지능화 캡스톤 프로젝트

프로젝트 #1 결과 발표

2024. 04. 29

충북대학교 산업인공지능학과

[3조] 이선경, 김정식(16)

프로젝트 수행체계

수행방법

- ✓ 1차. 딥러닝 개발환경 구축(~03.18)
- ✓ 2차. 데이터 파악, 구축, CNN 모델 설계 등에 대한 역할 분담(~3.25)
- ✓ 3차. 모델학습, 혼동행렬, 시각화 등을 수행(~04.22)
- ✓ 4차. 코드수정, 발표자료 작성 등을 수행(~04.29)

업무분장

이름	수행내용	비고
이선경	• 데이터 SET 구현, CNN 설계 및 구현, 모델학습 등	
김정식	• 자료조사, PPT 작성	

연구 방법

데이터 증량에 따른 모델 성능 비교(원본/증대 DataSet 비교)

- ✓ 클래스 불균형을 해결하기 위해 데이터 증대기술 적용*
  * 1.**이미지를 텐서로 변환하고 증강 적용**, 2. **텐서를** PIL **이미지로 다시 변환**, 3. **증강된 이미지 저장**
- ✓ 모델의 일반화 능력 향상, 과적합 방지, 클래스 불균형 해소, 성능지표 개선 등

다양한 딥러닝 아키텍처와 CNN-WDI, CNN-CREATE 모델의 비교

- ✓ CNN은 이미지 분류 및 객체 인식과 같은 간단한 작업에 적합하며, 모델 구조가 간단함
- ✓ CNN-WDI, CNN-CREATE는 더 복잡한 이미지 분류나 인식 작업에 사용하는것이 더 편리함
- ✓ Inception 모듈을 사용하여 다양한 크기와 종류의 특징을 추출하는데 용이함

가중치를 사용하여 특정 특징에 더 많은 가중치를 부여할 수 있어서 더 정확한 결과를 얻을 수 있을것으로 예상

데이터셋_데이터 증강 함수

데이터 증강 함수

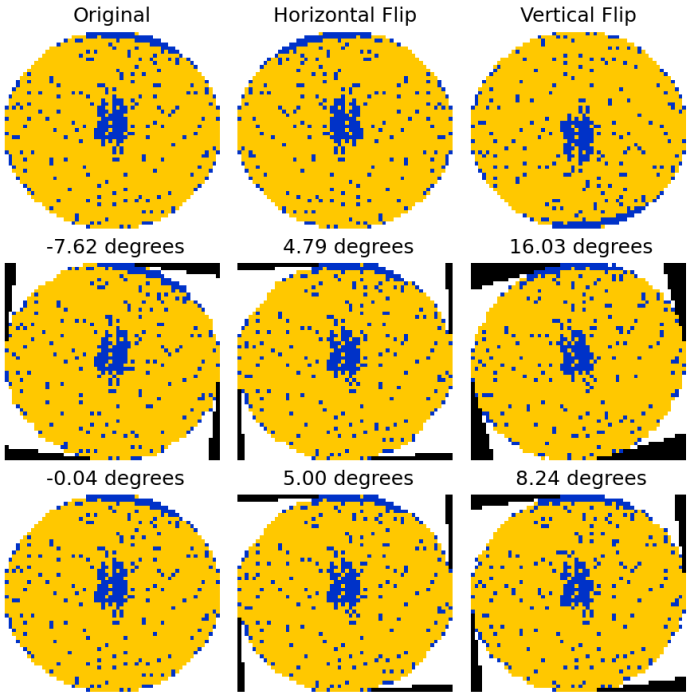
```python
# 데이터 증강 함수
def augment_images(image_folder, target_count=10000):
    images = [os.path.join(image_folder, img) for img in os.listdir(image_folder)]
    current_count = len(images)

    if current_count < target_count:
        print(f"Augmenting images in {image_folder}...")
        while current_count < target_count:
            for img_path in images:
                if current_count >= target_count:
                    break
                image = Image.open(img_path)
                # 이미지를 텐서로 변환하고 증강을 적용
                augmented_image = transform(image)
                # 텐서를 PIL 이미지로 다시 변환
                augmented_image = to_pil(augmented_image)
                # 증강된 이미지 저장
                augmented_image_path = f"{img_path.rsplit('.', 1)[0]}_aug_{current_count}.{img_path.rsplit('.', 1)[-1]}"
                augmented_image.save(augmented_image_path)
                current_count += 1

# 'train' 세트에 대해서만 데이터 증강을 수행
defect_types = ['Center', 'Donut', 'Edge-Loc', 'Edge-Ring', 'Loc', 'Near-full', 'none', 'Random', 'Scratch']
for defect_type in defect_types:
    print(f"Processing {defect_type} images in train set...")
    image_folder = os.path.join(data_dir, 'train', defect_type)
    augment_images(image_folder)

print("Image count adjustment completed for 'train' set only.")
```

