارای مزایـا و معایـب خـاص خـود هسـتند، بـرای پـیش بینـی تبـدیل MCI بـه آلزایمر، ممکن است نیاز به استفاده از ترکیبی از این روش ها باشد.

روش پیشنهادی مقاله

در مقاله مربوطه، یک CNN سه بعدی (3D-CNN) برای ترکیب و تجزیه و تحلیل مغناطیسی عملکردی حالت استراحت پیشنهاد شده است. تصویربرداری رزونانس (rs-fMRI)، نتایج ارزیابی بالینی و اطلاعات دموگرافیک برای پیش‌بینی تبدیل از MCI به آلزایمر در یک بازه متوسط ۵ ساله. در ابتدا، یک سی‌ان‌ان سه‌بعدی بر اساس حجم‌های منفرد fMRI از ۲۶۶ نمونه از۸۱ فرد; سپس، از لایه‌های عصبی برای ترکیب داده‌های بالینی با fMRI برای بهبود نتایج استفاده شده است.

روش اعمال الگوریتم بر روی داده‌های tumor

دیتاست tumor برای اعمال این الگوریتم انتخاب شده است. سپس داده ها را پیش‌پردازش می‌کنیم تا برای استفاده، آماده شده و ابعاد داده مناسب باشند.

پیاده سازی

برای کار بر روی دیتاست، به روش های مختلفی می‌توان داده را import کرد. یکی از این روش ها، upload کردن دیتاست روی google drive و mount کردن مسیر مربوط به دیتاست در کد است.

دیتاست تومور، شامل تصاویری از کلاس yes و no به همراه mask های آن‌ها می‌باشد.

```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

در مرحله بعد لازم است، package هایی که نیاز داریم را import کنیم.

```
import h5py
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, Dropout
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras import layers, models, optimizers
from tensorflow.keras.utils import to_categorical
from google.colab import drive
from tensorflow.keras.callbacks import ModelCheckpoint
from skimage.transform import resize
```

برای خواندنِ دیتاست و اعمال پیش‌پردازش بر روی تصاویر مربوط به تومورها، تابعی به نام load_and_preprocess_data تعریف می‌کنیم.

```
def load_and_preprocess_data(file_paths, target_size=(128, 128)):
    images = []
    labels = []
    tumor_masks = []

    for file_path in file_paths:
        with h5py.File(file_path, 'r') as file:
            data = file['cjdata']
            image = np.array(data['image'], dtype=np.float32) / 255.0
            label = data['label'][0, 0] - 1  # Convert labels to 0-indexing
            tumor_mask = np.array(data['tumorMask'], dtype=np.float32)

            # Resize the image to the target size
            image_resized = resize(image, target_size, anti_aliasing=True)
            mask_resized = resize(tumor_mask, target_size, anti_aliasing=True)

            images.append(image_resized)
            labels.append(label)
            tumor_masks.append(mask_resized)

    # Convert to NumPy arrays
    images = np.array(images)
    labels = np.array(labels)
    tumor_masks = np.array(tumor_masks)

    # Ensure the images have the correct shape
    images = images.reshape(images.shape[0], target_size[0], target_size[1], 1)
    tumor_masks = tumor_masks.reshape(tumor_masks.shape[0], target_size[0], target_size[1], 1)

    return images, labels ,tumor_masks
```

حال لازم است مسیری که فایل‌ها در آن ذخیره شده (google Drive) به عنوان ورودی به تابع load_and_preprocess_data داده شود و در ادامه این تایع فراخوانی شود.

```
# Get the list of file paths in the Google Drive folder
drive_folder = '/content/drive/MyDrive/figshare_dataset/data/'
file_paths = [drive_folder + f'{i}.mat' for i in range(1, 3065)]  # Assuming files are named 1.mat to 3064.mat

# Load and preprocess the data
images, labels, segmentation_masks = load_and_preprocess_data(file_paths)
```

بنابراین تا این مرحله، دیتاست خوانده شد و پیش‌پردازش بر روی آن انجام شد. حال می‌توان وارد مراحل آموزش مدل شویم.

ابتدا، داده‌ها را به با تسبت ۸۰ به ۲۰، به دو دسته‌ی داده‌های آموزش و داده‌های تست، تقسیم کنیم. همچنین از آنجایی که label های مربوط به دیتاست، غیرعددی هستند از one-hot encoding برای آنها استفاده می‌کنیم. (کدبندی وان هات روشی است که برای نمایش متغیرهای غیرعددی به صورت مقادیر عددی در مدل یادگیری ماشین استفاده می‌شود.)

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, random_state=42)

# Convert labels to one-hot encoding
label_encoder = LabelEncoder()
y_train_encoded = to_categorical(label_encoder.fit_transform(y_train))
y_test_encoded = to_categorical(label_encoder.transform(y_test))
```

حال می‌خواهیم یک شبکه عصبی کانولوشنال ساده برای دسته‌بندی تصاویر با ابعاد ۱۲۸x۱۲۸ و یک کانال (سیاه و سفید) بسازیم.

برای تعریف یک شبکه convolutionی از sequential در keras استفاده می‌کنیم. از ماژول Sequential برای مقداردهی‌های اولیه به شبکه عصبی مصنوعی استفاده می‌گردد، همچنین از ماژول Dense برای اینکه لایه‌های شبکه عصبی مصنوعی ساخته شود، استفاده می‌شود. حال نیاز است که شبکه عصبی مصنوعی را مقداردهی اولیه کنیم. این کار با ساختن یک نمونه از تابع Sequential انجام می‌گیرد. با کمک این تابع پشته خطی از لایه‌ها (keras layers) مقداردهی اولیه می‌شوند

ابتدا یک لایه کانولوشن با ۱۲۸ فیلتر و اندازه کرنل ۳x۳ قرار می‌دهیم که تابع فعال‌سازی relu برای اعمال تابع غیرخطی به خروجی هر نورون استفاده شده است. (ورودی به این لایه تصاویر با ابعاد (۱۲۸, ۱۲۸, ۱) است (تصاویر سیاه و سفید با ابعاد ۱۲۸x۱۲۸ پیکسل است).

در ادامه، از Max Pooling با ابعاد (۲, ۲) استفاده شده است که هر فاصله ۲x۲ از تصویر را به مقدار بزرگترین عنصر در آن ناحیه تبدیل می‌کند. و بعد از آن هم یک لایه کانولوشن دیگر با ۶۴ فیلتر و اندازه کرنل ۳x۳. و مجددا لایه Max Pooling دیگر با ابعاد (۲, ۲) برای کاهش ابعاد تصویر.

لایه Flatten برای تبدیل ماتریس چند بعدی به یک بردار یک بعدی استفاده می‌شود. در ادامه هم یک لایه کاملاً متصل (Fully Connected)با ۶۴ نورون و تابع فعال‌سازیrelu استفاده شده است. همچنین لایه Dropout با نرخ ۰.۲ برای جلوگیری از بیش‌برازش (overfitting) اضافه می‌شود.

پس از تکمیل مدل، لازم است compile کنیم و داده‌های آموزشش را به مدل بدهیم تا با آن‌ها آموزش ببیند.

