ณสิต ผลัญชัย 65010273

วิวัตร เตชะโกศล 65011001

Lab6

```python
### START CODE HERE ###
class CustomImageDataset(Dataset):
    def __init__(self, image_paths, gauss_noise=False, gauss_blur=None, resize=128, p=0.5):
        self.p = p
        self.resize = resize
        self.gauss_noise = gauss_noise
        self.gauss_blur = gauss_blur
        self.image_paths = image_paths

    def __len__(self):
        return len(self.image_paths)

    def apply_gaussian_noise(self, image):
        if random.random() < self.p:
            mean = random.uniform(-50, 50)
            std_dev = 25
            noise = np.random.normal(mean, std_dev, image.shape).astype(np.float32)
            noisy_image = cv2.add(image.astype(np.float32), noise)
            noisy_image = np.clip(noisy_image, 0, 255).astype(np.float32)
            return noisy_image
        return image

    def apply_gaussian_blur(self, image):
        if random.random() < self.p:
            ksize = random.choice(range(3, 12, 2))
            image = cv2.GaussianBlur(image, (ksize, ksize), 0)
        return image
```

```python
def __getitem__(self, idx):
    image_path = self.image_paths[idx]
    image = cv2.imread(image_path)
    image = cv2.resize(cv2.cvtColor(image, cv2.COLOR_BGR2RGB), (self.resize, self.resize))

    gt_image = image.copy()

    if self.gauss_noise:
        image = self.apply_gaussian_noise(image)

    if self.gauss_blur:
        image = self.apply_gaussian_blur(image)

    image = image.astype(np.float32) / 255.0
    gt_image = gt_image.astype(np.float32) / 255.0

    return torch.tensor(image).permute(2, 0, 1), torch.tensor(gt_image).permute(2, 0, 1)
```

สร้าง class CustomImageDataset() ที่จะทำหน้าที่ดึงภาพมา transform อย่าง gaussian noise และ gaussian blur รวมถึง resize ภาพ ก่อนจะ normalize และเปลี่ยนเป็น tensor โดยจะ return image คือภาพที่ถูก transform และ gt_image คือภาพที่ไม่ถูก transform

```
### START CODE HERE ###
def imshow_grid(images):
    fig, axes = plt.subplots(4, 4, figsize=(15, 15))
    axes = axes.flatten()

    for img, ax in zip(images, axes):
        img_np = img.permute(1, 2, 0).numpy()
        ax.imshow(np.clip(img_np, 0, 1))
        ax.axis('off')

    plt.tight_layout()
    plt.show()
### END CODE HERE ###
```

Imshow_grid() ใช้ในการแสดงผลรูปภาพโดยตั้งให้แสดง 4 columns 4 rows

```
### START CODE HERE ###
data_dir = r'C:\Users\Nickv\Documents\ImageProcessing\Week5\img_align_celeba'
image_paths = [os.path.join(data_dir, fname) for fname in os.listdir(data_dir) if fname.endswith('.jpg')]
dataset = CustomImageDataset(image_paths=image_paths, gauss_noise=True, gauss_blur=True, resize=128, p=0.5)
dataloader = DataLoader(dataset, batch_size=16, shuffle=True)
### END CODE HERE ###
```

ทำการเรียกใช้ CustomImageDataset() เพื่อ transform ภาพ ก่อนจะโหลดภาพและแบ่ง batch เป็น batch ละ 16 ภาพ

```
### START CODE HERE ###
batch, gt_img = next(iter(dataloader))

imshow_grid(batch)
imshow_grid(gt_img)

### END CODE HERE ###
```

ทำการแสดงผลภาพที่ถูก apply gaussian noise & gaussian blur และ gt_image ที่ไม่ถูก apply

```python
### START CODE HERE ###
class DownSamplingBlock(nn.Module):
    def __init__(self, in_channels, out_channels, kernel_size=3, stride=1, padding=1):
        super(DownSamplingBlock, self).__init__()
        self.conv = nn.Conv2d(in_channels, out_channels, kernel_size=kernel_size, stride=stride, padding=padding)
        self.relu = nn.ReLU()
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)

    def forward(self, x):
        x = self.conv(x)
        x = self.relu(x)
        x = self.pool(x)
        return x

class UpSamplingBlock(nn.Module):
    def __init__(self, in_channels, out_channels, kernel_size=3, stride=1, padding=1):
        super(UpSamplingBlock, self).__init__()
        self.conv = nn.Conv2d(in_channels, out_channels, kernel_size=kernel_size, stride=stride, padding=padding)
        self.relu = nn.ReLU()
        self.upsample = nn.Upsample(scale_factor=2, mode='bilinear', align_corners=True)

    def forward(self, x):
        x = self.conv(x)
        x = self.relu(x)
        x = self.upsample(x)
        return x
```

```python
class Autoencoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv_in = nn.Conv2d(3, 64, kernel_size=3, stride=1, padding=1)
        self.down1 = DownSamplingBlock(64, 128, kernel_size=3, stride=1, padding=1)
        ### DESIGN YOUR OWN MODEL ###
        self.down2 = DownSamplingBlock(128, 256, kernel_size=3, stride=1, padding=1)
        self.down3 = DownSamplingBlock(256, 512, kernel_size=3, stride=1, padding=1)

        self.up1 = UpSamplingBlock(512, 256, kernel_size=3, stride=1, padding=1)
        self.up2 = UpSamplingBlock(256, 128, kernel_size=3, stride=1, padding=1)
        self.up3 = UpSamplingBlock(128, 64, kernel_size=3, stride=1, padding=1)
        self.conv_out = nn.Conv2d(64, 3, kernel_size=3, stride=1, padding=1)

    def forward(self, x):
        # Encoding
        x = self.conv_in(x)
        x = self.down1(x)
        x = self.down2(x)
        x = self.down3(x)

        # Decoding
        x = self.up1(x)
        x = self.up2(x)
        x = self.up3(x)
        x = self.conv_out(x)
        return x
### END CODE HERE ###
```

