WM-811K 웨이퍼 맵 불량 패턴 분석 및 CNN 기반 분류

1. 프로젝트 개요

1.1 목적

반도체 웨이퍼 맵 이미지(비정형 데이터)를 분석해 불량 패턴의 분포와 특징을 파악하고, CNN 모델로 불량 유형을 자동 분류한 뒤, 그 결과를 바탕으로 품질관리 개선안을 도출한다.

1.2 데이터 및 환경

항목	내용
데이터	Kaggle WM-811K 웨이퍼 맵 데이터셋 (LSWMD.pkl, 웨이퍼 811,457 장). 실제 반도체 공장에서 수집된 웨이퍼 맵
주요 컬럼	waferMap(2 차원 배열: 0 배경, 1 정상 다이, 2 불량 다이), lotName(로트 ID), waferIndex(로트 내 순번), failureType(불량 유형 라벨)
불량 유형	Center, Donut, Edge-Loc, Edge-Ring, Loc, Random, Scratch, Near-full 8 종 + none(정상)
분석 환경	Kaggle Notebook (GPU T4), Python, pandas, OpenCV, TensorFlow/Keras

1.3 진행 단계

① 데이터 구조 확인 → ② 웨이퍼 이미지 샘플 확인 → ③ 불량 유형별 분포 분석 → ④ 대표 불량 패턴 시각화 → ⑤ CNN 불량 유형 분류 → ⑥ 품질관리 개선안 도출

2. 데이터 이해 및 전처리

2.1 라벨 구성

구분	웨이퍼 수	비율(%)
라벨 없음 (판정하지 않음)	638,507	78.69
none (정상)	147,431	18.17
불량 8 종	25,519	3.14
합계	**811,457**	**100.00**

- 전체의 78.7%는 전문가 판정이 없는 웨이퍼다. 실제 현장에서도 모든 웨이퍼를 사람이 판정하지 않기 때문으로 해석되며, 분석에는 라벨이 있는 172,950 장만 사용했다.
- none 은 '불량 다이가 없음'이 아니라 '특정 불량 패턴이 없음'을 뜻한다. 정상 웨이퍼에도 웨이퍼당 약 10%의 불량 다이가 흩어져 있다.

2.2 웨이퍼 맵 크기와 리사이즈

- 웨이퍼 맵 크기가 346 가지(세로 15~212, 가로 3~204)로 제각각이며, 대부분 25~45 사이에 몰려 있다. CNN 입력을 위해 모두 64×64 로 통일했다.
- 맵의 값 0·1·2 는 밝기가 아니라 범주 번호이므로, 이웃 값을 평균 내는 선형 보간 대신 가장 가까운 칸의 값을 복사하는 최근접(nearest) 방식으로 리사이즈했다.

비교 (웨이퍼 1 장, 64×64 변환)	기준	nearest	linear
값의 종류	0, 1, 2	0, 1, 2	358 가지
불량(값 2) 칸 수	약 167 칸	172 칸	1 칸

선형 보간을 쓰면 불량 값 2 가 주변 정상 값과 섞여 1.x 로 바뀌면서 불량이 거의 사라진다. 실험으로 최근접 방식의 필요성을 확인했다.

