a. How did you select the slice to be analyzed in Q3a (3 pts)?

```python
class Normalization(object):

    def __call__(self, image): # 定義有image之後要進行的資料處理
        changed_image = (image - image.min())/(image.max() - image.min())
        return changed_image

def build_transform(is_train):
    """
    Create a data transformation pipeline for image preprocessing in deep learning tasks.
    """
    t = []
    if is_train:
        # 若是包含 Random 系列的圖像增強，則只能加在train裡面
        t.append(v2.RandomRotation(degrees=5))
        # t.append(v2.RandomHorizontalFlip())
        t.append(v2.CenterCrop(size = (50, 50)))
        t.append(Normalization())
        return v2.Compose(t)

    t.append(v2.CenterCrop(size = (50, 50)))
    t.append(Normalization())
    return v2.Compose(t)
```

取在畫面中間 50*50 的部分

b. How did you can represent three channels of our grayscale medical images for pre-trained architectures (3 pts)?

```python
def build_transform_ViT(is_train):
    """
    Create a data transformation pipeline for image preprocessing in deep learning tasks.
    """
    t = []
    if is_train:
        t.append(v2.Grayscale(num_output_channels=1))  # Convert to grayscale
        t.append(v2.CenterCrop(size=(56, 56)))
        t.append(Normalization())
        return v2.Compose(t)
    t.append(v2.Grayscale(num_output_channels=1))  # Convert to grayscale (1 channel)
    t.append(v2.CenterCrop(size=(56, 56)))      # Crop to 56×56
    t.append(Normalization())
    return v2.Compose(t)
```

用 v2.grayscale()

c. How did you incorporate age and gender into the model you used (Q3c) (4 pts)?

```python
def forward(self, x, age, gender):
    output = self.backend(x)  # VGG 提取特徵
    output = torch.cat([output, age.view(-1, 1), gender.view(-1, 1)], dim=1)  # 拼接年齡和性別
    output = self.classifier(output)  # 分類器
    return output
```

用 torch.cat()

d. Reproductivity of the results (4 pts)

因為程式碼有 random，所以每次都有點不同，但是結果大致相同

e. Number of parameters (2 pts) (Please write the parameter count of the final selected model.)

```
myparameter = sum(p.numel() for p in model.parameters() if p.requires_grad)
myparameter
```
✓ 0.0s
```
14847299
```

我使用 VGG，parameters=14847299

f. The difficulty during training (6 pts)

了解其中的結構並且試圖改進模型最困難，常常不小心就把模型弄壞了，要很理解架構才能改，而且改之後的效果也沒有原本的好

g. Briefly explain the structures of the models you are using (You are required to do at least VGGNet, ResNet, and ViT) (8 pts)

