



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

공학석사 학위논문

딥 러닝 및 서포트 벡터 머신기반 실시간
센서 고장 검출 기법에 관한 연구

A Study on Realtime Sensor Fault Detection Scheme based on Deep
Learning and Support Vector Machine

울 산 대 학 교 대 학 원
전기전자컴퓨터공학과
양 재 완

딥 러닝 및 서포트 벡터 머신기반 실시간
센서 고장 검출 기법에 관한 연구

지도교수 구인수

이 논문을 공학석사학위 논문으로 제출함

2018 년 12 월

울산대학교 대학원

전기전자컴퓨터공학과

양재완

양재완의 공학석사학위 논문을 인준함

심사위원장 공 형 윤 인

심 사 위 원 구 인 수 인

심 사 위 원 노 영 태 인

울 산 대 학 교 대 학 원

2018 년 12 월

감사의 글

대학원 석사과정 동안 저를 도와주시고 응원해주신 분들 모두에게 이 글을 통해 감사의 말씀을 전합니다.

우선 석사과정 동안 부족한 저에게 항상 아낌없는 지원과 지도를 해주신 구인수 교수님께 감사드립니다. 그리고 더 좋은 논문이 될 수 있도록 조언과 격려를 해주신 공형윤 교수님, 노영태 교수님께 감사의 말씀을 전합니다.

연구실에서 항상 어려운 일이 있을 때마다 도와주시고 힘이 되어주신 이영두 박사님께 감사를 드립니다. 그리고 항상 티격태격 했지만 함께했던 이규형 학생과 연구실 멤버 모두에게 고맙다는 말을 전하고 싶습니다.

캡스톤 디자인부터 대학원 인연을 함께한 범석이, 태훈이, 규진이, 기석이 고맙다. 또한 언제나 응원해주던 율경이형도 감사해요.

마지막으로 항상 옆에서 응원해주고 믿어주는 가족들에게 감사의 마음을 드립니다. 앞으로도 많이 배우고 더 많이 노력하겠습니다. 감사합니다.

[국문요약]

딥 러닝 및 서포트 벡터 머신기반 실시간 센서 고장 검출 기법에 관한 연구

울산대학교 대학원
전기전자컴퓨터공학과
양재완

최근 자동화 기술과 4차 산업혁명의 핵심기술인 인공지능 기술의 발전으로 산업 현장에서의 공정 기계들은 단순한 공정을 반복하는 것만 아니라 고도의 제어 기술을 필요로 하는 작업까지 수행하면서 자동화 기계들의 관리 및 유지보수에 대한 관심이 커지고 있다. 대부분의 자동화 기계들은 다양한 센서들로부터 수집된 데이터를 기반으로 제어된다. 따라서 자동화 기계들에 부착되는 센서에 고장이 발생한다면 기계들의 오작동으로 인한 공정라인 운영에 막대한 경제적 손실뿐만 아니라 인명피해도도 이어질 수 있다. 이를 막기 위해 센서 고장을 실시간으로 감지하는 시스템이 필수적이다.

일반적으로 터빈, 모터, 베어링 등 기계의 고장 진단 분야에서는 신호 처리 및 분석기법이나 규칙기반 전문가 시스템을 적용한 연구가 활발히 이루어져 왔다. 하지만 기존의 연구방법을 적용하여 센서 자체의 고장을 판단하는 것은 힘들며, 새로운 고장 유형이 발생하였을 경우 알고리즘의 추가 및 수정이 어렵다.

본 논문에서는 이를 해결하기 위해 머신러닝 방법인 Support Vector Machine(SVM)과 딥 러닝 기법중 하나인 Convolution Neural Network(CNN)을 사용하여 센서 고장 진단 및 유형을 분류하였다. 또한 CNN 분류기를 사용한 실시간 모니터링 시스템을 구축하였다.

SVM을 사용한 고장 진단에서는 정확도와 효율을 높이기 위해 학습에 사용될 특징들(Time-domain statistical features, statistical features based on auto-encoder, latent variable of variational auto-encoder)에 유전 알고리즘을 적용하여 특징들을 선별하고 다중 계층 SVM을 구성하여 센서 고장을 분류하였다. CNN에서는 분류기의 성능을 높이기 위해 Convolutional Auto-Encoder(CAE)를 사용하여 pre-training 시킨 후 디코더 부분을 제거한 뒤 fully connected layer를 연결하여 전체 layer를 fine-tuning시켰다. 실시간 모니터링 시스템 구현에서는 센서 데이터를 서버로 보내고 서버에서는 CNN으로 고장을 판별 및 데이터를 데이터베이스에 저장한 뒤 웹 페이지를 통해 결과를 실시간으로 보여준다.

주요어 : 센서 고장 진단(Sensor Fault Diagnosis), 컨볼루션 신경망(Convolution Neural Network), 오토인코더(Auto-Encoder), 서포트 벡터 머신(Support Vector Machine), 시간 영역 통계적 특징(Time-domain Statistical Features), 변분 오토인코더(Variational Auto-Encoder), 유전 알고리즘(Genetic Algorithm), 실시간 모니터링(Realtime monitoring)

목 차

국문요약	I
목 차	III
그림목차	V
표 목 차	VII
1. 서 론	1
2. 연구의 이론적 배경	
2.1. 머신러닝 및 딥 러닝	4
2.1.1. 모델 성능 테스트 및 검증	6
2.1.2. 서포트 벡터 머신(Support Vector Machine)	10
2.1.3. 인공신경망(Artificial Neural Network)	14
2.1.4. 오토인코더(Auto-Encoder)	17
2.1.5. 변분 오토인코더(Variational Auto-Encoder)	18
2.1.6. 컨볼루션 신경망(Convolution Neural Network)	20
2.2. 유전 알고리즘	23
2.2.1. 유전 알고리즘 기본개념 및 연산과정	23
2.2.2. 적합도 함수(Fitness Function)	25
2.2.3. 선택연산(Selection)	26
2.2.4. 교차연산(Crossover)	27
2.2.5. 변이연산(Mutation)	28
2.2.6. 대치연산(Replacement)	29
3. 센서 고장 신호 유형	30
4. SVM 기반 센서 고장 분류	32

4.1. 센서 고장 검출 및 분류를 위한 특징추출	32
4.1.1. 시간 영역 통계적 특징	32
4.1.2. 오토인코더 이상치 기반 통계적 특징	34
4.1.3. 변분 오토인코더 잠재변수 기반 특징	37
4.2. 유전 알고리즘 기반의 특징 선택	40
4.3. 특징별 SVM 분류 결과 분석 및 다중 계층 SVM 구현	44
5. CNN 기반 센서 고장 분류	50
5.1. CNN 모델 구현 및 학습	50
5.2. CNN 모델 성능 개선 및 분류 결과	53
6. SVM, CNN 센서 고장 분류 성능 비교분석	57
7. CNN 기반 센서 상태 모니터링	59
8. 결 론	61
참 고 문 헌	62
ABSTRACT	67

그림 목 차

그림 2.1	학습목표에 따른 머신러닝 기법 분류	5
그림 2.2	머신러닝과 딥 러닝 학습 과정	6
그림 2.3	모델 평가를 위한 데이터 분배	8
그림 2.4	K겹 교차검증	9
그림 2.5	SVM 선형 분류	10
그림 2.6	커널 함수를 이용한 SVM 분류	13
그림 2.7	단층 퍼셉트론 구조	14
그림 2.8	다층 퍼셉트론과 심층 신경망	15
그림 2.9	활성화 함수	16
그림 2.10	오토인코더의 기본 동작 구조	17
그림 2.11	변분 오토인코더 네트워크	18
그림 2.12	컨볼루션 신경망 동작과정 및 구조	20
그림 2.13	컨볼루션층 동작과정	21
그림 2.14	풀링층 동작과정	22
그림 2.15	유전 알고리즘 구현	24
그림 2.16	선택연산 과정	26
그림 2.17	단일 점 교차 방식	27
그림 2.18	유전적 변이효과	28
그림 2.19	대치과정 예제	29
그림 3.1	정상 신호와 각 유형별 신호 비교	30
그림 3.2	무작위 지점에서 발생한 각 고장에 대한 데이터 샘플들	31
그림 4.1	시간 영역 통계적 특징 추출과 SVM을 사용한 센서 고장진단 흐름도	33
그림 4.2	컨볼루션셔널 오토인코더 기반의 특징 추출과 SVM을 사용한 센서 고장진단 순서도	34
그림 4.3	제안된 컨볼루션셔널 오토인코더 동작과정 및 구조	35
그림 4.4	컨볼루션셔널 오토인코더 기반의 특징 추출 과정	36
그림 4.5	VAE 기반 특징 추출과 SVM을 사용한 센서 고장진단 순서도	37
그림 4.6	변분 오토인코더 기반 특징 추출 동작 과정	38
그림 4.7	센서 입력 데이터(좌)와 복원된 센서 데이터(우)	39
그림 4.8	VAE에 의해 추출된 센서 데이터의 잠재변수	39

그림 4.9	해의 평균 적합도 점수 변화	43
그림 4.10	센서 고장 진단 전체과정	44
그림 4.11	One-versus-rest 방법	45
그림 4.12	다중 계층 SVM	48
그림 5.1	센서 고장 분류를 위한 CNN 동작 과정	50
그림 5.2	CNN의 손실과 에러율 모니터링	51
그림 5.3	제안된 CNN 구조 및 센서 고장감지 과정	52
그림 5.4	CNN 성능 개선 과정	53
그림 5.5	학습 및 검증 손실 모니터링	54
그림 5.6	CNN의 손실과 에러율 모니터링	56
그림 6.1	SVM과 CNN 정확도 비교(%)	57
그림 6.2	SVM, CNN의 AUC-ROC	58
그림 7.1	센서 고장 모니터링을 위한 전체 과정	59
그림 7.2	센서 고장 진단 결과 모니터링	60

표 목 차

표 2.1 비선형 분류를 위한 SVM 커널 함수	14
표 3.1 센서 데이터 고장 유형 및 특징	30
표 4.1 SVM에서 사용된 N 개의 데이터 포인트 x_k 로부터 추출된 시간 영역 통계적 특징	33
표 4.2 유전 알고리즘으로부터 선택된 특징들	44
표 4.3 특징에 따른 SVM 정확도(%)	46
표 4.4 SVM-Poly 분류 결과에 대한 혼동행렬(%)	48
표 5.1 CNN 분류 결과에 대한 혼동행렬(%)	52
표 5.2 Epoch에 따른 센서 데이터 복원	55
표 5.3 향상된 CNN 분류 결과에 대한 혼동행렬(%)	56
표 6.1 SVM과 CNN 분류 결과(%)	57

1. 서 론

1.1 연구의 배경 및 목적

디지털 변환 및 기계기구 구동 기술들의 급격한 발전에 힘입어 산업 현장에서 사용되는 기계들의 자동화 및 무인화가 가속되고 있으며, 최근 4차 산업혁명의 핵심 기술인 인공지능, 빅데이터, 사물인터넷과 같은 기술을 기반으로 제조 산업의 혁신적인 변화를 일으킬 것이라 전망되고 있다. 미국, 독일 등이 4차 산업혁명을 선도하는 가운데 우리나라도 제조업 3.0을 기반으로 2014년부터 스마트 팩토리의 도입을 적극적으로 추진하고 있다[1].

스마트 팩토리에서 자동화된 생산 공정라인을 최대 효율로 가동시키기 위해서는 공정에서 사용되는 장비들이 오작동 없이 지속적으로 동작해야한다. 이러한 이유로 자동화 기계들의 관리와 유지보수에 대한 관심이 커지고 있으며 관련된 연구 또한 활발히 이루어지고 있다[2]-[4]. 최근에는 사물인터넷 기술과 인공지능 기술이 기존 공정라인에 접목되어 단순한 공정을 반복하는 작업에서 고도의 기술을 필요로 하는 작업으로 빠르게 변화하면서 자동화 기계뿐만 아니라 기계에 부착되는 센서의 중요성도 더욱 커지고 있다. 스마트 팩토리 시스템의 핵심은 공장 내 기계와 설비에 부착된 다양한 센서들로부터 수집된 데이터를 기반으로 기계의 구동상태 및 제어량을 파악하고 전체 공정라인의 작업 속도와 양을 조정하여 생산을 최적화시키는 것이다. 따라서 센서에 이상이 생길 시 데이터 오류로 인한 기계들의 오작동으로 공정라인 운영에 막대한 경제적 손실은 물론이고 심하게는 인명피해로도 이어질 수 있다. 2016년에는 아시안항공 여객기가 연기 감지 센서의 오작동으로 인해 이륙한 지 1시간 만에 긴급 착륙하여 운행에 차질이 생겼고 2016년과 2018년 두 차례 테슬라의 자율주행 자동차의 센서 인식 오류로 인한 인명피해가 발생하였다[5]-[6]. 이러한 센서 고장으로 인한 피해를 막기 위해서는 센서 고장을 실시간으로 감지하는 시스템이 필수적이다.

기존의 고장진단 분야에서는 주어진 모델을 기반으로 다양한 신호 처리 및 분석

기법, 규칙기반 전문가시스템, 퍼지 이론들을 적용한 연구가 활발히 이루어져 왔다 [7]-[11]. 그러나 이러한 방법들은 작업 환경의 복잡성으로 인하여 일부의 방법들만 적용가능하다는 저가용성 문제와 새로운 고장 유형이 발생할 경우에는 분석 및 대처를 위한 알고리즘의 추가 및 수정이 어렵다는 문제가 있다. 최근에는 이를 해결하기 위해 Support Vector Machine(SVM), Neural Network(NN)와 같은 지능형 분류기법이 사용되고 있다[12]-[13]. 하지만 대부분의 고장 진단 연구들은 모터, 베어링 등 기계 고장에 적용된 방법으로 기존의 연구방법을 적용하여 센서의 고장을 판단하는 것은 힘들며, 센서자체의 고장과 관련된 연구는 미흡하다.

본 논문에서는 기계학습 방법인 Support Vector Machine(SVM)과 딥 러닝 기법 중 하나인 Convolution Neural Network(CNN)을 사용하여 erratic, drift, hard-over, spike, stuck의 총 5가지 대표적인 유형의 센서 고장을 진단 및 분류하는 방법을 제안한다.

SVM을 사용한 고장 분류에서는 효율적으로 모델을 학습시키기 위해 특징 추출, 특징 선택과정을 거친다. 특징 추출 단계에서는 센서 데이터로부터 시간 영역 통계적 특징, 오토인코더 기반의 시간 영역 통계적 특징, 변분 오토인코더 잠재변수 기반 특징들을 추출한다. 특징 선택과정에서는 유전 알고리즘(Genetic Algorithm, GA)을 사용하여 센서 데이터로부터 추출한 특징들 중에서 최적의 특징들을 선정한다. 선택된 특징들을 사용하여 고장 클래스별로 SVM을 학습시키고 이를 기반으로 multi-layer SVM을 구성한다. CNN 기반의 센서 고장 분류에서는 분류기의 성능향상을 위해 auto-encoder방식으로 pre-training하고 fine-tuning을 통해 CNN분류기를 구현하였다. 또한 비교적 성능이 우수한 CNN 분류기를 사용한 실시간 센서 고장 모니터링 시스템을 구현하였다.

1.2 논문구성

본 논문은 다음과 같은 순서로 구성되어 있다. 2장에서는 본 논문에 사용된 기계 학습, 딥 러닝, 유전 알고리즘에 관한 이론적 배경에 대해 기술한다. 3장에서는 본 논문에서 사용된 센서 고장 신호 유형에 대해 설명한다. 4장에서는 다양한 센서 특

정과 GA를 활용한 특징선택 기반의 SVM 센서 고장 분류를 기술한다. 5장에서는 CNN 기반 센서 고장 분류를 기술하며, 6장에서는 센서 고장 분류에 사용된 SVM, CNN 기반의 분류기 성능을 비교 분석한다. 7장에서는 CNN 기반의 센서 고장 진단 모니터링 시스템을 소개한 뒤 8장에서 본 논문의 결론을 맺는다.

2. 연구의 이론적 배경

2.1. 머신러닝 및 딥 러닝

국내에서 2016년 이세돌 9단과 구글 딥마인드의 알파고(AlphaGo)의 바둑대결로 알파고의 핵심기술과 관련된 인공지능, 머신러닝, 딥 러닝 그리고 이를 기반으로 하는 4차 산업혁명과 스마트 팩토리 등에 대한 관심이 폭발적으로 증가하였다. ‘알파고 쇼크’ 이후 인공지능과 관련된 이슈들은 연일 화제가 되었으며, 이와 관련된 투자 및 연구개발이 확대되었다. 한국 정부는 올해부터 5년간 인공지능과 관련된 연구개발에 2조 2000억 원을 투입한다고 발표하였고 국내 기업들 또한 많은 연구예산을 인공지능에 투입하고 있다[14].

인공지능이란 인간의 사고방식을 모방한 지능을 구현하기 위한 시스템 혹은 컴퓨터 기술을 말한다. 1950년대에 인공지능이라는 개념이 처음 등장하였으며, 최근 병렬 처리 성능이 우수한 GPU의 도입으로 연산능력이 증가함과 동시에 데이터들이 폭발적으로 늘어나면서 관련 분야가 확대되고 있는 중이다. 이러한 인공지능을 구현하기 위한 구체적인 접근 방법들이 있는데 바로 머신러닝과 딥 러닝이다.

머신러닝은 패턴인식이라고도 불리는데 이는 컴퓨터가 주어진 입력으로부터 패턴을 추출하여 스스로 학습하기 때문이다. 여기서 스스로 학습한다는 것에서 초기에 등장한 인공지능과는 많은 차이가 있다. 초기의 인공지능 시스템은 규칙기반의 인공지능으로 사용자가 어떻게 동작할 것인지에 대한 여러 가지 규칙을 설정하고 이를 기반으로 정해진 패턴에 따라 동작한다. 하지만 머신러닝은 사용자가 정의한 규칙으로 동작하는 것이 아닌 컴퓨터가 입력된 데이터를 통해 스스로 패턴을 찾아 학습하며, 학습한 내용을 기반으로 판단이나 예측을 수행한다. 다시 말하자면, 머신러닝은 알고리즘에 따라 변하는 것이 아니라 데이터에 따라 변하는 것이다. 머신러닝 기법에는 그림 2.1과 같이 학습 유형에 따라 크게 지도 학습, 비지도 학습, 강화 학습으로 분류할 수 있다. 지도 학습은 데이터와 주어진 데이터에 대한 레이블을 함께 학습시켜 데이터 입력에 따른 상태나 값을 출력한다. 비지도 학습은 레이블 없

이 데이터만으로 학습시키는 방법으로 데이터와 레이블 간에 관계 추론이 아닌 데이터 값 자체의 특성을 찾아내는 학습방법이다. 강화 학습은 프로그램에 목표가 주어지고 그 목표에 적합한 행동을 취하도록 학습을 강화시키는 기법이다. 대표적으로 사용되는 머신러닝 알고리즘은 의사결정트리(Decision Tree), 서포트 벡터 머신(Support Vector Machine), K-최근접 이웃(K-Nearest Neighbors), 로지스틱 회귀(Logistic Regression) 등이 있다[15]-[18]. 본 논문에서는 지도 학습과 비지도 학습방법을 사용하여 문제를 해결한다.

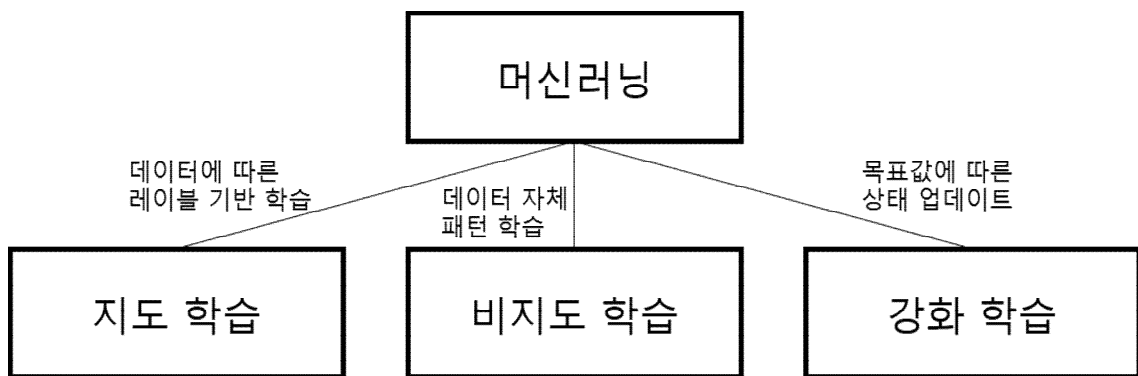


그림 2.1. 학습목표에 따른 머신러닝 기법 분류

Fig 2.1. Classification of machine learning techniques according to learning objectives

딥 러닝은 인공신경망을 기반으로 구현한 머신러닝의 한 갈래로 지도, 비지도, 강화 학습 같은 방식으로 문제를 풀기 위해 사용된다. 머신러닝과 딥 러닝의 다른 점은 데이터로부터의 특징 추출과정에 있다. 일반적으로 머신러닝 알고리즘을 사용할 경우 원래의 데이터로부터 특징을 추출하는 과정이 포함되며, 이 과정에서 인간이 개입하여 사용되는 특징 추출 방법에 따라 성능이 좌우된다. 반면에 딥 러닝은 자체적으로 특징 추출과정을 포함하기 때문에 따로 사람이 특징을 추출하는 과정 필요 없이 주어진 데이터가 그대로 학습에 사용된다. 따라서 처음부터 끝까지 기계가 학습한다는 의미에서 딥 러닝은 end-to-end machine learning이라고 불린다. 딥 러닝은 특징 추출부터 판별, 예측 기능까지 모두 소화가능하기 때문에 목적에 따라 다

양하게 응용되어 사용될 수 있다. 최근에는 딥 러닝을 활용하여 기존 머신러닝으로 해결하기 어려웠던 이미지 인식, 음성 인식 등과 같은 분야에서 매우 좋은 성능을 보여주고 있다[19]-[20]. 딥 러닝에서 대표적인 알고리즘은 컨볼루션 신경망 (Convolution Neural Network), 순환신경망(Recurrent Neural Network) 등이 있다.

