

석사학위 청구논문

지도교수 이 재 우

심층 학습 기반 공학 설계

NACA 에어포일 설계

심층신경망 학습

2019년 8월

건국대학교 대학원

항공우주정보시스템공학과

최 철 균

심층 학습 기반 공학 설계

NACA 에어포일 설계

심층신경망 학습

Deep Learning based Engineering Design

NACA Airfoil Design DNN Training

이 논문을 공학 석사학위 청구논문으로 제출합니다.

2019년 5월

건국대학교 대학원

항공우주정보시스템공학과

최 철 균

최철균의 공학 석사학위 청구논문을 인준함.

심사위원장

변 영 환

(인)

심사위원

이 재 우

(인)

심사위원

김 상 호

(인)

2019년 6월

건국대학교 대학원

목 차

목 차	I
표 목 차	IV
그 림 목 차	XIII
ABSTRACT	XXII
제 1 장 서 론	1
제 1절 연구의 필요성 및 목적	1
1. 연구의 필요성	1
2. 연구 목적	4
3. 연구 방법	4
제 2 장 이론적 배경	8
제 1절 심층신경망(Deep Neural Network)	8
1. 기계학습(Machine Learning)	8
2. 인공신경망(Artificial Neural Network)	16
3. 딥러닝(Deep Learning)	33
4. 정리	52
제 2절 에어포일 설계	53
1. 에어포일	53
2. 에어포일 설계	55
제 3 장 심층학습 기반 NACA 4-DIGIT 에어포일 설계	61
제 1절 NACA 4-digit 에어포일	61
제 2절 딥러닝 기반 NACA 4-digit 에어포일 설계 인공지능 학습 프로세스	68
1. 딥러닝 기법 선정과 입/출력 선정	70
2. 초기 데이터 획득	73

3. 인공지능 학습 모델 구축	80
4. 초기 하이퍼 파라미터 최적화 (Initial Hyper-parameter Optimization, IHO)	88
5. 데이터베이스 강화 루프(Database Enhancement Loop, DEL).....	116
6. 최종 하이퍼 파라미터 최적화 (Final Hyperparameter Optimization, FHO).....	151
7. 최적 하이퍼 파라미터를 이용한 학습	174
 제 3절 DLED NACA 4-digit Design 사용자 알고리즘	181
1. Python Script 기반 알고리즘	183
2. 에어포일 데이터베이스 분석	184
3. 설계 가능 영역 설정	186
4. GUI 및 실행파일 생성	192
 제 4 장 결론 및 향후 계획	194
 참 고 문 헌	197
 APPENDIX A. DATABASE ENHANCEMENT LOOP(DEL) TRAINING RESULT: NACA 4-DIGIT AIRFOIL CASE.....	205
 APPENDIX B. DEL LOOP 별 정답 데이터의 SCATTER MATRIX	215
 APPENDIX C. DEL LOOP 별 파라미터 오차의 정규분포와 표준정규분포	218
1. 평균과 표준편차	218
2. 정규분포도와 표준 정규분포도 ($\mu \pm 6\sigma$).....	226
3. $3\sigma, 6\sigma$	238
 APPENDIX D. DEL LOOP 별 형상 그룹에 따른 오차.....	248
1. MSE	248
2. Average of Errors.....	254
3. Standard Deviation of Errors.....	260

APPENDIX E. 최종 하이퍼 파라미터 최적화(FHO): 평균과 표준편차	266
1. Epoch Test	266
2. Number of Hidden Layers Test	269
3. Number of Neurons per Hidden Layer Test	272
국문초록	276

표 목 차

<표 2-1> 기본 활성화 함수	17
<표 2-2> 인공 뉴런의 입력 값에 따른 예측 값	18
<표 2-3> 학습 전후 예측값 변화	20
<표 2-4> 단일 인공 뉴런 예시	27
<표 2-5> 두 개의 인공 뉴런을 이용한 2층 인공신경망 예시	27
<표 3-1> [4]의 풍동실험 오차.....	65
<표 3-2> [4]의 지지대로 인한 풍동실험 오차.....	65
<표 3-3> DLED NACA 4-DIGIT 출력 파라미터	72
<표 3-4> DLED NACA 4-DIGIT 입력 파라미터 (AT AOA = 0 DEG.)	72
<표 3-5> DLED NACA 4-DIGIT 알고리즘의 XFOIL 해석 조건	77
<표 3-6> XFOIL 해석 경우의 수.....	78
<표 3-7> 초기 데이터 설정 범위	79
<표 3-8> 초기 학습을 위한 인공신경망 모델 구조	81
<표 3-9> 개발 환경	82
<표 3-10> TRAINING DEVICE 1 (TD1) SPECIFICATION	83
<표 3-11> TRAINING DEVICE 2 (TD2) SPECIFICATION	83
<표 3-12> DLED에 사용된 GPGPU 환경	84
<표 3-13> 프로세서에 따른 학습 속도 차이	85
<표 3-14> 유동 조건	86
<표 3-15> 유동 조건에 따른 초기 에어포일 데이터 수	86
<표 3-16> 유동 조건에 따른 학습 결과 차이	87
<표 3-17> 15회 학습 TEST ERROR 및 최소 TEST ERROR EPOCH 결과.	90
<표 3-18> 최소 테스트 오차의 최소값, 평균값, 최대값과 표준편차	92
<표 3-19> 최소 테스트 오차와 최소 테스트 오차의 표준 정규분포 값 .	93
<표 3-20> 예시 데이터와 예시 데이터의 표준 정규분포 값	94

<표 3-21> 최종 테스트 오차의 최소값, 평균값, 최대값과 표준편차	95
<표 3-22> 최종 테스트 오차와 최종 테스트 오차의 표준 정규분포 값 ..	95
<표 3-23> INITIAL HYPER-PARAMETER OPTIMIZATION(IHO): 은닉층 수 TEST CASES.....	97
<표 3-24> IHO: 은닉층 TEST 결과.....	98
<표 3-25> IHO: 은닉층 당 뉴런 수 TEST CASES	103
<표 3-26> IHO: 은닉층 당 뉴런 수 TEST 결과.....	104
<표 3-27> IHO: 은닉층 TEST CASE 별 학습 시간	110
<표 3-28> IHO: 은닉층 당 뉴런 수 별 학습 시간.....	110
<표 3-29> 초기 하이퍼 파라미터 최적화: 최적화 전/후 비교	111
<표 3-30> 최적 하이퍼 파라미터에서 15회 학습 TEST ERROR 및 최소 TEST ERROR EPOCH 결과	111
<표 3-31> 하이퍼 파라미터 최적화 전후 최소 테스트 오차의 최소값, 평 균값, 최대값과 표준편차 비교	113
<표 3-32> 하이퍼 파라미터 최적화 전후 최종 테스트 오차의 최소값, 평 균값, 최대값과 표준편차 비교	114
<표 3-33> 최적화된 하이퍼 파라미터로 학습한 모델들의 최소/최종 테스 트 오차	115
<표 3-34> DATABASE ENHANCEMENT LOOP로 증가한 에어포일 데이터 증가량	120
<표 3-35> DEL: 최소 테스트 오차와 최종 테스트 오차 그리고 최종 테스 트 오차의 EPOCH.....	121
<표 3-36> 학습 소요 시간과 테스트 결과의 XFOIL 해석 시간, 그리고 보정된 학습 소요 시간	123
<표 3-37> DLE LOOP 20: 15회 학습 TEST ERROR 및 최소 TEST ERROR EPOCH 결과	124
<표 3-38> 데이터베이스 강화 전(LOOP 1)과 후(LOOP 20)의 최소 테스트	

오차의 최소값, 평균값, 최대값과 표준편차 비교	126
<표 3-39> 데이터베이스 강화 전(LOOP 1)과 후(LOOP 20)의 최종 테스트 오차의 최소값, 평균값, 최대값과 표준편차 비교	126
<표 3-40> 각 LOOP의 인공신경망 모델 테스트 결과.....	129
<표 3-41> DEL: LOOP에 따른 인공지능 테스트 오차 평균: 형상 파라미 터 (LOOP 1과 LOOP 20).....	134
<표 3-42> DEL: LOOP에 따른 인공지능 테스트 오차 평균: 성능 파라미 터 (LOOP 1과 LOOP 20).....	134
<표 3-43> DEL: LOOP에 따른 인공지능 테스트 오차 표준편차: 형상 파 라미터 (LOOP 1과 LOOP 20).....	134
<표 3-44> DEL: LOOP에 따른 인공지능 테스트 오차 표준편차: 성능 파 라미터 (LOOP 1과 LOOP 20).....	134
<표 3-45> Σ 에 따른 내부 범위에 있을 확률과 바깥 범위에 있을 확률 [59]	135
<표 3-46> DEL: LOOP에 따른 인공지능 테스트 오차 3Σ : 형상 파라미터 (LOOP 1과 LOOP 20).....	136
<표 3-47> DEL: LOOP에 따른 인공지능 테스트 오차 3Σ : 성능 파라미터 (LOOP 1과 LOOP 20).....	136
<표 3-48> DEL: LOOP에 따른 인공지능 테스트 오차 ($M\pm3\Sigma$): 형상 파라 미터 (LOOP 1과 LOOP 20).....	136
<표 3-49> DEL: LOOP에 따른 인공지능 테스트 오차 ($M\pm3\Sigma$): 성능 파라 미터 (LOOP 1과 LOOP 20).....	137
<표 3-50> DEL: LOOP에 따른 인공지능 테스트 오차 6Σ : 형상 파라미터 (LOOP 1과 LOOP 20).....	137
<표 3-51> DEL: LOOP에 따른 인공지능 테스트 오차 6Σ : 성능 파라미터 (LOOP 1과 LOOP 20).....	137
<표 3-52> DEL: LOOP에 따른 인공지능 테스트 오차 ($M\pm6\Sigma$): 형상 파라	

미터 (LOOP 1과 LOOP 20).....	137
<표 3 -53> DEL: LOOP에 따른 인공지능 테스트 오차 ($M\pm6\Sigma$): 성능 파라미터 (LOOP 1과 LOOP 20).....	138
<표 3 -54> 형상 범위 설정	139
<표 3 -55> 형상 범위에 따른 데이터 수	140
<표 3 -56> NORMAL RANGE: 형상 파라미터 MSE	143
<표 3 -57> ZERO MAX. CAMBER: 형상 파라미터 MSE	143
<표 3 -58> OUT OF NORMAL RANGE: 형상 파라미터 MSE.....	143
<표 3 -59> NORMAL RANGE: 성능 파라미터 MSE	144
<표 3 -60> ZERO MAX. CAMBER: 성능 파라미터 MSE	144
<표 3 -61> OUT OF NORMAL RANGE: 성능 파라미터 MSE.....	144
<표 3 -62> NORMAL RANGE: 형상 파라미터 오차 평균	145
<표 3 -63> ZERO MAX. CAMBER: 형상 파라미터 오차 평균.....	145
<표 3 -64> OUT OF NORMAL RANGE: 형상 파라미터 오차 평균	145
<표 3 -65> NORMAL RANGE: 성능 파라미터 오차 평균	145
<표 3 -66> ZERO MAX. CAMBER: 성능 파라미터 오차 평균.....	145
<표 3 -67> OUT OF NORMAL RANGE: 성능 파라미터 오차 평균	145
<표 3 -68> NORMAL RANGE: 형상 파라미터 오차의 표준편차	148
<표 3 -69> ZERO MAX. CAMBER: 형상 파라미터 오차의 표준편차.....	148
<표 3 -70> OUT OF NORMAL RANGE: 형상 파라미터 오차의 표준편차	148
<표 3 -71> NORMAL RANGE: 성능 파라미터 오차의 표준편차	148
<표 3 -72> ZERO MAX. CAMBER: 성능 파라미터 오차의 표준편차.....	148
<표 3 -73> OUT OF NORMAL RANGE: 성능 파라미터 오차의 표준편차	148
<표 3 -74> 최종 하이퍼 파라미터 최적화: EPOCH TEST CASE.....	152
<표 3 -75> FHO – EPOCH TEST: 소요 시간.....	155
<표 3 -76> FHO - EPOCH TEST: 최소 오차가 나타난 TEST CASE와 가장 많이 나타난 TEST CASE(MODE).....	155

<표 3-77> FHO - EPOCH TEST PERCENTILE RANK: MAX. CAMBER.....	155
<표 3-78> FHO - EPOCH TEST PERCENTILE RANK: MAX. CAMBER LOCATION.....	156
<표 3-79> FHO - EPOCH TEST PERCENTILE RANK: MAX. THICKNESS.....	156
<표 3-80> FHO - EPOCH TEST PERCENTILE RANK: CL0.....	156
<표 3-81> FHO - EPOCH TEST PERCENTILE RANK: CLA0	156
<표 3-82> FHO - EPOCH TEST PERCENTILE RANK: CD0	157
<표 3-83> FHO - EPOCH TEST PERCENTILE RANK: CM0.....	157
<표 3-84> FHO - EPOCH TEST PERCENTILE RANK: CMA0.....	157
<표 3-85> 최종 하이퍼 파라미터 최적화(FHO): # OF HIDDEN LAYERS	160
<표 3-86> FHO – HIDDEN LAYER TEST: 소요 시간.....	162
<표 3-87> FHO – HIDDEN LAYER TEST: 최소 오차가 나타난 TEST CASE 와 가장 많이 나타난 TEST CASE(MODE).....	162
<표 3-88> FHO – HIDDEN LAYER TEST PERCENTILE RANK: MAX. CAMBER	162
<표 3-89> FHO – HIDDEN LAYER TEST PERCENTILE RANK: MAX. CAMBER LOCATION	163
<표 3-90> FHO – HIDDEN LAYER TEST PERCENTILE RANK: MAX. THICKNESS	163
<표 3-91> FHO – HIDDEN LAYER TEST PERCENTILE RANK: CL0	163
<표 3-92> FHO – HIDDEN LAYER TEST PERCENTILE RANK: CLA0	163
<표 3-93> FHO – HIDDEN LAYER TEST PERCENTILE RANK: CD0.....	164
<표 3-94> FHO – HIDDEN LAYER TEST PERCENTILE RANK: CM0	164
<표 3-95> FHO – HIDDEN LAYER TEST PERCENTILE RANK: CMA0	164
<표 3-96> 최종 하이퍼 파라미터 최적화: # OF NEURONS PER HIDDEN LAYER	167
<표 3-97> FHO – NEURON TEST: 소요 시간	169
<표 3-98> FHO – NEURON TEST: 최소 오차가 나타난 TEST CASE와 가장	

많이 나타난 TEST CASE(MODE).....	169
<표 3 -99> FHO – NEURON TEST PERCENTILE RANK: MAX. CAMBER..	169
<표 3 -100> FHO – NEURON TEST PERCENTILE RANK: MAX. CAMBER LOCATION.....	170
<표 3 -101> FHO – NEURON TEST PERCENTILE RANK: MAX. THICKNESS	170
<표 3 -102> FHO – NEURON TEST PERCENTILE RANK: CL0.....	170
<표 3 -103> FHO – NEURON TEST PERCENTILE RANK: CLA0.....	170
<표 3 -104> FHO – NEURON TEST PERCENTILE RANK: CD0	171
<표 3 -105> FHO – NEURON TEST PERCENTILE RANK: CM0.....	171
<표 3 -106> FHO – NEURON TEST PERCENTILE RANK: CMA0.....	171
<표 3 -107> 최적 하이퍼 파라미터	174
<표 3 -108> FINAL TRAINING CASE 분류	176
<표 3 -109> FINAL TRAINING: 최소 오차가 나타난 TEST CASE와 가장 많이 나타난 TEST CASE(MODE).....	176
<표 3 -110> FINAL TRAINING PERCENTILE RANK: MAX. CAMBER.....	176
<표 3 -111> FINAL TRAINING PERCENTILE RANK: MAX. CAMBER LOCATION.....	176
<표 3 -112> FINAL TRAINING PERCENTILE RANK: MAX. THICKNESS ..	177
<표 3 -113> FINAL TRAINING PERCENTILE RANK: CL0	177
<표 3 -114> FINAL TRAINING PERCENTILE RANK: CLA0	177
<표 3 -115> FINAL TRAINING PERCENTILE RANK: CD0.....	177
<표 3 -116> FINAL TRAINING PERCENTILE RANK: CM0	178
<표 3 -117> FINAL TRAINING PERCENTILE RANK: CMA0	178
<표 3 -118> FINAL TEST: 형상 오차의 평균	178
<표 3 -119> FINAL TEST: 성능 오차의 평균	179
<표 3 -120> FINAL TEST: 형상 오차의 표준편차	179
<표 3 -121> FINAL TEST: 성능 오차의 표준편차	180

<표 3-122> 21번째 데이터베이스 성능 파라미터의 1.5 IQR 최대/최소값	188
<표 3-123> 성능 파라미터의 최대 최소값	191
<APPENDIX C. 표-1> DEL-LOOP에 따른 테스트 오차 평균: 형상 파라미터	218
<APPENDIX C. 표-2> DEL-LOOP에 따른 테스트 오차 평균: 성능 파라미터	219
<APPENDIX C. 표-3> DEL-LOOP에 따른 테스트 오차 표준편차: 형상 파라미터	220
<APPENDIX C. 표-4> DEL-LOOP에 따른 테스트 오차 표준편차: 성능 파라미터	221
<APPENDIX C. 표-5> DEL LOOP 별 테스트 오차 3Σ : 형상 파라미터	238
<APPENDIX C. 표-6> DEL LOOP 별 테스트 오차 3Σ : 성능 파라미터	239
<APPENDIX C. 표-7> DEL LOOP 별 테스트 오차 범위($M\pm3\Sigma$): 형상 파라미터	240
<APPENDIX C. 표-8> DEL LOOP 별 테스트 오차 범위($M\pm3\Sigma$): 성능 파라미터	241
<APPENDIX C. 표-9> DEL LOOP 별 테스트 오차 6Σ : 형상 파라미터	243
<APPENDIX C. 표-10> DEL LOOP 별 테스트 오차 6Σ : 성능 파라미터	244
<APPENDIX C. 표-11> DEL LOOP 별 테스트 오차 범위($M\pm6\Sigma$): 형상 파라미터	245
<APPENDIX C. 표-12> DEL LOOP 별 테스트 오차 범위($M\pm6\Sigma$): 성능 파라미터	246
<APPENDIX D. 표-1> DEL 형상 그룹 별 MSE: NORMAL RANGE 형상 파라미터	248

<APPENDIX D. 표- 2> DEL 형상 그룹 별 MSE: NORMAL RANGE 성능 파라미터	249
<APPENDIX D. 표- 3> DEL 형상 그룹 별 MSE: ZERO MAX. CAMBER 형상 파라미터	250
<APPENDIX D. 표- 4> DEL 형상 그룹 별 MSE: ZERO MAX. CAMBER 성능 파라미터	251
<APPENDIX D. 표- 5> DEL 형상 그룹 별 MSE: OUT OF NORMAL RANGE 형상 파라미터	252
<APPENDIX D. 표- 6> DEL 형상 그룹 별 MSE: OUT OF NORMAL RANGE 성능 파라미터	253
<APPENDIX D. 표- 7> DEL 형상 그룹 별 오차의 평균: NORMAL RANGE 형상 파라미터	254
<APPENDIX D. 표- 8> DEL 형상 그룹 별 오차의 평균: NORMAL RANGE 성능 파라미터	255
<APPENDIX D. 표- 9> DEL 형상 그룹 별 오차의 평균: ZERO MAX. CAMBER 형상 파라미터	256
<APPENDIX D. 표- 10> DEL 형상 그룹 별 오차의 평균: ZERO MAX. CAMBER 성능 파라미터	257
<APPENDIX D. 표- 11> DEL 형상 그룹 별 오차의 평균: OUT OF NORMAL RANGE 형상 파라미터	258
<APPENDIX D. 표- 12> DEL 형상 그룹 별 오차의 평균: OUT OF NORMAL RANGE 성능 파라미터	259
<APPENDIX D. 표- 13> DEL 형상 그룹 별 오차의 평균: NORMAL RANGE 형상 파라미터	260
<APPENDIX D. 표- 14> DEL 형상 그룹 별 오차의 평균: NORMAL RANGE 성능 파라미터	261
<APPENDIX D. 표- 15> DEL 형상 그룹 별 오차의 평균: ZERO MAX.	

CAMBER 형상 파라미터	262
<APPENDIX D. 표- 16> DEL 형상 그룹 별 오차의 평균: ZERO MAX.	
CAMBER 성능 파라미터	263
<APPENDIX D. 표- 17> DEL 형상 그룹 별 오차의 평균: OUT OF NORMAL RANGE 형상 파라미터	264
<APPENDIX D. 표- 18> DEL 형상 그룹 별 오차의 평균: OUT OF NORMAL RANGE 성능 파라미터	265
<APPENDIX E. 표- 1> FHO EPOCH TEST: EPOCH TEST CASES.....	266
<APPENDIX E. 표- 2> FHO EPOCH TEST: 형상 오차의 평균.....	266
<APPENDIX E. 표- 3> FHO EPOCH TEST: 성능 오차의 평균.....	267
<APPENDIX E. 표- 4> FHO EPOCH TEST: 형상 오차의 표준편차.....	267
<APPENDIX E. 표- 5> FHO EPOCH TEST: 성능 오차의 평균.....	268
<APPENDIX E. 표- 6> FHO HIDDEN LAYER TEST: TEST CASES.....	269
<APPENDIX E. 표- 7> FHO HIDDEN LAYER TEST: 형상 오차의 평균.....	269
<APPENDIX E. 표- 8> FHO HIDDEN LAYER TEST: 성능 오차의 평균.....	270
<APPENDIX E. 표- 9> FHO HIDDEN LAYER TEST: 형상 오차의 표준편차	271
<APPENDIX E. 표- 10> FHO HIDDEN LAYER TEST: 성능 오차의 표준편차	271
<APPENDIX E. 표- 11> FHO HIDDEN LAYER TEST: TEST CASES.....	272
<APPENDIX E. 표- 12> FHO NEURON TEST: 형상 오차의 평균	272
<APPENDIX E. 표- 13> FHO NEURON TEST: 성능 오차의 평균	273
<APPENDIX E. 표- 14> FHO NEURON TEST: 형상 오차의 표준편차	274
<APPENDIX E. 표- 15> FHO NEURON TEST: 성능 오차의 표준편차	275

그 림 목 차

<그림 1 -1> 최적설계 기법과 역설계 기법	3
<그림 1 -2> 딥러닝을 이용한 익형 설계	4
<그림 1 -3> DLED: NACA4 실행 화면	7
<그림 2 -1> 인공지능, 기계학습, 그리고 딥러닝 [6].....	8
<그림 2 -2> 전통적인 프로그래밍 방법 [7].....	9
<그림 2 -3> 기계학습 기법을 이용한 방법 [7].....	9
<그림 2 -4> 붓꽃 분류 예시	10
<그림 2 -5> 지도학습의 예시	12
<그림 2 -6> 비지도 학습의 예시	12
<그림 2 -7> 온라인 학습(ONLINE TRAINING)	13
<그림 2 -8> K-NN 기법을 이용한 데이터 판단	14
<그림 2 -9> 이웃의 수에 따라 달라지는 레이블	15
<그림 2 -10> 모델기반 학습 예시	16
<그림 2 -11> 인공 뉴런	16
<그림 2 -12> 기본적인 활성화 함수 그래프	17
<그림 2 -13> 인공 뉴런 초기화 예시	18
<그림 2 -14> 인공 뉴런의 예측 값(y)과 실제 값(Y).....	19
<그림 2 -15> 인공 뉴런 100회 학습 결과	21
<그림 2 -16> 다중 입력 단일 출력 인공 뉴런	23
<그림 2 -17> 단층 인공신경망	24
<그림 2 -18> 두 개의 층을 갖는 인공신경망	24
<그림 2 -19> 두 개의 층을 갖는 인공신경망 예시	25
<그림 2 -20> 개의 인공 뉴런을 이용한 인공신경망	28
<그림 2 -21> FULLY CONNECTED NEURAL NETWORK 예시	29
<그림 2 -22> 과적합 문제 예시	30

<그림 2-23> 과적합 여부 판단 방법	31
<그림 2-24> K-NN의 과적합, 과소적합 예시	32
<그림 2-25> HOPFIELD NETWORKS, BOLTZMANN MACHINE, RBM.....	34
<그림 2-26> DEEP BELIEF NETWORK에서의 RESTRICTED BOLTZMANN MACHINE	35
<그림 2-27> SIGMOID를 사용한 인공신경망을 RANDOM INITIALIZATION을 실시했을 때 각 층의 ACTIVATION 값의 평균(가 로선)과 표준편차(세로선) [15]	38
<그림 2-28> SIGMOID FUNCTION의 SATURATING이 일어나는 지점	38
<그림 2-29> NORMALIZED INITIALIZATION 기법을 사용하지 않았을 때 (상)와 사용하였을 때(하), 각 레이어의 역전파 그래디언트 분포.....	40
<그림 2-30> RELU 함수와 RELU 계열 함수들	43
<그림 2-31> 타원 문제에서의 MOMENTUM 항의 유무 차이	47
<그림 2-32> MOMENTUM OPTIMIZER와 NAG 기법의 비교	48
<그림 2-33> 에어포일에 작용하는 힘	53
<그림 2-34> 에어포일 형상 구성도	54
<그림 2-35> UIUC AIRFOIL DATABASE [2].....	56
<그림 2-36> 최적설계 기법을 이용한 에어포일 설계 프로세스	59
<그림 2-37> 에어포일 역설계 과정	60
<그림 3-1> 비대칭형 익형 생성 방법 [4].....	63
<그림 3-2> [4]에서 만든 NACA 4-DIGIT 에어포일 형상 목록	64
<그림 3-3> DLED NACA 4-DIGIT 학습 프로세스	68
<그림 3-4> DLED NACA 4 DIGIT.....	71
<그림 3-5> 성능 해석 기법에 따른 E387 에어포일의 공력계수 [41]	75
<그림 3-6> 성능 해석 기법에 따른 S1223 에어포일의 공력 계수 [41]..	75
<그림 3-7> NACA 2412의 2D, 3D 실험값과 XFOIL 해석 값.....	76
<그림 3-8> XFOIL 해석 결과 파일.....	77

<그림 3-9> 초기 학습을 위한 DLED NACA 4-DIGIT 인공신경망 구조 (실 제로는 FULLY CONNECTED 구조이다).....	80
<그림 3-10> 프로세서에 따른 학습 정도 차이	85
<그림 3-11> 유동 조건에 따른 학습 정도 차이	86
<그림 3-12> DLED NACA 4-DIGIT 학습 프로세스 입력파일 (좌: INPUT.TXT, 우: XFOIL SETTING.TXT).....	87
<그림 3-13> 15회 학습 TEST ERROR 및 최소 TEST ERROR EPOCH 결과	90
<그림 3-14> 최소 테스트 오차의 정규 분포도	92
<그림 3-15> 최소 학습 오차와 예시 데이터의 표준 정규분포도	93
<그림 3-16> 최종 테스트 오차의 정규분포도	95
<그림 3-17> 최종 테스트 오차와 예시 데이터의 표준 정규분포도	96
<그림 3-18> 하이퍼 파라미터 TEST CASE의 오차와 오차 범위 예시.....	96
<그림 3-19> IHO - HIDDEN LAYER TEST: 최소/최종 테스트 오차	98
<그림 3-20> IHO - HIDDEN LAYER TEST: 최소오차 EPOCH	99
<그림 3-21> IHO – HIDDEN LAYER TEST 학습/테스트 오차: 은닉층 수 3 개	99
<그림 3-22> IHO – HIDDEN LAYER TEST 학습/테스트 오차: 은닉층 수 4 개	100
<그림 3-23> IHO – HIDDEN LAYER TEST 학습/테스트 오차: 은닉층 수 5 개(BASELINE).....	100
<그림 3-24> IHO – HIDDEN LAYER TEST 학습/테스트 오차: 은닉층 수 6 개	101
<그림 3-25> IHO – HIDDEN LAYER TEST 학습/테스트 오차: 은닉층 수 7 개	101
<그림 3-26> IHO – HIDDEN LAYER TEST 학습/테스트 오차: 은닉층 수 8 개	102

<그림 3-27> IHO – HIDDEN LAYER TEST 학습/테스트 오차: 은닉층 수 9 개	102
<그림 3-28> IHO – HIDDEN LAYER TEST 학습/테스트 오차: 은닉층 수 10 개	103
<그림 3-29> IHO – NUMBER OF NEURONS TEST: 최소/최종 테스트 오차	105
<그림 3-30> IHO – NUMBER OF NEURONS TEST: 최소 테스트 오차 EPOCH.....	105
<그림 3-31> IHO – NUMBER OF NEURONS TEST: 4 NEURONS PER HIDDEN LAYER.....	106
<그림 3-32> IHO – NUMBER OF NEURONS TEST: 50 NEURONS PER HIDDEN LAYER.....	106
<그림 3-33> IHO – NUMBER OF NEURONS TEST: 100 NEURONS PER HIDDEN LAYER.....	107
<그림 3-34> IHO – NUMBER OF NEURONS TEST: 300 NEURONS PER HIDDEN LAYER (BASELINE).....	107
<그림 3-35> IHO – NUMBER OF NEURONS TEST: 400 NEURONS PER HIDDEN LAYER.....	108
<그림 3-36> IHO – NUMBER OF NEURONS TEST: 500 NEURONS PER HIDDEN LAYER.....	108
<그림 3-37> IHO – NUMBER OF NEURONS TEST: 600 NEURONS PER HIDDEN LAYER.....	109
<그림 3-38> IHO – NUMBER OF NEURONS TEST: 700 NEURONS PER HIDDEN LAYER.....	109
<그림 3-39> 최적 하이퍼 파라미터에서 15회 학습 TEST ERROR 및 최소 TEST ERROR EPOCH 결과.....	112
<그림 3-40> 하이퍼 파라미터 최적화 전후의 최소 테스트 오차 분포.....	113
<그림 3-41> 최적화된 하이퍼 파라미터에서의 최종 테스트 오차 정규분	

포	114
<그림 3-42> 데이터베이스 강화 루프 개념도	116
<그림 3-43> 데이터베이스 강화 예시	117
<그림 3-44> DLED TRAINING PROCESS	118
<그림 3-45> DATABASE ENHANCEMENT LOOP로 증가한 에어포일 데이터 수	119
<그림 3-46> DEL: 최소 테스트 오차와 최종 테스트 오차	121
<그림 3-47> DEL: 소요 시간 비교	123
<그림 3-48> DEL LOOP 20: 15회 학습 TEST ERROR/최소 TEST ERROR EPOCH 결과	125
<그림 3-49> LOOP 20의 최소 테스트 오차 정규분포도	127
<그림 3-50> LOOP 20의 최소 테스트 오차 표준 정규분포도	127
<그림 3-51> LOOP 20의 최종 테스트 오차 정규분포도	128
<그림 3-52> LOOP 20의 최종 테스트 오차 표준 정규분포도	128
<그림 3-53> 각 LOOP의 인공신경망 모델 테스트 결과	130
<그림 3-54> DEL LOOP 1 정답 데이터의 SCATTER MATRIX	131
<그림 3-55> DEL LOOP 20 정답 데이터의 SCATTER MATRIX	131
<그림 3-56> 인공신경망 학습 단계에서의 오차 측정 방법	133
<그림 3-57> 학습된 인공신경망의 예측 형상 성능 측정 방법	133
<그림 3-58> 형상 범위 그룹에 따른 데이터 수	139
<그림 3-59> 형상 범위 그룹에 따라 데이터가 차지하는 비율	140
<그림 3-60> LOOP에 따른 그룹 별 최대 캠버 MSE 변화	141
<그림 3-61> LOOP에 따른 그룹 별 최대 캠버 위치 MSE 변화	142
<그림 3-62> LOOP에 따른 그룹 별 최대 캠버 위치 MSE 변화(LOG10 SCALE)	142
<그림 3-63> LOOP에 따른 그룹 별 최대 두께 MSE 변화	143
<그림 3-64> LOOP에 따른 그룹 별 오차의 평균 변화: 최대 캠버	146

<그림 3-65> LOOP에 따른 그룹 별 오차의 평균 변화: 최대 캠버 위치	146
<그림 3-66> LOOP에 따른 그룹 별 오차의 평균 변화: 최대 두께	147
<그림 3-67> LOOP에 따른 그룹 별 오차의 표준편차 변화: 최대 캠버	149
<그림 3-68> LOOP에 따른 그룹 별 오차의 표준편차 변화: 최대 캠버 위 치	149
<그림 3-69> LOOP에 따른 그룹 별 오차의 표준편차 변화: 최대 캠버 위 치 (LOG10 SCALE)	150
<그림 3-70> LOOP에 따른 그룹 별 오차의 표준편차 변화: 최대 두께	150
<그림 3-71> FHO: EPOCH 수에 따른 형상/성능 오차의 정규분포	153
<그림 3-72> FHO - EPOCH TEST PERCENTILE RANK: MAX. CAMBER, MAX. CAMBER LOCATION, MAX. THICKNESS와 CL0	158
<그림 3-73> FHO - EPOCH TEST PERCENTILE RANK: CLA0, CD0, CM0와 CMA0	159
<그림 3-74> FHO: HIDDEN LAYER 수에 따른 형상/성능 오차의 정규분포	161
<그림 3-75> FHO – HIDDEN LAYER TEST PERCENTILE RANK: MAX. CAMBER, MAX. CAMBER LOCATION, MAX. THICKNESS와 CL0	165
<그림 3-76> FHO – HIDDEN LAYER TEST PERCENTILE RANK: CLA0, CD0, CM0와 CMA0	166
<그림 3-77> FHO: NEURON 수에 따른 형상/성능 오차의 정규분포	168
<그림 3-78> FHO – NEURON TEST PERCENTILE RANK: MAX. CAMBER, MAX. CAMBER LOCATION, MAX. THICKNESS와 CL0	172
<그림 3-79> FHO – NEURON TEST PERCENTILE RANK: CLA0, CD0, CM0 와 CMA0	173
<그림 3-80> DEL 전, DEL 후, DEL과 파라미터 최적화 후의 정규분포	175
<그림 3-81> 학습이 완료된 알고리즘을 사용하는 방법	181
<그림 3-82> TUI 예시(파일관리자) [61]	182

<그림 3-83> COMMAND-LINE INTERFACE인 XFOIL [5].....	182
<그림 3-84> 학습이 완료된 인공신경망 모델 파일	183
<그림 3-85> 학습이 완료된 모델을 사용하는 방법	183
<그림 3-86> LOOP 1 에어포일 DB의 SCATTER MATRIX.....	184
<그림 3-87> LOOP 20 에어포일 DB의 SCATTER MATRIX.....	185
<그림 3-88> NACA 4-DIGIT CL0 VS CM0 관계(좌) CLA0 VS CMA0 관계 (우)	186
<그림 3-89> 상자수염 그래프 구성 [63].....	187
<그림 3-90> 21번째 데이터베이스의 성능 파라미터 상자수염 그래프 ..	188
<그림 3-91> AIRFOIL DB 21의 SCATTER MATRIX.....	189
<그림 3-92> 조건에 맞추어 걸러낸 AIRFOIL DB 21 SCATTER MATRIX	190
<그림 3-93> CLA0 VS CD0 SCATTER PLOT	191
<그림 3-94> CLA0 VS CMA0 SCATTER PLOT	191
<그림 3-95> DLED: NACA 4-DIGIT (기본 화면).....	192
<그림 3-96> DLED: NACA 4-DIGIT (RUN XFOIL을 누른 화면)	193
<APPENDIX A. 그림-1> DEL: NACA 4-DIGIT AIRFOIL – LOOP 1	205
<APPENDIX A. 그림-2> DEL: NACA 4-DIGIT AIRFOIL – LOOP 2	205
<APPENDIX A. 그림-3> DEL: NACA 4-DIGIT AIRFOIL – LOOP 3	206
<APPENDIX A. 그림-4> DEL: NACA 4-DIGIT AIRFOIL – LOOP 4	206
<APPENDIX A. 그림-5> DEL: NACA 4-DIGIT AIRFOIL – LOOP 5	207
<APPENDIX A. 그림-6> DEL: NACA 4-DIGIT AIRFOIL – LOOP 6	207
<APPENDIX A. 그림-7> DEL: NACA 4-DIGIT AIRFOIL – LOOP 7	208
<APPENDIX A. 그림-8> DEL: NACA 4-DIGIT AIRFOIL – LOOP 8	208
<APPENDIX A. 그림-9> DEL: NACA 4-DIGIT AIRFOIL – LOOP 9	209
<APPENDIX A. 그림-10> DEL: NACA 4-DIGIT AIRFOIL – LOOP 10	209
<APPENDIX A. 그림-11> DEL: NACA 4-DIGIT AIRFOIL – LOOP 11	210

<APPENDIX A. 그림- 12> DEL: NACA 4-DIGIT AIRFOIL – LOOP 12	210
<APPENDIX A. 그림- 13> DEL: NACA 4-DIGIT AIRFOIL – LOOP 13	211
<APPENDIX A. 그림- 14> DEL: NACA 4-DIGIT AIRFOIL – LOOP 14	211
<APPENDIX A. 그림- 15> DEL: NACA 4-DIGIT AIRFOIL – LOOP 15	212
<APPENDIX A. 그림- 16> DEL: NACA 4-DIGIT AIRFOIL – LOOP 16	212
<APPENDIX A. 그림- 17> DEL: NACA 4-DIGIT AIRFOIL – LOOP 17	213
<APPENDIX A. 그림- 18> DEL: NACA 4-DIGIT AIRFOIL – LOOP 18	213
<APPENDIX A. 그림- 19> DEL: NACA 4-DIGIT AIRFOIL – LOOP 19	214
<APPENDIX A. 그림- 20> DEL: NACA 4-DIGIT AIRFOIL – LOOP 20	214
<APPENDIX C. 그림- 1> LOOP에 따른 오차의 평균: 최대 캠버	222
<APPENDIX C. 그림- 2> LOOP에 따른 오차의 평균: 최대 캠버 위치	222
<APPENDIX C. 그림- 3> LOOP에 따른 오차의 평균: 최대 두께	223
<APPENDIX C. 그림- 4> LOOP에 따른 오차의 평균: 양력계수	223
<APPENDIX C. 그림- 5> LOOP에 따른 오차의 평균: 양력계수 기울기	224
<APPENDIX C. 그림- 6> LOOP에 따른 오차의 평균: 항력계수	224
<APPENDIX C. 그림- 7> LOOP에 따른 오차의 평균: 모멘트 계수	225
<APPENDIX C. 그림- 8> LOOP에 따른 오차의 평균: 모멘트 계수 기울기	225
<APPENDIX C. 그림- 9> DEL LOOP 1: 형상/성능 오차 정규분포	226
<APPENDIX C. 그림- 10> DEL LOOP 1: 형상/성능 오차 표준정규분포	227
<APPENDIX C. 그림- 11> DEL LOOP 2: 형상/성능 오차 정규분포	228
<APPENDIX C. 그림- 12> DEL LOOP 2: 형상/성능 오차 표준정규분포	229
<APPENDIX C. 그림- 13> DEL LOOP 5: 형상/성능 오차 정규분포	230
<APPENDIX C. 그림- 14> DEL LOOP 5: 형상/성능 오차 표준정규분포	231
<APPENDIX C. 그림- 15> DEL LOOP 10: 형상/성능 오차 정규분포	232
<APPENDIX C. 그림- 16> DEL LOOP 10: 형상/성능 오차 표준정규분포	233
<APPENDIX C. 그림- 17> DEL LOOP 15: 형상/성능 오차 정규분포	234

- <APPENDIX C. 그림- 18> DEL LOOP 15: 형상/성능 오차 표준정규분포...235
<APPENDIX C. 그림- 19> DEL LOOP 20: 형상/성능 오차 정규분포.....236
<APPENDIX C. 그림- 20> DEL LOOP 20: 형상/성능 오차 표준정규분포...237

ABSTRACT

Deep Learning based Engineering Design

NACA Airfoil Design DNN Training

Choi, Cheol kyun

Aerospace Information Engineering Department

Konkuk University

Deep Learning is a large-scale artificial neural network which overcomes the problems of gradient vanishing and overfitting which were limitations of traditional Artificial Neural Network (ANN)

Deep Learning began to receive the spotlight after it got a tremendous score in Computer Visioning (CV) that traditional CV algorithms such as the Support Vector Machine(SVM) or K-Nearest Neighborhood(K-NN) couldn't get before. In recent years, almost all image classification algorithms use deep learning, and it is used in various fields such as image recognition and processing, classification, and autonomous navigation.

Machine learning is basically used to show the relationship between data, and deep learning has a very good performance among other machine learning techniques. This study is based on that deep learning shows relationship very well between complex data, makes deep learning algorithms to do an engineering design. Engineering design refers to the process of how to make a mechanical device that meets a given performance requirement. Most of the design techniques and tools currently used follow the iterative process to reduce the error of the requirements using the performance analysis tools in the design process. The reason why these tools and methods have to repeat the analysis (check the performance error and correct the shape) is that the equations we have are based on the natural laws and are the methods to measure the performance against the shape. If an expression can be deriving the shape

using the performance indicators, the iterative process in the design will be reduced and accelerated.

In this study, airfoil design is used as an example among various design examples. Airfoil is a mechanical device that creates lift forces using wind and is a key device that enables airplanes such as airliners and helicopters to fly. Therefore, airfoil design is the key and foundation of aircraft design. The design of an aircraft begins with an analysis of the requirements and a conceptual design that draws a sketch based on this analysis. The concept design starts from the design of large parts such as fuselages, propulsion systems and wings to satisfy the given requirements and the latest trends. Among them, the wing design which generates the lifting force of the aircraft determines the shape, width, chord length, airfoil type and the like so as to satisfy required performance such as lifting force, drag force, and moment. Of the various shape elements of the wing, the airfoil has the greatest influence because it is a device designed to generate lift. Therefore, airfoil design is a fundamental element and an important part of the aircraft design that aerospace engineering students learn first. There are a variety of ways to design airfoils, but the most basic and simple way is to search for airfoils similar to the requirements in the database. Searches are simple to use, but it is hard to find the airfoils that meet the requirements exactly and sometimes may not be found.

In the professional domain, experts use optimization techniques to match the shape to the requirements, along with the searching method. The optimization technique is start from to select the baseline that gives performance similar to the requirement, and then to adjust the performance to the requirement by gradually changing the shape. Optimization techniques can derive more accurate results than search, but for the optimization, the user has to create objective function to express the optimization object and constraint functions using the requirements. In addition, the optimization process is not performed at once, but it changes the shape little by little and satisfies the requirement. Therefore, performance analysis should be performed to check whether the requirements are met each time. If the optimal shape was found only in 100 times, then 100 performance analyzes were performed. This makes design to take a lot of time and resources.

Another specialized method is the inverse engineering technique also known as airfoil inverse design to find the shape based on the exact performance parameters. The inverse design technique is also a technique that reduces the error of the basic shape's performance to satisfy the required performance. The difference from the optimal design is that the optimal design maximizes or minimizes any target performance within the constraints, but the reverse design approach approaches the proposed performance clearly.

In this study, similar to the inverse design method, a clear performance index is used to derive a shape very close to it, but there are two differences. The inverse design technique requires the user to draw the pressure distribution with the required performance. Here, the disadvantage of reverse engineering techniques is that it is not intuitive for designers to use. Since the performance required in aircraft design is mainly parameters of forces and moments, it is necessary to draw back them to pressure distributions. Even if the force and moment by calculating the pressure distribution that get from analysis tools, it is difficult to draw the pressure distribution by applying it in reverse, even though for the experts of airfoil design. Another disadvantage is that it takes a long time to interpret the errors because the process of checking and reducing them is repetitive.

Artificial neural networks and deep learning techniques were introduced to derive shapes more rapidly than the two techniques, based on the performance parameters that are clearer than the optimal design method and more intuitive than the inverse design method. The main reason why the design process takes a long time is that there is no definite relation expression that can derive the shape based on the performance. Using artificial neural networks, artificial neural networks themselves will find the relationship between performance and shape data without having to find a relationship between the two by human.

However, the training of artificial neural networks is based on vast amounts of data, and it is not easy to obtain the level of data required in the field of data science in design problems. This is why design techniques have evolved to solve problems based on minimal data. In this study, we developed a method to strengthen training data by training results using small dataset to solve the problem of lack of data in the

engineering design domain. In addition, artificial neural networks were trained by using the same data several times to study training patterns and optimized for various hyperparameters.

At the end of this paper, the predicted results of artificial neural networks were reanalyzed and the error between the performance parameters of input and the predicted shape performance were measured. From these errors, the percentile rank at top 1%, 5%, 10%, 15%, and 20%, the values of errors were obtained and compared. The database enhancement loop reduces percentile rank error by at least 50% and up to 98%, and hyperparameter optimization reduces errors by up to 40%. After the database enhancement loop, the results of hyperparameter optimization showed that the error was reduced by 50 ~ 98% compared to before the database hardening.

For user convenience, experience and accessibility, the artificial neural network model, which is finally completed training, is made independent program with GUI.

제 1 장 서 론

제 1절 연구의 필요성 및 목적

1. 연구의 필요성

공학 설계 문제는 매우 복잡하다. 특히 다양한 부품들과 요소들로 구성될수록 그 난이도는 높아진다. 항공기는 매우 복잡한 기계 중의 하나로, 엔진과 동체, 날개, 전기전자장치 등 수많은 요소들로 구성되어 있다. 그러므로 항공기의 설계는 정밀도와 정확성을 위해 개념설계, 기본설계, 상세설계 등 다양한 단계를 거쳐 진행하게 된다.