데이터셋_데이터 증강

데이터 증강 기법

➢ Horizontal Flip, Vertical Flip, Rotation +/-20 degrees

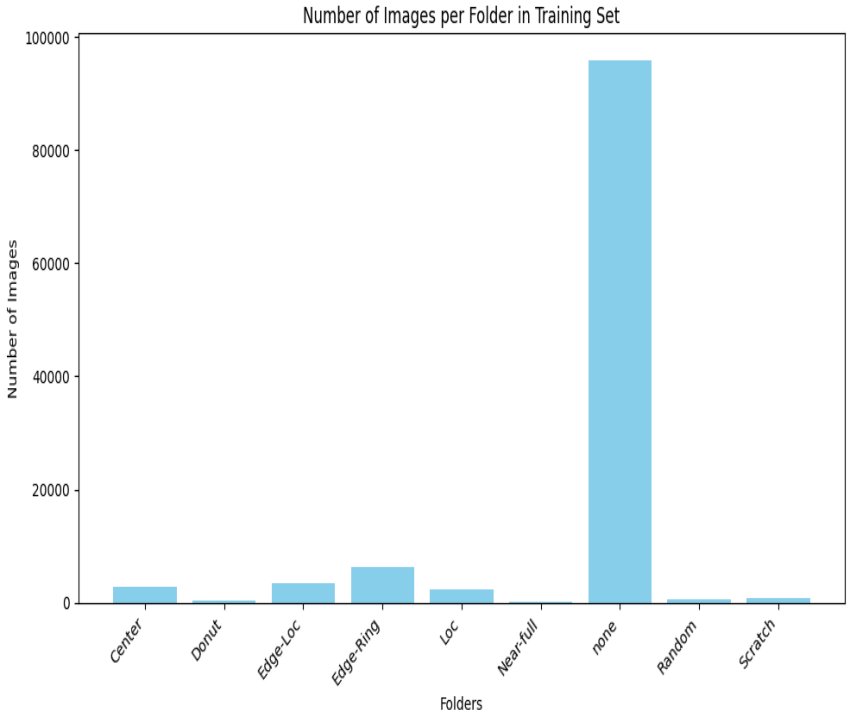
```python
# 시각화 함수에서 이 변환 리스트 사용
1 usage
def visualize_augmentations(image, transformations, descriptions, save_path=None, dpi=150):
    cols, rows = 3, 3
    fig, axes = plt.subplots(rows, cols, figsize=(cols * 2, rows * 2), dpi=dpi)
    axes = axes.flatten()

    # 원본 이미지 시각화
    axes[0].imshow(np.asarray(image), interpolation='nearest')
    axes[0].set_title('Original')
    axes[0].axis('off')

    # 뒤집기 변환 시각화
    for i, (transformation, description) in enumerate(zip(transformations, descriptions), start=1):
        transformed_image = transformation(image)
        axes[i].imshow(np.asarray(transformed_image), interpolation='nearest')
        axes[i].set_title(description)
        axes[i].axis('off')

    # 무작위 회전 변환 시각화 (6개)
    for i in range(3, 9):
        degree = random.uniform(-20, b: 20)
        rotation = transforms.RandomRotation(degrees=(degree, degree))
        transformed_image = rotation(image)
        axes[i].imshow(np.asarray(transformed_image), interpolation='nearest')
        axes[i].set_title(f' {degree:.2f} degrees')
        axes[i].axis('off')

    plt.tight_layout()
    if save_path:
        plt.savefig( *args: save_path, dpi=dpi)
    plt.show()
```



