Pusan National University
Computer Science and Engineering
Technical Report 2023-10

인공지능 비전 기반 RT 필름 결함 자동판독 기술

저자 1 201714532 이현규

저자 2 201824640 지민철

저자 3 201824170 정다현

지도교수 김 종 덕 교수

목 차

1. 서론

1.1. 연구 배경

조선소 및 기타 산업 분야에서의 용접 작업은 구조물 및 선박 제조에 있어 핵심적인 과정이다. 이러한 용접 부위에서의 결함은 구조물의 강도와 내구성에 영향을 미칠 수 있으며, 이를 조기에 발견하지 못하면 심각한 인명 사고 및 경제적 손실이 발생할 수 있다. 따라서 검사 기술을 통해 용접 부위의 결함을 탐지하고 조처하는 것은 필수적이다. 파괴 검사와 비 파괴 검사 중 비 파괴 검사 기술은 더 안전하고 신뢰할 수 있는 제품 생산을 지원하며, 구조물 및 선박의 수명을 증가하는 역할을 한다. 또한 경제적으로도 유리한 부분이 존재한다. 이러한 검사 방법 중 하나인 Radiography Film(이하 RT 필름) 촬영 기술은 우수한 결과를 제공하고 있으며, 안전한 제품 제조, 구조물 및 선박의 수명 연장에 기여하고 있다.

RT 필름은 이러한 RT 필름 촬영 기술을 활용해, 결함을 판독하기 쉽도록 시각적으로 기록하는 매체이다. X선 또는 감마선을 이용하여 물체 내부의 결함, 용접 부위의 불완전성, 금속 구조의 두께 등을 시각적으로 확인할 수 있게 기록한다. 이후, 검사원을 투입하여 결함을 판단하게 되는데 다수의 **RT 필름을 판독하려면 많은 인적 자원이 필요하며 시간 또한 많이 소모**되게 된다.

인공지능 비전(Artificial intelligence Vision) 기술은 이러한 문제점에 대처하고 향상된 판독 능력, 정확성 및 효율성을 제공할 수 있는 중요한 도구이다. 인공지능 비전 기술은 기계 학습과 이미지 분석을 통해 시각적 데이터를 이해하고 판단하는 기술이다. 특징 추출, 객체 감지, 분류, 세분화, 패턴 인식 등의 기술을 활용하여 이미지 내의 정보를 추출하고 해석하여, 인간보다 빠른 속도로 정확하게 시각적 데이터를 분석할 수 있다. 이를 활용한다면 **인간 검사원 대신 인공지능을 활용해 빠른 속도로 정확하게, 적은 비용을 사용하여 RT 필름을 판독하여 용접 부위의 결함을 파악하고 조처할 수 있다.** 이런 방식을 사용한다면 앞서 제기한 검사원을 통한 용접 부위 결함 판독에서 생기는 문제점, 정확도, 신뢰성, 속도 문제 등의 해결을 기대할 수 있다. 따라서 **본 과제에서는 인공지능 비전 기술을 활용해 RT 필름을 판독**하고자 한다.

1.2. 기존 문제점

기존에는 영사한 RT 필름을 검사원이 직접 눈으로 확인하고 결함의 유무, 결함의 위치, 크기, 종류 등을 분석하였다. 하지만 이는 여러 가지 한계점을 가지고 있다. 검사원이 직접 판단하기 때문에 주관적 판단이 개입할 수 있으며, 이에 따라 결과의 일관성이 부족할 수 있다. 또한, 검사원은 피로, 집중력 부족 또는 작업 중의 실수로 인해 잘못된 결과를 도출해 낼 수 있다. 대량의 데이터를 검사원이 직접 분석하는 데는 시간과 인적 자원이 많이 소모되며, 이에 따라 낮은 효율성을 보인다. 검사원의 눈으로는 미세한 결함이나 복잡한 패턴의 감지가 어려울 수 있다.

1.3. 연구 목표

본 연구의 목표는 **그림 1**과 같이 영사된 RT 필름을 서버에 업로드 후 인공지능 모델을 사용하여 용접 부위를 추출하는 것이다. 이 연구를 하는 이유는 본 연구 이후에 용접 부위를 인공지능 모델을 사용하여 분석 후, 결함의 유무, 결함의 위치, 크기, 종류 등을 분석하여 결과를 도출해 주는 것이 목표이기 때문이다. 그러한 목표를 달성하기 위해서는 본 과제의 목표인 용접 부위 추출이 필수적이다. 그 이유는 결함 탐지 방법과 관련이 있다.

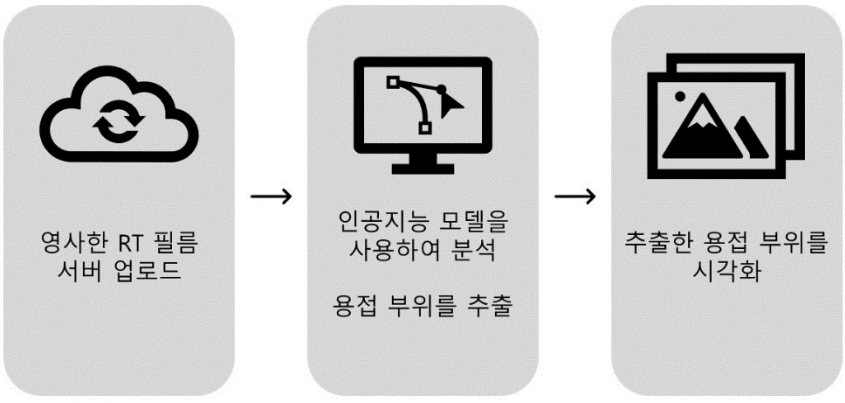


그림 1 연구 목표

결함을 탐지하는 방법에는 여러 가지 방법이 있다. 그 중에서 본 과제에서 사용하는 RT 필름에서 결함을 탐지하는데 적용할 수 있는 방법은 두 가지가 있다. 한 가지 방법은 **각 결함의 종류를 인공지능 모델에 직접 학습시켜 결함을 탐지하는 지도학습 방법**이다. 조선소에서 볼 수 있는 용접 결함별로 이미지 라벨링을 한 후에 이를 모델에 학습시켜 결함을 탐지하는 모델을 생성해 낼 수 있다. 두 번째로는 **정상 용접 부위의 위치만을 학습하는 비지도학습 방법**이다.

우리는 첫 번째 지도학습 방법을 목표로 하였으나 적용하기엔 문제점이 존재했다. **결함 자료가 매우 부족하다는 점**이다. 결함의 종류는 여러 종류인 총 9가지이다. 불충분한 결함 데이터에 더해, 결함 종류도 여러 가지이다 보니 각 사례에 대한 데이터는 더욱 부족할 수밖에 없다. 따라서 각 결함을 인공지능 모델에 직접 학습시킬 만한 충분한 데이터를 수집하기 어렵다. 그렇기에 **각 결함을 인공지능 모델에 직접 학습시켜 결함을 탐지하는 지도학습 방법은 현실적으로 불가능**하다.

따라서, 두 번째 **비 지도학습 방법을 채택하는 것이 현실적이고 효과적인 접근**일 것이다. 초기 학습 단계에서 용접 부위의 이미지를 인공지능 모델에 제공하여 모델이 이미지의 특징과 구조를 파악하도록 한다. 이런 과정을 통해 모델은 용접 부위를 식별하고, 이를 기준으로 **용접 부위를 예측하는 능력을 학습**한다.

이를 위해 선행하여야 하는 중요한 과제가 있다. 용접 부위를 정확하게 분리하고 추출하는 것이다. 이를 위해서 용접 부위의 경계를 정의하고, 필요 없는 부분을 제거하여 모델이 용접 부위를 정확하게 학습하도록 보정한다. 이렇게 얻은 용접 부위의 이미지를 기반으로 모델은 정상 용접 부위를 정확하게 인식하게 된다.

이러한 접근 방식은 효과적인 결함 감지 시스템을 구축하는 것의 발판이 될 수 있다. 따라서 **연구의 본 목표인 인공지능 모델을 학습하여 불량 검출을 수행하는 것뿐만 아니라 대체 목표인 용접 부위를 예측하고 추출해내는 것도 결함 판독을 위해 반드시 선행되어야 할 중요한 과제**이다.

따라서, 먼저 **용접 부위를 정확히 추출**해낸다. 이후 추출한 용접 부위 데이터를 통해 비지도 학습을 통해 정상 용접 부위를 인식하게 한다. 이후 훈련된 모델을 바탕으로 **용접 부위의 결함을 자동으로 판독**하게 한다.

2. 연구 배경

2.1. RT 필름

RT 필름은 RT 필름 촬영 기술을 활용해, 결함을 판독하기 쉽도록 시각적으로 기록하는 매체이다. X선 또는 감마선을 이용하여 물체 내부의 결함, 용접 부위의 불완전성, 금속 구조의 두께 등을 시각적으로 확인할 수 있게 기록한다. RT 필름은 투과한 방사선에 반응하여 투과한 방사선이 많은 경우 상대적으로 밝은 빛을 띠며, 투과한 방사선이 적은 경우 상대적으로 어두운 빛을 띠게 된다. 두꺼운 부분은 상대적으로 적은 방사선이 투과하고, 얇은 부분은 상대적으로 많은 양의 방사선이 투과하기 때문에, RT 필름을 보면 금속 구조의 두께를 확인할 수 있다. 파이프의 용접 부위를 찍은 **그림 2**에서 이를 확인할 수 있다.

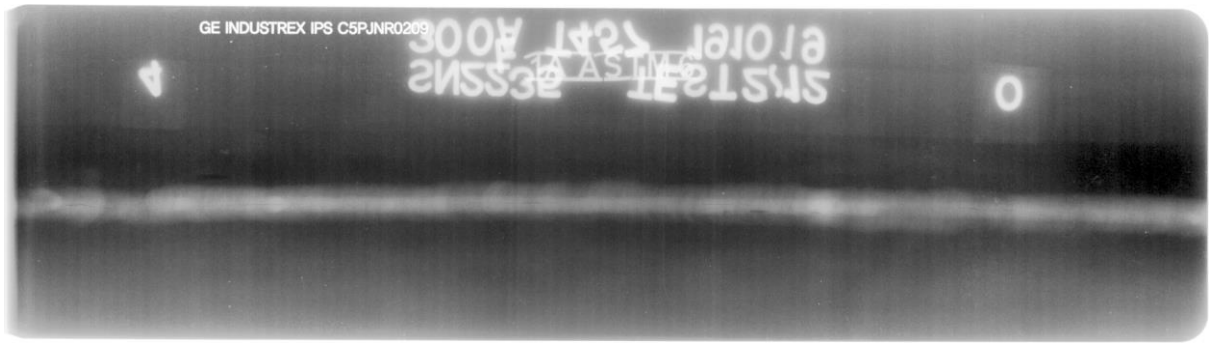


그림 2 RT 필름 예시

용접 금속의 용입으로 인해 용접 부위는 상대적으로 주변보다 두꺼워지게 된다. 따라서 적은 양의 방사선이 투과하고, RT 필름에서는 상대적으로 밝은 부분으로 나타나게 된다. 용접 부위가 아닌 부분은 상대적으로 용접 부위보다 얇기 때문에 용접 부위에 비해 상대적으로 어둡다.