3. 탐색적 분석

3.1 불량 유형별 분포

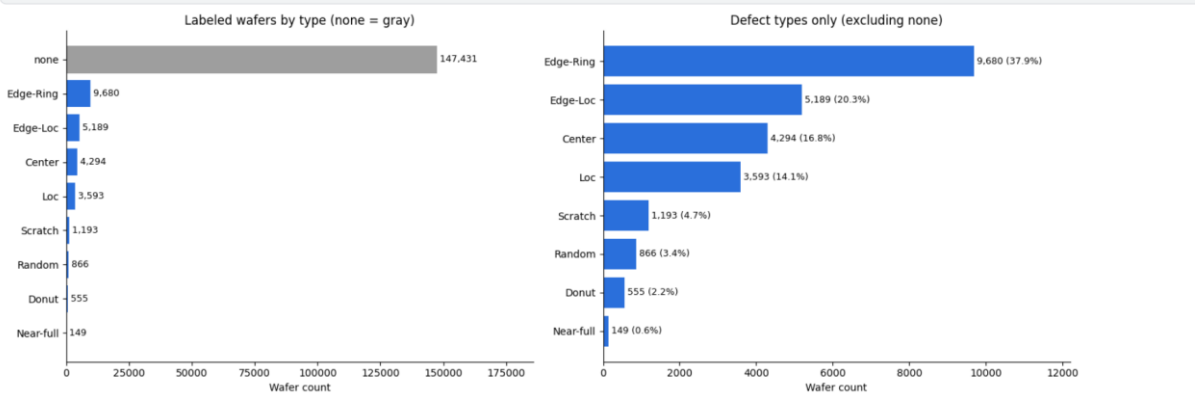


그림 1. 라벨 있는 웨이퍼의 유형별 개수(왼쪽)와 불량 8 종의 개수(오른쪽)

- **심한 클래스 불균형:** none 이 라벨 있는 웨이퍼의 85.2%다. 모든 웨이퍼를 none 으로 예측해도 정확도가 85%이므로, 정확도 대신 유형별 재현율과 macro F1 로 평가해야 한다.
- **불량끼리도 불균형:** 가장 많은 Edge-Ring(9,680 장)과 가장 적은 Near-full(149 장)은 65 배 차이가 난다.

- **가장자리 불량이 과반:** Edge-Ring(37.9%)과 Edge-Loc(20.3%)을 합치면 불량의 58.3%로, 불량의 절반 이상이 웨이퍼 가장자리에서 발생한다.

3.2 생산 관점 분석

- **로트 집중:** 10,762 개 로트의 불량 웨이퍼 비율이 0%(25% 지점) 또는 100%(75% 지점)로 양극화됐다. lot10183~10185 처럼 번호가 연속된 로트가 모두 불량인 사례도 있어, 불량이 특정 로트·기간에 집중되는 경향을 보인다. 단, 라벨이 로트 안에서 무작위로 붙지 않았을 수 있어 실제 공정 불량률로 해석하지는 않는다.
- **로트 내 순번:** 웨이퍼 순번(1~25)별 불량 비율은 13.9~15.5%로 차이가 1.5%p 에 그쳐, 로트 안 위치의 영향은 뚜렷하지 않았다.

3.3 유형별 대표 패턴

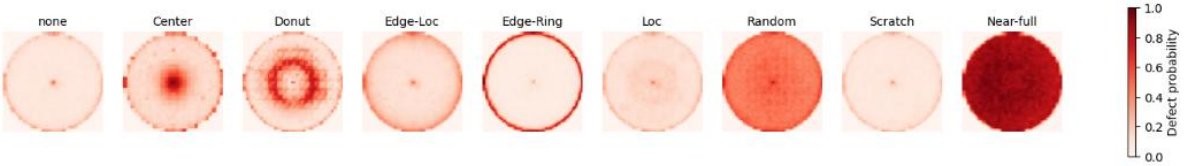


그림 2. 유형별 평균 불량 위치 (유형당 최대 1,000 장, 64×64, 진할수록 불량 확률이 높음)

- Center(중앙 덩어리), Donut(중간 반지름의 고리), Edge-Ring(가장자리 테두리), Near-full(거의 전체)은 불량 위치가 고정돼 평균 지도에서 모양이 뚜렷하다.
- Edge-Loc, Loc, Scratch 는 불량이 생기는 위치가 웨이퍼마다 달라 평균 지도가 흐릿하다. 이 유형들은 위치보다 모양으로 구분해야 하므로, 위치가 바뀌어도 모양을 잡아내는 CNN 이 적합하다.