개념설계는 밑그림을 그리는 단계로 추상적인 요구조건을 바탕으로 설계를 시작하는데, 이 과정에서 수 많은 형상을 설계하고 시뮬레이션 해본다. 적합한 엔진은 어떤 것인지, 추력 장치는 한 개가 좋을지 두 개가 좋을지, 날개의 모양은 직사각형이어야 할지 삼각형이어야 할지, 꼬리날개는 한 개여야 하는지 등 매우 기초적인 사항들에 대하여 설계를 한다.

항공기의 다양한 설계 요소들 중 날개는 양력을 발생시키기에 핵심적인 요소라 할 수 있다. 날개의 설계를 들여다보면 설계자는 날개의 모양, 길이, 너비, 그리고 단면의 모양 등을 선택해야 한다. 이들 중 설계 형상의 성능에 직접적인 영향을 미치는 것은 단연 단면의 모양인 에어포일(Airfoil)이라 할 수 있다. 양력을 발생시키기 위해 고안된 장치인 에어포일을 선정하는 방법은 크게 최적 설계 기법(Optimum design method)과 역설계 기법(Inverse design method)로 나눌 수 있다. 데이터베이스 검색을 통해 비슷한 성능을 내는 형상을 찾아낼 수 있으나, 항상 만족하는 답을 찾아낸다고 담보할 수 없으며 대중적으로 많이 사용되는 Airfoil Tools [1]나 UIUC Airfoil Database [2]조차도 등록된 에어포일 수는 1,600개에 못 미친다. 최적 설계 기법은 요구된 설계 요구 조건을 최대화하거나 최소화하는 기법이다. 그러나 아무것도 없이 시작할 수 없으므로, 보통 요구 조건과 가장 유사한 형상을 찾아 기본 형상으로 삼은 뒤, 조건에 맞게 최적화한다.

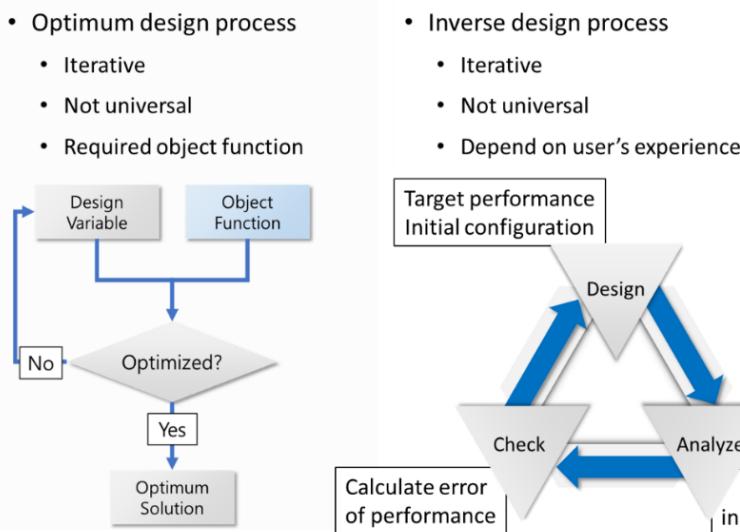
최적화 기법은 사용자가 요구한 조건에 근접한 형상을 만들어낼 수 있다는 장점이 있으나, 명료하지 않은 요구조건, 반복적인 과정으로 인한 시간 소모, 그리고 국지적인 문제에 특화되어 보편적으로 사용할 수 없는 등 사용하기 어려운 점이 있다.

최적설계 기법은 어떤 목표 성능을 최대화하거나 최소화나는 문제가 대부분인데, 이와 같은 부등식 요구조건은 문제를 더 유연하고 강건하며, Equality condition 보다 문제를 쉽게 만들지만, 그 목표가 명료하지 않다. 또한, 최적점을 찾아내는 과정이 형상을 해석하고 요구도와의 오차를 최소화하는 과정을 반복하기 때문에, 변형된 형상을 해석하느라 시간이 오래 걸린다. 반복 횟수는 문제의 난이도와 형상의 변형 정도에 따라 달라지는데, 형상을 크게 변형하면 오히려 오차가 커지거나 줄어들지 않는 경우가 발생하므로 보통 형상의 변형 정도는 작게 한다. 또한, 반복적인 과정을 통해 요구조건에 맞는 형상을 찾아내기 때문에 처음 설계한 문제에 특화되어 있어 보편적으로 사용할 수 없고, 다른 요구 조건을 갖는 에어포일을 설계하려면 기본형상 설정부터 다시 해야 한다. 최근에는 이러한 문제를 해결하기 위해 인공신경망이나 근사모델링(Surrogate modeling)과 같은 기법과 같이 사용하기도 한다.

에어포일 역설계 기법은 설계자가 필요한 에어포일 주변의 압력 분포에 맞는 에어포일을 찾아내는 방법이다. 요구 조건이 압력분포로 최적설계 기법보다 명료한 목표가 있다는 장점이 있지만, 압력분포를 그리는 것은 직관적이지 못하다. 일반적으로 설계 시 요구되는 파라미터들은 힘과 모멘트와 같이 직관적인 성능 파라미터들인데, 형상이 존재하지 않는 상황에서 요구조건인 양력, 항력, 모멘트를 바탕으로 목표 형상의 압력분포를 찾아내는 것은 설계자가 에어포일에 대해 많은 지식과 설계 경험이 있다 할지라도 어려운 작업이다. 즉, 이 기법은 설계자의 역량에 좌우된다.

에어포일 역설계 기법도 최적설계 기법과 마찬가지로 반복적인 과정을 사용한다. 우선 평판이나 요구조건과 비슷한 성능을 내는 에어포일을 기

본 형상으로 채택하고, 이를 해석하여 에어포일 주변의 압력 계수 분포와 요구 압력분포의 오차를 계산한다. 최적화 기법과 마찬가지로 오차를 줄일 수 있는 방향으로 에어포일 형상을 변형한다. 형상 변형과 해석, 오차 확인은 압력 분포와 요구 조건 오차가 일정 수준에 도달할 때까지 과정을 반복된다. 그러므로 역설계 기법 역시 해석에 많은 시간이 소요되며, 다른 요구조건을 갖는 에어포일을 설계하려면 기본형상 설정부터 다시 시작해야 한다. 또한 역설계 기법은 답을 찾아내지 못하는 경우도 있다.

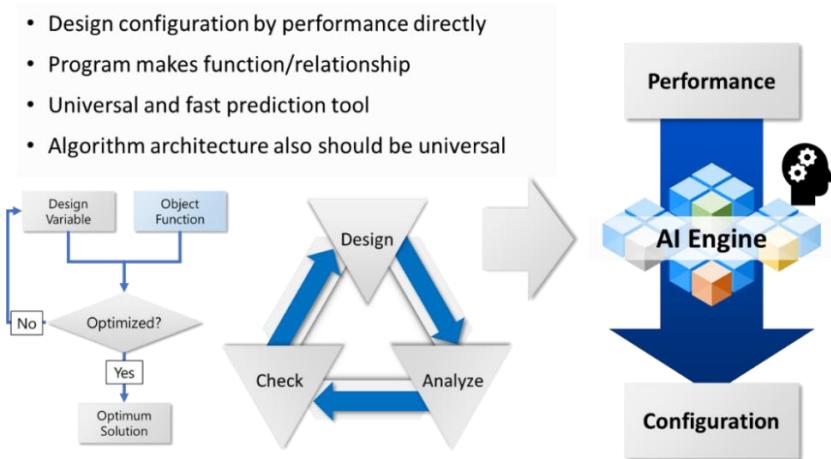


<그림 1-1> 최적설계 기법과 역설계 기법

이처럼 기존 에어포일 설계 방법들은 시간이 많이 소요되며, 보편적인 사용이 어려우며, 명료하지 못하거나 직관적이지 못한 단점들이 있다. 설계가 더 명료하고 직관적인 조건을 바탕으로 쉽고 빠르게 할 수 있다면, 시간과 비용의 절감은 물론, 설계자의 상상력과 실행력에 날개를 달아주며, 더 나아가 실시간 설계가 가능하다면 환경 변화에 스스로 적응하도록 하는 등의 응용도 가능할 것이다. 또한 명료하고 쉬운 설계 기법은 많은 일반인들이 항공기 설계를 쉽게 할 수 있도록 항공기 설계의 진입 장벽을 낮춰줄 수 있으며, 이는 대중들의 항공공학에 대한 관심을 넓히고 민간 소형 드론과 항공기의 시장을 넓힐 수 있을 것이다.

2. 연구 목적

본 연구의 목적은 설계를 직관적이고 명료하게 하며, 설계 속도를 가속시키는데 있다. 본 연구에서는 설계 가속화를 위해 강력한 기계학습 기법인 딥러닝을 이용하였으며, 직관성과 명료성을 위해 설계에 실제 사용되는 힘과 모멘트 기반의 성능 파라미터를 사용하여 형상을 도출할 수 있도록 하였다.



<그림 1 -2> 딥러닝을 이용한 역형 설계

본 연구의 목표는 실시간 설계를 가능하도록 하여 설계에 소요되는 시간과 비용을 줄일 뿐만 아니라, 환경 적응형 설계 등 더 높은 수준의 공학 설계가 가능하도록 함에 있다. 뿐만 아니라 명료하고 직관적이게 하여 항공공학에 익숙하지 않은 일반인들도 기초만 알면 쉽게 에어포일을 만들어 낼 수 있도록 하였다.

3. 연구 방법

본 연구에서 구축한 에어포일 설계 알고리즘은 성능과 형상의 관계를 강력한 기계학습 기법인 인공신경망을 사용하여 구축하였다. 기계학습 기법은 명료한 공식과 알고리즘이 없어도 입력값과 결과값을 학습하여 두 데

이터 사이의 관계를 알아낸다. 최근 기계학습 기법을 사용한 연구의 주류를 이루는 사진 분류나 글씨 판독과 같은 알고리즘도 사진과 사진의 정보를 바탕으로 스스로 분류하는 법을 찾아내는 것이다.

근래에 4차 산업혁명과 함께 자주 언급되는 딥 러닝(Deep Learning)은 기존의 인공신경망들의 학습의 한계를 극복한 알고리즘으로서, 그 구조는 기존 인공신경망에 바탕을 두고 있으나, RBM(Restrict Boltzmann Machine) [3]과 같은 효율적인 초기화 기법, 활성화 함수의 개선, 최적화 기법의 발전으로 학습의 능률을 높이고 대규모 신경망을 학습할 수 있게 되었다. 본 연구에서 사용한 딥러닝 기법은 심층신경망(Deep Neural Network, DNN)으로, 더 깊고 넓은 인공신경망을 통해 예측하기 어려운 에어포일 성능-형상 사이의 관계를 추론하도록 하였다.

본 논문에서는 딥러닝을 이용하여 두 가지 형상 함수를 바탕으로 에어포일 데이터베이스를 구축하고 인공신경망을 학습시켰다. 심층 신경망이 성능 파라미터를 바탕으로 에어포일 형상을 예측할 수 있는지 확인할 필요가 있어, NACA 4-digit 에어포일 [4]을 이용한 간단한 설계 문제를 풀도록 하였다. NACA4-digit 형상은 NASA의 전신인 미국 국가항공자문위원회(National Advisory Committee for Aeronautics, NACA)에 의해 개발된 형상으로, 간단한 세 개의 파라미터를 이용하여 형상을 만들 수 있기에 초기 연구에 매우 적합한 형상이었다. 테스트 연구는 이 NACA 4-digit 형상의 세 파라미터를 간단한 다섯 가지의 성능 파라미터를 이용하여 실시하였다.

기계학습의 학습 데이터 구축에는 에어포일 성능 해석도구 XFOIL [5]을 사용하였다. XFOIL은 MIT의 Mark Drela 교수가 개발한 저정밀 에어포일 해석/설계 도구로서, 낮은 유속의 에어포일 해석에서 매우 탁월한 성능을 발휘한다. 또한, 저정밀 해석도구임에도 불구하고 경계층에 대한 해석이 가능하여 표면 마찰 항력까지 포함한 전체 항력계수를 도출할 수 있다. XFOIL은 에어포일 주위의 압력 분포, 반음각 범위에 따른 양력계수, 항력계수, 모멘트 계수, 천이점을 얻어낼 수 있다. 본 연구에서는 구체적인 성

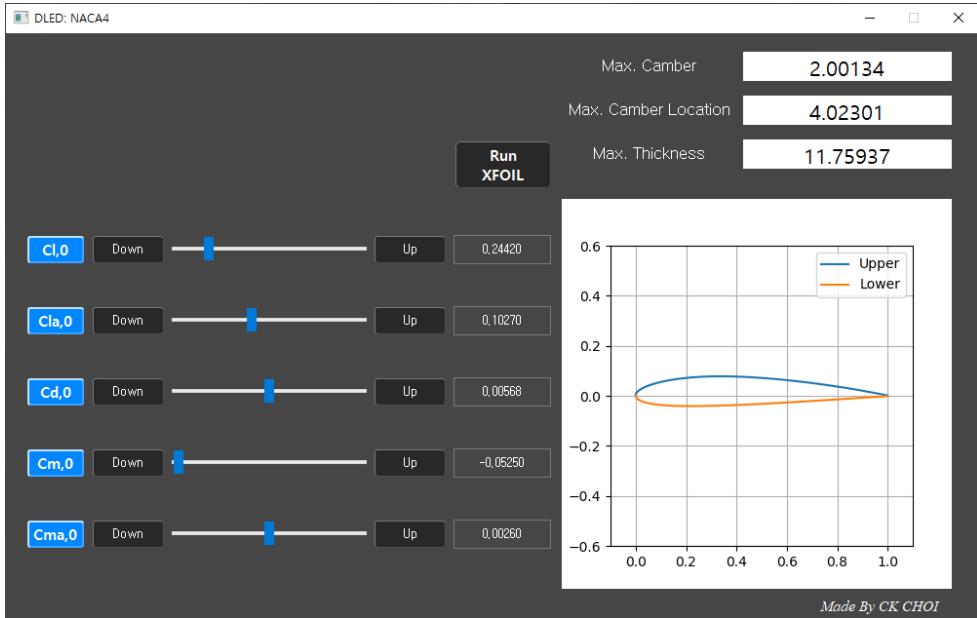
능 파라미터를 입력값으로 사용하기 위해 양력, 항력, 모멘트 계수와 선형 구간에서의 양력과 모멘트 계수 기울기를 사용하였으며, XFOIL 해석 시간을 줄이기 위해 받음각 0도에서의 값들을 사용하였다. 이를 다섯가지 성능 파라미터들을 이용하면 선형 구간의 에어포일 성능을 구할 수 있다.

인공신경망을 학습시킬 때, 충분한 양의 학습 데이터를 확보하지 못한다면 학습이 잘 되지 않는다. 그러나 인공신경망을 잘 학습시키기 위해서 데이터를 얼마나 모아야 하는지, 어떤 형상 영역에 대해서 데이터베이스를 강화해야 하는지 예측하기 어렵다. 본 연구에서는 인공신경망의 학습 결과를 이용해 알고리즘이 스스로 학습 데이터를 축적하도록 하여 충분한 양의 데이터를 확보할 수 있도록 하였다.

인공신경망이 잘 학습할 수 있도록 충분한 양의 학습 데이터가 누적되면, 모델 최적화를 해야 한다. 모델의 최적화는 하이퍼 파라미터를 조절해 가면서 실시한다. 모델의 학습은 내부의 가중치와 바이어스 등을 조절하는 과정으로 자동적으로 이루어지지만, 학습 모델을 구성하는 다양한 요소들, 하이퍼 파라미터들의 최적화는 전적으로 모델을 구축하는 사람에게 달려 있다. 최적화가 잘 된 모델은 효율적인 학습이 가능하며, 보다 나은 성능을 보여준다.

하이퍼 파라미터 최적화를 마치면 최적화된 하이퍼 파라미터를 따라 모델을 학습시키며, 이렇게 생성된 최종 모델이 실제 설계 엔지니어들이 사용하게 될 인공지능 모델이 된다. 학습이 완료된 인공신경망 모델은 2ms(0.002 second) 내외로 형상을 예측할 수 있다.

본 연구에서는 사용자가 쉽게 사용할 수 있도록 GUI를 구현하고 실행파일로 만들었다. GUI는 <그림 1-3>과 같이 구성되어 있으며, 각 성능 파라미터는 슬라이더와 버튼, 그리고 입력창을 통해 쉽게 바꿀 수 있도록 하였다. 또한 현재 생성된 형상을 XFOIL을 통해 해석하고 오차를 볼 수 있는 기능도 추가하였다.

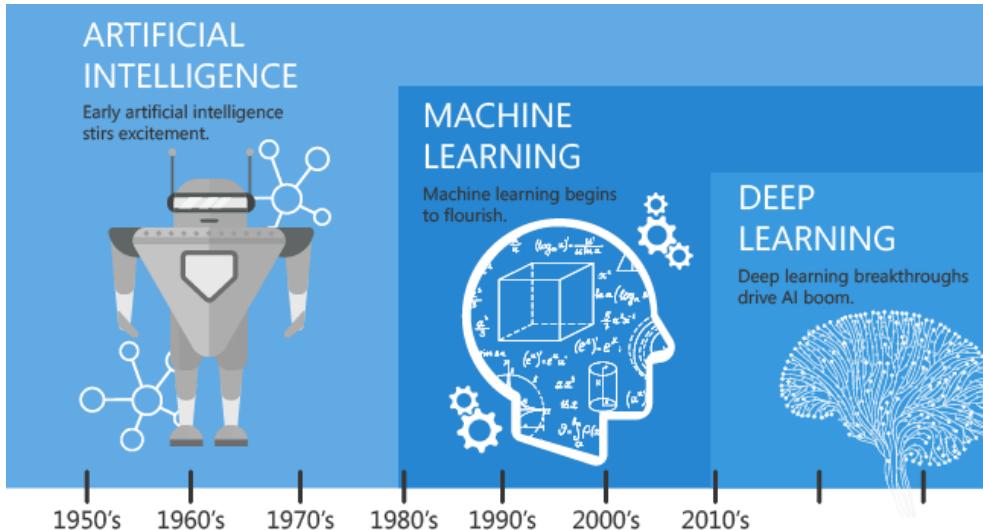


<그림 1-3> DLED: NACA4 실행 화면

제 2 장 이론적 배경

제 1절 심층신경망(Deep Neural Network)

1. 기계학습(Machine Learning)

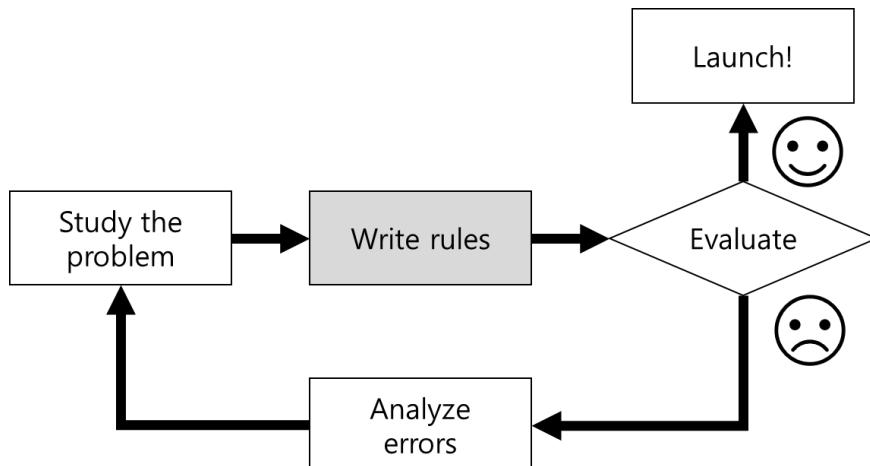


<그림 2-1> 인공지능, 기계학습, 그리고 딥러닝 [6]

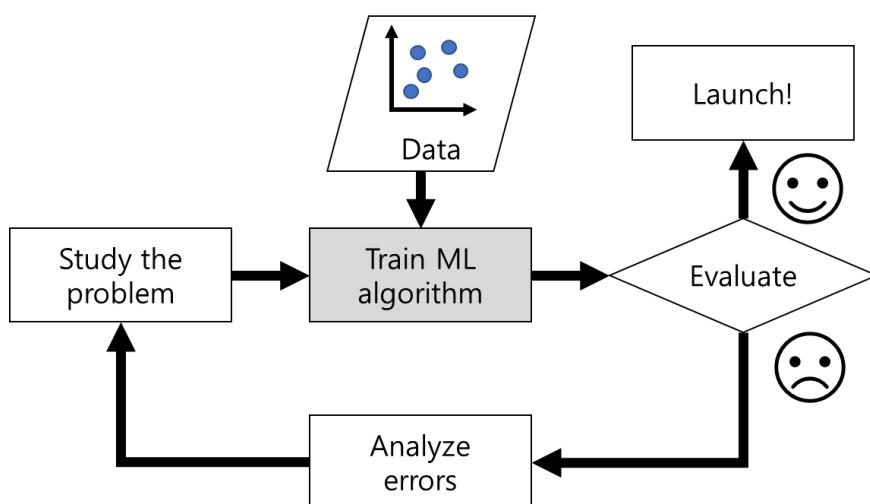
기계학습은 인공지능(Artificial Intelligence, AI)의 한 분야로서, 컴퓨터가 어떤 문제를 해결할 수 있도록 학습시키는 알고리즘과 그 기술을 말한다. 자연의 다양한 현상들은 실험과 고찰을 통해 수식으로서 정립이 되었지만, 보다 추상적인 문제에 있어서는 그 관계를 찾아내기 어렵다. 물체를 어떻게 구별하는가, 다음에 써야 할 적합한 단어는 무엇인가와 같은 것은 자연 현상과는 다른 종류의 문제였다. 그렇기에 많은 수학자들과 엔지니어들은 사람이 직접 관계를 명료하게 기술하는 것이 아닌, 데이터들을 이용하여 알고리즘이 스스로 관계를 찾아내도록 하는 기계학습 기법을 통해 추상적인 문제들을 풀어내고자 하였다.

<그림 2-2>는 고전적인 프로그래밍 방법을 나타낸 것이다. 전형적인 방법으로 어떤 문제를 해결하기 위해서는 사용자가 모든 규칙들을 지정해주어야 한다. If-then 알고리즘이나 규칙 기반 알고리즘같은 방법들은 한 두

개의 규칙을 이용하여 구분하는 것에는 매우 효과적이나, 거대하고 복잡하며 추상적인 문제를 해결하려 하다 보면 규칙과 분기점이 기하급수적으로 늘어나게 된다. 이러한 추상적인 문제를 해결하기 위해서는 개발자가 모든 규칙을 하나하나 작성해야 하며, 새로운 규칙이 등장하면 그 때마다 다시 코드를 수정해야 한다. 결국 규칙은 점차 복잡해지고 규칙간 상호작용을 쉽게 예측할 수 없어 프로그램을 유지하고 수정하기 어려워진다.



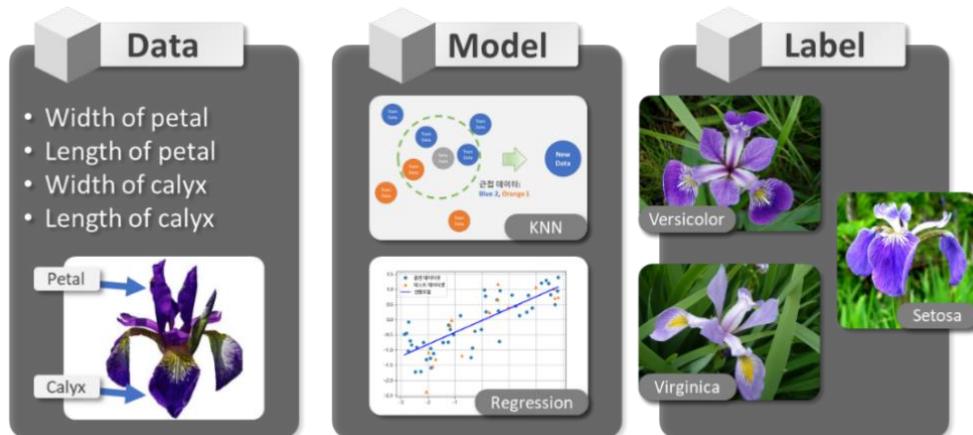
<그림 2-2> 전통적인 프로그래밍 방법 [7]



<그림 2-3> 기계학습 기법을 이용한 방법 [7]

이와는 다르게 기계학습 기법은 데이터들을 바탕으로 알고리즘이 스스로 문제를 해결하는 방법을 학습을 하므로, 복잡한 문제에서 더 단순한 알고리즘으로도 문제를 해결할 수 있다. 개발자는 단순히 알고리즘이 학습 데이터를 받아들여 스스로 규칙을 찾아가도록 하면 되며, 이는 상대적으로 프로그램을 유지하고 수정하기 쉽게 만들어준다. 만약 문제가 달라지거나 복잡해진다 할 지라도 새로운 데이터를 추가하여 학습시키면 된다.

최근에는 이러한 기계학습만의 강점을 바탕으로 컴퓨터공학 영역을 넘어 품종 분류, 물체 인식, 단어 예측과 글자 인식, 그리고 자율주행, 이미지 변환 및 생성, 의료 진단과 같은 다양한 영역에서도 사용되고 있다.



<그림 2-4> 붓꽃 분류 예시

<그림 2-3>의 붓꽃 품종 분류는 대표적인 기계학습 연습문제이다. 이는 붓꽃의 품종(Setosa, Virginica, Versicolor)을 꽃잎과 꽂받침의 너비와 길이로 구분하는 문제다. 꽃을 좋아하거나 연구하는 사람이라면 붓꽃 잎의 생김새를 바탕으로 어떤 품종인지를 구분할 수 있으며, 그들의 지식을 바탕으로 구분하는 알고리즘을 만들어낼 수 있을 것이다. 그러나 지식을 알고리즘에 구현한다는 것은 매우 어려운 과정이며, 구분해야 할 가지수가 수십 가지로 늘어난다면 변수를 정하고 지식을 알고리즘으로 명료하게 구현하는 것은 불가능한 일이 될 수 있다.

반면 기계학습은 사람이 지식을 찾아내고 알고리즘으로 구현하지 않아도 스스로 지식을 찾아내므로, 변수와 품종이 늘어나도 꽃들의 데이터들이 잘 구축되어 있다면 기계는 품종들을 구분하기 위한 최적의 모델을 알아서 찾아낼 수 있다. 필요한 것은 잘 구축된 데이터와 훈련모델 뿐이다.

A. 기계학습의 분류 [7]

기계학습을 나누는 방법은 매우 다양하나, 크게는 다음 세 가지 기준에 따라 분류할 수 있다.

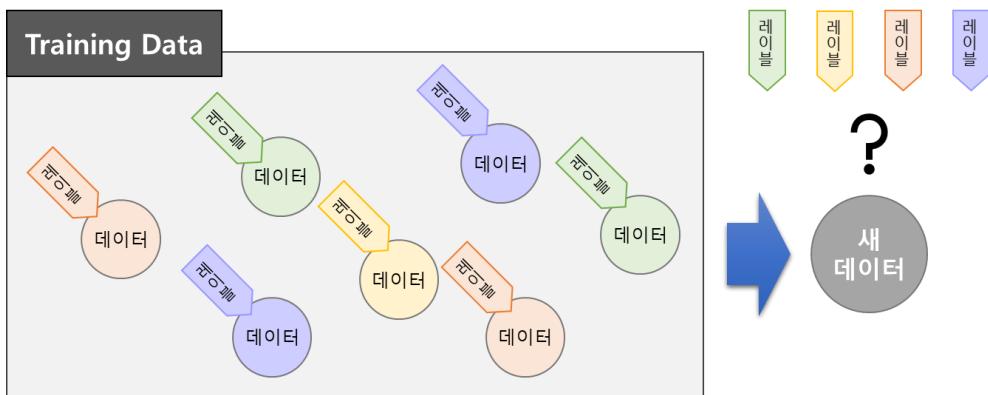
- 학습에 사용되는 데이터가 정답이 있는가?
 - 지도학습, 비지도 학습, 준지도학습, 강화학습
- 데이터 학습 방법
 - 온라인 학습, 배치(Batch) 학습
- 데이터 예측 방법
 - Instance 학습, 모델 기반 학습

B. 지도학습과 비지도학습

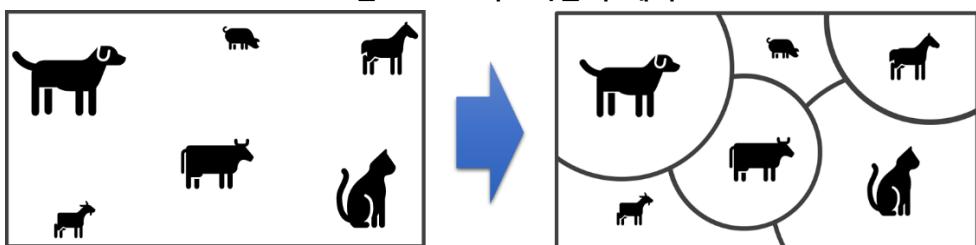
기계학습은 학습 데이터가 분류되어 있는지에 따라 지도학습과 비지도학습으로 나눌 수 있다. 지도학습(<그림 2-5>)은 입력 데이터에 정답(레이블)이 있는 것으로서, 입력과 출력 데이터를 바탕으로 학습모델을 만든다. 입력에 대한 정답 데이터가 있으므로, 이는 보통 주어진 입력을 바탕으로 출력을 예측하고자 할 때 사용된다. 비지도 학습(<그림 2-6>)은 어떤 데이터에 대해 출력값이나 정보 없이 학습을 시키는 것이다. 비지도 학습 알고리즘은 일반적으로 분류작업을 하는 클러스터링이나 데이터 분석을 보다 용이하게 하기 위하여 데이터들의 특징을 뽑아내는 비지도 변환에 쓰인다. 다음은 대표적인 지도/비지도 학습 기법들이다.

- 지도학습 기법
 - K-최근접 이웃 (K-Nearest Neighbor, K-NN)
 - 선형 회귀 (Linear Regression)

- 나이브 베이즈 분류기 (naïve Bayes)
- 결정트리 (Decision Tree)
- 서포트 벡터 머신 (Support Vector Machine, SVM)
- 인공신경망/딥러닝 (Artificial Neural Network, ANN/Deep Learning)
- 비지도학습 기법
 - 주성분 분석 (Principal Component Analysis, PCA)
 - 비음수 행렬 분해 (non-Negative Matrix Factorization, NMF)
 - 매니폴드 학습 (Manifold Learning)
 - K-평균 군집 (K-means)
 - 병합 군집 (Agglomerative Clustering)
 - DBSCAN (Density-Based Spatial Clustering of Applications with Noise)



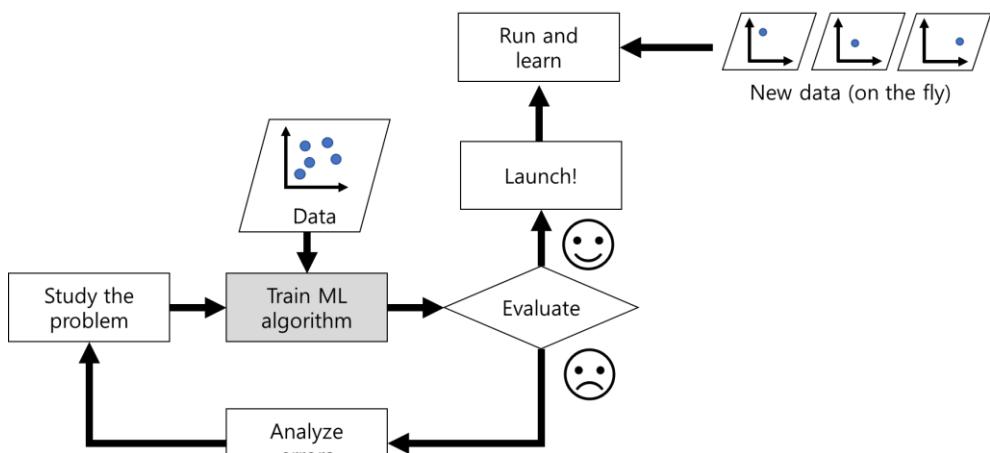
<그림 2-5> 지도학습의 예시



<그림 2-6> 비지도 학습의 예시

C. 온라인 학습과 배치학습

기계학습은 학습 데이터를 어떻게 학습하는지에 따라 온라인 학습과 배치학습으로 구분할 수 있다. 온라인 학습은 주가지수와 같이 데이터가 지속적으로 들어올 때마다 새로운 데이터를 학습하도록 하는 방법이다. 배치학습은 오프라인 학습이라고도 하는데, 학습 데이터가 충분히 모인 상태에서 전체 데이터를 이용해 학습하는 방법이다. 배치학습은 대량의 데이터를 한번에 학습하므로 학습에 소요되는 시스템의 메모리가 데이터 양에 따라 증가한다. 너무 많은 데이터를 배치학습 시킬 경우, 메모리 부족으로 인해 연산을 하지 못하는 경우도 생긴다. 그러나 배치학습은 전체 데이터에 대해 학습하기 때문에 전 데이터 영역에 대한 학습이 잘 된다. 반면 온라인 학습은 지속적인 데이터를 받아들여 새로운 상황에 적응할 수 있으나, 이로 인해 알고리즘이 최신 데이터에 편향되어 과거의 학습 데이터를 잊어버리기도 한다.



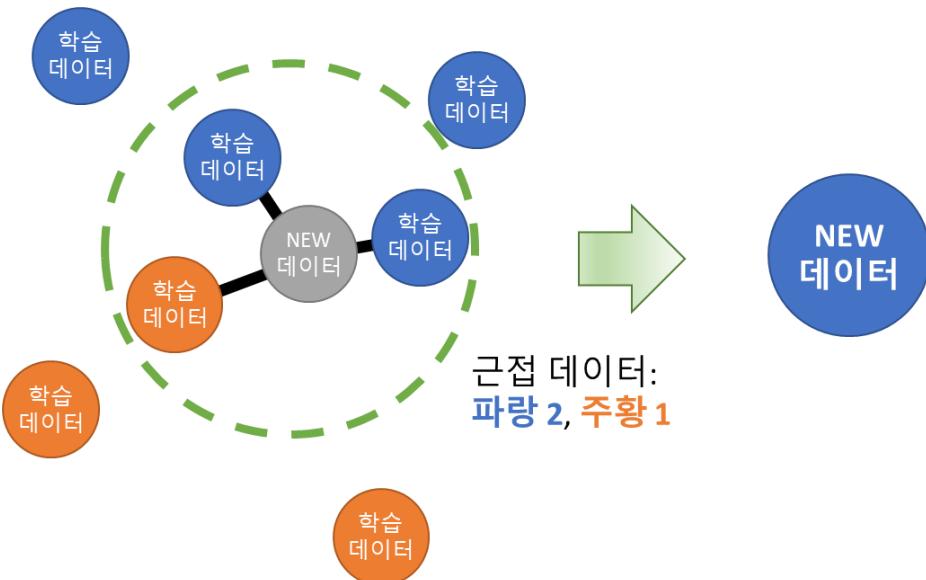
<그림 2-7> 온라인 학습(Online Training)

최근에는 효율적인 학습을 위해 배치학습과 온라인학습을 응용한 미니배치 학습을 사용하기도 한다. 미니배치 학습은 전체 학습데이터를 여러 개의 작은 데이터로 분할하여 학습하는 방법이다. 그러나 학습은 모든 데이터를 학습할 때까지 계속 한다. 10,000개의 데이터를 10개의 미니배치로

나누면 한 번에 1,000개씩 10번을 학습하는 셈이다. 일반적으로 각 미니배치에 대한 학습을 iteration이라 부르며, 전체 데이터에 대해 학습이 완료되면 1 epoch를 지난 것이다. 10,000개의 데이터를 10개의 미니배치로 나누어 20 epoch 학습시키면, 총 200회의 iteration이 일어나는 것이다. 미니배치를 사용할 때는 데이터의 분포를 고르게 하는 것이 좋다. 이는 온라인 학습과 마찬가지로 미니배치 학습 역시 이전의 학습 내용을 잊어버릴 수 있기 때문이다.

D. Instance 기반 학습과 모델 학습

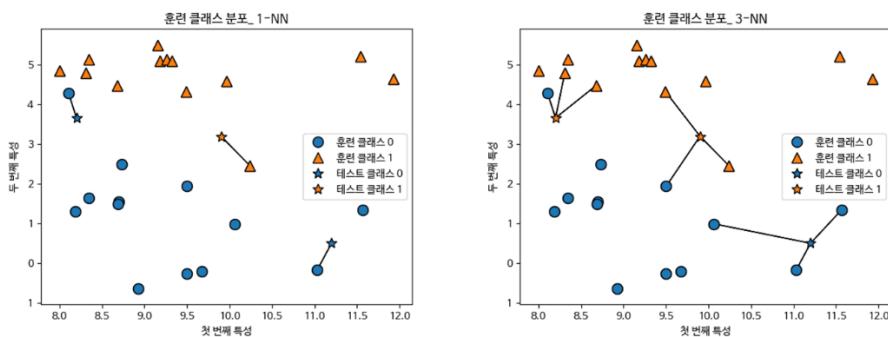
기계학습은 데이터를 바탕으로 모델을 만드는지, 아니면 주변 데이터를 바탕으로 판단하는지에 따라 모델학습과 Instance 기반 학습으로 나뉜다. Instance 기반 학습방법은 모르는 데이터가 들어왔을 때, 학습한 데이터들 중 어떤 부류에 더 유사한지를 계산하고, 어떤 부류인지를 도출한다. Instance 기반 학습을 적용한 대표적인 알고리즘은 K-NN(K-Nearest Neighborhood)로, 이 알고리즘은 임의의 데이터가 들어오면 가장 근접한 K 개의 데이터를 찾아내고, 가장 많은 데이터로 속성을 판별한다.



<그림 2-8> K-NN 기법을 이용한 데이터 판단

<그림 2-8>은 3개의 최근접 이웃(Nearest Neighbor)의 예시로서, 가장 가까운 세 개의 데이터를 추출하여 회색의 알 수 없는 데이터가 주황색 데이터인지 파란색 데이터인지 판별하는 방법을 설명한 것이다. <그림 2-8>에서 새 데이터에 가까운 3개의 데이터들 중, 2개의 데이터가 파란색이므로 알고리즘은 새로운 데이터를 파란색이라고 판단하게 된다.

Instance 기반 학습은 구현이 간단하지만 데이터가 많아질수록 연산량이 점진적으로 커지게 된다. 또한 판단에 사용하는 데이터 수에 따라 결과가 달라지는 문제도 있다. 그러므로 Instance 기반 학습은 예측 연산량을 줄이는 것과 판단에 사용할 최적의 데이터 수를 잘 정해야 한다.

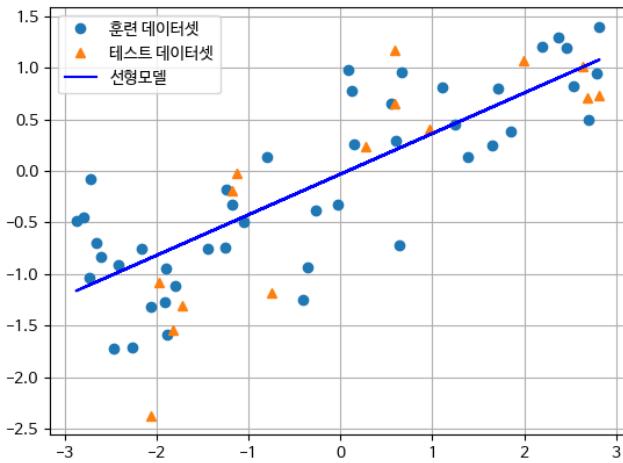


<그림 2-9> 이웃의 수에 따라 달라지는 레이블

반면, 모델기반 학습은 데이터들을 바탕으로 수학적인 모델을 생성하여

트렌드를 보거나 임의의 지점에 대한 예측을 하기 위해 사용된다. 모델이 완성되면 이전의 학습 데이터는 필요가 없으며, 입출력 데이터 사이의 함수가 만들어졌으므로 매우 빠른 예측이 가능하다. 대표적인 사례로는 부모의 키와 자식의 키의 상관관계, 소득과 행복도의 관계 등을 들 수 있다.

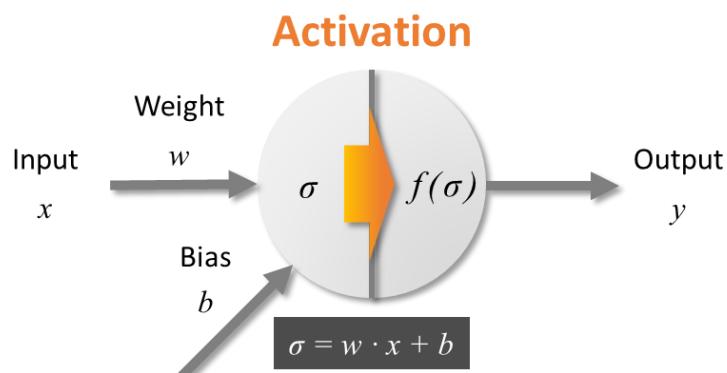
<그림 2-10>에서는 선형 회귀기법을 사용하여 산개되어 있는 데이터들을 대표할 수 있는 모델을 만들었다. 전체 데이터는 훈련 데이터와 테스트 데이터로 나뉘어져 있는데, 모델 생성은 훈련 데이터만 이용하여 실시한다. 모델을 만들 때는 일반적으로 훈련 데이터셋과 오차를 이용해, 전 훈련 데이터셋에 걸쳐 오차를 최소화할 수 있도록 한다.



<그림 2-10> 모델기반 학습 예시

2. 인공신경망(Artificial Neural Network)

딥러닝은 인공신경망에 기반한 기계학습 기법이다. 인공신경망은 인간의 뇌세포 및 신경망을 알고리즘으로 구현한 것으로, 일반적으로 수 많은 인공 뉴런들이 연결되어 있는 형태로 표현된다. 생물학적 뉴런은 전기적인 신호를 입력 받았을 때, 그 신호가 역치 넘으면 다음 뉴런에게 그 신호를 전달한다. 인공 뉴런도 이와 비슷하게 입력값을 받으면, 활성화 함수를 통과시켜 보다 복잡한 형태의 신호를 다음 인공 뉴런으로 전달한다.



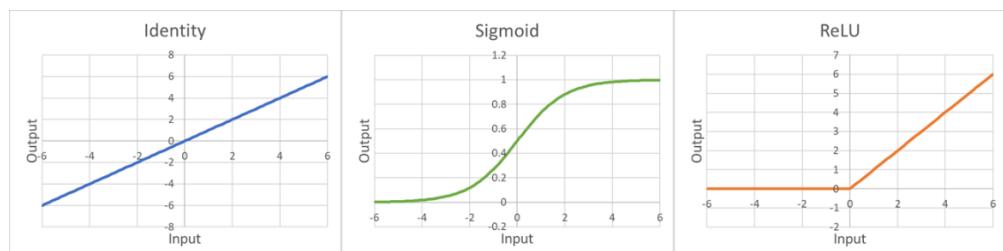
<그림 2-11> 인공 뉴런

<그림 2-11>은 인공 뉴런을 묘사한 그림으로, 인공 뉴런은 입력값 x 를 받으면 가중치(w)와 바이어스(b)로 선형 회귀식(σ)을 만들어낸다. 그리고 선형 회귀식을 보다 복잡하게 만들기 위해 활성화 함수(Activation function, $f(\sigma)$)에 통과시킨 후, 다음 인공 뉴런에게 전달한다.

인공 뉴런의 활성화함수로는 다양한 종류의 함수를 사용할 수 있다. 기초적인 활성화 함수는 Identity, Sigmoid, ReLU가 있다. Identity 함수를 사용하면 인공 뉴런은 비선형성을 갖지 않는 단순한 모델이 된다. Sigmoid는 보다 복잡한 모델을 만들 수 있지만, 신경망의 규모가 커질수록 학습을 못하는 현상이 발생한다는 단점이 있다. ReLU는 Sigmoid에서 발생하는 문제를 해결하면서 인공 뉴런에 비선형성을 주기 위해 사용되는 함수로서, 최근의 인공신경망은 기본적으로 ReLU를 사용하며, 상황에 따라 이를 변형한 함수를 사용하거나 특수한 함수를 쓰기도 한다. <표 2-1>과 <그림 2-12>는 각 활성화 함수와 그 그래프를 나타낸 것이다.

<표 2-1> 기본 활성화 함수

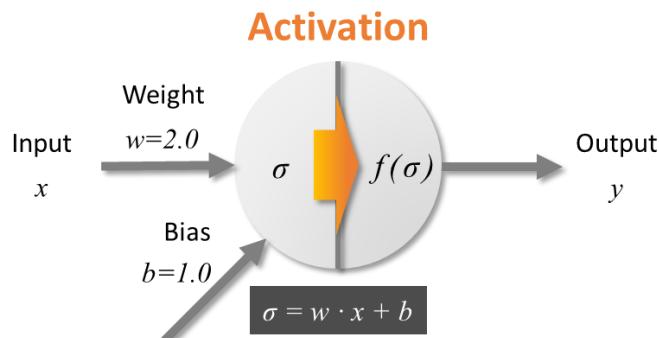
Basic Activation Function	
Identity	$f(\sigma) = \sigma$
Sigmoid	$f(\sigma) = \frac{1}{1 + e^{-\sigma}}$
Rectified Linear Unit (ReLU)	$f(\sigma) = \max(\sigma, 0)$



<그림 2-12> 기본적인 활성화 함수 그래프

A. 인공 뉴런의 학습

인공 뉴런의 학습은 앞전파와 역전파를 통해 이루어진다. 앞전파는 학습 데이터를 인공신경망에 그대로 넣는 것을 말한다. 아래는 인공뉴런을 설명하기 위한 예시로, 활성화 함수는 Identity 함수를 사용하고 인공 뉴런의 가중치와 바이어스가 아래의 그림과 같이 초기화하였다.