만약에 용접 부위에 결함이 있다면 RT 필름에 검은 점이나 선이 나타나게 된다. 이를 **그림3**에서 확인할 수 있다. 이 외에도 용접 부위에 과도한 덧살이 있는 경우 하얀 점이나 선으로 나타난다.

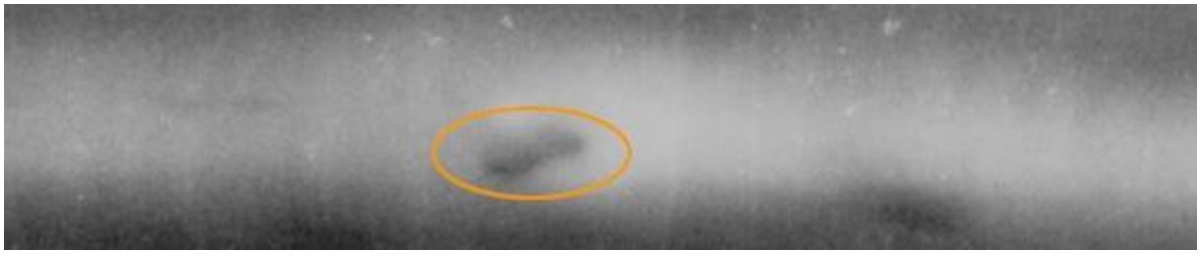


그림 3 용접 부위 결함 예시

이를 통해 결함의 위치, 크기, 밀도 및 형태를 판독한다. 결함은 총 9가지가 있다. 균열, 용입 과다, 용입 부족, 융합 부족, 기공, 루트 오목, 슬래그 라인, 텅스텐 혼입, 언더 컷이 용접 부위에서 발견할 수 있는 9가지 결함이다. 균열은 가장 드문 결함으로 용접 부위에 균열이 생긴 결함이다. 용입 과다는 용접 재료나 용접 부위가 과도하게 용융되어 다른 불순물이나 잘못된 금속 성분이 용접 부위에 혼입된 결함이다. 용입 부족은 용접 작업에서 필요한 용접재료가 충분히 용융되지 않거나 용접 부위에 충분한 금속이 혼합되지 않은 결함이다. 융합 부족은 용접 재료가 충분히 용융되지 않아 용접 부위의 강도나 내구성이 저하되는 결함이다. 기공은 용접 부위 내에, 기체나 공기가 갇혀 있는 작은 구멍이나 공간이 생기는 결함이다. 루트 오목은 용접의 시작 부위인 루트가 오목하게 파인 현상으로 충분히 금속이 충전되지 않아 부족한 부분이 오목하게 파인 결함이다. 슬래그 라인은 주로 용접재료와 용접 부위 사이에 용접 부산물 중의 불순물인 슬래그가 갇히거나 남아 있는 상황에서 발생하는 결함이다. 텅스텐 혼입은 용접에 사용되는 텅스텐 전극이 녹아서 용접 부위에 혼입된 결함이다. 언더 컷은 용접 부위 표면에서 용접 부위와 모재와의 연결부에 파인 홈으로 용접 중에 불필요하게 높은 온도로 가열되어 용접 부위가 녹은 현상을 말한다. 이 결함은 RT 필름을 통한 결함 판독보다는 실제 맨눈으로 용접부를 확인하여 검사하는 것이 일반적이다.

이 중 가장 치명적인 결함은 균열로 가장 큰 내구성 저하를 불러오지만, 발생 빈도는 가장 낮다. 융합 부족이 그다음으로 치명적인 결함으로, 결함이 응력 집중 지점이 되어 치명적인 내구성 저하를 유발한다. 발생 빈도 또한 9가지 결함 중에 두 번째 높다. 용입 부족도 마찬가지로 결함이 응력 집중 지점이 된다. 가장 많이 발견되는 결함은 기공이다. **아래 표 1을 보면 발생 빈도와 특이 사항을 확인할 수 있다.**

번호	이름	형태	빈도	특이 사항
1	균열	선	9위	가장 치명적 결함, 발생 빈도 가장 낮음
2	용입 과다	선	공동 3위	
3	용입 부족	선	공동 5위	융합 부족과 비슷한 위험도
4	융합 부족	선	2위	두 번째 치명적인 결함
5	기공	점	1위	발생 빈도 가장 높음
6	루트 오목	선	공동 3위	
7	슬래그 라인	선	공동 5위	
8	텅스텐 혼입	점	공동 5위	
9	언더 컷	선	공동 5위	RT 필름을 통한 탐지를 하지 않음

표 1 결함의 종류

본 연구에서는 조선소에서 촬영한 파이프의 용접 부위 RT 필름 데이터를 가지고 연구를 수행하고자 한다. 데이터는 총 3가지로 분류할 수 있다. 관의 크기에 따라 대형관, 중형관, 소형관이 있다.

	소 형	중 형	대 형
정 상	42	101	106
결 함	7	98	109
총 수	49	199	215

표 2 이미지의 종류와 수

대형관과 중형관, 소형관 이미지의 차이가 크기 때문에 같은 모델로 학습한다면 좋은 결과를 기대하기 어려워 보인다. 따라서 본 과제에서는 대형관, 중형관, 소형관 각각을 인공지능 모델에 학습하기로 한다. 또한 데이터 세트가 작기 때문에 본 연구 과정에서 이를 최대한 보완하며 진행하도록 한다.

2.2. 인공지능 비전

인공지능 비전은 현대 산업 분야에서 빠르게 성장하고 있는 기술 중 하나로, 시각적 정보를 이해하고 분석하는 데 인공지능을 활용하는 분야이다. 이 기술은 이미지와 비디오 데이터를 처리하며, 객체 감지, 분류, 세분화, 패턴 인식 등의 작업을 수행한다. 본 연구에서는 이러한 인공지능 비전 기술을 활용하여 용접 부위에서의 결함을 탐지하고 용접 부위 결함 탐지 품질을 향상하는 방안을 탐구한다.

인공지능 비전 기술을 사용하기 위해서는 여러 가지 인공지능 모델을 고려하고 가장 적합한 모델을 선정해야 한다. 인공지능 모델들의 특징을 조사하고 그 중에 몇 가지를 골라 특징을 비교하여 본 과제에 가장 적합한 모델을 선정했다. 우선, RT 필름 이미지에서 용접 부위를 추출해 낼 모델이 필요했다. 가장 우선순위에 둔 것은, **적은 데이터 세트로도 충분한 예측을 할 수 있는 모델이다. 데이터의 양이 충분하지 않기에 가장 우선**해야 했다. 이를 고려하여 후보로 고른 모델들은 3종류이다.

첫 번째 고려한 인공지능 모델은 Convolutional Neural Network(이하 CNN)이다. 이미지를 처리하는 데 특화된 모델이다. CNN은 이미지를 여러 개의 합성곱 레이어(convolutional layer)와 풀링 레이어(pooling layer)를 통과하면서 이미지로부터 특성을 추출하고 이미지를 단순화한다. 이중 합성곱 레이어는 입력 이미지에 커널(필터)을 적용하여 각 지점에서 특징 맵을 생성하는 역할을 한다. 이러한 합성곱 레이어는 이미지의 특정 패턴 및 특성을 감지하도록 설계되어 있다. 풀링 레이어는 특징 맵의 크기를 줄이고

이미지를 단순화하여 계산을 줄이는 역할을 한다. 이는 네트워크의 복잡성을 줄이고, 중요한 정보를 보존하면서 이미지의 공간 구조를 보다 효과적으로 학습하는 데 도움을 준다. 이러한 특징 추출 능력은 이미지 분류, 객체 감지, 얼굴 인식 및 다양한 컴퓨터 비전 작업에 유용하게 활용된다. CNN은 이미지 처리 분야에서 좋은 성과를 보이며, 딥 러닝 기술의 중요한 부분을 차지하고 있다. 또한 수많은 파생 모델이 존재하여, 더 발전된 CNN 모델을 필요에 맞추어 골라서 사용할 수 있다. 따라서 많은 연구에서 이를 활용하고 있어, 충분한 참조를 할 수 있는 장점이 있다. 다만 CNN은 적은 데이터 세트로 충분한 결과를 뽑아낼 것을 기대하기 어렵기 때문에 본 연구에 사용할 수 없었다.

두 번째 고려한 인공지능 모델은 ResNet이다. 이는 일반적인 인공지능 모델이 레이어를 많이 사용하면 할수록 성능이 좋아지지만, 일정 이상으로 많이 사용하면 오히려 성능이 저하되는 문제를 해결하기 위해 나온 모델이다. 레이어가 많으면 오차를 보정하기 위한 gradient가 레이어 깊은 층까지 전달되지 못하기 때문에 보정되지 못한 오차가 많이 생기게 된다. gradient란 손실 함수에 대한 입력 데이터의 가중치 및 편향의 편미분 값들이다. 손실 함수는 출력과 정답(label) 간의 차이로, gradient는 손실 함수가 최솟값을 찾기 위한 가중치 및 편향 조정 방향과 크기를 나타낸다. ResNet은 이러한 특성으로 많은 레이어를 사용할 수 있어, 적은 데이터로도 특징을 뽑아내 학습할 수 있어 본 과제에 적합하다. 하지만, ResNet 이상으로 적합한 모델이 있어 용접 부위 추출은 다른 모델을 활용하기로 했다.

세 번째로 고려한 모델이자 최종 선정된 모델은 UNet이다. UNet은 CNN 모델의 제한을 극복하기 위해 개발되었다. 기존의 CNN은 1차원 행렬에 대해서만 연산이 가능하고, 상세한 특징 추출이 어렵다는 한계가 있다. UNet은 FCN(Fully Convolutional Network)의 개념을 사용하여 모든 계층에서 합성곱 연산을 수행한다. FCN은 입력 이미지와 출력 분할 맵의 크기를 다르게 설정하여 픽셀 수준의 분할을 지원한다. 이를 통해 이미지 내에서 객체의 위치 및 윤곽을 정확하게 식별할 수 있게 되며, 픽셀 단위의 정확한 정보를 얻을 수 있다. UNet은 Contracting Path - Expanding Path 구조와 스킵 연결을 결합하여 더 정확하게 연산을 수행해 낸다. Contracting Path는 입력 이미지를 축소하고 특징을 추출하며, Expanding Path 는 추출된 특징을 확대하여 출력 이미지를 생성한다. 스킵 연결은 Contracting Path 와 Expanding Path 사이에 추가되어 높은 해상도와 저 수준의 정보를 보존한다. 이를 통해 보다 정확한 분할 결과를 얻고 빠른 연산이 가능해진다. 또한 주로 의료 현장에서 많이 사용되는데, 이는 본과제와 비슷한 환경이다. 데이터를 얻기 쉽지 않은 환경이기 때문에 데이터가 적다. 따라서 데이터가 부족한 본과제에서 비슷한 환경이라고 할 수 있다. 또한, 의료 현장에서 본 과제에서 사용하는 RT 필름과 비슷한 X-ray

와 같은 방사선 사진을 학습하는 데 많이 사용된다. 따라서 비슷한 데이터를 사용하는 본과제와 같은 환경이라고 할 수 있다.