2.1.1. 모델 성능 테스트 및 검증

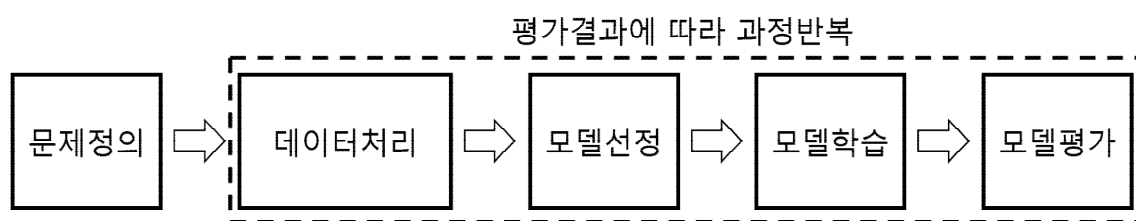


그림 2.2. 머신러닝과 딥 러닝 학습 과정

Fig 2.2. Learning process of machine learning and deep learning

머신러닝 혹은 딥 러닝 기법을 활용한 모델을 사용하기 위해 설계한 모델이 사용자의 목적에 적합하게 동작하는가를 확인하고 만약 기대했던 성능이 나오지 않는다면 무엇이 문제인가를 파악하고 적절한 조치를 취해야 할 필요가 있다. 모델의 성능을 향상시키기 위해 보유한 데이터가 적합한 데이터인지, 모델의 학습방법이 적절한지, 모델에 대한 평가가 잘 이루어졌는지 등을 고려해야 하며, 이와 관련된 기법들을 시도해보는 과정을 통해 좋은 해답을 얻어 낼 수 있다.

그림 2.2에서는 머신러닝 혹은 딥 러닝 모델을 학습시키기 위한 과정을 보여주고 있다. 우선 해결하고자 하는 문제를 정의하고 모델을 학습하기 위해 데이터 처리 단계에서 수집된 데이터를 가공한다. 여기서 데이터 처리는 머신러닝, 딥 러닝 모델을 효율적으로 학습시키기 위한 데이터 특징 추출, 원래의 데이터 혹은 데이터로부터 추출된 특징에서 상관관계를 추론하여 중복된 데이터를 제거하거나 선택하는 과정, 데이터 확장(Data Augmentation) 등이 포함될 수 있다[13].

모델선정 과정에서는 데이터에 맞는 적절한 모델을 선정하고 알고리즘에 필요한 손실함수를 설계한다. 여기서 말하는 손실함수(Loss function)는 오차함수(Error function), 비용함수(Cost function)과 같은 표현으로도 사용되며, 사용자가 선정한 모델의 학습목표임과 동시에 모델이 얼마나 잘 동작하는지를 산술적으로 표현한 지표가 된다. 또한 손실함수 식의 미분을 통해 모델의 파라미터 값들을 갱신하여 모델에 맞는 파라미터들로 조정할 수 있기 때문에 사용된다. 손실, 비용, 오차라는 표현이 사용되기 때문에 일반적으로 값을 최소화시키는 것을 목적으로 한다. 주로 사용되는 손실함수로는 평균 제곱 오차(Mean Squared Error, MSE), 교차 엔트로피 오차(Cross Entropy Error, CEE)로 식(2.1)과 같다[19], [21].

$$MSE = \frac{1}{N} \sum_k (y - \hat{y})^2, \quad CEE = -\frac{1}{N} \sum_k [y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})] \quad (2.1)$$

위의 식에서 y 는 실제 값, \hat{y} 는 모델 예측값을 의미하며, 손실함수는 실제값과 예측값에 대한 차이를 수치로 나타내고 오차를 줄이는 방향으로 모델이 학습된다.

모델학습에서는 설정된 모델, 수식, 데이터 특징 등을 고려하여 그에 적합한 파라미터 값을 최적화 방법들을 통해 모델을 학습시킨다. 학습한다는 것은 모델이 수학적으로 데이터를 잘 표현할 식을 갖춘다는 의미이며, 식의 파라미터 값을 갱신하여 손실함수의 값을 최소화하는 최적화가 진행된다. 주로 사용되는 최적화 방법으로는 경사하강법(Gradient descent), 확률적 경사하강법(Stochastic gradient descent), 역전파(Backpropagation), AdaGrad(Adaptive gradient)최적화 방법, Adam(Adaptive Moment Estimation)최적화 방법 등이 있다[18]-[23].

마지막으로 해당 모델에 대한 평가를 수행한다. 모델을 평가할 때는 손실함수와 정확도를 기반으로 진행하며, 학습 데이터뿐만 아니라 새로운 데이터를 입력으로 사용했을 경우에도 잘 동작하는지 측정해야한다. 이를 일반화라고 하며 전체 데이터를 그림 2.3과 같이 나눠 평가를 진행한다.



그림 2.3. 모델 평가를 위한 데이터 분배

Fig 2.3. Data distribution for model evaluation

학습을 위해 전체 데이터를 사용하지 않는 것은 데이터를 낭비하는 것처럼 보일 수 있지만 모든 데이터를 모델학습에 사용할 경우 평가가 제대로 수행되지 않으므로 과적합(Overfitting)과 같은 오류를 범할 수 있다. 과적합은 모델자체를 지나치게 학습 데이터의 정보에 의존하여 최적화시킨 경우를 말한다. 학습에 사용된 데이터의 분포는 패턴이 한정되어 있어 복잡한 모델을 사용하여 파라미터를 갱신하면 학습 데이터에게 완벽하게 맞춰 학습 데이터에 대한 예측 정확도는 매우 높게 나오게 된다. 하지만 새로운 데이터에 대한 정확도는 떨어지는데 심지어 같은 클래스에 속하는 데이터일지라도 완벽하게 학습 데이터 특성과 일치하지 않으면 다른 부류라고 분류한다. 즉, 일반화가 제대로 되지 않는 것이다. 따라서 그림 2.3과 같이 올바른 모델평가를 위해 전체 데이터를 분배한다.

그림 2.3의 상단은 일반적으로 모델을 평가하기 위해 중복되지 않도록 전체 데이터를 학습 데이터와 테스트 데이터로 나눈 것을 보여준다. 학습 데이터는 모델학습을 위해 사용되며, 테스트 데이터는 모델이 잘 일반화하는지 측정하기 위해 사용된다. 데이터를 나눌 때 테스트 데이터는 모델로부터 의미 있는 결과를 도출하기 위해 데이터 세트의 특성을 잘 대표할 수 있는 값으로 이루어져야한다. 테스트 데이터가 너무 지나치게 한 쪽 방향으로 치우쳐진 데이터로 구성되어 있다면 공정한 평

가가 될 수 없다.

검증 데이터는 학습과정 도중에 학습이 잘 이루어지는지 평가하기 위한 데이터 세트이다. 전체 학습 데이터에 대해 학습을 한번 완료하였을 시 1 에포크(epoch)라고 하는데 epoch마다 검증 데이터를 사용하여 최적화 과정이 얼마나 잘 진행되었는지 평가할 수 있다. 일반적으로 epoch마다 검증 데이터를 이용해 측정하기 때문에 과적합 현상을 피해 가장 성능이 좋았던 epoch의 모델을 다시 불러와 테스트한다. 하지만 전체 데이터가 적어 검증 데이터를 할당하기 어려울 경우 주로 교차검증을 사용한다. 교차검증은 학습 데이터와 테스트 데이터를 여러 번 나누어 실행한다. 그렇기 때문에 모든 데이터 샘플들은 테스트 받을 기회를 가져 데이터 사용과 검증 효율을 높일 수 있다. 대표적으로는 K겹 교차검증(K-fold cross-validation)방법이 있다[17]. 그림 2.4에서는 K-fold cross-validation의 예를 보여주고 있다. 그림 2.4처럼 전체 데이터를 4등분으로 분리한 뒤 한 세트는 테스트로 사용하고 나머지 세트는 학습 데이터로 사용한다. 이것을 번갈아가며 4번 진행하며 각 결과의 정확도 평균을 구해 검증한다. 데이터 세트가 많을 경우에는 여러 번 학습과 평가를 반복하기 때문에 연산량이 많아 시간이 오래 걸릴 수 있다.

이러한 전체 과정을 거쳐 원하는 모델을 획득할 때까지 반복하면서 모델을 최적화시킨다.



그림 2.4. K겹 교차검증

Fig 2.4. K-fold cross-validation

2.1.2. 서포트 벡터 머신(Support Vector Machine)

통계적 학습기법에 기반을 둔 서포트 벡터 머신(Support Vector Machine, SVM)은 Vapnik에 의해 개발되었으며 주로 지도학습에 의한 패턴인식분야에서 적용되었다. 최근에는 SVM 알고리즘이 음성인식, 영상인식, 뇌 신호처리, 금융데이터 분석 등 다양한 분야에서 적용되어 우수한 성능을 보여주고 있다[17]-[18].

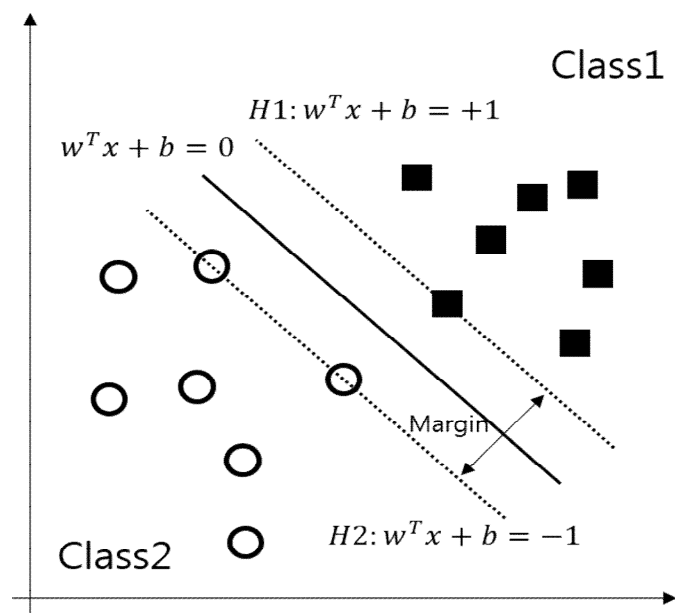


그림 2.5. SVM 선형 분류

Fig 2.5. SVM linear classification

SVM은 기본적으로 두 개의 클래스를 가진 학습 데이터들을 구분하기 위해 결정 경계와 가장 인접한 서포트 벡터를 이용하여 두 범주 사이의 거리를 최대화 시키는 최적의 초평면을 찾는 이진분류기법이다[17]. SVM에서는 클래스를 잘 분류하기 위해서 초평면으로부터 두 클래스 그룹의 거리가 멀어지도록 하는데 그 이유는 클래스 그룹간의 사이가 클수록 확실하게 특징구분이 가능하며 입력데이터를 판별하는데 더 높은 신뢰성을 줄 수 있기 때문이다. SVM은 초평면을 기준으로 가장 거리가

가까운 두 가지 클래스 데이터들의 거리가 최대가 되도록 하는 초평면을 설계하며, 여기서 각 클래스들의 데이터들 중 결정경계에 가장 거리가 짧은 데이터 벡터를 서포트 벡터라고 한다. SVM의 강점은 서포트 벡터가 결정이 되면 모든 데이터의 내적거리를 고려하지 않고 서포트 벡터와의 내적거리만 계산하면 되므로 효율적으로 계산비용을 줄일 수 있다.

SVM에서 그림 2.5와 같이 두 데이터 집합에 대한 초평면 식은 다음과 같이 표현된다.

$$w^T x + b = 0 \quad (2.2)$$

이 식에서 w 는 초평면의 가중치 벡터, x 는 데이터벡터 그리고 b 는 절편을 의미한다. 서포트 벡터 위에 있는 각각의 초평면(H1,H2)을 기준으로 입력데이터 x 가 속한 클래스는 다음과 같이 나타낼 수 있다.

$$\begin{aligned} H1 : w^T x + b &\geq +1 \text{ for } y_i = +1 \\ H2 : w^T x + b &\leq -1 \text{ for } y_i = -1 \end{aligned} \quad (2.3)$$

x 에 대한 클래스를 결정하는 함수는 SVM에서 다음과 같이 정의된다.

$$f(x) = \text{sign}(w^T x + b) \quad (2.4)$$

SVM에서 클래스를 판별하는 초평면의 H1과 H2의 사이의 거리 값은 최대가 되도록 시키는 문제를 최적화 문제에 적용시킬 수 있으며, 두 초평면의 거리는 $\frac{2}{\|w\|}$ 로 표현되며 이를 최대화 시키는 문제를 수학적 편의를 위해 $\frac{1}{2} \|w^T w\|$ 로 최소화 시

키는 문제로 바꿀 수 있다. 따라서 다음과 같이 식을 나타낼 수 있다.

$$\begin{aligned} & \text{Minimize } \frac{1}{2} \|w^T w\| \\ & \text{Subject to } y_i(w^T x + b) - 1 \geq 0, i = 1, 2, 3, \dots, l \end{aligned} \quad (2.5)$$

식 (2.5)와 같은 최적화 문제를 해결하기 위해 라그랑주 승수법을 적용시킨다. 라그랑주 승수법은 최적화 문제에서 최대 혹은 최소값을 찾기 위한 수학적 기법으로 목적함수와 제약 조건을 이용하여 제약이 없는 문제로 변경한 뒤 미분 방식으로 문제를 해결한다[24]-[25]. 식 (2.5)에 라그랑주 승수법을 적용시키면 다음과 같이 유도될 수 있다.

$$w = \sum_{i=1}^l a_i y_i x_i, \quad \sum_{i=1}^l a_i y_i = 0 \quad (2.6)$$

위의 식에서 w 는 가중치 벡터, a_i 는 라그랑지안 승수 그리고 y_i 는 클래스를 의미한다. 라그랑지안 승수를 사용하여 식 (2.7)과 같이 최소화가 아닌 최대화 문제인 쌍대 문제(Dual formulation)로 변환하여 해결할 수 있다.

$$\begin{aligned} & \text{Maximize } \sum_{i=1}^l a_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l a_i a_j y_i y_j x_i \cdot x_j \\ & \text{Subject to } \sum_{i=1}^l a_i y_i = 0, a_i \geq 0 \end{aligned} \quad (2.7)$$

또한 결정함수를 식 (2.6)를 이용하여 다시 표현한다면 다음과 같다.

$$f(x) = \text{sign}\left(\sum_i^l a_i y_i x_i \cdot x + b\right) \quad (2.8)$$

SVM에서 사용되는 데이터가 선형적인 특성을 가질 경우 위와 같은 과정을 통하여 우수한 분류 성능을 얻을 수 있다. 하지만 비선형 데이터의 경우에는 제대로 클래스를 판별할 수 없으며, 다양한 비선형 처리과정을 추가해야한다[17].

비선형 분류 문제를 해결하기 위해 비선형 데이터에 대해서도 잘 처리할 수 있도록 커널 함수(Kernel function)를 이용한 SVM 알고리즘을 설계한다. 커널 함수를 사용하게 되면 입력 데이터의 공간을 저차원에서 고차원으로 사상시킨 후 초평면을 이용하여 선형적으로 클래스를 분류할 수 있다. 그림 2.6은 커널 함수를 이용한 클래스 분류를 보여준다.

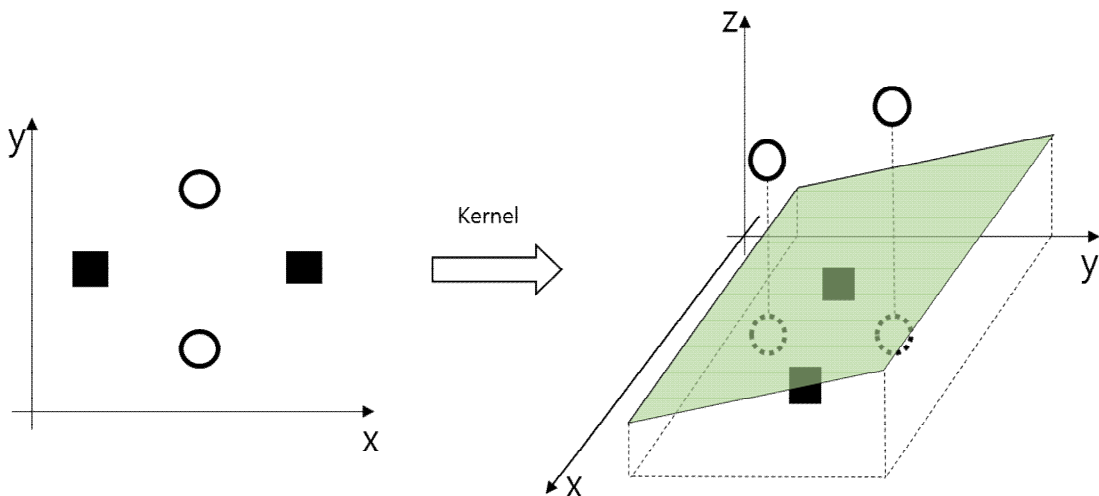


그림 2.6. 커널 함수를 이용한 SVM 분류

Fig 2.6. SVM classification using kernel function

SVM에서 일반적으로 커널 함수로는 표2.1과 같이 linear kernel, RBF(Radial basis function) kernel, polynomial kernel 함수 등이 사용된다. 커널 함수를 적용하여 나타낸 결정함수 식 다음과 같이 표현된다.

$$f(x) = \text{sign}\left(\sum_i^l a_i y_i K(x_i, x) + b\right) \quad (2.9)$$

표 2.1. 비선형 분류를 위한 SVM 커널 함수

Table 2.1. Kernel function of SVM for nonlinear classification

Kernel Type	Kernel Function
	$K(x_i, x), i = 1, 2, \dots, l$
linear	$x_i^T x + c$
RBF	$e^{-\frac{\ x_i - x\ ^2}{2\sigma^2}}$
polynomial	$(x_i^T x + 1)^d$

2.1.3. 인공신경망(Artificial Neural Network)

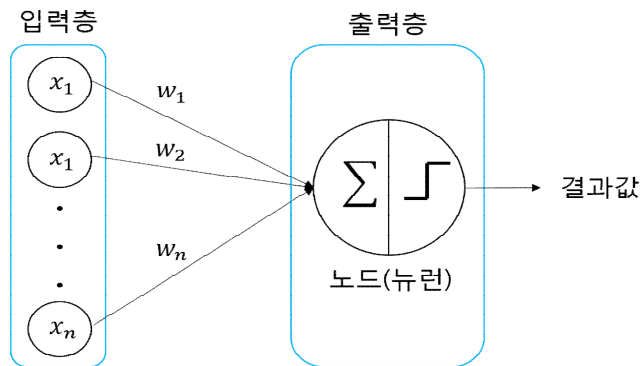


그림 2.7. 단층 퍼셉트론 구조

Fig 2.7. Structure of single-layer perceptron

인공신경망(Artificial Neural Network, ANN)은 딥 러닝의 기본적인 개념으로 인간의 뇌를 구성하는 신경세포, 뉴런의 동작 원리와 생물학적 구조를 모방하여 수학적 모델링한 것이다[26]. 뉴런은 시냅스를 통해 다른 뉴런으로부터 전기적 신호를 전달받는데 이 신호의 세기가 어떤 임계값을 초과할 경우 활성화를 통해 단일 신호를 다른 신경세포로 전달한다. 이러한 과정을 구현한 가장 기본적인 인공신경망 구조 중 하나는 1957년 Frank Rosenblatt가 제안한 퍼셉트론(Perceptron)이다.

$$X = [x_1, x_2, \dots, x_n], W = [w_1, w_2, \dots, w_n] \quad (2.10)$$

$$\begin{aligned} f &: \text{Activation function} \\ y &= f(W \cdot X + b) \end{aligned} \quad (2.11)$$

그림 2.7에서는 단층 퍼셉트론(Single-layer perceptron) 모델 구조를 보여주고 있다. 식 (2.10)와 (2.11)에 의해 인공 뉴런 혹은 노드는 n 개의 입력 X 를 받고 이에 대한 가중치 W 를 곱해 합산한 후 상수인 바이어스 항 b 를 더한다. 그 후 활성화 함수 f 에 의해 출력 y 가 결정된다.

많은 사람들이 퍼셉트론과 같은 인공신경망의 등장에 대해 관심과 기대를 많이 가졌지만 1968년 마빈 민스키(Marvin Minsky)와 시모어 페퍼트(Seymour Papert)가 퍼셉트론의 한계를 수학적으로 증명함으로써 결국 많은 연구자들이 포기하였다. 바로 퍼셉트론은 단순한 선형 분류기이기 때문에 배타적 논리합(XOR)과 같은 선형분리가 불가능한 문제들을 해결할 수 없다는 것이었다.