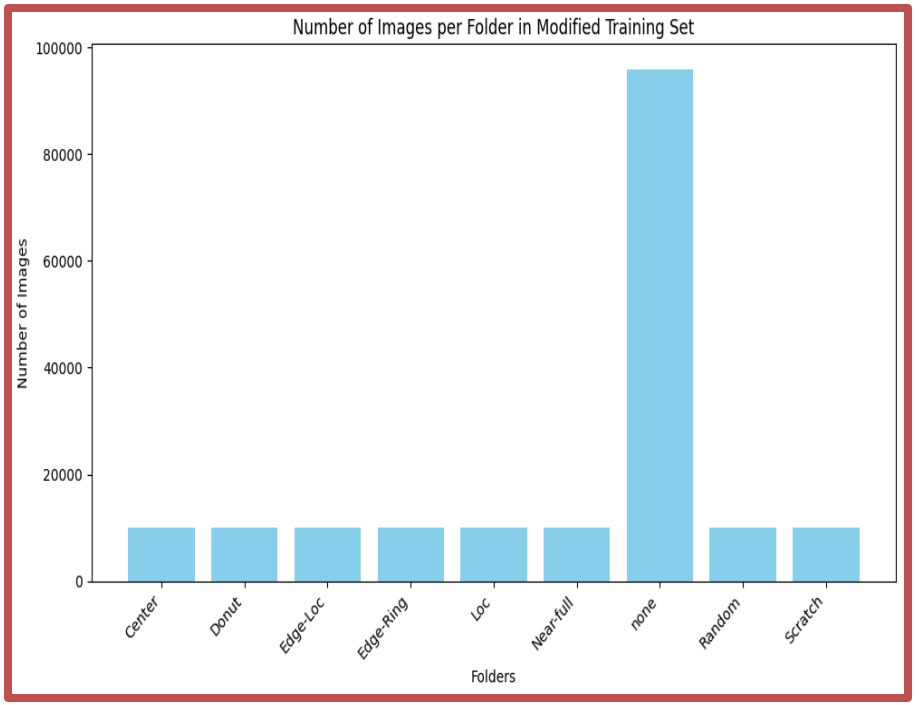
데이터셋_실행결과

데이터 증강

(기존) DATA SET



(변경) DATA SET 증강



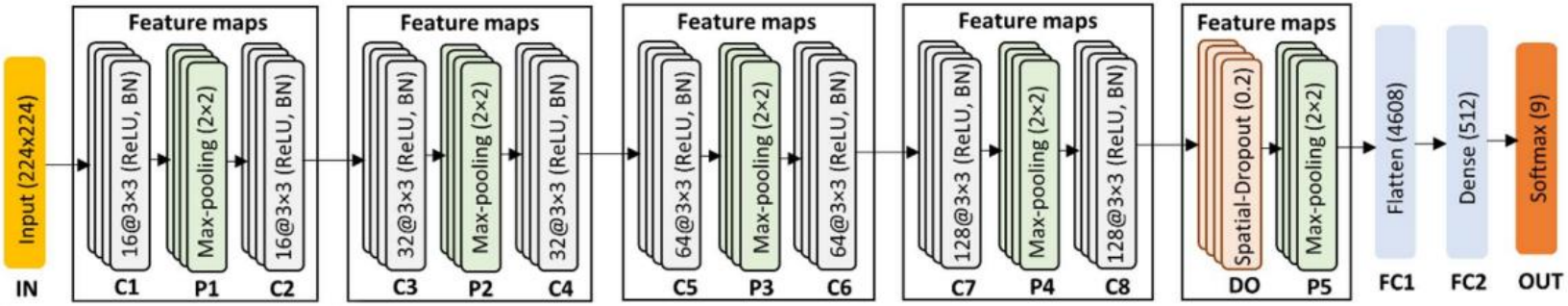
None	Center	Donut	Edge-Loc	Edge-Ring	Local	Random	Scratch	Near-full
24,252	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000

‘Train’ 세트에 대해서만 데이터 증강 수행

CNN 구조

CNN 구조

- ✓ **Model Architecture** : 여러 개의 컨벌루션 및 조밀한 레이어를 갖춘 CNN 구조 및 ReLU 활성화 함수를 사용, 다중 클래스 분류를 위한 Softmax 활성화
- ✓ **Performance Evaluation** : 정확도를 측정하기 위해 테스트 데이터를 기반으로 모델을 평가하였으며 혼돈 행렬, 정밀도, 재현율 및 F1 점수를 평가
- ✓ **Visualization** : 다양한 컨볼루셔널 레이어의 기능 맵 시각화, 입력 이미지 모델의 반응



*Note: IN denotes input layer; C convolutional layer; P pooling layer; DO dropout layer; FC fully connected layer; OUT output layer; and BN batch normalization