ใช้ model Autoencoder จาก Lab5_2 Architect [64, 128, 256, 512]

```
def train(model, opt, loss_fn, train_loader, test_loader, epochs=10,
    print("🚂 Training on", device)
    model = model.to(device)

    for epoch in range(epochs):
        model.train()
        avg_train_loss = 0.0
        step = 0
        train_bar = tqdm(train_loader, desc=f'🚀 Training Epoch [{ep

        for images, gt in train_bar:
            images = images.to(device)
            gt = gt.to(device)
            opt.zero_grad()
            output = model(images)
            loss = loss_fn(output, gt)
            loss.backward()
            opt.step()
```

```
            avg_train_loss += loss.item()
            step += 1

            train_bar.set_postfix(loss=loss.item())

        avg_train_loss /= step

        model.eval()
        avg_test_loss = 0.0
        test_bar = tqdm(test_loader, desc='📄 Testing', unit='batch')

        for images, gt in test_bar:
            images = images.to(device)
            gt = gt.to(device)
            output = model(images)
            loss = loss_fn(output, gt)
            avg_test_loss += loss.item()

        avg_test_loss /= len(test_loader)

    print(f'Epoch [{epoch+1}/{epochs}], Train Loss: {avg_train_loss:.4f}, Test Loss:

    if checkpoint_path:
        torch.save(model.state_dict(), f"{checkpoint_path}_epoch_{epoch+1}.pth")

    print("🏁 Training completed.")
```

ใช้ function train() จาก Lab5_2

```
### START CODE HERE ###

data_dir = r'C:\Users\Nickv\Documents\ImageProcessing\Week5\img_align_celeba'

files = os.listdir(data_dir)
files = [os.path.join(data_dir, file) for file in files]


train_files, test_files = train_test_split(files, test_size=0.3, random_state=42)


train_dataset = CustomImageDataset(image_paths=train_files, gauss_noise=True, gaus
test_dataset = CustomImageDataset(image_paths=test_files, gauss_noise=False, gauss
trainloader = DataLoader(train_dataset, batch_size=16, shuffle=True)
testloader = DataLoader(test_dataset, batch_size=16, shuffle=False)
### END CODE HERE ###
```

ทำการโหลดภาพโดยแบ่งสัดส่วน train : test เป็น 70 : 30 transform train dataset ด้วยการ apply gaussian noise & gaussian blur บางภาพ

```
### START CODE HERE ###

model = Autoencoder()
opt = optim.Adam(model.parameters(), lr=0.001)
loss_fn = nn.MSELoss()
train(model, opt, loss_fn, trainloader, testloader, epochs=2,

### END CODE HERE ###
```

ทดลอง train 2 epoch

```
🤖 Training on cuda
🚀 Training Epoch [1/2]: 100%|██████████| 1313/1313 [01:18<00:00, 16.77batch/s, loss=0.012]
📄 Testing: 100%|██████████| 563/563 [00:14<00:00, 39.52batch/s]
🚀 Training Epoch [2/2]: 100%|██████████| 1313/1313 [01:16<00:00, 17.06batch/s, loss=0.013]
📄 Testing: 100%|██████████| 563/563 [00:13<00:00, 40.22batch/s]
Epoch [2/2], Train Loss: 0.0109, Test Loss: 0.0089
🏁 Training completed.
```

Hyperparameter Random Search with Raytune

คือการหา parameter ที่ดีที่สุดในการ train ด้วยการ grid search

```python
### START CODE HERE ###
def train_raytune(config):
    architecture = config["architecture"]
    lr = config["lr"]
    batch_size = config["batch_size"]
    num_epochs = config["num_epochs"]
    optimizer = config["optimizer"]

    transform = transforms.Compose([transforms.ToTensor()])

    train_files, test_files = train_test_split(image_paths, test_size=0.2, random_state=42)
    train_dataset = CustomImageDataset(image_paths=train_files,
                                       gauss_noise=True,
                                       gauss_blur=True,
                                       resize=128,
                                       p=0.5,
                                       center_crop=True,
                                       transform=transform)
    test_dataset = CustomImageDataset(image_paths=test_files,
                                       gauss_noise=True,
                                       gauss_blur=True,
                                       resize=128,
                                       p=0.5,
                                       center_crop=True,
                                       transform=transform)
    trainloader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True, num_workers=0)
    testloader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False, num_workers=0)

    train_batch, train_gt = next(iter(trainloader))
    test_batch, test_gt = next(iter(testloader))


    device = "cuda" if torch.cuda.is_available() else "cpu"
    model = Autoencoder(architecture)
```

```python
device = "cuda" if torch.cuda.is_available() else "cpu"
model = Autoencoder(architecture)
model = model.to(device)
loss_fn = nn.MSELoss()

if optimizer == 'Adam':
    opt = torch.optim.Adam(model.parameters(), lr=lr)
elif optimizer == 'SGD':
    opt = torch.optim.SGD(model.parameters(), lr=lr)

for epoch in range(num_epochs):
    model.train()
    avg_train_loss = 0
    avg_test_loss = 0
    for batch in trainloader:
        images, train_gt_img = batch

        images = images.to(device)
        train_gt_img = train_gt_img.to(device)

        output = model(images)

        loss = loss_fn(output, train_gt_img)

        opt.zero_grad()
        loss.backward()
        opt.step()

        avg_train_loss += loss.item()

    avg_train_loss /= len(trainloader)

    total_psnr = 0
```