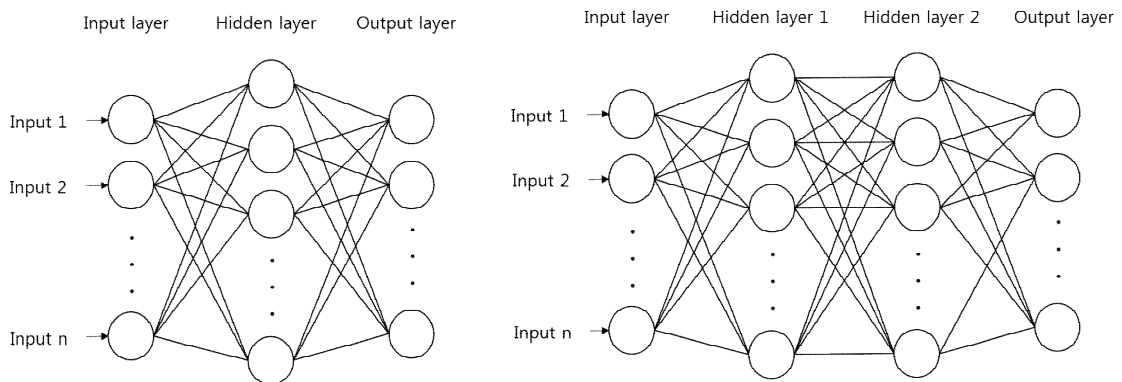


그림 2.8. 다층 퍼셉트론과 심층 신경망

Fig 2.8. Multi-layer perceptron and deep neural network

하지만 기존의 퍼셉트론의 한계를 극복한 다층 퍼셉트론(MLP, Multi-Layer Perceptron)의 등장과 함께 다시 인공신경망이 주목받기 시작하였다. 다층 퍼셉트론은 층을 여러 개 쌓은 구조로 그림 2.8의 좌측과 같이 입력층(Input layer), 은닉층

(Hidden layer), 출력층(Output layer)로 구성되어있다. 또한 그림 2.8의 우측에 나타난 것과 같이 은닉층이 2개 이상의 신경망 구조를 가질 경우 심층 신경망(DNN, Deep Neural Network)라고 통칭한다.

다층 퍼셉트론의 기존 문제해결로 인해 층을 늘려 더 깊은 구조를 만들면 복잡한 문제를 풀 수 있을 것이라고 기대하였지만 다층 퍼셉트론은 층이 복잡해질수록 학습이 제대로 되지 않는 문제가 발생하였다. 출력층은 목표값 혹은 레이블이 있지만 중간층의 은닉층의 경우는 정확히 어떤 값을 가져야한다는 정답이 없기 때문에 학습이 이루어지지 않는 것이었다. 이러한 문제는 역전파(Back propagation) 알고리즘의 도입으로 해결되었다. 역전파 알고리즘은 전방향(Forward)으로 연산하여 출력을 얻은 후 각 출력의 오차를 계산한다. 그리고 이 방법을 신경망의 역방향(Backward)으로 미분과 연쇄 법칙(Chain rule)을 수행하여 오차에 기여한 정도를 측정하고 오차가 줄도록 가중치를 업데이트한다[26].

하지만 이러한 역전파 알고리즘도 신경망 구조에서 층이 늘어날수록 역전파가 제대로 작동하지 않는 Vanishing Gradient 문제가 발생한다. 이러한 문제는 ReLU와 같은 새로운 활성화 함수의 활용으로 인해 해결되었다. 그림 2.9는 ReLU를 포함한 다양한 신경망에서 사용되고 있는 활성화 함수(Sigmoid, tanh, ReLU)들을 보여준다.

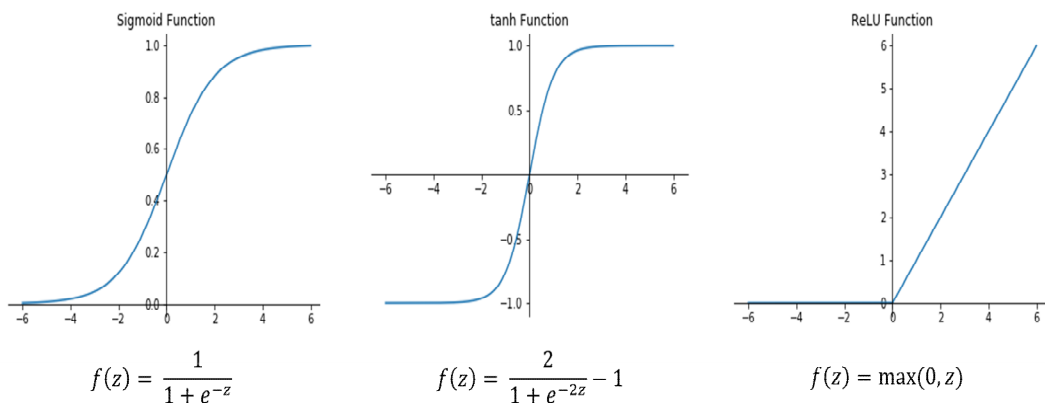


그림 2.9. 활성화 함수

Fig 2.9. Activation function

이외에도 학습도중 일부노드를 비활성화 시키는 드롭아웃(Dropout), 여러 모델의 예측 결과를 종합하여 판단하는 앙상블(Ensemble), 전체 학습 데이터를 배치 사이즈로 등분하여 속도와 정확도를 높이는 미니 배치(Mini batch) 등 다양한 기법들의 발전으로 인한 딥 신경망의 정확도를 높였으며, 이를 기반으로 이미지 인식, 영상 인식, 음성 인식 등 다양한 영역에서 활용되고 있다.

2.1.4. 오토인코더(Auto-Encoder)

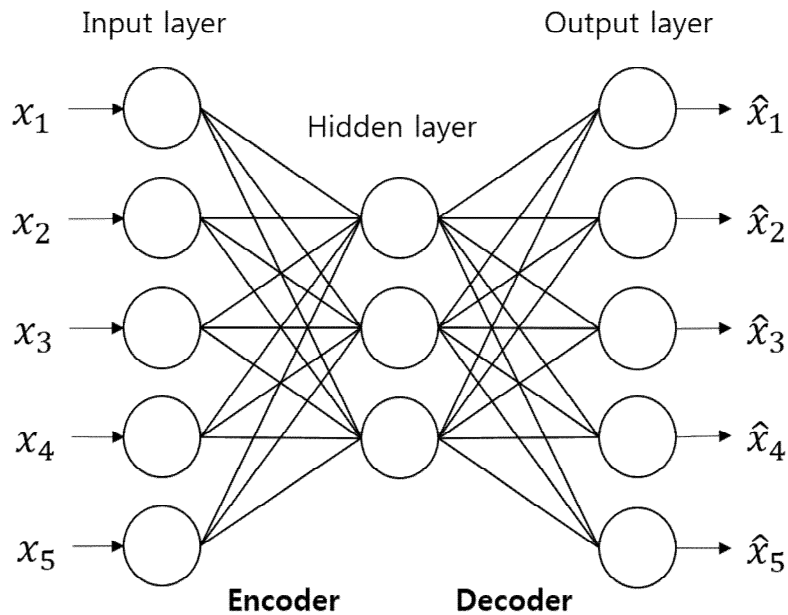


그림 2.10. 오토인코더의 기본 동작 구조

Fig 2.10. Basic operation structure of auto-encoder

오토인코더는 머신러닝에서 레이블 정보 없이 동작하는 비지도 학습 방법 중 하나로, 출력을 입력 데이터와 같게 하는 것을 목적으로 학습한다[21]. 오토인코더의 구조는 인코더(Encoder)와 디코더(Decoder)가 대칭적으로 연결된 신경망 구조로 그림 2.10에서 보여준다.

입력 데이터는 인코더 구간을 통과하면서 데이터의 차원이 축소되는데 인코더로부터 축소된 데이터를 데이터의 압축된 표현 혹은 특징이라고 한다. 디코더 구간에

서는 복원 과정을 거쳐 입력 데이터와 동일한 값을 반환한다. 그리고 디코더의 복원된 데이터 출력값과 입력 데이터의 차이를 줄여가면서 오토인코더를 학습시킨다. 오토인코더의 학습을 위한 손실함수로는 일반적으로 평균 제곱 오차가 사용되며, 역전파 방법과 경사 하강법 등을 이용하여 학습한다.

오토인코더에서는 일반적으로 은닉층의 노드 개수가 입력층의 수보다 적은 구조를 가지고 있다. 이러한 구조로 인해 입력된 데이터는 저차원으로 압축되는 효과를 가지게 된다. 이는 입력 데이터에서 불필요한 정보를 버리고 필요한 정보만을 추출하는 과정이라고 볼 수 있다. 그러므로 입력 데이터의 상태에 따라 저차원 압축을 통해 추출한 특징이 다를 수 있고 출력되는 복원 데이터 또한 변할 수 있다.

오토인코더는 주로 데이터의 차원 축소, 특징 추출, 노이즈 제거, 데이터 복원을 통한 이상감지 등 여러 분야에서 연구되어 지고 있다. 본 논문에서는 센서 데이터의 특징 추출을 위해 오토인코더가 사용되었다.

2.1.5. 변분 오토인코더(Variational Auto-Encoder)

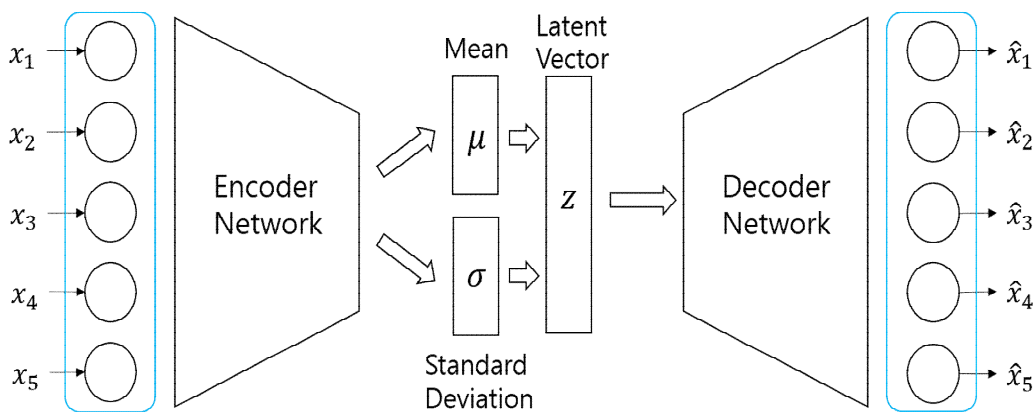


그림 2.11. 변분 오토인코더 네트워크

Fig 2.11. Variational auto-encoder network

변분 오토인코더(Variational Auto-Encoder, VAE)는 오토인코더의 종류중 하나로

오토인코더와 마찬가지로 인코더와 디코더의 구조를 가지며, 입력과 동일한 출력값을 얻기 위해 학습된다. 하지만 단순히 동일한 입출력 데이터의 결과를 원하는 오토인코더와는 달리 VAE는 데이터 자체의 확률분포를 학습한다. VAE의 경우 데이터 압축을 표현하는 잠재변수(Latent variable)를 입출력 데이터와의 매개변수로 설정한다. 이로 인해 오토인코더에서는 없었던 입력 데이터와 잠재변수의 분포 간의 상관관계를 높일 수 있다. 그림 2.11은 VAE의 기본 구조를 보여주고 있다[27].

VAE의 인코더를 통해서 고차원 입력 데이터는 데이터의 분포를 표현하는 잠재변수의 평균(Mean)과 표준편차(Standard deviation)을 출력으로 얻을 수 있다. 이 평균과 분산을 가지는 가우시안 확률분포로부터 잠재변수를 샘플링하고 디코더에서는 이 잠재변수를 기반으로 데이터를 재구성한다.

VAE 학습에서는 오토인코더 학습과정에서 고려되는 복원 오차(Reconstruction error) 뿐만 아니라 디코더에서 데이터 재구성을 위해 사용되는 잠재변수의 확률분포가 이상적인 확률분포에 가까워지도록 학습한다. 수식 (2.12)은 VAE를 학습시키기 위한 손실함수를 정의하였다[27].

$$Loss\ function = KL(q_{\phi}(z|x_i)|p(z)) - E_{q_{\phi}(z|x_i)}[\log(x_i|g_{\theta}(z))] \quad (2.12)$$

위 식에서 ϕ 와 θ 는 각각 인코더와 디코더의 파라미터이며, x_i 는 입력 데이터, $q_{\phi}(z|x_i)$ 는 잠재변수의 확률분포, $p(z)$ 는 이상적인 확률분포, $g_{\theta}(z)$ 는 디코더의 출력인 복원된 데이터를 의미한다. 손실함수의 왼쪽 항은 확률분포 사이의 거리를 계산하는 Kullback Leibler Divergence를 사용하여 잠재변수의 확률분포가 목표분포와 유사하게 한다는 조건을 부여한다. 오른쪽 항은 입력 데이터와 출력 데이터 간의 오차를 줄이기 위해 Cross Entropy를 사용한다.

2.1.6. 컨볼루션 신경망(Convolution Neural Network)

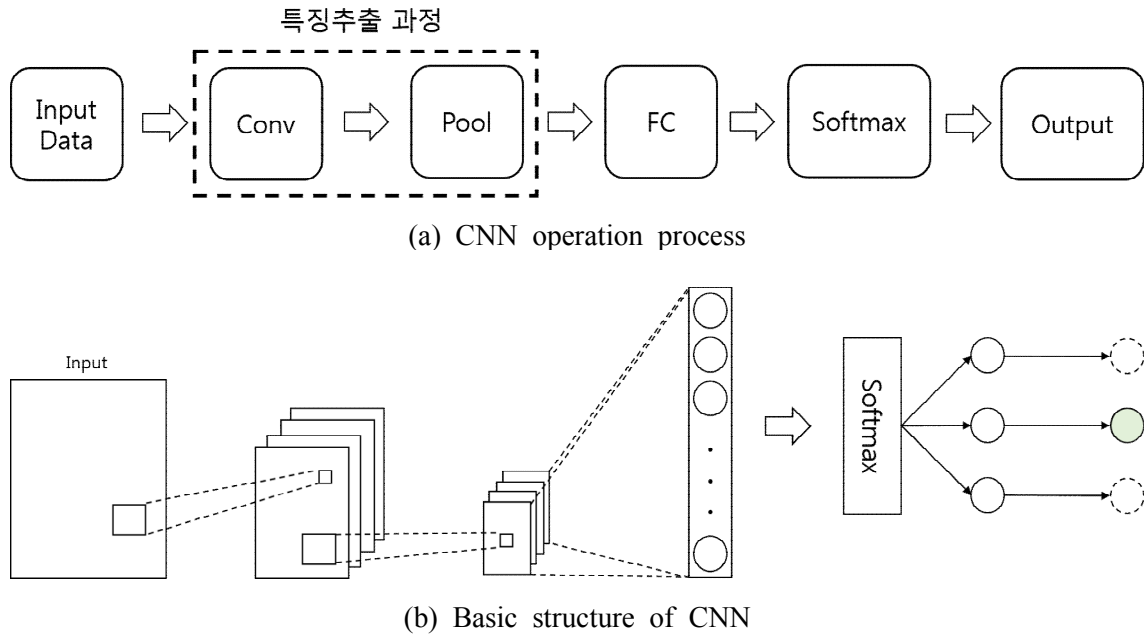


그림 2.12. 컨볼루션 신경망 동작과정 및 구조

Fig 2.12. Convolution neural network operation process and structure

컨볼루션 신경망(Convolution Neural Network, CNN)은 컨볼루션(Convolution) 연산을 수행하는 인공 신경망을 여러 층 쌓아 올린 딥 러닝 기법중 하나이다[18]-[19]. CNN은 시신경의 시각 정보 처리 과정을 모방하여 설계되었으며, 특히 이미지 인식 분야에서 우수한 성능을 보여주고 있다. 2012년 ‘컴퓨터 비전 분야의 올림픽’이라고 불리는 ILSVRC(ImageNet Large-Scale Visual Recognition Challenge)에서 CNN 구조를 사용한 AlexNet이 약 오차율(16%)로 2위(26.1%)를 제치고 1위를 기록하였다[28]. 이후 대회에서도 CNN구조의 신경망이 대회 우승을 차지하면서 그 성능을 입증하였다. 최근에는 이미지 처리 분야에 국한되지 않고 자연어처리, 음성 인식 등 다양한 분야에 적용되어 활발하게 연구되고 있다.

CNN은 컨볼루션층(Convolution layer), 풀링층(Pooling layer) 그리고 완전 연결 층(fully connected layer) 세 가지 계층으로 구성된다. 그림 2.12는 기본적인 CNN 동작

과정과 구조를 나타내고 있다.

Convolution layer는 입력 데이터로부터 특징을 추출하기 위해 사용된다. 가중치와 편향을 할당한 윈도우 크기만큼 Convolution 연산 과정을 통해 계산한 결과를 얻는다. 이때 윈도우를 필터(Filter) 혹은 커널(Kernel)이라고 한다. 일반적으로 CNN은 이미지 처리를 위해 주로 사용되기 때문에 2차원 convolution 연산이 많이 사용되는데 식 (2.13)와 같이 나타낼 수 있다.

$$(f * g)[n, m] = \sum_{j, k} f[j, k] g[n - j, m - k] \quad (2.13)$$

f 는 2차원 입력 데이터, g 는 필터를 의미한다.

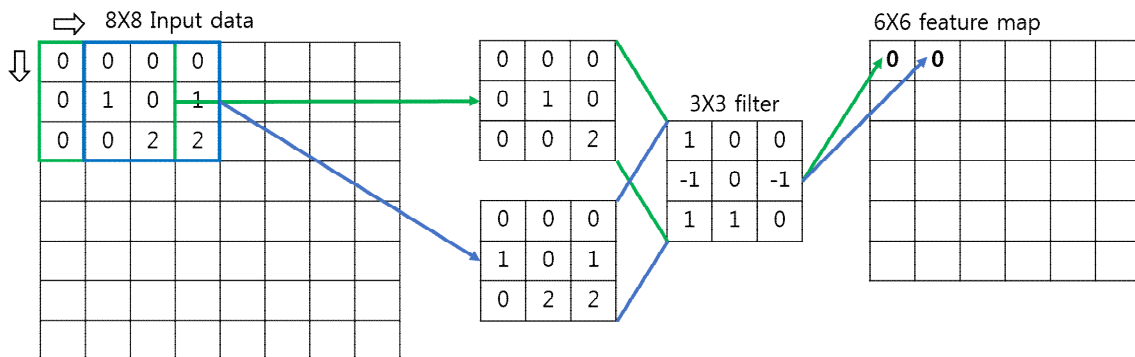


그림 2.13. 컨볼루션층 동작과정

Fig 2.13. Convolution layer operation process

그림 2.13은 2차원 convolution 연산 과정을 보여주고 있다. Convolution 단계에서는 필터의 크기와 이동간격(Stride)을 설정하는데 일반적으로 필터는 $N \times N$ 의 크기를 사용하며, stride의 경우 실험적으로 계산량과 시간을 고려하여 설정된다. 그림 2.13에서는 필터는 3×3 의 크기를 stride는 1로 설정한 것을 볼 수 있다. Convolution 연산을 적용한 뒤 결과를 특징 맵(feature map)이라고 하는데 feature map의 크기는 8×8 에서 6×6 으로 작아진 것을 볼 수 있다. 이는 feature map의 크기는 convolution 연산

에 적용되는 필터의 크기, stride에 달라진다는 것을 의미한다. 그림 2.13과 같은 경우 이미지의 모서리 부분은 계산이 되지 않아 완전한 특징 추출이 어렵다. 따라서 모서리 부분까지 고려할 수 있도록 padding을 추가하여 convolution을 수행할 수도 있으며, 이 경우 feature map을 적당한 크기로 유지할 수 있다. 필터에 할당되는 초기 가중치는 무작위로 설정되며 학습을 통해 최적의 값을 가지게 된다. 필터를 통해 추출된 feature map은 ReLU, Sigmoid 등의 activation function를 적용시킨 후 pooling layer를 거친다.

Pooling layer에서는 feature map에서 sub-sampling을 수행한다. 이 과정을 통해서 feature map의 크기가 줄어들어 그 만큼의 정보를 잃어버리는 것이라고 할 수 있지만 sub-sampling을 통해 feature map에서 대표 특징 값을 추출할 수 있으며, feature map의 크기가 축소되므로 메모리 효율과 속도 향상에 기여한다. 대표적으로 사용되는 방법은 max pooling과 average pooling이 있다. 그림 2.14는 필터 범위 안에서 가장 큰 값을 선택하는 max pooling과정을 보여준다.

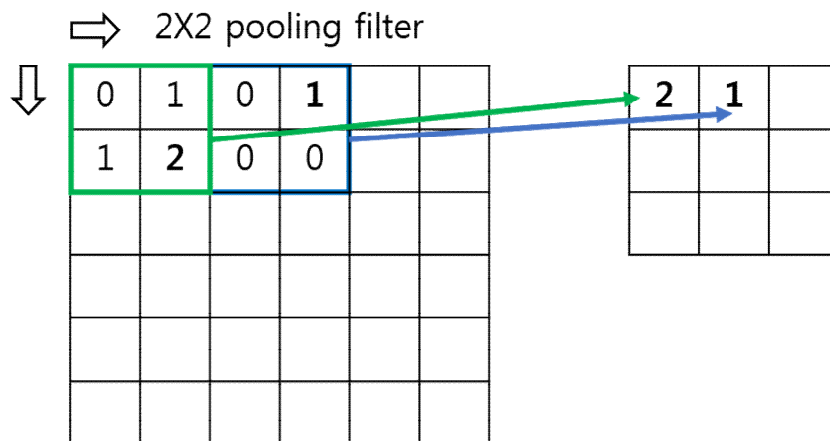


그림 2.14. 풀링층 동작과정

Fig 2.14. Pooling layer operation process

Fully connected layer에서는 모든 노드들이 각 층의 모든 노드들과 서로 연결되어 있으며 pooling layer에서 얻은 특징들을 입력으로 사용하여 내적 연산을 수행한다.