다음 **그림 4**는 UNet의 구조를 보여준다. UNet의 이름처럼 U 형태의 구조를 가진 것을 볼 수 있다. U자 왼쪽이 Contracting Path, 오른쪽이 Expanding Path이며, 수평으로 연결된 화살표가 스킵 연결이다.

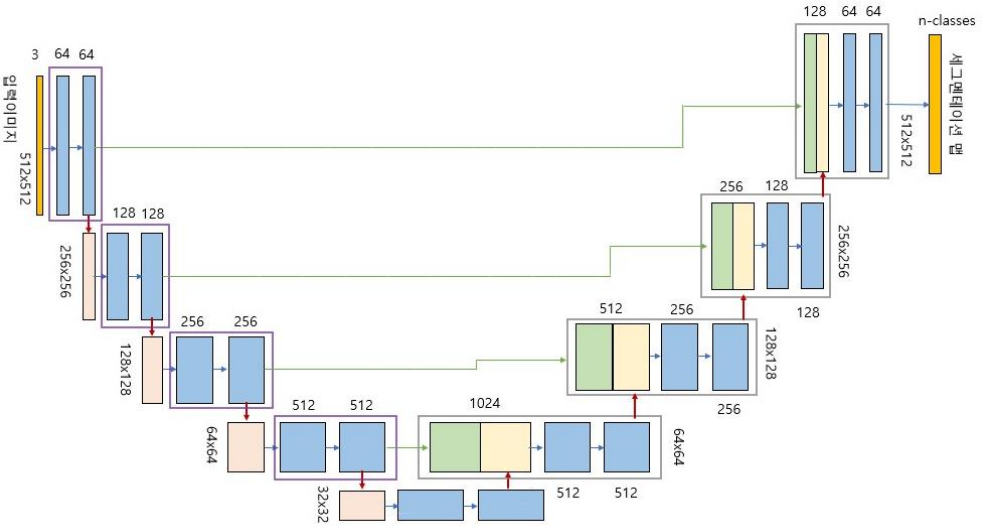


그림 4 UNet의 구조

Contracting Path에서는 이미지를 다운 샘플링하고 특징을 추출한다. Expanding Path에서는 업 샘플링을 통해 이미지를 원래 크기로 되돌리면서 예측을 수행한다. 또한 스킵 연결을 통해 Contracting Path에서 추출한 저차원 정보를 고차원 정보와 결합하여 정밀한 이미지 분석을 한다.

UNet의 Contracting Path와 Expanding Path는 그런 구조를 가지고 정밀한 예측을 수행할 수 있다. 이러한 특성들 덕분에, 본 과제에서 용접 부위를 추출하는데 사용할 모델로 UNet을 선정했다.

용접 부위 결함을 판독하는 모델 또한 선정이 필요했다. 용접 부위를 추출하는 모델을 생성하는데 꽤 오랜 시간이 걸렸기에, 간단한 방법을 사용하여 결함을 판독하고자 하여, 파이썬 라이브러리에서 제공하는 OCSVM(One Class Support Vector Machine) 방식인 sklearn 라이브러리를 사용해 보았다. 이는 데이터들을 N차원의 좌표축으로 뿌린 후, 원점과의 거리를 기준으로 불량을 판독한다. 하지만 이는 이미지 데이터 같은 고차원 데이터를 처리하기에는 어려웠고, 입력 데이터 형식도 1차원 입력밖에 받지 못해 충분히 결함을 탐지해내지 못했다. 정상 이미지의 70%를 정상으로 판독하였으나, 이상 이미지의 33.33%만을 이상 이미지라 판독해 일관성 없는 결과를 보여주었다. 다른 머신 러닝 방식

인 k-nearest neighbors algorithm(KNN) 알고리즘 또한 사용해 보았으나 이 또한 결과가 무작위 값에 가까워 정상적인 판독을 한다고 볼 수 없었다.

Autoencoder 방식 또한 사용해 보았다. 이 모델은 이미지를 저차원으로 인코딩했다가 디코딩하면서 이미지를 원래 상태로 복원하게 훈련된다. 이후 훈련된 모델은 이미지를 입력 받고, 이 이미지를 저차원으로 인코딩했다가 훈련된 정상 이미지에 가깝게 디코딩한다. 이때, 이상 이미지를 복원할 때는 정상 이미지보다 많은 복원 오차를 갖게 되고, 이를 기준으로 불량을 판독한다. 하지만, 용접 부위와 결함 부위가 큰 차이가 나지 않는 탓에, 이 또한 40% 정도의 낮은 판독률로 다른 모델 사용이 불가피하였다.

최종적으로 선정된 모델은 앞서 용접 부위 추출 모델 후보 중 하나였던 ResNet이었다. Cut-Paste를 사용하여 이상 이미지를 만들고, 이를 이용하여 정상 이미지와 이상 이미지를 분류하여 학습하는 형태로 모델을 제작하였다. 학습을 통해 정상 이미지와 이상 이미지의 특징은 떨어져서 각각 분포하게 된다. 이후 가우시안 밀도 추정을 이용하여 입력한 이미지가 정상 이미지에 가까운지, 이상 이미지에 가까운지 분석하게 되고, 그 결과를 바탕으로 이미지에 이상이 있는지 판독하게 된다.

2.3. IoU, Dice Coefficient

인공지능 모델을 사용해서 용접 부위 예측을 수행하면, 예측이 잘 되었는지 확인할 필요가 있다. 이를 위해 사용되는 평가 척도는 IoU, Dice coefficient 등이 있다. IoU, Dice Coefficient는 0~1 사이의 값을 가지며 실제 영역(Ground truth)과 예측 영역(Prediction)에 대한 수식으로 이루어져 예측을 평가한다. 그림을 통해 계산 방법을 볼 수 있다. IoU의 분모는 실제 영역과 예측 영역 합집합이고, 이와 달리 Dice Coef.의 분모는 실제 영역과 예측 영역의 넓이를 각각 더한 것이다.

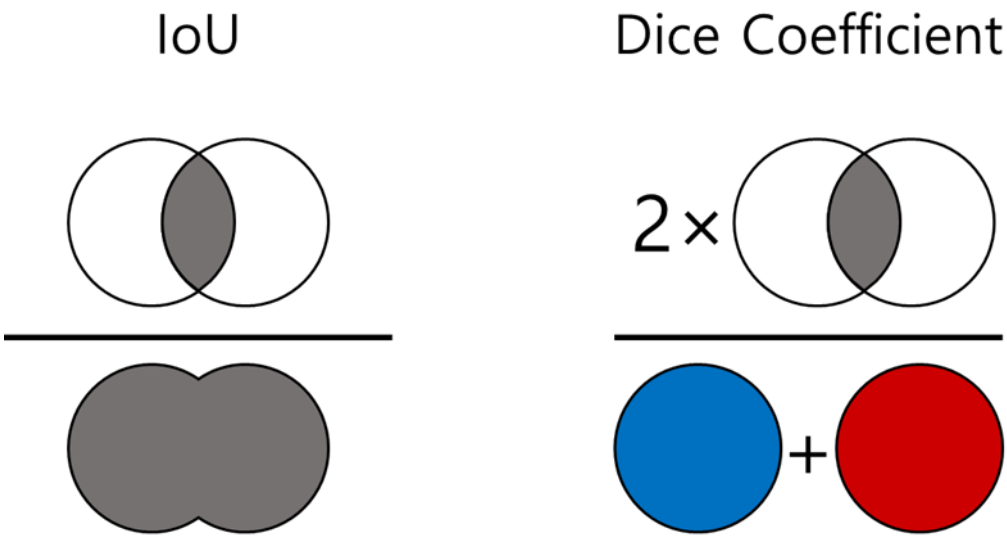


그림 5 IoU, Dice Coefficinet의 계산 방법

수식으로 나타내면 다음과 같다.

$$IoU = \frac{A \cap B}{A \cup B} = \frac{TP}{TP + FP + FN}$$

$$Dice = \frac{2|A \cap B|}{|A| + |B|} = \frac{2TP}{(TP + FP) + (TP + FN)} = \frac{2TP}{2TP + FP + FN}$$

IoU와 Dice Coefficient는 두 영역(예측 및 실제) 간의 유사성을 측정한다. IoU에 비해 Dice Coefficient는 겹치는 영역을 더욱 중요하게 본다. 실제 영역과 예측 영역의 교집합의 넓이에 2배의 가중치를 주었기 때문에, IoU 값보다 Dice coefficient 값이 크게 나온다. 따라서 물체 경계가 중요한 작업이라면 IoU가 Dice coefficient보다 적합하다고 할 수 있다. 객체 감지가 더 중요한 작업이라면 Dice coefficient가 Iou보다 적합하다. 예측 영역이 실제 영역보다 크거나 작더라도, 보정된 값을 계산하기 때문이다. 아래 **표 3와 4**에서 IoU와 Dice Coefficient의 차이를 볼 수 있다. **표 3**는 실제 영역과 예측 영역의 넓이가 같을 때, **표 4**은 실제 영역의 넓이보다 예측 영역의 넓이가 2배인 경우를 나타낸다.

교집합/실제 영역	IoU	Dice Coefficient
0.1	0.0526	0.1
0.2	0.1111	0.2
0.3	0.1765	0.3
0.4	0.25	0.4
0.5	0.3333	0.5
0.6	0.4286	0.6
0.7	0.5385	0.7
0.8	0.6667	0.8
0.9	0.8182	0.9
1.0	1.0	1.0

표 3 실제 영역과 예측 영역의 넓이가 같은 경우

교집합/실제 영역	IoU	Dice Coefficient
0.1	0.0345	0.0667
0.2	0.0714	0.1333
0.3	0.1111	0.2
0.4	0.1538	0.2667
0.5	0.2	0.3333
0.6	0.25	0.4
0.7	0.3043	0.4667
0.8	0.3636	0.5333
0.9	0.4286	0.5
1.0	0.5	0.6667

표 4 실제 영역의 넓이보다 예측 영역의 넓이가 2배 큰 경우

이미지 segmentation에서는 주로 Dice Coefficient를 사용하는 편인데, IoU는 비연속적인 값인 데 비해 Dice Coefficient는 미분할 수 있는 연속적인 값이어서 손실 함수로도 활용이 가능하기 때문이다. 1 - Dice Coefficient를 손실 함수로 정의하여 손실을 계산해 낼 수 있다.

3. 연구 내용

3.1. 전처리

인공지능 모델을 사용하여 용접 부위를 추출하기 전에 이미지 전처리를 활용해 용접 부위를 추출해 보고자 했다. 인공지능 모델이 아닌 전처리를 통해 추출한다면 훨씬 적은 자원으로 빠르게 용접 부위를 추출할 수 있고, 최종적인 목표인 용접 부위의 결함 탐지를 더 쉽게 시도해 볼 수 있다. 이하는 전 처리 과정이다. 우선 용접 부위를 최대한 포함하도록 이미지를 잘라냈다. 배경에 인식 번호가 있어서 이를 용접 부위로 인식할 수 있기에 이를 배제하고자 배경을 최대한 제거한 것이다.