در انتها نیز باید مدلِ آموزش دیده را توسط داده‌های تست، ارزیابی کنیم. برای بیان نتایج ارزیابی می‌توان از معیار دقت، استفاده کرد.

```python
# Define the CNN model
model = Sequential([
    Conv2D(128, kernel_size=3, activation='relu', input_shape=(128, 128, 1)),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, kernel_size=3, activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(3, activation='softmax')
])

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

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

# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(X_test, y_test_encoded)
print(f'Test Accuracy: {test_acc}')
```

```
Epoch 1/10
62/62 [==============================] - 178s 3s/step - loss: 0.8832 - accuracy: 0.6954 - val_loss: 0.4348 - val_accuracy: 0.8411
Epoch 2/10
62/62 [==============================] - 168s 3s/step - loss: 0.3107 - accuracy: 0.8679 - val_loss: 0.3421 - val_accuracy: 0.8452
Epoch 3/10
62/62 [==============================] - 178s 3s/step - loss: 0.2252 - accuracy: 0.9092 - val_loss: 0.3541 - val_accuracy: 0.8900
Epoch 4/10
62/62 [==============================] - 168s 3s/step - loss: 0.1671 - accuracy: 0.9337 - val_loss: 0.3121 - val_accuracy: 0.9063
Epoch 5/10
62/62 [==============================] - 178s 3s/step - loss: 0.1087 - accuracy: 0.9541 - val_loss: 0.4617 - val_accuracy: 0.8574
Epoch 6/10
62/62 [==============================] - 168s 3s/step - loss: 0.0898 - accuracy: 0.9663 - val_loss: 0.3806 - val_accuracy: 0.9226
Epoch 7/10
62/62 [==============================] - 168s 3s/step - loss: 0.0779 - accuracy: 0.9714 - val_loss: 0.3229 - val_accuracy: 0.9022
Epoch 8/10
62/62 [==============================] - 168s 3s/step - loss: 0.0457 - accuracy: 0.9862 - val_loss: 0.3824 - val_accuracy: 0.9267
Epoch 9/10
62/62 [==============================] - 178s 3s/step - loss: 0.0491 - accuracy: 0.9837 - val_loss: 0.7112 - val_accuracy: 0.8676
Epoch 10/10
62/62 [==============================] - 176s 3s/step - loss: 0.0943 - accuracy: 0.9643 - val_loss: 0.4808 - val_accuracy: 0.9165
20/20 [==============================] - 12s 606ms/step - loss: 0.3074 - accuracy: 0.9217
Test Accuracy: 0.9216966032981873
```

آنچه مشاهده می‌شود این است که مدل CNN طراحی شده، با دقت ۹۲ درصد می‌تواند label درست را پیش‌بینی کند.

استفاده از شبکه U-Net برای segmentation

در ادامه از شبکه U-Net که برای semantic segmentation انتخاب مناسبی است، بهره گرفتیم.

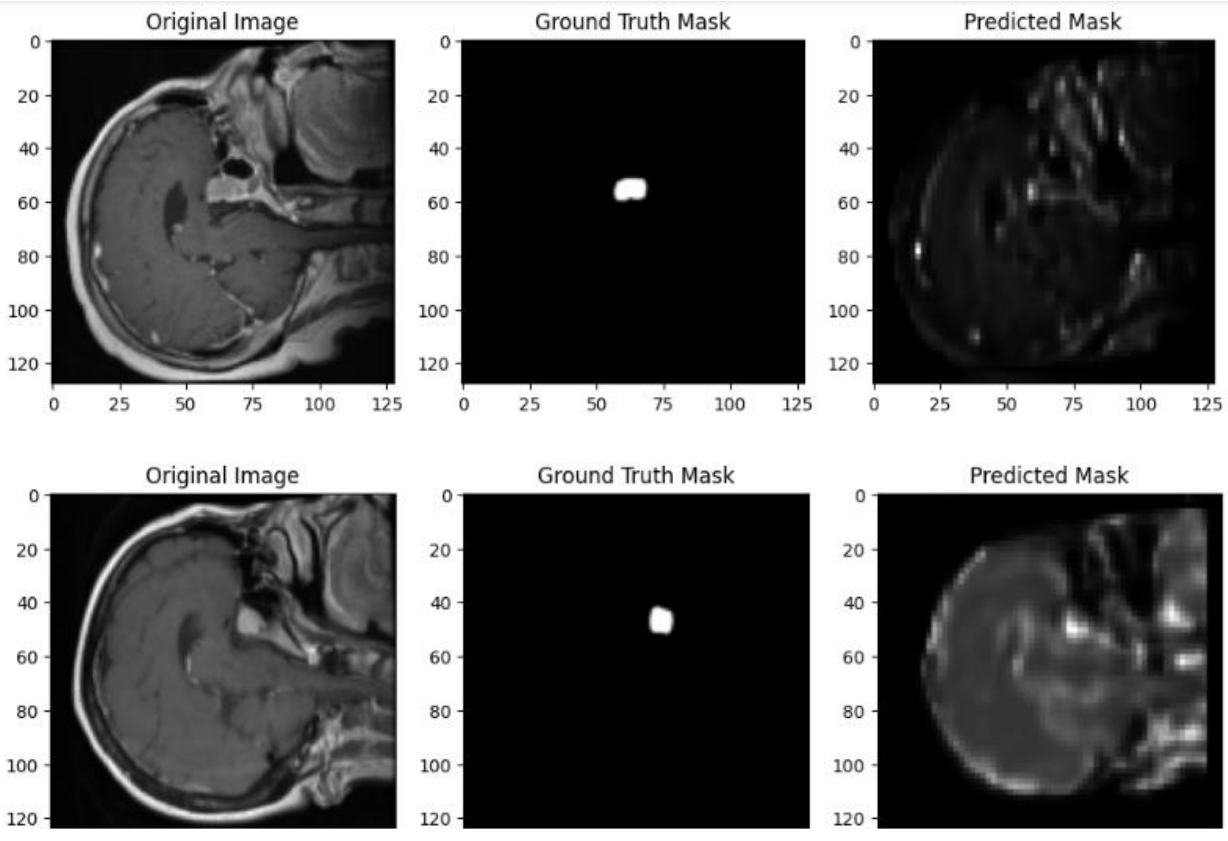
```
# Split the data into training and testing sets
X_train_seg, X_test_seg, y_train_seg, y_test_seg = train_test_split(images, segmentation_masks, test_size=0.2, random_state=42)

# Build a U-Net model for segmentation
def unet_model(input_shape):
    inputs = layers.Input(input_shape)

    # Encoder
    conv1 = layers.Conv2D(64, 3, activation='relu', padding='same')(inputs)
    conv1 = layers.Conv2D(64, 3, activation='relu', padding='same')(conv1)
    pool1 = layers.MaxPooling2D(pool_size=(2, 2))(conv1)

    # Decoder
    conv2 = layers.Conv2D(64, 3, activation='relu', padding='same')(pool1)
    conv2 = layers.Conv2D(64, 3, activation='relu', padding='same')(conv2)
    up1 = layers.UpSampling2D(size=(2, 2))(conv2)

    output = layers.Conv2D(1, 1, activation='sigmoid')(up1)

    model = models.Model(inputs, output)
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

    return model

# Instantiate the U-Net model
input_shape = (128, 128, 1)  # Assuming images are resized to 128x128
unet = unet_model(input_shape)

# Checkpoint to save the best model during training
checkpoint = ModelCheckpoint('unet_brain_tumor_segmentation.h5', save_best_only=True)
```