CNN 구조_모델 아키텍처

모델 아키텍처 정의

```python
# 모델 아키텍처 정의
class Create_CNN(nn.Module):
    def __init__(self):
        super(Create_CNN, self).__init__()
        # Conv-Pool-Conv 그룹 1
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=16, kernel_size=3, padding=1)
        self.bn1 = nn.BatchNorm2d(num_features=16)
        self.pool1 = nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv2 = nn.Conv2d(in_channels=16, out_channels=16, kernel_size=3, padding=1)
        self.bn2 = nn.BatchNorm2d(num_features=16)

        # Conv-Pool-Conv 그룹 2
        self.conv3 = nn.Conv2d(in_channels=16, out_channels=32, kernel_size=3, padding=1)
        self.bn3 = nn.BatchNorm2d(num_features=32)
        self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv4 = nn.Conv2d(in_channels=32, out_channels=32, kernel_size=3, padding=1)
        self.bn4 = nn.BatchNorm2d(num_features=32)

        # Conv-Pool-Conv 그룹 3
        self.conv5 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, padding=1)
        self.bn5 = nn.BatchNorm2d(num_features=64)
        self.pool3 = nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv6 = nn.Conv2d(in_channels=64, out_channels=64, kernel_size=3, padding=1)
        self.bn6 = nn.BatchNorm2d(num_features=64)

        # Conv-Pool-Conv 그룹 4
        self.conv7 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, padding=1)
        self.bn7 = nn.BatchNorm2d(num_features=128)
        self.pool4 = nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv8 = nn.Conv2d(in_channels=128, out_channels=128, kernel_size=3, padding=1)
        self.bn8 = nn.BatchNorm2d(num_features=128)

        self.dropout = nn.Dropout2d(p=0.2)  # Spatial Dropout

        # 완전연결 계층
        self.fc1 = nn.Linear(in_features=128 * 4 * 4, out_features=512)
        self.bn_fc1 = nn.BatchNorm1d(num_features=512)

        self.fc2 = nn.Linear(in_features=512, out_features=9)  # 클래스 수에 맞게 조정

    def forward(self, x):
        x = F.relu(self.bn1(self.conv1(x)))
        x = self.pool1(F.relu(self.bn2(self.conv2(x))))
        x = F.relu(self.bn3(self.conv3(x)))
        x = self.pool2(F.relu(self.bn4(self.conv4(x))))
        x = F.relu(self.bn5(self.conv5(x)))
        x = self.pool3(F.relu(self.bn6(self.conv6(x))))
        x = F.relu(self.bn7(self.conv7(x)))
        x = self.pool4(F.relu(self.bn8(self.conv8(x))))

        x = self.dropout(x)  # Spatial Dropout 적용

        x = x.view(-1, 128 * 4 * 4)  # Flatten the tensor
        x = F.relu(self.bn_fc1(self.fc1(x)))
        x = self.fc2(x)

        return F.log_softmax(x, dim=1)  # Softmax 활성화 함수
```

CNN 구조

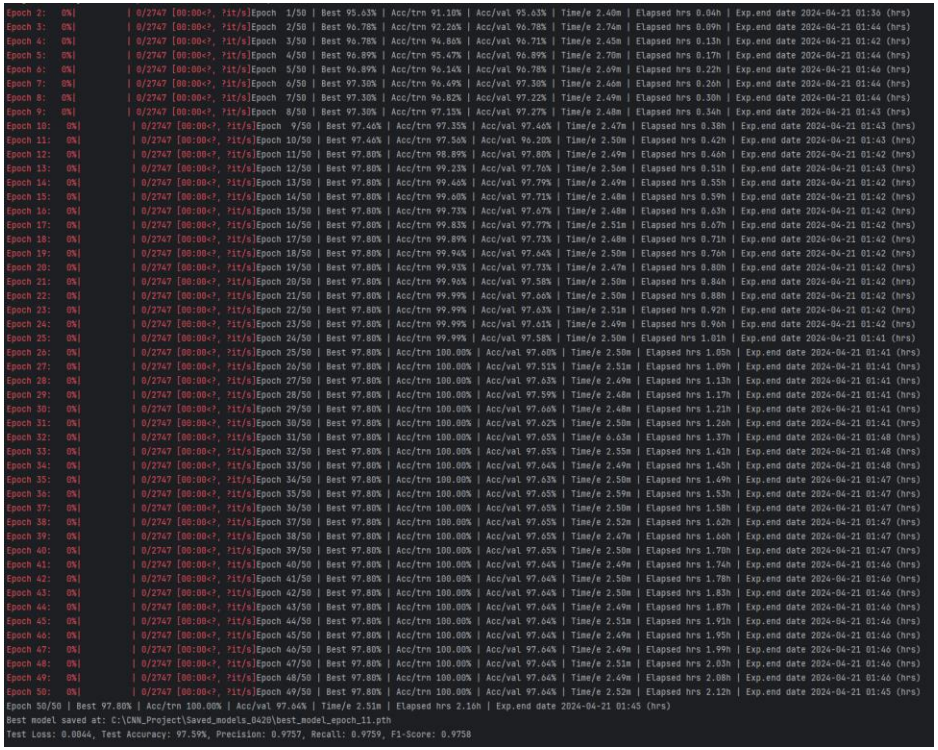
주요 코드 및 실행 결과

- ✓ 딥러닝 프레임워크(PyCharm, pytorch, Python)
- ✓ 오토인코더 모델 생성 및 학습(epoch = 50, batch_size = 1024)