如圖上 code 的註解:

```python
class VGGplus(nn.Module):
    def __init__(self, num_classes, input_size = (3, 50, 50), features_grad = False): # 默認輸出為分到每一類的機率
        super().__init__()
        # 取出 vgg16 中的特徵層
        vgg16 = models.vgg16(weights = 'IMAGENET1K_V1', progress = True) # progress = True：顯示下載進度條。
        # nn.Identity() 是 Pytorch 中的特殊層，不會改變輸入，只是作為佔位符。這樣你以方便修改模型結構而不會影響其整體的輸出 size
        vgg16.features[30] = nn.Identity()
        # 將 VGG16 的輸出特徵圖壓縮 1x1，可以得到一個 512 維的特徵向量，便於與後面的 Fully Connected Layer 結合並處理輸入的不同尺寸的圖像。
        vgg16.avgpool = nn.AdaptiveAvgPool2d(1)
        # 移除 VGG16 預設的分類器
        vgg16.classifier = nn.Identity()
        # 固定/不固定特徵層的參數值: False/True
        # features_grad: 控制是否對 VGG16 的特徵提取層進行微調(參與梯度計算)
        # 因為資料量不大，因此將 param.requires_grad 設為 False，只訓練新的分類層，減少訓練時間和 overfitting 風險。
        for param in vgg16.features.parameters():
            param.requires_grad = features_grad
        self.backend = vgg16

        self.classifier = nn.Sequential(
            nn.Linear(514, 256),
            nn.ReLU(),
            nn.Dropout(0.5),  # 添加 Dropout 层
            nn.Linear(256, num_classes),
            nn.Softmax(dim=1)
        )

        self.softmax = nn.Softmax(dim=1) # 每一個row的總和都是1

    def forward(self, x, age, gender):
        output = self.backend(x)  # VGG 提取特征
        output = torch.cat([output, age.view(-1, 1), gender.view(-1, 1)], dim=1)  # 拼接年龄和性别
        output = self.classifier(output)  # 通过分类器
        return output
```

```python
class ResNetplus(nn.Module):
    def __init__(self, num_classes, input_size=(3, 50, 50), features_grad=False):
        super().__init__()
        # Load the pre-trained ResNet50 model
        resnet50 = models.resnet50(weights='IMAGENET1K_V1', progress=True)
        # Replace the final fully connected layer with nn.Identity() to act as a placeholder
        # This lets us add our own classification layers later.
        resnet50.fc = nn.Identity()
        # Freeze or unfreeze ResNet50 feature layers
        for param in resnet50.parameters():
            param.requires_grad = features_grad
        self.backend = resnet50
        # Add the custom classifier that combines image features with age and gender
        # resnet50 outputs a 2048-dimensional feature vector, and we add age & gender (2 features)
        self.classifier = nn.Sequential(
            nn.Linear(2048 + 2, num_classes),  # 2048: ResNet50 feature vector size, 2: age & gender
            nn.Softmax(dim=1)
        )
        self.softmax = nn.Softmax(dim=1) # 每一個row的總和都是1

    def forward(self, x, age, gender):
        features = self.backend(x)  # Pass the input through the ResNet50 backbone
        combined_features = torch.cat([features, age.view(-1, 1), gender.view(-1, 1)], dim=1)  # Concatenate along the column
        # Pass the combined features through the custom classifier
        outputs = self.classifier(combined_features)
        return outputs
```

```python
class VisionTransformerCustom(nn.Module):
    def __init__(self, num_classes):
        super(VisionTransformerCustom, self).__init__()

        self.model = VisionTransformer(
            img_size=56,                # 影像大小
            patch_size=8,               # 一個 patch 大小
            in_chans=1,                 # input channels 大小
            embed_dim=96,               #  patch 經過展平再 linear 的維度
            embed_layer=PatchEmbed,     # 使用 PatchEmbed 完成上述的動作
            norm_layer=nn.LayerNorm,    # 使用 LayerNorm 來 norm
            depth=12,                   # 使用 12個 Blocks(Encoders)
            num_classes=num_classes     # 輸出的維度
        )

        self.model.head = nn.Identity() # 將最後的分類弄成 Identity()

        self.classifier = nn.Sequential(
            nn.Linear(96 + 2, num_classes),  # 把 age, gender 放進其中
            nn.Softmax(dim=1)
        )


    def forward(self, x, age, gender):
        x = self.model(x)
        x = torch.cat([x, age.view(-1, 1), gender.view(-1, 1)], dim = 1)
        x = self.classifier(x)
        return x
```

```python
class SwinTransformerCustom(nn.Module):
    def __init__(self, num_classes):
        super(SwinTransformerCustom, self).__init__()

        self.model = SwinTransformer(
            img_size=56,
            patch_size=4,
            in_chans=1,
            embed_dim=96,
            depths=[2, 2, 6, 2],
            num_heads=[3, 6, 12, 24],
            window_size=7,
            mlp_ratio=4.0,
            norm_layer=nn.LayerNorm,
            num_classes=num_classes
        )

        self.model.head.fc = nn.Identity()

        self.classifier = nn.Sequential(
            nn.Linear(768 + 2, num_classes),  # 把 age, gender 放進其中
            nn.Softmax(dim=1)
        )


    def forward(self, x, age, gender):
        x = self.model(x)
        x = torch.cat([x, age.view(-1, 1), gender.view(-1, 1)], dim=1)
        x = self.classifier(x)
        return x
```

h. You should submit compile HTML file and ipynb notebook with name prefix to e3 platform. - {studentID}.ipynb {studentID}.html {studentID}.pdf

i. Note: make sure your ipynb file print out the number of parameters of the model.