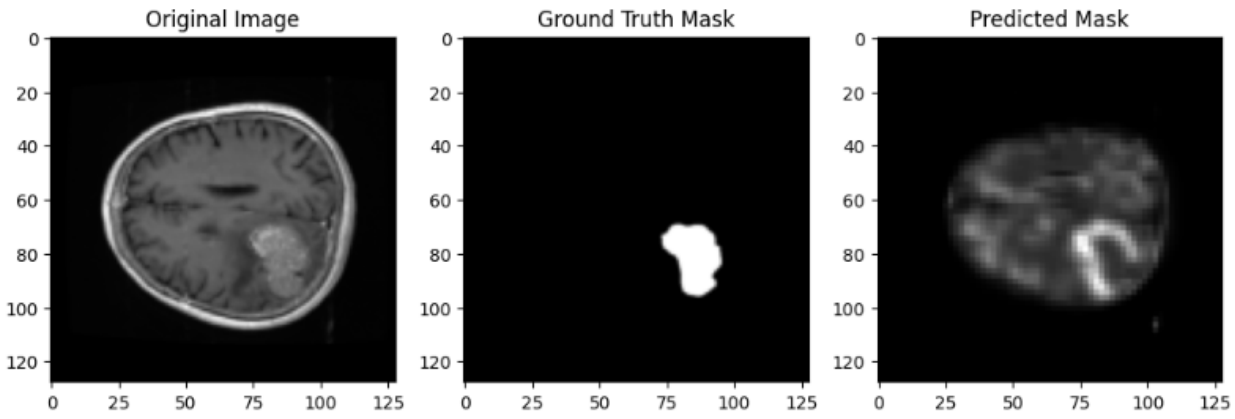
```
# Train the U-Net model for segmentation
unet.fit(X_train_seg, y_train_seg, epochs=10, batch_size=32, validation_split=0.2, callbacks=[checkpoint])

# Evaluate the U-Net model on the test set
test_loss, test_acc = unet.evaluate(X_test_seg, y_test_seg)
print(f'Test Accuracy: {test_acc}')
```

```
Epoch 1/10
62/62 [==============================] - 402s 6s/step - loss: 0.1657 - accuracy: 0.9608 - val_loss: 0.0744 - val_accuracy: 0.9748
Epoch 2/10
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered
  saving_api.save_model(
62/62 [==============================] - 391s 6s/step - loss: 0.0711 - accuracy: 0.9747 - val_loss: 0.0692 - val_accuracy: 0.9748
Epoch 3/10
62/62 [==============================] - 388s 6s/step - loss: 0.0715 - accuracy: 0.9747 - val_loss: 0.0700 - val_accuracy: 0.9748
Epoch 4/10
62/62 [==============================] - 381s 6s/step - loss: 0.0688 - accuracy: 0.9747 - val_loss: 0.0710 - val_accuracy: 0.9748
Epoch 5/10
62/62 [==============================] - 379s 6s/step - loss: 0.0686 - accuracy: 0.9747 - val_loss: 0.0724 - val_accuracy: 0.9748
Epoch 6/10
62/62 [==============================] - 364s 6s/step - loss: 0.0692 - accuracy: 0.9747 - val_loss: 0.0671 - val_accuracy: 0.9748
Epoch 7/10
62/62 [==============================] - 380s 6s/step - loss: 0.0672 - accuracy: 0.9747 - val_loss: 0.0664 - val_accuracy: 0.9748
Epoch 8/10
62/62 [==============================] - 377s 6s/step - loss: 0.0664 - accuracy: 0.9747 - val_loss: 0.0654 - val_accuracy: 0.9748
Epoch 9/10
62/62 [==============================] - 377s 6s/step - loss: 0.0650 - accuracy: 0.9747 - val_loss: 0.0662 - val_accuracy: 0.9748
Epoch 10/10
62/62 [==============================] - 361s 6s/step - loss: 0.0655 - accuracy: 0.9747 - val_loss: 0.0638 - val_accuracy: 0.9748
20/20 [==============================] - 29s 1s/step - loss: 0.0623 - accuracy: 0.9756
Test Accuracy: 0.9755972027778625
```

که به دقت ۹۷.۵ رسیده است. در ادامه عملکرد مدل U-Net بر روی چند نمونه از داده های تست و استخراج mask از آن‌ها را برای مشاهده نتایج segmentation به نمایش می‌گذاریم.

```python
# Make predictions on a few test samples
import matplotlib.pyplot as plt
num_samples = 3
test_samples = X_test_seg[:num_samples]
ground_truth_masks = y_test_seg[:num_samples]
predicted_masks = unet.predict(test_samples)

# Visualization
for i in range(num_samples):
    plt.figure(figsize=(12, 4))

    plt.subplot(1, 3, 1)
    plt.imshow(test_samples[i, ..., 0], cmap='gray')
    plt.title('Original Image')

    plt.subplot(1, 3, 2)
    plt.imshow(ground_truth_masks[i, ..., 0], cmap='gray')
    plt.title('Ground Truth Mask')

    plt.subplot(1, 3, 3)
    plt.imshow(predicted_masks[i, ..., 0], cmap='gray')
    plt.title('Predicted Mask')

    plt.show()
```




Original Image
Ground Truth Mask
Predicted Mask
0
20
40
60
80
100
120
0
25
50
75
100
125

استفاده از شبکه pretrain به نام ResNet

برای استفاده از شبکه‌های pre-train دو رویکرد کلی وجود دارد:

- استفاده برای feature extraction

از آنجایی که فرآیند آموزش، هزینه محاسباتی بالایی دارد و زمان‌بر است، از یک شبکه‌ی train شده استفاده می‌کنیم و فقط classifier خودمان را learn می‌کنیم.

- استفاده برای fine tuning

لایه‌ی classifier این شبکه‌ها را به طور کلی کنار می‌گذاریم ولی به تمام لایه‌های conv نیز دست نمی‌زنیم. برخی را freeze کرده و آن‌هایی را که می‌خواهیم تغییر بدهیم را unfreeze می‌کنیم و پارامترهای این لایه ها را آموزش می‌دهیم.

```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import h5py
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, Dropout
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras import layers, models, optimizers
from tensorflow.keras.utils import to_categorical
from google.colab import drive
from tensorflow.keras.callbacks import ModelCheckpoint
from skimage.transform import resize
```

```
    # Convert to NumPy arrays
    images = np.array(images)
    labels = np.array(labels)
    tumor_masks = np.array(tumor_masks)

    # Ensure the images have the correct shape
    images = images.reshape(images.shape[0], target_size[0], target_size[1], 1)
    tumor_masks = tumor_masks.reshape(tumor_masks.shape[0], target_size[0], target_size[1], 1)

    return images, labels ,tumor_masks

# Get the list of file paths in the Google Drive folder
drive_folder = '/content/drive/MyDrive/figshare_dataset/data/'
file_paths = [drive_folder + f'{i}.mat' for i in range(1, 3065)]  # Assuming files are named 1.mat to 3064.mat

# Load and preprocess the data
images, labels, segmentation_masks = load_and_preprocess_data(file_paths)
```

```
from tensorflow.keras.applications import ResNet50
import cv2
from keras.preprocessing import image
from keras.applications.resnet50 import preprocess_input, decode_predictions
import numpy as np
```

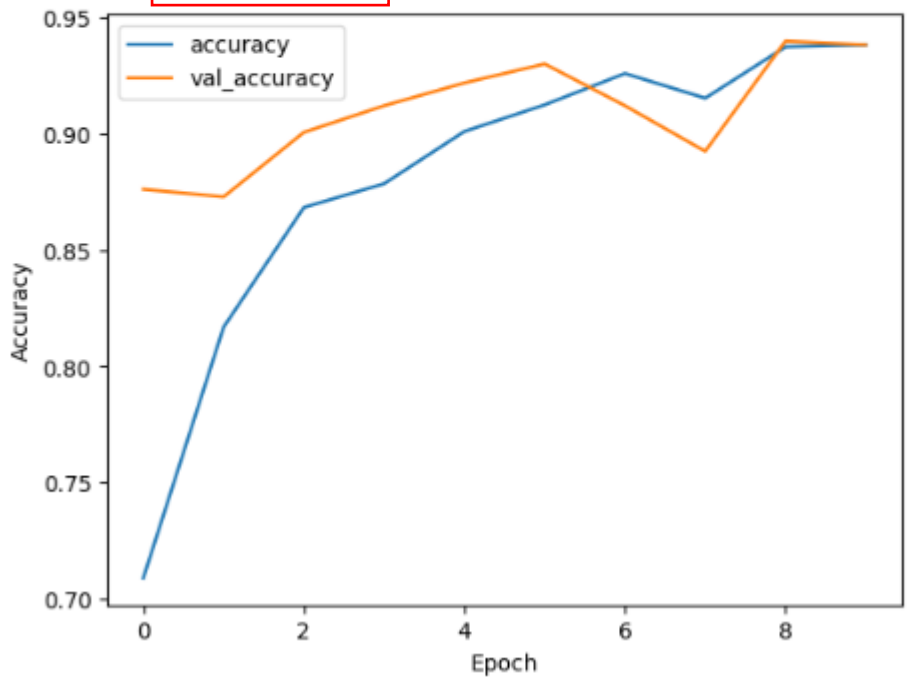
```
rgb_image = np.repeat(images, 3, axis=-1)
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(rgb_image, labels, test_size=0.2, random_state=42)

# Convert labels to one-hot encoding
label_encoder = LabelEncoder()
y_train_encoded = to_categorical(label_encoder.fit_transform(y_train))
y_test_encoded = to_categorical(label_encoder.transform(y_test))

# Define ResNet50 model using the pre-trained weights on ImageNet
base_model = ResNet50(input_shape=(224, 224, 3), include_top=False, weights='imagenet')

# Freeze convolutional layers
for layer in base_model.layers:
    layer.trainable = False

# Create a new model on top
model = models.Sequential()
model.add(base_model)
model.add(layers.Flatten())
model.add(layers.Dense(128, activation=('relu')))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(64, activation=('relu')))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(3, activation=('softmax')))
```