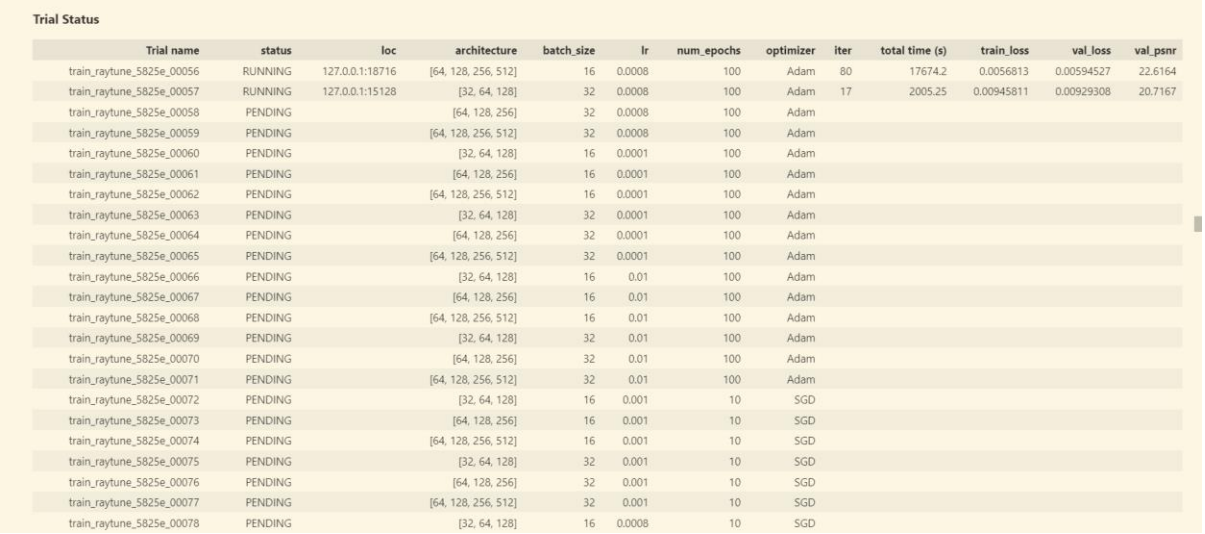
```python
total_psnr = 0
total_ssim = 0
model.eval()
with torch.no_grad():
    for batch in testloader:
        images, test_gt_img = batch
        images = images.to(device)
        test_gt_img = test_gt_img.to(device)

        output = model(images)

        loss = loss_fn(output, test_gt_img)
        avg_test_loss += loss.item()

        output_np = output.cpu().numpy().transpose(0, 2, 3, 1)
        images_np = test_gt_img.cpu().numpy().transpose(0, 2, 3, 1)

        for i in range(images_np.shape[0]):
            img = images_np[i]
            rec_img = output_np[i]
            total_psnr += psnr(img, rec_img, data_range=1.0)

            min_dim = min(img.shape[0], img.shape[1])
            win_size = min(7, min_dim)
            if win_size % 2 == 0:
                win_size -= 1
            total_ssim += ssim(img, rec_img, win_size=win_size, data_range=1.0, channel_axis=2)

avg_psnr = total_psnr / (len(testloader.dataset))
avg_ssim = total_ssim / (len(testloader.dataset))
avg_test_loss /= len(testloader)

session.report({
    "train_loss": avg_train_loss,
    "val_loss": avg_test_loss,
    "val_psnr": avg_psnr,
    "val_ssim": avg_ssim,
})
```

```python
### START CODE HERE ###
ray.init(num_gpus=1, ignore_reinit_error=True)

# Define the trial directory naming function
def short_dirname(trial):
    return "trial_" + str(trial.trial_id)

config = {
    'architecture': tune.grid_search([[32, 64, 128], [64, 128, 256], [64, 128, 256, 512]]),
    "lr": tune.grid_search([1e-3, 8e-4, 1e-4, 1e-2]),
    "batch_size": tune.grid_search([16, 32]),
    "num_epochs": tune.grid_search([10, 50, 100]),
    'optimizer': tune.grid_search(['Adam', 'SGD']),

}

result = tune.run(
    tune.with_resources(train_raytune, resources={"gpu": 0.5}),
    config=config,
    metric="val_psnr",
    mode="max",
    trial_dirname_creator=short_dirname
)

print("Best config: ", result.get_best_config(metric="val_psnr", mode="max"))
### END CODE HERE ###
```

✓ 3455m 2.9s

โดยสามารถปรับ layer, lr, batch_size, num_epochs, optimizer โดยจะมีtrial ทั้งหมด 144 trials เพราะว่าใช้ทั้งหมด architecture 3 แบบ : [32, 64, 128], [64, 128, 256], [64, 128, 256, 512] , lr ทั้งหมด 4 แบบ : [1e-3, 8e-4, 1e-4, 1e-2], batch_size ทั้งหมด 2 size: [16, 32], nums_epochs ทั้งหมด 3 แบบ : [10, 50, 100], optimizer 2 แบบ : [‘Adam’, ‘SDG’] แล้วจะได้best config ออกมา