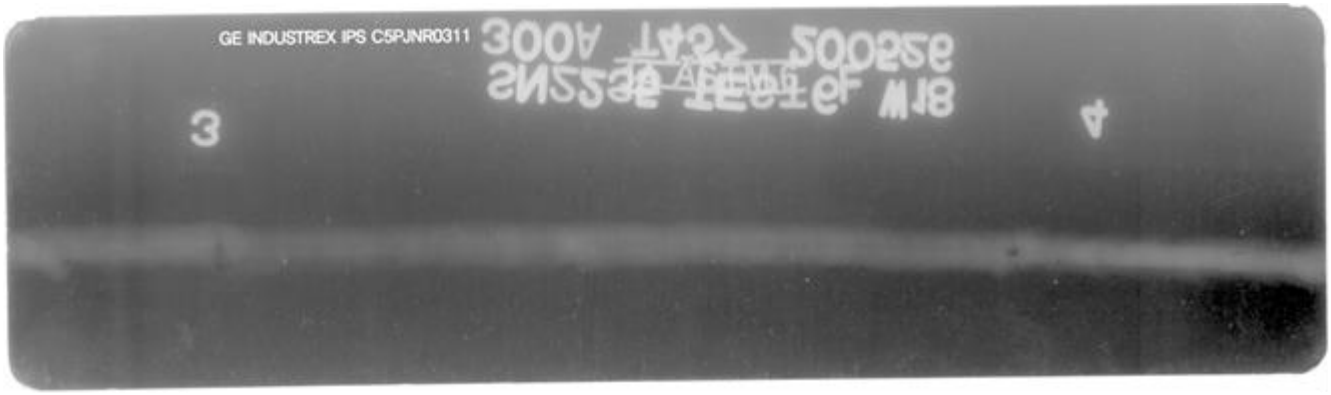


그림 6 전처리 전 원본 이미지

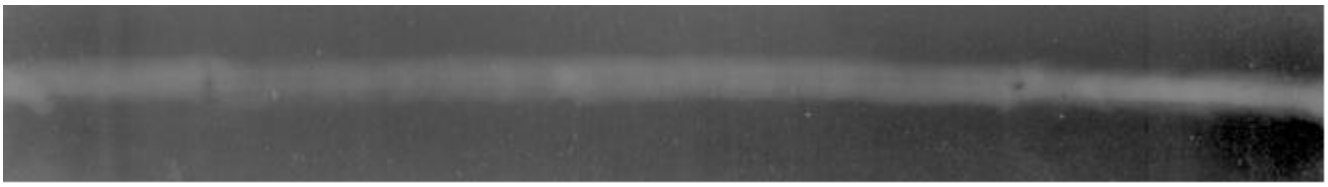


그림 7 잘라낸 이미지

이후 이미지를 grayscale 변환하여 **그림 8**을 얻었다. 이는 이미지를 0과 255까지의 단일 채널 값으로 표현하는 것을 의미한다. 제공된 필름 이미지는 이미 grayscale처럼 보이지만 실제로는 3채널 값을 포함하고 있다. 후에 threshold라는 전 처리 방법을 사용하기 위해서는 변환이 필수적이고, 단일 채널로 변환됨으로써 후에 딥러닝 모델에서 처리할 때 속도 향상도 꾀할 수 있다.

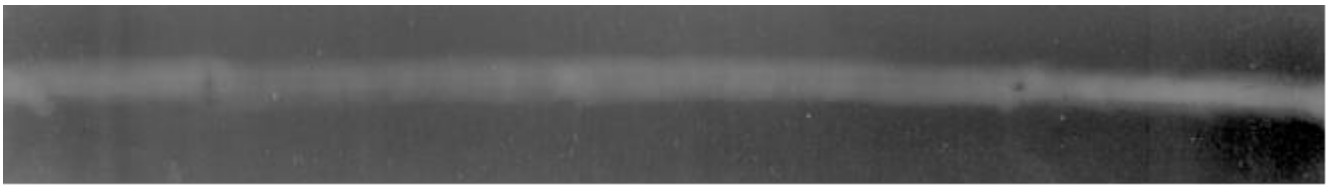


그림 8 grayscale화 한 이미지

이후 Threshold 변환을 통해 **그림 9**를 얻었다. Threshold 변환이란 임곗값을 기준으로 이하의 픽셀은 0으로, 이상의 픽셀은 255로 이미지를 이진화하는 것이다. 이를 통해 두 가지 값으로 구분하여 간단한 객체 분할이 가능하다. 임의의 임곗값 하나만 사용하면 이미지의 조명이 일정하지 않거나 배경색이 나뉘는 경우에 좋은 결과를 얻지 못하기 때문에, 임곗값은 이미지를 여러 영역으로 나눈 후에 그 주변 픽셀값을 기준으로 계산하는 adaptive Threshold를 사용하여 정하였다.

그림 9 Threshold 변환 후 이미지

이후 Eroding 연산을 수행하였다. Eroding 연산은 구조 요소(element)를 활용해 이웃한 픽셀을 최소 픽셀값으로 대체한다. 즉, 밝기가 255인 부분과 밝기가 0인 경계면에서 밝기가 0인 부분이 밝기가 255인 부분을 침식하게 된다. 따라서, 이 Eroding 연산을 적용하면 밝은 영역이 줄어들고 어두운 영역이 늘어난다. 이를 통해 이미지의 노이즈를 줄이고자 사용하였다. **그림 10**에서 Eroding 연산 후 배경에 있던 흰색 노이즈가 사라진 것을 볼 수 있다.

그림 10 Eroding 연산 후 이미지

이후 Dilation 연산을 수행했다. 구조 요소(element)를 활용해 이웃한 픽셀들을 최대 픽셀값으로 대체한다. 즉, 밝기가 255인 부분과 밝기가 0인 경계면에서 밝기가 255인 부분이 밝기가 0인 부분을 침식하게 된다. Dilation 연산을 적용하면 어두운 영역이 줄어들고 밝은 영역이 늘어난다. Eroding의 반대 연산이며 노이즈를 제거한 후 용접 부위의 외곽을 복구하고자 사용하였다. **그림 11**에서 Eroding을 적용한 후 줄어든 용접 부위가 다시 복구된 것을 확인할 수 있다.

그림 11 Dilation 연산 후 이미지

이후 Contours 함수를 이용해 용접 부위과 배경에 경계선을 그었다. Contours 함수는 일정한 값을 가지는 점들을 연결하여 곡선을 나타내는 함수이다. 이미지의 중앙에서부터 시작하여 지도의 등고선처럼 영역과 우선순위를 구분하여 처리하게 된다. 전처리할 원본 이미지들의 용접 부위가 중앙에서 시작되고, 용접 부위와 배경 부분을 구분하고자 사용하였다. 아래에서 최종 전 처리된 이미지 **그림 12**를 확인할 수 있다.

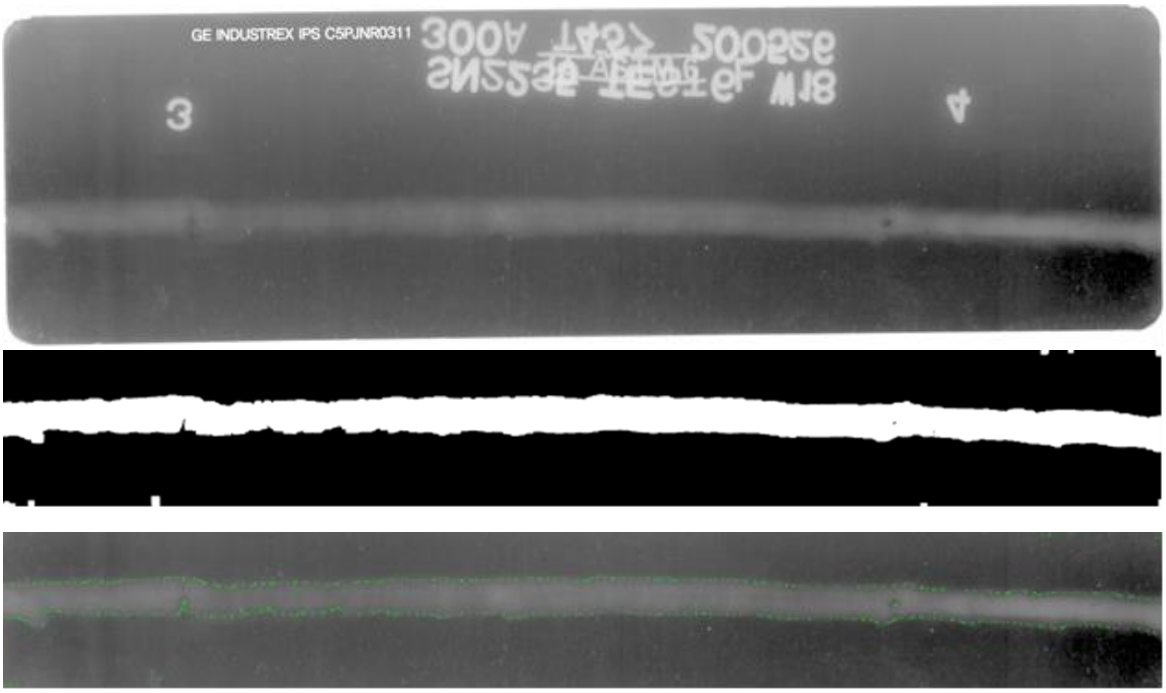


그림 12 Contour 함수를 이용해 최종 전처리한 이미지

하지만 전처리로는 용접 부위를 충분히 추출할 수 없었다. **그림 12**는 전처리가 잘 된 케이스를 보여준 것으로, 아래 **그림 13**은 전처리를 통해 용접 부위를 추출해 낼 수 없었던 이미지이다. 배경 이미지가 균일한 색이 아니고, 특정 부위는 용접 부위만큼 밝은 부분도 있어 Threshold 연산을 적용하면 배경 또한 용접 부위로 인식하게 된다. 이는 본 과제에서 기대하는 바가 아니고, 치명적인 오류를 발생시킬 수 있는 원인 중 하나이다.