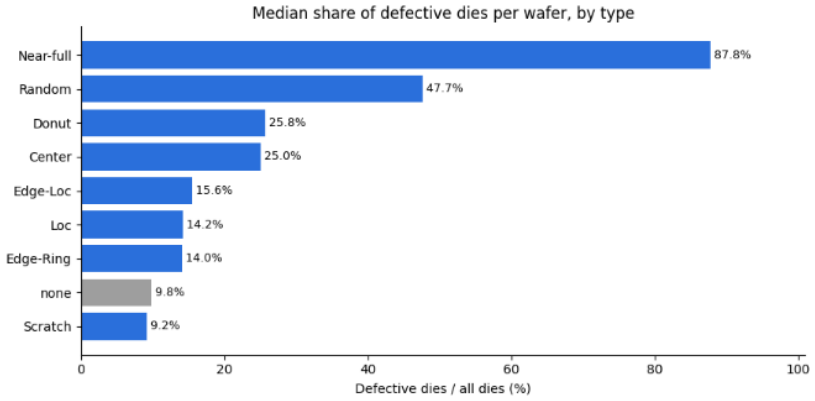


그림 3. 유형별 웨이퍼당 불량 다이 비율 중앙값

- Near-full(87.8%)과 Random(47.7%)은 웨이퍼 절반 이상이 불량으로 수율 손실이 가장 크다.

- Scratch(9.2%), Edge-Ring(14.0%), Loc(14.2%)는 none(9.8%)과 불량 다이 비율이 비슷하다. 불량의 '양'만으로는 유형을 구분할 수 없고 '모양과 위치'를 봐야 한다는 점이 이미지 기반 CNN 을 쓰는 근거다.

4. CNN 기반 불량 유형 분류

4.1 데이터 구성

- none 을 10,000 장으로 줄이고(다운샘플링) 불량 25,517 장은 모두 사용해 총 35,519 장을 구성했다. 한 변이 10 미만인 웨이퍼 2 장은 제외했다.
- 유형 비율을 유지하며 학습 24,863 장, 검증 5,328 장, 테스트 5,328 장(70/15/15)으로 나눴다.
- 입력은 64×64×2 채널이다. 첫 번째 채널은 웨이퍼 영역(값>0), 두 번째 채널은 불량 위치(값=2)를 나타내, 범주형 값을 숫자 크기로 오해하지 않도록 했다.

4.2 모델 구조와 학습

구성	내용
데이터 증강	학습 중 무작위 상하·좌우 반전과 회전(최근접 보간). 회전·반전해도 불량 유형은 바뀌지 않음
합성곱 층	[Conv 3×3 ×2 + BatchNorm + MaxPool] × 3 블록 (필터 32 → 64 → 128)
분류 층	GlobalAveragePooling → Dense 128 → Dropout 0.5 → Dense 9 (softmax)
학습 설정	Adam(학습률 0.001), batch 128, 최대 40 에폭, 조기 종료와 학습률 감소 적용, 시드 42 고정

4.3 클래스 가중치 비교 실험

불량 유형 간 불균형에 대응하기 위해 적은 유형의 오류를 더 크게 반영하는 클래스 가중치(balanced)를 적용한 모델과, 적용하지 않은 모델을 같은 조건에서 비교했다.

지표	가중치 적용	가중치 미적용 (최종)
macro F1	0.891	0.924
정확도	0.925	0.941
틀린 개수 (테스트 5,328 장)	400	313
정상을 불량으로 오판정한 비율	약 7.1%	약 4.5%

- 가중치를 적용하면 희귀 유형의 재현율은 조금 높았지만(예: Near-full 1.000 vs 0.955, 테스트 22 장 중 1 장 차이), 희귀 유형으로 잘못 예측하는 경우가 많아 정밀도가 크게 낮았다(예: Scratch 0.712 vs 0.927).
- none 다운샘플링만으로 불균형이 상당히 완화된 상태에서 큰 가중치(Near-full 약 26 배)를 더하자 소수 유형으로 과하게 쏠린 것으로 판단했다. 이에 따라 가중치를 적용하지 않은 모델을 최종 모델로 선택했다.