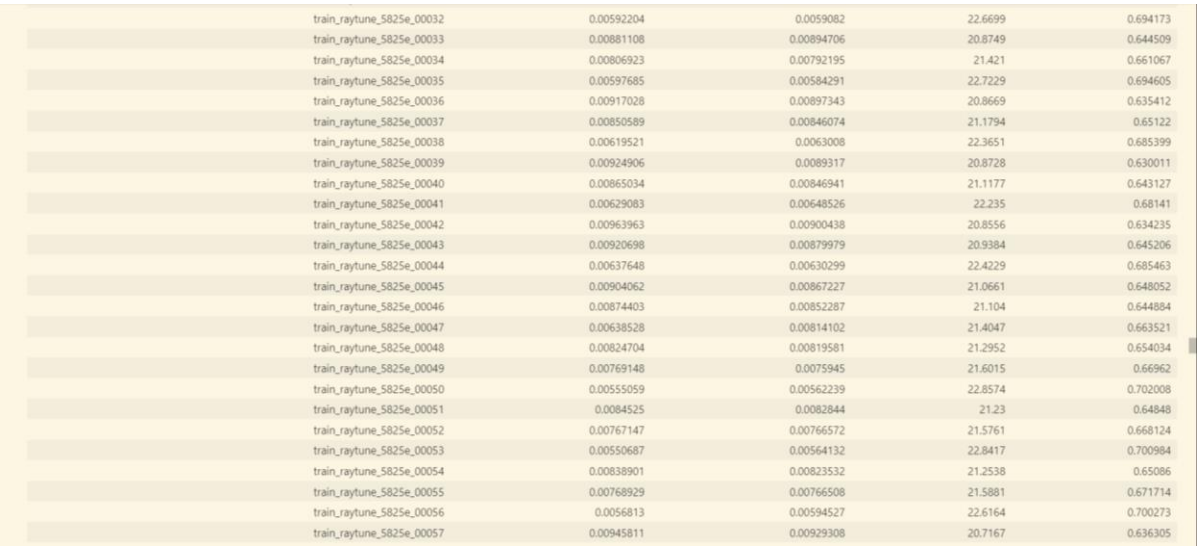
Trial Status



Trial Status

Trial name	status	loc	architecture	batch_size	lr	num_epochs	optimizer	iter	total time (s)	train_loss	val_loss	val_psnr
train_raytune_5825e_00056	RUNNING	127.0.0.1:18716	[64, 128, 256, 512]	16	0.0008	100	Adam	80	17674.2	0.0056813	0.00594527	22.6164
train_raytune_5825e_00057	RUNNING	127.0.0.1:15128	[32, 64, 128]	32	0.0008	100	Adam	17	2005.25	0.00945811	0.00929308	20.7167
train_raytune_5825e_00058	PENDING		[64, 128, 256]	32	0.0008	100	Adam					
train_raytune_5825e_00059	PENDING		[64, 128, 256, 512]	32	0.0008	100	Adam					
train_raytune_5825e_00060	PENDING		[32, 64, 128]	16	0.0001	100	Adam					
train_raytune_5825e_00061	PENDING		[64, 128, 256]	16	0.0001	100	Adam					
train_raytune_5825e_00062	PENDING		[64, 128, 256, 512]	16	0.0001	100	Adam					
train_raytune_5825e_00063	PENDING		[32, 64, 128]	32	0.0001	100	Adam					
train_raytune_5825e_00064	PENDING		[64, 128, 256]	32	0.0001	100	Adam					
train_raytune_5825e_00065	PENDING		[64, 128, 256, 512]	32	0.0001	100	Adam					
train_raytune_5825e_00066	PENDING		[32, 64, 128]	16	0.01	100	Adam					
train_raytune_5825e_00067	PENDING		[64, 128, 256]	16	0.01	100	Adam					
train_raytune_5825e_00068	PENDING		[64, 128, 256, 512]	16	0.01	100	Adam					
train_raytune_5825e_00069	PENDING		[32, 64, 128]	32	0.01	100	Adam					
train_raytune_5825e_00070	PENDING		[64, 128, 256]	32	0.01	100	Adam					
train_raytune_5825e_00071	PENDING		[64, 128, 256, 512]	32	0.01	100	Adam					
train_raytune_5825e_00072	PENDING		[32, 64, 128]	16	0.001	10	SGD					
train_raytune_5825e_00073	PENDING		[64, 128, 256]	16	0.001	10	SGD					
train_raytune_5825e_00074	PENDING		[64, 128, 256, 512]	16	0.001	10	SGD					
train_raytune_5825e_00075	PENDING		[32, 64, 128]	32	0.001	10	SGD					
train_raytune_5825e_00076	PENDING		[64, 128, 256]	32	0.001	10	SGD					
train_raytune_5825e_00077	PENDING		[64, 128, 256, 512]	32	0.001	10	SGD					
train_raytune_5825e_00078	PENDING		[32, 64, 128]	16	0.0008	10	SGD					