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

# Display the model architecture and layer sizes
model.summary()

# Train the model
history = model.fit(X_train, y_train_encoded, epochs=10, validation_data=(X_test, y_test_encoded))

# Print accuracy after each epoch
for epoch, acc in enumerate(history.history['accuracy']):
    print(f"Epoch {epoch + 1}/{len(history.history['accuracy'])} - Accuracy: {acc:.4f}")

# Plot accuracy
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 3)                 195

=================================================================
Total params: 36441347 (139.01 MB)
Trainable params: 12853635 (49.03 MB)
Non-trainable params: 23587712 (89.98 MB)
_________________________________________________________________
Epoch 1/10
77/77 [==============================] - 616s 8s/step - loss: 3.0975 - accuracy: 0.7091 - val_loss: 0.4582 - val_accuracy: 0.8760
Epoch 2/10
77/77 [==============================] - 588s 8s/step - loss: 0.5593 - accuracy: 0.8168 - val_loss: 0.3376 - val_accuracy: 0.8728
Epoch 3/10
77/77 [==============================] - 549s 7s/step - loss: 0.3481 - accuracy: 0.8682 - val_loss: 0.2939 - val_accuracy: 0.9005
Epoch 4/10
77/77 [==============================] - 548s 7s/step - loss: 0.3366 - accuracy: 0.8784 - val_loss: 0.2689 - val_accuracy: 0.9119
Epoch 5/10
77/77 [==============================] - 576s 8s/step - loss: 0.2723 - accuracy: 0.9009 - val_loss: 0.2108 - val_accuracy: 0.9217
Epoch 6/10
77/77 [==============================] - 585s 8s/step - loss: 0.2258 - accuracy: 0.9123 - val_loss: 0.1881 - val_accuracy: 0.9299
Epoch 7/10
77/77 [==============================] - 579s 8s/step - loss: 0.1900 - accuracy: 0.9257 - val_loss: 0.2392 - val_accuracy: 0.9119
Epoch 8/10
77/77 [==============================] - 545s 7s/step - loss: 0.2128 - accuracy: 0.9151 - val_loss: 0.3156 - val_accuracy: 0.8923
Epoch 9/10
77/77 [==============================] - 582s 8s/step - loss: 0.1579 - accuracy: 0.9372 - val_loss: 0.1911 - val_accuracy: 0.9396
Epoch 10/10
77/77 [==============================] - 578s 8s/step - loss: 0.1535 - accuracy: 0.9380 - val_loss: 0.1721 - val_accuracy: 0.9380
```

```
Epoch 1/10 - Accuracy: 0.7091
Epoch 2/10 - Accuracy: 0.8168
Epoch 3/10 - Accuracy: 0.8682
Epoch 4/10 - Accuracy: 0.8784
Epoch 5/10 - Accuracy: 0.9009
Epoch 6/10 - Accuracy: 0.9123
Epoch 7/10 - Accuracy: 0.9257
Epoch 8/10 - Accuracy: 0.9151
Epoch 9/10 - Accuracy: 0.9372
Epoch 10/10 - Accuracy: 0.9380
```



تمارین (امتیازی)

تمرین۱: اعمال دو فیلتر داده شده ۳*۳ بر روی یک تصویر ورودی با ابعاد ۸*۸. از آنجایی که قبل از اعمال فیلترها، از zero-padding استفاده نشده است، ابعاد تصویر خروجی ۶*۶ خواهد بود.

Exercise 1

```
import numpy as np
#from scipy.ndimage import convolve2d
from scipy.ndimage import convolve1d
from scipy import signal
```

```
matrix = np.array([[0,5,7,7,5,8,7,8],
                   [7,2,6,2,6,5,6,8],
                   [6,9,7,7,0,7,2,7],
                   [6,6,1,7,6,7,7,5],
                   [9,6,0,7,8,2,6,7],
                   [2,8,8,2,7,6,7,8],
                   [7,3,2,6,1,7,5,8],
                   [9,9,5,6,7,7,7,7]])
```

```
filter01 = np.array([[1,1,1],
                     [1,1,1],
                     [1,1,1]])
filter02 = np.array([[2,3,2],
                     [-1,0,-1],
                     [2,3,2]])
```

```
new_matrix01 = signal.convolve2d(matrix,filter01,mode='valid')
new_matrix02 = signal.convolve2d(matrix,filter02,mode='valid')
```

```
print(new_matrix01)
print('////////////////////////////')
print(new_matrix02)
```

نتیجه پیاده سازی تمرین ۱:

```
[[49 52 47 47 46 58]
 [50 47 42 47 46 54]
 [50 50 43 51 45 50]
 [46 45 46 52 56 55]
 [45 42 41 46 49 56]
 [53 49 44 49 54 62]]
////////////////////////////
[[69 94 68 66 61 74]
 [51 39 58 64 84 75]
 [82 66 65 56 46 58]
 [67 60 63 74 79 85]
 [53 40 46 63 53 67]
 [90 80 75 71 89 83]]
```

تمرین۲: اعمال فیلتر لبه یابی و مقایسه آن با زمانی که خود تصویر نیز به لبه‌ها اضافه شود بر روی تصویر ورودی به همراه zero-padding:

Exercise 2

```
[ ]  import numpy as np
     #from scipy.ndimage import convolve2d
     from scipy.ndimage import convolve1d
     from scipy import signal
```

```
[ ]  matrix = np.array([[0,5,7,7,5,8,7,8],
                        [7,2,6,2,6,5,6,8],
                        [6,9,7,7,0,7,2,7],
                        [6,6,1,7,6,7,7,5],
                        [9,6,0,7,8,2,6,7],
                        [2,8,8,2,7,6,7,8],
                        [7,3,2,6,1,7,5,8],
                        [9,9,5,6,7,7,7,7]])
```

```
[ ]  matrix02 = np.array([[0,0,0,0,0,0,0,0,0,0],
                        [0,0,5,7,7,5,8,7,8,0],
                        [0,7,2,6,2,6,5,6,8,0],
                        [0,6,9,7,7,0,7,2,7,0],
                        [0,6,6,1,7,6,7,7,5,0],
                        [0,9,6,0,7,8,2,6,7,0],
                        [0,2,8,8,2,7,6,7,8,0],
                        [0,7,3,2,6,1,7,5,8,0],
                        [0,9,9,5,6,7,7,7,7,0],
                        [0,0,0,0,0,0,0,0,0,0]])
```

```
[ ]  filter01 = np.array([[0,1,0],
                        [1,-4,1],
                        [0,1,0]])
     filter02 = np.array([[0,-1,0],
                        [-1,5,-1],
                        [0,-1,0]])
```

```
[ ]  new_matrix03 = signal.convolve2d(matrix02,filter02,mode='valid')
     new_matrix04 = signal.convolve2d(matrix02,filter01,mode='valid')
     new_matrix05 = matrix - new_matrix04
```