<그림 2-13> 인공 뉴런 초기화 예시

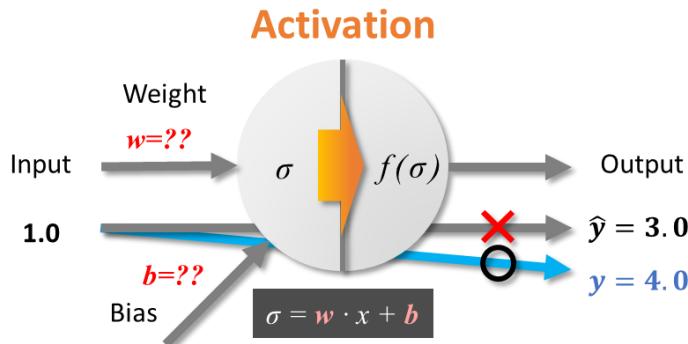
<그림 2-13>에서 인공신경망의 가중치는 2, 바이어스는 1로 초기화 되었다. 이 인공 뉴런에 입력 값을 0부터 3까지 넣으면 그 결과는 <표 2-2>와 같다.

<표 2-2> 인공 뉴런의 입력 값에 따른 예측 값

Input(x)	Prediction(\hat{y})
0	$\hat{y} = f(2 \cdot 0 + 1) = 1$
1	$\hat{y} = f(2 \cdot 1 + 1) = 3$
2	$\hat{y} = f(2 \cdot 2 + 1) = 5$
3	$\hat{y} = f(2 \cdot 3 + 1) = 7$

그러나 앞서 인공신경망이 예측한 값은 우리가 원하는 값이 아닐 수도 있다. 예를 들어, 신경망이 $x=1$ 에서 예측해야 하는 정답이 4이면, 오차 1이 발생한다. 인공 뉴런이 정확한 값을 예측하도록 하기 위해서는 모델을 수정해야 한다. 초창기 인공신경망은 모델을 구성하는 가중치와 바이어스

를 사람이 직접 수정하며 모델을 튜닝하였다. 그러나 1974년, 폴 웨어보스 (Paul Werbos)가 PhD 논문 [8]에서 자동으로 인공 뉴런을 업데이트 하는 법을 발표하고, 1986년 Rumelhart, Hinton, J. Williams^{o)} 후속 연구 [9]를 발표하면서, 인공신경망의 파라미터 업데이트를 자동화할 수 있게 되었다.



<그림 2-14> 인공 뉴런의 예측 값(\hat{y})과 실제 값(y)

역전파 알고리즘은 인공 뉴런의 예측 값과 정답 값을 바탕으로 오차 함수를 규정하고, 이를 최소화하는 최적화 과정이다. 간단한 모델에서는 다음과 같은 함수가 오차함수로 사용된다.

$$E = \frac{1}{2}(y - \hat{y})^2 \quad (1)$$

이 오차 함수를 바탕으로 가중치를 업데이트하기 위해 다음과 같은 연쇄 법칙(Chain rule)^{o)}이 사용된다.

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial w} = (y - \hat{y}) \cdot 1 \cdot x = 1 \quad (2)$$

그리고 오차에 따른 가중치의 업데이트는 다음과 같다.

$$w_{updated} = w - \alpha \cdot \frac{\partial E}{\partial w} = 2 - 0.1 \cdot (-1) = 2.1 \quad (3)$$

여기서 α 는 학습률(Learning rate)로서 학습의 속도를 결정짓는다. 수치해석 분야에서 사용되는 Step size와 같은 것이다. 위 예시에서 학습률은 0.1로 하였으며, 이때 가중치는 2.0에서 2.1로 업데이트 된다. 바이어스의 업데이트 과정도 이와 동일하며, 다음과 같다.

$$E = \frac{1}{2}(y - \hat{y})^2 \quad (4)$$

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial b} = (y - \hat{y}) \cdot 1 \cdot 1 = -1 \quad (5)$$

$$b_{updated} = b - \alpha \cdot \frac{\partial E}{\partial b} = 1 - 0.1 \cdot (-1) = 1.1 \quad (6)$$

<표 2-3>은 1회 학습 전후의 예측값 변화를 나타낸 것이다. 가중치는 2.0에서 2.1, 바이어스는 1.0에서 1.1로 업데이트 되었다. 이 때, 모델의 예측 값은 3.0에서 3.2로 변하였으며, 오차는 1에서 0.8로 줄어들었다.

<표 2-3> 학습 전후 예측값 변화

	Model	Predicted Value
Initial	$\hat{y} = f(2 \cdot x + 1) = 2x + 1$	3.0
Updated	$\hat{y} = 2.1x + 1.1$	3.2

예제를 통해 알 수 있듯이, 인공 뉴런의 학습은 앞전파와 역전파를 통해 자동적으로 이루어지게 할 수 있다. 그러나 한 번의 학습만으로는 모델의 오차가 충분히 줄어들지 못하고 여전히 0.8의 오차가 발생하는 것을 볼 수 있다. 그러므로 학습은 여러 번 반복하여 실시하게 된다.

<그림 2-15>는 앞전파와 역전파를 통한 학습을 100회 실시한 결과이며, 여기에 사용된 알고리즘은 아래의 1) 인공 뉴런 예제 코드에서 확인할 수 있다. <그림 2-15>를 보면 인공 뉴런을 100회 반복하여 학습시킨 결과, 예측값이 4에 매우 근접한 값을 나타내는 것을 볼 수 있다. 이 때, 인공

뉴런의 가중치(Neuron1.w_)는 2.4999..., 바이어스(Neuron1.b_)는 1.4999...로 업데이트 된 것을 확인할 수 있다.

```
Iter = 94, Output = 3.9999999992229327
Iter = 95, Output = 3.999999999378346
Iter = 96, Output = 3.999999999502677
Iter = 97, Output = 3.9999999996021414
Iter = 98, Output = 3.9999999996817133
Iter = 99, Output = 3.999999999745371
Iter = 100, Output = 3.9999999997962963

In [52]: Neuron1.w_
Out[52]: 2.4999999998981486

In [53]: Neuron1.b_
Out[53]: 1.4999999998981477

In [54]: |
```

<그림 2-15> 인공 뉴런 100회 학습 결과

1) 인공 뉴런 예제 코드

다음 페이지에 있는 코드는 <그림 2-15>에 사용된 Python 3 [10] 예제 코드로, [11]의 C++ 코드를 Python 3에서 동작하도록 한 것이다. 단일 뉴런은 하나의 클래스로 작성되었으며, 초기 가중치와 바이어스를 입력값으로 받는다. `getAct` 함수는 Identity 활성화 함수로서, 입력한 값을 그대로 받는다. `getActGrad`는 역전파를 위해 필요한 활성화 함수의 기울기로, 아래의 예시에서는 Identity 함수가 사용되었으므로 어떠한 값을 받아도 1을 전달한다. `ForwardProp`는 앞전파를 구현한 함수다. 입력값 `_input`을 받으면 가중치와 바이어스를 통해 선형 회귀함수를 만들어낸다. `BackProp`는 역전파를 구현한 함수로서, 목표값을 받으면 현재 뉴런의 예측값과의 오차를 계산하여 인공뉴런 모델이 더 정확한 예측을 할 수 있도록 가중치와 바이어스를 업데이트한다. 예시에서는 `Neuron1=Neuron(2.0,1.0)`을 통해 가중치와 바이어스를 각각 2.0과 1.0으로 초기화한 인공뉴런을 생성하였다. 그리고 `for i in range(0,100):`으로 총 100번의 학습을 하도록 하였으며, 예측 목표값은 for문 안의 `Neuron1.BackProp(4.0)`를 통해 설정하였다. 여기서 목표값은 4.0으로 하였다.

- 인공 뉴런 예제 코드(Python 3) [11]

```

class Neuron:
    def __init__(self, w_, b_):
        self.w_=w_
        self.b_=b_

    def getAct(self, x):
        # for linear or Identity activation function
        return x

    # Activation function 의 Gradient 를 구합니다.
    def getActGrad(self, x):
        # for linear activation function
        return 1

    # Forward Propagation 을 실시합니다.
    def ForwardProp(self, _input):
        self.input_ = _input

        sigma = self.w_* self.input_ + self.b_
        output = self.getAct(sigma)
        return output

    # Back propagation 을 실합니다.
    def BackProp(self, target):
        alpha = 0.1 # Learning Rate
        grad = (self.ForwardProp(self.input_) - target) *
        self.getActGrad(self.ForwardProp(self.input_))
        self.w_= self.w_- alpha * grad* self.input_
        self.b_= self.b_- alpha * grad*1.0

Neuron1=Neuron(2.0,1.0)

print("Input = 1.0, Output =
{}".format(Neuron1.ForwardProp(1.0)))

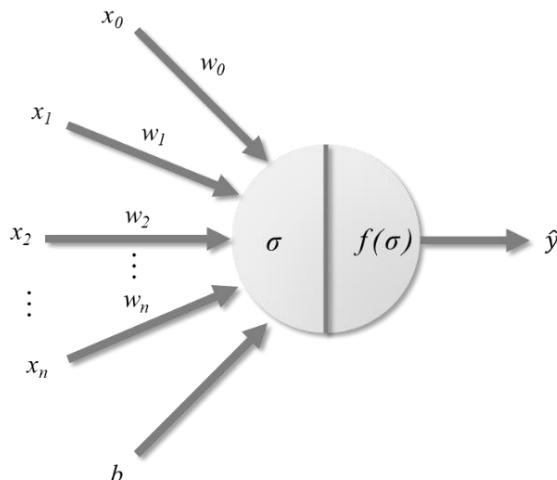
for i in range(0,100):
    Neuron1.BackProp(4.0)
    print("Iter = {}, Output = {}".format((i+1),
Neuron1.ForwardProp(1.0)))

```

B. 인공신경망의 학습

인공신경망은 인공 뉴런을 복수로 연결한 형태로, 인공신경망의 학습은 기본적으로 인공 뉴런의 학습 과정과 동일하다. 다만, 층과 너비가 커진 만큼 더 복잡하다. 인공 뉴런이 다중의 입력 값을 받을 경우, σ 는 <그림 2-16>처럼 표현된다.

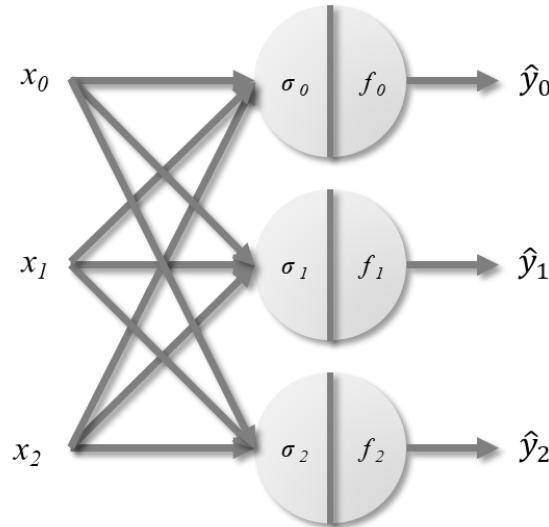
$$\sigma = w_0 \cdot x_0 + w_1 \cdot x_1 + \cdots + w_n \cdot x_n + b = [w_0 \ w_1 \ \cdots \ w_n \ b] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \\ 1 \end{bmatrix} \quad (7)$$



<그림 2-16> 다중 입력 단일 출력 인공 뉴런

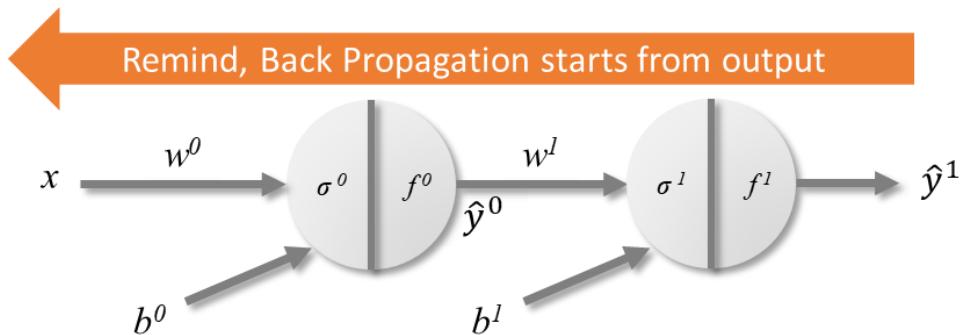
<그림 2-17>은 하나의 층에 여러 인공 뉴런을 배치한 인공신경망으로서, 이 경우 선형 함수와 가중치, 바이어스는 행렬 형태로 표현할 수 있다. 마찬가지로 앞서 설명한 인공 뉴런의 학습 방법을 이용하여 모든 가중치와 바이어스에 대해 업데이트를 실시하여 학습시킨다.

$$\begin{bmatrix} \sigma_0 \\ \sigma_1 \\ \sigma_2 \end{bmatrix} = \begin{bmatrix} w_{00} & w_{01} & w_{02} & b_1 \\ w_{10} & w_{11} & w_{12} & b_2 \\ w_{20} & w_{21} & w_{22} & b_3 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ 1 \end{bmatrix} \quad (8)$$



<그림 2-17> 단층 인공신경망

인공 뉴런은 병렬로도 배치할 수 있지만 직렬로도 여러 층을 이루게 할 수 있다. <그림 2-18>은 인공 뉴런 두 개를 이용하여 2층의 인공신경망을 구현한 것이다.



<그림 2-18> 두 개의 층을 갖는 인공신경망

앞서 인공 뉴런과 마찬가지로 단순한 예시를 위해 활성화 함수를 Identity 함수를 사용할 경우, 인공신경망은 다음과 같은 수식으로 나타낼 수 있다.

$$\hat{y} = f^1(\sigma^1) = f^1(w^1 \hat{y}_0 + b^1) = f^1(w^1 f^0(w^0 x + b^0) + b^1) \quad (9)$$

역전파는 예측 값과 정답을 이용한 오차 함수로부터 시작되므로 마지막 층부터 실시하게 된다. 각 층의 업데이트는 다음과 같이 진행된다.

- Error function

$$E = \frac{1}{2}(y - \hat{y}_1)^2 \quad (10)$$

- Last layer

$$\frac{\partial E}{\partial w^1} = \frac{\partial E}{\partial \hat{y}^1} \cdot \frac{\partial \hat{y}^1}{\partial f^1} \cdot \frac{\partial f^1}{\partial \sigma^1} \cdot \frac{\partial \sigma^1}{\partial w^1} = (\hat{y}^1 - y) \cdot 1 \cdot 1 \cdot \hat{y}^0 \quad (11)$$

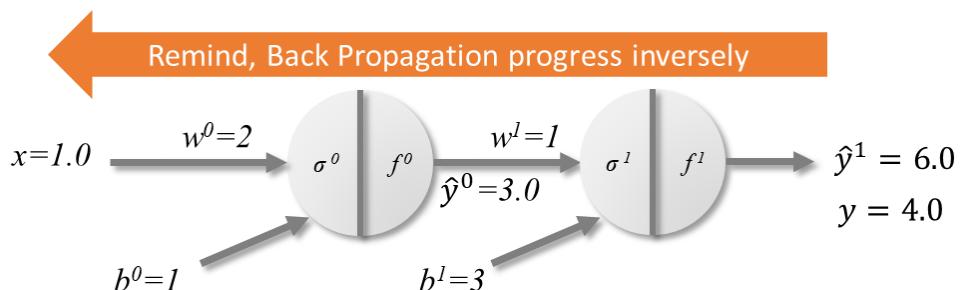
$$\frac{\partial E}{\partial b^1} = \frac{\partial E}{\partial \hat{y}^1} \cdot \frac{\partial \hat{y}^1}{\partial f^1} \cdot \frac{\partial f^1}{\partial \sigma^1} \cdot \frac{\partial \sigma^1}{\partial b^1} = (\hat{y}^1 - y) \cdot 1 \cdot 1 \cdot 1 \quad (12)$$

- First layer

$$\frac{\partial E}{\partial w^0} = \frac{\partial E}{\partial \hat{y}^1} \cdot \frac{\partial \hat{y}^1}{\partial f^1} \cdot \frac{\partial f^1}{\partial \sigma^1} \cdot \frac{\partial \sigma^1}{\partial f^0} \cdot \frac{\partial f^0}{\partial \sigma^0} \cdot \frac{\partial \sigma^0}{\partial w^0} = (\hat{y}_1 - y) \cdot 1 \cdot 1 \cdot w^1 \cdot 1 \cdot x \quad (13)$$

$$\frac{\partial E}{\partial b^0} = \frac{\partial E}{\partial \hat{y}^1} \cdot \frac{\partial \hat{y}^1}{\partial f^1} \cdot \frac{\partial f^1}{\partial \sigma^1} \cdot \frac{\partial \sigma^1}{\partial f^0} \cdot \frac{\partial f^0}{\partial \sigma^0} \cdot \frac{\partial \sigma^0}{\partial w^0} = (\hat{y}_1 - y) \cdot 1 \cdot 1 \cdot w^1 \cdot 1 \cdot 1 \quad (14)$$

간단한 예시로 아래 <그림 2-19>와 같은 인공신경망을 만들고, 입력 값 1.0에 대하여 4.0을 예측하도록 한다면 업데이트는 다음과 같이 진행된다.



<그림 2-19> 두 개의 층을 갖는 인공신경망 예시

- Last layer

$$\frac{\partial E}{\partial w^1} = \frac{\partial E}{\partial \hat{y}^1} \cdot \frac{\partial \hat{y}^1}{\partial f^1} \cdot \frac{\partial f^1}{\partial \sigma^1} \cdot \frac{\partial \sigma^1}{\partial w^1} = (\hat{y}^1 - y) \cdot 1 \cdot 1 \cdot \hat{y}^0 \quad (15)$$

$$\frac{\partial E}{\partial b^1} = \frac{\partial E}{\partial \hat{y}^1} \cdot \frac{\partial \hat{y}^1}{\partial f^1} \cdot \frac{\partial f^1}{\partial \sigma^1} \cdot \frac{\partial \sigma^1}{\partial b^1} = (\hat{y}^1 - y) \cdot 1 \cdot 1 \cdot 1 \quad (16)$$

$$\frac{\partial E}{\partial w^1} = 2 \cdot 1 \cdot 1 \cdot 3 = 6 \quad (17)$$

$$\frac{\partial E}{\partial b^1} = 2 \cdot 1 \cdot 1 \cdot 1 = 2 \quad (18)$$

$$w_{updated}^1 = w^1 - \alpha \cdot \frac{\partial E}{\partial w^1} = 1 - 0.1 \cdot 6 = 0.4 \quad (19)$$

$$b_{updated}^1 = b^1 - \alpha \cdot \frac{\partial E}{\partial b^1} = 3 - 0.1 \cdot 2 = 2.8 \quad (20)$$

- First layer

$$\frac{\partial E}{\partial w^0} = \frac{\partial E}{\partial \hat{y}^1} \cdot \frac{\partial \hat{y}^1}{\partial f^1} \cdot \frac{\partial f^1}{\partial \sigma^1} \cdot \frac{\partial f^0}{\partial \sigma^0} \cdot \frac{\partial \sigma^0}{\partial w^0} = (\hat{y}^1 - y) \cdot 1 \cdot 1 \cdot w^1 \cdot 1 \cdot x \quad (21)$$

$$\frac{\partial E}{\partial b^0} = \frac{\partial E}{\partial \hat{y}^1} \cdot \frac{\partial \hat{y}^1}{\partial f^1} \cdot \frac{\partial f^1}{\partial \sigma^1} \cdot \frac{\partial f^0}{\partial \sigma^0} \cdot \frac{\partial \sigma^0}{\partial w^0} = (\hat{y}^1 - y) \cdot 1 \cdot 1 \cdot w^1 \cdot 1 \cdot 1 \quad (22)$$

$$\frac{\partial E}{\partial w^0} = 2 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1.0 = 2 \quad (23)$$

$$\frac{\partial E}{\partial b^1} = 2 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 = 2 \quad (24)$$

$$w_{updated}^1 = w^0 - \alpha \cdot \frac{\partial E}{\partial w^0} = 2 - 0.1 \cdot 2 = 1.8 \quad (25)$$

$$b_{updated}^1 = b^1 - \alpha \cdot \frac{\partial E}{\partial b^0} = 1 - 0.1 \cdot 2 = 0.8 \quad (26)$$

<표 2-4>와 <표 2-5>는 단일 인공 뉴런에서의 예시와 두 개의 층을 갖는 인공신경망의 1회 학습 결과를 비교한 것이다. 1회 학습 후, 인공신경망의 예측 값은 6.0에서 3.84로 감소하여 단일 인공 뉴런보다 정답(4.0)에 더 근접한 것을 볼 수 있다.

일반적으로 인공신경망의 층이 늘어나면 더 복잡한 문제를 쉽게 풀 수 있다. 고전적인 인공신경망인 퍼셉트론(Perceptron)의 경우, 1개만으로 Or, And 문제는 풀 수 있었으나, XOR 문제는 퍼셉트론이 여러 개의 층으로 연결되어야 풀 수 있던 것과 같은 맥락이다. [7] [12]

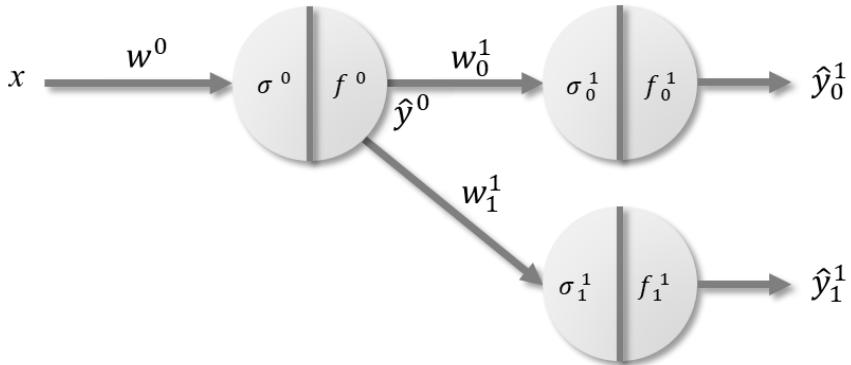
<표 2-4> 단일 인공 뉴런 예시

1 Layer	Model	Predicted Value	Error
Initial	$\hat{y} = f(2 \cdot x + 1) = 2x + 1$	3.0	1.0
Updated	$\hat{y} = 2.1x + 1.1$	3.2	0.8

<표 2-5> 두 개의 인공 뉴런을 이용한 2층 인공신경망 예시

2 Layer	Model	Predicted Value	Error
Initial	$\hat{y} = f^1(1 \cdot f^0(2 \cdot x + 1) + 3)$	6.0	2.0
Updated	$\hat{y} = f^1(0.4 \cdot f^0(1.8 \cdot x + 0.8) + 2.8)$	3.84	0.16

<그림 2-20>은 보다 복잡한 인공신경망을 구현한 것으로 3개의 인공뉴런을 이용하여 두 개의 은닉층을 갖는 다중 출력 인공신경망을 나타낸 것이며, 업데이트 방법은 다음과 같다.



<그림 2-20> 개의 인공 뉴런을 이용한 인공신경망

- Last layer, 1st neuron

$$E_0 = \frac{1}{2}(\hat{y}_0^1 - y_0)^2 \quad (27)$$

$$\frac{\partial E_0}{\partial w_0^1} = \frac{\partial E_0}{\partial \hat{y}_0^1} \cdot \frac{\partial \hat{y}_0^1}{\partial f_0^1} \cdot \frac{\partial f_0^1}{\partial \sigma_0^1} \cdot \frac{\partial \sigma_0^1}{\partial w_0^1} = (\hat{y}_0^1 - y_0) \cdot 1 \cdot 1 \cdot \hat{y}^0 \quad (28)$$

$$\frac{\partial E_0}{\partial b_0^1} = \frac{\partial E_0}{\partial \hat{y}_0^1} \cdot \frac{\partial \hat{y}_0^1}{\partial f_0^1} \cdot \frac{\partial f_0^1}{\partial \sigma_0^1} \cdot \frac{\partial \sigma_0^1}{\partial b_0^1} = (\hat{y}_0^1 - y_0) \cdot 1 \cdot 1 \cdot 1 \quad (29)$$

- Last layer, 2nd neuron

$$E_1 = \frac{1}{2}(\hat{y}_1^1 - y_1)^2 \quad (30)$$

$$\begin{aligned} \frac{\partial E_1}{\partial w_1^1} &= \frac{\partial E_1}{\partial \hat{y}_1^1} \cdot \frac{\partial \hat{y}_1^1}{\partial f_1^1} \cdot \frac{\partial f_1^1}{\partial \sigma_1^1} \cdot \frac{\partial \sigma_1^1}{\partial w_1^1} \\ &= (\hat{y}_1^1 - y_1) \cdot 1 \cdot 1 \cdot \hat{y}^0 \end{aligned} \quad (31)$$

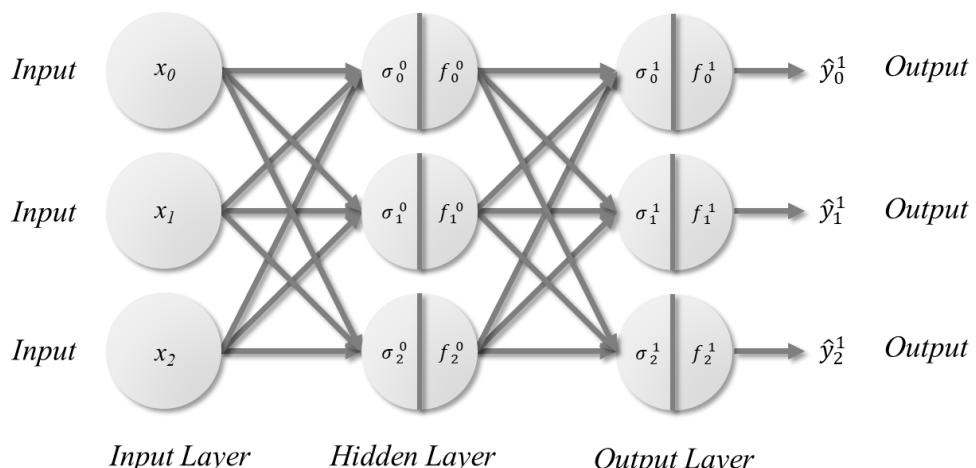
$$\begin{aligned} \frac{\partial E_1}{\partial b_1^1} &= \frac{\partial E_1}{\partial \hat{y}_1^1} \cdot \frac{\partial \hat{y}_1^1}{\partial f_1^1} \cdot \frac{\partial f_1^1}{\partial \sigma_1^1} \cdot \frac{\partial \sigma_1^1}{\partial b_1^1} \\ &= (\hat{y}_1^1 - y_1) \cdot 1 \cdot 1 \cdot 1 \end{aligned} \quad (32)$$

- First layer

$$\begin{aligned}
 \frac{\partial E}{\partial w^0} &= \frac{\partial E}{\partial \hat{y}_0^1} \cdot \frac{\partial \hat{y}_0^1}{\partial w^0} + \frac{\partial E}{\partial \hat{y}_1^1} \cdot \frac{\partial \hat{y}_1^1}{\partial w^0} \\
 &= \frac{\partial E_0}{\partial \hat{y}_0^1} \cdot \frac{\partial \hat{y}_0^1}{\partial f_0^1} \cdot \frac{\partial f_0^1}{\partial \sigma_0^1} \cdot \frac{\partial \sigma_0^1}{\partial f^0} \cdot \frac{\partial f^0}{\partial \sigma^0} + \frac{\partial E_1}{\partial \hat{y}_1^1} \cdot \frac{\partial \hat{y}_1^1}{\partial f_1^1} \cdot \frac{\partial f_1^1}{\partial \sigma_1^1} \cdot \frac{\partial \sigma_1^1}{\partial f^0} \cdot \frac{\partial f^0}{\partial \sigma^0} \quad (33) \\
 &= (\hat{y}_0^1 - y_0) \cdot 1 \cdot 1 \cdot w_0^1 \cdot 1 \cdot x + (\hat{y}_1^1 - y_1) \cdot 1 \cdot 1 \cdot w_1^1 \cdot 1 \cdot x
 \end{aligned}$$

$$\begin{aligned}
 \frac{\partial E}{\partial b^0} &= \frac{\partial E}{\partial \hat{y}_0^1} \cdot \frac{\partial \hat{y}_0^1}{\partial b^0} + \frac{\partial E}{\partial \hat{y}_1^1} \cdot \frac{\partial \hat{y}_1^1}{\partial b^0} \\
 &= \frac{\partial E_0}{\partial \hat{y}_0^1} \cdot \frac{\partial \hat{y}_0^1}{\partial f_0^1} \cdot \frac{\partial f_0^1}{\partial \sigma_0^1} \cdot \frac{\partial \sigma_0^1}{\partial f^0} \cdot \frac{\partial f^0}{\partial \sigma^0} \cdot \frac{\partial \sigma^0}{\partial b^0} + \frac{\partial E_1}{\partial \hat{y}_1^1} \cdot \frac{\partial \hat{y}_1^1}{\partial f_1^1} \cdot \frac{\partial f_1^1}{\partial \sigma_1^1} \cdot \frac{\partial \sigma_1^1}{\partial f^0} \cdot \frac{\partial f^0}{\partial \sigma^0} \cdot \frac{\partial \sigma^0}{\partial b^0} \quad (34) \\
 &= (\hat{y}_0^1 - y_0) \cdot 1 \cdot 1 \cdot w_0^1 \cdot 1 \cdot 1 + (\hat{y}_1^1 - y_1) \cdot 1 \cdot 1 \cdot w_1^1 \cdot 1 \cdot 1
 \end{aligned}$$

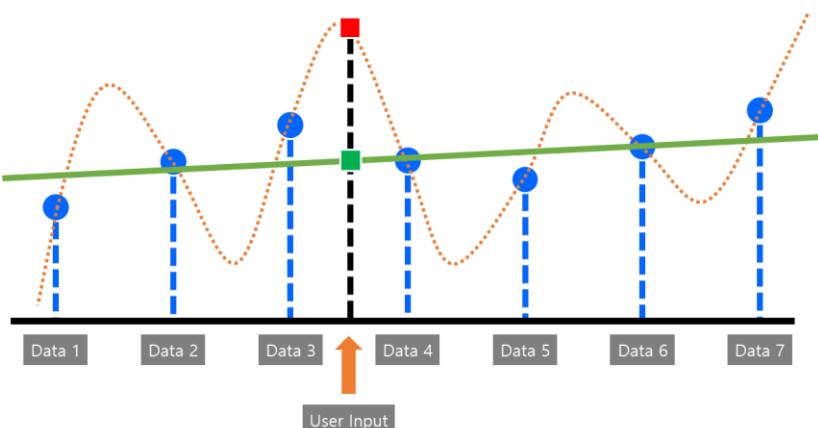
그리고 이를 더 확장하면 <그림 2-21>과 같이 복잡한 인공신경망을 구성할 수 있다. <그림 2-21>은 세 가지 층으로 이루어져 있다. 입력 값이 들어가는 층은 입력층(Input layer), 마지막 출력 값이 나오는 층을 출력층(Output layer), 그리고 그 사이에 있는 층을 은닉층(Hidden layer)라고 한다. 은닉층은 인공신경망의 구성에 따라 여러 개가 될 수 있으며, 아래 그림과 같이 전후 레이어의 모든 인공 뉴런이 연결된 인공신경망을 Fully Connected Neural Network(FCNN), 혹은 Feed forward neural network라고 한다.



<그림 2-21> Fully Connected Neural Network 예시

C. 과적합 문제(Overfitting Problem)

기계학습은 데이터를 바탕으로 알고리즘을 학습시켜, 데이터의 상관관계를 잘 표현하는 모델을 만드는 기법이다. 그러나 모델이 학습 데이터를 지나치게 따르다 보면, 모델을 이용해 프로그램을 배포하고 사용시 잘못된 예측을 하는 경우가 발생하기도 한다. 이러한 현상을 과적합이라 한다. <그림 2-22>는 과적합 문제를 가시화한 예시로, 7개의 데이터로 두 개의 모델을 만든 것이다. 점선 모델은 7개의 데이터에 대해 오차가 거의 나지 않는 반면, 실선의 모델은 일부 데이터들에 대해서 오차가 있다는 것을 확인할 수 있다. 각각의 두 모델에 임의의 지점에서 사용자가 데이터를 준다면 예측값은 사각형 점과 같이 나타날 것이다. 어떤 값이 더 정확한지는 두 데이터에 대한 실제 값을 계산해봐야 알겠지만, 직관적으로 다른 7개의 데이터와 확연히 다른 값을 보여주는 값보다 비슷한 값이 더 나은 예측값이라고 할 수 있다.



<그림 2-22> 과적합 문제 예시

과적합 문제의 원인은 다양하지만 대표적으로 다음과 같은 것들이 있다.

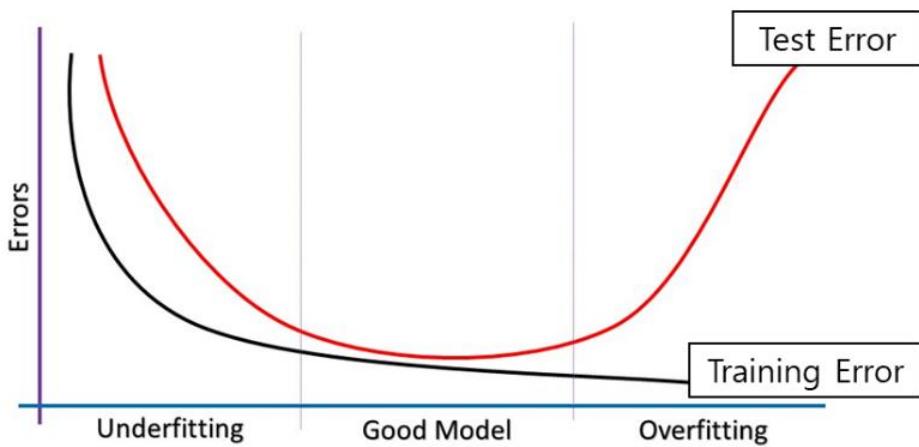
- 모델이 너무 복잡하다.
- 학습이 지나치게 오래 되었다.
- 학습 데이터가 부족하다.
- 학습 데이터에 오류가 있다.

반면, 과적합과 반대되는 현상인 과소적합(Underfitting) 문제도 있는데, 과소적합의 경우 모델이 데이터를 제대로 대표하지 못한다. 과소적합이 일어나면 학습 데이터에 대해서도 오차가 클 뿐만 아니라, 실제 사용에서도 예상치 못한 행동을 보일 수 있다. 과소적합의 원인은 다음과 같다.

- 모델이 너무 단순하다.
- 학습이 너무 빨리 끝났다.

1) 과적합의 판단

과적합의 원인은 다양하지만, 문제를 해결하기 위해서는 현상이 일어나는 것을 확인할 수 있어야 한다. 그러나 기계학습은 학습 오차가 최소가 되도록 진행되기 때문에, 학습 데이터만으로는 이를 판별해내기 어렵다. 그렇기에 일반적으로 기계학습을 실시할 때는 모아 놓은 데이터들의 일부만을 이용하여 학습을 하고, 학습에 사용하지 않은 나머지 데이터들로 테스트를 실시한다.



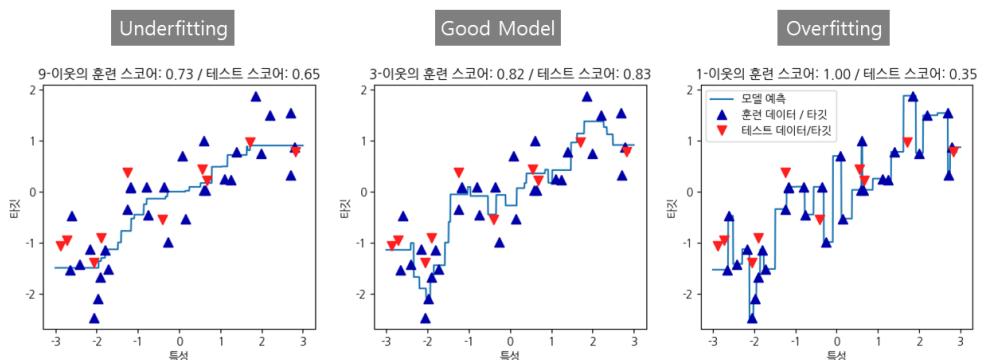
<그림 2-23> 과적합 여부 판단 방법

과적합의 특징은, 학습 데이터에 대한 오차는 작지만 학습해본 데이터가 아닌 테스트 데이터에 대해 예측할 때는 오차가 커진다는 것이다. <그림 2-23>은 학습 진행에 따른 학습 오차와 테스트 오차를 나타낸 것이다. 그림에서 학습 오차는 계속 줄어드나 테스트 오차는 처음에는 줄어들다가

어느 지점 이후로 증가하는 것을 볼 수 있는데, 이 때 과적합이 일어난다.

<그림 2-24>는 기계학습 기법 중 하나인 K-NN(K-Nearest Neighbor, K-최근접 이웃)을 사용한 학습 예시로, 모델의 복잡도에 따른 과적합과 과소적합을 보여준다. 삼각형 모양은 학습과 테스트에 사용한 데이터들이며 실선이 생성된 모델이다. 여기서는 오차가 아니라 맞춘 점수인 스코어를 사용했고, 점수가 1이면 모든 예측이 맞았다는 것을 의미한다.

각각을 살펴보면 9개의 이웃을 사용할 경우 가장 간단한 모델을 생성하며, 모델의 수가 줄어들수록 매우 복잡한 형태의 모델이 만들어지는 것을 알 수 있다. 최적의 모델은 3개의 이웃을 사용한 경우이며, 훈련 점수가 0.82, 테스트 점수가 0.83이다. 반면 이웃을 하나만 사용할 경우, 훈련 점수는 1로 가장 높은 점수를 기록했으나, 테스트 점수는 0.35로 가장 낮은 결과를 보여주었다.



<그림 2-24> K-NN의 과적합, 과소적합 예시

과적합과 과소적합의 판단은 이처럼 데이터를 학습용과 테스트용으로 나누어 사용하면 된다. 중요한 것은 학습에는 테스트 데이터를 사용하면 안 된다는 것이다. 일반적으로 데이터를 무작위로 섞어 75%를 학습에, 25%를 테스트에 사용한다. 좋은 인공지능 모델을 만들기 위해서는 학습이 지나쳐서도 안되고 부족해도 안되며 이는 반복적으로 모델을 개선해가며 학습해가며 확인할 수 있다.

3. 딥러닝(Deep Learning)

인공신경망은 층이 깊어질수록 표현력이 강해진다. 즉, 보다 복잡한 문제를 해결할 수 있다. 수학자와 컴퓨터 공학자들은 다층의 신경망을 자동적으로 업데이트 할 수 있는 역전파 알고리즘을 통해 인공신경망으로 더 추상적이고 복잡한 문제를 해결할 수 있을 것이라 생각했다. 앞서 예시로 설명한 두 개의 층을 갖는(입력층은 단순히 입력값을 받을 뿐 연산하지 않는다.) 신경망은 역전파를 통한 가중치와 바이어스 업데이트가 손쉽게 가능했다. 그러나 신경망이 더 깊어질 수록 학습은 어려워지고, 혹은 전혀 학습하지 못하는 상황이 발생하였다. 당시 연구자들이 마주친 문제는 아래 세 개의 문제들로 정리할 수 있다. [7] 이 문제들은 다시한번 인공신경망 연구에 빙하기를 초래하여 인공신경망 연구는 사장되었고, 대신 SVM이나 Random forest 등 다른 기계학습 기법들이 각광받았게 되었다.

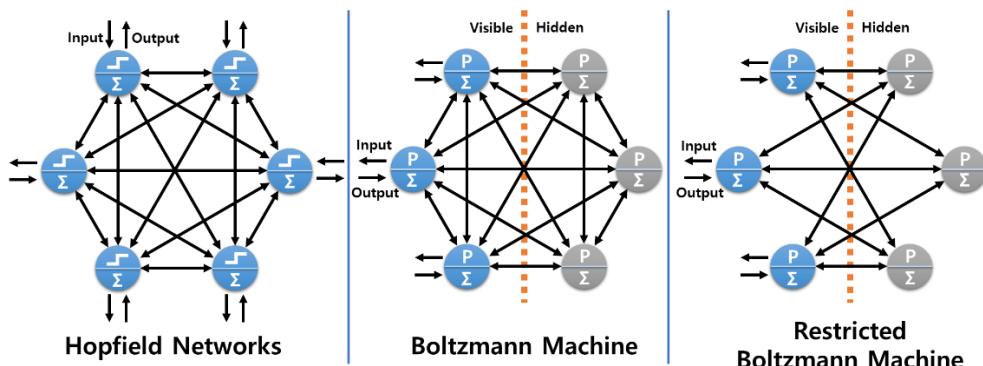
- Gradient Vanishing 문제
 - 신경망이 깊어질수록 입력층과 가까운 뉴런들이 학습되지 않았다.
- 대규모 신경망을 학습시키는데 시간이 너무 많이 소요되었다.
- Overfitting(과적합)문제
 - 신경망이 복잡해질 수록 파라미터의 수도 증가하는데, 이는 과적합 문제를 발생시킨다.

A. Gradient Vanishing 문제

인공신경망의 업데이트를 자동화하기 위해 고안된 역전파 알고리즘은 출력층으로부터 연산을 하며 업데이트를 한다. 그러므로 출력층으로부터 멀어질 때마다 기울기는 누적되며 곱해지고, 이는 역전파가 진행될수록 기울기의 곱들이 0에 수렴하는 현상을 야기하였다. 이처럼 역전파가 진행될수록 출력층으로부터 먼 뉴런의 기울기가 사라져가는 현상을 Gradient vanishing 문제라 한다. 반면, 이와 반대되는 현상인 Exploding gradient 문제도 존재하나, 대부분 Recurrent Neural Network에서 나타난다. [7]

1) 초기화 기법(Initialization method)

Gradient vanishing 문제를 해결하는 방법은 여러가지가 있지만, 보편적으로 사용할 수 있는 방법은 파라미터를 잘 초기화 하는 것과 적합한 활성화 함수를 사용하는 것이다. 파라미터를 잘 초기화 하는 방법은 2006년에 Geoffrey Hinton[○]] Restricted Boltzmann Machine(RBM) [3]을 응용하여 심층 신경망을 pre-training시킨 뒤, 학습을 시키는 Deep Belief Network를 만들면서 알려지게 되었다. [13] RBM은 Boltzmann machine [14]에서 각 층 내에 연결된 신경망들을 끊어낸 것이다. Boltzmann machine은 신경망이 층을 이루지 않고 모든 인공뉴런이 연결된 상태이다. 이 신경망은 Hopfield network에서 발전한 것으로, 몇 가지 차이점을 갖는다. Hopfield network는 Step function을 이용해 뉴런의 출력을 나타내고 모든 뉴런이 입출력을 담당한다. Boltzmann machine은 Boltzmann distribution을 이용하여 뉴런의 출력을 나타내며, 데이터 입출력이 가능한 가시화 유닛과 그렇지 않은 은닉 유닛으로 나뉘어져 있다. [7] <그림 2-25>는 Hopfield Network와 Boltzmann machine, 그리고 Restricted Boltzmann Machine(RBM)을 나타낸 것이다.

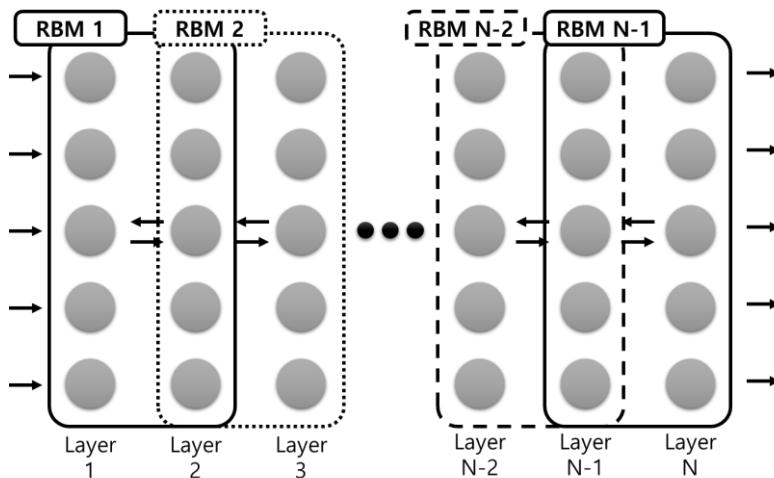


<그림 2-25> Hopfield Networks, Boltzmann Machine, RBM

Deep Belief Network에서는 RBM의 특징을 이용하여 신경망의 가중치와 바이어스를 초기화 하는 pre-training을 실시하는데, 이 때는 라벨링 된 학습 데이터 중 입력값만 사용한다. 이러한 특징 때문에 RBM을 이용한

pre-training은 unsupervised learning으로 구분되기도 한다.