4.4 최종 모델 성능

유형	정밀도	재현율	F1	테스트 수
none	0.957	0.955	0.956	1,500
Center	0.949	0.953	0.951	644
Donut	0.846	0.928	0.885	83
Edge-Loc	0.899	0.904	0.901	779
Edge-Ring	0.984	0.988	0.986	1,452
Loc	0.864	0.846	0.855	539
Random	0.891	0.946	0.918	130
Scratch	0.927	0.855	0.890	179
Near-full	1.000	0.955	0.977	22
macro 평균	**0.924**	**0.926**	**0.924**	**5,328**

- Edge-Ring(F1 0.986), Near-full(0.977), none(0.956), Center(0.951)이 가장 잘 분류됐다. 대부분 위치가 뚜렷한 유형이다.
- 가장 약한 유형은 Loc(F1 0.855)이다. 위치가 정해지지 않은 국부 덩어리라, 덩어리가 중앙에 가까우면 Center, 가장자리에 가까우면 Edge-Loc 과 경계가 애매하다.

4.5 오분류 분석

- **유형 경계가 애매한 경우(가장 많음):** 가장자리에 닿은 큰 덩어리(Loc↔Edge-Loc), 중앙에서 약간 벗어난 덩어리(Center↔Loc), 일부만 이어진 고리(Edge-Ring→Edge-Loc). 모델 오류라기보다 라벨 기준 자체가 모호한 사례다.
- **패턴이 약한 경우:** 작고 흐린 덩어리를 정상으로 보거나, 우연히 몰린 흩어진 불량을 패턴으로 오인했다.

- **가는 선:** 잡음에 묻히거나 축소 과정에서 끊긴 짧은 스크래치를 정상으로 판정했다.

5. 품질관리 개선안

개선안	근거	실행 방안
① 가장자리 공정 집중 관리	Edge 계열이 불량의 58.3%	가장자리 다이 수율을 별도 지표로 관리하고, Edge 계열 비율 초과 시 가장자리 공정 조건(식각·증착·세정 균일도)을 우선 점검
② 로트 단위 이상 감지	로트별 불량 비율 양극화, 연속 로트 불량	로트별 불량 비율 관리도(SPC) 운영. 기준 초과 또는 연속 로트에서 같은 유형 반복 시 설비 점검 경보
③ AI 1 차 분류 + 엔지니어 재검토	판정되지 않은 웨이퍼 78.7%, 모델 macro F1 0.924	전 웨이퍼를 CNN 으로 분류하고 확신도가 낮은 웨이퍼만 엔지니어가 재검토. 검토 결과로 모델을 주기적으로 재학습
④ 심각도 기반 대응	불량 다이 비율 Near-full 87.8%, Random 47.7%	긴급(Near-full·Random): 즉시 로트 보류 / 주의(Center·Donut): 설비 점검 / 관찰(Edge·Loc·Scratch): 추세 모니터링
⑤ 판정 기준 표준화	오분류 대부분이 Loc·Edge-Loc·Center 경계 사례	유형별 정량 기준(예: 불량 덩어리 중심과 가장자리의 거리)과 경계 사례 이미지를 담은 판정 기준서 마련

유형과 공정 원인의 연결(예: Scratch 는 웨이퍼 이송·연마 등 물리적 접촉, Random 은 파티클 등 환경 오염과 자주 연관)은 일반적으로 알려진 경향이며, 실제 원인은 공장의 설비·공정 이력 데이터와 결합해 검증해야 한다.