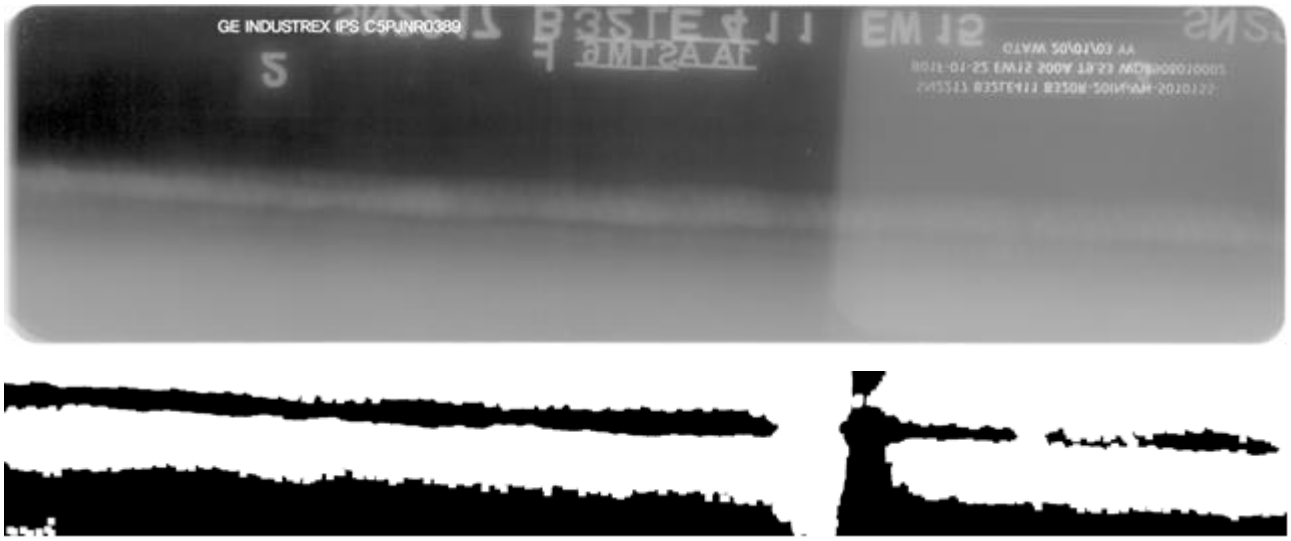


그림 13 전처리 실패 이미지

또한, 전처리를 수행하면서, 배경의 식별 번호 등을 제거하기 위해 임의로 이미지를 잘라내었는데, 이미지 내 용접 부위가 일정한 위치에 존재하지 않았다. 아래 **그림 14**에서 그 예시를 볼 수 있다. 비교하기 쉽게 대형관 RT 필름 두 장의 일부를 가로로 붙였다. 왼쪽 RT 필름 사진은 용접 부위가 상단에, 오른쪽 RT 필름 사진은 용접 부위가 하단에 위치한 것을 볼 수 있다.

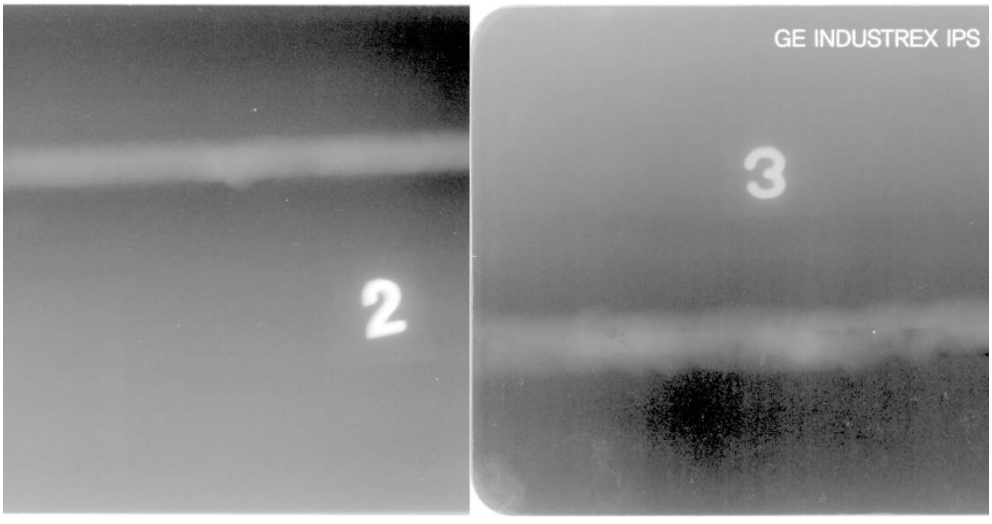


그림 14 서로 다른 이미지의 용접 부위 비교

또한, 이미지의 크기가 모두 달랐다. 어떤 대형관 이미지는 8693 X 2400 픽셀의 크기를, 4353 X 1187 픽셀의 크기를 가진다. 중형관 이미지도 크기가 제각각이었다. 어떤 중형관 이미지는 2182 X 1195, 다른 중형관 이미지는 4367 X 1289의 크기를 가졌다.

소형관의 이미지 크기는 거의 균일했다. 용접 부위를 정확히 추출하기 위해서는 이미지 크기, 용접 부위의 위치를 사람이 이미지마다 고려하여 자를 부분을 선정해야 한다. 이는 본래 목적인 검사원을 인공지능 모델로 대체한다는 목적에서 크게 벗어난 것으로, 본 과제의 목적과 맞지 않는다.

또한, 대형관은 배경에 파이프 외의 배경이 포함되어 있지 않아 어느정도 일정한 수치의 전처리를 해낼 수 있지만 중형관과 소형관은 그렇지 못했다. 29개의 대형관 전처리 결과의 Dice Coefficients의 평균은 0.7993으로 준수한 결과를 냈으나, 중형관과 소형관의 전처리 결과의 Dice Coefficients를 계산해 보면 0.1 이하의 값들을 출력해, 전처리를 사용해 용접 부위를 추출할 수 없다는 점을 보여준다.

아래 **그림 15**는 중형관 라벨링과 전처리를 보여준다. 맨눈으로도 좋지 못한 결과를 볼 수 있고, 실제 Dice Coefficients 또한 0.0108로 매우 좋지 못한 수치를 보인다.

그림 15 중형관 라벨링과 전처리를 통한 용접 부위 추출

따라서 전처리로만 용접 부위를 추출하는 방법은 좋은 결과를 내기 어렵기 때문에, 인공지능 모델을 사용하여 용접 부위를 추출하기로 했다.

3.2. UNet 모델

인공지능 모델은 앞서 배경에서 설명했듯이, UNet을 사용하기로 했다. UNet 모델을 사용하기 위해서는 우선 이미지 데이터를 가공하여 UNet에서 사용할 수 있는 데이터로 가공해야 한다. **코드 1**에서 그러한 과정을 수행한다.

```
resize_size = (512, 512)

## 이미지 변환 및 저장 함수
def resize_and_save_as_npy(input_dir, output_dir, image_filenames):

    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    for filename in image_filenames:
        input_path = os.path.join(input_dir, filename)

        image = cv2.imread(input_path, flags=cv2.IMREAD_UNCHANGED)

        new_img = cv2.resize(image, resize_size, interpolation=cv2.INTER_AREA)

        # .npy로 저장
        output_filename = 'input_' + os.path.splitext(filename)[0] + '.npy'
        output_path = os.path.join(output_dir, output_filename)
        np.save(output_path, new_img)

## 실행
resize_and_save_as_npy(test_image_dir, test_output_dir, test_image_filenames)
resize_and_save_as_npy(train_image_dir, train_output_dir, train_image_filenames)
```

코드 1 UNet 모델을 위한 데이터 가공

우선 이미지 데이터를 읽어온 후, 크기를 512 X 512로 조정한다. 그리고 NumPy 배열로 이미지 데이터를 변환 후 저장한다. 각 픽셀의 색상 정보 및 다른 이미지 속성이 NumPy 다차원 배열로 표현되어 저장되게 된다. NumPy 배열로 데이터를 저장하면 딥러닝 모델을 훈련하거나 다른 이미지 처리 작업에 쉽게 활용할 수 있다. 이미지의 각 픽셀에 대한 정보, 채널 값 등을 쉽게 조작하고 분석할 수 있기 때문이다.

이후 **코드 2**에서 볼 수 있듯이, 미리 라벨링 한 데이터를 불러온다. 라벨링 한 데이터를 마스크 형식의 이미지로 변환한다. 크기를 조절한 이미지 데이터에 맞추어 마스크 크기도 512 X 512로 조절한다.

```
# 마스크 생성
mask = Image.new('L', image.size, 0)  # 마스크 검정색으로
draw = ImageDraw.Draw(mask)
for region in regions:
    shape_attributes = region['shape_attributes']
    points_x = shape_attributes['all_points_x']
    points_y = shape_attributes['all_points_y']
    polygon_points = list(zip(points_x, points_y))
    draw.polygon(polygon_points, fill=255)  # 라벨링 영역만 흰색으로

# 마스킹 이미지 생성
masked_image = Image.new('RGB', image.size)
masked_image.paste(image, mask=mask)

# 색 반전
inverted_masked_image = Image.new('RGB', masked_image.size)
for y in range(masked_image.height):
    for x in range(masked_image.width):
        r, g, b = masked_image.getpixel((x, y))
        inverted_color = (255 - r, 255 - g, 255 - b)
        inverted_masked_image.putpixel((x, y), inverted_color)

# 리사이징
resized_image = inverted_masked_image.resize(resize_size, Image.LANCZOS)
```

코드 2 라벨링 데이터 마스크 생성

라벨링은 VCG image annotate라는 온라인 라벨링 툴을 사용했다. **그림 16**에서 라벨링 프로젝트를 확인 할 수 있다. **삼성중공업 측에서 용접부 상•하단 2~3mm 정도까지도 포함해서 검사한다고 하여 사각형 모양으로 라벨링을 진행**하였으며, 대, 중형관은 잘라서 라벨링을 진행했다. 용접 부위가 기울어진 모양이면 사각형 내에 용접 부위가 아닌 부분이 많이 포함되어 정보가 오염될 수 있기 때문이다. 가로 1000픽셀 단위로 잘라서 이미지를 라벨링 했는데, 이는 최대한 이미지를 정사각형 모양에 가깝도록 만들어 모델에 데이터를 넣을 때 왜곡을 줄이기 위해서이다.

그림 16 VCG image annotate를 사용한 데이터 라벨링

이후 사용할 UNet 모델을 정의한다. UNet 모델에는 데이터 증강을 위한 이미지 회전 및 반전 함수가 포함되어 있다. 이를 통해 부족한 데이터를 보완하고 성능을 개선할 수 있다. UNet은 크게 Contracting path와 Expanding Path로 구성된다.