```
[ ]  print(new_matrix03)
     print('////////////////////////////')
     #print(new_matrix04)
     print('////////////////////////////')
     print(new_matrix05)
```

```
[[-12  16  17  21   4  23  13  25]
 [ 27 -17  12 -16  18  -2   8  19]
 [  8  24  12  19 -26  21 -17  20]
 [  9   8 -15  14   8  13  15   4]
 [ 31   7 -22  18  18 -17   7  16]
 [-14  21  28 -18  18   7  10  18]
 [ 21 -11 -12  19 -22  16  -4  20]
 [ 29  28   8  12  21  14  16  20]]
////////////////////////////
////////////////////////////
[[-12  16  17  21   4  23  13  25]
 [ 27 -17  12 -16  18  -2   8  19]
 [  8  24  12  19 -26  21 -17  20]
 [  9   8 -15  14   8  13  15   4]
 [ 31   7 -22  18  18 -17   7  16]
 [-14  21  28 -18  18   7  10  18]
 [ 21 -11 -12  19 -22  16  -4  20]
 [ 29  28   8  12  21  14  16  20]]
```

تمرین ۳: اعمال فیلتر میانگین نرمالیز شده (که یک فیلتر هموارساز خطی است) بر روی تصویر ورودی.

انتظار می‌رود لبه‌های تیز در شدت، کاهش یابند. تصویر کمی blur می‌شود چون لبه‌ها کمی محور می‌شوند.

Exercise 3

```
[ ]  matrix02 = np.array([[0,0,0,0,0,0,0,0,0,0],
                          [0,0,5,7,7,5,8,7,8,0],
                          [0,7,2,6,2,6,5,6,8,0],
                          [0,6,9,7,7,0,7,2,7,0],
                          [0,6,6,1,7,6,7,7,5,0],
                          [0,9,6,0,7,8,2,6,7,0],
                          [0,2,8,8,2,7,6,7,8,0],
                          [0,7,3,2,6,1,7,5,8,0],
                          [0,9,9,5,6,7,7,7,7,0],
                          [0,0,0,0,0,0,0,0,0,0]])
```

```
[ ]  filter03 =(1/9) * np.array([[1,1,1],
                                 [1,1,1],
                                 [1,1,1]])
```

```
[ ]  new_matrix06 = signal.convolve2d(matrix02,filter03,mode='valid')
```

```
[ ]  new_matrix07 = matrix - new_matrix06
```

```
[ ]  new_matrix08 = new_matrix07 + matrix
```

```
[ ]  print(new_matrix08)
```

```
[[-1.55555556  7.         10.77777778 10.33333333  6.33333333 11.88888889
   9.33333333 12.77777778]
 [10.77777778 -1.44444444  6.22222222 -1.22222222  6.77777778  4.88888889
   5.55555556 11.77777778]
 [ 8.         12.44444444  8.77777778  9.33333333 -5.22222222  8.88888889
  -2.         10.11111111]
 [ 7.33333333  6.44444444 -3.55555556  9.22222222  6.33333333  9.
   8.44444444  6.22222222]
 [13.88888889  6.88888889 -5.          8.88888889 10.22222222 -2.22222222
   5.88888889  9.55555556]
 [ 0.11111111 11.         11.33333333 -0.55555556  8.88888889  6.55555556
   7.77777778 11.44444444]
 [ 9.77777778  0.11111111 -1.44444444  7.11111111 -3.44444444  8.
   3.11111111 11.33333333]
 [14.88888889 14.11111111  6.55555556  9.         10.22222222 10.22222222
   9.44444444 11.        ]]
```

تمرین۴: از یک تصویر benchmark به عنوان تصویر ورودی استفاده کنید. تبدیل فوریه روی عکس ورودی اعمال شود (مقادیر را ببینید). سپس عکس تبدیل فوریه بگیرید و با عکس original مقایسه کنید. (برای مقایسه، دو تصویر را میتوان از هم کم کرد. اعداد حاصل بسیار کوچک هستند، عملاً نزدیک به صفر) از تابع fftshift استفاده می‌کنیم که هر چهار تابع را shift دهد و مبدأ همه به نقطه وسط منتقل می‌شود. تابع abs طیف تصویر را می‌دهد.

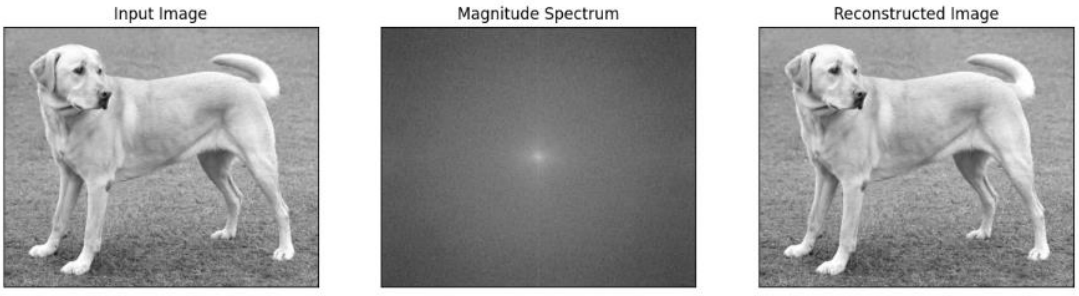
Exercise 4

```
[ ] from google.colab import drive
    drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

```
[ ] from PIL import Image
    img = Image.open("/content/drive/My Drive/dog.jpg")
```

```
[ ] import cv2
    import numpy as np
    from matplotlib import pyplot as plt

    image = cv2.imread("/content/drive/My Drive/dog.jpg", 0)  # Load the image in grayscale
    img = image

    f_transform = np.fft.fft2(img)
    f_transform_shifted = np.fft.fftshift(f_transform)
    magnitude_spectrum = 20 * np.log(np.abs(f_transform_shifted))

    f_inverse = np.fft.ifft2(f_transform)
    f_inverse = np.abs(f_inverse)

    plt.figure(figsize=(20,20))
    plt.subplot(141),plt.imshow(img, cmap = 'gray')
    plt.title('Input Image'), plt.xticks([]), plt.yticks([])
    plt.subplot(142), plt.imshow(magnitude_spectrum, cmap='gray')
    plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
    plt.subplot(143),plt.imshow(f_inverse, cmap = 'gray')
    plt.title('Reconstructed Image'), plt.xticks([]), plt.yticks([])
    plt.show()
```




```
[ ] img
```

```
array([[158, 153, 148, ..., 129, 122, 120],
       [155, 152, 149, ..., 132, 130, 130],
       [149, 150, 151, ..., 135, 138, 139],
       ...,
       [ 61,  65,  89, ...,  61,  80, 129],
       [124, 104, 142, ...,  69,  88, 134],
       [161, 183, 158, ...,  67,  81, 111]], dtype=uint8)
```

```
[ ] f_transform
```

```
array([[ 6.49685830e+07+3.85625754e-10j, -1.91342309e+04+4.43889063e+05j,
        -1.58368448e+06+9.75233775e+05j, ...,
        -9.99375147e+05-6.09734118e+05j, -1.58368448e+06-9.75233775e+05j,
        -1.91342309e+04-4.43889063e+05j],
       [-2.34273555e+06-4.15220217e+06j,  9.26592170e+05+7.51458667e+05j,
        -5.38927128e+04-7.67378956e+05j, ...,
         5.47056976e+05-6.91278994e+04j, -1.22428728e+05+1.17049750e+06j,
        -4.06639283e+05+1.36693870e+06j],
       [-1.55254718e+06-2.11442420e+06j,  4.87810491e+05+1.17866189e+04j,
        -2.38256461e+05+3.74311169e+05j, ...,
        -1.56202176e+05-1.23901302e+05j,  1.50213701e+05+2.97290122e+05j,
         1.07204600e+06+6.75895324e+05j],
       ...,
       [-2.54545526e+05+1.00662706e+06j, -3.82409639e+05+2.18422398e+05j,
         7.66868135e+05-6.14557120e+05j, ...,
        -3.50963359e+04+8.10482853e+05j,  4.84937054e+05-3.74276122e+05j,
        -7.40423891e+05-3.96968279e+04j],
       [-1.55254718e+06+2.11442420e+06j,  1.07204600e+06-6.75895324e+05j,
         1.50213701e+05-2.97290122e+05j, ...,
         1.98556175e+05+1.58173355e+05j, -2.38256461e+05-3.74311169e+05j,
         4.87810491e+05-1.17866189e+04j],
       [-2.34273555e+06+4.15220217e+06j, -4.06639283e+05-1.36693870e+06j,
        -1.22428728e+05-1.17049750e+06j, ...,
        -4.54356099e+05-8.91560527e+05j, -5.38927128e+04+7.67378956e+05j,
         9.26592170e+05-7.51458667e+05j]])
```

```
[ ]  f_inverse
```

```
array([[158., 153., 148., ..., 129., 122., 120.],
       [155., 152., 149., ..., 132., 130., 130.],
       [149., 150., 151., ..., 135., 138., 139.],
       ...,
       [ 61.,  65.,  89., ...,  61.,  80., 129.],
       [124., 104., 142., ...,  69.,  88., 134.],
       [161., 183., 158., ...,  67.,  81., 111.]])
```

```
[ ]  img - f_inverse
```

```
array([[ 2.84217094e-14,  2.84217094e-14,  5.68434189e-14, ...,
         2.84217094e-14,  4.26325641e-14,  5.68434189e-14],
       [-8.52651283e-14,  0.00000000e+00,  0.00000000e+00, ...,
        -5.68434189e-14, -2.84217094e-14, -2.84217094e-14],
       [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
        -2.84217094e-14, -5.68434189e-14, -2.84217094e-14],
       ...,
       [-9.94759830e-14, -4.26325641e-14, -1.27897692e-13, ...,
        -2.13162821e-14, -5.68434189e-14, -2.84217094e-14],
       [ 5.68434189e-14,  9.94759830e-14,  2.84217094e-14, ...,
         7.10542736e-14,  2.84217094e-14,  5.68434189e-14],
       [ 0.00000000e+00,  2.84217094e-14,  5.68434189e-14, ...,
         1.42108547e-14,  5.68434189e-14, -2.84217094e-14]])
```