학습 코드

```
for epoch in range(num_epochs):  # 전체 데이터셋을 여러 번(50) 반복
    running_loss = 0.0
    correct = 0
    total = 0
    epoch_start = datetime.datetime.now()

    for images, labels in tqdm(train_loader, desc=f'Epoch {epoch+1}', leave=False):
        images, labels = images.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
```

학습 결과

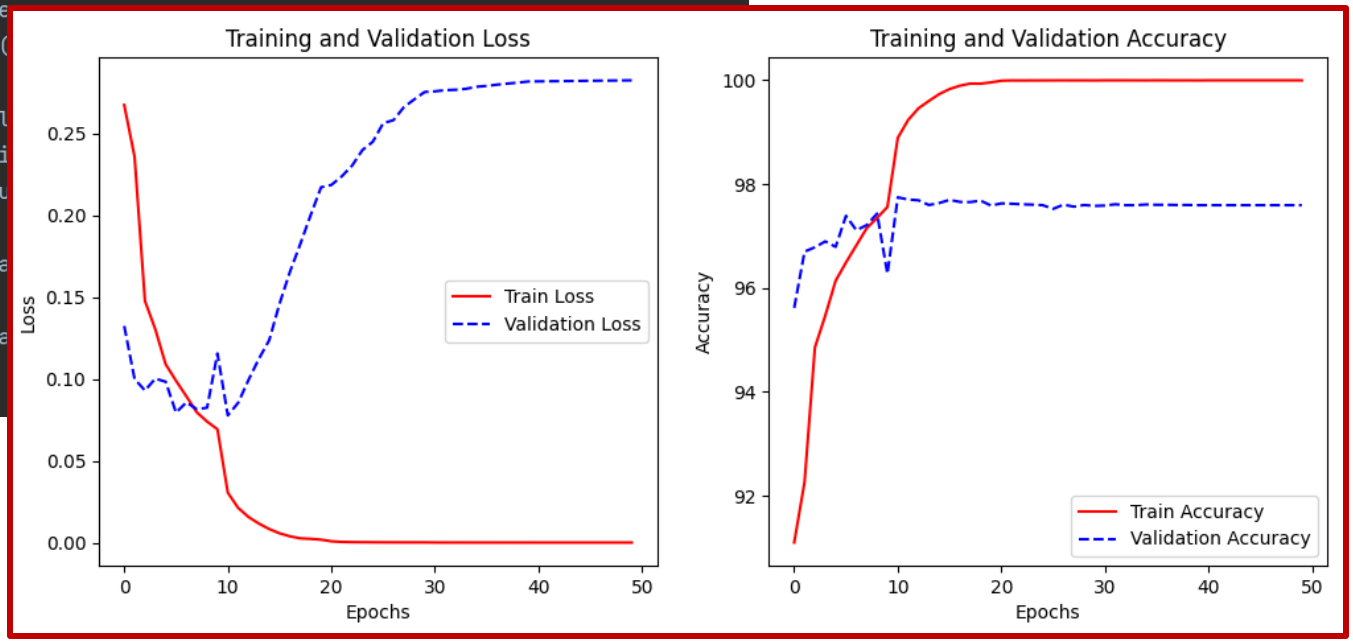


CNN 시각화_정확도 및 손실

정확도 및 손실 상세코드

```python
# 모델, 손실 함수 및 최적화 알고리즘 인스턴스화
model = Create_CNN().to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.01)
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=10, gamma=0.1)  # 학습률 스케줄러 설정

def validate(model, val_loader, criterion):
    model.eval()
    val_loss = 0.0
    correct = 0
    total = 0
    with torch.no_grad():
        for images, labels in val_loade
            images, labels = images.to(
            outputs = model(images)
            loss = criterion(outputs, l
            val_loss += loss.item() * i
            _, predicted = torch.max(ou
            total += labels.size(0)
            correct += (predicted == la
    val_loss /= total  # 평균 손실 계산
    val_accuracy = 100 * correct / tota
    return val_loss, val_accuracy
```