Trial Progress



train_raytune_5825e_00032	0.00592204	0.0059082	22.6699	0.694173
train_raytune_5825e_00033	0.00881108	0.00894706	20.8749	0.644509
train_raytune_5825e_00034	0.00806923	0.00792195	21.421	0.661067
train_raytune_5825e_00035	0.00597685	0.00584291	22.7229	0.694605
train_raytune_5825e_00036	0.00917028	0.00897343	20.8669	0.635412
train_raytune_5825e_00037	0.00850589	0.00846074	21.1794	0.65122
train_raytune_5825e_00038	0.00619521	0.0063008	22.3651	0.685399
train_raytune_5825e_00039	0.00924906	0.0089317	20.8728	0.630011
train_raytune_5825e_00040	0.00865034	0.00846941	21.1177	0.643127
train_raytune_5825e_00041	0.00629083	0.00648526	22.235	0.68141
train_raytune_5825e_00042	0.00963963	0.00900438	20.8556	0.634235
train_raytune_5825e_00043	0.00920698	0.00879979	20.9384	0.645206
train_raytune_5825e_00044	0.00637648	0.00630299	22.4229	0.685463
train_raytune_5825e_00045	0.00904062	0.00867227	21.0661	0.648052
train_raytune_5825e_00046	0.00874403	0.00852287	21.104	0.644884
train_raytune_5825e_00047	0.00638528	0.00814102	21.4047	0.663521
train_raytune_5825e_00048	0.00824704	0.00819581	21.2952	0.654034
train_raytune_5825e_00049	0.00769148	0.0075945	21.6015	0.66962
train_raytune_5825e_00050	0.00555059	0.00562239	22.8574	0.702008
train_raytune_5825e_00051	0.0084525	0.0082844	21.23	0.64848
train_raytune_5825e_00052	0.00767147	0.00766572	21.5761	0.668124
train_raytune_5825e_00053	0.00550687	0.00564132	22.8417	0.700984
train_raytune_5825e_00054	0.00838901	0.00823532	21.2538	0.65086
train_raytune_5825e_00055	0.00768929	0.00766508	21.5881	0.671714
train_raytune_5825e_00056	0.0056813	0.00594527	22.6164	0.700273
train_raytune_5825e_00057	0.00945811	0.00929308	20.7167	0.636305

Best Config

เนื่องจากเวลาที่จำกัดจึงไม่สามารถ trial raytune จนครบ 144 trials ได้ ดังนั้นผมจึงจะใช้ best config ที่ดีที่สุดที่ผม trial ไปแล้ว ดังนั้น best config ที่จะใช้คือ

architecture : [64, 128, 256, 512], lr : 0.001, batch_size : 16, num_epochs : 100, optimizer : ‘Adam’

```python
def train(model, opt, loss_fn, train_loader, test_loader, epochs=10, checkpoint_path=None, device='cuda'):
    print(":robot: Training on", device)
    model = model.to(device)

    for epoch in range(epochs):
        model.train()
        avg_train_loss = 0.0
        step = 0
        train_bar = tqdm(train_loader, desc=f':rocket: Training Epoch [{epoch+1}/{epochs}]', unit='batch')

        for images, gt in train_bar:
            images = images.float() / 255.0
            images = images.permute(0, 3, 1, 2).to(device)
            gt = gt.float() / 255.0
            gt = gt.permute(0, 3, 1, 2).to(device)
            opt.zero_grad()
            output = model(images)
            loss = loss_fn(output, gt)
            loss.backward()
            opt.step()

            avg_train_loss += loss.item()
            step += 1

            train_bar.set_postfix(loss=loss.item())

        avg_train_loss /= step

        model.eval()
        avg_test_loss = 0.0
        test_bar = tqdm(test_loader, desc=':page_facing_up: Testing', unit='batch')

        with torch.no_grad():
            for images, gt in test_bar:
                images = images.float() / 255.0
                images = images.permute(0, 3, 1, 2).to(device)
                gt = gt.float() / 255.0
                gt = gt.permute(0, 3, 1, 2).to(device)
```

```python
### START CODE HERE ###
data_dir = r"C:\Users\ADMIN\Desktop\coding\imageprocessing\lab6\img_align_celeba"
files = os.listdir(data_dir)
files = [os.path.join(data_dir, file) for file in files]


train_files, test_files = train_test_split(files, test_size=0.3, random_state=42)


train_dataset = CustomImageDataset(image_paths=train_files, gauss_noise=True, gauss_blur=True)
test_dataset = CustomImageDataset(image_paths=test_files, gauss_noise=False, gauss_blur=False)
trainloader = DataLoader(train_dataset, batch_size=16, shuffle=True)
testloader = DataLoader(test_dataset, batch_size=16, shuffle=False)
model = Autoencoder()
opt = optim.Adam(model.parameters(), lr=0.001)
loss_fn = nn.MSELoss()
train(model, opt, loss_fn, trainloader, testloader, epochs=100, checkpoint_path='autoencoder.pth')
### END CODE HERE ###
```

✓ 279m 55.2s

```
:page_facing_up: Testing: 100%|██████████| 563/563 [00:12<00:00, 45.19batch/s]
:rocket: Training Epoch [12/100]: 100%|██████████| 1313/1313 [02:41<00:00,  8.12batch/s, loss=0.00701]
:page_facing_up: Testing: 100%|██████████| 563/563 [00:11<00:00, 47.48batch/s]
:rocket: Training Epoch [13/100]: 100%|██████████| 1313/1313 [02:40<00:00,  8.19batch/s, loss=0.00779]
:page_facing_up: Testing: 100%|██████████| 563/563 [00:11<00:00, 47.53batch/s]
:rocket: Training Epoch [14/100]: 100%|██████████| 1313/1313 [02:40<00:00,  8.19batch/s, loss=0.00783]
:page_facing_up: Testing: 100%|██████████| 563/563 [00:11<00:00, 47.50batch/s]
:rocket: Training Epoch [15/100]: 100%|██████████| 1313/1313 [02:38<00:00,  8.31batch/s, loss=0.00717]
:page_facing_up: Testing: 100%|██████████| 563/563 [00:11<00:00, 48.01batch/s]
:rocket: Training Epoch [16/100]: 100%|██████████| 1313/1313 [02:52<00:00,  7.60batch/s, loss=0.00663]
:page_facing_up: Testing: 100%|██████████| 563/563 [00:12<00:00, 44.09batch/s]
:rocket: Training Epoch [17/100]: 100%|██████████| 1313/1313 [02:40<00:00,  8.16batch/s, loss=0.00652]
:page_facing_up: Testing: 100%|██████████| 563/563 [00:12<00:00, 45.79batch/s]
:rocket: Training Epoch [18/100]: 100%|██████████| 1313/1313 [02:37<00:00,  8.32batch/s, loss=0.0106]
:page_facing_up: Testing: 100%|██████████| 563/563 [00:11<00:00, 47.95batch/s]
:rocket: Training Epoch [19/100]: 100%|██████████| 1313/1313 [02:41<00:00,  8.13batch/s, loss=0.00676]
```