아래 **코드 3**에서는 UNet의 Contracting path를 볼 수 있다. 총 5번의 Convolution(합성곱), Batch Normalization(정규화), ReLU 활성화 함수를 거친다. 이 과정을 통해 context을 추출하며, 특징 맵을 Skip connection을 통해 Expanding Path로 보낸다. 이를 통해 Convolution을 하면서 생기는 border pixel pixel에 대한 정보 손실을 복구할 수 있다. 이 정보를 가지고 Expanding Path에서는 특징 맵과 위치정보를 결합하여 각 Pixel마다 어떤 객체에 속하는지 구분한다. Contracting path에서 총 5번의 단계를 거치며, 단계를 넘어갈 때마다 2x2 max pooling 연산을 수행하여 특징 맵의 크기가 1/4로 줄어들며, 채널의 수가 2배로 늘어난다.

```
# Contracting path
self.enc1_1 = CBR2d(in_channels=1, out_channels=64)          # onvolution(합성곱),Batch Normalization(정규화), ReLU 활성화 함수
self.enc1_2 = CBR2d(in_channels=64, out_channels=64)

self.pool1 = nn.MaxPool2d(kernel_size=2)                     # Max Pooling

self.enc2_1 = CBR2d(in_channels=64, out_channels=128)
self.enc2_2 = CBR2d(in_channels=128, out_channels=128)

self.pool2 = nn.MaxPool2d(kernel_size=2)

self.enc3_1 = CBR2d(in_channels=128, out_channels=256)
self.enc3_2 = CBR2d(in_channels=256, out_channels=256)

self.pool3 = nn.MaxPool2d(kernel_size=2)

self.enc4_1 = CBR2d(in_channels=256, out_channels=512)
self.enc4_2 = CBR2d(in_channels=512, out_channels=512)

self.pool4 = nn.MaxPool2d(kernel_size=2)

self.enc5_1 = CBR2d(in_channels=512, out_channels=1024)
```

코드 3 UNet의 Contracting path

코드 4에서는 Expansive path를 볼 수 있다. 이 경로에서는 Contracting path에서 줄어든 이미지를 2x2 up-convolution를 사용하여 크기를 다시 확장한다. 이때, Contracting path에서 추출한 고차원 context와 스킵 연결로 전달된 저차원 정보를 바탕으로 UNet 모델은 최종 예측을 수행한다.

Contracting path에서 스킵 연결로 얻은 저차원정보는 손실된 특징 맵들을 보충하며, UNet은 더욱 정확한 예측을 하게 된다. 코드 5에서는 스킵 연결을 확인할 수 있는데 PyTorch 라이브러리의 torch.cat 함수를 사용해 텐서 데이터를 전달하게 구현되어 있다.

```
# 확장 경로(Expansive path)
self.dec5_1 = CBR2d(in_channels=1024, out_channels=512)

self.unpool4 = nn.ConvTranspose2d(in_channels=512, out_channels=512,
                                  kernel_size=2, stride=2, padding=0, bias=True)

self.dec4_2 = CBR2d(in_channels=2 * 512, out_channels=512)
self.dec4_1 = CBR2d(in_channels=512, out_channels=256)

self.unpool3 = nn.ConvTranspose2d(in_channels=256, out_channels=256,
                                  kernel_size=2, stride=2, padding=0, bias=True)

self.dec3_2 = CBR2d(in_channels=2 * 256, out_channels=256)
self.dec3_1 = CBR2d(in_channels=256, out_channels=128)

self.unpool2 = nn.ConvTranspose2d(in_channels=128, out_channels=128,
                                  kernel_size=2, stride=2, padding=0, bias=True)

self.dec2_2 = CBR2d(in_channels=2 * 128, out_channels=128)
self.dec2_1 = CBR2d(in_channels=128, out_channels=64)

self.unpool1 = nn.ConvTranspose2d(in_channels=64, out_channels=64,
                                  kernel_size=2, stride=2, padding=0, bias=True)

self.dec1_2 = CBR2d(in_channels=2 * 64, out_channels=64)
self.dec1_1 = CBR2d(in_channels=64, out_channels=64)

self.fc = nn.Conv2d(in_channels=64, out_channels=1, kernel_size=1, stride=1, padding=0, bias=True)
```

코드 4 Expansive path

```
unpool4 = self.unpool4(dec5_1)
cat4 = torch.cat((unpool4, enc4_2), dim=1)
dec4_2 = self.dec4_2(cat4)
dec4_1 = self.dec4_1(dec4_2)
```

코드 5 스킵 연결

이후 epoch, 전체 훈련 데이터 세트를 한 번 순회할지, batch size, 한 번의 모델 가중치 업데이트를 위해 사용되는 데이터 샘플 수, 등의 훈련에 필요한 수치를 정하고 UNet 모델을 실제로 학습시키게 된다.

```
TRAIN: EPOCH 1000 / 1000 | BATCH 0001 / 0019 | LOSS 0.0064
TRAIN: EPOCH 1000 / 1000 | BATCH 0002 / 0019 | LOSS 0.0059
TRAIN: EPOCH 1000 / 1000 | BATCH 0003 / 0019 | LOSS 0.0055
TRAIN: EPOCH 1000 / 1000 | BATCH 0004 / 0019 | LOSS 0.0060
TRAIN: EPOCH 1000 / 1000 | BATCH 0005 / 0019 | LOSS 0.0059
TRAIN: EPOCH 1000 / 1000 | BATCH 0006 / 0019 | LOSS 0.0057
TRAIN: EPOCH 1000 / 1000 | BATCH 0007 / 0019 | LOSS 0.0058
TRAIN: EPOCH 1000 / 1000 | BATCH 0008 / 0019 | LOSS 0.0058
TRAIN: EPOCH 1000 / 1000 | BATCH 0009 / 0019 | LOSS 0.0060
TRAIN: EPOCH 1000 / 1000 | BATCH 0010 / 0019 | LOSS 0.0060
TRAIN: EPOCH 1000 / 1000 | BATCH 0011 / 0019 | LOSS 0.0059
TRAIN: EPOCH 1000 / 1000 | BATCH 0012 / 0019 | LOSS 0.0060
TRAIN: EPOCH 1000 / 1000 | BATCH 0013 / 0019 | LOSS 0.0061
TRAIN: EPOCH 1000 / 1000 | BATCH 0014 / 0019 | LOSS 0.0060
TRAIN: EPOCH 1000 / 1000 | BATCH 0015 / 0019 | LOSS 0.0059
TRAIN: EPOCH 1000 / 1000 | BATCH 0016 / 0019 | LOSS 0.0059
TRAIN: EPOCH 1000 / 1000 | BATCH 0017 / 0019 | LOSS 0.0059
TRAIN: EPOCH 1000 / 1000 | BATCH 0018 / 0019 | LOSS 0.0059
TRAIN: EPOCH 1000 / 1000 | BATCH 0019 / 0019 | LOSS 0.0060
VALID: EPOCH 1000 / 1000 | BATCH 0001 / 0004 | LOSS 0.0066
VALID: EPOCH 1000 / 1000 | BATCH 0002 / 0004 | LOSS 0.0059
VALID: EPOCH 1000 / 1000 | BATCH 0003 / 0004 | LOSS 0.0067
VALID: EPOCH 1000 / 1000 | BATCH 0004 / 0004 | LOSS 0.0066
```

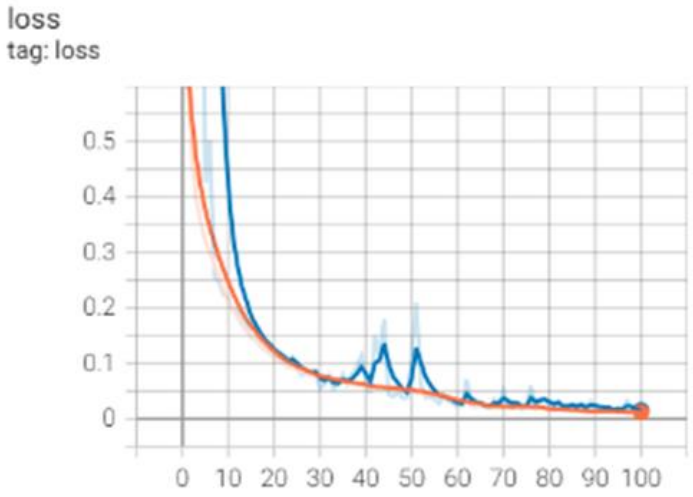


그림 17 중형관 batch size 4, epoch 1000 학습의 loss률

3.3. 결함 판독 모델

결함 판독을 위해 우선 결함 마스킹 이미지가 필요했다. 원래 Cut-Paste 방식은 정상 이미지의 일부를 잘라낸 후 이미지의 픽셀 등을 변화시키고, 회전, 변형 등을 통해 마스킹 이미지를 생성한다. 이후 정상 이미지의 임의의 위치에 마스킹 이미지를 붙여 결함 이미지를 생성한다. 다만, 본 연구에서 이미 결함 이미지를 가지고 있었기 때문에, 결함 이미

지에서 결함 부분을 잘라내어 마스킹 이미지로 활용했다. 이를 통해 좀 더 정확한 결함 데이터를 가지고 학습을 시도할 수 있었다. **코드 6**에서 마스킹 이미지를 가지고 결함 이미지를 만드는 모습을 볼 수 있다.

```
def __call__(self, img):
    #TODO: we might want to use the pytorch implementation to calculate the patches
    # from https://pytorch.org/vision/stable/_modules/torchvision/transforms/transforms.html#RandomErasing
    h = img.size[0]
    w = img.size[1]

    # ratio between area_ratio[0] and area_ratio[1]
    ratio_area = random.uniform(self.area_ratio[0], self.area_ratio[1]) * w * h

    # sample in log space
    log_ratio = torch.log(torch.tensor((self.aspect_ratio, 1/self.aspect_ratio)))
    aspect = torch.exp(
        torch.empty(1).uniform_(log_ratio[0], log_ratio[1])
    ).item()

    cut_w = int(round(math.sqrt(ratio_area * aspect)))
    cut_h = int(round(math.sqrt(ratio_area / aspect)))

    # one might also want to sample from other images. currently we only sample from the image itself
    from_location_h = int(random.uniform(0, h - cut_h))
    from_location_w = int(random.uniform(0, w - cut_w))

    box = [from_location_w, from_location_h, from_location_w + cut_w, from_location_h + cut_h]
    patch = img.crop(box)

    if self.colorJitter:
        patch = self.colorJitter(patch)

    to_location_h = int(random.uniform(0, h - cut_h))
    to_location_w = int(random.uniform(0, w - cut_w))

    insert_box = [to_location_w, to_location_h, to_location_w + cut_w, to_location_h + cut_h]
    augmented = img.copy()
    augmented.paste(patch, insert_box)

    return super().__call__(img, augmented)
```

코드 6 Cut-Paste 방법

이후, ResNet 모델을 활용하여, 정상 이미지는 Index 0에 이상 이미지는 Index 1에 위치하도록 모델을 훈련시킨다. 코드 7에서 사용한 ResNet18 모델을 확인할 수 있다.

```python
class ProjectionNet(nn.Module):
    def __init__(self, pretrained=True, head_layers=[512,512,512,512,512,512,512,512,128], num_classes=2):
        super(ProjectionNet, self).__init__()
        self.resnet18 = resnet18(pretrained=pretrained)
        last_layer = 512
        sequential_layers = []
        for num_neurons in head_layers:
            sequential_layers.append(nn.Linear(last_layer, num_neurons))
            sequential_layers.append(nn.BatchNorm1d(num_neurons))
            sequential_layers.append(nn.ReLU(inplace=True))
            last_layer = num_neurons

        head = nn.Sequential(
            *sequential_layers
          )
        self.resnet18.fc = nn.Identity()
        self.head = head
        self.out = nn.Linear(last_layer, num_classes)

    def forward(self, x):
        embeds = self.resnet18(x)
        tmp = self.head(embeds)
        logits = self.out(tmp)
        return embeds, logits
```

코드 7 ResNet18 모델

3.4. 서버 구현

Flask 웹 애플리케이션을 사용하여 이미지 처리와 데이터베이스 상호작용을 수행하는 기능을 제공한다. 촬영 담당자와 검사 담당자가 다를 것을 예상하고 검사 담당자는 이미 서버에 업로드 된 이미지를 불러와서 작업을 하도록 제작하였다.