정확도 및 손실도 시각화

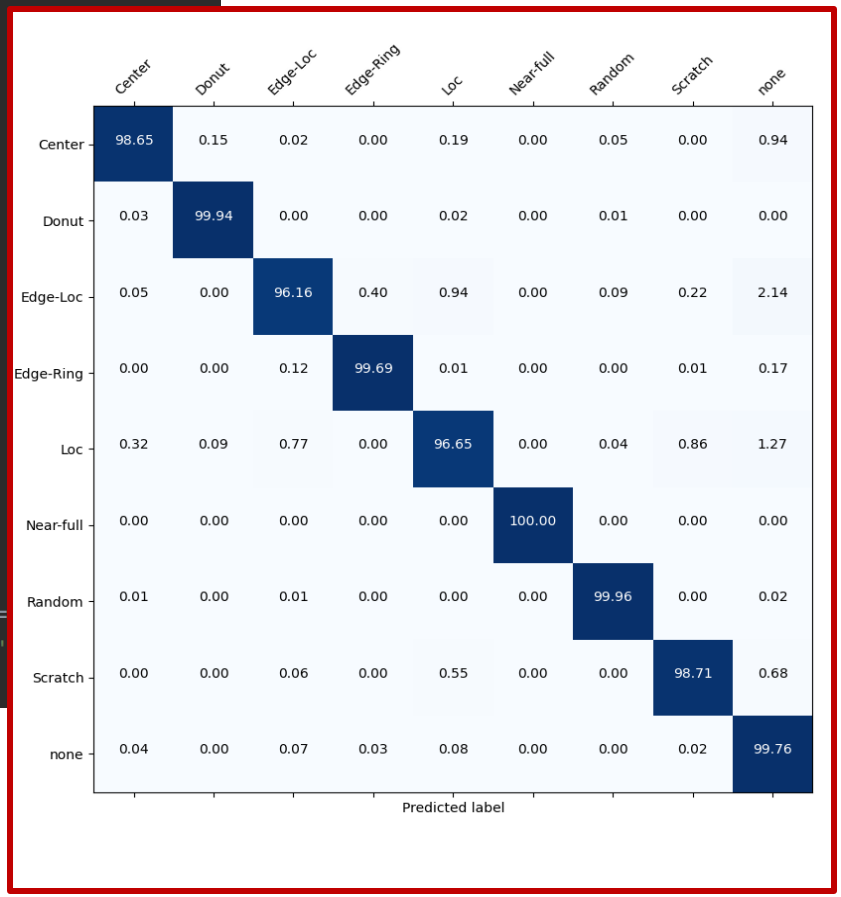


CNN 시각화_혼돈행렬

혼돈행렬 상세코드

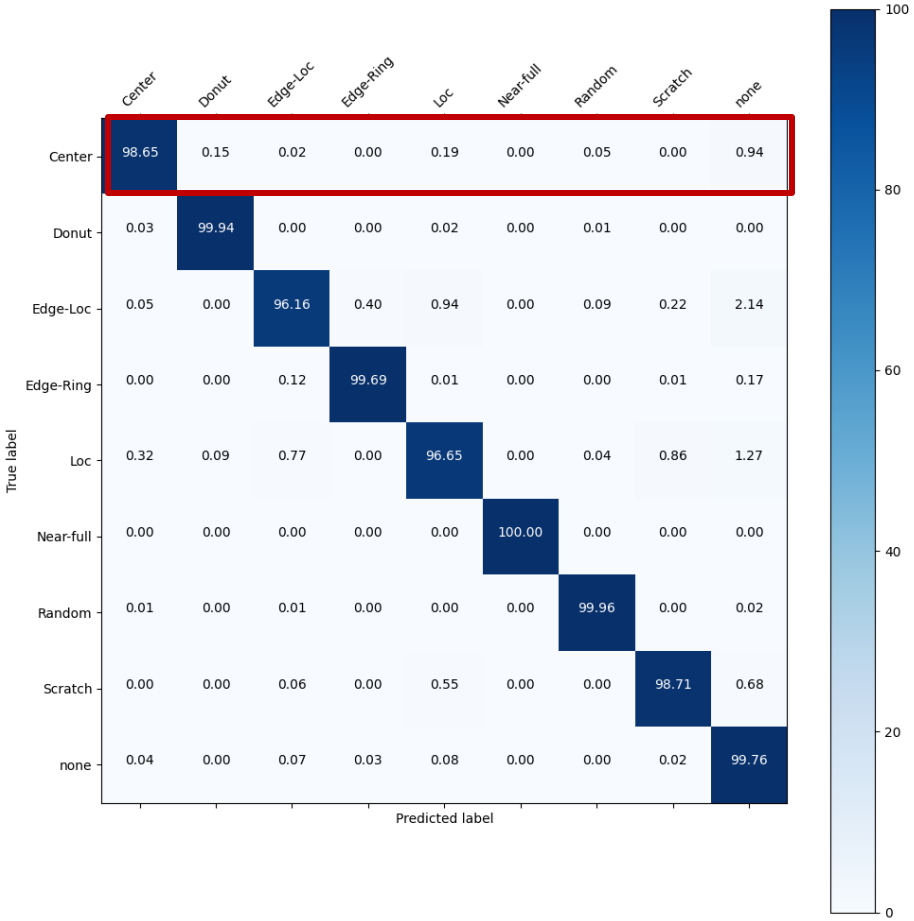
```python
# Confusion Matrix 계산
def compute_confusion_matrix(model, data_loader, device):
    nb_classes = 9
    confusion_matrix = np.zeros((nb_classes, nb_classes))
    with torch.no_grad():
        for i, (inputs, classes) in enumerate(data_loader):
            inputs = inputs.to(device)
            classes = classes.to(device)
            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)
            for t, p in zip(classes.view(-1), preds.view(-1)):
                confusion_matrix[t.long(), p.long()] += 1
    return confusion_matrix

# Train 및 Test 데이터셋에 대한 모델 평가
train_confusion_mtx = compute_confusion_matrix(model, train_loader, device)
test_confusion_mtx = compute_confusion_matrix(model, test_loader, device)
class_names = train_data.classes

# 두 데이터셋의 오차 행렬을 시각화하는 함수
def visualize_performance(train_cm, test_cm, classes):
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))
    plot_confusion_matrix(train_cm, classes=classes, ax=ax1, normalize=True, title=
    plot_confusion_matrix(test_cm, classes=classes, ax=ax2, normalize=True, title='
    plt.show()
```