หลังจากที่ได้ best config มาก็จะนำbest config มา train Autoencoder() model

```python
### START CODE HERE ###
import math

class FeatureMapVisualizer:
    def __init__(self, model, layers, save_dir):
        self.model = model
        self.layers = layers if isinstance(layers, list) else [layers]
        self.activations = {}
        self.save_dir = save_dir

        os.makedirs(self.save_dir, exist_ok=True)

        self._register_hooks()

    def _register_hooks(self):
        for name, layer in self.model.named_modules():
            if name in self.layers:
                layer.register_forward_hook(self._hook_fn(name))

    def _hook_fn(self, layer_name):
        def hook(module, input, output):
            print(f'Hooking layer: {layer_name}')
            self.activations[layer_name] = output.detach()
        return hook
    def visualize(self, input_tensors):
        if not isinstance(input_tensors, list):
            input_tensors = [input_tensors]

        for idx, img_tensor in enumerate(input_tensors):
            with torch.no_grad():
                self.model(img_tensor)

            for layer_name, activation in self.activations.items():
                print(f'Visualizing and saving layer: {layer_name}')
                img_feature_map_dir = os.path.join(self.save_dir, f'image_{idx}')
                os.makedirs(img_feature_map_dir, exist_ok=True)
```

```python
    def _save_feature_maps(self, activation, layer_name, img_feature_map_dir):
        num_channels = activation.shape[1]
        cols = 8
        rows = math.ceil(num_channels / cols)

        fig, axes = plt.subplots(rows, cols, figsize=(cols * 2, rows * 2))

        if rows == 1:
            axes = [axes]
        axes = np.array(axes).reshape(rows, cols)

        for i in range(num_channels):
            ax = axes[i // cols, i % cols]
            feature_map = activation[0, i].cpu().numpy()
            ax.imshow(feature_map, cmap='viridis')
            ax.axis('off')

        for j in range(i + 1, rows * cols):
            fig.delaxes(axes[j // cols, j % cols])

        plt.suptitle(f"Feature Maps from Layer: {layer_name}")
        plt.tight_layout()

        plt.savefig(os.path.join(img_feature_map_dir, f"{layer_name}_feature_maps.png"))
        plt.close()


### END CODE HERE ###
```

ใช้ class FeatureMapVisualizer() ในการ visualize feature maps โดยจะสามารถกำหนด layers ที่ต้องการ visualize และ save ภาพใน folder ที่ต้องการได้

```
model = Autoencoder()
model.load_state_dict(torch.load(r'C:\Users\ADMIN\Desktop\coding\imageprocessing\lab6\autoencoder.pth_epoch_100.pth'))
model.eval()
```

ทำการโหลด model Autoencoder() ที่ได้ save เอาไว้

```
### START CODE HERE ###
data_dir = r'C:\Users\ADMIN\Desktop\coding\imageprocessing\lab6\img_align_celeba'
image_paths = [os.path.join(data_dir, fname) for fname in os.listdir(data_dir) if fname.endswith('.jpg')]
dataset = CustomImageDataset(image_paths, gauss_noise=True, gauss_blur=True)
dataloader = DataLoader(dataset, batch_size=16, shuffle=True)
images, labels = next(iter(dataloader))


### END CODE HERE ###
```

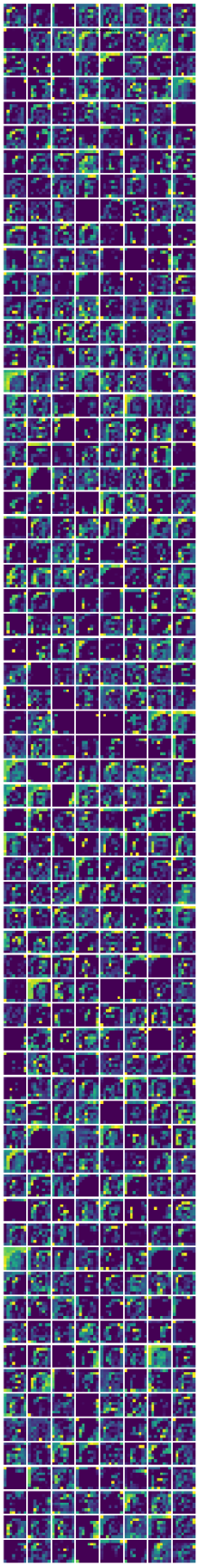
[38] ✓ 0.0s

```
layers_to_visualize = ['encoder', 'decoder', 'final_conv']
visualizer = FeatureMapVisualizer(model, layers_to_visualize, r'C:\Users\ADMIN\Desktop\coding\imageprocessing\lab6')
visualizer.visualize(images)
```