RBM을 이용한 pre-training은 <그림 2-26>과 같이, 다층으로 구성된 신경망에서 입력층으로부터 차례대로 두 개씩 실시한다. 처음에는 입력층과 첫 번째 은닉층을 학습시켜서 입력값과 최대한 유사한 출력값을 내도록 학습시킨다. 첫 번째 RBM 학습이 끝나면 첫 번째 은닉층과 두 번째 은닉층을 RBM을 통해 학습시킨다. 이런 과정은 출력층 앞에 있는 은닉층과 출력층이 pre-training 될 때까지 계속된다. RBM을 통해 가중치와 바이어스 초기화를 마치면 실제 학습데이터를 이용해 학습을 시키면 된다. RBM을 이용한 DBN은 이전보다 더 깊고 거대한 신경망을 빠르게 학습시킬 수 있었으며 예측 성능이 좋아지는 효과를 낳았다.



<그림 2-26> Deep Belief Network에서의 Restricted Boltzmann Machine

DBN을 통해 인공신경망을 잘 초기화하면 깊고 거대한 네트워크를 빠르게 학습시킬 수 있을 뿐만 아니라 예측 성능도 좋아진다는 것이 밝혀지자, 다양한 초기화 기법들이 연구되었다. 이들 중, 현재 가장 기본적으로 쓰이는 기법은 Xavier initialization [15]과 He initialization [16]이다. 두 초기화 기법은 RBM과 같은 복잡한 과정을 거치지 않고 단순히 앞의 층과 뒤의 층의 뉴런 수(노드 수)를 사용한다. 초기화 방법은 해당 층에서의 가중치가 균등 분포를 따르게 하는 방법과 정규분포를 따르게 하는 방법이 있다.

균등한 분포를 따르는 경우에는 경계값을 계산하고, 정규분포를 따르는 경우에는 평균을 0으로 두고 앞뒤층의 뉴런 수를 이용해 표준편차를 계산하여 분포도를 만든 뒤, 무작위로 샘플링하여 사용한다. 두 기법 모두 가중치만 초기화하며, 바이어스의 경우 일반적으로 0으로 초기화한다.

아래 수식들은 Xavier initialization과 He initialization으로 인공신경망을 초기화 시키는 방법이다. He initialization은 Xavier initialization 기법이 ReLU 계열 함수를 활성화함수로 사용할 때 효율적이지 못한 것을 발견하고 이를 수정한 것이다. 이는 ReLU 함수가 입력값이 음수일 경우 출력을 모두 0으로 낸다는 특징으로 인한 것이다. Xavier initialization 기법과의 차이점은 이전 층의 뉴런 수만을 사용한다는 것이다.

- Xavier initialization
 - Uniform distribution $[-r, r]$

$$r = \sqrt{\frac{6}{n_{\text{inputs}} + n_{\text{outputs}}}} \quad (35)$$

- Normal distribution

$$\sigma = \sqrt{\frac{2}{n_{\text{inputs}} + n_{\text{outputs}}}} \quad (36)$$

- He initialization
 - Uniform distribution $[-r, r]$:

$$r = \sqrt{\frac{6}{n_{\text{inputs}}}} \quad (38)$$

- Normal distribution:

$$\sigma = \sqrt{\frac{2}{n_{\text{inputs}}}} \quad (39)$$

각 기법이 제시된 논문에는 Xavier initialization [15]의 경우 균등 분포 기법만, He initialization [16]의 경우 정규분포 기법만 기술되어 있으나, 최근에는 각 기법의 정규분포 기법 [17], 균등분포 기법, 그리고 활성화 함수에 따라 초기화 기법의 변수들을 설정하는 법도 연구되었다. [7]

2) Xavier Initialization(Normalized Initialization) [15]

Gradient vanishing, 혹은 Vanishing gradient 문제를 피하는 방법 중 하나는 적합한 활성화 함수를 사용하는 것이다. 매우 초창기의 인공신경망은 on/off 회로를 구현하기 위해 활성화함수로 Step function을 사용하였으나, 이진 분류(Binary classification)을 넘어 이미지 분류 등 보다 복잡한 문제를 해결하기 위해 Logistic function, Sigmoid function을 사용하였다.

- Logistic function

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}} \quad (40)$$

- Sigmoid function

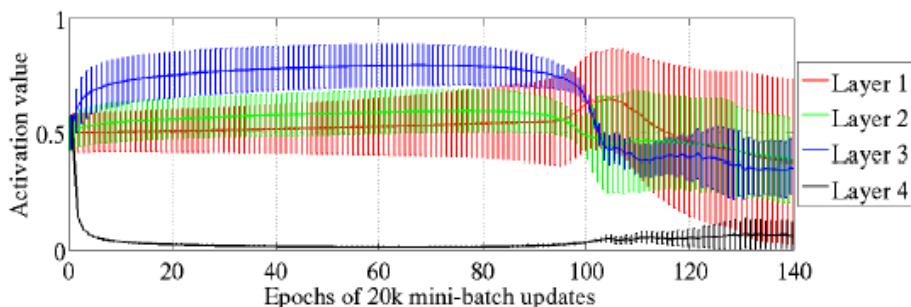
$$f(x) = \frac{1}{1 + e^{-x}} \quad (41)$$

Logistic function의 L은 함수의 최댓값을 정하며, x_0 는 함수의 중간 지점을, k는 함수의 가파름 정도를 정한다. Sigmoid function은 함수의 최댓값(L)을 1로 정하고 중간값(x_0)을 0으로 두며 k=1로 설정한 Logistic 함수의 기본 형태이다. [18] [19]

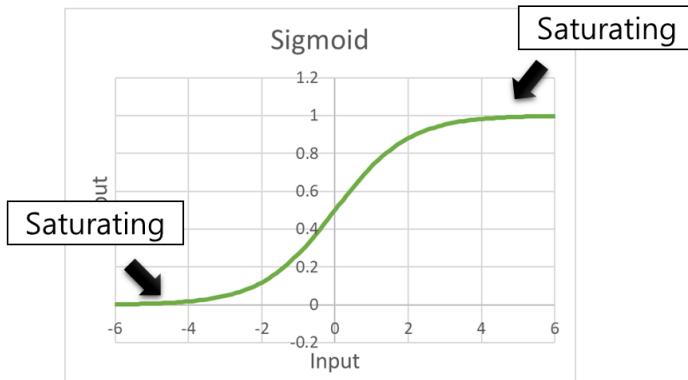
[15]에서 Xavier와 Bengio는 랜덤 초기화 기법과 당시 주류로 사용된 활성화함수 Sigmoid function의 조합이 학습을 저해한다는 것을 밝혔다. <그림 2-27>은 이러한 조합을 사용한 인공신경망을 학습시켰을 때, 각 Epoch마다 은닉층의 출력들의 평균과 표준편차를 나타낸 것이다. 그래프를 보면 마지막 층인 Layer 4는 학습 초기에 그 평균이 0에 빠르게 수렴하는 것을 알 수 있으며, 표준편차가 매우 작은 것을 볼 수 있다. 여기서 Layer 4는

출력층이므로 활성화 값이 0이 나왔다는 것은 출력값이 0이 나왔다는 것이며, 표준편차가 작다는 것은 대부분의 값이 0으로 출력되었다는 것이다. 이처럼 활성화 함수의 값이 극단적으로 치닫아 기울기가 0인 구역으로 가는 현상을 Saturating이라 한다. <그림 2-27>을 보면, 100 epoch를 넘어서면서 Layer 4의 평균이 0의 값에서 벗어나는 것을 확인할 수 있다. 또한 [15]에서는 여기에 은닉층을 하나를 더하여 5개의 레이어를 갖는 신경망을 학습시키면 Saturation에서 아예 나오지 못한다고 기술하였다.

Saturation 문제는 학습을 위한 역전파에 문제를 야기한다. 역전파는 Change rule에 기반하여 가중치와 바이어스를 출력층부터 입력층으로 업데이트 하는 기법이다. 그러나 신경망의 활성화 값이 0, 혹은 1에 머무르면 <그림 2-28>과 같이 활성화 함수의 기울기는 0이 되므로 업데이트가 전혀 일어나지 않게 된다.



<그림 2-27> Sigmoid를 사용한 인공신경망을 Random Initialization을 실시했을 때 각 층의 Activation 값의 평균(가로선)과 표준편차(세로선) [15]

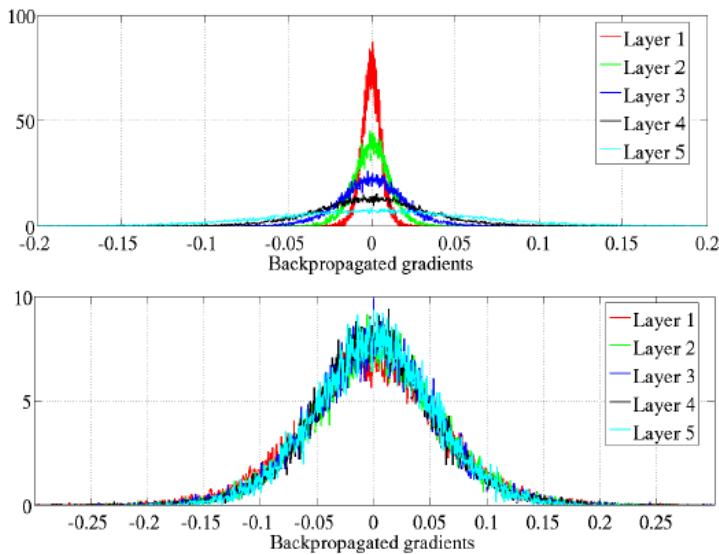


<그림 2-28> Sigmoid Function의 Saturating이 일어나는 지점

Xavier와 Bengio는 [15]에서 RBM과 같은 비지도학습 기법을 이용해 초기화하면, Sigmoid function을 활성화 함수로 사용하더라도 이러한 현상이 나타나지 않는다고 하였으나, RBM을 이용한 초기화 기법은 구조가 복잡하며 학습을 두 차례(Pre-training과 실제 학습) 하는 것과 마찬가지이므로 깊은 층의 신경망을 구축할 때는 사용하지 않는다. Xavier와 Bengio는 Sigmoid 이외에도 Hyperbolic tangent와 Softsign이라는 활성화함수를 사용하여 은닉층들의 가중치 분포와 활성화 값을 분석하여 Normalized Initialization 기법을 선보였다. 이를 요약하면 다음과 같다.

- Sigmoid 나 Hyperbolic tangent(\tanh)를 활성화 함수로 사용하고 표준정규 분포를 따라 가중치를 초기화할 경우, 학습이 매우 느리고, 안좋은 학습 결과를 보여준다.
- Softsign 을 활성화 함수로 사용한 신경망은 \tanh 함수보다 초기화 기법에 대해 강건한 양상을 보이는데, 이는 \tanh 함수보다 비선형성이 덜하기 때문으로 보인다.
- Tanh 함수를 사용한 신경망은 [15]에서 제시한 정규화 된 초기화기법 (Normalized Initialization)이 상당히 도움이 되었는데, 이는 각 층을 통과할 때마다 활성화 값과 역전파 시 그래디언트의 크기를 유지하기 때문이다.
- Xavier 와 Bengio 가 제시한 Normalized Initialization 기법은 정규분포를 따르는 초기화 기법을 사용하였기에 Normalized 라 한 것이 아니라, 학습이 진행되더라도 가중치의 그래디언트나 활성화 값들의 분포가 비슷하게 유지되기 때문이다. 실제로 [15]에 사용된 초기화 기법은 균등분포를 사용하였으며, 그 식은 다음과 같다.

$$W \sim U \left[-\frac{\sqrt{6}}{\sqrt{n_{in} + n_{out}}}, \frac{\sqrt{6}}{\sqrt{n_{in} + n_{out}}} \right] \quad (42)$$



<그림 2-29> Normalized Initialization 기법을 사용하지 않았을 때(상)와 사용하였을 때(하), 각 레이어의 역전파 그래디언트 분포

3) 적합한 활성화 함수의 선정

[15]에서 두 연구진들은 Gradient vanishing/explosion은 잘못된 활성화 함수의 선정과 초기화 기법에 그 원인이 있는 것을 밝혔다. 초창기에는 Sigmoid나 Hyperbolic tangent 함수가 보다 생물학적 뉴런에 가까운 행동을 보일 것이라 생각하였다. 이는 초기 인공 뉴런이 Step function을 사용하여 0과 1의 바이너리 출력을 낸 것을 생각하면 당연한 것이었다. 그러나 [15]의 연구로 컴퓨터 공학에서 인공신경망을 잘 학습시키기 위해 적합한 활성화 함수를 찾게 되었다. Nair와 Hinton은 Deep Belief Networks(DBN) [13]과 여기에 사용된 RBM [3]에서 ReLU 함수를 사용하였을 때 인공신경망의 학습이 개선되는 것을 확인하였다. ReLU를 사용하면 인공신경망 내부의 연결 중 일부가 끊어지게 되여 신경망에 Sparsity가 생기는데, 이로 인해 신경망이 더 잘 학습되고 예측 성능도 좋아지게 된다. Sparsity에 관한 연구는 ReLU 함수가 아니라 기존의 활성화 함수들을 사용할 때도 연구가 되었으며, [20]에 잘 정리되어 있으며, 정리하면 다음과 같다.

- Information disentangling
 - 신경망이 매우 조밀하게 얹히고 설계 있으면 모델의 표현력 밀도가 높아진다. 이로 인해 입력값이 조금만 바뀌어도 결과가 쉽게 바뀐다. 고양이 얼굴 사진을 입력값으로 넣었는데 고양이 얼굴의 각도가 달라져서 서로 다른 동물종으로 예측한다면, 이는 보편적으로 사용하기 어렵다. 더 낮은 밀도의 연결성, 희소성(Sparsity)을 갖는 인공신경망은 입력값의 작은 변화에 대해 더 강건하다. 일부 신경망의 연결이 끊어지는 대신, 여전히 연결되어 있는 신경망들에 의해 입력값이 보전되기 때문이다.
- Efficient variable-size representation
 - 같은 것을 나타내는 데이터라 할지라도 정보의 크기가 다양할 수 있다. 동일하게 1280x720 사이즈의 같은 강아지를 찍은 사진이라 하여도 근접해서 촬영한 사진과 먼 거리에서 촬영한 사진은 강아지에 대한 정보의 크기가 다르다. 활성 뉴런의 수를 다양하게 하면, 신경망 모델은 주어진 입/출력 데이터(학습 데이터)를 표현하기 위해 유효한 차원의 수를 조절할 수 있게 된다.
- Distributed but sparse
 - Distributed representation 이란 분류를 보다 분산되게 하는 것이다. 비행기의 종류를 구분한다고 할 때, 분류를 Airbus 380, Airbus 320, F-15, F-16 과 같이 모델별로 실시하면 필요한 뉴런의 수는 기하급수적으로 증가한다. 대신 크기를 나타내는 뉴런과 여객기/전투기를 나타내는 뉴런으로 나누어 표현한다면 보다 효율적인 표현이 가능하다. 밀도 높은 분산표현은 강력한 표현력을 가지며, 효율은 기하급수적으로 좋아진다. 희소한 분산표현 역시 기하급수적인 효율을 보여주지만, 효율을 끌어내는 힘은 살아있는 뉴런의 수가 된다. 그러므로 신경망의 연결 밀도를 정하는 것은 상황에 따라 절충되어야 한다.

ReLU 함수는 매우 쉽게 신경망의 연결 구성을 적절히 끊어준다는 장점이 있다. 균등 분포를 이용한 초기화를 한다면 절반의 신경망들은 음수 값을 활성화 함수로 넘겨주게 되므로 완벽한 0을 출력한다. 완벽한 0이라 한 이유는 이전의 Sigmoid나 Hyperbolic tangent 함수를 사용하는 경우, 신경망을 비활성화 시키는 추가적인 조치를 취하더라도 0.001과 같은 매우 작은 수를 출력하기 때문이다. 이런 값들은 0과 의미상 차이는 없지만 계산에 더 많은 부하를 준다.

또한 ReLU 함수는 다른 활성화 함수들과는 다르게 RBM과 같은 사전 훈련이 없어도 이에 근접하거나 능가하는 학습 결과를 보여준다. 즉, ReLU 함수를 사용하는 것은 모델의 Sparsity를 쉽게 구현할 수 있는 방법 일 뿐만 아니라, 사전 훈련이 필요 없어 학습 속도를 빠르게 하며, 훈련 정확도도 비슷하게 가져가거나 더 좋게 하는 방법이다.

그러나 ReLU는 일부 뉴런의 출력을 완전히 0으로 만들어낸다는 특징 때문에 일부 뉴런들이 학습 도중에 죽어 0만을 출력하는 dying ReLU 현상을 보이기도 한다. 특히 입력값이 음수일 경우 이러한 현상이 두드러지므로, 이를 피하면서 ReLU의 장점을 유지하기 위해 다양한 ReLU 함수의 변종들이 등장하였다. 입력값 0 근처에서의 기울기를 살리고자 한다면 Softplus를 사용하고, 음수 출력을 학습해야 할 경우, 음수를 출력할 수 있는 Leaky ReLU나 ELU를 사용하기도 한다. 그러나 일반적으로는 ReLU를 기본값으로 사용하여 학습해본 뒤, 다른 활성화 함수들을 시도해가며 최적의 모델을 찾는다. <그림 2-30>은 ReLU 함수와 ReLU 함수의 변종들을 나타낸 것이다. [7] [21]

- ReLU

$$\max(0, x) \quad (43)$$

- Softplus

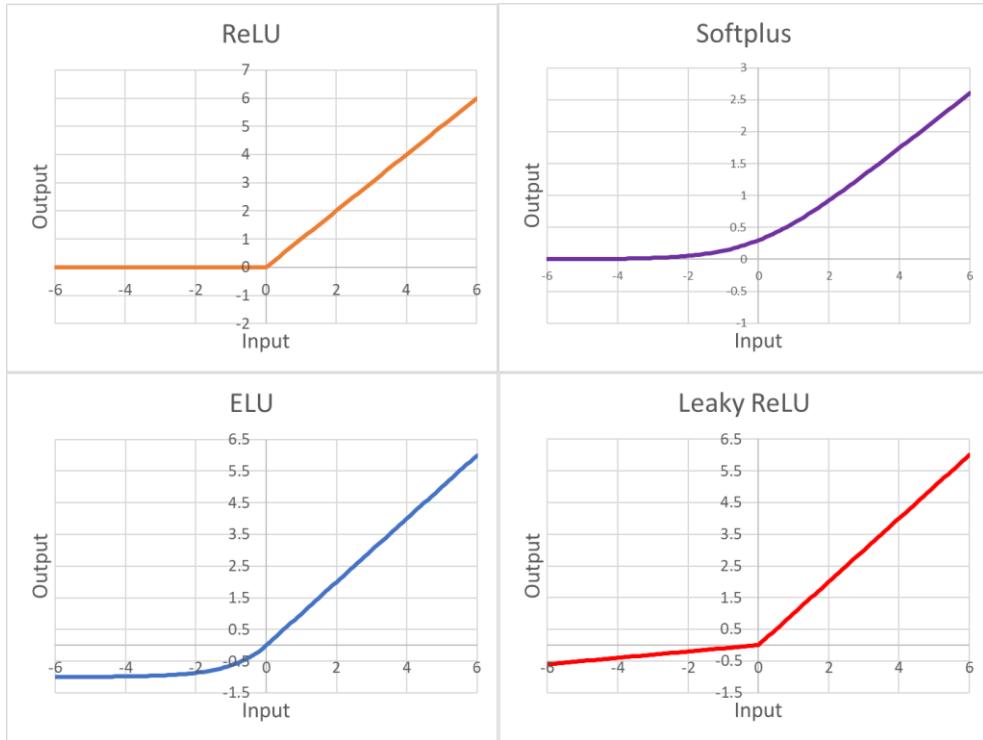
$$f(x) = \log(1 + e^x) \quad (44)$$

- ELU

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ a(e^x - 1) & \text{otherwise} \end{cases} \quad (45)$$

- Leaky ReLU

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases} \quad (46)$$



<그림 2-30> ReLU 함수와 ReLU 계열 함수들

B. 최적화 기법 [7] [22] [23]

인공신경망의 학습은 역전파 알고리즘을 통해 실시된다. 역전파 알고리즘의 핵심은 오차를 계산하고, 경사 하강법에 기반한 최적화 기법을 통해 가중치와 바이어스를 조정하여 인공신경망이 이전보다 더 나은 예측 성능을 보이도록 하는 것이다. 그러므로 어떤 최적화 기법을 선택하는지에 따라 학습의 성능이 달라지게 된다. 최적화 기법은 두 가지를 나누어 선택한다. 하나는 오차 함수를 계산할 때 사용하는 데이터셋을 사용하는지를 선택하고, 다른 하나는 어떤 경사 하강기법을 사용하는지를 선택하는 것이다. 일반적으로 오차함수로는 평균 제곱 오차(Mean Squared Error, MSE)를 사용하며, 문제와 모델에 따라 평균 제곱근 오차(Root Mean Squared Error, RMSE)를 사용하기도 한다.

$$MSE = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 \quad (47)$$

$$\begin{aligned} RMSE &= \sqrt{MSE} \\ &= \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2} \end{aligned} \quad (48)$$

- n: 사용한 학습 데이터의 수
- \hat{y} : 신경망의 예측값
- y: 데이터의 정답 값

1) 오차 계산 데이터셋에 따른 분류

인공신경망의 학습에서, 최적화를 위한 목적함수로는 오차함수가 사용된다. 오차함수의 값에 대해 업데이트가 필요한 모든 파라미터들의 편미분을 계산하고, 이를 이용하여 오차가 줄어드는 방향으로 모든 파라미터들을 업데이트한다. 이 때, 오차를 구하기 위해 학습 데이터셋을 어떻게 사용하는지에 따라 방법이 세 가지로 나뉘어진다.

- Batch Gradient Descent(BGD)

BGD는 오차 함수 계산 시, 모든 학습데이터에 대한 오차를 계산하여 하나의 값으로 출력하고, 그 값에 기반하여 모든 파라미터들을 업데이트하는 방법이다. MSE에서 학습 데이터 수 n은 모든 학습데이터의 수와 같다.

BDG는 모든 데이터를 이용하기 때문에 전체 학습 데이터에 대해 콜고루 학습이 가능하고, 구현하기 쉽다는 장점이 있으나, 출력하고 오차를 계산해야 하므로 학습이 오래 걸리며, 학습 데이터가 많아지면 메모리 부족 문제가 생길 수 있다.

- Stochastic Gradient Descent(SGD)

SGD는 무작위로 학습 데이터들 중 하나를 선정하여 학습을 시키는 기법이다. 그러므로 연산의 부담은 줄어들지만, 학습이 완만하게 진행되지 않고 요동치며 진행되어 최적점에 가만히 있지 않는다. 현재 학습 단계에서 어떤 최적점에 있다고 하더라도 다음 학습을 진행하면 오차가 커질 수 있다. 그러나 요동치는 특징 때문에 지역 최적점에 빠졌다 할지라도 빠져나올 수 있어 BGD보다 전역 최적점을 빠르게 잘 찾는다는 장점이 있다.

그러나 SGD도 전체 데이터에 대해 학습해야 하므로, 학습 데이터 수만큼 훈련을 반복한다. 전체 학습 데이터에 대한 훈련이 완료되면 이를 1 epoch라고 부르며, 비로소 한 번의 학습이 완료되었다고 한다. 예를 들어, 100개의 학습 데이터로 인공신경망을 50회 학습시킨다면, BGD는 계산 반복 횟수(iteration)^o 100번 일어날 것이며, 전체 데이터를 한번에 학습하므로 epoch 역시 100번 일어났다고 할 수 있다. 반면, SGD는 한 번의 업데이트에 하나의 데이터만 사용하므로, 1 epoch의 학습에는 100번의 계산이 반복(iteration)되는 것이다. 그러므로 총 50회의 학습을 실시하면 5,000번의 신경망 업데이트가 일어난다.

- Mini-batch Gradient Descent(MGD)

MGD는 BGD와 SGD를 절충한 방법이다. 전체, 혹은 하나의 데이터만 이용하여 학습을 하지 않고, 학습 데이터를 여러 개의 소규모 그룹(Mini-batch)으로 나누어 학습을 시킨다. 앞의 예제와 같이 100개의 학습 데이터를 50회 학습시킬 때, 10개의 미니배치로 나누어 학습시킨다면, 1 epoch당 10번의 파라미터 업데이트가 일어나며, 총 500회의 iteration^o 일어난다.

미니 배치를 구성할 때는 데이터 영역에 걸쳐 고른 학습을 하기 위해 무작위로 데이터를 선정한다. MGD는 SGD보다 안정적인 학습양상을 보이며, BGD보다 빠르게 학습하며, 미니배치의 크기를 조절하여 메모리 사용량을 조절할 수 있다. 그러나 SGD에 비해 우둔하고 BGD에 비해 요동치는 특성을 보인다. 이는 미니배치의 크기를 통해 절충해가며 사용해야 한다.

2) 경사 하강 방법에 따른 분류 [7] [24] [25]

인공신경망의 학습 방법은 학습 데이터를 어떻게 사용하느냐 뿐만 아니라 하강 기법에 따라서도 나눌 수 있다. 두 가지 모두 Gradient Descent 기법으로서 사용되나, 일반적으로 경사 하강 방법은 Optimizer라고 부른다. 본 논문에서는 이들을 확연하게 구분 짓고자, 경사 하강 방법은 Optimizer로 부른다.

a. Vanilla Optimizer

프로그래밍 분야에서 Vanilla라 호칭되는 것은 기본 형태, 기본 코드를 의미한다. Vanilla optimizer 역시 기본적인 경사 하강법과 BGD를 사용한 기법을 말한다. 다음은 파라미터 최적화를 위한 Vanilla Optimizer 수식이다.

$$\theta = \theta - \alpha \nabla_{\theta} J(\theta) \quad (49)$$

- θ : 신경망의 파라미터(가중치, 바이어스)
- α : 학습률(Learning rate, 혹은 step size라고도 불린다.)
- $J(\theta)$: 오차함수
- $\nabla_{\theta} J(\theta)$: θ 에 대한 오차함수 기울기

b. Momentum Optimizer

Vanilla Optimizer는 매우 간단하게 구현할 수 있었지만, <그림 2-31>과 같은 문제에서는 매우 느려지거나 요동치는 특성을 보인다. 이런 타원형 문제에 대해서는 Steepest optimization 기법이 뛰어난 성능을 보이지만, 기계 학습 기법은 함수의 형태를 모르는 상태에서 시작하기 때문에 해결법이 되지 못한다. Optimizer가 풀어야 할 문제는 타원형 문제일 수도 있고, 안장형 문제일 수도 있다. 혹은 더 높은 차원의 문제가 될 수 있다. Momentum optimizer는 단순히 기본적인 경사 하강기법에 가속화 시키는

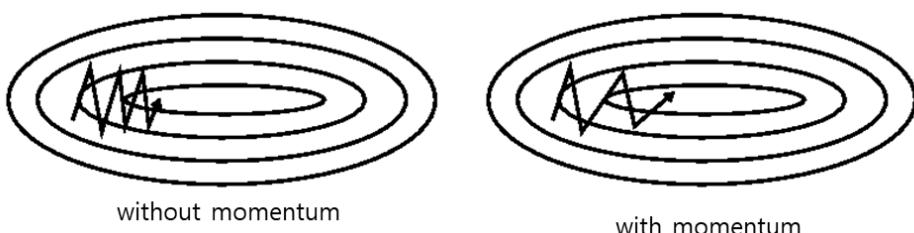
항을 추가해 이를 해결하였다. 기본적인 원리는 언덕에서 공을 굴리는 것과 같아, 첫 학습 단계는 Vanilla optimizer와 같으나 학습이 진행될 수록 경사의 하강 속도가 빨라진다. 다음은 Momentum optimizer 전개 방법이다.

$$v_t = \gamma v_{t-1} + \alpha \nabla_{\theta} J(\theta) \quad (50)$$

$$\begin{aligned} \theta &= \theta - v_t \\ &= \theta - (\gamma v_{t-1} + \alpha \nabla_{\theta} J(\theta)) \end{aligned} \quad (51)$$

- θ : 신경망의 파라미터(가중치, 바이어스)
- α : 학습률(Learning rate, 혹은 step size라고도 불린다.)
- $J(\theta)$: 오차함수
- $\nabla_{\theta} J(\theta)$: θ 에 대한 오차함수 기울기
- v_t : 모멘텀
- γ : 항력(모멘텀 v_t 에 제약을 가하며, 보통 0.9 근처 값을 사용)

Momentum optimization 기법에서 첫 모멘텀은 0으로 시작하며, 이전의 파라미터 업데이트 수치가 다음 파라미터 업데이트에 추가적으로 일어나게 된다. 여기서 항력 γ 는 모멘텀에 제약을 가하는 항으로서, 가속을 조절한다. 학습이 진행될 수록 가속되는 특성 때문에 Momentum optimizer는 Vanilla optimizer보다 다양한 문제에 대해서 빠르게 학습할 수 있다. 그러나 최적점에 도달할 때, 가속이 지나쳐 그대로 최적점을 통과하는 경우도 발생한다. 반대쪽 방향의 언덕이 충분히 크면 다시 최적점으로 가겠지만, 그렇지 않다면 의도하지 않은 곳에 가게 된다.



<그림 2-31> 타원 문제에서의 Momentum 항의 유무 차이

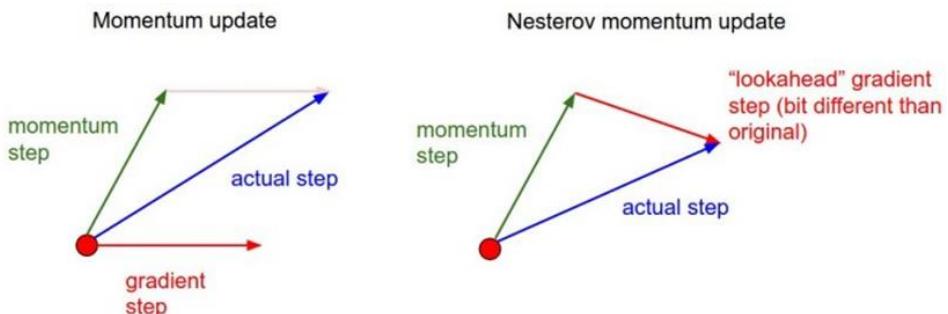
c. Nesterov Accelerated Gradient(NAG)

NAG는 모멘텀 기법이 최적점 근처에서 너무 빨라지는 현상을 조절하기 위해 고안되었다. 최적점에 가까워질 때, NAG는 모멘텀 항의 크기를 줄이도록 한다. 다음은 NAG 기법을 나타낸 수식이다.

$$v_t = \gamma v_{t-1} + \alpha \nabla_{\theta} J(\theta + \gamma v_{t-1}) \quad (52)$$

$$\theta = \theta - v_t \quad (53)$$

- θ : 신경망의 파라미터(가중치, 바이어스)
- α : 학습률(Learning rate, 혹은 step size라고도 불린다.)
- $J(\theta)$: 오차함수
- $\nabla_{\theta} J(\theta)$: θ 에 대한 오차함수 기울기
- v_t : 모멘텀
- γ : 항력(모멘텀 v_t 에 제약을 가하여, 보통 0.9 근처 값을 사용)



<그림 2-32> Momentum optimizer와 NAG 기법의 비교

NAG는 새로운 항을 만들어 모멘텀 항의 크기를 조절한 것이 아니라, 오차함수에 이전 모멘텀을 추가하여 조절하였다. Momentum optimizer와 다른 점은 파라미터에 대한 오차함수의 기울기를 구하는 $\nabla_{\theta} J(\theta + \gamma v_{t-1})$ 항에서 발생한다. 오차함수 계산에 θ 가 아닌 $\theta + \gamma v_{t-1}$ 를 사용해서, 현재 파라미터 단계에서 모멘텀만큼 움직였을 때의 기울기가 어떻게 되는지 계산한다.

<그림 2-32>는 두 기법의 차이를 잘 보여주는 그림이다. Momentum 기법은 현재 스텝(원)에서의 기울기와 모멘텀을 더해 다음 파라미터(Actual step의 화살표 머리)로 넘어간다. 그러나 NAG는 현재 스텝에서 모멘텀만큼 갔을 때, 즉, Momentum step의 화살표 머리 지점에서의 기울기를 구하고, 이를 모멘텀 항과 더하여 다음 지점으로 움직인다.

d. Adaptive Gradient(AdaGrad) [26]

AdaGrad는 John Duchi가 제안한 기법으로서, 학습률을 일정하게 가져가는 것이 효율적인지를 고민하고 연구해낸 결과다. 학습률을 크게 가져가면 최적점에 빨리 도착하지만 최적점 근처에서는 요동치며 도달하지 못하거나 오히려 발산하는 현상이 발생한다. 반대로 학습률을 적게 가져가면 최적점에는 더 잘 도달하지만 학습 속도가 느려지며, 지역 최적점에 빠져버릴 수 있다. AdaGrad는 파라미터의 변화가 크면 학습률을 줄이고, 파라미터의 변화가 적으면 학습률을 키운다.

$$G_t = G_{t-1} + (\nabla_{\theta} J(\theta))^2 \quad (54)$$

$$\theta = \theta - \frac{\alpha}{\sqrt{G_t + \epsilon}} \nabla_{\theta} J(\theta) \quad (55)$$

- θ : 신경망의 파라미터(가중치, 바이어스)
- α : 학습률(Learning rate, 혹은 step size라고도 불린다.)
- $J(\theta)$: 오차함수
- $\nabla_{\theta} J(\theta)$: θ 에 대한 오차함수 기울기
- G_t : 기울기 제곱의 누적 값
- ϵ : 0으로 나누어지는 것을 방지하기 위한 값 ($1e-4 \sim 1e-8$)

AdaGrad는 학습률을 일정하게 가져가는 기법들보다 더 부드럽게 최적점에 도달하지만, 때로는 G_t 가 급격하게(Aggressive) 커져 학습이 도중에 멈추는 경우도 발생한다. 이는 특히 딥러닝 시 두드러지게 나타나는 것으로 알려져 있다. [7] [25]

e. RMSProp(Root Mean Square Propagation)

RMSProp은 AdaGrad에서 G_t 항이 급격하게 커져 학습이 진행될수록 학습률이 급격하게 줄어드는 현상을 지수 평활법(Exponential Smoothing)을 통해 완화시킨 기법이다. 아래는 RMSProp 알고리즘을 나타낸 것이다.

$$G_t = \gamma G_{t-1} + (1 - \gamma)(\nabla_{\theta} J(\theta))^2 \quad (56)$$

$$\theta = \theta - \frac{\alpha}{\sqrt{G_t - \epsilon}} \nabla_{\theta} J(\theta) \quad (57)$$

- θ : 신경망의 파라미터(가중치, 바이어스)
- α : 학습률(Learning rate, 혹은 step size라고도 불린다.)
- $J(\theta)$: 오차함수
- $\nabla_{\theta} J(\theta)$: θ 에 대한 오차함수 기울기
- G_t : 기울기 제곱의 지수평활 누적 값
- ϵ : 0으로 나누어지는 것을 방지하기 위한 값 ($1e-4 \sim 1e-8$)
- γ : Decay rate (일반적으로 0.9, 0.99, 0.999를 사용)

RMSProp은 Hinton에 의해 만들어졌지만, 개별적인 논문으로 발표되지 않고, Hinton의 Coursera 강의자료에 수록되어 있다. 비록 정규 논문으로 발표되진 않았으나, 이를 이용한 많은 연구진들이 해당 강의를 “slide 29 in lecture 6”라고 수록하며, 본 논문에서도 이를 따랐다. [27]

f. Adaptive Momentum Estimation(Adam)

Adam은 현재 딥러닝 학습에서 기본적으로 사용되는 Optimizer로서, Momentum optimizer와 RMSProp을 합친 것이다. 그렇기에 최적화 과정이 상당히 복잡하지만, 학습률을 조절하면서 동시에 학습을 가속화할 수 있다. 다음은 Adam의 파라미터 업데이트 과정이다.

$$m = \beta_1 m + (1 - \beta_1) \nabla_{\theta} J(\theta) \quad (58)$$

$$v = \beta_2 v + (1 - \beta_2) (\nabla_{\theta} J(\theta))^2 \quad (59)$$

$$\hat{m} = \frac{m}{1 - \beta_1^t} \quad (60)$$

$$\hat{v} = \frac{v}{1 - \beta_2^t} \quad (61)$$

$$\theta = \theta - \frac{\alpha}{\sqrt{\hat{v}} + \epsilon} \hat{m} \quad (62)$$

- θ : 신경망의 파라미터(가중치, 바이어스)
- α : 학습률(Learning rate, 혹은 step size라고도 불린다.)
- $J(\theta)$: 오차함수
- $\nabla_{\theta} J(\theta)$: θ 에 대한 오차함수 기울기
- m : 모멘텀 항
- v : 기울기의 누적 값(지수 평탄화를 통해 완화)
- β_1 : Momentum 기법의 항력과 같지만, 지수 평탄화를 도입(일반적으로 0.9를 사용)
- β_2 : Decay rate(일반적으로 0.999를 사용)

- ε : 0 으로 나누어지는 것을 방지하기 위한 값 ($1e-4 \sim 1e-8$)

- t: 업데이트 횟수(Iteration step)

Adam^{o)} Momentum과 RMSProp과 다른 점은 \hat{m} 과 \hat{v} 항을 도입한 것이다. Adam은 m과 v가 0으로 초기화되므로 학습 초기에 0으로 가려는 움직임을 보여 이를 방지하기 위함이다. [28]

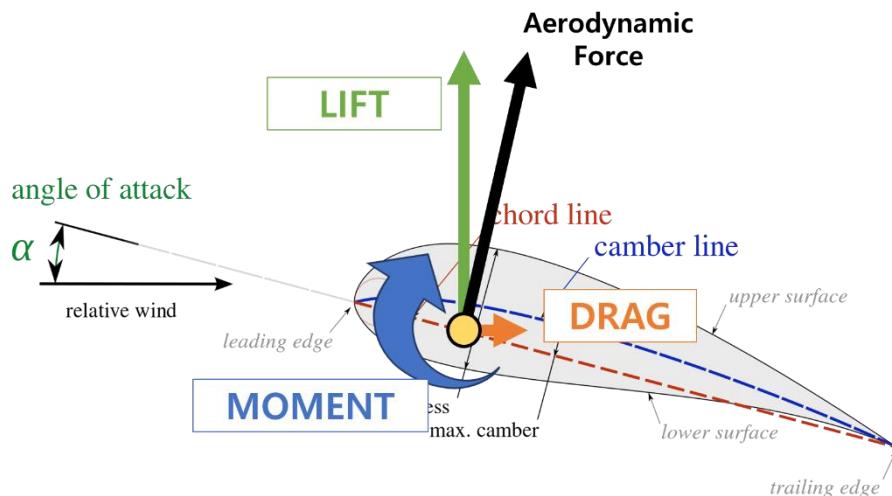
4. 정리

- 인공신경망은 기계학습 기법 중 하나이다.
- 딥러닝은 인공신경망을 다양한 기법으로 강화한 것이다.
 - 딥러닝 기법을 통해 대규모 인공신경망을 효율적으로 훈련시킬 수 있게 되었다.
- 대표적인 딥러닝 기법은 다음과 같다.
 - DNN: 심층 신경망. 층이 깊고 넓은 대규모 인공신경망.
 - CNN: 이미지 처리에 특화된 신경망
 - RNN: 순차 데이터 처리에 특화된 신경망
- 최근 딥러닝 기법들은 기계학습 라이브러리에 함수화가 잘 되어 있으므로, 딥러닝 모델을 만들 때 잘 모르겠다면 다음 조건들을 사용한다.
 - 초기화 기법: Xavier Initialization
 - 활성화 함수: ReLU
 - 경사하강기법: Adam

제 2절 에어포일 설계

1. 에어포일

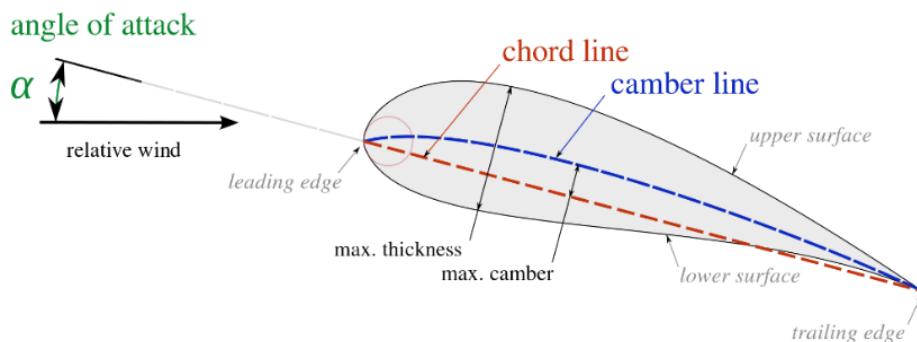
에어포일은 공기역학적 힘(Aerodynamic force, 공력)을 발생시키는 비행기 날개를 수직으로 자른 단면을 말한다. [29] 공력을 발생시키기 위해서는 유체와 에어포일 사이에 상대적인 속도가 필요한데, 상대적인 속도를 날개를 회전시켜서 얻는가, 앞으로 움직여서 얻는가에 따라 회전익과 고정익기로 나뉘어진다. 공력은 유체의 움직임(유동)과 에어포일의 형상에 영향을 받기 때문에, 마하 수(Mach number), 레이놀즈 수(Reynolds number), 유체의 종류 등의 유동 특성과 에어포일의 모양에 따라 다르게 나타난다. 그렇기에 에어포일의 표준화된 공력 특성을 계산하기 위해 일반적으로 공력을 동압력으로 나눈 공력 계수를 이용하여 성능을 평가하고 나타낸다.



<그림 2-33> 에어포일에 작용하는 힘

에어포일의 성능 지표를 나타내는 공력계수는 대표적으로 양력계수, 항력계수, 모멘트 계수가 있다. 양력계수는 공력을 유동 방향의 수직한 방향으로 나눈 힘으로서, 양력계수가 클수록 더 많은 중량을 띄울 수 있다. 항

력계수는 공력을 유동방향의 수평 방향으로 나눈 것으로 항력계수가 작을 수록 더 효율적인 비행을 할 수 있다. 모멘트 계수는 피칭 모멘트를 뜻하며, 공력이 발생할 때, 에어포일이 위아래로 돌아가려는 특성을 나타낸다. 이는 항공기의 안정성을 좌우하며, 일반적으로 에어포일 길이의 1/4 지점의 모멘트를 사용한다. 에어포일의 형상은 성능에 영향을 주기 때문에, 에어포일의 형상 특징들을 다음과 같이 체계화하여 정의한다. [29] [30]



<그림 2-34> 에어포일 형상 구성도

- Upper surface(Suction surface): 에어포일의 윗면
- Lower surface(Pressure surface): 에어포일의 아랫면
- Leading edge(앞전): 에어포일의 맨 앞 지점
- Trailing edge(뒷전): 에어포일의 맨 끝 지점
- Chord line(시위선): Leading edge 와 Trailing edge 를 잇는 선
- Thickness: 두께를 측정하는 방법은 두 가지로, Camber line 에 수직하게 측정하는 방법과 Chord line 에 수직하게 측정하는 방법이 있다.
- Mean camber line(Camber line, 평균 캠버): 윗면과 아랫면 중앙을 이은 선
- Angle of Attack(AOA, 받음각): 유동 방향과 시위선 사이의 각도

2. 에어포일 설계

에어포일을 설계하는 방법은 크게 검색, 최적화 기법, 역설계 기법 세 가지로 나눌 수 있다. 이들은 본 논문에서 기술한 딥러닝 기반의 설계 기법과 구분하기 위하여 전통적인 방법이라 부를 것이다. 전통적인 방법이라 하여 부정적인 것은 아니다. 최근 개발되고 있는 다양한 인공지능 알고리즘은 딥러닝 알고리즘이 큰 부분을 차지함에도 불구하고, 여전히 Rule based model, decision tree, 최적화 기법 등 다양한 전통적인 알고리즘들과 결합하여 사용되고 있다. 그러므로 전통적인 설계 기법은 여전히 유효하며, 딥러닝 알고리즘과 결합하여 더 높은 성능과 효율을 낼 수 있다.

본 단락에서는 검색, 최적화 기법, 역설계와 같은 전통적인 에어포일 설계 기법들과, 다른 연구자들이 에어포일 설계에 딥러닝을 접목한 사례들을 소개하며 각 기법들의 장점과 단점을 살펴볼 것이다.

A. 검색을 통한 설계

에어포일을 설계하는 가장 기초적이고 기본적인 방법은 검색을 하는 것이다. 구글에 에어포일을 검색하면 다양한 에어포일 데이터베이스들을 찾을 수 있는데, 가장 대중적인 것은 Airfoil Tools [1]과 UIUC Airfoil Data Site [2]를 꼽을 수 있다. 특히, [1]에는 단순 익형 좌표만이 아니라, 유동조건을 달리하여 XFOIL [5]로 해석한 결과를 볼 수 있다.

검색은 가장 기초적이며, 쉽게 선정할 수 있는 방법이다. 마치 도서관에 책을 빌리려 가는 것과 같다. 찾아내야 할 책이 무엇인지 알면 도서관에 가서 그 책만 빌려오면 된다. 그러나 명료하게 어떤 책을 빌려야 할지 모르겠다면, 시간을 들여가며 도서관에서 하나씩 찾아봐야 할 것이다. 마찬가지로, 기초적인 에어포일 지식만 알고 있다면 어떤 에어포일이 적합한지 찾아 사용할 수 있다. 그러나 아직 잘 모르겠다면 시간을 들여 데이터베이스를 탐색해서 최적의 에어포일을 찾아낼 수 있다. [1]과 [2]에는 1,600여 개의 에어포일 데이터가 있으니 시간만 충분하다면 어쩌면 찾아낼 수 있을 것이다.