تمرین ۵: بررسی فاز و طیف در تبدیل فوریه

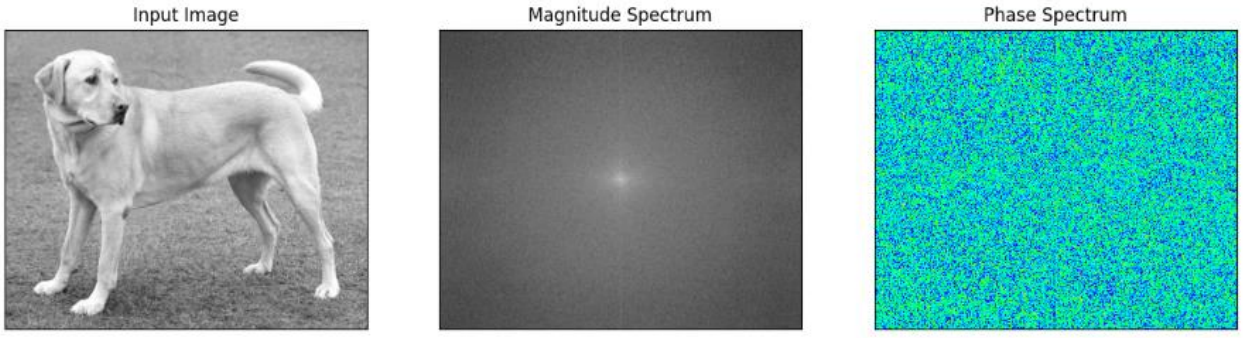
فاز، اطلاعات تأخیر مولفه‌ها نسبت به هم است ولی در قالب یک تصویر، قابل توصیف نیست و برفکی مانند است لذا با وجود اینکه اطلاعات مهمی دارد ولی در نمایش قابل درک نیست.

Exercise 5

```
[ ] from google.colab import drive
    drive.mount('/content/drive/')
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive.mount("/content/drive/", force_remount=True).
```

```
[ ]
    import cv2
    import numpy as np
    import matplotlib.pyplot as plt

    image = cv2.imread("/content/drive/My Drive/dog.jpg", 0)

    f_transform = np.fft.fft2(image)
    f_transform_shifted = np.fft.fftshift(f_transform)
    magnitude_spectrum = 20 * np.log(np.abs(f_transform_shifted))
    phase_spectrum = np.angle(f_transform_shifted)

    plt.figure(figsize=(15, 15))
    plt.subplot(131), plt.imshow(image, cmap='gray')
    plt.title('Input Image'), plt.xticks([]), plt.yticks([])
    plt.subplot(132), plt.imshow(magnitude_spectrum, cmap='gray')
    plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
    plt.subplot(133), plt.imshow(phase_spectrum, cmap='hsv')
    plt.title('Phase Spectrum'), plt.xticks([]), plt.yticks([])
    plt.show()
```



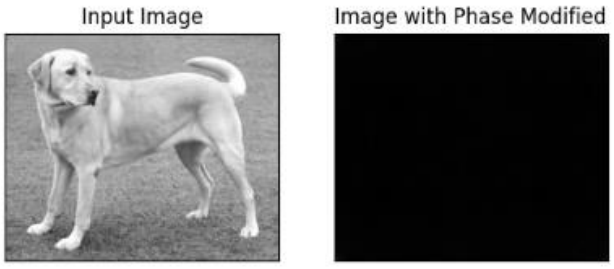
تمرین ۶: اصلاح تصاویر با استفاده از تغییر مقادیر فاز و طیف در تبدیل فوریه.

Exercise 6

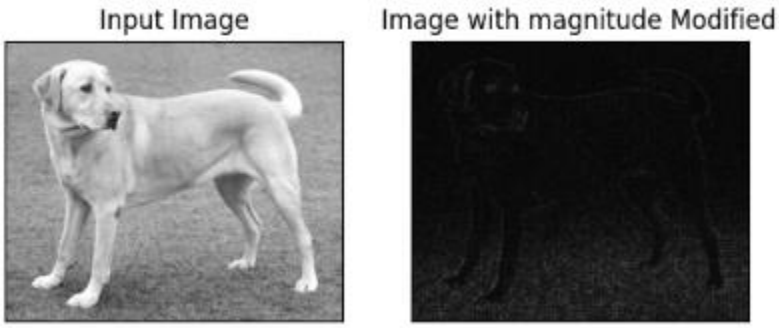
```
[ ]  from google.colab import drive
     drive.mount('/content/drive/')
```

```
[ ]
     import cv2
     import numpy as np
     from matplotlib import pyplot as plt

     image = cv2.imread("/content/drive/My Drive/dog.jpg", 0)

     fourier_transform = np.fft.fft2(image)
     fshift = np.fft.fftshift(fourier_transform)

     magnitude_spectrum = np.abs(fshift)
     phase_spectrum = np.angle(fshift)
     phase_spectrum[:] = 1  # تنظیم تمام مقادیر فاز به یک


     new_fshift = magnitude_spectrum * np.exp(1j * phase_spectrum)
     new_image = np.fft.ifft2(np.fft.ifftshift(new_fshift))
     new_image = np.abs(new_image)

     plt.subplot(121), plt.imshow(image, cmap='gray')
     plt.title('Input Image'), plt.xticks([]), plt.yticks([])
     plt.subplot(122), plt.imshow(new_image, cmap='gray')
     plt.title('Image with Phase Modified'), plt.xticks([]), plt.yticks([])
     plt.show()
```