Softmax에서는 다중 클래스를 분류 문제의 경우 출력값을 확률적으로 표현하기 위한 방법으로 수식 (2.14)과 같이 정의된다[19]. Softmax의 입력값을 0~1사이의 출력값으로 정규화하며 확률을 표현하기 때문에 총합은 1이 되도록 만든다. 가장 높은 확률 값을 가진 출력을 해당 클래스로 판별하며, 원-핫 인코딩(one-hot encoding)을 통해 가장 큰 값은 1, 나머지는 0으로 출력한다.

$$p_j = \frac{e^{s_j}}{\sum_i e^{s_i}} \quad (2.14)$$

CNN을 통해서 원하는 값을 얻기 위해서는 위에서 언급한 filter크기, filter개수, stride, padding, pooling 이외에도 learning rate, epoch 등 하이퍼파라미터(Hyper-parameter) 값들을 설정해 주어야한다.

2.2. 유전 알고리즘

2.2.1. 유전 알고리즘 기본개념 및 연산과정

유전 알고리즘은 자연세계의 생물학적 진화과정을 기반으로 시뮬레이션하여 주어진 설계영역 안에서 최적의 해를 찾는 방법으로 존 홀랜드에 의해 고안되었다[29]. 유전 알고리즘은 진화가 일어나는 유전 과정을 모방하며 유성 생식과 적자생존의 법칙을 기반으로 모델링한다. 유성 생식은 두 부모의 유전자 정보를 자손에게 전달되는 과정이고 적자생존은 개체 집단에서 다양한 환경에 잘 적응한 우수한 개체들만 살아남는 다는 것이다. 다시 말해서, 세대가 거듭 될수록 환경에 적합한 개체는 살아남아 자식 개체를 남기며 다시 자식 개체는 자신의 유전 정보를 다음 세대로 전달하면서 진화하는 것이다. 유전 알고리즘에서는 주어진 환경에 얼마나 적합한지를 나타내는 적합도(Fitness)를 염색체마다 부여하며, 적합도 결과에 따라 그림 2.15에 나타낸 것처럼 크게 선택(Selection), 교차(Crossover), 변이(Mutation) 그리고

대치(Replacement)연산 순서로 진행된다.

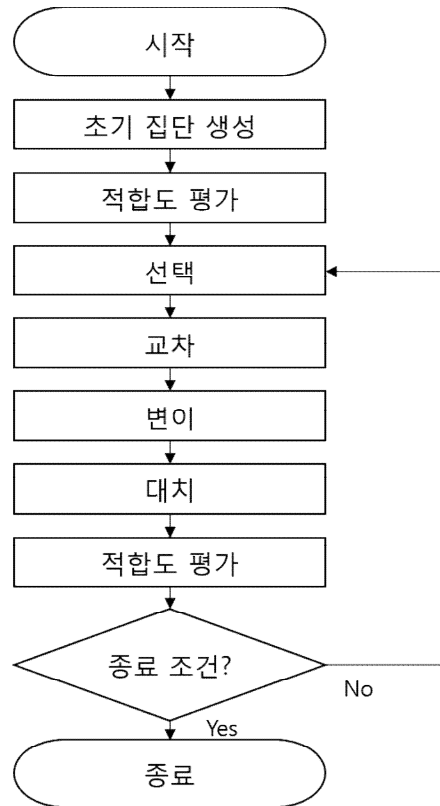


그림 2.15. 유전 알고리즘 구현

Fig 2.15. Genetic algorithm implementation

초기 해의 생성 단계에서는 염색체 집단을 형성한다. 염색체(Chromosome)는 유전 정보를 담고 있는 생물학적인 집합을 연속된 문자열로 추상화시킨 것이며, 염색체는 유전자(Gene)라는 요소로 이루어져 있다. 염색체 개체의 유전자는 문제를 효과적으로 풀어내기 위해 코드화 시키는데 일반적으로 0과 1의 2진 비트코드로 표현되며, 염색체는 비트 스트링 형태가 된다. 염색체의 유전자가 1로 표현이 되면 해당 유전 정보가 사용되는 것이고 0이면 해당 유전자는 사용되지 않는 것을 의미한다. 적합도 평가에서는 수식으로 표현된 적합도 함수에 의해 각 염색체는 유전 정보에 따라 적합도 점수가 부여된다. 예를 들어 염색체가 [0, 1, 0, ... , 1]과 같이 비트 스트링 형태로 표현된다면 1로 표현된 두 번째와 마지막의 유전 정보가 적합도를 고

려할 때 사용되며 나머지 0으로 표현된 것들은 사용되지 않는다. 선택과정에서는 큰 적합도 점수를 보유한 염색체 개체일수록 높은 확률로 선택되고 교차와 변이 단계를 거쳐 부모의 유전 정보를 보유한 자식 개체들을 형성하게 된다. 대치과정에서는 다음 세대로 가기 위해 부모해의 집단을 자식 해로 교체한다. 이와 같은 과정이 반복적으로 진행되면서 전체적인 해집단의 품질은 향상되며 최종적으로 종료 조건에 부합되는 최적의 해를 얻을 때까지 연산과정을 반복한다.

유전 알고리즘은 미분값을 이용하는 Gradient기반 최적화 알고리즘과는 달리 적합도 함수를 활용하여 최적 해를 탐색하기 때문에 다양한 응용이 가능하며, 설계영역에 전체에 분포하는 개체들을 사용함으로써 전역최적화를 보장한다[30].

2.2.2. 적합도 함수(Fitness function)

유전 알고리즘을 통해서 얻은 해가 최적의 값이라고 판단하는 기준은 오직 적합도 함수에 따라 부여되는 적합도 점수에 의해 결정된다. 적합도 함수의 설계에 따라 유전 알고리즘 자체 속도나 성능이 달라질 수 있다. 그렇기 때문에 유전 알고리즘에서 적합도 함수를 적절하게 설계하는 것이 가장 중요하다. 적합도 함수는 다음과 같이 식으로 나타낼 수 있다.

$$f(x) = \max\{o(x) - ap(x)\} \quad (2.15)$$

위의 식에서 $o(x)$ 는 목적함수, a 는 벌점계수, $p(x)$ 는 벌점함수 그리고 $f(x)$ 는 적합도 함수를 의미하며 적합도 함수의 값을 최대로 하는 것이 최적 값을 의미한다[31]. 유전 알고리즘에서 개체가 적합도 함수가 단순히 목적함수에 의해서만 평가된다면 매우 높은 적합도 점수를 가진 개체는 다음 세대로 전달된다. 하지만 높은 적합도 점수를 가짐에도 불구하고 그 개체가 최적화 문제에서 요구되는 해의 조건에 맞지 않는다면 원하는 해답이 될 수 없으므로 다음세대에서 해당 개체의 생존 확률을 낮출 필요가 있다. 따라서 목적함수 이외 추가적으로 벌점함수를 부여하여 조건에 위

배되지 않는 방향으로 해를 개선해야한다. 별점계수는 적합도 함수에 별점함수를 얼마나 반영할 것인가를 정할 수 있다. 별점계수의 값이 클수록 적합도 함수가 별점함수에게 많이 영향을 받으며 높은 목적함수를 가지는 개체라도 조건에 벗어나면 큰 별점을 부여하여 다음 세대에 포함될 가능성은 낮아진다. 별점계수를 작게 설정한다면 조건에 많이 벗어나더라도 높은 적합도 점수를 가진 개체는 다음 세대에 포함될 확률은 높아진다.

2.2.3. 선택연산(Selection)



그림 2.16 선택연산 과정

Fig 2.16. Selection process

선택연산에서는 다음 세대인 자식 개체들을 생성하기 위해 현재 세대의 개체 집단 내에서 대표되는 부모개체를 선택하는 과정으로 새로운 세대의 진화에 중요한 영향을 미친다. 선택된 부모개체들의 유전 정보를 자식 대체들이 물려받기 때문에 일반적으로 선택은 적합도 평가를 통해 비교적 높은 적합도 점수를 가진 상위 염색체들을 확률적으로 선택하여 다음 세대에 사용한다. 하지만 무조건 좋은 적합도를 가진 개체들만 선택한다면 특정 유전 정보들을 가진 개체들만 살아남아 일정 수준에서 더 이상 적합도가 개선되지 않는 경우가 발생한다. 따라서 유전적 다양성을

지키기 위해 낮은 적합도를 가진 개체들도 확률적으로 선택가능하게 하거나 적합성에 비례하여 선택하는 방법들을 고려할 수 있다. 대표적인 선택 방법으로는 룰렛 휠 선택(Roulette wheel), 토너먼트 선택(Tournament selection), 순위에 기초한 선택(Rank-based selection) 등이 폭넓게 사용되고 있다[18], [30].

2.2.4. 교차연산(Crossover)

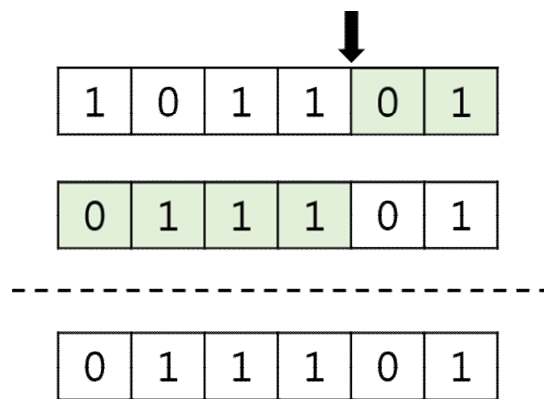


그림 2.17. 단일 점 교차 방식

Fig 2.17. One point crossover

교차연산은 부모 염색체로부터 각각 유전 정보를 물려받는 유성 생식을 모방한 방법이며, 선택과정에서 선택된 두 개의 염색체로부터 부분적으로 서로 교차하여 자식개체를 생성한다. 교차연산을 통해 높은 적합도를 가진 부모 개체는 자식 개체로 우수한 유전자를 전달하여 다음 세대로의 생존 확률을 높이며 우수한 유전자를 계속해서 보존하고 평균적으로 개체 집단의 적합도를 높일 수 있다.

교차 방식 중 가장 일반적인 방법은 무작위로 비트 스트링에서 교차(Crossover)지점을 선택하고 교차지점을 기점으로 교차지점 전까지는 첫 번째 부모로부터 유전자를 상속받고 나머지는 두 번째 부모로부터 유전자를 받는 것이다. 이를 단일 점 교차방식(One-point crossover)이라고 하며 그림 2.17에 나타내고 있다. 단일 점 교차 방식 이외에도 여러 개의 교차지점을 사용하는 다점 교차방식(Multi-point crossover), 각 유전 요소를 무작위로 물려받는 균등교차방식(Uniform crossover), 실수 염색체를

사용하는 경우 두 부모 염색체 요소의 평균값을 이용하는 산술교차방식(Arithmetic crossover) 등이 있다[32].

2.2.5. 변이연산(Mutation)

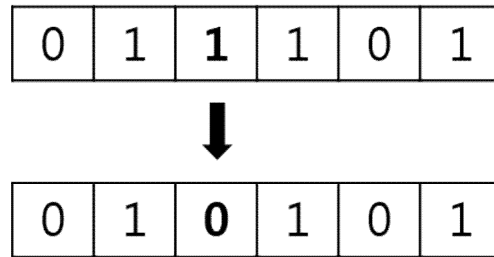


그림 2.18. 유전적 변이효과

Fig 2.18. Genetic mutation effect

유전 알고리즘에서 세대가 거듭될수록 선택과 교차연산에 의해 부모와 자식 개체의 유전 정보가 유사해지는데 이런 경우 해의 다양성은 없어지고 적합도는 어느 수준에 이르면 개선되지 못하고 수렴하게 되어 최적의 해를 얻지 못하는 경우가 발생한다. 이를 해결하기 위해 염색체의 변이를 통해 부모해가 가지지 못한 성질을 부여하여 다양성을 유지시키고 탐색범위를 확대함으로써 전역적 탐색효과를 극대화시킨다. 변이는 그림 2.18에 나타낸 것과 같이 각 염색체 혹은 유전자에 대해 미리 설정된 확률에 의해 임의의 위치에서 0을 1로 혹은 1을 0으로 비트 반전시킨다. 변이에는 확률을 일정하게 고정하여 발생시키는 정적 변이방법과 변이의 강도를 해의 품질에 따라 변화시키는 동적 변이방법이 사용된다.

변이의 발생확률을 낮게 설정했을 경우, 세대가 거듭될수록 해의 다양성은 사라지고 탐색범위는 제한되어 원치 않는 해에 수렴하게 된다. 반면에 변이의 확률이 높아지면, 다양한 해의 생성으로 인한 유전 알고리즘의 탐색범위를 확대할 수 있지만 자주 발생하는 유전자 변형에 의해 방향성을 상실하여 수렴성이 떨어져 알고리즘 수행 시간이 늘어날 수 있다.

2.2.6. 대치연산(Replacement)

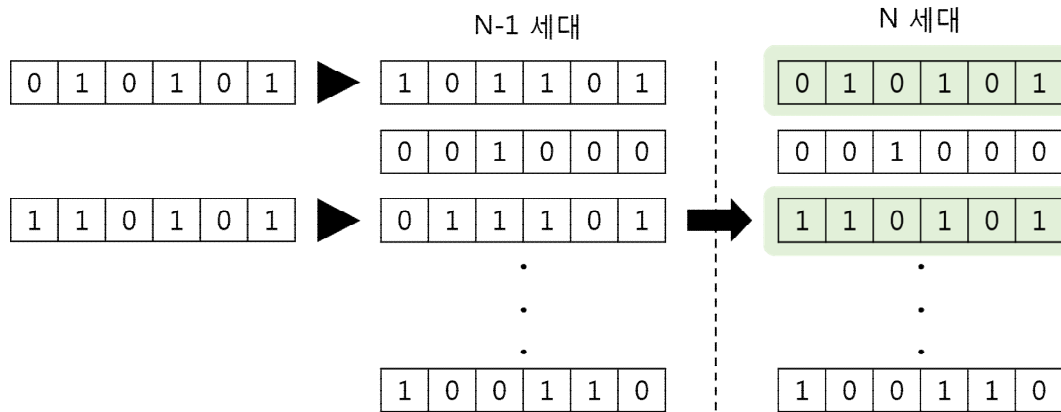


그림 2.19. 대치과정 예제

Fig 2.19. Example of replacement process

유전 알고리즘에서 부모 해는 자식 해를 만들어 새로운 해집단을 생성하고 교체를 통해 다음 세대로 넘어가게 된다. 하지만 단순히 부모 해 모두를 새로운 자식 개체로 교체한다면 부모 세대에서 최적의 적합도를 가진 개체가 다음 세대에는 소멸하게 된다. 그럴 경우 각 세대의 최대 적합도 값이 줄어들고 전체적인 해집단 품질이 저하되어 최적 해의 탐색이 침체되는 경우가 발생할 수 있다. 따라서 이를 막기 위해 엘리트전략(Elitism), 토너먼트방법(Tournament) 같은 방법들이 사용된다.

엘리트전략은 현세대에서 우수한 해들을 다음 세대에 보존하는 방법으로 자식 해집단에서 무작위로 선택하거나 낮은 품질을 가진 해를 선택하여 교체한다. 토너먼트 방식은 두 부모 객체와 자식 객체를 임의로 선택하여 서로의 적합도를 비교한 후 넷 중 높은 적합도 점수를 가진 개체들을 새로운 세대로 전달한다. 두 전략을 사용한다면 우수한 해를 계속 유지시킬 수 있지만, 초기 선택된 우수한 해들이 계속해서 유지될 경우 유전 알고리즘이 진행되면서 점점 유사한 해들이 생성되어 다양성을 잃게 되고 조기 수렴하게 된다. 이럴 경우 최적화 값을 찾을 수 없기 때문에 후보 해의 개수 설정이나 선택을 확률적으로 설계하는 방법들을 고려해야 한다.

3. 센서 고장 신호 유형

본 연구에서는 센서 고장 검출 및 분류를 위해 참고문헌[17]에서 활용된 데이터를 사용하였다. 데이터는 TC1047/TC1047A Precision Temperature-to-Voltage Converter로부터 측정 되었으며, 정상 신호(Normal)와 5가지 유형의 센서 고장 신호들을 사용하였다[17]. 표 3.1에서는 상기 5가지 고장유형들을 정의한다.

표 3.1. 센서 데이터 고장 유형 및 특징

Table 3.1. Sensor data fault type and characteristics

Fault type	Fault description
1 : Erratic fault	센서 출력에 노이즈가 급격히 증가함
2 : Drift fault	센서 출력이 계속해서 감소하거나 증가하는 현상
3 : Hard-over fault	센서 출력이 한계치를 넘어서 biased되는 현상
4 : Spike fault	센서 출력이 일정 간격 갑작스럽게 spike되는 현상
5 : Stuck fault	센서로부터 고정된 출력이 나오는 현상

센서의 고장 신호와 표 3.1에서 정의한 erratic, drift, hard-over, spike, stuck 고장유형 5가지를 동시에 출력한 상태를 그림 3.1에서 보여준다.

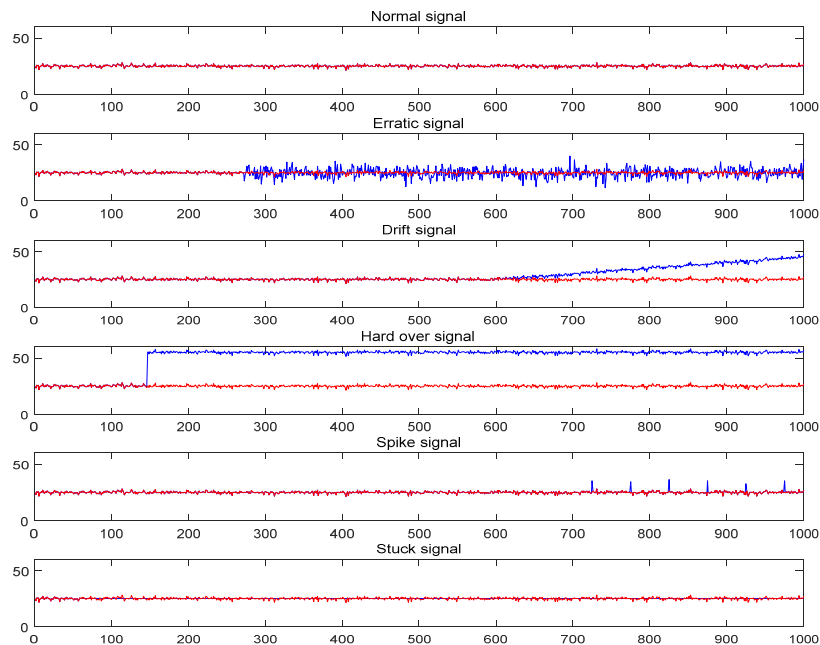


그림 3.1. 정상 신호와 각 유형별 신호 비교

Fig 3.1. Comparison of normal signal and each signal type

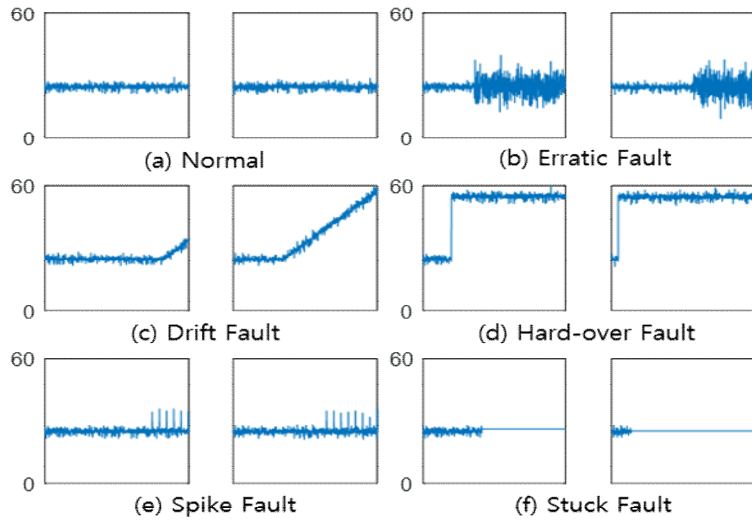


그림 3.2. 무작위 지점에서 발생한 각 고장에 대한 데이터 샘플들

Fig 3.2. Data samples for each class that occurred at random points

본 논문에서 사용된 데이터 셋은 정상 신호와 상기 5개의 고장 유형에 대해 각각 100개의 데이터 샘플로 구성되었다. 따라서 총 600개의 데이터 샘플들이 사용되었으며, 각 데이터 샘플들은 1,000개의 데이터 포인트를 가지고 있다. 각 고장데이터 샘플들은 그림 3.2에 나타낸 것과 같이 데이터 포인트 0부터 1,000사이에 무작위로 고장이 발생한 것을 볼 수 있다. 실질적으로 센서 고장이 어느 지점에서 발생하는지는 미지수이기 때문에 고장 발생지점이 어디서 발생하든 빠르게 고장의 패턴을 찾아 감지하는 것이 중요한 문제이다.

본 논문에서는 딥 러닝 및 SVM의 학습과 테스트를 위해 고장 유형별로 40개의 데이터 샘플들은 학습에 사용되고 나머지 60개는 테스트에 사용되었다.