A-Z Directory

[2032c.dat](#) \ Dillner 20-32-C low Reynolds number airfoil \ [2032c.gif](#) \ \
[a18.dat](#) \ Archer A18 F1C free flight airfoil(original) \ [a18.gif](#) \ \ Low Reynolds number data
[a18sm.dat](#) \ Archer A18 F1C free flight airfoil (smoothed) \ [a18sm.gif](#) \ \ Low Reynolds number data
[a63a108c.dat](#) \ NASA/AMES modification of the NACA 63A-108 airfoil \ [a63a108c.gif](#) \ \
[ag03.dat](#) \ Drela AG03 (flat aft bottom) airfoil \ [ag03.gif](#) \ \ Ref [3]
[ag04.dat](#) \ Drela AG04 airfoil \ [ag04.gif](#) \ \ Ref [3]
[ag08.dat](#) \ Drela AG08 airfoil \ [ag08.gif](#) \ \ Ref [3]
[ag09.dat](#) \ Drela AG09 airfoil \ [ag09.gif](#) \ \ Ref [3]
[ag10.dat](#) \ Drela AG10 airfoil \ [ag10.gif](#) \ \ Ref [3]
[ag11.dat](#) \ Drela AG11 airfoil \ [ag11.gif](#) \ \ Ref [3]
[ag12.dat](#) \ Drela AG12 airfoil \ [ag12.gif](#) \ \ Ref [3]
[ag13.dat](#) \ Drela AG13 airfoil \ [ag13.gif](#) \ \ Ref [3]
[ag14.dat](#) \ Drela AG14 airfoil \ [ag14.gif](#) \ \ Ref [3]
[ag16.dat](#) \ Drela AG16 airfoil \ [ag16.gif](#) \ \ Ref [3]
[ag17.dat](#) \ Drela AG17 airfoil \ [ag17.gif](#) \ \ Ref [3]
[ag18.dat](#) \ Drela AG18 airfoil \ [ag18.gif](#) \ \ Ref [3]
[ag19.dat](#) \ Drela AG19 airfoil \ [ag19.gif](#) \ \ Ref [3]
[ag24.dat](#) \ Drela AG24 airfoil used on the Bubble Dancer R/C DLG \ [ag24.gif](#) \ \ Ref [3]
[ag25.dat](#) \ Drela AG25 airfoil used on the Bubble Dancer R/C DLG \ [ag25.gif](#) \ \ Ref [3]
[ag26.dat](#) \ Drela AG26 airfoil used on the Bubble Dancer R/C DLG \ [ag26.gif](#) \ \ Ref [3]
[ag27.dat](#) \ Drela AG27 airfoil used on the Bubble Dancer R/C DLG \ [ag27.gif](#) \ \ Ref [3]
[ag35.dat](#) \ Drela AG35 airfoil \ [ag35.gif](#) \ \ Ref [3]
[aacc.dat](#) \ Drela AG36 airfoil \ [aacc.gif](#) \ \ Ref [3]

<그림 2-35> UIUC Airfoil Database [2]

검색을 통해 에어포일을 찾아낸다는 것은 매우 쉽다. 다른 프로그램이나 도구가 필요할 것이 아니고, 수식을 만들어서 문제를 풀어낼 필요도 없다. 인터넷이 연결된 컴퓨터나 스마트폰만 있다면, 그리고 웹 브라우저가 있다면 찾아낼 수 있다. 그러나 검색의 한계는 데이터베이스에 원하는 것이 없다면 찾아낼 수 없으며, 설계자의 요구 조건을 정밀하게 만족시킬 수 없다. 결국엔 데이터베이스 안에 있는 자료들만 찾아낼 수 있기 때문이다. 만약 설계자가 이를 감수하고 오차를 허용하여 다른 설계 조건들을 변경 할 것이라면 이것 만으로도 충분하다. 그러나 더 정밀한 설계를 하고자 한다면 최적화 기법이나 역설계 기법을 사용해야 한다.

B. 최적화 기법

설계의 가장 큰 문제는 데이터가 아무리 많이 축적된다 할지라도 이는 결국 과거의 디자인이며, 기술을 발전시키기 위해서는 새로운 방법이 필요하다는 것이다. 검색으로 설계를 한다는 것은 과거의 디자인을 답습한다는 것과 같으며, 새로운 요구와 문제에 적합하지 않은 결과를 도출하기도 한다.

최적설계는 문제에 대한 최적의 답안을 찾는 방법으로서, 설계에 필요한

변수, 설계자가 요구한 조건, 그리고 물리적 상관관계 등 다양한 조건과 수식들에 기반한 최적의 답이 무엇인가를 도출하여 문제를 해결하기 위한 최적의 답은 무엇인지 알려준다. 최적 설계를 하기 위해서는 무엇을 이용해 어떤 것을 최적화할지를 잘 선정해야 한다. 그리고 물리적 한계, 요구도에 의한 한계 등 한계지점을 명확히 해야 한다. 아래는 최적 설계를 하기 위해 필요한 세 가지 필수 요소들이다.

- 설계 변수

- 설계 변수는 설계자가 조절을 하고자 하는 변수를 말한다. 무엇을 이용하여 설계를 진행할 것인지를 설계 변수를 통해 지정한다. 설계 변수가 많아진다면 문제의 차원수는 높아지고 복잡해진다. 지나치게 복잡한 문제는 최적화를 힘들게 하므로 최소한의 설계 변수를 이용하여 설계를 하는 것이 바람직하다.

- 목적함수

- 목적 함수란 최적화 대상을 설계 변수들을 이용하여 함수로 만든 것이다. 목적함수를 만들기 위해서는 최적화 대상과 설계 변수들 사이의 상관관계를 잘 알아야 한다. 설계자가 둘 사이의 관계를 잘 이해하지 못한 채 목적함수를 만들게 되면 잘못된 설계 결과를 낳을 수 있다.

- 제약조건

- 모든 문제에는 제약조건이 존재한다. 여기에는 물리적으로 말이 안되는 조건들도 있으며, 비용과 같은 문제들도 제약이 된다. 제약 조건은 문제의 범위를 줄여주며, 굳이 해 볼 필요가 없는 경우를 제거해준다. 일반적으로 등호나 부등호를 이용하여 제약조건을 선정한다.

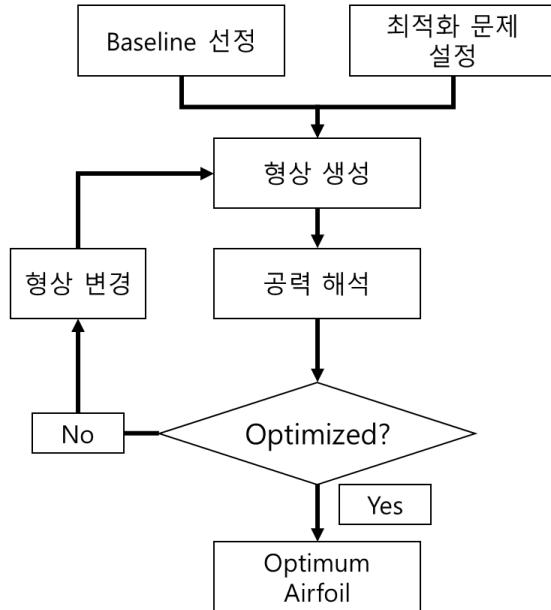
최적 설계기법은 수많은 연구자들이 최적점을 더 빠르고 잘 찾아내는 방법을 연구하였기에 접근법과 풀이법이 매우 방대하다. 최적화 함수를 만

들어내는 방법도 Simplex method나 LaGrange method 등 다양하다. 최적화하는 과정도 Gradient Decent 기법부터 Golden Section Search, Hessian Matrix 등 매우 많다. 그러나 공통적으로는, 설계 변수로 목적함수와 제약조건들을 만들어내고, 목적함수와 제약조건들로 최적화 대상 함수를 만들어내며, 초기 설계 지점을 설정하고 최적화 기법들(Gradient Descent 등)을 이용하여 최적점을 찾아낼 때까지 반복 계산한다. 전역 최적점을 찾고 싶으면 Genetic Algorithm과 같이 변칙적인 결과를 낼 수 있는 알고리즘과 융합하여 사용한다.

최적설계 기법을 이용하면, 설계자는 목표값에 최적화된 값을 얻을 수 있다. 검색 기법보다 정밀하며, 물리적/기하학적 성질에 근본하기에 확실한 방법이다. 하지만 그렇기에 최적설계 기법은 매우 전문적인 방법이며, 관계식을 잘못 세울 경우 의도와는 전혀 다른 값을 도출할 수 있다. 만약 전역 최적점을 찾고자 한다면 Genetic Algorithm이나 Stochastic Algorithm을 사용하는데, 두 기법은 매우 많은 연산량을 요구하므로 최적점을 찾는데 시간이 오래 걸린다.

1) 에어포일 최적설계 기법

쉬운 최적설계 기법 예제들은 목적 함수와 변수들 사이의 명료한 수식을 통해 이루어진다. 하지만 에어포일 최적설계기법은 명료한 관계식이 없기에 직접적인 수식을 사용하지 않는다. 대신 단순히 Baseline으로부터 반복적인 연산을 통해 1st order, 혹은 2nd order 경사 하강기법을 사용하여 최적점을 찾거나, 더 세련되게 설계계획법(Design of Experience, DOE)과 근사함수 모델링(Surrogate Modeling)으로 목적 함수를 근사적으로 구현하여 최적점을 찾는다. 형상 파라미터는 단순하게 NACA 시리즈의 파라미터들을 사용할 수 있고, 보다 복잡한 Bezier curve [31]나 CST curve [32]를 사용할 수도 있다. 혹은 x 좌표에 대한 y 값을 사용할 수도 있다. 해석 도구는 빠른 XFOIL을 사용하거나, 느리지만 고속 유동에 대한 해석도 가능하며 정확도도 더 높은 CFD를 사용할 수도 있다. [33] [34]



<그림 2-36> 최적설계 기법을 이용한 에어포일 설계 프로세스

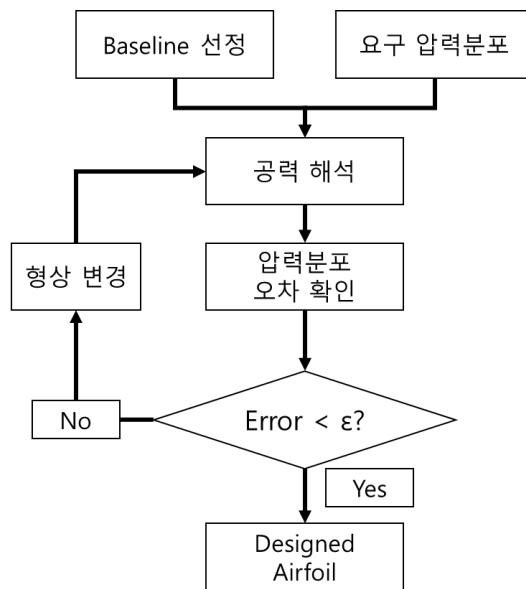
일반적으로 에어포일 최적화는 <그림 2-36>과 같은 프로세스를 통해 이루어지며, 목적함수 생성 기법에 따라 그 방법을 달리 할 수 있다. 그러나 일반적인 최적설계 기법은 다음과 같은 문제들을 갖고 있다.

- 일회성이다.
 - 다른 요구조건에 맞는 에어포일을 설계하기 위해서는 기본형상 설정부터 모든 프로세스를 다시 시작해야 한다.
- 시간이 오래 걸린다.
 - 최적점에 도달할 때까지 형상이 변경될 때마다 공력 해석을 실시해야 한다. 그러나 저정밀도 해석 도구로 사용한다 하여도 수 시간이 걸리며, 고정밀 해석 도구는 수 일이 걸린다.

이러한 문제들을 개선하기 위해 DOE나 근사 모델링 기법들이 많이 사용된다. 일반적으로는 Latin Hyper Cube(LHC)와 같은 DOE를 통해 모델을 생성하기 위한 기본 데이터를 확보하고, Radial Basic Function(RBF)나 Kriging 등을 이용하여 형상 파라미터에 대한 성능 함수를 만든다.

C. 역설계 기법

역설계 기법은 설계하고자 하는 에어포일의 압력 분포와 기준 에어포일이 필요하다. 가장 기초적인 역설계 기법은 기준 에어포일로 평판을 사용하지만, 더 빠르게 찾아내기 위해서 비슷한 압력분포를 갖는 에어포일을 기준 에어포일로 사용하기도 한다. 처음에는 현재 갖고 있는 에어포일의 압력분포와 요구 조건 압력분포의 차이는 클 것이다. 그러므로 형상이 요구조건 성능을 낼 수 있도록 그 모양을 조금씩 바꾸어 나간다. 조금씩 바꾸어 나간다는 것은 최적설계기법과 마찬가지로 에어포일 성능을 해석하고 오차를 분석하는 과정을 반복적으로 실시한다는 것이다. 그러므로 역설계 기법 역시, 형상을 변경하는 횟수와 사용하는 해석 도구에 따라 설계 속도가 달라지게 된다.



<그림 2-37> 에어포일 역설계 과정

역설계 기법의 또 다른 단점은 입력값으로 에어포일 주변의 압력분포를 요구한다는 것이다. 그러나 에어포일의 압력분포는 설계에 실질적으로 사용되는 요구조건이 아니며, 이를 구하기 위해서는 각종 힘과 모멘트 조건들을 압력분포로 전환시키는 추가적인 과정이 필요하다.

제 3 장 심층학습 기반 NACA 4-digit 에어포일 설계

제 1절 NACA 4-digit 에어포일

NACA 에어포일은 현 NASA의 전신인 미국 국가항공자문위원회(NACA)가 체계적으로 개발한 에어포일 형상이다. 당시 많이 사용되던 R.A.F 15나 U.S.A 27과 같은 에어포일은 너무 낮은 레이놀즈 수에서 개발이 되어 실제 크기의 기체를 설계할 때 공력 성능이 정밀하지 못하였다. 또한 어떠한 규칙을 갖고 에어포일 형상을 만든 것이 아니라, 필요에 따라 즉각적으로 만들고 이름을 붙였기 때문에 체계적이지 못했다. NACA는 에어포일을 체계적으로 표준화하기 위해 다양한 레이놀즈 수에서 에어포일에 대한 풍동실험을 실시하였으며, 1935년에 네 자릿수로 표현되는 NACA 4-digit 에어포일을 규격화 하였다. [4]

NACA는 4-digit을 확장한 5-digit, 6-digit 등 다양한 종류의 에어포일 규격을 만들었으나, 본 연구에서는 네 자리로 표현되는 세 개의 형상 파라미터만으로도 간단히 만들 수 있는 NACA 4-digit 에어포일을 사용하였다. NACA 4-digit은 최대 캠버, 최대 캠버 위치, 최대 두께를 형상 파라미터로 사용한다. 예를 들어, NACA 2412는 비대칭형 에어포일로 시위 길이의 2%의 최대 캠버를 갖고, 최대 캠버의 위치는 Leading edge로부터 시위 길이의 40% 떨어진 지점에 있으며, 시위 길이의 12%에 해당하는 최대 두께를 갖는다. 반면, NACA 0010은 대칭형 에어포일로 최대 두께만으로 형상을 만들어낼 수 있으며, 시위 길이의 10%에 해당하는 최대 두께를 갖는다.

[4]에서 NACA는 실험 결과들을 바탕으로 형상과 공력 성능 사이에는 어떠한 관계가 있다는 것을 확인하고, 에어포일의 두 가지 형상 특성을 주목하였다. 하나는 두께이고 다른 하나는 캠버이다. 두께는 구조적인 관점에서 중요하며 캠버는 영양력 받음각과 피치모멘트에 영향을 준다. 이는 보고서의 3페이지에 다음과 같이 기술되어 있다.

“The major shape variables then become two, the thickness form and the mean-line form. The thickness form is of particular importance from a structural standpoint. On the other hand, the form of the mean line determines almost independently some of the most important aerodynamic properties of the airfoil section, e.g., the angle of zero lift and the pitching-moment characteristics.” [4]

Jacobs et. al.는 에어포일의 종류를 대칭형과 비대칭형으로 나누었다. 대칭형은 익형의 최대 두께만을 변형하면서 사용하였고, 비대칭형은 세 가지 형상 특징(최대 캠버, 최대 캠버 위치, 최대 두께)을 이용하였다. 다음은 대칭형 NACA 4-digit의 y좌표를 생성해내는 방법이다. [4] [35]

$$\pm y_t = 5t(0.29690\sqrt{x} - 0.12600x - 0.35160x^2 + 0.28430x^3 - 0.10150x^4) \quad (63)$$

- x: 대칭형 NACA 4-digit 에어포일의 x 좌표
- y_t: 대칭형 NACA 4-digit 에어포일의 y 좌표
- t: NACA 4-digit 에어포일의 최대 두께 값

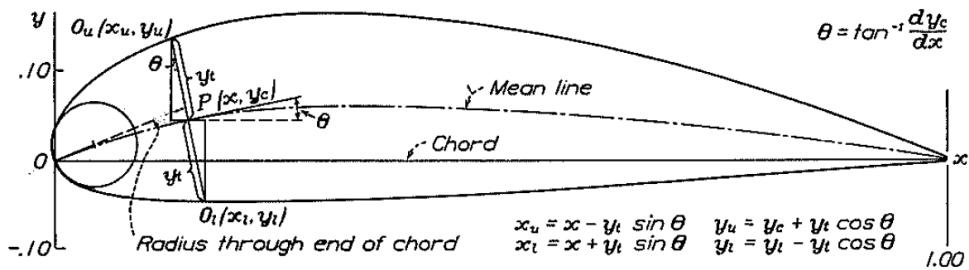
일반적으로 에어포일의 길이는 1로 사용하므로, x는 0과 1 사이의 값을 갖는다. t는 최대 두께를 시위선 길이로 나눈 값을 사용하며, NACA 4-digit의 맨 뒤 두자리를 100으로 나눈 값과 같다. 예를 들어 NACA 0012를 만드는 경우, 최대 두께는 시위선 길이의 12%이며, t=0.12가 된다. 이 때, Leading-edge의 곡률은 다음과 같다.

$$r = \frac{1}{2}(5ta_0)^2 = 1.10187t^2 \quad (63)$$

$$a_0 = 0.29690 \quad (64)$$

비대칭형 NACA 4-digit은 만드는 과정이 더 복잡하다. 비대칭형은 최대 캠버(m)과 최대 캠버 위치(p)가 변수로 추가가 되며, 이들을 이용해 평균 캠버선(Mean line)을 먼저 그려야 한다. 에어포일의 윗면과 아랫면은 이 평균 캠버선으로부터 y_t만큼 수직으로 떨어져 있으며, 평균 캠버선의 기울기

를 구해 이들의 좌표점을 구한다. <그림 3-1>은 [4]에 수록된 NACA 4-digit 형상 예시 그림이며, 수식들은 비대칭형 NACA 4-digit 에어포일의 윗면과 아랫면 좌표를 구하는 과정을 나타낸 것이다.



Sample calculations for derivation of N.A.C.A. 6321

<그림 3-1> 비대칭형 익형 생성 방법 [4]

- 평균 캠버선의 y 좌표

$$y_c = \frac{m}{(1-p)^2} [(1-2p) + 2px - x^2] \quad (65)$$

- 평균 캠버선의 기울기

$$\theta = \tan^{-1} \frac{dy_c}{dx} \quad (66)$$

- 윗면의 x 좌표와 y 좌표

$$x_u = x - y_t \sin \theta \quad (67)$$

$$y_u = y_c + y_t \cos \theta \quad (68)$$

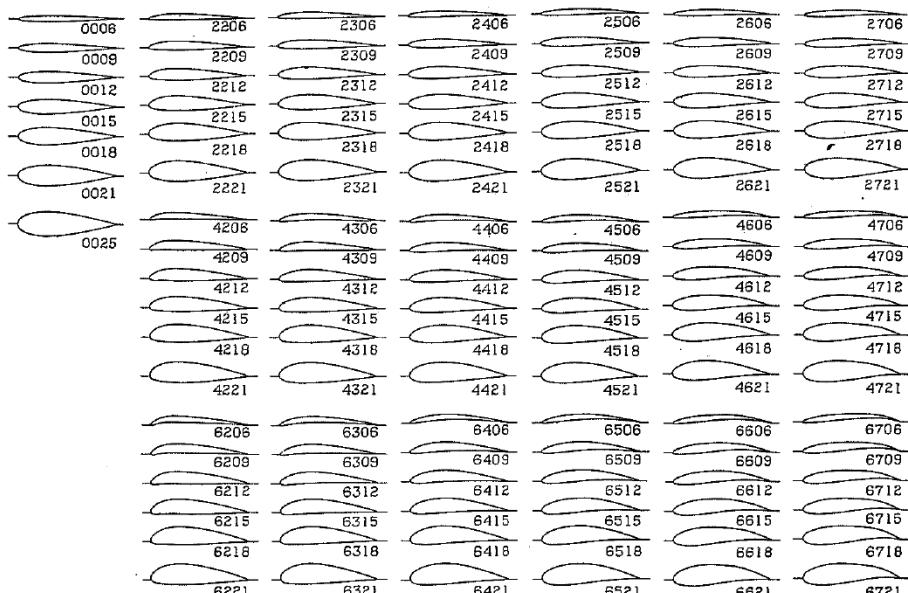
- 아랫면의 x 좌표와 y 좌표

$$x_l = x + y_t \sin \theta \quad (69)$$

$$y_l = y_c - y_t \cos \theta \quad (70)$$

- x : 캠버의 x 좌표
- y_c : 캠버의 y 좌표
- m : NACA 4-digit 의 최대 캠버 값
- p : NACA 4-digit 의 최대 캠버 위치 값
- θ : 캠버의 기울기
- x_u : NACA 4-digit 에어포일의 윗면 x 좌표 값
- y_u : NACA 4-digit 에어포일의 윗면 y 좌표 값

[4]에서 실시한 풍동실험의 유동 조건은 $Re=3,000,000$ 이며, 유동 속도는 약 70ft/s (약 $M=0.06$)로 기술되어 있다. 각 에어포일 모델은 듀랄루민으로 만들어졌으며, 5인치의 코드와 30인치의 스팬을 갖는 날개로 테스트를 진행하였다. <그림 3-2>는 [4]에서 형상화한 NACA 4-digit 에어포일들이며, 이들 중 68 개의 에어포일과 10개의 변형 에어포일을 사용하여 풍동실험을 실시하였다.



<그림 3-2> [4]에서 만든 NACA 4-digit 에어포일 형상 목록

[4]에서는 하나의 에어포일 형상에 대해 25회의 반복적인 실험을 실시하여 풍동실험의 <표 3-1>과 같이 획득하였다.

<표 3-1> [4]의 풍동실험 오차

α	$\pm 0.15^\circ$
$C_{L,max}$	-0.03~0.01
$C_{m,c/4}$	± 0.003
$C_{D,0}(C_L=0)$	-0.0002~0.0006
$C_{D,0}(C_L=1)$	-0.0008~0.0015

또한 지지대로 인해 발생한 오차도 <표 3-2>와 같이 획득하였다.

<표 3-2> [4]의 지지대로 인한 풍동실험 오차

α	$\pm 0.05^\circ$
$C_{L,max}$	-0.02~0.00
$C_{m,c/4}$	± 0.001
$C_{D,0}(C_L=0)$	-0.0002~0.0000
$C_{D,0}(C_L=1)$	± 0.0010

보고서 [4]에는 풍동실험 결과와 이들을 분석한 내용들도 있으며, 보고서는 결론에 다음과 같이 NACA 4-digit의 특성들을 정리하였다.

- 에어포일 두께에 따른 변화
 - 선형구간에서 양력곡선 기울기는 에어포일의 두께가 증가할수록 이론적인 수치인 $2\pi/\text{rad}$ 의 약 81~91% 수준까지 줄어들었다.
 - Angle of zero lift(영양력 받음각)은 에어포일의 두께가 증가할수록 0으로 다가갔다. (9~12% 이상)
 - 최대 양력이 가장 높은 에어포일들은 일반적으로 사용하는 두께의 에어포일에서 찾아볼 수 있었다. (9~15%)
 - 적당히 두껍고 낮은 캡버를 갖는 에어포일들은 최대 양력구간에서 매우 불안정한 유동흐름을 보였다.

- Moment at zero lift(영양력 모멘트)는 두께가 증가할수록 Thin-airfoil 이론으로 얻을 수 있는 값의 87~64% 수준까지 줄어들었다.
- 공력중심은 모멘트가 일정한 부분을 사용하는데, 이론적으로는 시위 길이의 1/4 지점이다. 그러나 두께가 증가할수록 모멘트가 일정한 지점이 앞으로 움직인다.
- 최소 Profile drag 는 다음과 같이 두께에 대한 방정식으로 표현할 수 있다.(k 값은 캠버 크기에 따라 달라진다.)

$$C_{D0,min} = k + 0.0056 + 0.01t + 0.1t^2 \quad (71)$$

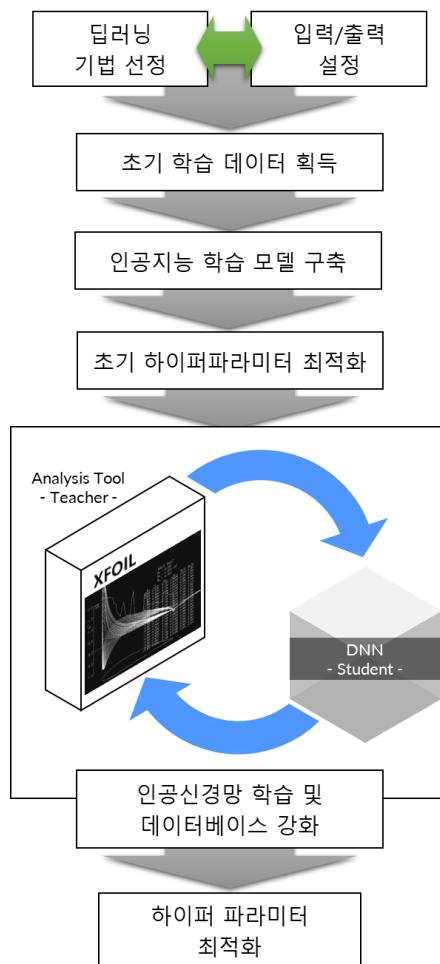
- $C_{D0,min}$ = 최소 형상 항력(Minimum profile drag)
- t : NACA 4-digit 에어포일 최대 두께 값
- 최소 Profile drag 에서의 양력계수(Optimum lift coefficient)는 두께가 증가할수록 0 으로 다가간다.
- $C_{L,max}$ 와 $C_{D0,min}$ 의 비는 중간 정도의 두께에서 가장 크게 나타난다. (9~12 %)
- 에어포일 캠버에 따른 변화
 - 선형구간에서의 양력곡선 기울기는 캠버의 위치가 뒤로 갈수록 약간씩 줄어드는 경향을 보인다.
 - Angle of zero lift 는 Thin-airfoil 이론의 100~75% 사이의 값을 나타내며, 일반적인 캠버 위치($0.3c\sim0.5c$)를 갖는 에어포일 형상일수록 이론값과 차이가 적다.
 - 최대 양력은 캠버 크기가 커질수록 증가하며, 캠버 위치가 시위의 30% 지점($0.3c$)를 기준으로 앞으로 가거나 뒤로 가면 더 급격하게 증가한다.
 - 일반적인 캠버 위치($0.3c\sim0.5c$) 범위 안에서, 캠버가 증가할수록 최대 양력 지점에서 공기의 흐름이 가장 안정적으로 나타난다.

- Moment at zero lift 는 캡버 크기에 비례하는 모습을 보인다.
- 모멘트가 일정한 지점은 캡버의 위치가 뒤로 갈수록 앞으로 움직인다.
- 최소 Profile drag 는 캡버가 커질수록, 캡버의 위치가 뒤로 갈수록 증가한다.
- Optimum lift coefficient 는 캡버가 커질수록 증가하며, 캡버 크기가 매우 커지면, 캡버 위치가 앞으로 갈수록 확연하게 증가한다.
- $C_{L,max}$ 와 $C_{D0,min}$ 의 비는 캡버가 $0.02c$ 이상일 때, 캡버가 커지면 줄어드는 양상을 보인다. 또한 매우 큰 캡버값에서는 캡버 위치가 뒤로 갈수록 줄어든다.

제 2절 딥러닝 기반 NACA 4-digit 에어포일 설계

인공지능 학습 프로세스

본 연구에서는 NACA 4-digit 에어포일을 이용하여 딥러닝 기법을 이용한 에어포일 설계가 가능한지를 확인하였다. 그렇기에 NACA 4-digit을 이용한 예제는 문제 구성을 매우 단순하게 하였다. 딥러닝 기반 프로그램은 일단 학습이 완료가 되면 설계는 신경망 모델을 이용하면 되지만, 바로 그 인공신경망을 학습시키는 것이 가장 큰 문제이다. 신경망을 학습시키기 위해서는 충분한 양의 데이터가 필요하며, 오랜 시간이 걸린다.



<그림 3-3> DLED NACA 4-digit 학습 프로세스

<그림 3-3>은 딥러닝 기반 익형 설계 프로그램을 만들기 위해 본 연구에서 심층 신경망을 학습시킨 과정을 나타낸 것이다. 본 연구에서 핵심으로 작용하는 단계는 인공신경망 학습과 이를 바탕으로 한 데이터베이스 강화 프로세스다. 딥러닝의 핵심은 데이터의 양인데, 앞서 언급한 NACA의 보고서 [4]에도 수록된 데이터의 수는 78개이며, [1]에는 NACA 4-digit 뿐만 아니라 모든 에어포일을 합쳐서 1,636개, [2]에도 1,550개의 데이터만 있을 뿐이다. 설계라는 공학적 영역에서 보면 이는 매우 많은 수의 데이터이겠지만, 딥러닝의 관점에서 보면 이는 매우 적은 것이다.

딥러닝이 가장 많이 쓰이는 이미지 분류 문제의 경우, 수백만장의 이미지를 사용하여 학습을 시킨다. 처음으로 인간의 이미지 분류 점수를 넘어선 [16]의 경우, 2012년 ImageNet [36]에서 제공한 1,000개의 카테고리로 구성된 120만장의 이미지를 사용하였다. 당시 ImageNet에는 10,000개 이상의 카테고리로 구성된 1,000만개의 이미지 데이터가 있었으며, 현재는 21,000개 이상의 카테고리로 구성된 1,400만개 이상의 이미지 데이터가 수록되어 있다. 이들 이미지는 모두 수작업으로 분류가 되어있다.

물론 이미지 분류는 3차원의 텐서 데이터를 이용하기 때문에 에어포일 설계 문제보다 더 많은 데이터를 요구할 수 있다. 그러나, 120만개에 비해 1,600여개는 현저히 낮은 수의 데이터다. 본 문제에 사용되는 NACA 4-digit만 추려낼 경우 그 수는 현저히 줄어들 것이다. 그러므로 본 연구에서는 데이터 부족 문제를 해결하기 위해 인공신경망의 예측 값에 기반하여 데이터베이스를 자동으로 강화하는 루프를 구축하였으며, 이를 통해 예측 오차를 현저히 줄일 수 있었다.

본 섹션에서는 데이터베이스 강화루프 뿐만 아니라, NACA 4-digit 예제를 이용하여 인공신경망을 학습시키고 실제 응용 프로그램으로 어떻게 만들 어내는지를 기술하였다. 기술 순서는 앞서 나타낸 DLED NACA 4-digit 학습 프로세스를 따랐다.

1. 딥러닝 기법 선정과 입/출력 선정

인공신경망을 만들고 학습하기에 앞서, 어떤 기법을 사용할 것인지를 생각해야 한다. 그러나 이는 인공신경망으로 학습시킬 데이터를 선정하는 과정과 맞물려 있기도 하다. 이미지 정보를 다룰 것이라면 Convolution Neural Network(CNN)을 사용해야 하며, 순차 데이터를 사용할 것이라면 Recurrent Neural Network(RNN)이나 Long-Short Term Memory(LSTM) 모델을 사용해야 한다.

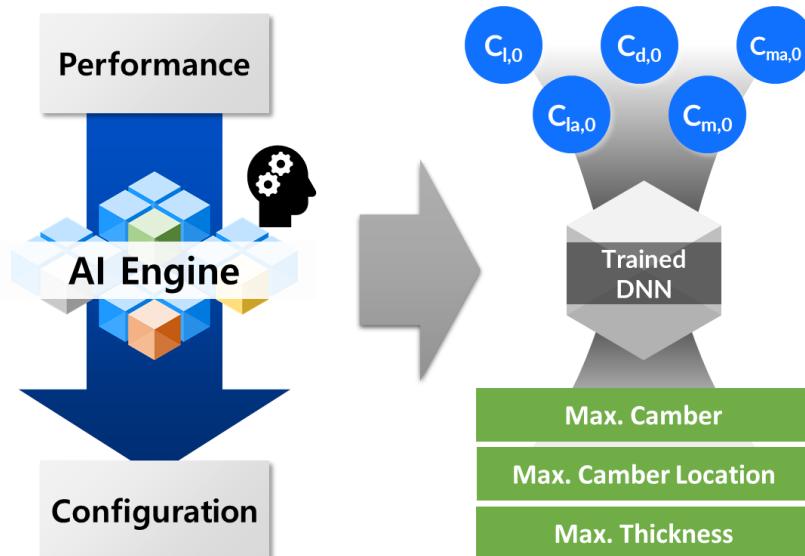
A. 입/출력 선정

본 연구는 성능 지표를 이용하여 형상 파라미터를 도출해내는 인공지능을 구현하는 것을 목표로 하였다. 그러므로 출력 데이터는 당연 형상 파라미터이며, 입력 데이터는 성능 지표가 된다. 우선 입력 데이터로 사용할 수 있는 데이터 종류는 이미지나 단순 수치 데이터를 생각해볼 수 있다. 이미지 데이터를 사용하고자 한다면 성능 지표의 그림, 예를 들어 압력의 분포도나 양력, 항력, 모멘트 곡선을 생각해볼 수 있다. 그러나 설계자가 얼마나 정확하게 그림들을 그려낼지 생각해본다면 이는 적합하지 않다. 그림의 정확도 측면만이 아니라, 설계자의 의도가 잘 반영될지의 여부도 불투명하다.

인공신경망과 성능 지표들을 이용해 형상을 만들고자 했던 이전 연구들은 압력의 분포를 사용하거나 [37], 넓은 받음각 영역에서의 모든 양력, 항력, 모멘트 계수를 사용하였다. [38] 혹은 x좌표와 양력, 항력계수들을 같이 입력값으로 사용하거나 [39], 단순히 특정 받음각에서의 양력과 항력 계수를 사용하기도 하였다. [40] 반면, 출력값으로는 입력값으로 받아들인 x좌표의 y좌표를 찾아내도록 하거나 [39], 지정된 x 좌표의 y값을 결정하도록 [40] 한 연구도 있는 반면, 본 연구와 마찬가지로 형상을 결정짓는 특정 파라미터를 이용한 연구도 있었다. [38] 이들이 입/출력 값을 어떤 이유로 선정하였는지는 불분명하지만, 확실한 것은 설정할 입력값과 출력값이 많으면 사용자의 편의성은 물론이고 인공신경망의 학습에 좋은 영향을

주지 않는다는 것이다.

본 연구는 사용자들이 최소한의, 그리고 실질적인 파라미터를 이용하여 형상을 도출하도록 하였다. 입력값이 너무 많다면, 설계 시 사용자가 입력해야 할 변수들이 매우 많아진다. 하지만 너무 적다면 인공신경망이 적합한 값을 찾아내기 어려워질 것이다. 출력값은 사용자가 결과로서 얻어내는 값이므로, 혹자는 그 수가 적든 많은 사용자에게 영향은 없을 것이라 생각할 수 있다. 그러나 사용자의 편의를 증가시키기 위해 사용하는 것이 인공지능인 만큼, 출력값 역시 간편하게 설정할 필요가 있다. 또한 출력값 수가 많아지면 적합한 학습에 필요한 데이터의 수도 많아지는 경향이 있으므로, 되도록 출력값 수를 최소화하는 것이 학습에도 도움이 된다. 그러므로 출력 데이터 역시 최소한으로 할 필요가 있다. 다음 그림은 DLED NACA 4-digit 알고리즘에서 설정한 입/출력 파라미터를 도식화한 것이다.



<그림 3-4> DLED NACA 4 digit

이번 섹션에서 예시를 보인 NACA 4-digit 에어포일은 단 세개의 파라미터만으로도 에어포일을 만들어낼 수 있어 출력 파라미터 수를 최소화하는데 도움이 된다.

<표 3-3> DLED NACA 4-digit 출력 파라미터

Maximum Camber	Location of Maximum Camber	Maximum Thickness
최대 캠버	최대 캠버 위치	최대 두께

<표 3-4> DLED NACA 4-digit 입력 파라미터 (at AOA = 0 deg.)

Cl,0	Cla,0	Cd,0	Cm,0	Cma,0
양력계수 기울기	양력계수 기울기	항력계수	모멘트계수	모멘트계수 기울기

입력 파라미터로는 상기 표와 같이 선형 구간의 다섯가지 공력 계수들을 선정하였다. [4]에서 언급한 것과 같이, NACA 4-digit 익형은 zero-lift AOA과 모멘트에 영향을 주는데, 이들은 에어포일 선형구간 특성에도 영향을 미친다. 또한, 반음각(AOA) 0도를 고정한 것은 zero-lift AOA를 찾아내는 것 보다 연산 수를 줄일 수 있기 때문이다. Zero-lift AOA를 찾기 위해서는 더 넓은 반음각 범위내의 양력 특성을 모두 계산할 필요가 있다. 만약 정밀성을 10배 높이고자 한다면 연산이 필요한 경우의 수도 10배로 늘어나게 될 것이다. 그러므로 최소한의 연산으로 선형구간의 양력, 모멘트 계수를 계산할 수 있도록, 반음각 0도에서의 양력, 모멘트 계수와 그 기울기를 입력값으로 선정하였다. 항력 계수는 최소 항력계수를 계산하는 것이 가장 바람직하나, 마찬가지로 연산량의 증가를 막기 위해 반음각 0도에서의 항력계수를 사용하였다.

B. 딥러닝 기법 설정

DLED NACA 4-digit에 사용된 입출력 파라미터는 수치 데이터들로, 실수 값들이다. 이런 부류의 입출력 파라미터들은 이미지 처리나 언어처리에 사용되는 Convolution layer나 Recurrent 기법이 필요 없다. 그러므로 본 연구에서는 **심층 신경망(Deep Neural Network, DNN)**를 온전하게 묶은 딥러닝 알고리즘을 사용하였다. 심층 신경망은 인공신경망의 은닉층이 다수인 경

우를 말하며, 3개 이상의 FCNN 은닉층이 있다면 층이 깊다고 할 수 있다. [15]에서 알 수 있듯이, 3개 이하의 층으로 이루어진 얇은 인공신경망은 특별한 기법을 도입하지 않아도 충분한 학습이 가능하다. 하지만 층이 깊어질수록, 인공신경망이 표현력(Representation)은 강해지며, 더 복잡한 문제도 학습할 수 있게 된다.

본 연구에서 사용한 딥러닝 기법과 인공신경망의 입출력은 다음과 같다

- 입력과 출력
 - 입력: Cl,0, Cla,0, Cd,0, Cm,0, Cma,0
 - 출력: Maximum Camber, Location of Maximum Camber, Maximum Thickness
- 딥러닝 기법: Deep Neural Network (DNN, 심층 신경망)

2. 초기 데이터 획득

앞서 DLED NACA 4-digit 설계 인공지능을 구현하기 위해 입력과 출력 파라미터를 정하고, 딥러닝 기법을 개략적으로 선정하였다. 본 연구에서 구현한 설계 인공지능은 성능 파라미터를 이용해 형상 파라미터를 예측하는데, 이와 같은 인공신경망 모델을 학습시키기 위해서는 두 파라미터들로 구성된 수많은 데이터가 필요하다. 역설계 도구나 다른 설계 도구들을 이용해서 데이터베이스를 구성할 수도 있으나, 이들은 데이터를 대규모로 양산해내는 작업에는 적합하지 않다. 설계 도구들을 사용하여 데이터를 만들어낼 경우, 이들이 생성해낸 형상들이 의도한 성능을 제대로 내는지 다시 확인하는 작업이 필요하기 때문이다. 그러므로 추가적인 작업을 줄이기 위해 본 연구에서는 데이터를 구축 시, 형상을 바탕으로 성능을 계산해내는 해석 도구를 이용하였다.

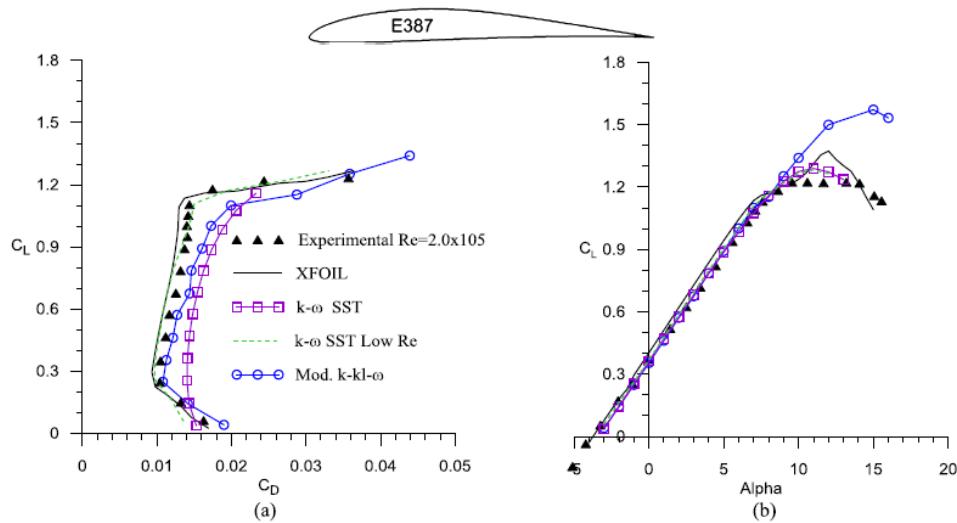
A. 해석 도구 설정

에어포일의 성능 해석 도구는 CFD와 Panel Method, 그리고 Thin-Airfoil Theory에 기반한 프로그램들을 사용할 수 있다. CFD는 해석 도구들 중 가장 정밀한 결과값을 계산해낼 수 있다. 그러나 CFD는 해석 전에 실시하

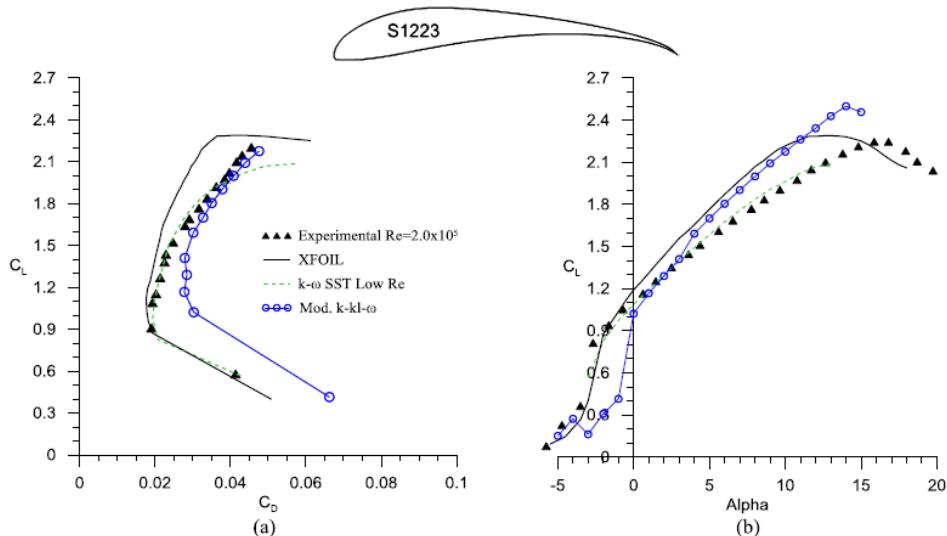
게 되는 메쉬를 입히는 작업이 매우 까다롭고, 해석 시간이 오래 걸린다. 해석 시간은 컴퓨터의 성능과 요구 정밀도에 따라 다르겠지만, 일반적으로 하나의 익형을 한 받음각에서 해석할 경우, 수 분에서 한시간정도 소요된다. Thin-Airfoil Theory는 에어포일이 두께가 없고 캠버만 있다고 가정하는 방법으로, 해석 속도가 빠르다는 장점이 있지만, 두꺼운 에어포일 정확도는 떨어지며, 항력을 예측하지 못한다. Panel method는 2차원 에어포일을 작은 패널들로 나누어, 각 패널에서의 압력 분포를 찾아낸다. Panel method는 에어포일의 모양을 그대로 사용하기 때문에, Thin-Airfoil Theory보다 더 정확한 결과를 얻을 수 있으며 압력 항력 또한 계산해낼 수 있다. 본 연구에서는 빠른 연산을 위해 CFD 대신 XFOIL [5]을 사용하였다. XFOIL은 MIT의 Drela 교수가 제작한 아음속 에어포일 해석도구로, Panel method와 Boundary layer theory를 결합하여 비점성/점성 유동에 의한 공력 계수를 계산해낼 수 있다. XFOIL을 이용하면, 제한된 연산 성능의 환경에서도 멀티스레드를 이용한 병렬 연산이 가능하다. CFD를 사용하면 더 정확한 결과를 얻어낼 수 있지만, 병렬 연산을 위해서는 서버 수준의 연산 장비가 필요하며, 해석 전에 Mesh를 잘 만들어주어야 한다. 또한 충분히 수렴된 값을 얻기 위해서는 XFOIL보다 많은 시간이 필요하다.