혼돈행렬 시각화

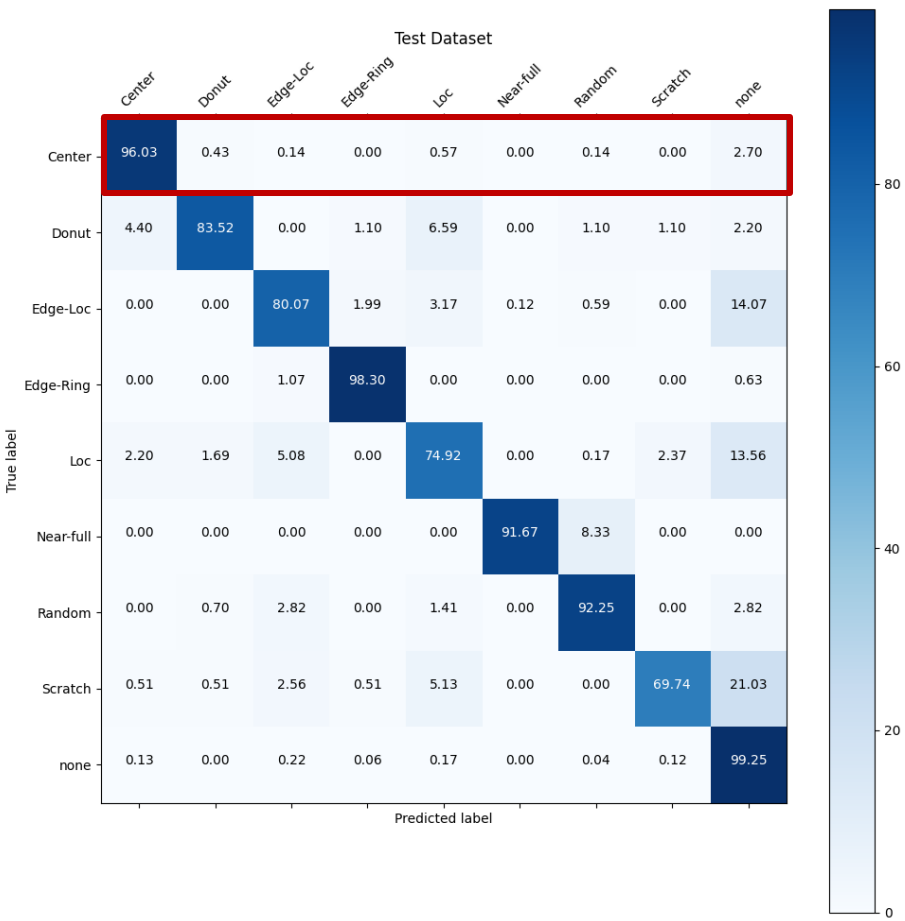


CNN 시각화_Balanced data 비교

Confusion matrix 비교(actual label 기준 비율)