4. SVM 기반 센서 고장 분류

4.1. 센서 고장 검출 및 분류를 위한 특징추출

머신러닝을 사용한 분류기의 성능은 설계된 알고리즘 성능뿐만 아니라 학습에 사용되는 데이터의 영향을 많이 받는다. 원래의 데이터를 그대로 사용하여 지능형 분류기를 학습시킬 경우 데이터 수에 의해 계산량이 너무 많거나 데이터 자체로는 의미가 없는 경우가 있다. 따라서 성능을 높이기 위해 데이터로부터 특성을 잘 반영하는 요인을 새로 정의한 뒤 이를 활용하여 학습시키는 것이 필요하다.

데이터 특징 추출에 많이 사용되는 방법 중 하나는 통계학적 특징들을 사용하는 것이다[12]-[13],[33]. 참고문헌 [12]-[13]에서는 데이터로부터 시간 영역에서의 통계적 특징뿐만 아니라 주파수 상에서 통계적 특징을 추출하여 머신러닝 알고리즘의 일종인 KNN, SVM을 학습시켜 우수한 분류 성능을 보여주고 있다. 또한 참고문헌 [13]에서는 웨이블릿 패킷(Wavelet Package), [34]에서는 Principal Component Analysis(PCA), Linear Discriminant Analysis(LDA)와 같은 방법들을 통해 얻은 특징들을 머신러닝 알고리즘에 적용시켜 효율적으로 성능을 높였다.

본 논문에서는 SVM의 효율적인 학습을 위해 센서 데이터로부터 딥 러닝 및 통계 기반의 특징을 추출한다.

4.1.1. 시간 영역 통계적 특징

본 논문에서는 센서 고장 분류를 위해 센서 데이터로부터 시간영역 통계적 특징을 추출하였다. 시간영역 통계적 특징들은 여러 연구들을 통해 머신러닝 분류문제에서 좋은 결과들을 보여주었다[12]-[13].

그림 4.1은 시간 영역 통계적 특징을 사용한 SVM 센서 고장 진단 순서도를 보여준다. 센서 데이터를 입력 받아 센서 데이터로부터 statistical features를 추출한 뒤 학습된 SVM 모델에 의해 센서의 고장을 판별한다.

표 4.1은 SVM 학습에 사용된 시간영역 통계 특징들을 나타내며 다음과 같다.

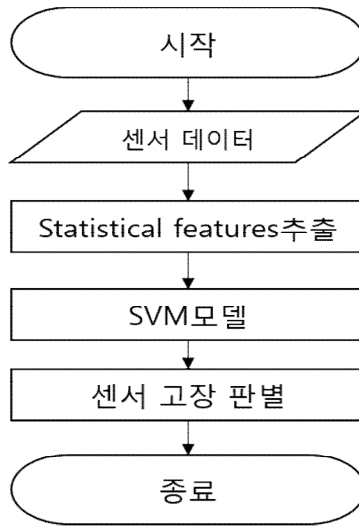


그림 4.1. 시간 영역 통계적 특징 추출과 SVM을 사용한 센서 고장진단 흐름도

Fig 4.1. Flowchart of sensor fault diagnosis using time-domain statistical feature extraction and SVM

표 4.1. SVM에서 사용된 N 개의 데이터 포인트 x_k 로부터 추출된 시간 영역 통계적 특징

Table 4.1. The time-domain statistical features extracted from the N data points x_k used in the SVM

$K_{rms} = \left(\frac{1}{N} \sum_{k=1}^N x_k^2\right)^{1/2}$	$K_{if} = \frac{\max(x_k)}{\frac{1}{N} \sum_{k=1}^N x_k }$	$K_{mf} = \frac{\max(x_k)}{K_{sra}}$	$K_{kf} = \frac{K_{kv}}{K_{rms}^4}$
$K_{kv} = \frac{1}{N} \sum_{k=1}^N \left(\frac{x_k - m}{\sigma}\right)^4$	$K_{sf} = \frac{\max(x_k)}{\left(\frac{1}{N} \sum_{k=1}^N x_k \right)^{1/2}}$	$K_{sv} = \frac{1}{N} \sum_{k=1}^N \left(\frac{x_k - m}{\sigma}\right)^3$	$K_{mean} = \left(\frac{1}{N} \sum_{k=1}^N x_k\right)$
$K_{ppv} = \max(x_k) - \min(x_k)$	$K_{sra} = \left(\frac{1}{N} \sum_{k=1}^N \sqrt{ x_k }\right)^2$	$K_{cf} = \frac{\max(x_k)}{K_{rms}}$	$K_{cm} = \frac{\frac{1}{N} \sum_{k=1}^N (x_k - m)^n}{\sigma^n}$

RMS(root mean square), KV(kurtosis value), PPV(peak-to-peak value), IF(impulse factor), SF(shape factor), SRA(square root of the amplitude), MF(margin factor), SV(skewness value), CF(crest factor), KF(kurtosis factor), Mean, CM(central moment, $n=5$)[12]-[13]. 위에서 언급한 RMS부터 CM까지 12가지 특징에 대해 차례대로 1부터 12까지 번호를 부여하여 특징을 구분하는데 사용하였다.

4.1.2. 오토인코더 이상치 기반 통계적 특징

오토인코더의 경우 출력 결과가 입력 데이터와 동일하도록 학습시키는데 이 과정은 학습 데이터의 패턴을 학습하는 것을 의미한다. 테스트 데이터가 학습 데이터와 동일한 패턴을 보일 경우 복원이 제대로 이루어져 입력 데이터와 출력 데이터의 차이가 거의 없다. 반면에 학습되지 않은 새로운 데이터에 대해서는 복원이 제대로 이루어지지 않아 입출력 간의 차이가 크고 그 만큼 정보를 잃어버리게 된다. 따라서 오토인코더는 특징 추출 이외에도 이러한 복원 값의 차이를 이용하여 비정상 신호 탐지 분야에 사용될 수 있다[21].

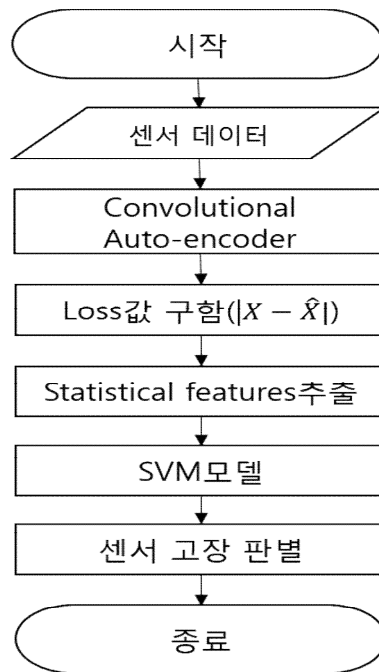
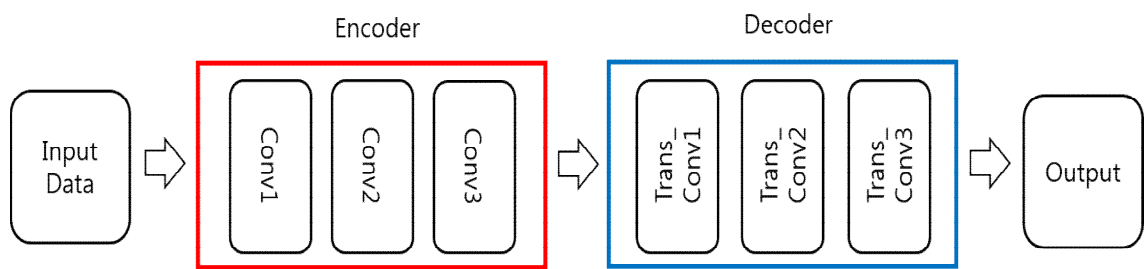


그림 4.2. 컨볼루셔널 오토인코더 기반의 특징 추출과 SVM을 사용한 센서 고장진단 순서도

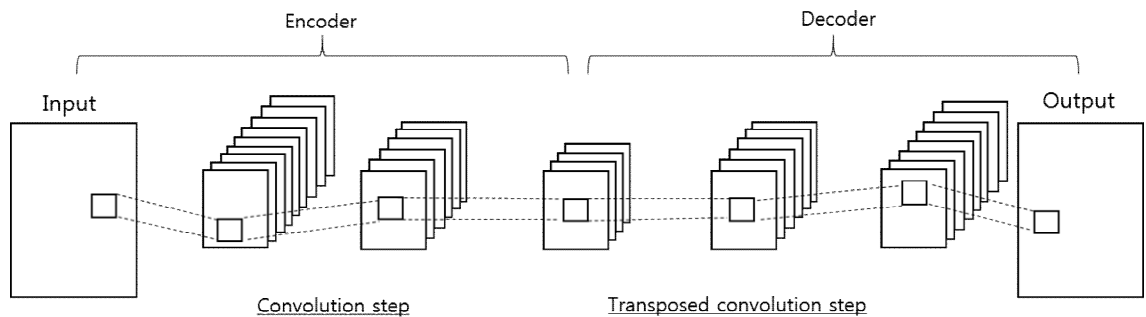
Fig 4.2. Flowchart of sensor fault diagnosis using convolutional auto-encoder based feature extraction and SVM

그림 4.2에서는 컨볼루셔널 오토인코더 기반의 특징 추출과 SVM을 사용한 센서 고장진단 순서도를 보여주고 있다. 그림 4.2에서 센서 데이터는 정상 신호로 학습된

CAE의 입력으로 들어가고 출력으로 복원된 데이터를 얻게 된다. 그리고 입력 센서 데이터(X)와 출력 데이터(\hat{X})의 차이 값에 절대 값을 취한다. 만약 입력 데이터가 정상 신호일 경우 값의 변화가 거의 없을 것이다. 하지만 고장 신호일 경우 큰 차이를 가지며, 고장신호의 유형에 따라 결과가 달라진다. 따라서 이 변화하는 값을 통해 statistical features를 구하고 이 특징들을 기반으로 SVM 모델을 학습하여 센서 고장을 진단한다.



(a) Convolutional auto-encoder operation process



(b) Proposed structure of convolutional auto-encoder

그림 4.3. 제안된 컨볼루셔널 오토인코더 동작과정 및 구조

Fig 4.3. Proposed convolutional auto-encoder operation and structure

본 논문에서 SVM 분류기의 학습 데이터로 사용될 특징을 추출하기 위한 컨볼루셔널 오토인코더(Convolutional Auto-Encoder, CAE)는 기존 오토인코더의 성능을 높이기 위해 인코더와 디코더에 convolution 연산을 적용시켜 구현하였다.

그림 4.3은 센서 데이터 특징 추출을 위해 제안된 컨볼루셔널 오토인코더의 구조를

보여준다. 위의 그림에서 나타낸 것처럼 제안된 컨볼루션 오토인코더는 인코더와 디코더의 구조를 가지며, convolution layer와 transposed convolution layer로 구성된다. 입력 데이터의 경우 하나의 샘플 당 1,000포인트의 데이터가 있으며, convolution 계산을 위해 50×20의 2차원 데이터 형태로 재구성한다. 인코더는 세 개의 convolution layer로 이루어져 있으며, 각 convolution layer를 통과할 경우 150, 100, 50개의 feature maps를 생성한다. 디코더는 인코더와 대칭구조를 가지도록 연산되며, 압축된 데이터를 복원한다. 사용된 필터의 크기는 3×3, stride는 첫 번째와 마지막 layer의 경우 2, 나머지는 1을 사용하였고 zero padding을 사용하여 convoluion 연산 후에도 feature map의 크기를 동일하게 유지하였다. 활성화 함수로는 ReLU가 사용되었다. 손실함수는 MSE로 설정하였으며, 학습을 위한 역전파 과정에서는 Adam Optimization이 사용되었다. 학습 데이터로는 정상 상태의 신호만을 사용하였다. 구현에는 Python 기반의 TensorFlow 라이브러리가 사용되었다.

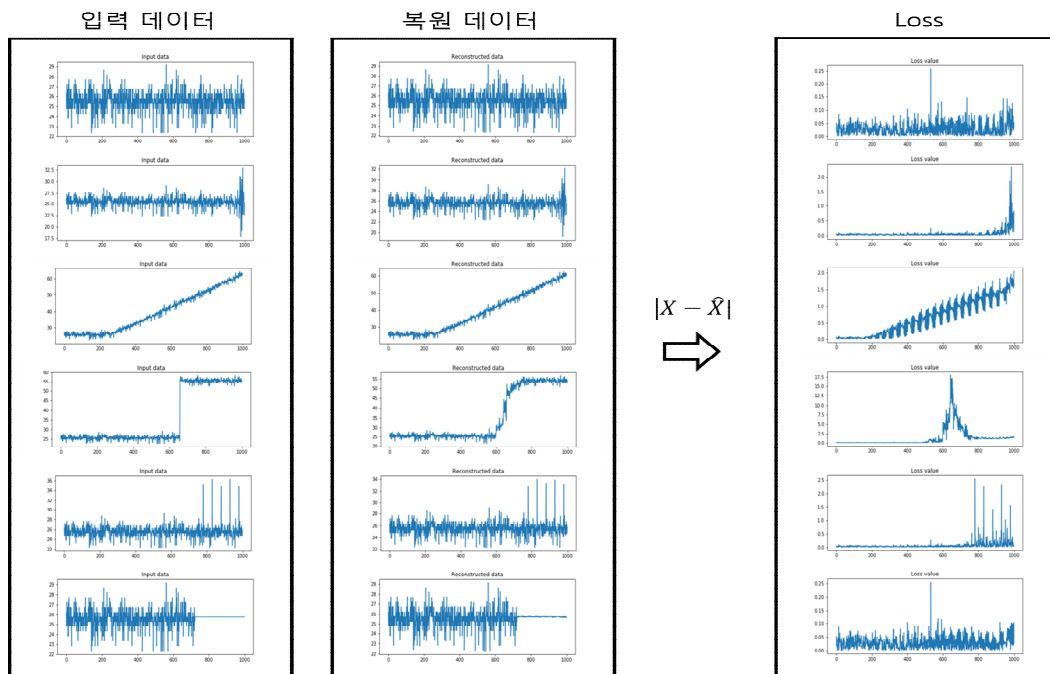


그림 4.4. 컨볼루션 오토인코더 기반의 특징 추출 과정

Fig 4.4. Feature extraction process based on convolutional auto-encoder

그림 4.4에서는 센서 유형별로 입출력 차이의 절대값을 취한 loss를 나타낸다. 특징 추출 결과에서 정상 신호의 loss의 크기는 다른 고장 유형에 비해 상당히 작아 차이가 거의 나지 않지만 다른 데이터는 고장 패턴에 따라서 입출력 차이의 크기나 위치가 다른 것을 확인할 수 있다.

4.1.3. 변분 오토인코더 잠재변수 기반 특징

VAE는 기존 오토인코더와는 달리 잠재변수를 입력데이터와의 매개변수로 설정하여 학습시킨다. 잠재변수 분포는 입력 데이터의 특성을 반영할 수 있으며, 디코더에서는 데이터 특징별로 분포해 있는 잠재변수 값에 따라 데이터를 재구성한다.

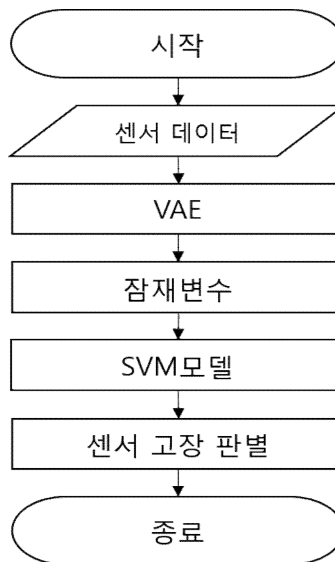


그림 4.5. VAE 기반 특징 추출과 SVM을 사용한 센서 고장진단 순서도

Fig 4.5. Flowchart of sensor fault diagnosis using variational auto-encoder based feature extraction and SVM

그림 4.5는 VAE 기반 특징 추출과 SVM을 사용한 센서 고장 진단 순서도를 보여주고 있다. 위의 과정에서 센서 데이터는 학습된 VAE의 입력으로 들어가고 VAE의 입출력 복원과정에서 2차원의 잠재변수를 추출한다. 센서 데이터의 잠재변수를 사

용하여 SVM 모델에 의해 고장 유무 및 유형이 결정된다.

그림 4.6은 특징 추출을 위해 구현된 VAE의 동작 과정을 나타낸다.

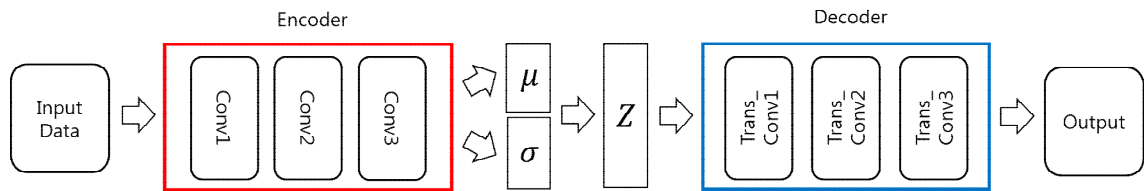


그림 4.6. 변분 오토인코더 기반 특징 추출 동작 과정

Fig 4.6. Feature extraction process based on variational auto-encoder

구현된 VAE의 인코더와 디코더의 기본적인 구조와 설정은 앞의 convolutional auto-encoder와 거의 유사하다. 다른 점은 VAE 인코더의 출력이 잠재변수(Z)의 평균과 분산이며 디코더에서는 잠재변수가 입력으로 사용되는 것이다. 그리고 입력 데이터를 정규화 시켜 사용하였기 때문에 디코더 맨 끝에서의 활성화 함수는 sigmoid가 사용되었다. 손실함수는 식 (2.11)을 사용하였으며, Adam optimization을 통해 손실함수의 값을 최소화 시키는 방향으로 학습되었다. 모든 클래스의 센서 데이터가 학습데이터로 사용되었으며, Python에서 TensorFlow 라이브러리로 구현되었다.

추출된 잠재변수가 유의미한 특징이 되기 위해서는 VAE에서 복원과정과 잠재변수의 분포 표현이 잘 수행되어야 한다. 그림 4.7에서는 클래스별 센서 데이터와 VAE에 의해 복원된 센서 데이터를 2차원 데이터로 나타내고 있다. 그림 4.7에서 가장 왼쪽에서부터 x축 기준 40간격으로 normal, erratic, drift, hard-over, spike, stuck의 입력 데이터와 복원 데이터를 보여주고 있으며, 40간격에서 왼쪽은 입력 데이터 오른쪽은 복원된 센서 데이터를 나타낸다. 가로 y축 기준 50간격은 클래스별로 다른 데이터 샘플들을 의미한다. 그림 4.7의 결과에서 drift, hard-over, spike, stuck 클래스들에 대해서는 어느 정도 복원이 잘 이루어져 각 클래스의 특성을 잘 반영한 것을 알 수 있지만, normal과 erratic 클래스는 복원 기능이 잘 작동되지 않았다는 것을 확인할 수 있다.

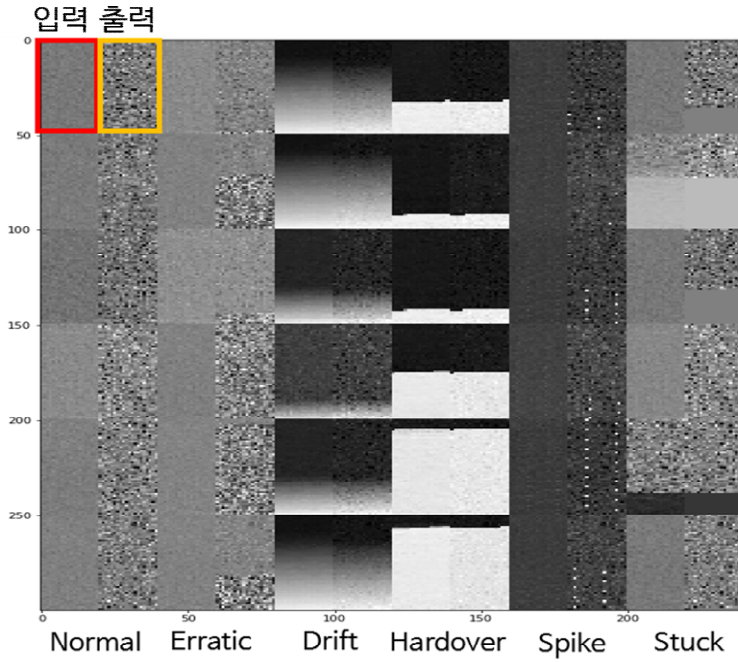


그림 4.7. 센서 입력 데이터(좌)와 복원된 센서 데이터(우)

Fig 4.7. Sensor input data(left) and reconstructed sensor data(right)

본 논문에서 구현한 VAE로부터 추출된 센서 데이터의 잠재변수들은 그림 4.8과 같이 나타낼 수 있다.

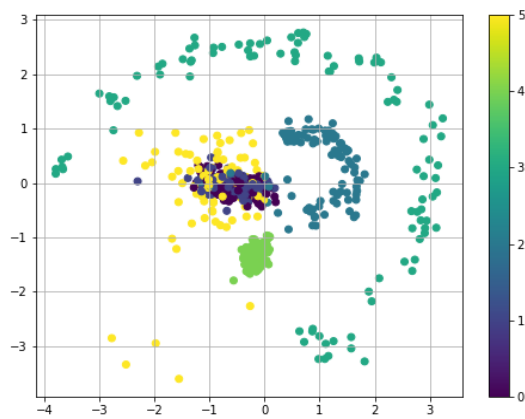


그림 4.8. VAE에 의해 추출된 센서 데이터의 잠재변수

Fig 4.8. Latent variables of sensor data extracted by VAE

위 그림에서 막대그래프에 표시된 0부터 5는 센서 고장 클래스인 normal, erratic, drift, hard-over, spike, stuck를 의미한다. 데이터 분포를 살펴보면 drift, hard-over, spike, stuck을 표현한 번호 2, 3, 4, 5의 점들은 다른 클래스들과 구분 지을 수 있는 분포를 가지지만 0번과 1번인 normal과 erratic 클래스는 서로 겹치는 분포를 형성하여 구분이 어렵다. 그림 4.7의 결과에서는 normal과 erratic은 복원이 잘 이루어지지 않았고 그림 4.8에서 두 클래스의 잠재변수 분포가 겹치는 것을 확인할 수 있다. 이는 잠재변수의 분포와 복원이 상관관계가 있다는 것을 의미한다.