1) XFOIL의 검증

XFOIL은 개발된 지 오래되었고 많은 사람들이 사용하여 버전의 업데이트가 상당히 오래 지속되어왔을 뿐만 아니라, 검증 또한 많이 이루어졌다. [41]은 2차원 에어포일의 XFOIL과 CFD, 그리고 실험 데이터 공력 값을 비교하였다. 이 때, Reynolds 수는 20만과 46만을 사용하였다. 에어포일은 E387과 S1223을 사용하였는데, E387은 일반적인 형상의 에어포일이며, S1223은 캠버가 크고 최대 캠버의 위치가 에어포일 앞쪽에 있는 특별한 형상의 에어포일이다. E387의 경우 XFOIL은 CFD 기법과 실험 결과에 매우 유사한 결과를 보여준다. 하지만 S1223에서 XFOIL은 k-w SST 기법에 비해 실험 데이터에 대한 오차가 크게 나타난 것을 볼 수 있다.



<그림 3-5> 성능 해석 기법에 따른 E387 에어포일의 공력계수 [41]



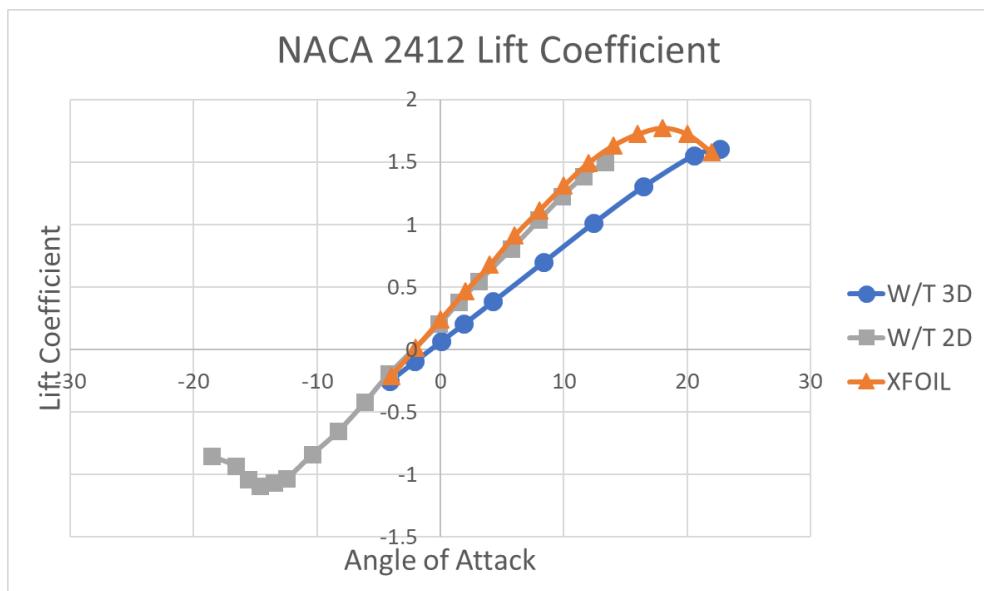
<그림 3-6> 성능 해석 기법에 따른 S1223 에어포일의 공력 계수 [41]

그러나 <그림 3-6>에서 볼 수 있듯이, XFOIL은 최대 양력계수의 값을 실험값과 유사하게 잡아냈으며, 다만 최대 양력의 받음각을 낮게 측정했을 뿐이다. 항력 역시 좌측의 양력-항력 곡선을 보면 실험값보다 더 낮게 측정하는 것을 확인할 수 있다.

본 연구에서 참고한 [4]에는 다양한 형상의 NACA 4-digit 에어포일의 풍

동실험 결과가 수록되어 있다. 그러나 풍동실험의 결과로 XFOIL 검증을 실시하지 않은 이유는 2D 에어포일과 3D 날개는 공기역학적 특성에 차이가 있기 때문이다. 풍동실험에서 사용하는 3D 형상 날개의 경우, 익단에서 Downwash 현상이 일어나, 양력이 전반적으로 낮아지고, 덩달아 양력곡선의 기울기도 낮춘다. [42]

본 연구에서는 2D 에어포일에 대한 검증으로 [43]에 수록된 실험 데이터를 사용하였으며, 그 결과는 <그림 3-7>과 같다.



<그림 3-7> NACA 2412의 2D, 3D 실험값과 XFOIL 해석 값

<그림 3-7>에서 3D 날개의 풍동실험 결과는 양력 계수와 그 기울기가 다른 두 경우보다 낮은 것을 확인할 수 있다. 그러나 2D 풍동실험 결과와 XFOIL 해석 결과는 매우 유사한 것을 확인할 수 있다.

2) XFOIL 해석 조건

해석 도구로 어떤 형상의 성능을 측정한다면 해석 조건을 어떻게 설정하였는지가 중요하다. 본 연구의 NACA 4-digit 예시에서는 <표 3-5>와 같은 XFOIL 해석 조건을 사용하였다. Reynolds number는 [4]에서 풍동실험에 사용한 값들에 기반하여 설정하였다.

<표 3-5> DLED NACA 4-digit 알고리즘의 XFOIL 해석 조건

Number of Points	Upper	99
	Lower	99
	Flow Property	Viscous
	Iteration of Viscous-Solution	200
	Reynolds Number	3,000,000
	Mach Number	0.2

B. 공력계수 계산 방법

DLED NACA 4-digit은 입력 데이터로 다섯가지 기본 공력계수들을 사용한다. 그러나 받음각 0° 에서만 이들을 계산한다면 종종 결과값이 수렴하지 않거나, 아예 발산해버리는 경우도 생긴다. 하지만 실제 받음각 0° 에서 이런 일은 일어나지 않으며, 주변 받음각에서의 공력 계수와 유사한 값을 선형적으로 보인다. 이런 선형적인 특성을 이용하면, 받음각 -1° 와 1° 의 값으로 0° 에서의 공력 계수를 계산할 수 있다.

```

XFOIL          Version 6.99
Calculated polar for: Predicted NACA
1 1 Reynolds number fixed           Mach number fixed
xtrf =    1.000 (top)      1.000 (bottom)
Mach =    0.100      Re =    1.000 e 6      Ncrit =    9.000
alpha     CL       CD      CDp      CM   Top_Xtr   Bot_Xtr
-----  -----
-1.000    0.1360    0.00600    0.00121   -0.0536    0.7312    0.4739
  0.000    0.2434    0.00559    0.00122   -0.0524    0.6618    0.6853
  1.000    0.3479    0.00544    0.00140   -0.0501    0.5936    0.8655

```

<그림 3-8> XFOIL 해석 결과 파일

<그림 3-8>은 XFOIL의 해석 결과 파일이다. 해석 파일은 받음각(alpha), 양력계수(CL), 항력계수(CD), 압력 항력계수(CDp), 모멘트 계수(CM), 윗면 천이지점(Top_Xtr), 아랫면 천이지점(Bot_Xtr)을 저장하게 된다.

XFOIL로 모든 받음각에서 공력계수가 산출된다면 받음각 0° 에서의 공력계수와 공력계수 기울기를 구하는 것은 매우 쉽다. 하지만 어떤 받음각에서 해석이 불가능하여 모든 경우를 다 구하지 못하는 경우도 있으며, 특히 0° 에서 값을 얻어내지 못한다면 대안을 찾아야 한다.

<표 3-6> XFOIL 해석 경우의 수

Case 1	$-1^\circ, 0^\circ, 1^\circ$	Case 3	$0^\circ, 1^\circ$
Case 2	$-1^\circ, 0^\circ$	Case 4	$-1^\circ, 1^\circ$

본 연구에서는 공력 데이터를 획득할 수 있는 경우를 <표 3-6>과 같이 네 가지 경우로 나누었다. 모든 받음각에 대한 해석이 되었거나(Case 1), 받음각 0° 와 다른 한 받음각이 있다면 파라미터를 구하기 쉽다. 그러나 받음각 0° 에서 해석이 안되는 경우도 많기에 (Case 4)와 같은 경우도 고려해야 할 필요가 있다. 해석이 전혀 안되거나 한 받음각에 대한 값만 도출한 경우는 모두 제외하였다. 아래 수식들은 <표 3-6>의 네 가지 경우에 대해 필요한 다섯가지 공력 파라미터들을 구하는 방법이다.

$$C_{l,0} = \frac{(C_{l,+1} - C_{l,-1})}{(\alpha_{+1} - \alpha_{-1})}(x - 1) + C_{l,-1} = \frac{(C_{l,+1} - C_{l,-1})}{(\alpha_{+1} - \alpha_{-1})}(0 - 1) + C_{l,-1} \quad (72)$$

$$C_{la,0} = \frac{(C_{l,+1} - C_{l,-1})}{(\alpha_{+1} - \alpha_{-1})} \quad (73)$$

$$C_{d,0} = \frac{(C_{d,+1} - C_{d,-1})}{(\alpha_{+1} - \alpha_{-1})}(x - 1) + C_{d,-1} = \frac{(C_{d,+1} - C_{d,-1})}{(\alpha_{+1} - \alpha_{-1})}(0 - 1) + C_{d,-1} \quad (74)$$

$$C_{m,0} = \frac{(C_{m,+1} - C_{m,-1})}{(\alpha_{+1} - \alpha_{-1})}(x - 1) + C_{m,-1} = \frac{(C_{m,+1} - C_{m,-1})}{(\alpha_{+1} - \alpha_{-1})}(0 - 1) + C_{m,-1} \quad (75)$$

$$C_{ma,x} = \frac{(C_{m,+1} - C_{m,-1})}{(\alpha_{+1} - \alpha_{-1})} \quad (76)$$

C. 초기 에어포일 데이터 확보 범위 설정 및 수집

본 연구에서 초기 에어포일 데이터는 NACA 4-digit 형상 파라미터의 범위를 설정한 후, 에어포일 공력 해석 도구 XFOIL을 이용하여 구축하였다. 초기 데이터의 형상 파라미터는 정수형으로 다음과 같이 설정하였다.

<표 3-7> 초기 데이터 설정 범위

Shape Parameter	Min.	Max.	Step Size
Maximum Camber	0	9	1
Location of Maximum Camber	0	9	1
Maximum Thickness	1	20	1

위 표에 따라 생성할 수 있는 에어포일은 총 2,000개지만, 실제 해석에서는 해석 데이터가 없거나 한 받음각에서만 해석이 가능하여 데이터베이스에 수록되지 못하는 경우를 제외하여 1,496개의 데이터를 확보하였다.

1) 멀티 스레드 해석

XFOIL은 텍스트 기반의 콘솔 프로그램이며, 스레드를 하나만 사용한다. 2,000개의 에어포일 해석은 단일 스레드에서 해석해도 충분할지 모르나, 추후 데이터베이스 강화로 인해 해석해야 할 에어포일 수가 증가하게 되면 해석에 너무 많은 시간이 소모된다. 그러므로 본 연구에서는 다중 스레드를 이용해, 한번에 여러 개의 XFOIL 프로그램을 사용하여 데이터를 축적하였다.

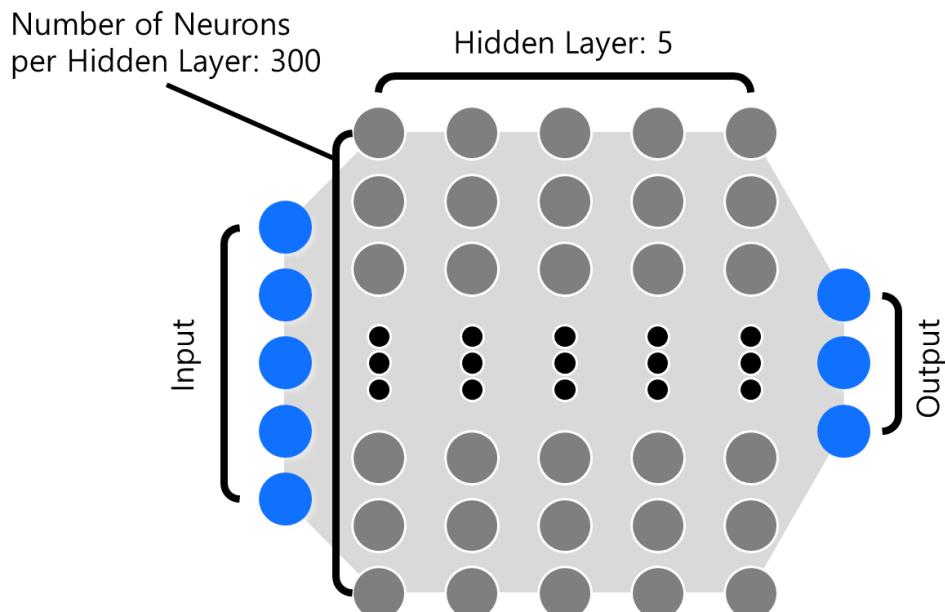
본 연구에서 사용한 프로그래밍 언어 Python [10]은 다중 스레드를 이용한 응용프로그램 실행이 가능하다. 연구에 사용한 장비로 가용할 수 있는 스레드는 12개이나, 전부 사용할 경우 컴퓨터가 멈추는 현상이 발생하므로, 10개의 프로세서만 사용하였다.

D. Troubleshooting

Panel method를 이용하여 해석을 할 경우, Panel을 신중하게 만들어야 한다. 일반적으로 에어포일에서 해석이 어려운 부분은 Leading edge와 Trailing edge 부분이기에 양 끝단에 Panel을 조밀하게 만든다. 반면, 중간부분은 상대적으로 해석이 쉽기 때문에 양 끝단보다 크게 만들어 해석 부하를 줄인다. 다른 주의할 점은 Panel의 크기를 연속적으로 만들어야 한다는 것이다. 본 연구 초기에 에어포일 중간(50% of Chord)에서 Panel 크기가 연속적이지 못하였는데, 이는 XFOIL 해석 시간을 두 배 가까이 증폭시켰다. 본 논문에서 사용된 알고리즘은 해당 문제가 해결되었다.

3. 인공지능 학습 모델 구축

학습을 위한 데이터가 모였다면 학습을 시킬 인공신경망 모델을 구성해야 한다. 본 연구에서는 심층신경망을 사용하였으므로, 다른 딥러닝 모델에 비해 간단한 구조의 신경망을 구축할 수 있다. DLED NACA 4-digit에서 초기 학습을 위해 구현한 인공신경망 구조는 다음과 같다.



<그림 3-9> 초기 학습을 위한 DLED NACA 4-digit 인공신경망 구조
(실제로는 Fully Connected 구조이다.)

<표 3-8> 초기 학습을 위한 인공신경망 모델 구조

입력 뉴런 수	Input Neuron	5
출력 뉴런 수	Output Neuron	3
은닉층 수	Hidden Layer	5
은닉층 당 뉴런 수	Neuron per Hidden Layer	300
파라미터 초기화 기법	Parameter Initialization Method	Xavier Initializer [15] [17]
정규화	Normalization	Input Normalization Layer Normalization [44]
최적화 기법	Optimizer	Adam [28]
학습률	Learning Rate	0.001
활성화 함수	Activation Function	ReLU
미니배치 크기	Mini-batch Size	5,000
부동소수점	Float Precision	FP32
라이브러리	Library	Tensorflow [45]
오차 함수	Error Function	MSE [22]

일반적으로 기계학습은 전체 데이터의 무작위로 섞은 뒤, 일부만 사용하여 학습을 실시하고 나머지는 테스트를 하는데 사용한다. 이는 모델 학습이 지나치게 되어 과적합 문제(Overfitting problem)에 빠지는지를 확인하는데 요긴하다. 본 연구에서는 75%의 데이터를 학습 데이터로, 나머지 25%는 테스트 데이터로 사용했다.

DLED NACA 4-digit 알고리즘은 입력값을 정규화(Normalization, Min-Max Scaling) 하였는데, 인공신경망의 입력값 범위가 다르면 학습이 어렵기 때문이다. 특히 양력계수와 항력계수는 소수점 자릿수가 다를 정도로 데이

터 분포 범위가 다르다. 그러므로 본 알고리즘에서는 성능 파라미터들이 입력층에 들어가기 전에 정규화 되도록 하였으며, 이를 통해 학습을 가속하고 전역 최적값에 더 잘 도달하도록 하였다.

Layer Normalization [44]은 각 층에서 일어나는 정규화 과정으로, 전 은닉 층의 출력값을 정규화하고, 이를 다음 은닉층에서 받아들인다. Layer Normalization 역시 학습을 가속시킨다는 장점이 있다.

부동소수점은 FP 32를 사용하였다. 부동 소수점은 연산을 소수점 몇 자리까지 하는지를 결정한다. 작은 부동소수점을 사용하면 연산은 빨라지나 정확도가 줄어들고, 큰 부동소수점을 사용하면 연산은 느리나 정확도가 늘어난다. 본 연구는 실수형 데이터를 다루는 만큼 정확성이 요구되지만, FP 64를 사용하여도 연산 시간만 증가할 뿐 정확도가 크게 나아지지 않아 FP 32를 사용하였다.

오차 함수로는 인공신경망 학습에서 보편적으로 사용하는 MSE [22]를 사용하였으며, 이는 오차 제곱들의 평균을 의미한다.

A. 개발 환경

본 연구는 Python 3 [10] 기반으로 진행되었으며, Anaconda [46] 플랫폼 상에서 Spyder 통합 개발환경(IDE)을 사용하였다. 기계학습 라이브러리로는 Tensorflow [45]를 사용하였으며, Scikit-learn [47]이 일부 사용되었다.

<표 3-9> 개발 환경

Platform	Anaconda
IDE	Spyder
Machine Learning	Tensorflow
Library	scikit-learn

인공신경망의 학습은 NACA 4-digit과 CST Curve [32] 연구를 동시에 진행하기 위해, 다음과 같은 두 대의 장비를 사용하였다. 본 논문에서는 Training Device 1을 사용하였다면 TD 1, Training Device 2를 사용하였다면 TD 2로 명시하였다. 세부 제원은 <표 3-10>과 <표 3-11>과 같다.

인공신경망을 GPU로 학습시키기 위해서는 CUDA [48] [49] Core를 사용하는 것이 간단하다. CUDA는 Compute Unified Device Architecture의 약자로, NVIDIA에서 GPU를 범용 계산 문제에도 사용할 수 있도록 개발한 플랫폼이다. GPU를 풀어 쓴 Graphical Processing Unit에서도 알 수 있듯이 GPU의 본 목적은 컴퓨터에서 그래픽 연산을 담당하여 가속화하기 위한 장비였다. 그러나 GPU를 이용하면 대규모 병렬 연산에도 효과적이고 효율적으로 사용할 수 있다는 것이 알려지면서, 범용 연산 목적의 GPGPU 기술이 주목받게 되었다.

<표 3-10> Training Device 1 (TD1) Specification

CPU	Ryzen 1600	6 Core 12 Thread 3.2 to 3.6 GHz
RAM	16 GB	8 GB 2666 Mhz (x 2)
SSD	256GB M.2.	System
	256GB SATA 3	DLED Script
GPU	NVIDIA GTX 1070 ti	2,432 CUDA core 1.607 to 1.683 GHZ 8 GB GDDR5

<표 3-11> Training Device 2 (TD2) Specification

CPU	Ryzen 2600	6 Core 12 Thread 3.4 GHz to 3.9 GHz
RAM	16 GB 3200 Mhz	8 GB 2666 Mhz (x 2)
SSD	256GB M.2.	System
	256GB SATA 3	DLED Script
GPU	NVIDIA GTX 1080	2,560 CUDA core 1.632 to 1.771 GHZ 8 GB GDDR5

GPGPU 시장에 가장 적극적으로 뛰어든 NVIDIA는 2006년 GPGPU를 위

한 플랫폼 CUDA를 발표하였고, 지금까지 GPGPU 시장에 적극적인 행보를 보이며 대중적인 GPGPU 제조사가 되었다. 현재 대부분의 GPU 기반의 인공지능 플랫폼은 NVIDIA의 CUDA에 기반하고 있다. AMD 역시 GPGPU 시장에 뒤늦게 뛰어들었지만, 이미 많은 수의 라이브러리가 CUDA 기반에 작성되었기에 전문적으로 AMD의 GPGPU 기반 프로그램이 가능한 수준이 아닌 이상 NVIDIA의 CUDA 기반의 라이브러리들을 사용하는 것이 편리하다. 본 연구에서는 인공신경망 라이브러리로 Tensorflow를 사용하였으므로, CUDA 10.0 [50]과, CUDA의 딥러닝 라이브러리인 cuDNN 7.5 [51]를 사용하였다.

<표 3-12> DLED에 사용된 GPGPU 환경

CUDA	10.0
cuDNN	7.5

B. CPU vs GPU

최근 딥러닝이 떠오르게 된 배경에는 수많은 연구진들의 지속적인 연구로 학습 기법이 발전한 것도 있지만, 컴퓨터 성능의 비약적으로 증가한 것도 큰 부분을 차지한다. 공정의 미세화 한계로 인해 무어의 법칙이 더 이상 힘을 쓰지 못하면서 단일 프로세서를 통한 성능 향상에는 한계가 드러났고, 대규모 행렬 연산에서 CPU는 효율적인 모습을 보이지 못했다.

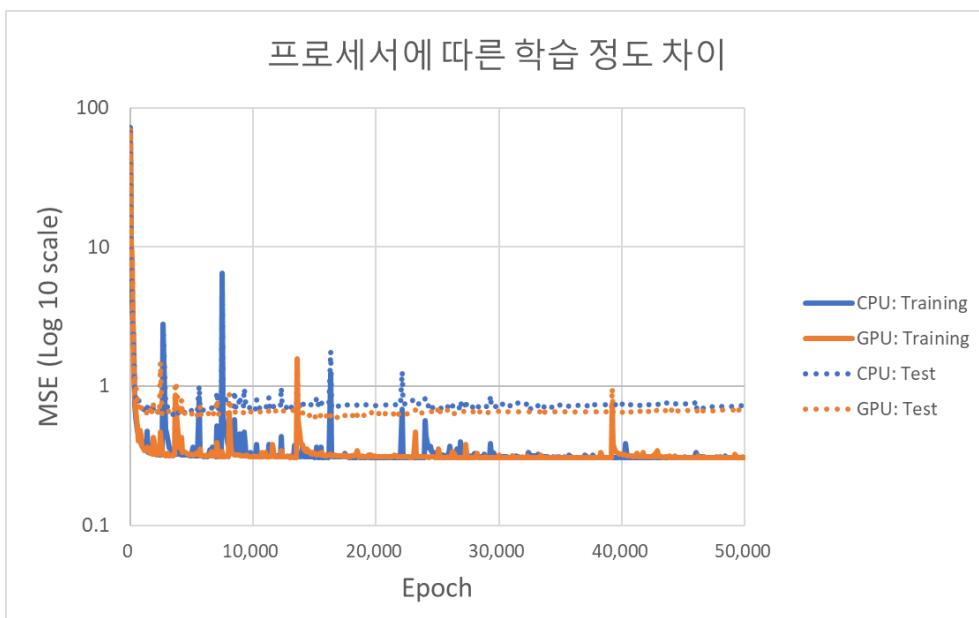
인공신경망이 깊어지고 넓어질수록 더 큰 행렬 연산이 필요하기 때문에, 순차적인 연산에 특화된 CPU보다 태초부터 병렬연산에 특화된 GPU가 더 유리하다. 범용으로 쓰이는 부류의 CPU와 GPU를 비교해보면, CPU는 칩셋당 코어의 수가 100 개를 넘지 못하는 수준이지만, GPU는 수천개가 넘는 연산 코어를 가지고 있다.

<표 3-13>은 본격적인 학습에 앞서, 미리 구축한 초기 에어포일 데이터를 이용하여 CPU 연산과 GPU 연산의 속도 차이를 측정한 것이며, <그림 3-10>은 프로세서에 따른 학습 양상을 비교한 것이다. 학습에는 에어포

일 데이터의 75%를 사용하여 총 1,496개 중 1,122개가 학습에 사용되었고, 나머지 25%의 데이터는 매 학습마다 학습의 정도를 측정하기 위해 사용되었다. <그림 3-10>을 보면 학습 정도에는 큰 차이가 없으나, 학습 속도는 <표 3-13>에서 확인할 수 있듯이, GPU를 이용한 학습이 약 8.5배 빠른 것을 알 수 있다. 이후 본 연구에서 실시한 인공신경망 학습은 모두 GPU를 사용하여 실시하였다.

<표 3-13> 프로세서에 따른 학습 속도 차이

Training Device	Processor		Number of Training Data	Training Time
TD 1	CPU	Ryzen 1600	1,122	63.83 min (1.06 hr)
TD 2	GPU	GTX 1080		7.52 min



<그림 3-10> 프로세서에 따른 학습 정도 차이

C. 학습 데이터의 유동 조건에 따른 차이

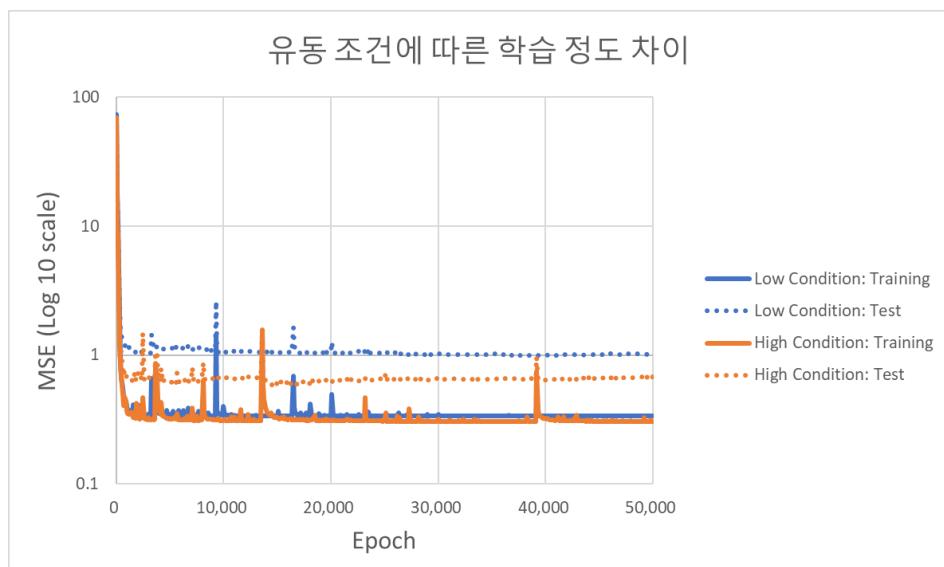
이전 연구들 [52]~[53]에서는 본 논문보다 낮은 유동조건들($Re=1,000,000$, $M=0.1$)을 사용하였다. 그러나 두 유동조건의 수치를 높이면 학습 결과가 개선되는 결과를 확인할 수 있었으며, 더 많은 데이터를 확보할 수 있었다.<표 3-14>~<표 3-16>, <그림 3-11>은 유동조건과 유동 조건에 따른 초기 에어포일 데이터 수, 그리고 학습 결과를 정리한 것이다. <그림 3-11>을 보면 학습 오차는 크게 차이 나지 않지만, 테스트 오차는 크게 줄어든 것을 볼 수 있다.

<표 3-14> 유동 조건

	Low Condition	High Condition
Reynolds Number	1,000,000	3,000,000
Mach Number	0.1	0.2

<표 3-15> 유동 조건에 따른 초기 에어포일 데이터 수

Low Condition	High Condition
1,403	1,496



<그림 3-11> 유동 조건에 따른 학습 정도 차이

<표 3-16> 유동 조건에 따른 학습 결과 차이

Low Condition		High Condition	
Final Training MSE	Final Test MSE	Final Training MSE	Final Test MSE
0.33592	1.02345	0.30624	0.67602

D. DLED: NACA 4-digit 학습 프로세스 입력 파일

본 연구에서 사용되는 학습 프로세스는 모델 구조를 유동적으로 바꾸면서 학습시킬 뿐만 아니라, 학습을 통해 데이터베이스를 강화하는 과정도 포함되어 있다. 그러므로 이를 일관되게 관리하기 위해서는 입력파일을 이용하여 모델을 동적으로 만들면서 통합적으로 관리할 필요가 있다.

```

input.txt
1 #0 Notice
2 # All data should be distinguished by tab #
3 # $is for the information of this file #
4 # For example, To set Max iteration to 2 #
5 # Write Max loop:(\tab)2
6 $
7
8 #1 Training info #
9 Start_loop: 1
10 Max_loop: 20
11 Max_epoch: 20000
12 mini_batch_size: 5000
13 Initial_DB: 0
14 re_Analysis: 1
15 $
16
17 #2 Hyperparameter of ANN #
18 No_Hidden_Layer: 6
19 No_Neuron: 500
20
21 No_input: 5
22 No_output: 3
23
24 Learning_rate: 0.001
25 Activation: ReLU
26 Float_Precision: FP32
27 $

Normal text file length : 319 lines : 14 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS .
```

```

Xfoil setting.txt
1 #0 Notice
2 # All data should be distinguished by tab #
3 # $is for the information of this file #
4 # For example, To set Max iteration to 2 #
5 # Write Max iter:(\tab)2
6 $
7
8 #1 Xfoil setting #
9 No_point: 100
10 Iter: 200
11 Re: 3000000
12 Mach: 0.2
13 timelim: 30
14 $

Windows (CR LF) UTF-8 INS .
```

<그림 3-12> DLED NACA 4-digit 학습 프로세스 입력파일

(좌: input.txt, 우: Xfoil setting.txt)

<그림 3-12>는 DLED NACA 4-digit 학습 프로세스에 사용되는 입력파일이다. 좌측 파일은 학습에 필요한 정보들을 담고 있고, 우측의 파일은 XFOIL 해석 옵션 정보를 담고 있다. 학습 정보 입력 파일(input.txt)은 #1

Training info와 #2 Hyperparameter of ANN으로 구분되어 있다. Training info의 값들 역시 하이퍼 파라미터이나, 학습에 직접적인 영향을 미치는 요소들을 따로 구분하였다. 사용자는 해당 파일의 각종 파라미터들을 여기서 간단히 조절할 수 있다. XFOIL settint.txt 파일은 XFOIL의 유동 조건과 해석 대상인 에어포일의 좌표점 수, 그리고 해석의 반복 계산 횟수 등을 조절 할 수 있다. 마지막의 timelim은 XFOIL의 해석 한계 시간을 정한다.

4. 초기 하이퍼 파라미터 최적화

(Initial Hyper-parameter Optimization, IHO)

인공신경망은 가중치나 바이어스와 같이 신경망 모델을 만드는 파라미터들 외에도, 어떻게 학습을 할 것인지를 정하는 파라미터들이 존재하는데, 이를 하이퍼 파라미터라 하며, 대표적으로는 다음과 같은 것들이 있다.

- Learning rate(학습률)
- Cost function
- Mini-batch size
- Activation function
- Epoch
- Number of hidden layers
- Number of neurons per hidden layer
- Float precision
- Initializer

하이퍼 파라미터들은 모델의 모양과 학습 양상을 결정하므로 매우 중요한 요소들이다. 최적의 하이퍼 파라미터를 찾아내면 모델은 더 나은 예측 성능을 보인다. 그러나 기계학습으로 학습이 되는 요소들이 아니므로, 사용자가 지식과 경험으로 지정해야 한다.

본 연구에서는 반복적인 학습을 통해 학습 데이터를 강화하는 프로세스

를 포함하고 있기 때문에, 그 전에 하이퍼 파라미터를 간단히 최적화하여 앞으로 진행될 반복적인 학습에 대비했다. 하이퍼 파라미터 최적화는 최종 데이터베이스를 이용하여 학습할 때에도 실시하였기에 본 논문에서는 이와 구분하여 초기 하이퍼 파라미터 최적화(Initial Hyper-parameter Optimization, IHO)라 구분하였다.

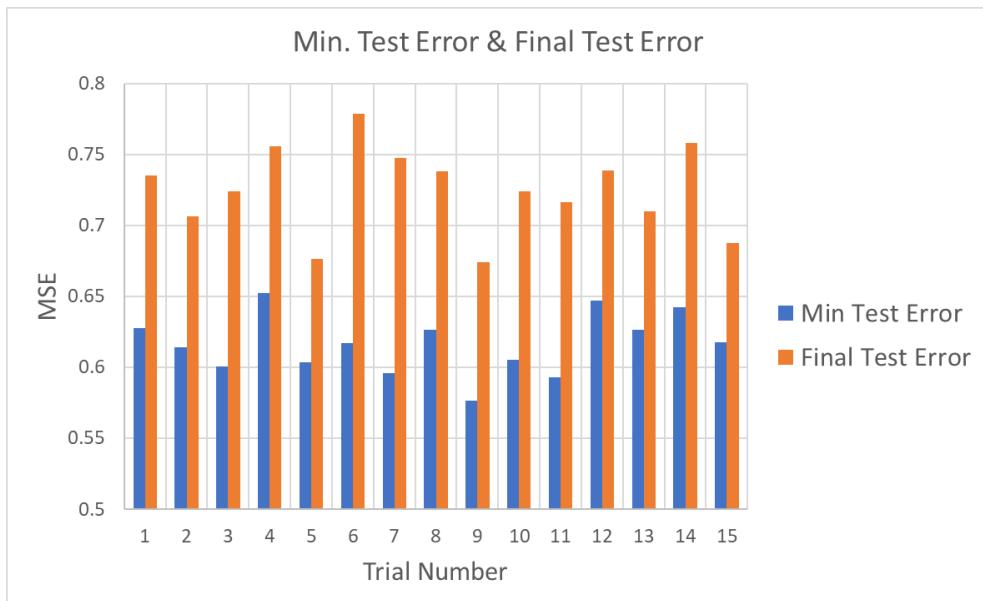
본 연구에서는 초기 하이퍼 파라미터 최적화에 앞서, 기본 하이퍼 파라미터들로 학습을 여러 번 반복하여 학습의 편차를 계산하였다. 인공신경망의 하이퍼 파라미터들 중, 초기화 기법은 무작위 추출을 통해 이루어지며, 또한 학습이 경사 하강법을 이용하기 때문에 매번 같은 학습 양상이 나오지 않기 때문이다. 본 논문에서는 총 15번 학습을 반복하여 편차를 측정하였으며, 초기 하이퍼 파라미터 최적화는 은닉층 수(Number of hidden layers)와 은닉층 당 뉴런 수(Number of neurons per hidden layer)에 대해 실시하였다. 각 비교에서 다른 하이퍼 파라미터들은 <표 3-8>의 값을 사용하였다.

A. 동일 조건에서 반복 학습을 통한 학습 편차 확인

인공신경망의 학습은 할 때마다 다른 양상을 보이는데, 이는 인공신경망의 파라미터 초기화를 무작위 선출을 통해 실시하며, 오차를 줄이는 방법으로 경사 하강법을 사용하기 때문이다. 가중치와 바이어스를 무작위로 선출하여 인공신경망을 초기화하면, 신경망 오차함수 지도에서 최적값을 찾는 시작점은 학습 시행마다 그 위치가 달라지게 된다. 동일한 지도에서 오차 최저점을 찾기 위한 시작점이 달라지면 최적점으로 가는 방법도 달라지며 이로 인해 학습의 양상이 달라지게 되는 것이다. 인공신경망을 무작위로 초기화 하는 이유는 깊은 인공신경망을 잘 학습시키면서 학습 속도를 가속시키기 위한 것으로, [16]와 [20]에 잘 설명되어 있다.

이번 단계에서 진행하게 되는 최적의 하이퍼 파라미터 선정에서는 각 Test Case에 대해 단 한번의 학습만 진행하였다. 그러나 학습의 양상은 매 학습마다 다르기 때문에, 한 번의 학습으로 최적의 값을 결정하는 것은

적절하지 못하다. 한 번의 Test Case 학습에서 우연치 않게 매우 나쁜 값이 나온 것일 수도 있으며, 운이 좋게 매우 좋은 값이 나온 것일 수도 있기 때문이다. 그러므로 본 연구에서는 최적 파라미터를 선정하기 위한 오차의 범위를 정하기 위해, 동일한 하이퍼 파라미터와 데이터를 사용하여 15회 반복적으로 학습을 시키고 그 편차를 알아보았다. <그림 3-13>과 <표 3-17>은 최소 테스트 오차와 최소 테스트 오차의 Epoch, 그리고 최종 테스트 오차를 비교한 그래프와 그 수치를 나타낸 표다.



<그림 3-13> 15회 학습 Test Error 및 최소 Test Error Epoch 결과

<표 3-17> 15회 학습 Test Error 및 최소 Test Error Epoch 결과

Trial	Min. Test Error	Epoch of Min. Test Error	Final Test Error
1	0.62743	8,500	0.73494
2	0.61437	4,400	0.70613
3	0.60043	10,700	0.72393
4	0.65216	2,400	0.75554
5	0.60375	12,800	0.6762
6	0.617	11,400	0.77884
7	0.59584	2,500	0.74755

8	0.62647	5,900	0.73781
9	0.57653	2,900	0.67414
10	0.60538	5,200	0.72379
11	0.59297	16,000	0.71643
12	0.64727	5,100	0.73859
13	0.62635	2,500	0.71003
14	0.64214	9,500	0.75819
15	0.6178	5,900	0.68729

15회의 학습 데이터를 바탕으로, 최소 테스트 오차의 최소값과 최대값은 각각 0.57653과 0.65216임을 확인하였다. 그러나 허용 오차를 최대와 최소의 값으로 정하게 되면, 허용하는 범위가 넓어져 최적의 하이퍼 파라미터를 정하기 어려워진다. 그러므로 본 연구에서는 앞서 획득한 15회의 데이터를 바탕으로 정규분포(Normal distribution)와 표준 정규분포(Standard normal distribution)를 찾아냈다.

정규 분포는 가우시안 분포(Gaussian distribution)라고도 불리는데, 이는 획득한 자료들의 확률 분포를 표현하는데 매우 유용하다. 정규 분포의 확률 밀도는 아래와 같은 수식을 통해 얻어낼 수 있다. [54] [55]

$$f(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (77)$$

- x: 데이터 값
- μ: 데이터 평균
- σ: 데이터 표준 편차

표준 편차(Standard deviation)는 다음과 같이 구할 수 있다. [56] [57]

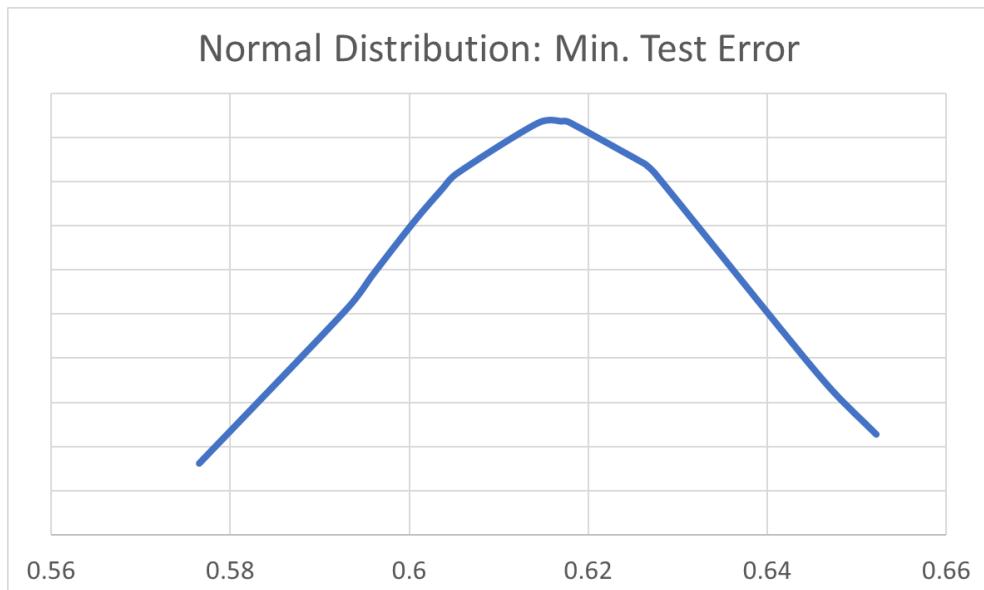
$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{(n - 1)}} \quad (78)$$

- \bar{x} : 데이터 평균
- n: 데이터 수

아래 표와 그림들은 최소 테스트 오차의 최소, 평균, 최대값, 그리고 표준 편차와 정규 분포도를 나타낸 것이다.

<표 3-18> 최소 테스트 오차의 최소값, 평균값, 최대값과 표준편차

Min.	Average	Max.	σ
0.57653	0.61639	0.65216	0.02125



<그림 3-14> 최소 테스트 오차의 정규 분포도

정규 분포는 데이터의 종류에 따라 크기가 달라지기 때문에, 표준편차를 1, 평균을 0으로 맞춘 표준 정규분포도 나타내기도 한다. 표준 정규 분포와 데이터 표준화 식은 다음과 같으며, 이를 통해 구한 표준 정규분포는 <표 3-19>와 같으며, <그림 3-15>는 표준 정규분포도다.

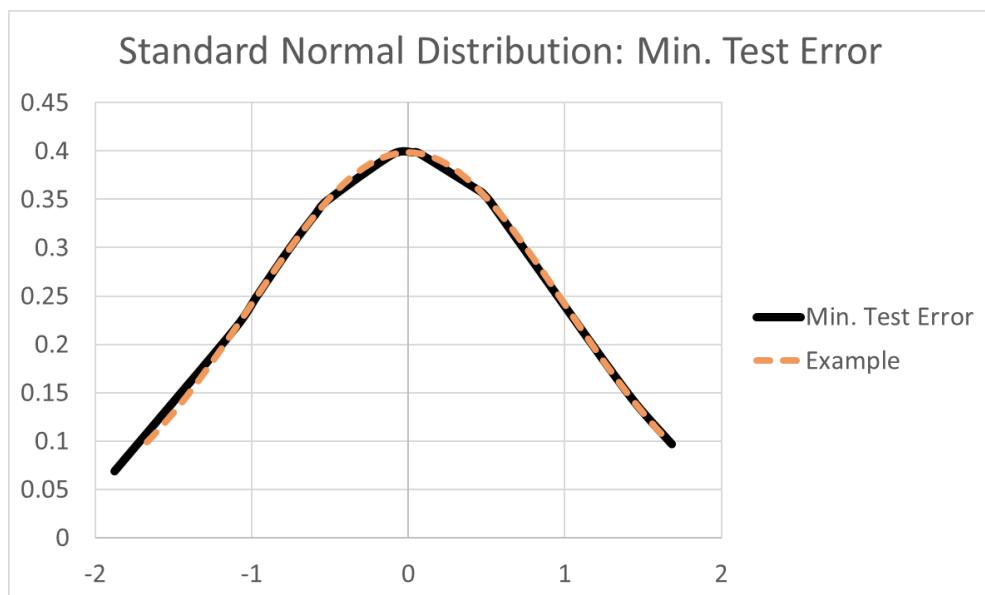
$$z = \frac{(x - \mu)}{\sigma} \quad (79)$$

$$f(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} \quad (80)$$

- x : 데이터 값
- z : 표준화된 데이터 값
- μ : 데이터 평균
- σ : 데이터 표준 편차

<표 3-19> 최소 테스트 오차와 최소 테스트 오차의 표준 정규분포 값

X	Z	σ	X	Z	σ
0.57653	-1.87608	0.068647	0.6178	0.066234	0.398068
0.59297	-1.10236	0.217288	0.62635	0.468628	0.357455
0.59584	-0.96728	0.249885	0.62647	0.474276	0.356505
0.60043	-0.75126	0.300853	0.62743	0.519457	0.348591
0.60375	-0.59501	0.33422	0.64214	1.211762	0.191451
0.60538	-0.5183	0.348801	0.64727	1.453199	0.138785
0.61437	-0.09519	0.397139	0.65216	1.68334	0.096737
0.617	0.028583	0.398779			



<그림 3-15> 최소 학습 오차와 예시 데이터의 표준 정규분포도

<그림 3-15>에서 실선은 최소 테스트 오차이며, 점선은 -2부터 2까지 0.1씩 증가하는 임의의 데이터를 통해 만들어낸 표준 정규분포 예시 그래프이다. 두 그래프를 비교해보면 최소 테스트 오차 데이터가 예시 그래프를 잘 따르는 것을 확인할 수 있으며, 15회의 반복 학습만으로도 확률 분포를 구하기에 충분하다는 것을 알 수 있다. <표 3-20>은 예시 데이터와 예시 데이터의 표준 정규분포 값이다.