[ ]



```
[ ]
    import cv2
    import numpy as np
    from matplotlib import pyplot as plt

    image = cv2.imread("/content/drive/My Drive/dog.jpg", 0)

    fourier_transform = np.fft.fft2(image)
    fshift = np.fft.fftshift(fourier_transform)

    magnitude_spectrum = np.abs(fshift)
    phase_spectrum = np.angle(fshift)
    magnitude_spectrum[:] = 1  # تنظیم تمام مقادیر طیف به یک

    new_fshift = magnitude_spectrum * np.exp(1j * phase_spectrum)
    new_image = np.fft.ifft2(np.fft.ifftshift(new_fshift))
    new_image = np.abs(new_image)

    plt.subplot(121), plt.imshow(image, cmap='gray')
    plt.title('Input Image'), plt.xticks([]), plt.yticks([])
    plt.subplot(122), plt.imshow(new_image, cmap='gray')
    plt.title('Image with magnitude Modified'), plt.xticks([]), plt.yticks([])
    plt.show()
```



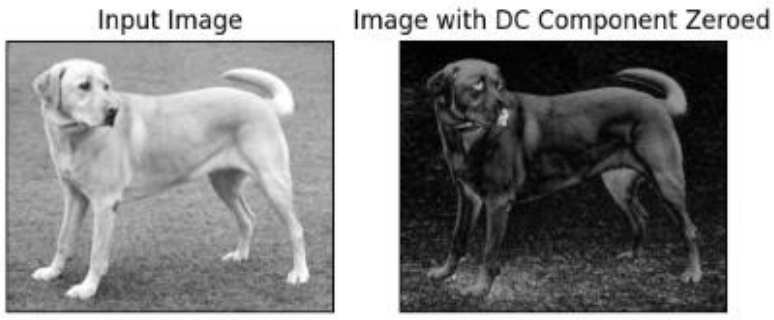
تمرین ۷: بررسی نقش ترم dc: در تبدیل فوریه، مولفه dc در مرکز یعنی نقطه (0,0) است. بعد از به دست آوردن dc آن را صفر کنید و بعد دوباره تصویر را بازسازی کنید. نتیجه را مشاهده نمایید.

Exercise 7

```
[ ] from google.colab import drive
    drive.mount('/content/drive/')
```

```
[ ] import cv2
    import numpy as np
    from matplotlib import pyplot as plt

    image = cv2.imread("/content/drive/My Drive/dog.jpg", 0)

    fourier_transform = np.fft.fft2(image)
    fshift = np.fft.fftshift(fourier_transform)

    rows, cols = image.shape
    mid_row, mid_col = int(rows/2), int(cols/2)
    dc_value = np.abs(fshift[mid_row, mid_col])

    print("مقدار مولفه DC: ", dc_value)

    # صفر کردن مقدار مولفه DC
    fshift[mid_row, mid_col] = 0

    new_fshift = np.fft.ifftshift(fshift)
    new_image = np.fft.ifft2(new_fshift)
    new_image = np.abs(new_image)

    plt.subplot(121), plt.imshow(image, cmap='gray')
    plt.title('Input Image'), plt.xticks([]), plt.yticks([])
    plt.subplot(122), plt.imshow(new_image, cmap='gray')
    plt.title('Image with DC Component Zeroed'), plt.xticks([]), plt.yticks([])
    plt.show()
```

```
مقدار مولفه DC:  64968583.0
```



تمرین۸: اسلاید ۲۱ در chapter4 مطابق مراحل زیر انجام شده است.

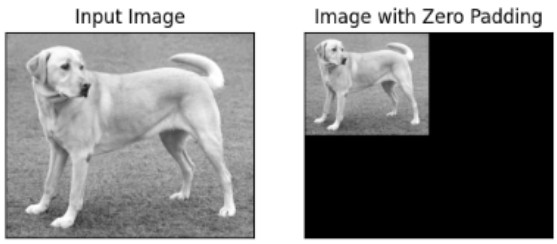
1. به تصویر ورودی $f(x,y)$ به اندازه $M \times N$، صفرهای ضروری (zero padding) را به تصویر اضافه کنید تا تصویر به اندازه $P \times Q$ درآید ($P = 2M, Q = 2N$)
2. تصویر $f(x,y)$ را در $(-1)^{x+y}$ ضرب کنید تا مرکز تبدیل تعیین شود
3. DFT مربوط به تصویر یعنی $F(u,v)$ را محاسبه کنید
4. یک تابع فیلتر متقارن به نام $H(u,v)$ به اندازه $P \times Q$ ایجاد کنید که مختصات مرکز آن $(P/2,Q/2)$ باشد. با استفاده از ضرب آرایه ها $G(u,v)$ را محاسبه کنید

$$G(u,v)=H(u,v)F(u,v)$$

Exercise 8

```
import numpy as np
image = cv2.imread("/content/drive/My Drive/dog.jpg", 0)
def zero_pad_image(image, p, q):
    m, n = image.shape
    padded_image = np.zeros((p, q))
    padded_image[:m, :n] = image
    return padded_image

M, N = image.shape
p = 2 * M
q = 2 * N
padded_image = zero_pad_image(image, p, q)
```

```
plt.subplot(121), plt.imshow(image, cmap='gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(padded_image, cmap='gray')
plt.title('Image with Zero Padding'), plt.xticks([]), plt.yticks([])
plt.show()
```

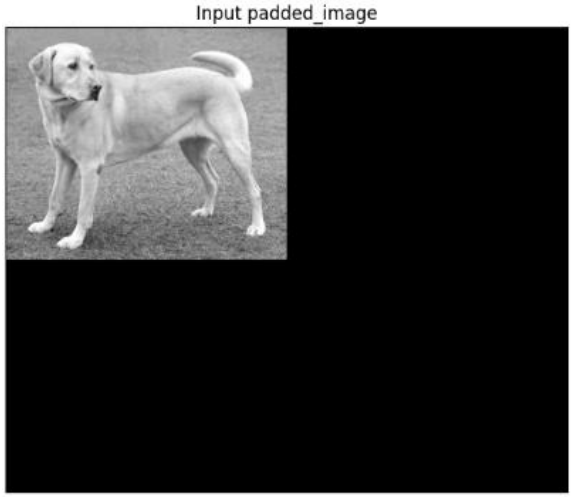



```
def find_image_center(image):
    rows, cols = image.shape
    center_row = rows // 2
    center_col = cols // 2
    return (center_row, center_col)

center = find_image_center(padded_image)
print("The center of the image is located at pixel coordinates:", center)
```