그림 4.8의 결과로 보아 normal과 erratic의 잠재변수를 특징으로 사용하기는 어렵다. 하지만 데이터의 분포 특성 충분히 반영하는 나머지 클래스의 잠재변수를 사용하면 유의미한 결과를 얻어낼 수 있다. 또한 특징 선택과정을 거쳐 VAE 잠재변수 특징들과 다른 특징들을 조합함으로써 분류기의 성능을 충분히 향상시킬 수도 있다.

4.2. 유전 알고리즘 기반의 특징 선택

본 논문에서는 SVM을 효과적으로 학습시키기 위해 유전 알고리즘을 사용하여 추출된 특징들 중에서 우수한 특징들을 선별한다. 특징 선택을 통해 결과 예측에 영향이 없는 불필요한 특징들을 걸러낼 수 있고 데이터의 차원을 줄여 연산 속도를 높일 수 있다.

본 논문에서는 시간 영역 통계적 특징 12개, 오토인코더 이상치 기반 통계적 특징 12개, 변분 오토인코더 잠재변수 기반 특징 2개 총 26개의 특징들을 0과 1의 2진 비트 코드로 표현하여 하나의 염색체로 사용하였다. 해가 1로 표현되면 특징이 사용되고, 0이면 해당 특징은 사용되지 않는다.

초기 세대 생성과정에서는 바이너리 스트링을 랜덤하게 초기화하였으며 개체는 100개를 생성하였다.

적합도 평가에서 적합도 함수는 선택된 해들을 사용하여 학습 데이터에 대해

cross validation을 사용한 SVM의 정확도가 높고 사용된 특징들의 개수가 적을수록 높은 점수를 부여하였다. 따라서 적합도 함수는 수식 (4.1)과 같이 정의하였다.

$$F_i = f_i - a|n_i| \quad (4.1)$$

F 는 적합도 점수, f 는 정확도, n 은 사용된 특징들의 개수, a 는 개수 제한의 가중치를 의미한다. a 는 특징 개수가 얼마나 적합도 함수에 영향을 주는지에 대한 설정이며, 0.001로 설정하여 사실상 같은 정확도의 해가 있는 상황에서 개수가 작은 경우를 선택하도록 고려되었다. 수식 (4.1)에서 정확도에 큰 비중을 두고 특징 개수를 제한 조건으로 설정하여 해가 정확도를 높게 유지하면서 데이터 차원을 줄이도록 유도하였다.

선택 과정에서는 선택된 해에 대한 적합도 함수의 평가에 따라 부모 개체를 선택하는데 적합도 함수가 높은 개체가 확률적으로 더 많이 선택되는 roulette-wheel 방법을 사용하였다. 수식 (4.2)을 기반으로 각 개체가 선택될 확률을 적합도 점수에 비례하도록 부여한다. P 는 선택 확률, F 는 적합도 점수(혹은 적합도)를 의미한다.

$$p_j = \frac{F_j}{\sum_i F_i} \quad (4.2)$$

교차 단계에서는 부모 개체를 선택하여 무작위로 2개의 교차지점을 선택하고 교차지점을 기점으로 부모 개체의 유전자 일부분을 번갈아 조합하는 2점 교차방식을 사용하였다.

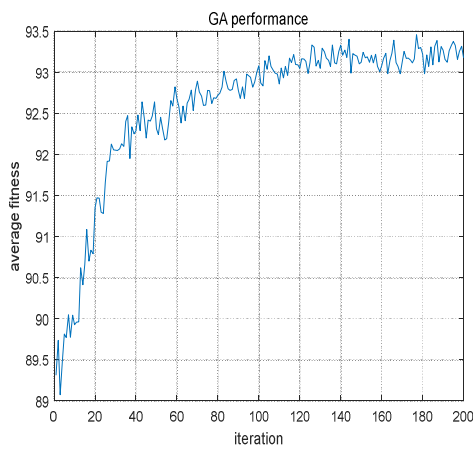
변이 과정은 각 개체마다 2%의 확률로 하나의 유전자 비트를 반전시키도록 설정하였다.

대치 과정에서는 전체적인 해집단의 품질을 향상시키기 위해서 기존 세대의 우수

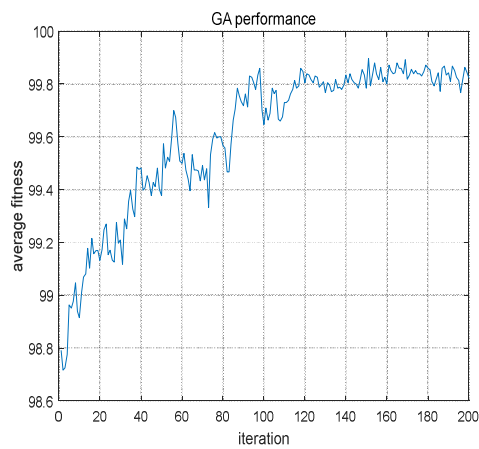
한 해를 다음 세대로 보존시키는 elitism 전략을 사용하였다. 부모 개체 중 적합도 점수가 높은 상위 5개를 다음 세대로 넘겨주도록 설정하였다.

종료 조건은 200세대 이상 즉 알고리즘을 200번 반복하게 되거나 특징 개수가 한 개이고 정확도 100%인 해를 찾게 되면 종료하도록 설정하였다.

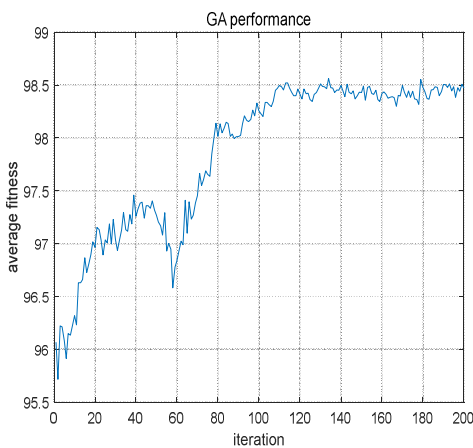
본 논문에서는 SVM의 학습을 위해 linear, RBF, polynomial 커널 함수들이 사용되었다. 따라서 유전 알고리즘은 각 클래스에 대해 커널별로 최적의 해를 찾기 위해 설계되었으며, MATLAB을 통해 구현되었다.



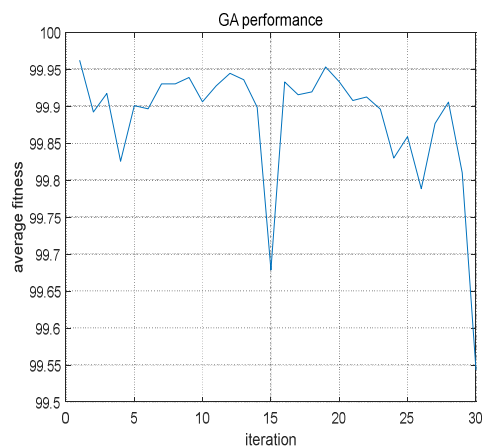
(a) Normal



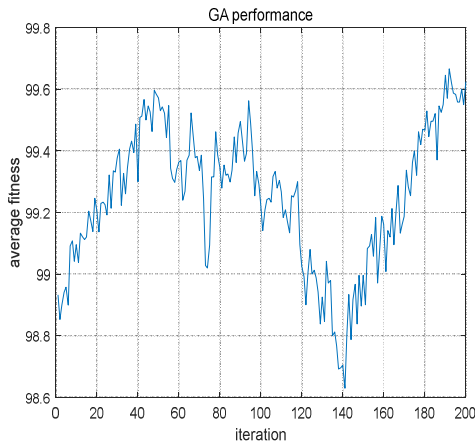
(b) Erratic



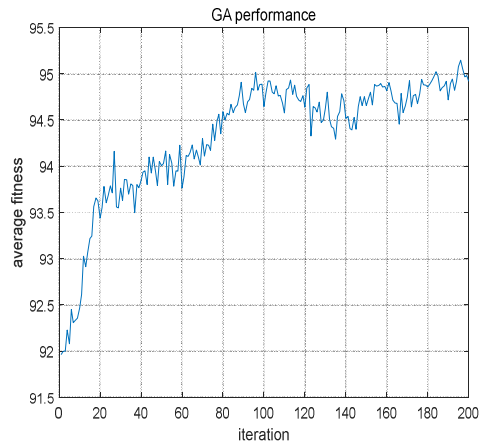
(c) Drift



(d) Hard-over



(e) Spike



(f) Stuck

그림 4.9. 해의 평균 적합도 점수 변화

Fig 4.9. Change in average fitness score of the solution

그림 4.9은 유전 알고리즘이 진행되는 동안 모든 해의 적합도 점수를 평균한 값을 세대마다 기록한 것을 나타내며, polynomial 커널 함수에 대해 적용한 결과이다. 그래프에서 확인할 수 있는 것처럼 대부분 클래스에 대해 평균 적합도 점수는 증가하다가 iteration 100이상부터는 어느 정도 수렴하는 것을 볼 수 있다. 이를 통해 유전 알고리즘이 진행됨에 따라 전체 해의 품질이 향상되고 있다는 것을 의미한다. Hard-over 고장 유형의 경우 유전 알고리즘이 30회 진행되었는데, 30회에서 최적의 해를 찾았기 때문에 종료 조건에 따라 종료되었다.

표 4.2에서는 유전 알고리즘에 의해 선택된 특징들을 보여주고 있다. 선정 기준은 유전 알고리즘 종료 시에 가장 높은 적합도 점수를 가진 해이다. 상단 0부터 5까지는 각각 normal, erratic, drift, spike, stuck을 의미하며, 상단 밑의 숫자 1부터 26은 시간 영역 통계적 특징(1~12), 오토인코더 이상치 기반 통계적 특징(13~24), VAE 잠재 변수 기반 특징(25,26)을 의미한다. 표 4.2에서 가장 많이 선택된 특징의 개수는 linear kernel을 사용하여 normal 클래스 판별한 경우 15개로 유전 알고리즘을 통해 효과적으로 개수를 줄인 것이라고 볼 수 있다. 그리고 hard-over 클래스의 경우 데이터 특징 1개로 최적의 값을 찾아낸 것을 확인할 수 있다.

표 4.2. 유전 알고리즘으로부터 선택된 특징들

Table 4.2. Features selected from the genetic algorithm

Kernel Function	0	1	2	3	4	5
Linear	1, 2, 3, 6, 8, 10, 11, 12, 15, 17, 19, 21, 24, 25, 26	1, 2, 3, 14, 16, 26	10, 15, 19, 20, 22, 23, 25	24	2, 5, 7, 8, 17, 24	1, 2, 3, 4, 5, 10, 11, 12, 15, 16, 23, 25
RBF	1, 2, 3, 5, 8, 9, 10, 11, 15, 17, 20, 26	2, 3, 8, 26	3, 8, 20, 22	13	3, 26	1, 9, 17, 20, 22
Polynomial	1, 2, 6, 9, 10, 11, 13, 15, 16, 20, 23, 26	3, 9, 23	1, 20, 23, 25	24	3, 7, 16, 18	2, 4, 6, 7, 9, 10, 11, 20, 23, 25, 26

4.3. 특징별 SVM 분류 결과 분석 및 다중 계층 SVM 구현

본 연구에서는 앞서 설명한 유전 알고리즘을 사용한 특징 선택 과정을 거쳐 SVM 학습 및 테스트를 실행한다. 그림 4.10은 센서 고장 진단 및 검출을 위한 전체 프로세스를 보여주고 있다.

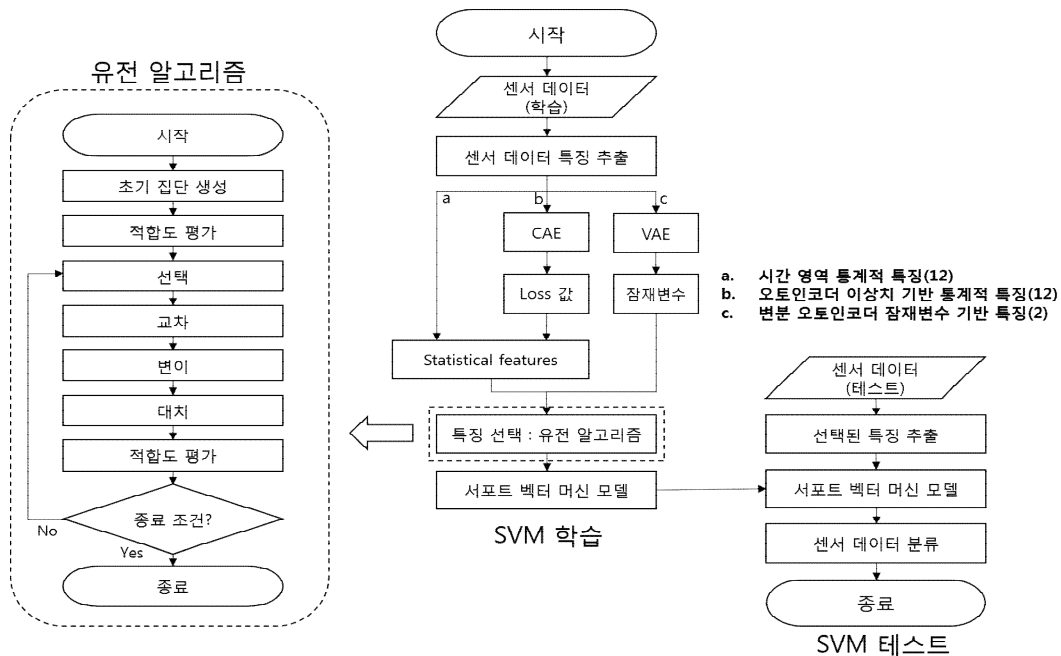


그림 4.10. 센서 고장 진단 전체과정

Fig 4.10. Sensor fault diagnosis whole process

본 논문에서는 SVM은 one-versus-rest 방법을 활용하여 각 결함 유형에 따라 분류하였다. One-versus-rest 방법은 데이터의 전체 클래스의 수가 N개인 경우, 분류하고 싶은 i번째 클래스와 나머지 N-i개의 클래스를 이진 분류시키는 것으로 그림 4.11와 같은 방식으로 분류한다. 센서 고장 분류의 경우 분류하고 싶은 클래스가 normal인 경우, normal을 +1 나머지 클래스인 erratic, drift, spike, stuck을 -1로 두고 결정하는 방식이다.

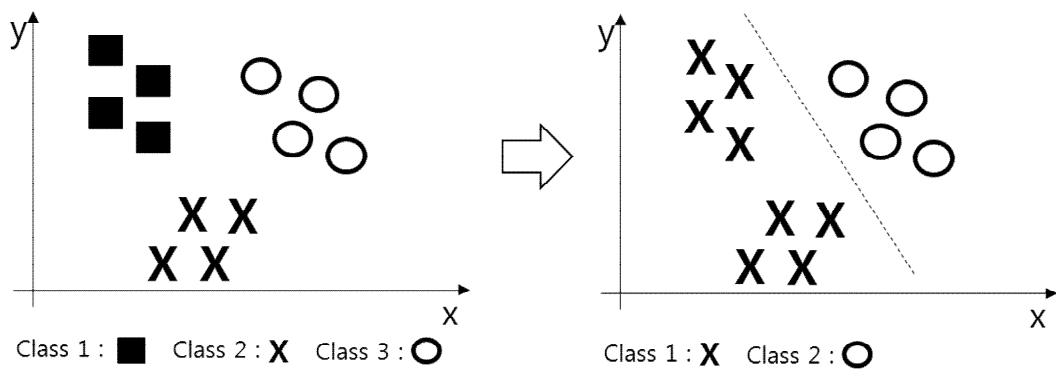


그림 4.11. One-versus-rest 방법

Fig 4.11. One-versus-rest method

본 논문에서 센서 고장 진단을 위해 사용된 SVM 분류기는 MATLAB에서 `fitcsvm`을 통해 구현되었다. SVM을 학습시키기 위해 각 고장 클래스 별로 40개의 샘플들이 사용되었고 나머지 60개의 샘플들이 테스트를 위해 사용되었다. SVM을 학습을 위해 linear, RBF, polynomial 커널 함수들이 사용되었으며, 커널 함수별로 학습된 SVM을 사용하여 각각 센서 고장 감지 및 분류를 수행하였다. RBF와 polynomial 커널 함수에서 kernel scale은 1, kernel function order은 3으로 설정하였다. SMO(Sequential minimal optimization) 알고리즘이 사용되었으며, cost값은 1로 설정되었다.

표 4.3. 특징에 따른 SVM 정확도(%)

Table 4.3. Accuracy of SVM for each features(%)

(a) Accuracy of SVM(%) with time-domain statistical features

Kernel Function	Time-domain statistical features					
	0	1	2	3	4	5
Linear	87.22	100	83.33	92.78	97.5	86.11
RBF	92.22	99.72	97.5	99.44	99.44	92.5
Polynomial	93.06	90	96.39	100	99.72	94.44

(b) Accuracy of SVM(%) with convolutional auto-encoder based features

Kernel Function	Convolutional auto-encoder based features					
	0	1	2	3	4	5
Linear	85	100	91.39	100	98.61	88.33
RBF	90.83	99.44	97.78	98.61	98.06	91.39
Polynomial	89.17	99.72	96.11	100	99.44	91.94

(c) Accuracy of SVM(%) with VAE latent variables based features

Kernel Function	VAE latent variables based features					
	0	1	2	3	4	5
Linear	83.33	83.33	83.33	91.67	95.83	83.33
RBF	83.33	83.33	97.5	99.72	100	85.28
Polynomial	83.33	83.33	96.39	100	99.44	85

(d) Accuracy of SVM(%) with All features

Kernel Function	All features					
	0	1	2	3	4	5
Linear	88.06	100	97.78	100	100	93.06
RBF	90.56	98.61	98.06	97.5	97.78	88.33
Polynomial	90	99.44	94.72	100	100	93.33

(e) Accuracy of SVM(%) with selected features by GA

Kernel Function	Selected features					
	0	1	2	3	4	5
Linear	87.78	100	97.78	100	100	91.94
RBF	91.39	99.72	98.61	99.72	100	93.06
Polynomial	94.17	99.72	98.06	100	100	95.28

표 4.3은 특징에 따른 SVM 정확도 테스트 결과를 확인할 수 있다. 0부터 5는 normal, erratic, drift, spike, stuck 고장 유형을 의미하며, 커널 함수별로 각 클래스 판별에 대한 정확도를 측정하였다.

본 논문에서 사용된 3가지 특징 추출 방법으로 추출된 특징을 각각 SVM에 사용한 경우, RBF 커널에서 시간 영역 통계학적 특징을 기반으로 한 SVM 정확도가 평균 96.81%로 가장 높은 정확도를 기록했다. VAE 잠재변수 기반 SVM의 경우 특징 추출과정에서 확인한 것처럼 normal과 erratic 클래스에 대해서는 정확도가 높지 않지만 RBF 커널 함수를 사용한 경우에는 spike 유형 분류, polynomial 커널에서 hard-over 유형 분류에 있어서는 소수의 특징들만 사용하여 100%분류 성능을 보여준다. 추출된 26가지의 특징들을 전부 사용하여 학습과 테스트를 진행한 경우, linear 커널 함수에서 평균 96.48%로 가장 높은 정확도를 보여준다. 모든 특징을 사용한 분류 정확도는 전체적으로 높지만 가장 뛰어난 성능을 보여주지는 않는다. 그러므로 많은 특징들을 사용한다고 해서 분류기의 성능이 향상되는 것은 아닌 것을 확인할 수 있다.

반면 유전 알고리즘을 적용시켜 선별된 특징을 사용한 SVM 분류기의 정확도는 polynomial 커널 함수에서 가장 높은 평균 97.87%의 정확도를 보여준다. Hard-over과 spike 클래스 분류에서 각각 한 개, 네 개의 특징들만 사용하여 100%의 정확도를 보여주며, normal과 stuck 유형 분류 결과는 94.17%, 95.28%로 표 4.3에서 가장 높은 성능을 보여준다.

표 4.3를 통해 추출된 특징들 종류에 따라 성능이 좌우되는 것을 확인할 수 있었다. 또한 유전 알고리즘을 사용한 데이터 선택이 효과적으로 데이터 차원을 줄였을 뿐만 아니라 분류 성능도 향상시킬 수 있음을 확인할 수 있다.

본 연구에 SVM은 one-versus-rest 방법을 사용하였으므로 입력 데이터가 들어왔을 때 그 데이터의 클래스를 직접적으로 판별하기 위해 다중 계층 SVM 분류기 (Multi-layer SVM)를 그림 4.12와 같이 설계하였다. 입력이 들어오면 SVM분류기는 하나의 클래스에 대해 분류하고 해당 클래스가 아닐 경우 다른 클래스 학습된 SVM

분류기로 판별한다. Multi-layer SVM은 표 4.3에서 가장 정확도가 높은 (d)의 polynomial 커널 함수를 사용한 결과에 설계되었으며, 정확도가 높은 클래스에 해당하는 SVM 분류기가 상위 계층에 위치하도록 설계하였다.