[37] ✓ 8.1s

```
Hooking layer: encoder
Hooking layer: decoder
Hooking layer: final_conv
Visualizing and saving layer: encoder
Visualizing and saving layer: decoder
Visualizing and saving layer: final_conv
```

นำภาพมา visualize feature maps จะได้ผลลัพธ์ดังภาพด้านล่าง



Feature Maps from Layer: final_conv

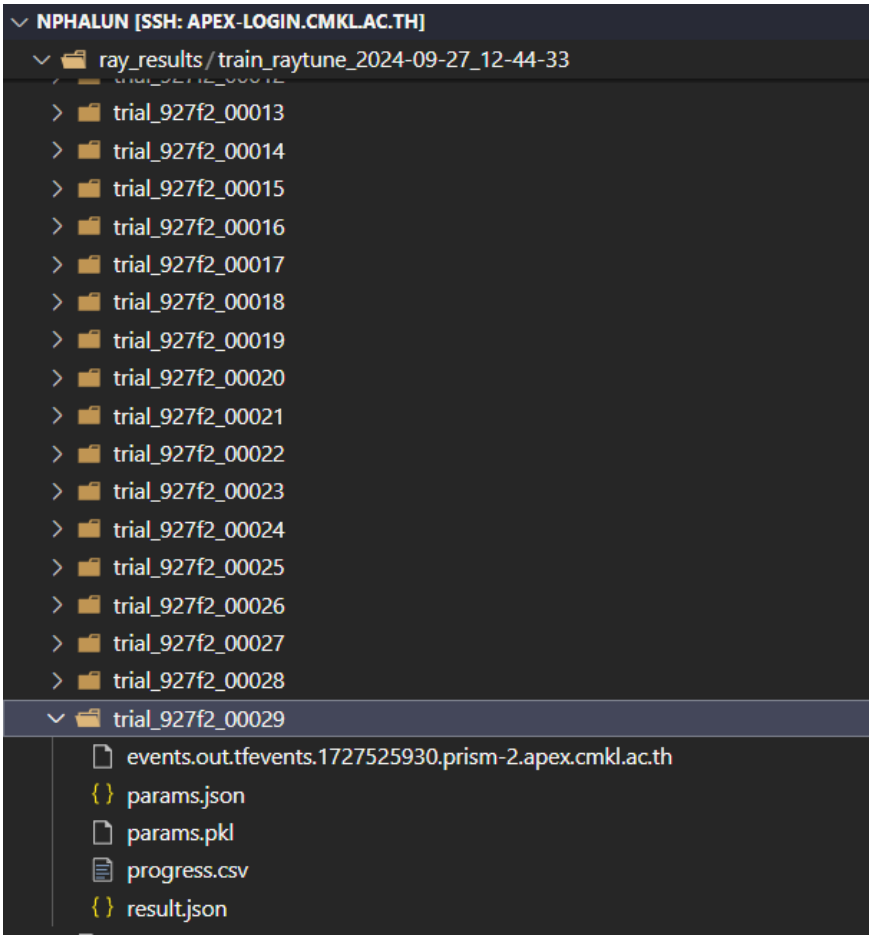
Hyperparameter Random Search with Raytune

คือการหา parameter ที่ดีที่สุดในการ train ด้วยการ random

```
### START CODE HERE ###
ray.shutdown()
ray.init(num_gpus=1, ignore_reinit_error=True)

def short_dirname(trial):
    return "trial_" + str(trial.trial_id)

config = {
    'architecture': tune.choice([[32, 64, 128], [64, 128, 256], [64, 128, 256, 512]]),
    "lr": tune.uniform(1e-4, 1e-2),
    "batch_size": tune.randint(16, 33),
    "num_epochs": tune.randint(10, 101),
    'optimizer': tune.choice(['Adam', 'SGD']),
}

result = tune.run(
    tune.with_resources(train_raytune, resources={"gpu": 0.5}),
    config=config,
    metric="val_psnr",
    mode="max",
    num_samples=30,
    trial_dirname_creator=short_dirname,
)

print("Best config: ", result.get_best_config(metric="val_psnr", mode="max"))
### END CODE HERE ###
```

โดยตั้งค่า config มีการสุ่ม architecture 3 แบบ สุ่ม learning rate ตั้งแต่ 1e-4 ถึง 1e-2 สุ่ม batch size ตั้งแต่ 16 ถึง 32 สุ่มจำนวน epoch ตั้งแต่ 10 ถึง 100 และสุ่ม optimizer ระหว่าง Adam และ SDG

ตามโจทย์ให้ random search with 80 samples แต่เนื่องด้วยเวลาที่จำกัดและระยะเวลาที่ใช้ค่อนข้างนานจึงขอลดเหลือ 30 samples โดยจะทำการ save ผลลัพธ์ไว้ใน folder ray_results



โดยหลังจาก random search 30 samples ได้ best config เป็น

Architecture: [64, 128, 256, 512], Learning rate: 0.0007686928707777884, Batch size: 19,