6. 한계 및 향후 과제

- **실제 분포에서의 검증:** 테스트 세트도 none 을 줄인 데이터라, 정상이 약 85%인 실제 공정에서는 오경보가 늘 수 있다. 실제 비율의 데이터로 재검증이 필요하다.

- **라벨 없는 데이터 활용:** 라벨 없는 78.7%를 활용하는 준지도 학습(확신도가 높은 예측을 임시 라벨로 사용)을 검토할 수 있다.
- **리사이즈 정보 손실:** 큰 웨이퍼를 64×64 로 줄일 때 가는 스크래치가 끊길 수 있어, 더 큰 입력 크기나 채널별 비율 리사이즈를 검토할 수 있다.
- **실험 반복:** 한 번의 실행 결과이므로 여러 시드로 반복해 성능의 평균과 편차를 확인해야 한다.
- **원인 검증:** 공정·설비 데이터가 없어 불량 유형과 실제 원인의 관계는 검증하지 못했다.

7. 결론

WM-811K 웨이퍼 맵 분석 결과, 불량은 웨이퍼 가장자리(불량의 58.3%)와 특정 로트에 집중되는 경향을 보였다. 불량 다이의 양만으로는 유형 구분이 어렵다는 점을 확인하고, 웨이퍼 맵을 이미지로 학습하는 CNN 을 설계해 9 개 유형을 macro F1 0.924 로 분류했다.

모델 설계 과정에서 범주형 데이터에 맞는 최근접 리사이즈와 2 채널 입력을 사용했고, 클래스 가중치는 예상과 달리 성능을 떨어뜨린다는 것을 비교 실험으로 확인했다. 이는 모델 설계의 가정을 실험으로 검증하는 과정이 중요함을 보여준다.

이 결과를 바탕으로 가장자리 공정 집중 관리, 로트 단위 이상 감지, AI 1 차 분류와 엔지니어 재검토, 심각도 기반 대응, 판정 기준 표준화를 품질관리 개선안으로 제시한다.

[강사님께 궁금한 점]

4.3 클래스 가중치 비교 실험에서 원래는 클로드가 가중치를 줘야 한다고만 해서 가중치 준 상태에서만 진행하려고 했는데 가중치를 주지 않았을 때는 무슨 문제가 발생하는지, 아직 관련 개념이 이해가 안가는 이유 때문에 단순히 그냥 한번 비교를 해보고 싶어서 비교를 해달라고 했습니다. 그런데, 실제 모델 성능은 가중치를 안 준 쪽이 더 성능이 좋게 나왔다고 하더라고요..! 그래서 혹시나 이런 애매모호한 부분들은 실무에서는 어떻게 판단할 수 있을지가 궁금합니다. 매번 모든 경우의 수를 다 할 수는 없다고 생각이 듭니다.

또한, 생각보다 코드를 제가 처음부터 다 짜지 않는데 혼자 먼저 짜보는게 실무에서도 좋을까요?
소위 말하는 "딸깍"의 느낌이라서 클로드가 짜준 코드를 확인한다고 해서 그걸 검증하는 과정이 쉽지는 않은 것 같아요. 저보다 클로드가 더 똑똑한 느낌이라 저도 잘 모르는 게 크기 때문입니다..!
이런 ai 를 어떤 식으로 활용해야 좋을 지 고민이 많이 됩니다. 지금은 공개된 데이터라 아예 코드 실행

결과를 input 할 수 있다고 생각하지만 실무에서는 그럴 수 없으니..! 강사님은 어떤 식으로 진행하는지 궁금합니다.

마지막으로, 결론 부분도 제가 스스로 생각한 부분은 아닙니다. 아직 반도체 관련 지식이 부족해서 관련 분석 인사이트를 뽑아내기가 힘듭니다..! 혹시 강사님은 잘 모르는 도메인에 대한 프로젝트를 진행해보신 적이 있으실까요? 있으셨다면 어떤 과정들을 거치시며 프로젝트를 완수하셨는지도 궁금합니다.