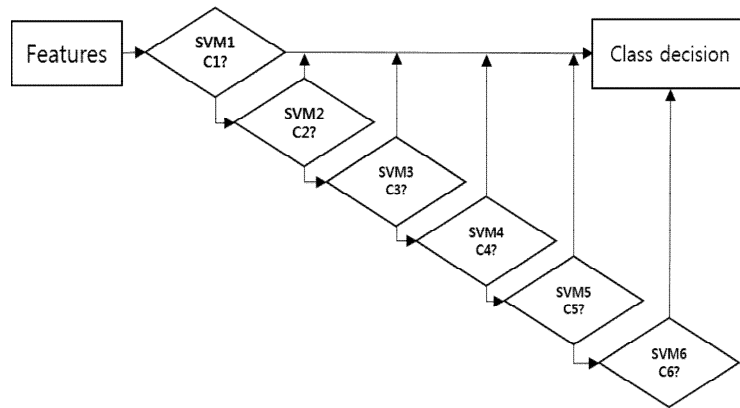


그림 4.12. 다중 계층 SVM

Fig 4.12. Multi-layer SVM

표 4.4. SVM-Poly 분류 결과에 대한 혼동행렬(%)

Table 4.4. Confusion matrix for SVM-Poly classification result(%)

	0 (normal)	1 (erratic)	2 (drift)	3 (hardover)	4 (spike)	5 (stuck)
0 (normal)	100	0	0	0	0	0
1 (erratic)	1.67	98.33	0	0	0	0
2 (drift)	11.67	0	88.33	0	0	0
3 (hardover)	0	0	0	100	0	0
4 (spike)	0	0	0	0	100	0
5 (stuck)	23.33	0	0	0	0	76.67

표 4.4은 multi-layer SVM을 사용하여 센서 고장 데이터를 판별한 결과를 보여준다. 표 4.4에서 좌측과 상단에 0부터 5까지의 숫자는 각각 실제 클래스와 예측된 클

래스로 표시하여 정확도를 나타내었다. 표 4.4의 좌측 1(erratic)은 98.33%가 동일한 클래스인 1(erratic)으로 분류되었고 나머지 1.67%는 0(normal) 클래스로 분류되었음을 확인할 수 있다. 2(drift)는 88.33%의 정확도로 11.67%가 0(normal)로 잘못 분류되었으며, 5(stuck)의 경우 23.33%를 0(normal)로 관별해 76.67%의 정확도를 보인다. 오분류된 클래스를 살펴보면 모두 0(normal)로 인지하였으며, 그 중에서 5(stuck)이 높은 비율을 보이고 있다. 이는 stuck의 특성이 normal과 가장 구분 짓는 것이 힘든 것으로 판단되며, SVM 분류기 성능에 가장 영향을 주는 요인이라고 분석된다.

5. CNN 기반 센서 고장 분류

5.1. CNN 모델 구현 및 학습

본 논문에서는 센서 고장 감지 및 분류를 위해 CNN 분류기를 설계하였다. 그림 5.1에는 본 논문에서 제안한 CNN 모델을 보여준다.

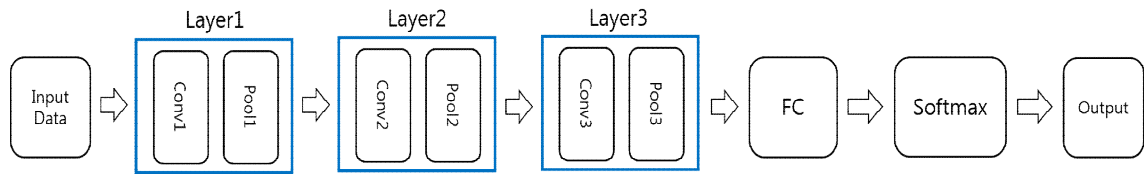


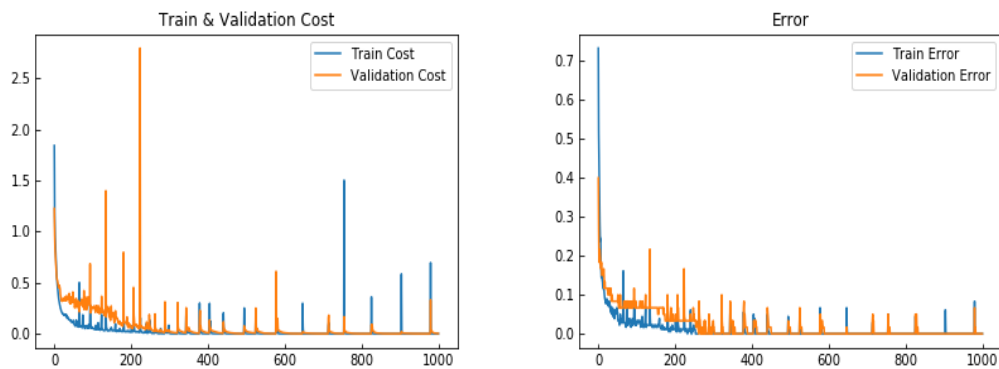
그림 5.1. 센서 고장 분류를 위한 CNN 동작 과정

Fig 5.1. CNN operation process for sensor fault classification

구현된 CNN 모델은 세 개의 convolution layer, 세 개의 pooling layer 그리고 fully connected layer를 가진다. 입력 단계에서는 1,000개의 데이터 포인트를 가진 하나의 데이터 샘플을 convolution 연산을 위해 40×25 형태의 행렬로 변환하여 사용하였다. 세 개의 convolution layer에서 convolution 연산에 사용된 필터의 크기는 3×3 , stride는 1로 설정하였고, zero padding을 사용하여 convolution 연산 후에도 feature map의 크기를 동일하게 유지시켰다. 첫 번째 convolution layer에서는 50 feature maps가 사용되었고 100, 150 feature maps가 두, 세 번째 convolution layer에 사용되었다. 추출된 feature map에는 활성화 함수인 ReLU를 적용하였다. 각 pooling layer에서는 max pooling 기법이 사용되었고 2×2 크기로 sub-sampling 되었으며 stride는 2로 설정되었다. Fully connected layer는 100개의 노드와 6개의 출력을 가진다. 분류 결과를 얻기 위해 softmax 함수가 사용되었으며, softmax 함수에 대한 cross entropy 함수의 평균이 손실함수로 사용되었다. CNN 학습을 위한 역전파 과정에서는 ADAM optimization이 사용되었고 over-fitting을 방지하기 위해 0.7의 확률로 dropout 기법이 적용되었다. CNN 모델 학습 및 테스트를 위해 Python 기반의 TensorFlow 라이브러

리가 사용되었다.

본 논문에서 CNN 학습에서는 각 유형별로 30개의 데이터 샘플들로 학습 시키면서 10개의 검증 데이터로 각 epoch마다 cost와 에러율에 대해 모니터링하고 모델을 저장하였다. 각 epoch마다 기록된 cost와 에러율에서 가장 낮은 모델을 테스트에 사용하였다. 테스트 데이터는 각 고장 클래스별로 60개의 데이터 샘플이 사용되었다. 그림 5.2는 매 epoch마다 학습과 검증 데이터에 대해 cost와 에러율을 기록한 것을 나타낸다.



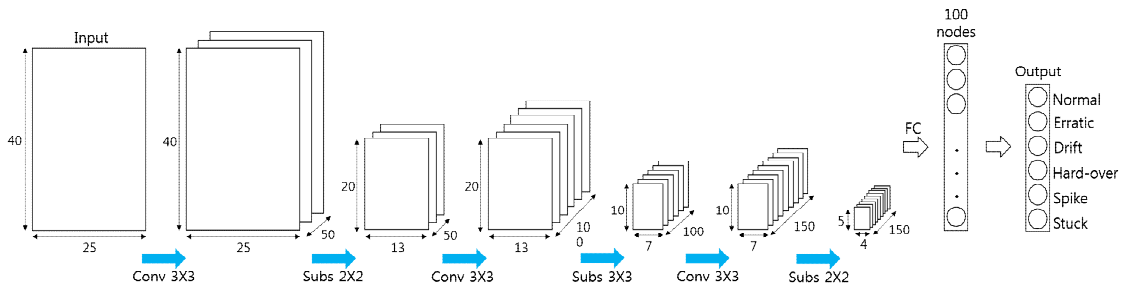
(a) Train and validation cost per epoch (b) Train and validation error per epoch

그림 5.2. CNN의 손실과 에러율 모니터링

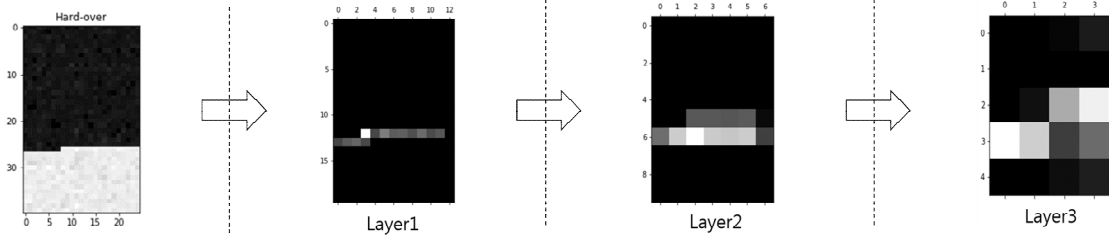
Fig 5.2. CNN cost and error monitoring

그림 5.2에서 확인할 수 있듯이 학습이 진행되면서 학습 데이터의 cost와 error가 감소함에 따라 검증 데이터도 감소하고 있다. 이를 통해 과적합을 피해 잘 학습시킨다는 것을 확인할 수 있다.

그림 5.3은 데이터가 CNN을 통과하면서 어떤 식으로 센서 고장이 감지되는지 과정을 층별로 보여준다. 2차원 데이터는 CNN의 입력으로 사용되며, convolution layer와 pooling layer를 거치면서 특징을 추출하게 된다. 그림 5.3의 (b)에서는 hard-over 클래스가 CNN을 통해 특징이 추출되는 과정을 layer별로 보여주고 있다.



(a) Proposed CNN structure



(b) Sensor fault recognition process by CNN

그림 5.3. 제안된 CNN 구조 및 센서 고장감지 과정

Fig 5.3. Proposed CNN structure and sensor fault recognition process

표 5.1. CNN 분류 결과에 대한 혼동행렬(%)

Table 5.1. Confusion matrix for CNN classification result(%)

	0 (normal)	1 (erratic)	2 (drift)	3 (hardover)	4 (spike)	5 (stuck)
0 (normal)	96.67	0	0	0	0	3.33
1 (erratic)	0	98.33	1.67	0	0	0
2 (drift)	3.33	0	96.67	0	0	0
3 (hardover)	3.33	0	0	96.67	0	0
4 (spike)	0	0	1.67	0	95	3.33
5 (stuck)	3.33	0	0	0	0	96.67

표 5.1에서는 혼동행렬을 통해 학습된 CNN의 분류 결과를 확인할 수 있다. CNN의 분류는 전체 96.67%의 정확도 결과를 보여준다.

5.2. CNN 모델 성능 개선 및 분류 결과

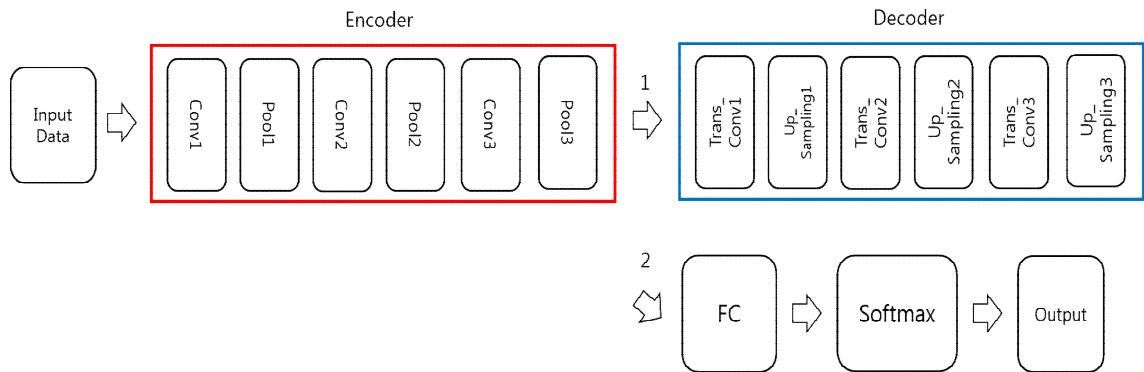


그림 5.4. CNN 성능 개선 과정

Fig 5.4. CNN performance improvement process

본 논문에서는 앞에서 구현된 CNN 분류 성능을 향상시키기 위해 컨볼루션 오토인코더를 사용한다. 우선 컨볼루션 오토인코더를 사용하여 각 고장 클래스별로 사전학습(pre-training)을 진행한다. Pre-training을 수행하게 되면 미리 고장 클래스에 대한 특징들을 학습할 수 있으며, 이미 학습된 모델을 사용하기 때문에 빠르게 파라미터들이 업데이트 될 수 있다. 학습을 완료한 후 인코더와 디코더 부분을 분리한다. 그리고 학습된 인코더 네트워크에 fully connected layer를 연결하여 센서 고장 분류에 맞게 다시 파라미터들을 업데이트하는 파인튜닝(Fine tuning)을 수행한다.

그림 5.4은 CNN 모델 성능 개선 과정을 보여준다. 구현된 컨볼루션 오토인코더의 구조는 앞에서 구현된 CNN의 fully connected 이전까지의 convolution layer들과 대칭되는 구조로 설정된다. 손실함수는 MSE가 사용되었으며, 역전파 방법으로는 ADAM optimization이 사용되었다.

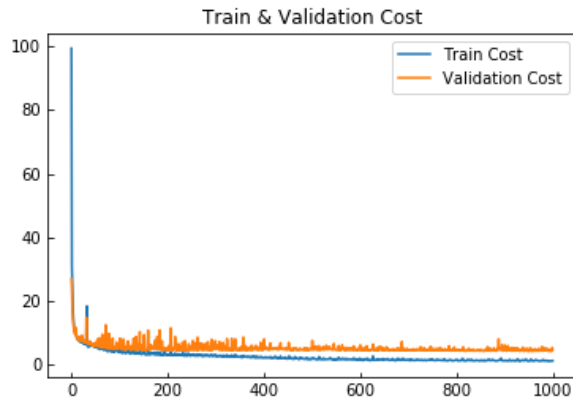


그림 5.5. 학습 및 검증 손실 모니터링

Fig 5.5. Monitoring train and validation cost

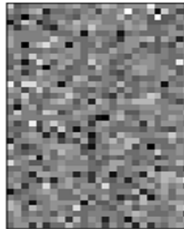
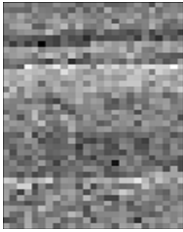
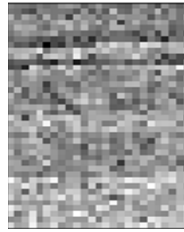
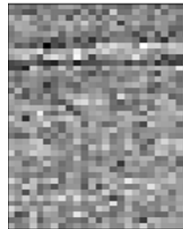
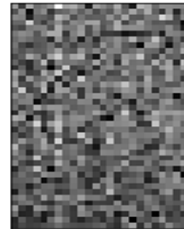
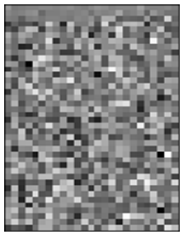
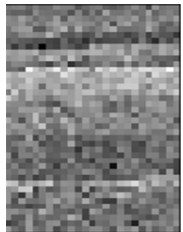
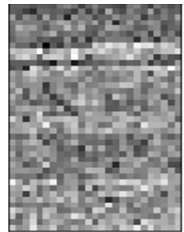
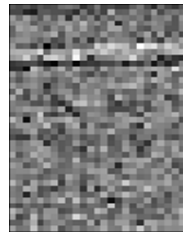
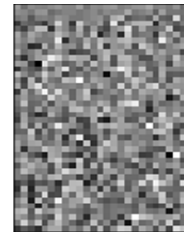
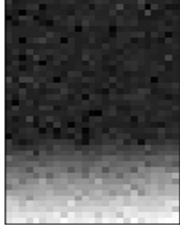
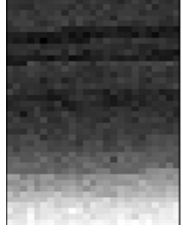

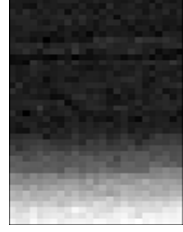

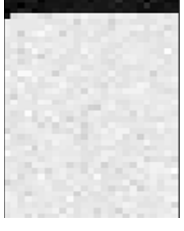
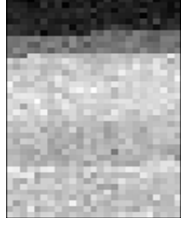



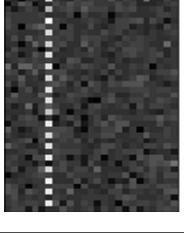
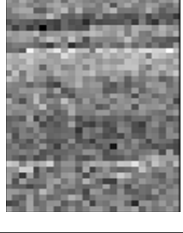
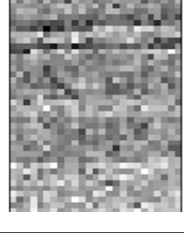
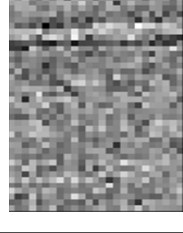
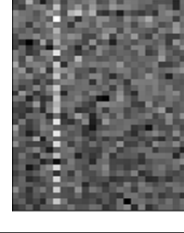
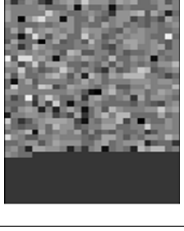
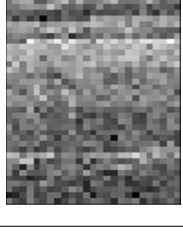
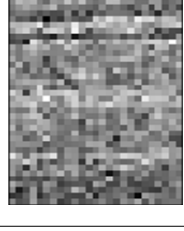
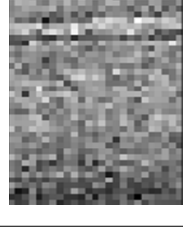
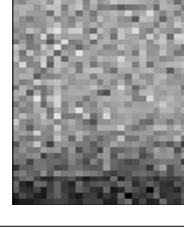
그림 5.5는 컨볼루션 오토인코더 학습 시에 cost를 모니터링한 그래프를 보여주고 있다. 그래프에서 학습 데이터에 대한 cost는 계속해서 감소 추세를 보이지만 검증 데이터에 대한 cost는 일정 수치에서 멈춰 더 이상 내려가지 않는다. 따라서 과적합 모델을 피하기 위해 가장 낮은 cost를 가진 모델이 분류기 구현 시 재사용되었다.

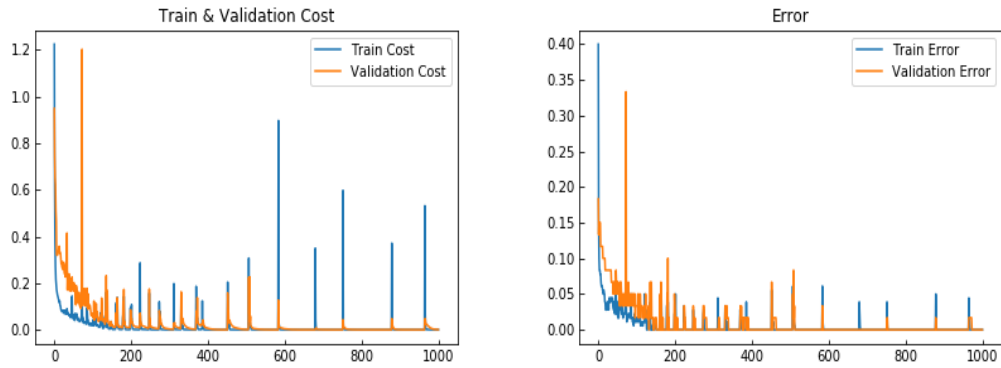
표 5.2는 컨볼루션 오토인코더를 사용한 센서 고장 데이터 복원 상태를 epoch에 따라 나타낸다. 학습이 진행될수록 좀 더 복원이 원래의 데이터에 가깝게 되는 것을 확인할 수 있으며, 가장 손실함수가 작은 958 epoch의 복원 결과를 최종 복원상태로 보여준다. 이 과정에서 학습이 완료된 후, 인코더를 분리하고 fully connected를 연결하여 학습시킨다. 학습을 위한 설정은 5.1절에서 구현된 CNN과 동일하다.

그림 5.6에서는 pre-training시킨 CNN의 학습 과정을 모니터링한 결과를 보여준다. 그림 5.6의 결과로 그래프의 cost와 에러가 기존 CNN 학습과정을 나타낸 그림 5.2의 cost와 에러보다 훨씬 더 빠르게 감소한다는 것을 알 수 있다. 두 그래프를 비교했을 때 검증 데이터에 대해 그림 5.6의 그래프에서는 200 epoch이 되기 전에 어느 정도 수렴하는 것을 볼 수 있다. 하지만 그림 5.2의 그래프에서는 300 epoch까지 계속해서 감소한다.