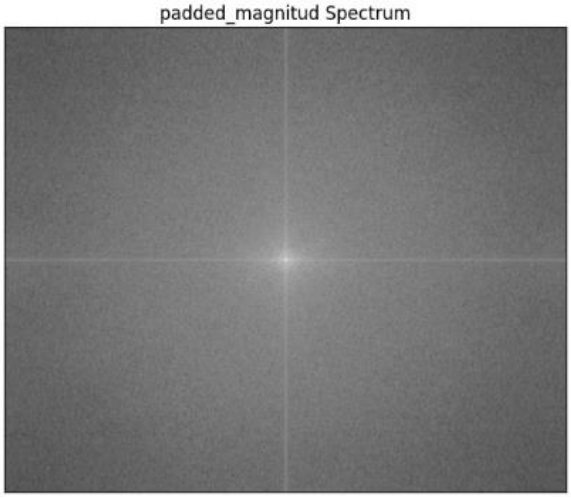
```
The center of the image is located at pixel coordinates: (577, 700)
```

```
import cv2
padded_f_transform = np.fft.fft2(padded_image)
padded_f_transform_shifted = np.fft.fftshift(padded_f_transform)
padded_magnitude_spectrum = 20 * np.log(np.abs(padded_f_transform_shifted))

plt.figure(figsize=(15, 15))
plt.subplot(121), plt.imshow(padded_image, cmap='gray')
plt.title('Input padded_image'), plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(padded_magnitude_spectrum, cmap='gray')
plt.title('padded_magnitud Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```

Input padded_image

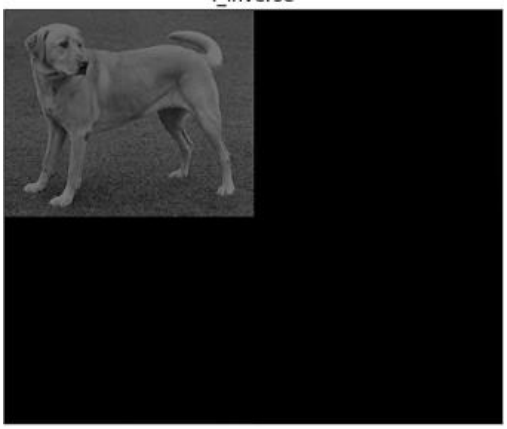


padded_magnitud Spectrum

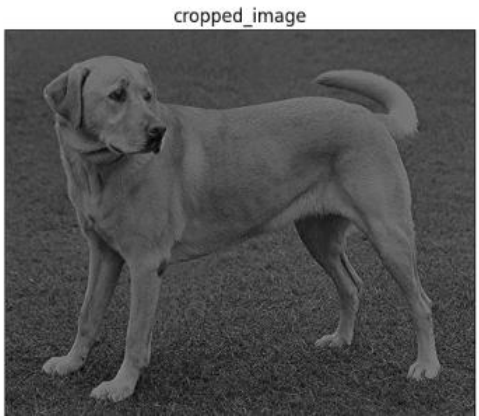


```
[ ]  import numpy as np

     def create_padded_gaussian_filter(p, q, sigma):
         size = (p, q)
         symmetrical_filter = np.zeros(size)
         center_i, center_j = p // 2, q // 2
         for i in range(p):
             for j in range(q):
                 symmetrical_filter[i, j] = np.exp(-((i - center_i)**2 + (j - center_j)**2) / (2 * sigma**2))
         symmetrical_filter /= np.sum(symmetrical_filter)  # Normalize the filter
         return symmetrical_filter


     def apply_filter_to_spectrum(image_spectrum, symmetrical_filter):
         filtered_spectrum = np.multiply(image_spectrum, symmetrical_filter)
         return filtered_spectrum

     sigma = 400
     p, q = padded_f_transform.shape
     symmetric_filter = create_padded_gaussian_filter(p, q, sigma)
     filtered_image_spectrum = apply_filter_to_spectrum(padded_f_transform, symmetric_filter)

     f_inverse = np.fft.ifft2(filtered_image_spectrum)
     f_inverse = np.abs(f_inverse)

     plt.imshow(f_inverse, cmap='gray')
     plt.title('f_inverse'), plt.xticks([]), plt.yticks([])
     plt.show()
```

f_inverse



```
[ ]  height, width = f_inverse.shape
     cropped_image = f_inverse[0:height//2, 0:width//2]

     plt.imshow(cropped_image, cmap='gray')
     plt.title('cropped_image'), plt.xticks([]), plt.yticks([])
     plt.show()
```



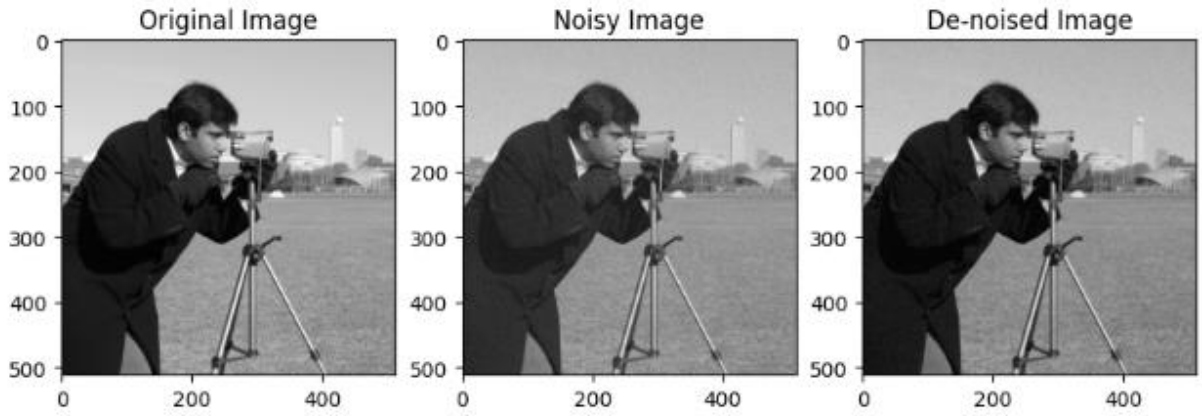
تمرین ۹ (نمره اضافی): یک تصویر را به عنوان تصویر ورودی انتخاب کنید و به کمک تابع داده شده در اسلاید ۵۴ فصل۵ تخریب کنید (H در حوزه فوریه است- مانند یک تصویر است، تولید کنید)

$$g(x,y)=\int_0^T f[x-x_0(t),y-y_0(t)]dt$$

سپس در تبدیل فوریه‌ی عکس، نظیر به نظیر ضرب نمایید (هر دو شیفت پیدا می‌کند) (توجه: مرکز تصویر مرجع فرکانس شده باشد، از ffshift استفاده می‌کنیم). پس از اعمال ضرب، عکس تبدیل فوریه می‌گیریم تا در حوزه مکان بتوان تخریب را مشاهده کرد.

```
[ ]  import numpy as np
     import matplotlib.pyplot as plt
     from scipy.signal import wiener
     from skimage import data
     from skimage.color import rgb2gray

     original_image = data.camera()

     gray_image = original_image

     noise = np.random.normal(0, 10, gray_image.shape)
     noisy_image = gray_image + noise


     f = np.fft.fft2(noisy_image)
     fshift = np.fft.fftshift(f)

     h, w = noisy_image.shape
     downsampling_filter = np.ones((h, w))
     import math
     k = 0.00005
     e = math.e
     for i in range(h):
       for j in range(w):
         downsampling_filter[i][j] = np.power(e,(-k * np.power((np.power(i-(h/2),2))+(np.power(j-(w/2),2)),(5/6))))

     fshift_downsampled = fshift * downsampling_filter

     f_ishift = np.fft.ifftshift(fshift_downsampled)
     image_back = np.fft.ifft2(f_ishift)
     image_back = np.abs(image_back)
```

```
plt.figure(figsize=(10, 5))

plt.subplot(131)
plt.imshow(original_image, cmap='gray')
plt.title('Original Image')

plt.subplot(132)
plt.imshow(noisy_image, cmap='gray')
plt.title('Noisy Image')

plt.subplot(133)
plt.imshow(image_back, cmap='gray')
plt.title('De-noised Image')

plt.show()
```


Original Image
Noisy Image
De-noised Image
0
100
200
300
400
500
0
200
400

تمرین ۱۰ (نمره اضافی): اعمال فیلتر Wiener بر روی یک عکس. اینجا از یک تصویر benchmark استفاده شده است.

```
[ ]  import numpy as np
     import matplotlib.pyplot as plt
     from scipy.signal import wiener
     from skimage import data
     from skimage.color import rgb2gray

     original_image = data.camera()

     gray_image = original_image

     noise = np.random.normal(0, 10, gray_image.shape)
     noisy_image = gray_image + noise

     de_noised_image = wiener(noisy_image, (5, 5))

     plt.figure(figsize=(10, 5))

     plt.subplot(131)
     plt.imshow(original_image, cmap='gray')
     plt.title('Original Image')

     plt.subplot(132)
     plt.imshow(noisy_image, cmap='gray')
     plt.title('Noisy Image')

     plt.subplot(133)
     plt.imshow(de_noised_image, cmap='gray')
     plt.title('De-noised Image')

     plt.show()
```