<표 3-20> 예시 데이터와 예시 데이터의 표준 정규분포 값

X	Z	σ	X	Z	σ
-2	-1.66957	0.098997	0.1	0.083478	0.397555
-1.9	-1.58609	0.113406	0.2	0.166957	0.393421
-1.8	-1.50261	0.129011	0.3	0.250435	0.386626
-1.7	-1.41913	0.145743	0.4	0.333914	0.37731
-1.6	-1.33565	0.163503	0.5	0.417392	0.365662
-1.5	-1.25218	0.182153	0.6	0.50087	0.351912
-1.4	-1.1687	0.20152	0.7	0.584349	0.336327
-1.3	-1.08522	0.221399	0.8	0.667827	0.319201
-1.2	-1.00174	0.24155	0.9	0.751305	0.300842
-1.1	-0.91826	0.261704	1	0.834784	0.281571
-1	-0.83478	0.281571	1.1	0.918262	0.261704
-0.9	-0.75131	0.300842	1.2	1.001741	0.24155
-0.8	-0.66783	0.319201	1.3	1.085219	0.221399
-0.7	-0.58435	0.336327	1.4	1.168697	0.20152
-0.6	-0.50087	0.351912	1.5	1.252176	0.182153
-0.5	-0.41739	0.365662	1.6	1.335654	0.163503
-0.4	-0.33391	0.37731	1.7	1.419133	0.145743
-0.3	-0.25044	0.386626	1.8	1.502611	0.129011
-0.2	-0.16696	0.393421	1.9	1.586089	0.113406
-0.1	-0.08348	0.397555	2	1.669568	0.098997
0	0	0.398942			

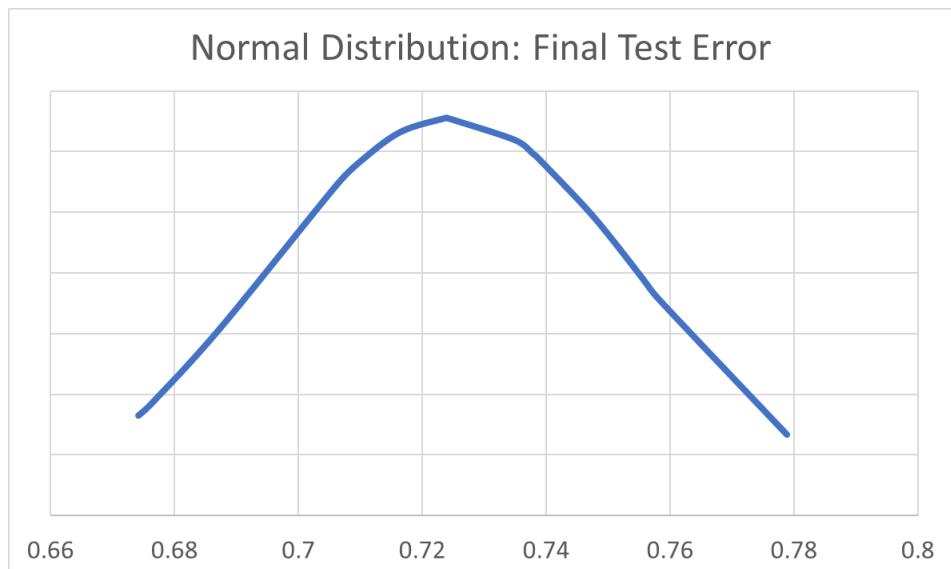
<표 3-21>은 최종 테스트 오차의 최소, 평균, 최대, 표준편차 값, <그림 3-16>은 최종 테스트 오차의 정규분포도, <표 3-22>와 <그림 3-17>은 표준 정규분포 값과 그래프이다. 최종 테스트 오차 역시 표준 정규분포를 잘 따르고 있음을 알 수 있다.

<표 3-21> 최종 테스트 오차의 최소값, 평균값, 최대값과 표준편차

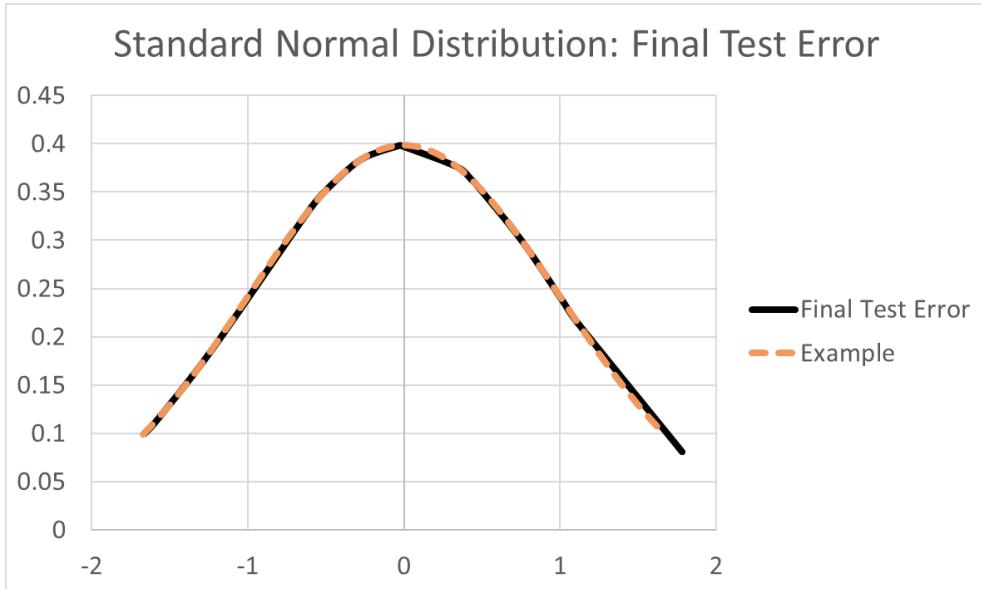
Min.	Average	Max.	σ
0.67414	0.72463	0.77884	0.03039

<표 3-22> 최종 테스트 오차와 최종 테스트 오차의 표준 정규분포 값

X	Z	σ	X	Z	σ
0.67414	-1.66105	0.100411	0.73494	0.339317	0.376625
0.6762	-1.59328	0.112118	0.73781	0.433742	0.363126
0.68729	-1.22841	0.187603	0.73859	0.459405	0.358988
0.70613	-0.60856	0.331506	0.74755	0.754196	0.300189
0.71003	-0.48024	0.355491	0.75554	1.017074	0.23784
0.71643	-0.26968	0.384696	0.75819	1.453199	0.138785
0.72379	-0.02753	0.398791	0.77884	1.783662	0.081296
0.72393	-0.02292	0.398837			

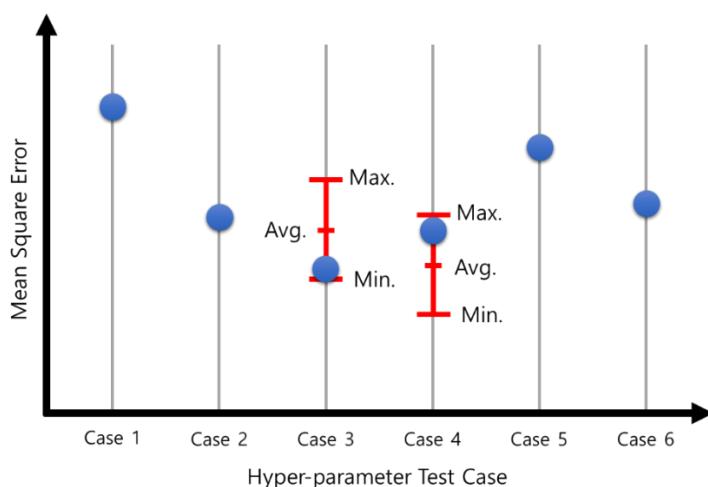


<그림 3-16> 최종 테스트 오차의 정규분포도



<그림 3-17> 최종 테스트 오차와 예시 데이터의 표준 정규분포도

앞서 언급하였듯이, 표준 편차를 확인하는 이유는 최적의 하이퍼 파라미터 값을 강건하게 찾기 위한 오차 범위를 정하기 위함이다. <그림 3-18>은 6가지 Test case에 대한 MSE 예시를 나타낸 것이다. 학습을 한번씩만 시행하면 학습 오차에 대한 확률분포함수인 정규분포를 만들 수 없기에, 한 번의 테스트 결과로 최적 모델을 결정해야 한다.



<그림 3-18> 하이퍼 파라미터 Test Case의 오차와 오차 범위 예시

예시 그림의 경우, 6개의 Test case에서 Case 3가 가장 낮은 오차를 보여준다. 하지만 이 결과는 Case 3는 운이 좋게 가장 좋은 결과를 보여주고 Case 4는 운이 나쁘게도 가장 높은 오차를 보여준 것일 수도 있다. 그러면 다음 학습에서도 Case 3가 여전히 좋은 결과를 보여줄 수 있는지 장담하기 어렵다. 만약 그 분포를 실제로 측정해 본 결과, <그림 3-18>과 같다면, 확률적으로 Case 4가 더 나은 결과를 보여줄 가능성성이 크다.

좋은 모델을 만들어내는 가장 좋은 방법은 모든 경우마다 표준 편차를 구하는 것이지만, 이는 시간의 소모가 크다. Test case가 여섯 개만 되어도 최소 15개의 표본을 모으기 위해서는 총 90개의 모델을 만들어야 하는데, 병렬적으로 학습하지 않는 이상 많은 시간이 필요하다. 그러므로 본 연구에서는 기준 하이퍼 파라미터를 사용하여 구한 표준 편차를 다른 하이퍼 파라미터 조건에서도 사용하여, 각 하이퍼파라미터 테스트에서 가장 작은 오차로부터 일정 표준편차 내에 있는 Test case를 최적 후보군으로 선택했다. 이 때, 허용 오차가 너무 크면 실제 분포 범위를 벗어나기 쉽고, 너무 좁으면 선택의 폭이 좁아진다. 본 연구에서는 허용오차 범위를 $+1\sigma$ 로 하였으며, 가장 작은 오차로부터 해당 범위에 있는 Case들을 최적 하이퍼 파라미터 후보군으로 선정하였다.

B. 은닉층 수 최적화

인공신경망은 은닉층의 수가 깊을수록 더 복잡한 문제를 해결할 수 있다. 그러나 은닉층 수가 많으면 연산해야 할 양이 많아져 Forward propagation에 시간이 더 많이 소요된다. 본 연구에서는 이전 연구에서 사용된 최종 최적 뉴런 수(5개)를 기본값으로 사용하였다. 또한 최적값을 찾기 위해 은닉층 수를 3개부터 9개까지 늘려가면서 비교하였다.

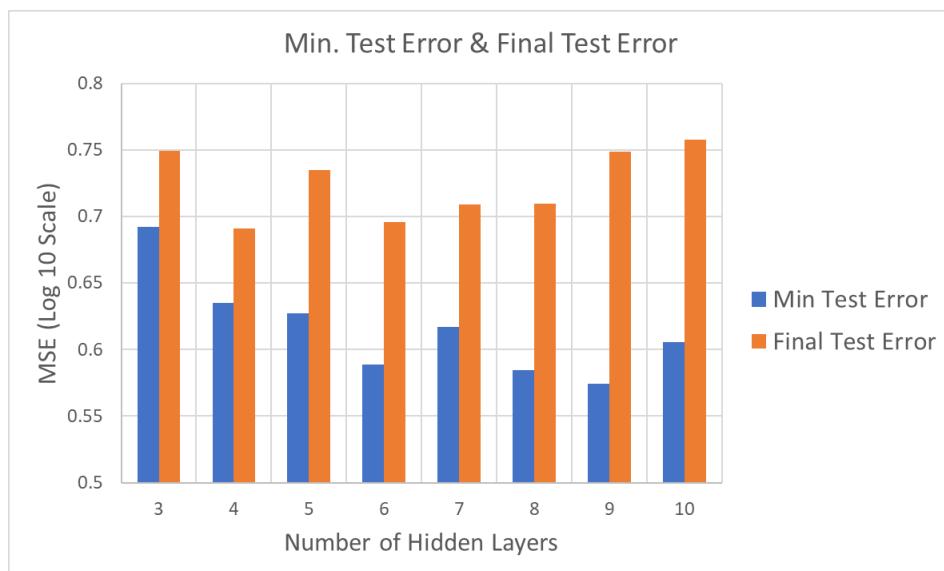
<표 3-23> Initial Hyper-parameter Optimization(IHO): 은닉층 수 Test Cases

Case 1	Case 2	Case 3 Baseline	Case 4	Case 5	Case 6	Case 7	Case 8
3	4	5	6	7	8	9	10

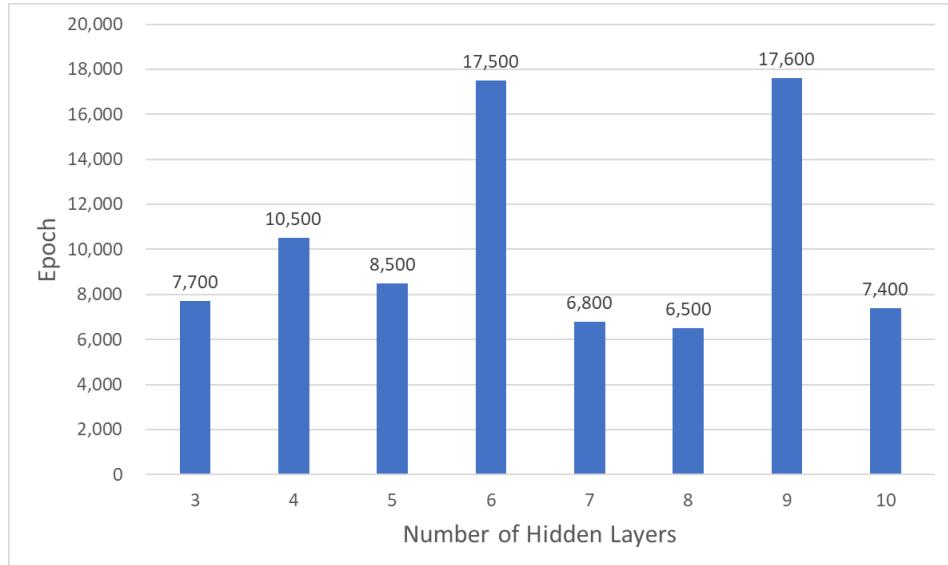
아래 표와 그림들은 각 Case에 대하여 테스트 한 결과로, 최소 테스트 오차와 이 때의 Epoch, 그리고 최종 테스트 오차를 정리한 것이다. 굵게 표시한 수치들은 가장 작은 최소 테스트 오차로부터 허용 오차 내에 있는 최적 하이퍼 파라미터 후보군이다. 가장 작은 값은 Case 9의 0.57450이며, $+1\sigma$ 값은 0.59575이다. 범위 내에 있는 Test case는 Case 6과 Case 8이다.

<표 3-24> IHO: 은닉층 Test 결과

Case	Number of Hidden Layers	Min. Test Error	Epoch of Min. Test Error	Final Test Error
Case 1	3	0.6924	7,700	0.74925
Case 2	4	0.63518	10,500	0.69081
Case 3	5	0.62743	8,500	0.73494
Case 4	6	0.5889	17,500	0.69582
Case 5	7	0.61697	6,800	0.70872
Case 6	8	0.58425	6,500	0.70946
Case 7	9	0.5745	17,600	0.74867
Case 8	10	0.60571	7,400	0.75752

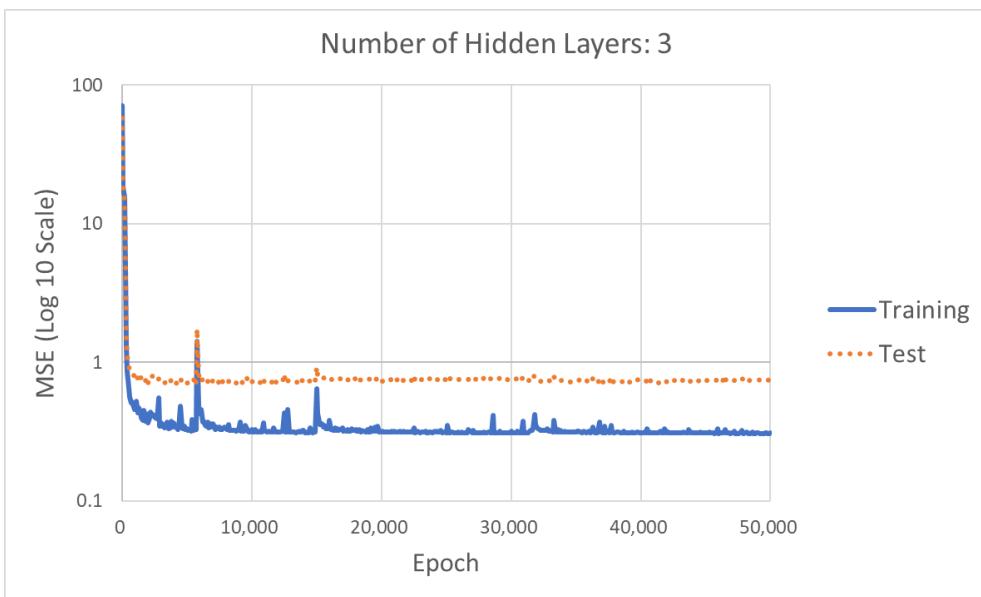


<그림 3-19> IHO - Hidden Layer Test: 최소/최종 테스트 오차

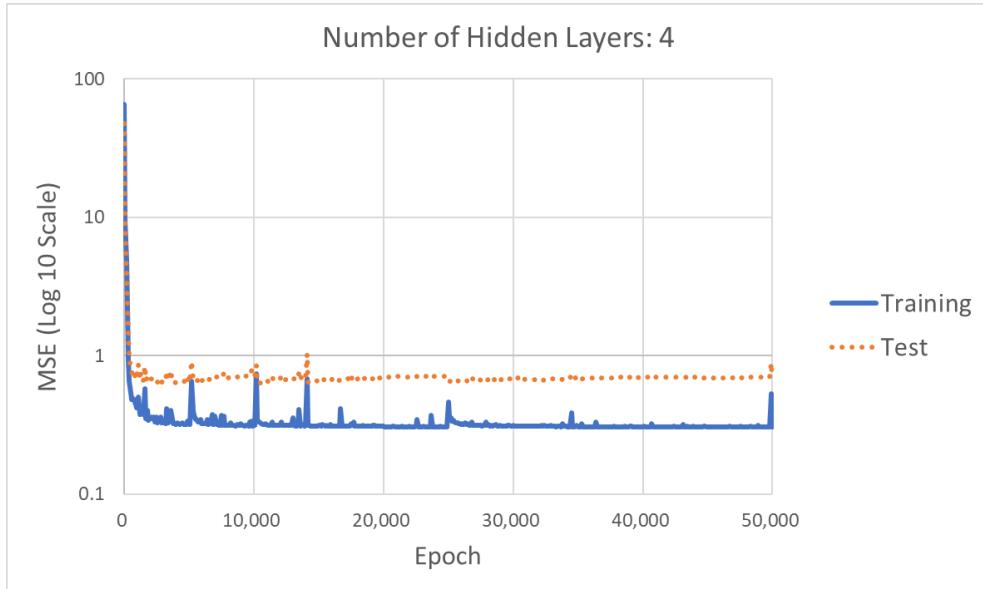


<그림 3-20> IHO - Hidden Layer Test: 최소오차 Epoch

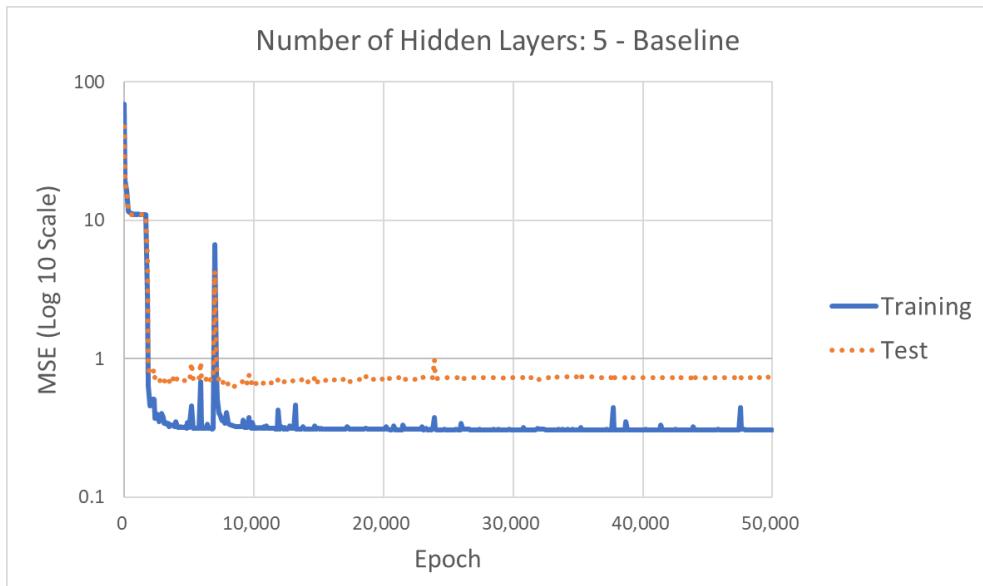
다음은 학습 양상을 나타낸 그래프로서, Epoch에 따른 신경망의 학습/테스트 오차를 나타낸 것이다. 그래프에서 볼 수 있듯이 Epoch가 증가해도 오차가 줄지 않는 한계가 있음을 알 수 있다.



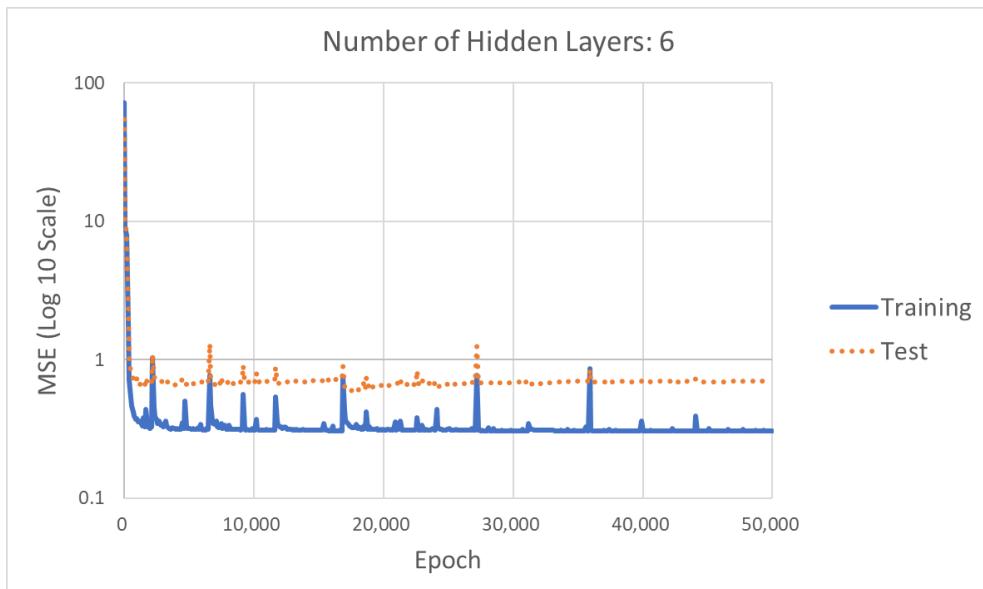
<그림 3-21> IHO – Hidden Layer Test 학습/테스트 오차: 은닉층 수 3개



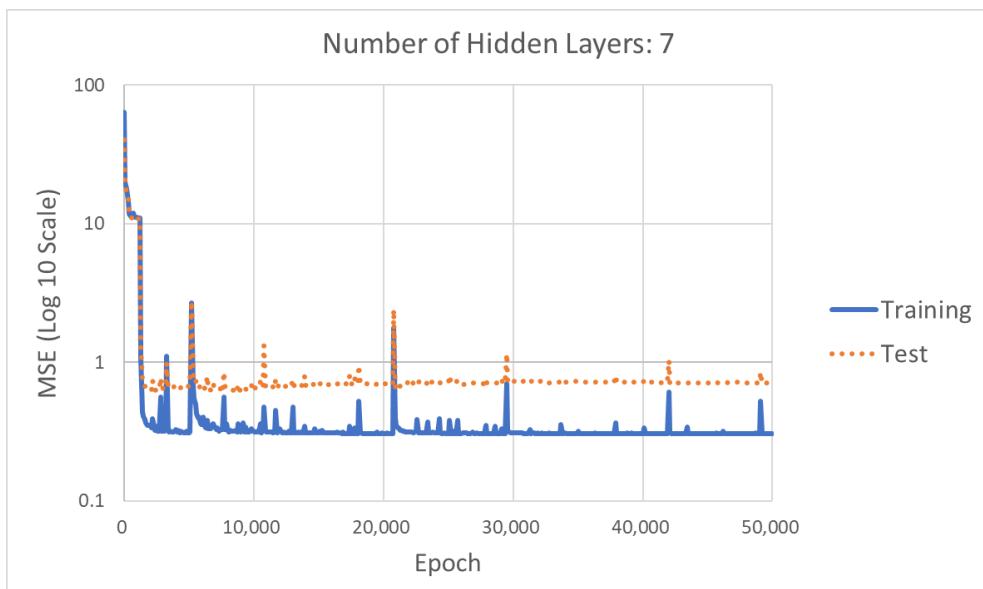
<그림 3-22> IHO – Hidden Layer Test 학습/테스트 오차: 은닉층 수 4개



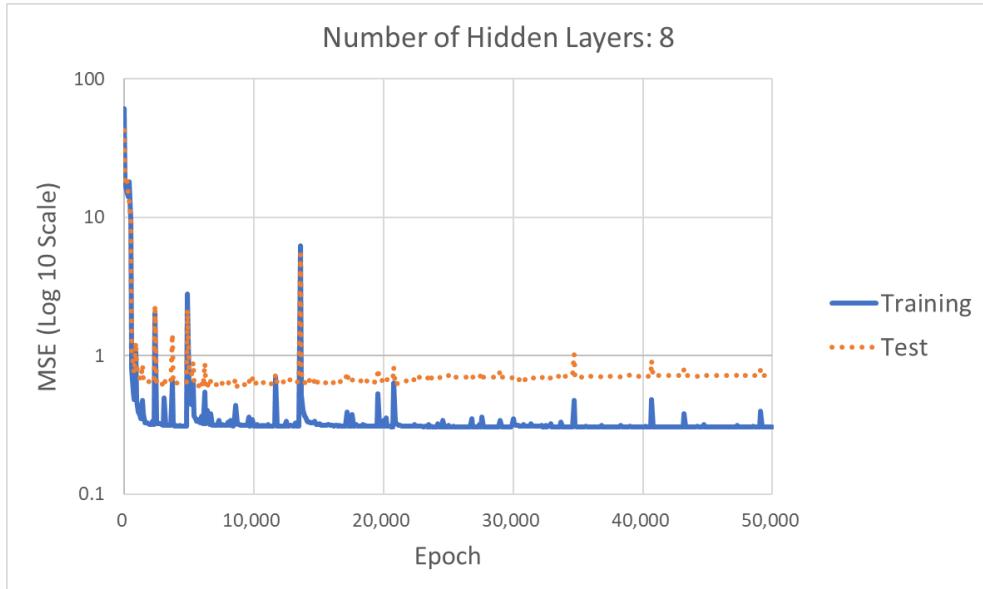
<그림 3-23> IHO – Hidden Layer Test 학습/테스트 오차: 은닉층 수 5개
(Baseline)



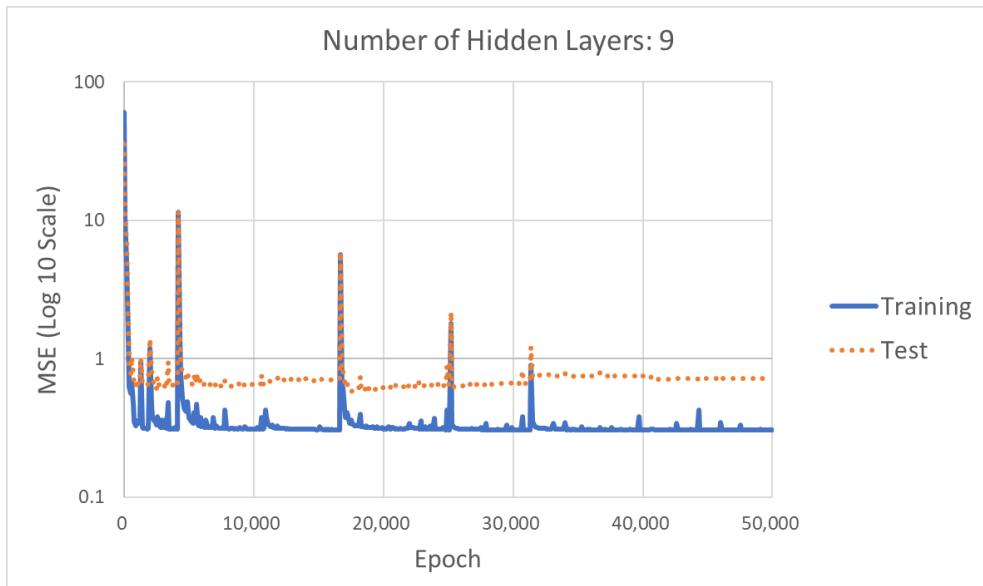
<그림 3-24> IHO – Hidden Layer Test 학습/테스트 오차: 은닉층 수 6개



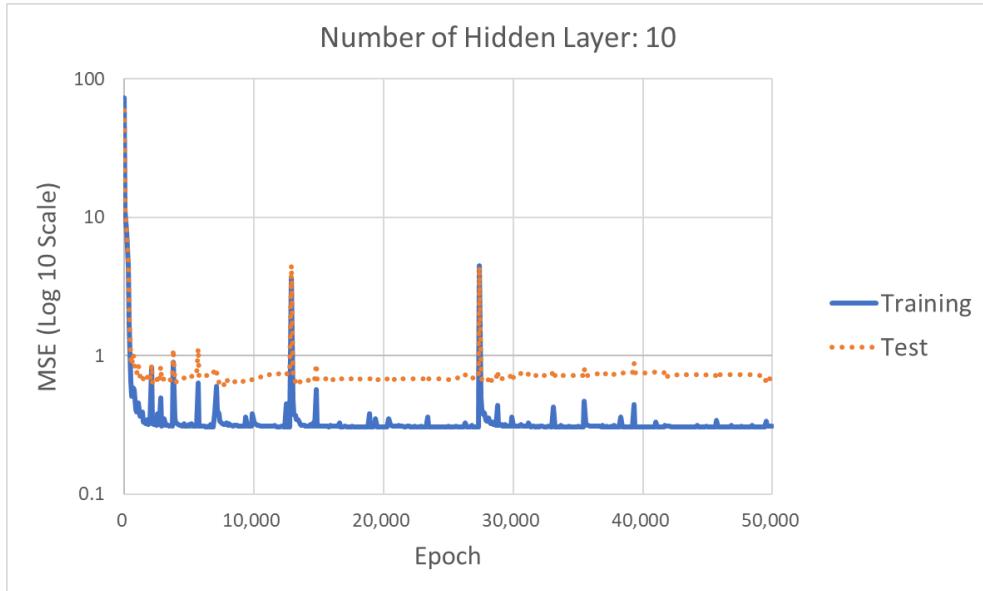
<그림 3-25> IHO – Hidden Layer Test 학습/테스트 오차: 은닉층 수 7개



<그림 3-26> IHO – Hidden Layer Test 학습/테스트 오차: 은닉층 수 8개



<그림 3-27> IHO – Hidden Layer Test 학습/테스트 오차: 은닉층 수 9개



<그림 3-28> IHO – Hidden Layer Test 학습/테스트 오차: 은닉층 수 10개

C. 은닉층 당 뉴런 수 최적화

일반적으로 뉴런 수는 입력값과 출력값의 중간 값을 사용한다. 그러나 이는 이미지 학습과 같은 문제에서 사용되는 방법이며, 아직까지 명료하게 적절한 은닉층의 뉴런 수를 찾는 방법은 밝혀지지 않았다. [7] 다만 이는 신경망이 각 층에서 표현할 수 있는 차원의 수를 나타낸다고 알려져 있다. 본 연구에서는 기본 값을 앞서 300개로 설정하였으며, 입력층과 출력층 뉴런 수의 중간 값인 4개부터 시작하여 700개까지 증가시켜가며 비교하였다.

<표 3-25> IHO: 은닉층 당 뉴런 수 Test Cases

Case 1	Case 2	Case 3	Case 4 Baseline	Case 5	Case 6	Case 7	Case 8
4	50	100	300	400	500	600	700

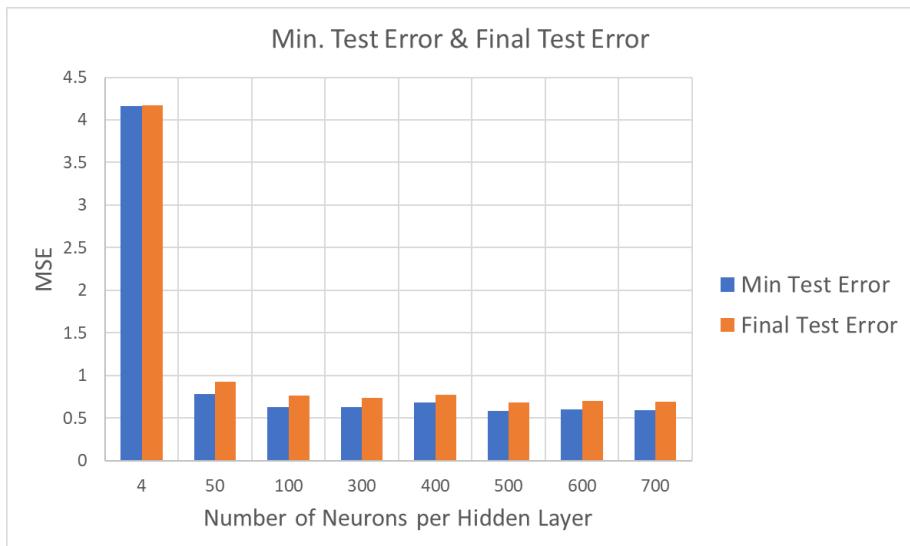
다음 표와 그림들은 각 Case의 테스트 결과로, 최소 테스트 오차와 이 때의 Epoch, 그리고 최종 테스트 오차를 정리한 것이다.

<표 3-26> IHO: 은닉층 당 뉴런 수 Test 결과

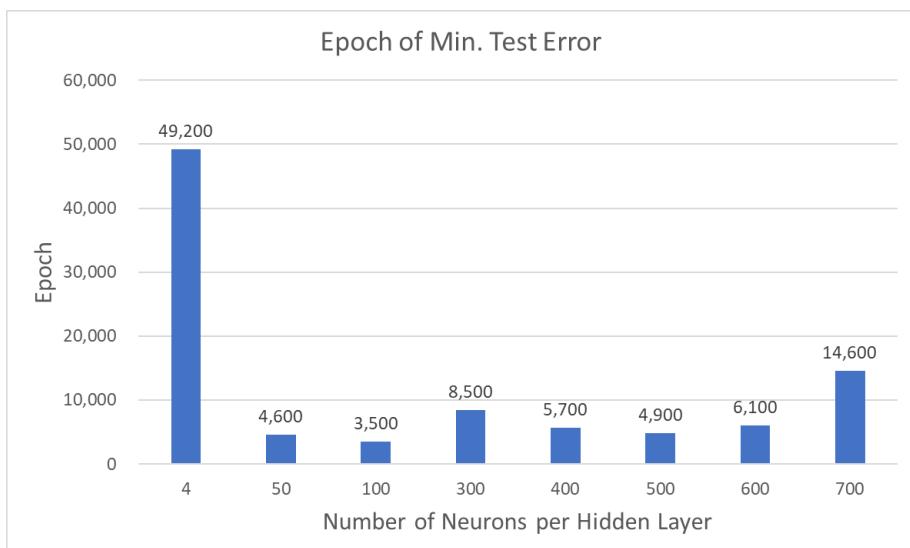
Case	Number of Neurons	Min. Test Error	Epoch of Min. Test Error	Final Test Error
Case 1	4	4.16029	49,200	4.16691
Case 2	50	0.78508	4,600	0.92131
Case 3	100	0.63081	3,500	0.76711
Case 4	300	0.62743	8,500	0.73494
Case 5	400	0.68219	5,700	0.77033
Case 6	500	0.58056	4,900	0.6825
Case 7	600	0.59721	6,100	0.70088
Case 8	700	0.5926	14,600	0.68839

<표 3-26>에서 가장 작은 오차를 내는 Case는 6번이며, 1σ 범위 내에 있는 후보군은 7번과 8번이다. 특이한 점은, 인공신경망에서 일반적으로 사용하는 방법에 따라 뉴런 수를 선정한 Case 1이 매우 나쁜 학습 결과를 보여주었다는 것이다. Case 2~5도 후보군에 비하면 그 오차가 크지만 그 차이는 매우 작다. Case 1은 최종 오차, 최소 오차 모두 4를 넘겨, 소수점 자리수가 차이 날 만큼 오차가 특이하게 크다.

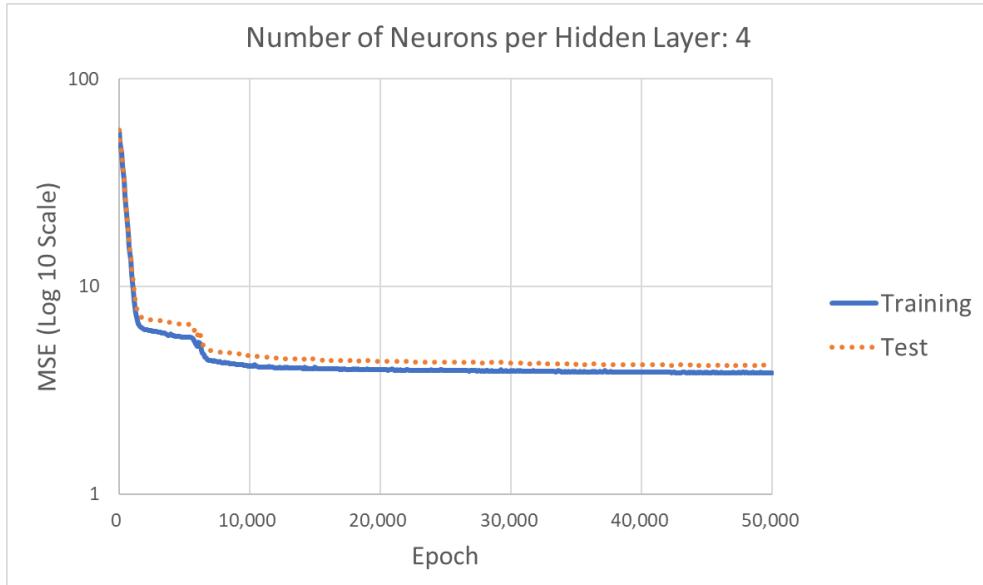
은닉층 당 뉴런 수는 각 은닉층의 차원을 결정하므로, 오차가 특이하게 큰 Case 1의 경우, 4개의 뉴런으로는 문제를 풀기에 충분한 차원을 확보하지 못한 것으로 추측된다. 앞서 언급한 것처럼, Case 1은 일반적으로 알려진 방법에 따라 입/출력 뉴런 수의 중간값을 사용한 것이다. 그러나 최근에 개발되고 연구되는 대부분의 인공신경망은 이미지나 음성과 같은 문제를 해결하는 것들이다. 이러한 문제들의 특징은 입력과 출력 데이터가 많기 때문에 입/출력 뉴런 수가 수백~수천개로 매우 많다는 것이다. 반면, 본 문제는 입력층은 5개, 출력층은 3개이다. 그러므로 동일한 방법을 사용해도 좋은 결과가 나오지 않는다는 것은, 문제가 달라지면 신경망의 뉴런 수를 정하는 방법도 달라져야 한다는 것으로 볼 수 있다



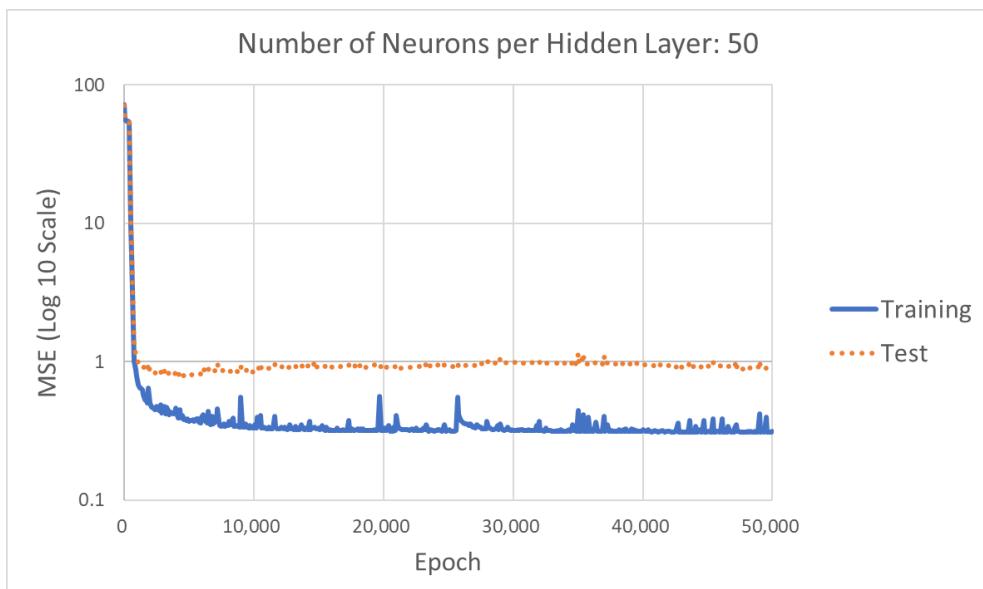
<그림 3-29> IHO – Number of Neurons Test: 최소/최종 테스트 오차



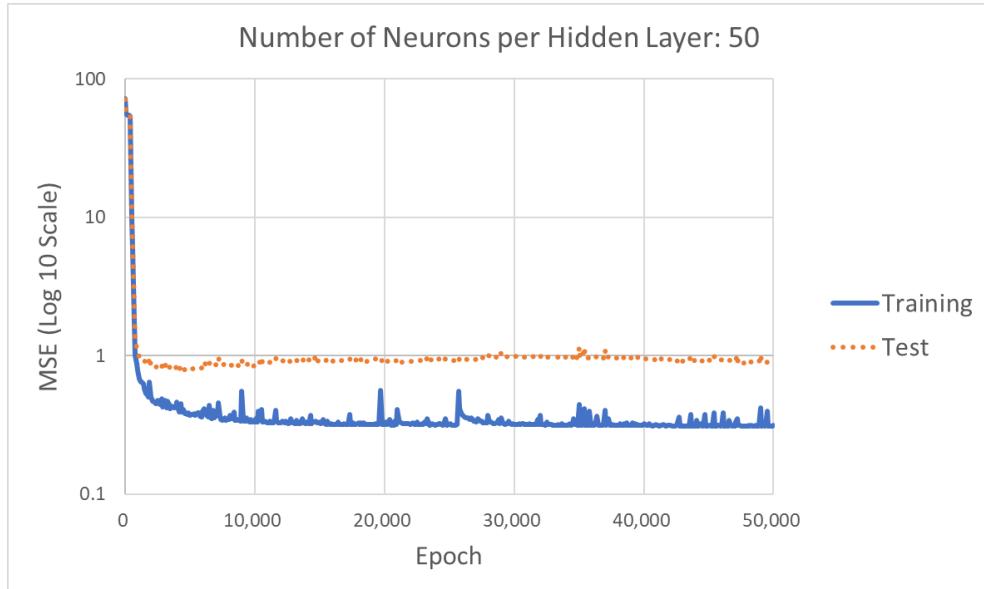
<그림 3-30> IHO – Number of Neurons Test: 최소 테스트 오차 Epoch



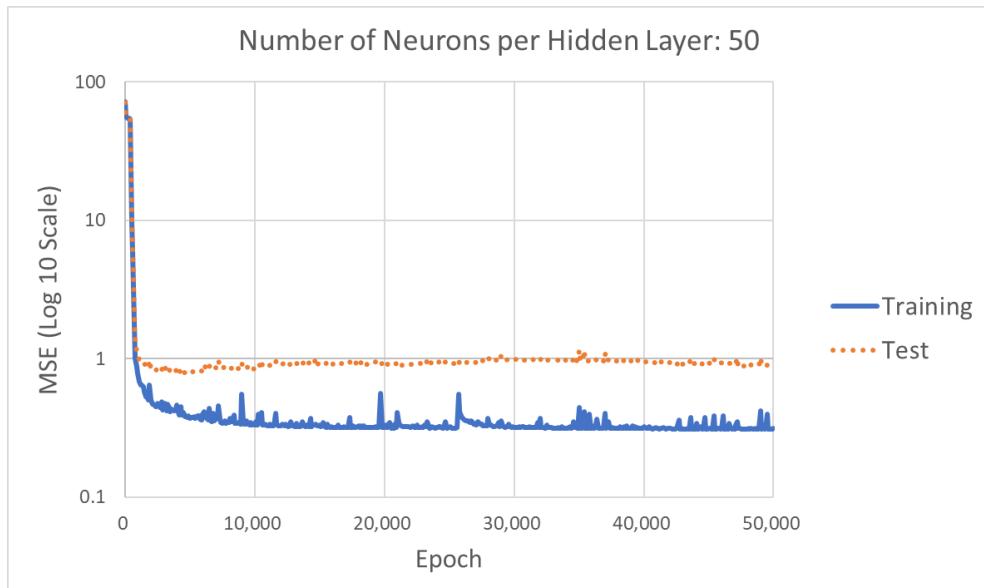
<그림 3-31> IHO – Number of Neurons Test: 4 Neurons per Hidden Layer