표 5.2. Epoch에 따른 센서 데이터 복원

Table 5.2. Sensor data reconstruction by epoch

	Input	10 epoch	50 epoch	100 epoch	958 epoch
Normal					
Erratic					
Drift					
Hard-over					
Spike					
Stuck					



(a) Train and validation cost per epoch (b) Train and validation error per epoch

그림 5.6. CNN의 손실과 에러율 모니터링

Fig 5.6. CNN cost and error monitoring

표 5.3에서는 혼동행렬을 통해 성능이 향상된 CNN의 분류 결과를 확인할 수 있다. 전체적인 CNN 분류 정확도는 97.22%로 기존 CNN보다 높은 성능을 보여준다. Spike 클래스에 대한 분류 성능은 91.67%로 기존 95%의 정확도 보다 떨어졌지만 normal, erratic, stuck 클래스에 대한 분류 성능이 향상되어 전체적인 분류기 정확도는 높아진 것을 확인할 수 있다.

표 5.3. 향상된 CNN 분류 결과에 대한 혼동행렬(%)

Table 5.3. Confusion matrix for improved CNN classification result(%)

	0 (normal)	1 (erratic)	2 (drift)	3 (hardover)	4 (spike)	5 (stuck)
0 (normal)	100	0	0	0	0	0
1 (erratic)	0	100	0	0	0	0
2 (drift)	3.33	0	96.67	0	0	0
3 (hardover)	0	0	0	96.67	0	3.33
4 (spike)	5	1.67	1.67	0	91.67	0
5 (stuck)	1.67	0	0	0	0	98.33

6. SVM, CNN 센서 고장 분류 성능 비교분석

센서 고장 진단을 위해 구현된 SVM과 CNN 분류기의 성능을 아래의 그림 6.1과 표 6.1에 나타내었다. 그림과 표에서 SVM과 CNN은 4.3절의 multi-layer SVM과 5.2절에서 구현된 CNN을 지칭한다.

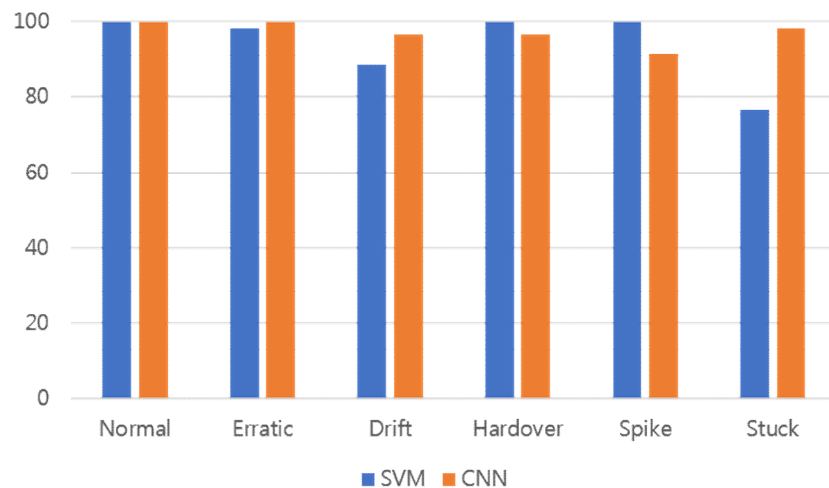


그림 6.1. SVM과 CNN 정확도 비교(%)

Fig. 6.1. Comparison of SVM and CNN accuracy(%)

표 6.1. SVM과 CNN 분류 결과(%)

Table 6.1. SVM and CNN classification results(%)

	Normal	Erratic	Drift	Hardover	Spike	Stuck
SVM	100	98.33	88.33	100	100	76.67
CNN	100	100	96.67	96.67	91.67	98.33

전체 정확도는 SVM이 93.89% 그리고 CNN이 97.22%로 CNN이 SVM보다 3.33% 정도 높다는 것을 확인할 수 있으며, drift, spike, stuck 유형에서 정확도 차이가 크게 발생하는 것을 알 수 있다. Spike 클래스에서 정확도는 CNN이 SVM에 비해 8.33%정도 떨어지며, drift와 stuck의 경우 CNN이 SVM보다 8.34%, 21.66% 높은 것을 확인할 수 있다. 특히 stuck의 경우 가장 큰 차이를 보여주는데 이는 CNN을 통

한 특징 추출이 다른 추출기법보다 뛰어나다는 것을 알 수 있으며, 특징 추출 기법이나 데이터에 따라 성능이 달라 질 수도 있다는 것을 증명한다.

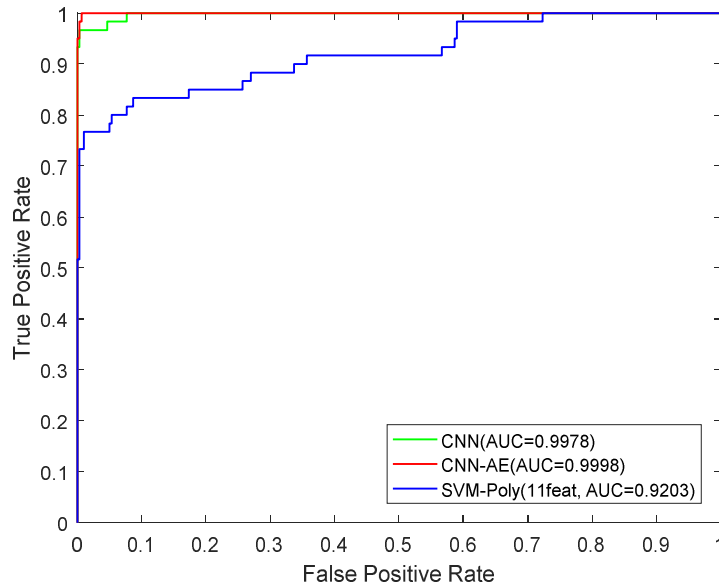


그림 6.2. SVM, CNN의 AUC-ROC

Fig. 6.2. AUC-ROC of SVM and CNN

분류기의 성능을 효과적으로 표현하기 위해 그림 6.2와 같이 SVM, CNN 그리고 pre-training을 사용한 CNN에 대한 Receiver Operator Characteristic curve(ROC)와 Area Under the Curve(AUC)를 나타낸다[17]. Stuck 클래스의 데이터 샘플들이 positive class로 사용되었고, 나머지 클래스들은 negative class로 사용되었다. 그래프에서 pre-training을 사용한 CNN의 경우 가장 먼저 true positive rate 1에 도달하였으며, AUC 값도 0.9998로 SVM 보다 0.0795나 높은 수치를 기록했다.

7. CNN 기반 센서 상태 모니터링

본 논문 5.2절에서 제안된 CNN 센서 고장 진단 모델을 기반으로 센서 고장 진단 결과 모니터링을 구현하고 시뮬레이션하였다. 그림 7.1은 전체적인 시스템 구성도를 보여준다.

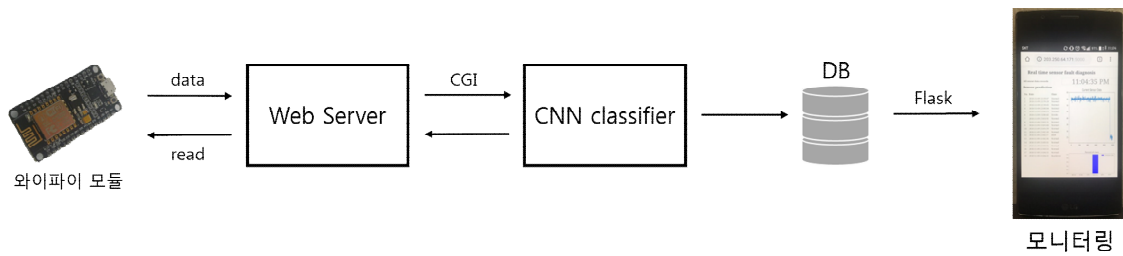


그림 7.1. 센서 고장 모니터링을 위한 전체 과정

Fig. 7.1. Process for monitoring sensor fault

센서 데이터는 와이파이 모듈에 의해 웹 서버(Web server)로 전송된다. 웹 서버는 CGI(Common Gateway Interface)를 통해 Python에서 센서 데이터 고장을 판별하고 결과를 받는다. 센서 고장 데이터와 분류 결과는 데이터베이스(Database, DB)에 저장되고 이 결과를 스마트폰을 통해 디스플레이한다.

위 그림에서 데이터 송신은 ESPRESSIF사의 ESP8266-12모듈을 장착한 NodeMCU V1.0가 사용되었으며, 아두이노 IDE를 통해 구현되었다. 웹 서버는 비트나미(bitnami)를 통해 구축되었으며, 데이터베이스는 MySQL를 사용하였다. 또한 모니터링을 위해 Python 웹 프레임워크인 Flask가 사용되었다.

그림 7.2는 아두이노 시리얼 모니터와 스마트폰을 통해 센서 데이터 모니터링 결과를 보여준다.

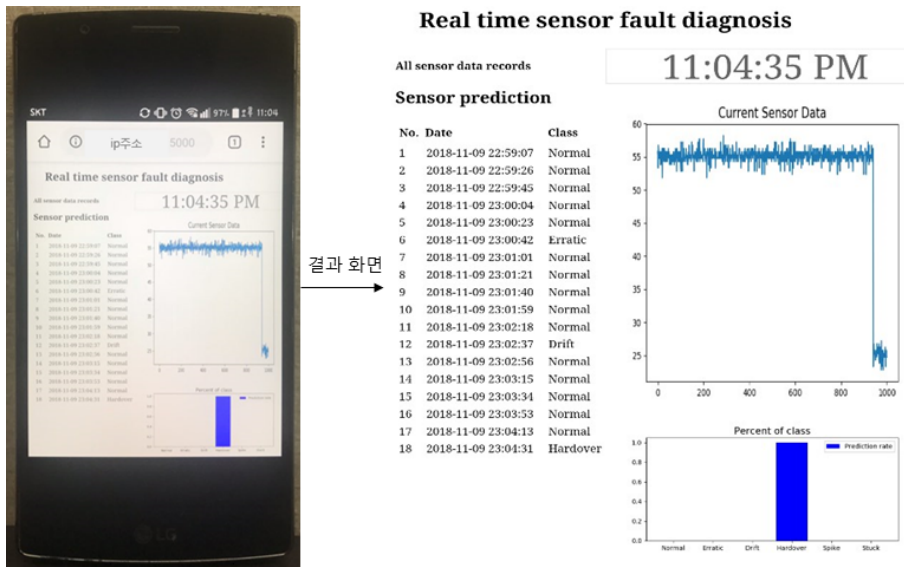
```

COM6
====Database Connection====
host : 127.0.0.1
port : 3307
DB_name : thesis_test1
Sensor Prediction Operation
Prediction : [0]
Normal(%) : 100.00%
Erratic(%) : 0.00%
Drift(%) : 0.00%
Hardover(%) : 0.00%
Spike(%) : 0.00%
Stuck(%) : 0.00%
0

sensor_class : 0
sensor_class_order : 0
connection Success
Requesting GET: /thesis2/thesis2.py?&sendin_val=26.74&
connection Success
Requesting GET: /thesis2/thesis2.py?&sendin_val=25.27&
connection Success
Requesting GET: /thesis2/thesis2.py?&sendin_val=24.78&
connection Success
Requesting GET: /thesis2/thesis2.py?&sendin_val=25.27&

```

(a) Arduino serial monitor



(b) Sensor fault diagnosis result display

그림 7.2. 센서 고장 진단 결과 모니터링

Fig. 7.2. Monitoring of sensor fault diagnosis result

8. 결론

본 논문에서는 센서 고장 검출 및 분류를 위해 SVM과 CNN을 적용하였다. 센서 데이터 샘플들로부터 시간 영역 통계 특징, 오토인코더 이상치 기반 통계적 특징, VAE 잠재변수 기반 특징들을 추출하였고, 유전 알고리즘을 사용하여 선별된 특징들을 이용하여 SVM 분류기의 분류 성능을 개선시켰다. 선택된 특징들을 사용한 SVM 분류 결과 97.87%로 가장 높았으며, 선택된 특징들을 기반으로 multi-layer SVM을 설계하고 테스트 한 결과 93.89%의 정확도로 분류하였다. CNN을 사용한 분류에서는 CNN의 성능을 향상시키기 위해 비지도 학습 방법으로 pre-training을 수행한 뒤 센서 고장 분류를 학습시키는 방법을 사용하였으며 전체 97.22%의 높은 정확도를 기록하였다. 가장 큰 정확도 차이를 보여준 Stuck 클래스의 AUC-ROC를 통해 SVM과 CNN의 성능을 비교하였으며, CNN의 성능이 우수함을 확인할 수 있었다.

참 고 문 헌

- [1] 장경석, “국내 외 스마트 팩토리 동향”, KB 지식 비타민(17-37호), 2017.
- [2] Don-Ha Hwang, Young-Woo Youn, Jong-Ho Sun, Kyeong-Ho Choi, Jong-Ho Lee and Yong-Hwa Kim, “Support Vector Machine Based Bearing Fault Diagnosis for Induction Motors Using Vibration Signals”, Journal of Electrical Engineering & Technology, Vol. 10, no. 4, pp. 1558-1565, 2015.
- [3] Wade A. Smith, Robert B. Randall, “Rolling element bearing diagnostics using the Case Western Reserve University data: A benchmark study”, Mechanical Systems and Signal Processing, Vol. 64-65, pp. 100-131, 2015.
- [4] Subhasis Nandi, Hamid A. Toliyat and Xiaodong Li, "Condition Monitoring and Fault Diagnosis of Electrical Motors", IEEE Transactions On Energy Conversion, Vol. 20, no. 4, pp. 719-729, 2005.
- [5] 이병희, “아시아나항공 여객기, 연기센서 오류로 긴급 회항...9월에도 센서 과열로 문제”, 조선비즈, 2016.10.06.
- [6] 강기준, “테슬라 자율주행차 사고 태양 ‘역광’ 탓 ... 센서 오작동”, 머니투데이, 2018.04.10.
- [7] Ming Liu, Xibin Cao, and Peng Shi, “Fuzzy-Model-Based Fault-Tolerant Design for Nonlinear Stochastic Systems Against Simultaneous Sensor and Actuator Faults,” IEEE Transactions on Fuzzy Systems, vol. 21, no. 5, pp. 789-799, 2015.
- [8] Gilbert Hock Beng Foo, Xinan Zhang, and D. M. Vilathgamuwa, “A Sensor Fault Detection and Isolation Method in Interior

Permanent-Magnet Synchronous Motor Drives Based on an Extended Kalman Filter,” IEEE Transactions on Electronics, vol. 60, no 8, pp. 3485-3495, 2013.

[9] H. Monsef, "Fuzzy rule-based expert system for power system fault diagnosis", Proc. Inst. Elect. Eng. Gen. Transm. Distrib., vol. 144, no. 2, pp. 186-192, Mar. 1997.

[10] N.G. Nikolaou, I.A. Antoniadis, "Rolling element bearing fault diagnosis using wavelet packets", NDT&E International, vol. 35, pp. 197-205, 2002

[11] X.F. Fan, M.J. Zuo, "Gearbox fault detection using Hilbert and wavelet packet transform", Mechanical Systems and Signal Processing, vol. 20, pp. 966-982, 2006.

[12] B. Samanta, "Gear fault detection using artificial neural networks and support vector machines with genetic algorithms," Mech. Syst. Signal Process., vol. 18, no. 3, pp. 625-644, 2004.

[13] T. W. Rauber, F. De Assis Boldt, and F.M. Varejão, "Heterogeneous feature models and feature selection applied to bearing fault diagnosis," IEEE Trans. Ind. Electron., vol. 62, no.1, pp. 637-646, 2015.

[14] 이승우, 김현석, "AI 연구에 2.2兆 투자... 전문가 5000명 키운다", 한국경제신문, 2018.05.16.

[15] Wei Dai and Wei Ji, "A MapReduce Implementation of C4.5 Decision Tree Algorithm", International Journal of Database Theory and Application, vol 7. no. 1, pp. 49-60, 2014.

[16] Nam-Young Lee, "A Logistic Regression for Random Noise Removal in Image Deblurring", Journal of Korea Multimedia Society,

vol. 20, pp. 1671-1677, 2017.

[17] Sana Ullah Jan, Young Doo Lee, Jungpil Shin and Insoo Koo, "Sensor Fault Classification Based on Support Vector Machine and Statistical Time-Domain Features", IEEE Access, Vol. 5, pp. 8682-8690, 2017.

[18] Jae-Wan Yang, Young-Doo Lee, In-Soo Koo, "Sensor Fault Detection Scheme based on Deep Learning and Support Vector Machine", The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 18, No. 2, pp. 185-195, 2018.

[19] Min Meng, Yiting jacqueline Chua, Erwin Wouterson, and Chin Peng Kelvin Ong, "Ultrasonic signal classification and imaging system for composite materials via deep convolutional neural networks", Neurocomputing, Vol. 257, pp. 128-135, 2017.

[20] Donghyun Lee, Minkyu Lim, Hosung Park and Ji-Hwan Kim, "LSTM RNN-based Korean Speech Recognition System Using CTC", Journal of Digital Contents Society, vol. 18, no. 1, pp. 93-99, 2017.

[21] Dong-wook, Ha, Ki-tae Kang and Yeonseung Ryu, "Detecting Insider Threat Based on Machine Learning: Anomaly Detection Using RNN Autoencoder", Journal of The Korea Institute of Information Security & Cryptology, vol. 27, no. 4, pp. 763-773, 2017.

[22] Dean Lee, Vincent Siu, Rick Cruz, and Charles Yetman, "Convolutional Neural Net and Bearing Fault Analysis", in Proc. Int. Conf. Data Min., Las Vegas, NV, USA, 2016, pp. 194-.200.

[23] 김승연, 정용주, "처음 배우는 머신러닝", 한빛미디어, 59-73쪽, 2017.

[24] Hyun-Suk Kang, Young-Seok Lee, "Support Vector Machines-based classification of video file fragments", Journal of the

Korea Academia-Industrial cooperation Society, vol. 16, no. 1, pp. 652-657, 2015.

[25] Hong-Mo Je, Sung-Yang Bang, "Improving SVM Classification by Constructing Ensemble", Korean Institute of Information Scientists and Engineers, vol. 30, no. 3, pp. 251-258, 2003.

[26] 나킬 부두마, "딥러닝의 정석", 한빛미디어, 17-53쪽, 2017.

[27] D. P. Kingma and M. Welling. "Auto-encoding variational bayes", arXiv:1312.6114, 2013.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012. 1, 2, 3, 5

[29] David E. Goldberg and John H. Holland, "Genetic Algorithms and Machine Learning", Machine Learning, vol. 3, no 2-3, pp. 95-99, 1988.

[30] Juhee Lee, Sanghwan Lee and Kyoungwoo Park, "Global Shape Optimization of Airfoil Using Multi-objective Genetic Algorithm", The Korean Society of Mechanical Engineers, vol. 29, no. 10, pp. 1163-1171, 2005.

[31] Yoon Sik Cho, Jeom Kee Paik, "Optimal Design of Submarine Pressure Hull Structures Using Genetic Algorithm", Journal of the Society of Naval Architects of Korea, vol. 54, no. 5, pp. 378-386, 2017.

[32] N. Soni, T. Kumar, "Study of various crossover operators in genetic algorithms", International Journal of Computer Science and Information Technologies, vol. 5, pp. 7235-7238, 2014.

[33] Min Xia, Teng Li, Lin Xu, Lizhi Liu and Clarence W. de Silva, "Fault Diagnosis for Rotating Machinery Using Multiple Sensors and Convolutional Neural Networks", IEEE/ASME Trans Mehatron, vol. 23, pp. 101-110, 2018.

[34] Y. Xie, T. Zhang, "A fault diagnosis approach using SVM with data dimension reduction by PCA and LDA method", Chinese Automation Congress (CAC), pp. 869-874, 2015.

A study on Realtime Sensor Fault Detection Scheme based on Deep Learning and Support Vector Machine

Jae-Wan Yang
School of Electrical Engineering,
The Graduate School,
University of Ulsan
Supervised by Prof. In-Soo Koo

ABSTRACT

With recent development of automation technology and artificial intelligence technology, which is the core technology of the fourth industrial revolution, process machines in the industrial field perform not only simple repetitive operation but also operation requiring advanced control technology. Therefore, interest in the management and maintenance of automation machines is growing. Most automation systems are controlled based on data collected from various sensors. Therefore, if there are faults in sensors attached to automation machines, it can lead to massive economic loss as well as human injury due to malfunction of the machine. To prevent this, a system that detects sensor fault in real time is essential.

In general, researches using signal processing and analysis techniques or rule-based expert systems have been actively studied in the field of fault diagnosis of machines such as turbines, motors, and bearings. However, it is difficult to determine the fault condition of the sensor itself by applying the existing research method, and it is difficult to add or modify the algorithm when a new fault type occurs.

In order to solve this problem, sensor fault diagnosis and type are classified using Support Vector Machine(SVM) which is a machine learning method and Convolution Neural Network(CNN) which is one of deep learning method in this paper. Also, a real-time monitoring system using CNN classifier was designed.

In fault diagnosis using SVM, genetic algorithms are applied to features which are used in learning to improve accuracy efficiency (time-domain statistical features, statistical features based on auto-encoder and latent variable of variational auto-encoder) for feature selection and a multi-layer SVM was designed to classify sensor faults. In order to improve the performance of the classifier using CNN, pre-training was performed using Convolutional Auto-Encoder (CAE). After that, decoder part was removed and fine-tuning was performed after fully connected layer was connected. In real-time monitoring system implementation, the sensor data is sent to the server, the server detects the fault with CNN, stores the data in the database, and displays the result on the web in real time.