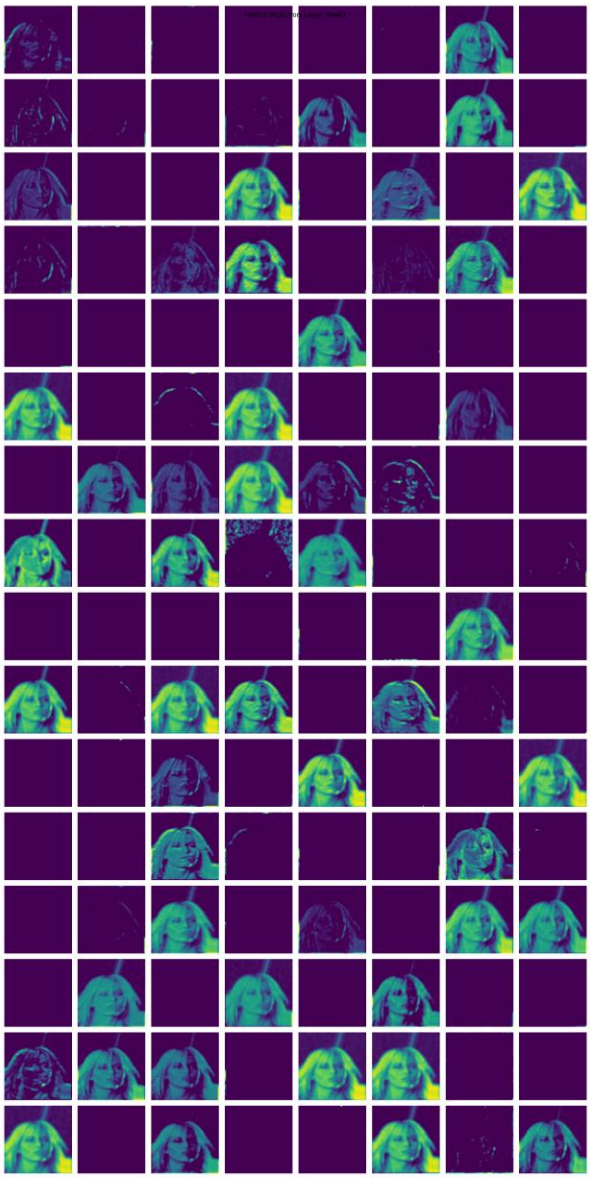
Epochs: 88, Optimizer: Adam

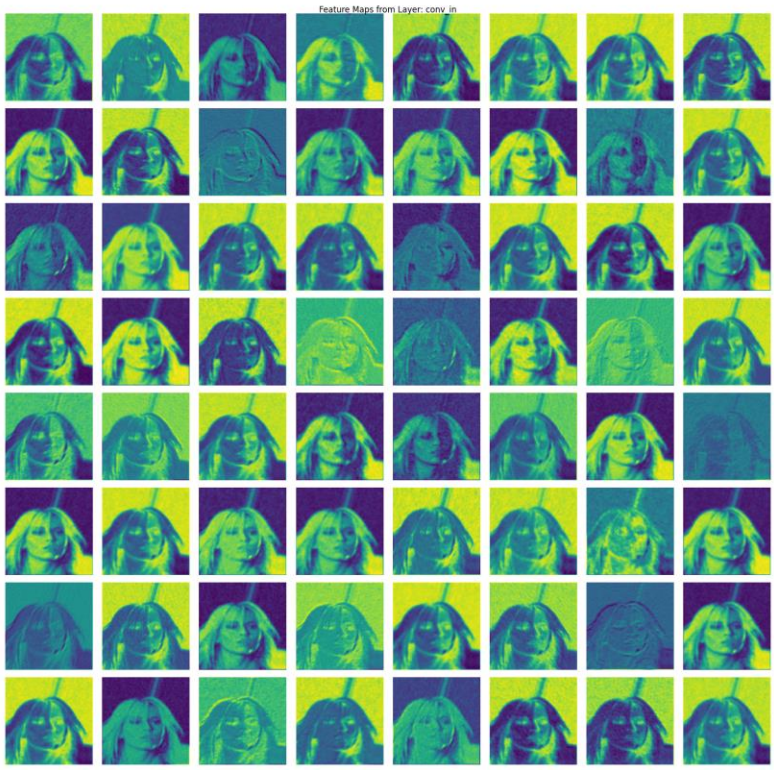
```
### START CODE HERE ###

data_dir = r'C:\Users\Nickv\Documents\ImageProcessing\Week5\img_align_celeba'

files = [os.path.join(data_dir, fname) for fname in os.listdir(data_dir) if fname.en

train_files, test_files = train_test_split(files, test_size=0.3, random_state=2024)


train_dataset = CustomImageDataset(image_paths=train_files, gauss_noise=True, gauss_
test_dataset = CustomImageDataset(image_paths=test_files, gauss_noise=False, gauss_b

trainloader = DataLoader(train_dataset, batch_size=19, shuffle=True, num_workers=0)
testloader = DataLoader(test_dataset, batch_size=19, shuffle=False, num_workers=0)

model = Autoencoder()
opt = optim.Adam(model.parameters(), lr=0.0007686928707777884)
loss_fn = nn.MSELoss()
train(model, opt, loss_fn, trainloader, testloader, epochs=88, checkpoint_path='auto

### END CODE HERE ###
```

นำ Architecture ที่ได้มาปรับ model Autoencoder() แล้วทำการ train โดยปรับค่าต่างๆตาม best config

```
model = Autoencoder()
model.load_state_dict(torch.load(r'C:\Users\Nickv\Documents\ImageProcessing\Week6\aut
model.eval()
```

เมื่อ train เสร็จเรียบร้อยก็ทำการโหลด model เพื่อนำมาใช้แสดง feature maps โดยจะได้ผลลัพธ์ดังนี้




Feature Maps from Layer: conv_in