촬영 담당자가 이미지를 서버에 업로드하면 이미지는 가로 1000 픽셀 단위로 분할하여 저장된다. 이후 용접 부위 추출 모델의 예측을 통해 용접 부위를 추출한다. 추출한 용접 부위를 전 처리하여, 정사각형 형태로 만들고, 결함 탐지 모델을 활용하여 용접 부위에 결함이 있는지 탐지하고 heat map 을 보여준다

아래 **그림 18** 에서 실제로 웹페이지를 통해 사용자에게 모델이 예측한 이미지를 보여주는 화면을 확인할 수 있다. 이미지 별로 다른 모델을 사용하기 때문에 각각 분리되어 있는 것을 확인할 수 있다.

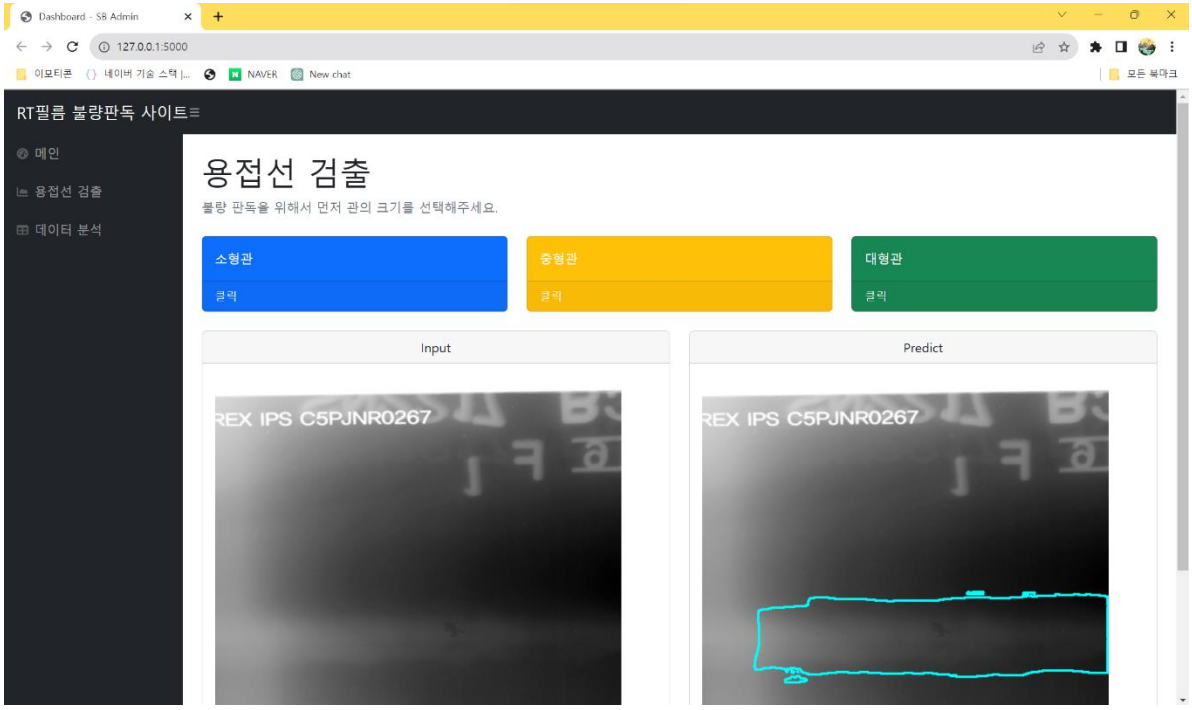


그림 18 용접 부위 추출 사이트

4. 연구 결과 분석 및 평가

본 연구의 1차 목표는 인공지능 모델 UNet을 활용하여 용접 부위를 정확하게 감지하여 추출하는 것이다. 용접 부위의 정확한 식별을 통해, 최종 목표인 결함 감지를 위한 데이터 세트를 만들어 내기 위함이다. **각 epoch 값과 batch size 값을 대형관 4, 100. 중형관 4, 1000. 소형관 4, 1000으로 설정해 UNet 모델을 훈련**했다. 아래 **그림 19**에서 실제로 모델이 예측한 데이터들을 확인할 수 있다.

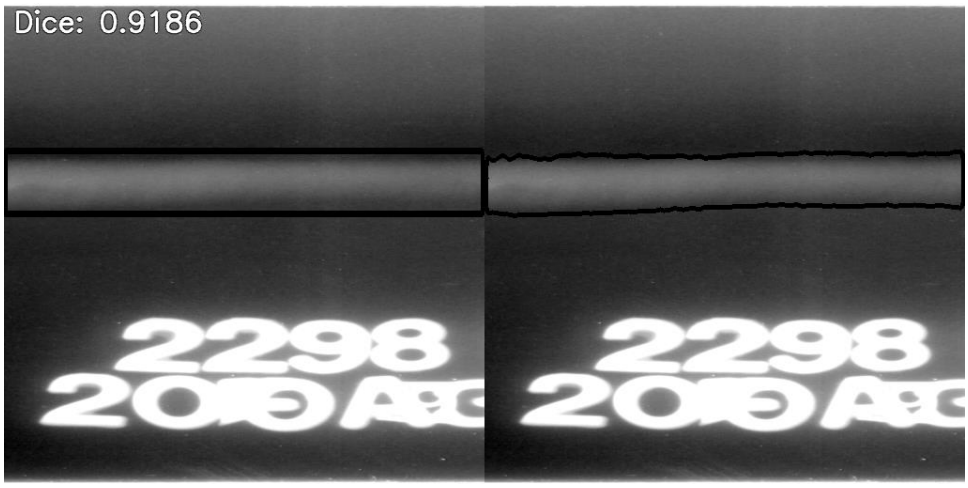


그림 19 대형관 라벨링과 예측 결과 비교 및 Dice coefficient

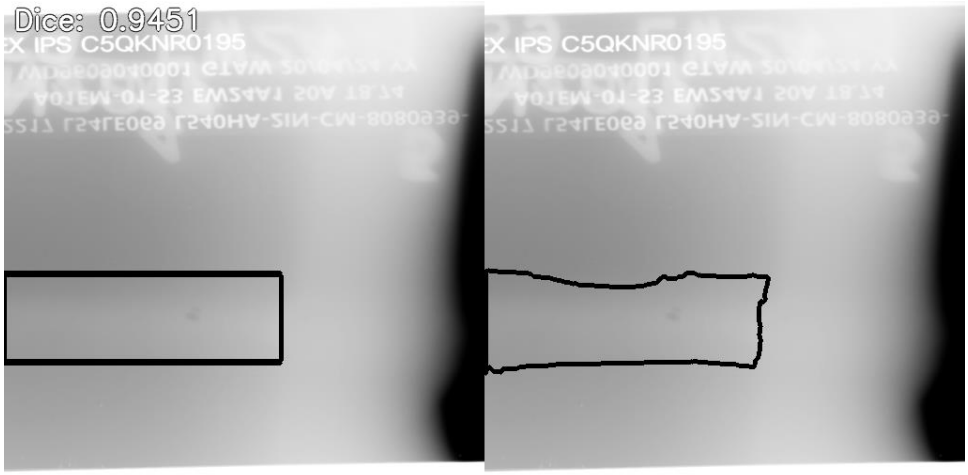


그림 20 중형관 라벨링과 예측 결과 비교 및 Dice coefficient

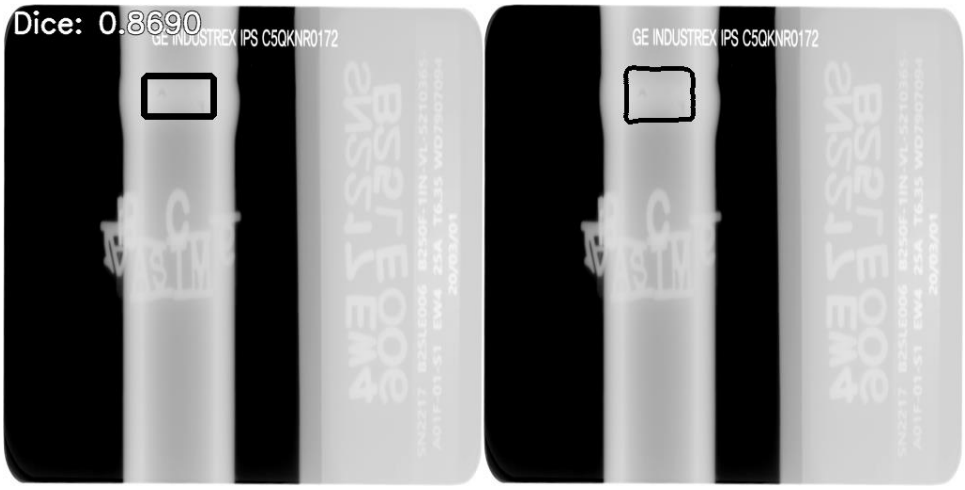
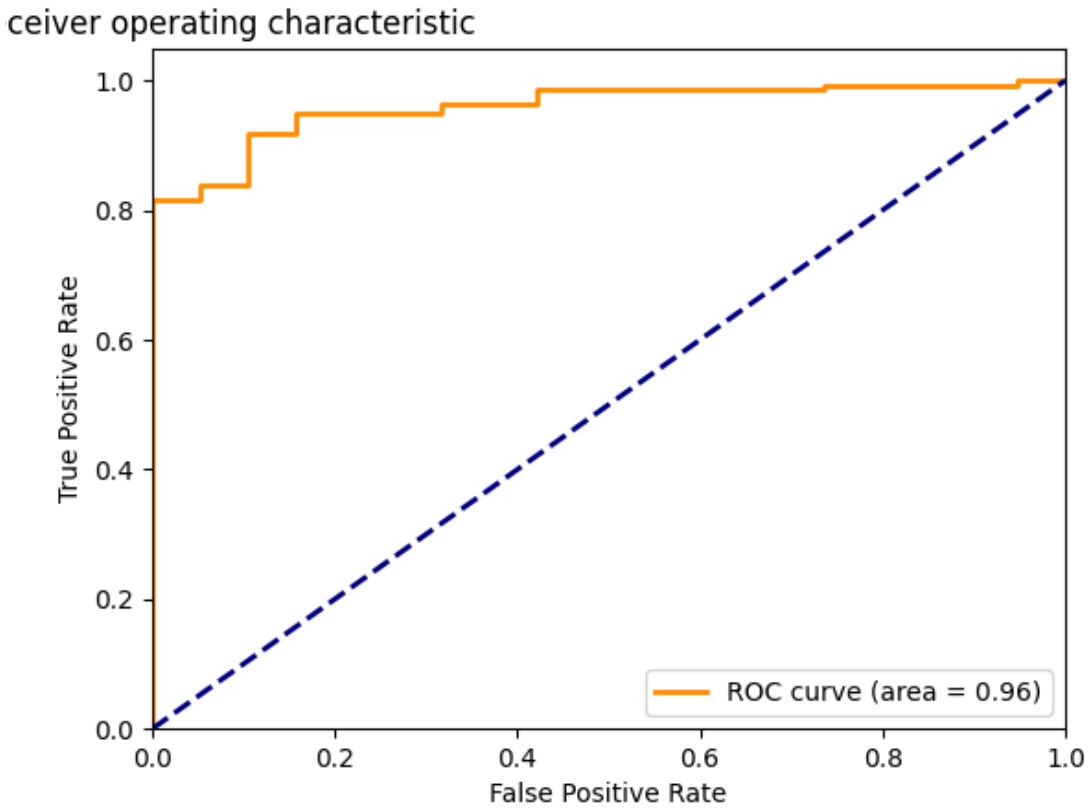


그림 21 소형관 라벨링과 예측 결과 비교 및 Dice coefficient

모델 학습의 loss률은 평균 0.006, Dice coeffcient는 평균 0.83 정도이다. Dice coefficinet는 0.7 이상이면 어느 정도 유사하다고 정의할 수 있고, 1이면 완전히 동일하다. 이를 통해 UNet 모델이 예측해 낸 데이터를 보면 높은 정확성과 신뢰성을 보인다.