Train Dataset



Test Dataset

학습 방법

딥러닝 학습 조건

- **(HW) PC 사양, 학습시간**

  CPU : Intel Xeon CPU E5-2696 v5 @ 4.40GHz

  RAM : 512GB

  GPU : NVIDIA GeForce GTX 1080 24GB

GPU 사용량

```
C:\Users\user>nvidia-smi
Fri Apr 12 20:09:10 2024
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 531.79                 Driver Version: 531.79       CUDA Version: 12.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name                     TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf          Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                                       |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce GTX 1650 T...  WDDM | 00000000:02:00.0 Off |                  N/A |
| N/A   64C    P0              33W /  N/A|   1360MiB /  4096MiB |     48%      Default |
|                                       |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

모델의 성능

딥러닝 학습 결과

- 불균형 데이터셋과 균형잡힌 데이터셋에 대한 모델의 성능 비교

	Precision	Recall	F1-score
Center	99.2	98.6	98.9
Donut	99.8	99.9	99.8
Edge-Loc	98.3	96.2	97.2
Edge-Ring	99.3	99.7	99.5
Loc	97.5	96.6	97.1
Near-full	100.0	100.0	100.0
Random	99.8	100.0	99.9
Scratch	98.7	98.7	98.7
none	99.5	99.8	99.6
Average	99.1	98.8	99.0

Train Dataset

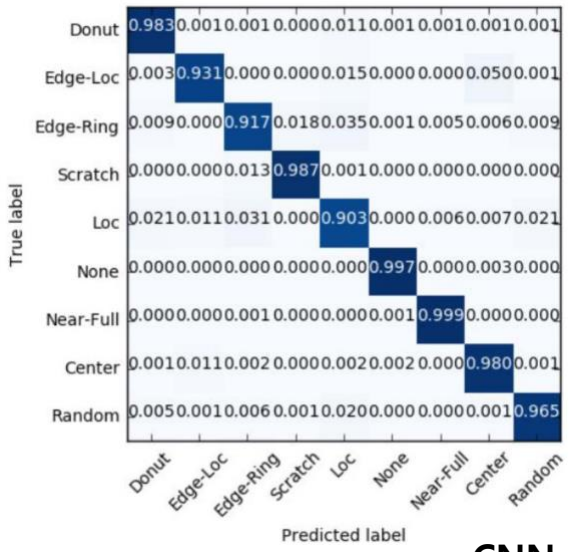
	Precision	Recall	F1-score
Center	93.3	96.0	94.6
Donut	82.6	83.5	83.1
Edge-Loc	86.0	80.1	82.9
Edge-Ring	97.9	98.3	98.1
Loc	83.1	74.9	78.8
Near-full	95.7	91.7	93.6
Random	87.3	92.3	89.7
Scratch	75.1	69.7	72.3
none	98.9	99.3	99.1
Average	88.9	87.3	88.0

Test Dataset

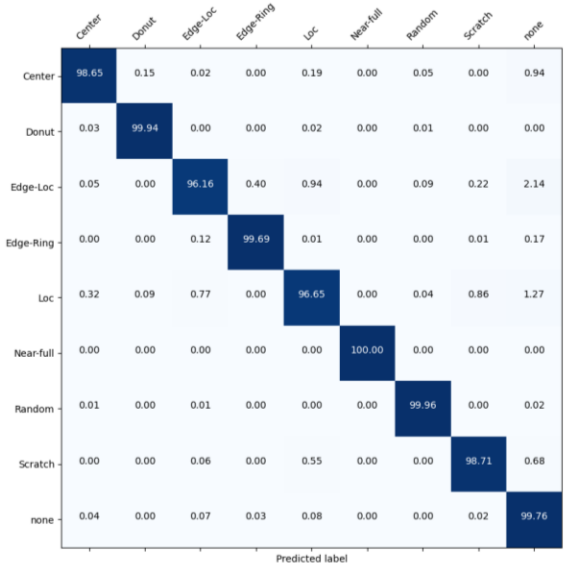
결과 및 토의

모델별 분류 성능 비교

- CNN-CREATE 모델이 전반적으로 높은 성능을 보임.



CNN-WDI



CNN-CREATE

Classfier	Training Acc	Validation Acc	Testing Acc	Precision	Recall	F1-Score
CNN-CREATE	98.8	**97.8**	**97.8**	**97.5**	**97.5**	**97.5**
CNN-WDI	98.9	96.4	96.2	96.2	96.2	96.2
CNN-D	97.6	95.5	95.2	95.2	95.2	95.2
CNN-BN	**99.4**	95.6	95.6	95.6	95.6	95.6
CNN-SD	98.6	94.7	94.8	94.8	94.8	94.8
VGG-16	82.3	80	80.1	80.3	80.1	79.9
ANN	95.9	95.9	72	95.2	95.9	95.4
SVM	91.3	91	32.6	87.5	91	88

결론

결과 요약 및 의미

- CNN-CREATE 모델은 일관되게 높은 정확도, 성능 지표 나타냄. 해당 문제에 대한 특화된 접근 방식과 최적화를 통해 강력한 일반화 능력을 갖춤
- 맞춤형 아키텍처, 고급 성능 지표의 향상, 효율적인 학습 전략을 결합하여 높은 수준의 성과를 달성

개선점

- 계산 효율성 측면에서 개선하고자 모델의 계산 복잡성을 낮추고 추론 속도를 향상시키는 방안 탐색
- 모델의 Robustness를 추가적으로 평가하고, 특정 클래스에 대한 분류 성능을 개선하기 위한 추가적인 데이터 증강 기법 고려

감사합니다