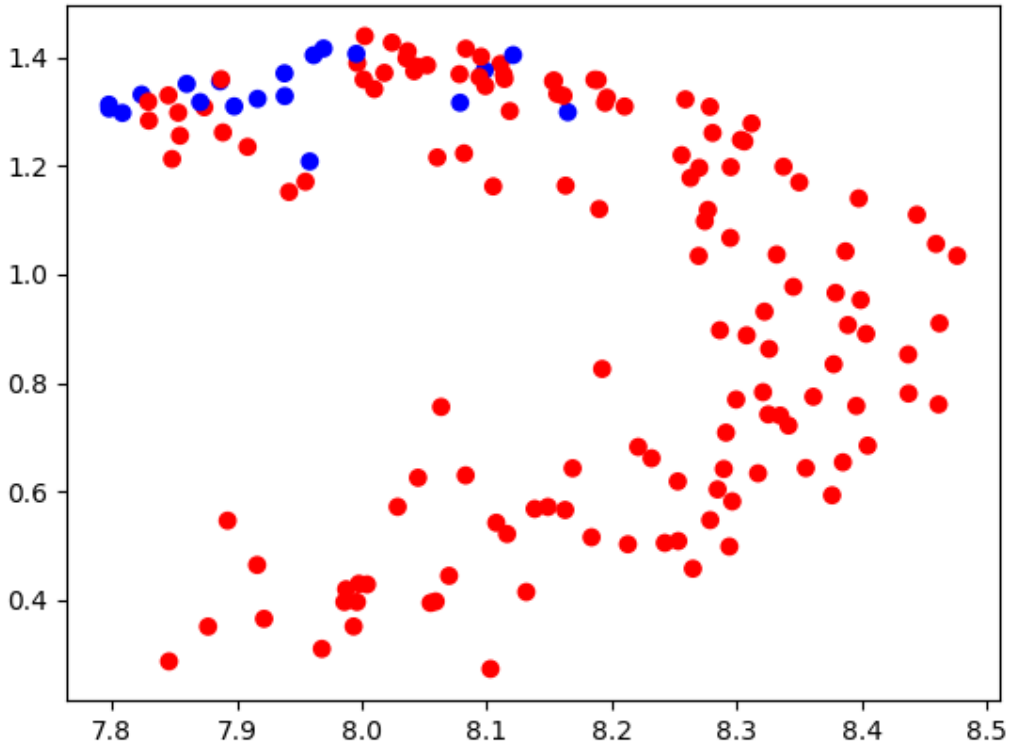
이러한 결과는 UNet을 활용하여 용접 부위를 정확하게 감지하여 추출하는 1차 목표를 어느 정도 성공적으로 달성하였음을 나타낸다. 이는 용접 부위의 이미지를 결함 탐지 인공지능 모델에 제공하여 모델이 정상 이미지의 특징과 구조를 파악하도록 도울 것이다. 이후 **더 많은 데이터를 수집하고 모델의 세부 파라미터를 조절 및 모델을 더 많이 학습한다면, 더 나은 성과를 얻을 수 있을 것으로 기대**된다. 또한, 실제 환경에서의 적용 가능성과 확장 가능성에 대한 추가적인 연구가 필요하다.

ResNet 을 이용해 Cut-Paste 방식으로 용접 부위의 결함 판독한 결과도 좋은 결과를 보였다. Epoch 은 512, batch size 는 64 이다. 아래 그래프에서 ROC curve 를 확인할 수 있다.



그래프 1 ROC curve

정상 이미지 판독률인 ROC curve 가 0.96 으로 1 에 근접한 좋은 값을 보여준다. 모델이 정상 이미지로 예측한 이미지가 96% 정상 이미지라는 것이다. 따라서 모델이 이미지를 정확히 판독하고 있다고 볼 수 있다. 아래 **그래프 2** 에서는 테스트 이미지의 분포를 확인할 수 있다. 정상 이미지와 결함 이미지의 분포를 확인할 수 있다. 결함 이미지는 빨간색, 정상 이미지는 파란색 점으로 맵핑 되어있다. 훈련된 모델은 학습된 정상 이미지와의 거리를 통해 이미지에 결함이 있는지 판독한다.



그래프 2 특징 맵 에서의 결함 이미지와 정상 이미지의 분포

5. 결론 및 향후 연구 방향

이 연구는 RT 필름을 분석하는 인공지능 모델을 개발하고, 용접 부위를 추출하는 과제를 다루었다. 인공지능 모델, UNet을 활용하여 용접 부위를 높은 정확도로 추출할 수 있었다. 세그멘테이션 평가 척도인 Dice coefficient를 사용해 평가한 결과 **대형관 평균 0.798 중형관 평균 0.879, 소형관 평균 0.815으로 학습 수에 비해 괜찮은 정확도**를 보여주었다. 이를 통해 이후 있을 연구인 인공지능 비전 기반 RT 필름 결함 자동 판독 기술 연구에서 사용할 데이터 세트를 만들어낼 수 있었다. 이후 이를 가지고 비지도 학습 형태로 결함 탐지 모델을 설계하고 만들어 냈다. 이 결함 탐지 모델은 **ROC curve가 0.96**로 준수한 정확도를 보여주었다

이 두 모델을 바탕으로 결함을 탐지할 수 있는 서버를 구축하였다. 이 서버는 조선소에서 RT 필름을 영사하여 서버에 이미지를 올리면, 용접 부위를 추출해 내고, 용접 부위에 결함이 있는지 판독해 낸다. 이 서버를 사용한다면, 이 연구 서두에 언급한 인간 검

사원으로 발생하는 문제 해결을 기대해 볼 수 있다. 그러한 문제에는 검사원의 주관적 판단의 개입, 실수로 인한 오류, 미세한 결함이나 복잡한 패턴 감지의 어려움, 많은 시간과 인적 자원 소모 등이 있다. 이를 인공지능 모델로 대체한다면 빠른 속도로 정확하게, 적은 비용을 사용하여 RT 필름을 판독하여 용접 부위의 결함을 파악하고 조치를 취할 것을 기대할 수 있다.

따라서 앞으로의 연구 방향은 이러하다. Dice coefficient 대형관 평균 0.798, 중형관 평균 0.879, 소형관 평균 0.815으로 괜찮은 정확도를 보였지만, **데이터 부족으로 인한 한계가 보이므로, 데이터를 더 수집하여 더욱 다양한 이미지를 학습시켜 모델 능력을 향상한다. 또한 UNet에 대한 연구를 지속해 모델을 개선하여 정확도를 더욱 향상한다. 추가로 부족했던 모델 학습 수를 보강하여 학습한다. 이렇게 개선된 모델을 통해 만들어진 데이터 세트는 높은 정확성으로 용접 부위 결함 판독을 더 정확하게 해 줄 것이다.**

또한, 용접 부위 결함 탐지 모델 또한 개선이 필요해 보인다. **ROC curve가 0.96**로 준수한 정확도를 보여주지만, 결함 탐지는 1에 정말 가까운 값을 보여주어야 사고를 방지할 수 있다. 따라서 데이터 셋을 보강하고 모델을 개선하여, RT 필름에서 자동으로 더 정확하게 판독해 줄 수 있게 인공지능 모델을 개선하여야 할 것이다. 이는 용접 부위의 품질을 향상하는 데 중요한 역할을 할 것을 기대해 볼 수 있다. 결함 부분을 신속하게 감지하고 조치함으로써 제품의 품질 향상을 꾀할 수 있기 때문이다. 이 모델은 사람이 탐지하기 어려운 미세한 결함이나 복잡한 결함 또한 탐지해 내 결함 탐지 정확성을 향상할 수 있다. 빠른 속도의 탐지를 제공하며, 비용을 절감할 수 있다. 따라서, 본 연구를 이어서 인공지능 비전 기반 RT 필름 결함 자동 판독 기술을 위한 인공지능 모델 개선은 매우 중요한 과업이다. 따라서 더 많은 연구와 학습을 통해 RT필름 결함 탐지 모델을 이른 시일 내에 개선하는 것을 목표로 해야 한다.

6. 참고 문헌

[1] 김형준, (2019, April, 06), Python and OpenCV - 19: Isolation lines for images (Contours) - 5/5, Available: http://www.gisdeveloper.co.kr/?p=6617. (in Korean)

[2] Wikipedia, The Free Encyclopedia, (2023, September, 22), Erosion (morphology), Available: https://en.wikipedia.org/wiki/Erosion_(morphology).

[3] Wikipedia, The Free Encyclopedia, (2021, October, 22), Dilation (morphology), Available: https://en.wikipedia.org/wiki/Dilation_(morphology).

[3] Angelo Monteux, (2019, May, 10), Metrics for semantic segmentation, Available: https://ilmonteux.github.io/2019/05/10/segmentation-metrics.html.

[4] Prateek Chhikara, (2022, Mar, 30), Understanding Morphological Image Processing and Its Operations, Available: https://towardsdatascience.com/understanding-morphological-image-processing-and-its-operations-7bcf1ed11756.

[5] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI) (pp. 234-241)

[6] PyTorch, Available: https://pytorch.org/.

[7] Samsung Heavy Industries Co., Ltd Shipbuilding and Marine Research Institute, (2023, January, 13), Auto-reading RT faults based on deep learning. (in Korean)

[8] Lieberk, (2022, Aug, 29), github, Available: https://github.com/Lieberk/Paddle-Cutpaste/blob/main/model.py

[9] ffighting, Anomaly Detection,Available:https://ffighting.net/deep-learning-paper-review/anomaly-detection/anomaly-detection/