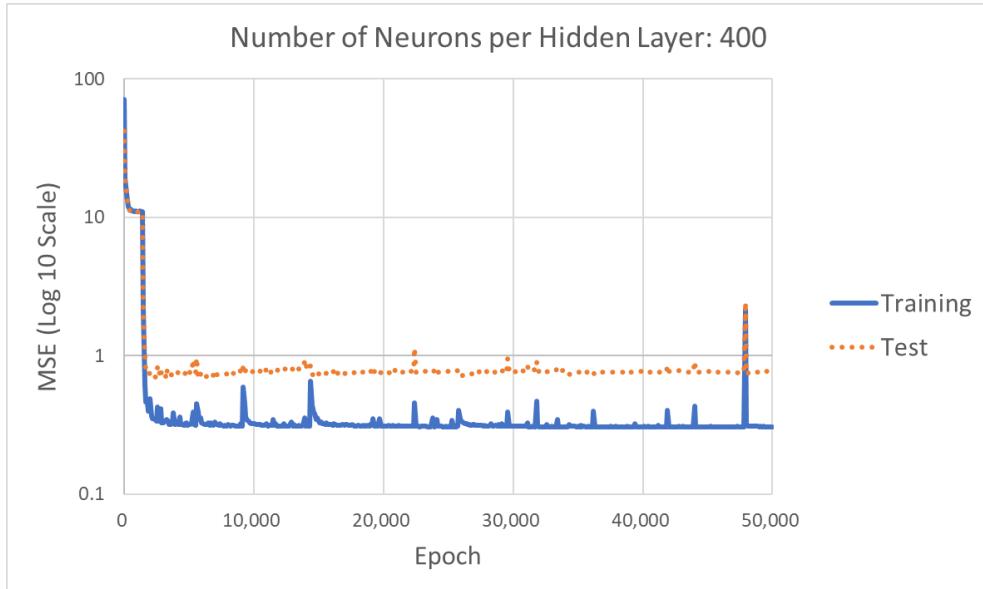
<그림 3-32> IHO – Number of Neurons Test: 50 Neurons per Hidden Layer



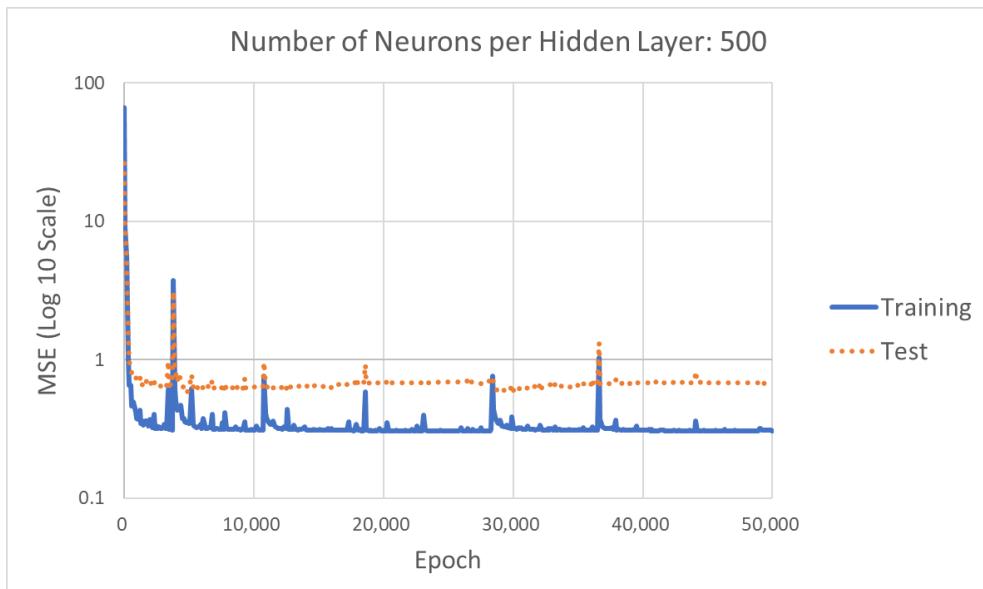
<그림 3-33> IHO – Number of Neurons Test: 100 Neurons per Hidden Layer



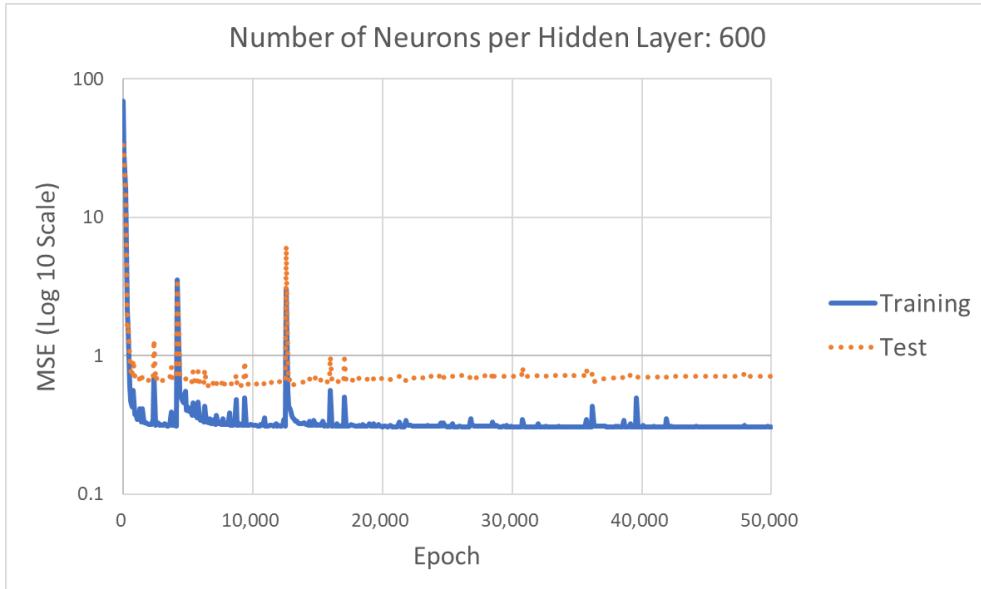
<그림 3-34> IHO – Number of Neurons Test: 300 Neurons per Hidden Layer
(Baseline)



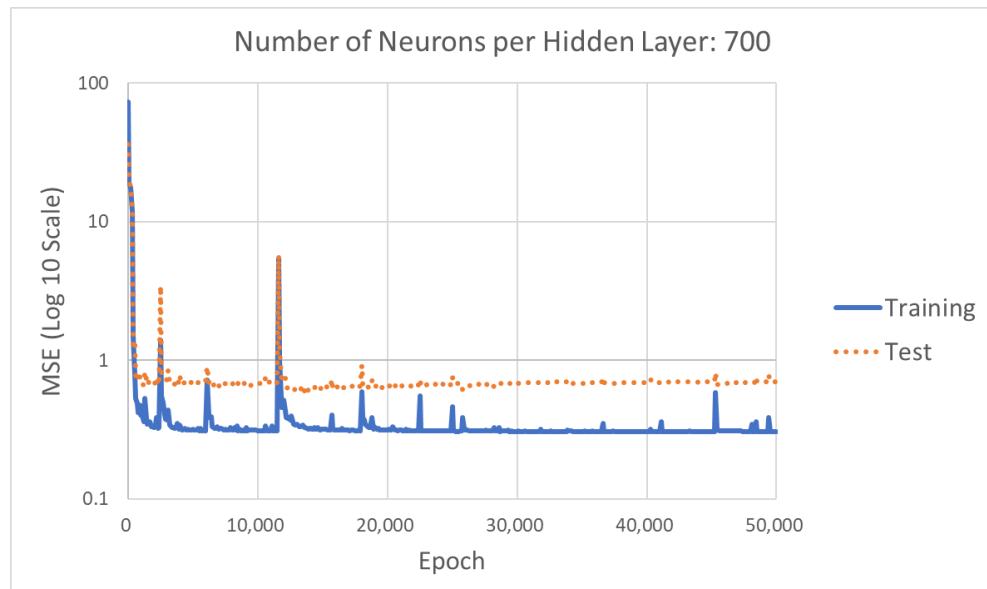
<그림 3-35> IHO – Number of Neurons Test: 400 Neurons per Hidden Layer



<그림 3-36> IHO – Number of Neurons Test: 500 Neurons per Hidden Layer



<그림 3-37> IHO – Number of Neurons Test: 600 Neurons per Hidden Layer



<그림 3-38> IHO – Number of Neurons Test: 700 Neurons per Hidden Layer

D. 최적 하이퍼 파라미터 선정 및 학습

앞서 은닉층 수와 은닉층 당 뉴런 수를 조절하며 학습의 정도를 살펴보고, 최적의 하이퍼 파라미터 후보군을 선정하였다. 은닉층 수는 6개, 8개, 9개가 후보군으로 선정되었고, 은닉층 당 뉴런 수는 500개, 600개, 700개가 선정되었다.

일반적으로 인공신경망을 구현할 때, 신경망 구조는 최대한 단순하게 만드는데, 이는 신경망이 복잡해지면 학습하기 어렵고 시간이 오래 걸리며, 실제 학습이 완료된 모델을 사용할 때에도 예측 시간을 증가시키기 때문이다. <표 3-27>과 <표 3-28>은 각 Test case의 학습 시간을 기록한 것이다. 두 하이퍼 파라미터의 Test case에서, 은닉층이 깊어지거나 뉴런 수가 많아질수록 학습에 소요되는 시간이 많아지는 것을 알 수 있다. 그러나 예외로 은닉층 당 뉴런 수가 4개일 때, 뉴런 수가 가장 적음에도 불구하고 오히려 학습 시간이 오래 걸렸다. 이는 뉴런 수가 너무 작아 적합한 모델을 만들기 어려워 시간이 오래 걸리는 것으로 추측된다.

<표 3-27> IHO: 은닉층 Test Case 별 학습 시간

Case	1	2	3	4	5	6	7	8
Hidden Layer	3	4	5	6	7	8	9	10
Time (min.)	7	6.53	7.52	8.33	9.7	10.27	13.95	12.33

<표 3-28> IHO: 은닉층 당 뉴런 수 별 학습 시간

Case	1	2	3	4	5	6	7	8
Neuron per HL	4	50	100	300	400	500	600	700
Time (min.)	8.31	6.97	6.65	7.52	8.14	8.59	11.7	10.44

본 연구에서는 앞서 선정된 최적 하이퍼 파라미터 후보군들 중, 가장 수가 적고 학습 시간이 짧은 것들을 선택하였다. 이는 추후 진행할 데이터 베이스 강화에서 학습에 걸리는 시간을 최대한 줄이기 위함이다. 또한, 이번 은닉층 당 뉴런 수 비교를 통해, 최소 테스트 오차가 일어나는 Epoch 가 대부분 20,000을 넘지 않는다는 것을 확인하고, 최대 학습 횟수도 20,000번으로 줄였다. 선정된 최적 하이퍼 파라미터는 <표 3-29>와 같다.

<표 3-29> 초기 하이퍼 파라미터 최적화: 최적화 전/후 비교

Hyper-parameter	Before	After
Number of Hidden Layers	5	6
Number of Neurons per Hidden Layer	300	500
Epoch	50,000	20,000

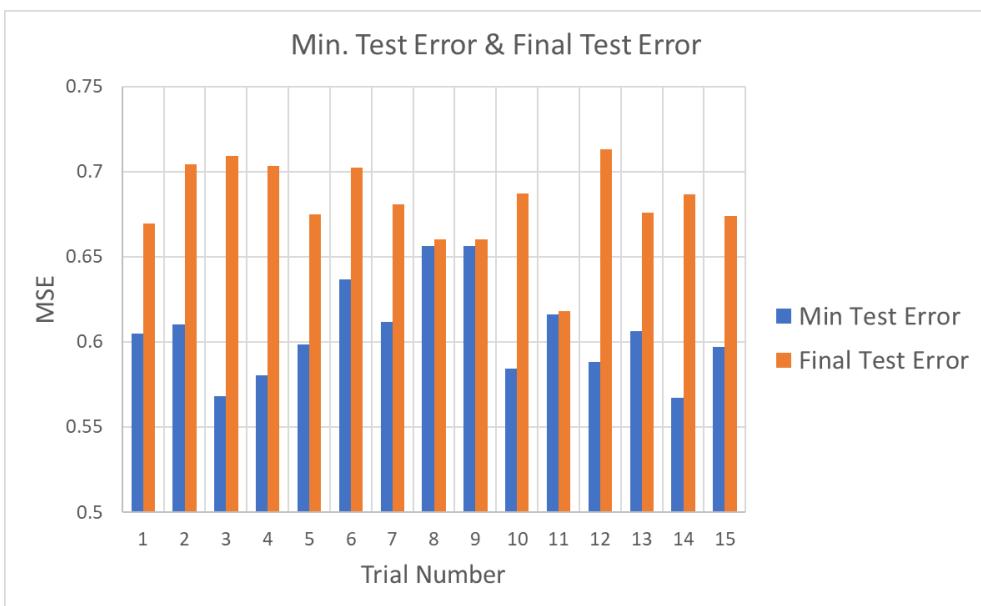
1) 최적 하이퍼 파라미터에서의 학습 결과

앞서 선정한 최적 하이퍼 파라미터 조건에 따른 학습 양상을 살펴보기 위해 해당 조건에서 15회 반복학습을 실시하였다. <표 3-30>을 보면, 최소 학습 오차는 모두 20,000 Epoch 미만에서 나타난 것을 알 수 있다. 또한 최적화 이전에 실시한 반복 학습과 마찬가지로, 최적 하이퍼 파라미터를 이용한 15개 모델들의 최소/최종 테스트 오차들을 이용하여 평균과 표준편차를 구하였다. 이를 통해 하이퍼 파라미터 최적화의 영향을 볼 수 있었다.

<표 3-30> 최적 하이퍼 파라미터에서
15회 학습 Test Error 및 최소 Test Error Epoch 결과

Trial	Min. Test Error	Epoch of Min. Test Error	Final Test Error
1	0.60491	8,500	0.66931
2	0.6103	5,100	0.70438
3	0.56834	5,800	0.70922
4	0.58025	4,600	0.7034

5	0.5985	5,900	0.67474
6	0.63676	4,100	0.70222
7	0.6116	2,000	0.68063
8	0.65651	19,900	0.66013
9	0.65651	19,900	0.66013
10	0.58443	2,500	0.68704
11	0.61608	1,9600	0.61797
12	0.58815	5,400	0.71308
13	0.60632	2,700	0.67587
14	0.56734	6,300	0.68675
15	0.59711	8,900	0.67392

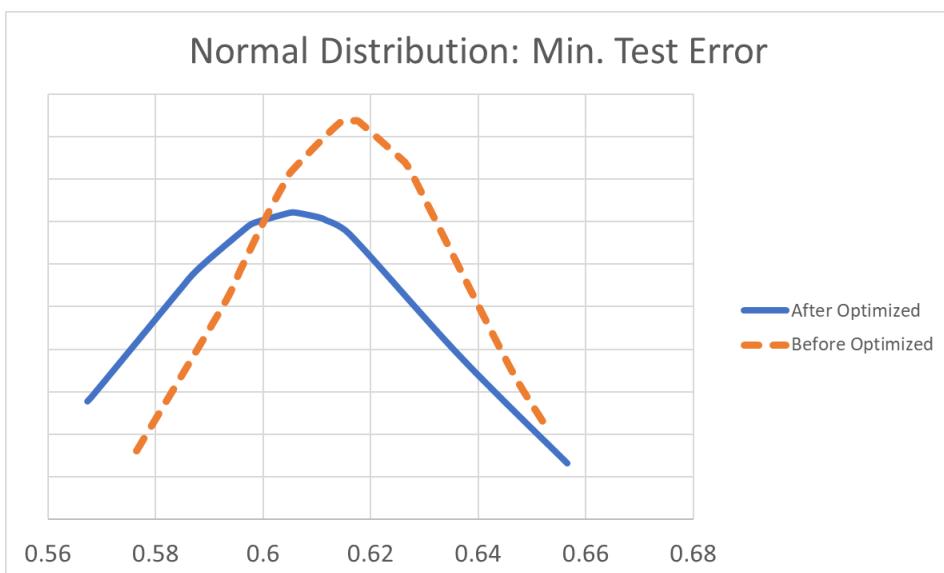


<그림 3-39> 최적 하이퍼 파라미터에서
15회 학습 Test Error 및 최소 Test Error Epoch 결과

<표 3-31>을 보면 최적화를 통해 평균과 최소값은 줄어들었으나, 최대값과 표준편차는 늘어났다. 이는 평균이 낮아 더 좋은 학습 결과를 기대할 수 있지만, 분포가 넓어 좋은 학습 결과를 얻기 위해서는 여러 번 학습을 시도해야 한다는 것으로 볼 수 있다.

**<표 3-31> 하이퍼 파라미터 최적화 전후
최소 테스트 오차의 최소값, 평균값, 최대값과 표준편차 비교**

	Min.	Average	Max.	σ
Before Optimized	0.57653	0.61639	0.65216	0.02125
After Optimized	0.56734	0.60554	0.65651	0.02764
Difference	-0.00919	-0.01085	+0.00435	+0.00639

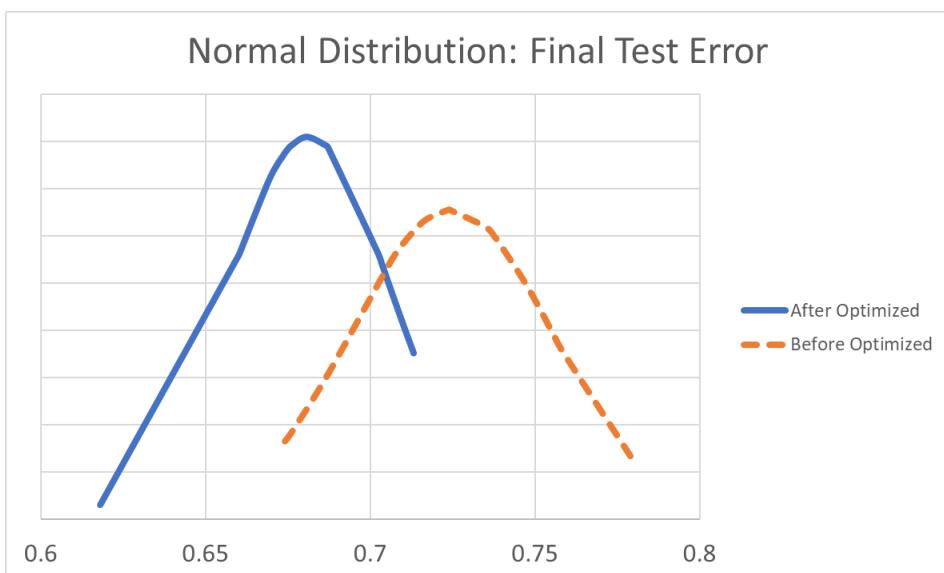


<그림 3-40> 하이퍼 파라미터 최적화 전후의 최소 테스트 오차 분포

<표 3-32>에서 최종 테스트 오차의 최대, 최소, 그리고 평균과 표준편차의 변화를 볼 수 있는데, 모든 값들이 작아진 것을 알 수 있다. 학습의 양상은 매 실행마다 다르게 나타나기에, 최소 테스트 오차가 나타나는 Epoch는 매 학습마다 다르게 나타난다. 그러므로 일반적으로는 Epoch를 적절한 수로 정한 채, 반복적인 학습을 통해 최종 모델을 얻어내거나 도중에 학습을 중지한다.

**<표 3-32> 하이퍼 파라미터 최적화 전후
최종 테스트 오차의 최소값, 평균값, 최대값과 표준편차 비교**

	Min.	Average	Max.	σ
Before Optimized	0.67414	0.72463	0.77884	0.03039
After Optimized	0.61797	0.68125	0.71308	0.02462
Difference	-0.05617	-0.04338	-0.06576	-0.00577



<그림 3-41> 최적화된 하이퍼 파라미터에서의 최종 테스트 오차 정규분포

하이퍼 파라미터 최적화를 통해 최종 오차가 전반적으로 줄어들었다는 것은, 사용자가 추가적인 알고리즘을 넣지 않고 Epoch를 일정하게 유지한 채 학습을 하여도 오차를 줄이는 데 효과를 낸다는 것이며, 그 분포가 전반적으로 오차가 감소하는 것으로 나타난 것은 무작위로 학습을 진행해도 최적화 이전보다 더 좋은 결과를 얻어낼 수 있다는 것이다.

앞서 살펴본 오차의 통계적 수치들을 보면, Epoch를 20,000으로 제한하면 인공신경망 모델의 오차는 평균적으로 0.68125를 기대할 수 있을 것이다. 만약 학습을 적절한 시점에서 중단한다면, 평균적으로 기대할 수 있는 값

은 0.60554일 것이다. 이보다 더 정확한 결과를 원한다면, 최소 테스트 오차의 분포를 이용하여 알고리즘이 원하는 수준 이하의 오차를 내는 모델을 만들어낼 때까지 초기화하고 학습시키는 과정을 반복하면 되지만, 그래도 여전히 최소 0.56 정도의 오차가 있을 것임을 알 수 있다. 0.56은 약 1.65σ 로, 이보다 더 낮은 오차를 갖는 모델을 획득할 확률은 약 5%이다. <표 3-33>은 최적 하이퍼 파라미터로 15개의 모델을 만들고 학습한 결과이며, 가장 낮은 오차를 기록한 모델 14은 최소 오차가 0.56734였다.

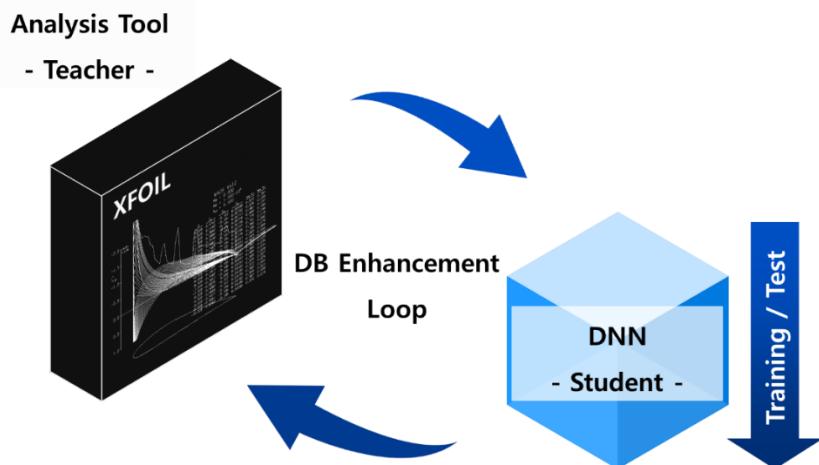
**<표 3-33> 최적화된 하이퍼 파라미터로
학습한 모델들의 최소/최종 테스트 오차**

Trial	Min Test Error	Final Test Error
1	0.60491	0.66931
2	0.6103	0.70438
3	0.56834	0.70922
4	0.58025	0.7034
5	0.5985	0.67474
6	0.63676	0.70222
7	0.6116	0.68063
8	0.65651	0.66013
9	0.65651	0.66013
10	0.58443	0.68704
11	0.61608	0.61797
12	0.58815	0.71308
13	0.60632	0.67587
14	0.56734	0.68675
15	0.59711	0.67392

5%의 확률로 0.56 이하의 오차를 갖는 모델을 얻는 것은 운에 결정되는 것이며 불확실한 방법이다. 그리고 0.56의 오차는 상당히 큰 수치이다. 그러므로 보다 확실하게 더 적은 오차를 갖는 모델을 만들 방법이 필요하다.

5. 데이터베이스 강화 루프(Database Enhancement Loop, DEL)

인공신경망의 학습이 더 이상 개선되지 않고 학습을 지속할수록 오차가 증가하는 추세를 보일 때, 학습 방법을 개선하거나, 알고리즘을 개선하거나, 더 많은 데이터를 획득하는 등 다양한 방법들을 고려할 수 있다. 공학 설계 문제에서 생각해볼 수 있는 문제 중 가장 치명적인 것은 데이터 수의 부족이다. 자주 인용되고 사용되는 에어포일 데이터베이스 사이트들 조차도, 전체 데이터의 수는 1,600여개에 불과하다.[1][2] 이미지 학습 분야에서는 수천만개의 이미지를 수만가지의 카테고리로 분류하고 [36], 학습에는 백만개의 이미지를 사용하는 것 [16]과 비교할 때, 1,600여개는 그 수가 매우 적다. 그러므로 본 연구에서는 <그림 3-42>와 같은 데이터베이스 강화 루프를 통해, 데이터 수가 절대적으로 부족할 수밖에 없었던 공학 설계 분야에서 자동적으로 데이터베이스를 강화하고 학습할 수 있도록 하였다.

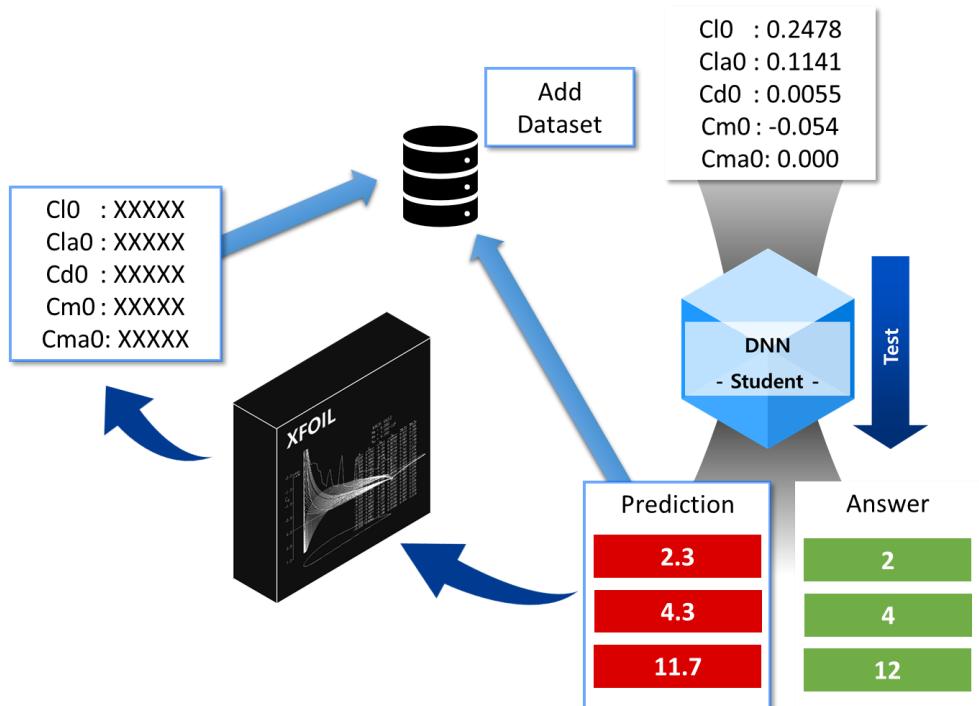


<그림 3-42> 데이터베이스 강화 루프 개념도

데이터베이스 강화 루프를 실행하기 위해서는 두 가지가 필요하다. 하나는 인공신경망의 테스트 결과이며, 다른 하나는 테스트 결과로 나온 형상들의 성능을 해석해 줄 해석 도구이다. 인공신경망이 예측한 형상을 항상

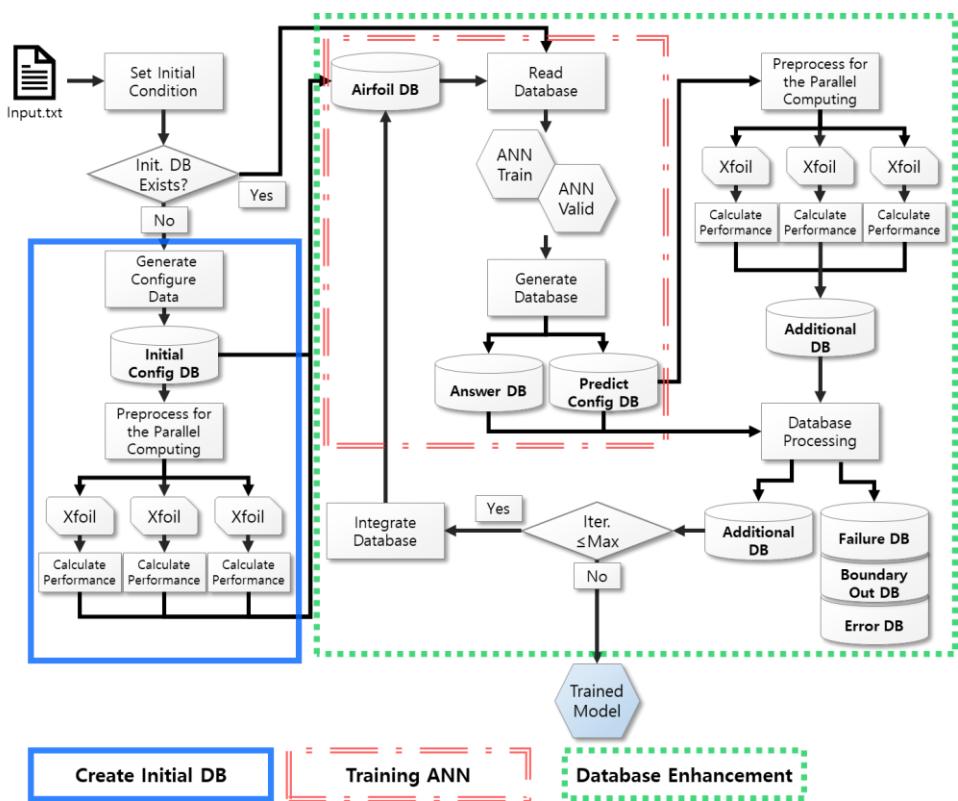
정답 형상과 동일하지 않다. 그 오차는 매우 커서 전혀 다른 형상을 예측했을 수도 있고, 오차가 매우 작아 아주 유사한 형상을 예측하였을 수도 있다. 데이터베이스 강화 루프는 이처럼 맞을 수도, 틀릴 수도 있는 예측 형상들을 한번 더 해석해 줌으로서, 새로운 데이터 셋을 만들어내 준다.

본 연구에서는 초기 데이터 획득에 사용한 XFOIL [5]을 데이터베이스 강화를 위한 해석 도구로 사용하였으며, 초기 하이퍼 파라미터 최적화를 통해 구성한 인공신경망 구조를 이용해 학습할 인공지능 모델을 구성하였다. <그림 3-42>는 데이터베이스 강화루프가 일어나는 과정을 쉽게 설명하기 위해 단순화한 개념도다. 인공신경망이 학습을 마친 뒤 테스트를 하면, XFOIL이 이를 채점하고 정확한 답을 알려준다. 정확한 답과 신경망의 예측 결과는 이전에 없던 새로운 데이터셋이 되는데, 이를 기준 데이터베이스에 추가하여 에어포일 학습 데이터를 늘려 나간다.



<그림 3-43> 데이터베이스 강화 예시

예를 들어 NACA 2412를 예측해야 하는 문제를 정확하게 푼다면, 최대 캠버 2, 최대 캠버 위치 4, 최대 두께 12를 예측해야 한다. 그러나 예측한 값이 각각 2.3, 4.3, 11.7이라면 이는 잘못 예측한 것이다. 특히 초기 데이터셋은 정수형 NACA 에어포일로만 이루어져 있으므로, 해당 에어포일에 대한 데이터는 존재하지 않는다. 일반적인 인공신경망의 학습 과정은 0.1, 0.2, 0.5라는 오차가 작아지도록 한다. 반면, 데이터베이스 강화 루프는 이전 데이터셋에 없던 NACA(2.3)(4.3)(11.7)이라는 에어포일을 해석하고, 이에 대한 형상-성능 학습 데이터셋을 만들어 이전 데이터베이스에 추가를 하는 것이며, 데이터 수는 1,496개에서 하나가 늘어난 1,497개가 된다. 본 연구에서는 테스트 데이터를 전체 데이터의 25%를 할당하였으므로, 한번의 데이터베이스 강화 루프마다 에어포일 형상-성능 데이터 수가 25%씩 기하급수적으로 늘어날 것이다.

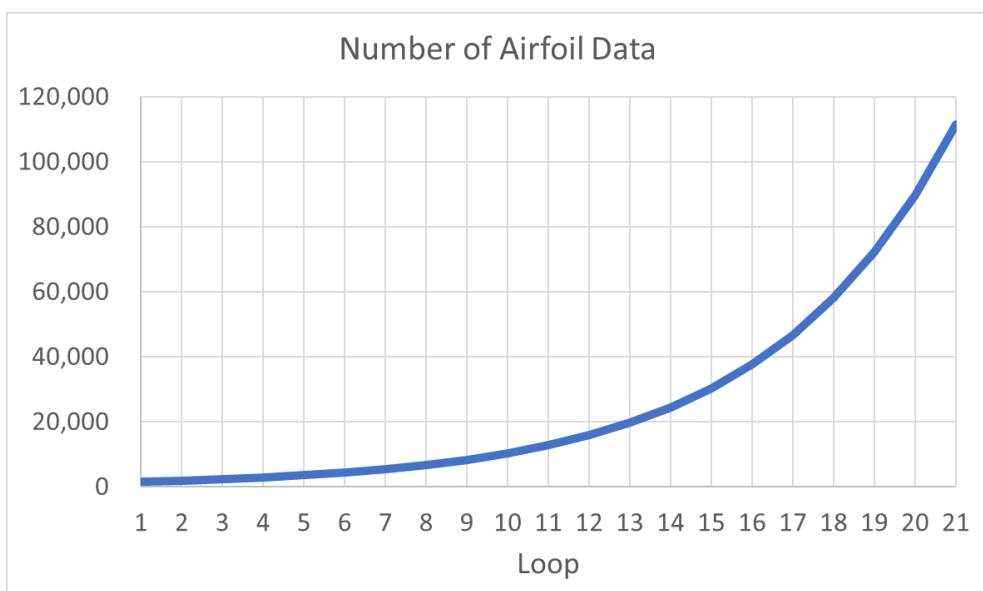


<그림 3-44> DLED Training Process

<그림 3-44>는 데이터베이스 강화 루프가 포함된 DLED의 Training 알고리즘이다. Initial DB를 생성하고 알고리즘 학습을 시키며, 학습 결과로 데이터 수를 늘려 DB를 강화한다. 본 연구에서는 강화 루프가 20회 실시되었다.

A. 데이터베이스 강화 루프 결과

<그림 3-45>는 에어포일 데이터 수의 증가 양상을 나타낸 그래프이며, <표 3-34>는 데이터 수와 그 증가량, 그리고 이상적으로 25%씩 증가하였을 때 데이터 수를 나타낸 것이다. <그림 3-45>를 보면 데이터 수는 기하급수적으로 늘어나는 것처럼 보이지만, 실제 증가량은 23.6%에서 24.4%까지 다양하다. 이는 해석이 불가능하거나, 해석 데이터가 충분치 않거나, 데이터 경계조건을 넘어서는 데이터들이 있어, DB에 추가되지 못한 에어포일 형상들이 존재하기 때문이다. 본 연구에서는 형상 파라미터가 최대 캠버와 최대 캠버의 위치가 0 미만이거나 10 이상인 경우, 그리고 두께가 0 이하인 경우를 제외하도록 하였다.

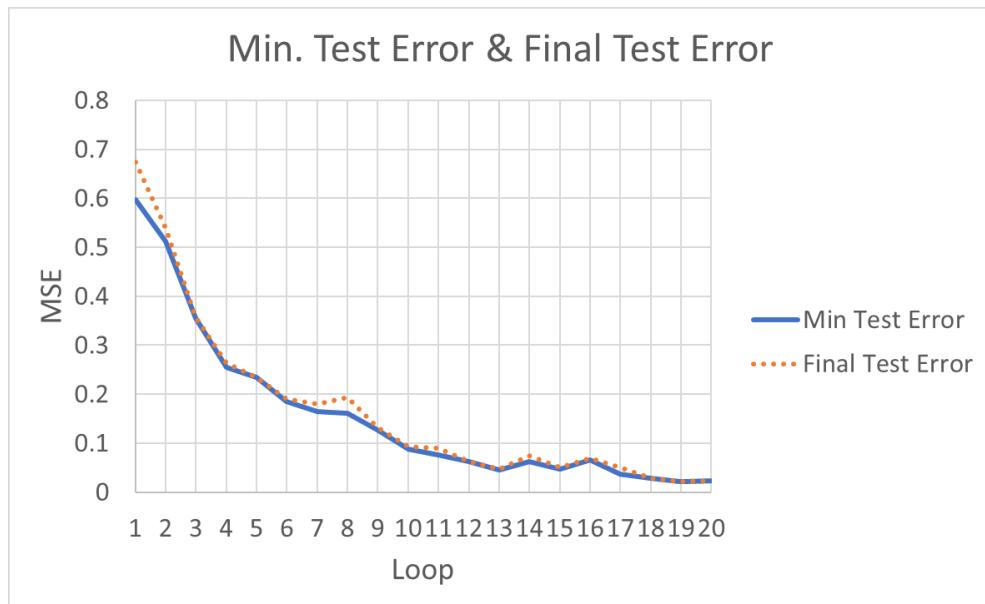


<그림 3-45> Database Enhancement Loop로 증가한 에어포일 데이터 수

<표 3-34> Database Enhancement Loop로 증가한 에어포일 데이터 증가량

Loop	Number of Airfoil Data (a)	Increasement	Ideal Number of Airfoil Data (b)	(b)-(a)
1	1,496	-	-	
2	1,849	23.6%	1,870	21
3	2,286	23.6%	2,311	25
4	2,827	23.7%	2,858	31
5	3,499	23.8%	3,534	35
6	4,341	24.1%	4,374	33
7	5,372	23.8%	5,426	54
8	6,658	23.9%	6,715	57
9	8,256	24.0%	8,323	67
10	10,260	24.3%	10,320	60
11	12,748	24.2%	12,825	77
12	15,805	24.0%	15,935	130
13	19,616	24.1%	19,756	140
14	24,348	24.1%	24,520	172
15	30,238	24.2%	30,435	197
16	37,542	24.2%	37,798	256
17	46,676	24.3%	46,928	252
18	58,016	24.3%	58,345	329
19	72,151	24.4%	72,520	369
20	89,702	24.3%	90,189	487
21	111,493	24.3%	112,128	635

<그림 3-46>은 각 루프 별 최소 테스트 오차와 최종 테스트 오차를 나타낸 것이다. 초기 하이퍼 파라미터 최적화 과정에서 0.5 아래로 내려가지 않았던 MSE가 10회 Loop에서부터 0.1 아래로 내려간 것을 볼 수 있다.



<그림 3-46> DEL: 최소 테스트 오차와 최종 테스트 오차

<표 3-35>는 최소 테스트 오차와 최소 테스트 오차가 나타난 Epoch, 그리고 최종 테스트 오차를 기술한 것이다. 각 루프에서의 학습 양상은 0에 첨부하였다. <표 3-17>과 <표 3-30>에서 볼 수 있듯이, 초기 하이퍼파라미터 최적화 전후 모두에서 최소 테스트 오차가 나타난 Epoch는 대부분 20,000번 보다 낮았다. 그러나 <표 3-35>를 보면, 대부분 10,000 이상에서 최소 테스트 오차가 나타났으며, 최종 테스트 오차와 최소 테스트 오차가 같은 경우도 두 번 나타났다. 이는 학습에 더 많은 Epoch가 필요한 것으로 추측할 수 있다.

<표 3-35> DEL: 최소 테스트 오차와 최종 테스트 오차

그리고 최종 테스트 오차의 Epoch

Loop	Min Test Error	Epoch of Min Test Error	Final Test Error
1	0.59660	2,400	0.67359
2	0.51216	18,200	0.53951
3	0.35483	18,300	0.35763

4	0.25423	18,500	0.26344
5	0.23411	20,000	0.23411
6	0.18465	13,500	0.18962
7	0.16400	11,800	0.18021
8	0.16189	11,600	0.19382
9	0.12642	15,100	0.13277
10	0.08855	17,600	0.09348
11	0.07687	12,000	0.09004
12	0.06165	20,000	0.06165
13	0.04545	14,000	0.04648
14	0.06270	3,700	0.07371
15	0.04698	7,500	0.05079
16	0.06518	900	0.06971
17	0.03760	13,300	0.05020
18	0.02773	14,800	0.02904
19	0.02153	19,600	0.02241
20	0.02306	11,900	0.02405

<그림 3-47>과 <표 3-36>은 데이터베이스 강화 루프에서 각 루프 별 인공신경망 학습에 소요된 시간과, 테스트 결과를 XFOIL로 해석하는데 걸린 시간을 나타낸 것이다. XFOIL 해석 시간은 <그림 3-45>의 데이터 증가량 그래프처럼 증가하는 것을 알 수 있다. 데이터베이스의 데이터 증가량의 평균은 24.1%이며, 소요시간의 증가량의 평균도 24.16%로 매우 근접한 결과를 보이며, Loop1의 재해석 시간에 데이터 증가량을 이용하여 20번째 Loop의 재해석 시간을 예측하여도 실제 소요시간인 250.87분에 근사한 250.69분이 나온다.

인공신경망 학습 시간의 경우, Loop 11, 13, 17, 18에서 다른 경향을 보이는 데, 이는 내부 알고리즘이 10,000번째 Epoch에서 학습에 진전이 없으면 가중치와 바이어스를 초기화하고 다시 학습하도록 하였기 때문이다.



<그림 3-47> DEL: 소요 시간 비교

학습 기록을 살펴보면 Loop 11, 13, 17, 18은 10,000번의 학습 후, 학습 진도가 적정 수준에 미치지 못하자 인공신경망을 초기화 한 뒤 20,000번의 학습을 실시하였으므로, 총 30,000번의 학습이 일어난 것으로 볼 수 있다. <그림 3-47>의 점선(Training Time(x 2/3))은 재 학습이 일어난 Loop의 학습 소요 시간에 2/3을 곱하여 이를 보정한 결과를 나타낸 것이다. 이는 원본 데이터보다 추세를 잘 나타내고 있는 것을 볼 수 있다.

<표 3-36> 학습 소요 시간과 테스트 결과의 XFOIL 해석 시간,
그리고 보정된 학습 소요 시간

Loop	Training Time	XFOIL Analysis Time	Training Time (x2/3 for *)
1	3.94	4.36	3.94
2	4.28	5.86	4.28
3	4.70	7.51	4.70

4	5.26	8.79	5.26
5	5.88	9.99	5.88
6	6.88	14.97	6.88
7	8.22	15.02	8.22
8	9.73	20.66	9.73
9(*)	13.19	24.53	13.19
10	15.30	28.40	15.30
11(*)	27.29	36.75	18.19
12	23.11	45.95	23.11
13	41.45	56.54	27.63
14	34.64	68.83	34.64
15	42.69	85.25	42.69
16	52.84	105.25	52.84
17(*)	100.60	132.28	67.07
18(*)	121.05	163.68	80.70
19	100.21	201.69	100.21
20	124.24	250.87	124.24

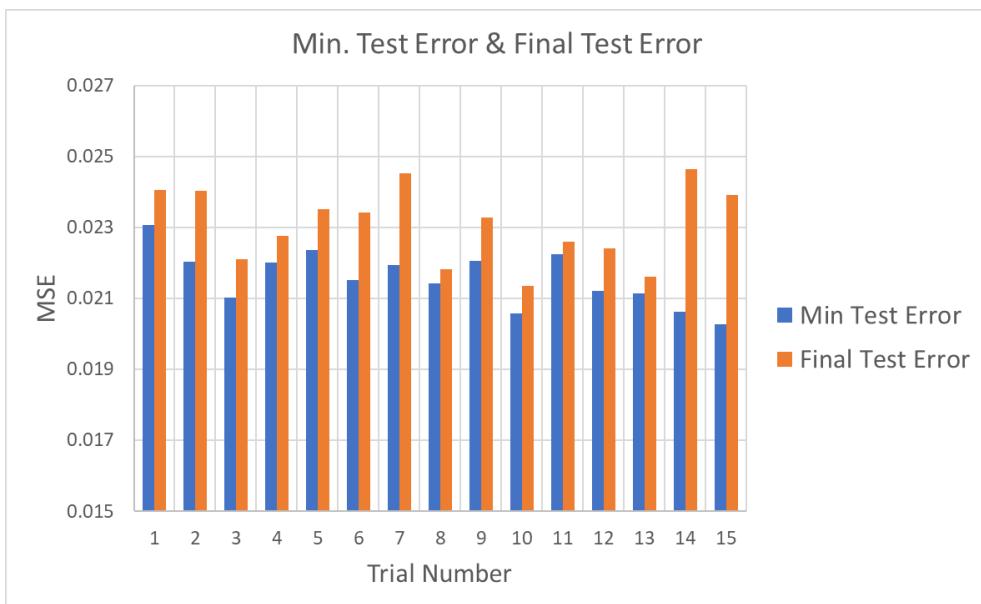
B. 데이터베이스 강화 루프 후 학습 편차 확인

본 단락에서는 앞서 초기 하이퍼 파라미터 최적화(IHO) 단계에서 진행한 반복 학습을 통한 학습 편차 확인을 실시하였다. 마지막 Loop인 20번째 데이터베이스로 IHO 단계와 마찬가지로 15개의 모델을 만들어 학습시켰으며, <표 3-37>과 <그림 3-48>의 결과를 얻었다.

<표 3-37> DLE Loop 20: 15회 학습 Test Error 및 최소 Test Error Epoch 결과

Trial	Min. Test Error	Epoch of Min. Test Error	Final Test Error
1	0.02306	11,900	0.02405
2	0.02203	2,800	0.02403

3	0.02102	17,900	0.02210
4	0.02200	7,900	0.02277
5	0.02236	15,600	0.02351
6	0.02152	3,300	0.02341
7	0.02194	15,200	0.02453
8	0.02143	10,300	0.02183
9	0.02206	19,000	0.02328
10	0.02058	17,400	0.02135
11	0.02224	13,700	0.02260
12	0.02121	17,400	0.02241
13	0.02113	8,600	0.02162
14	0.02061	11,200	0.02463
15	0.02027	10,400	0.02390



<그림 3-48> DEL Loop 20: 15회 학습 Test Error/최소 Test Error Epoch 결과

초기 하이퍼 파라미터 최적화(IHO) 과정과 마찬가지로, 평균과 표준편차를 구하고 정규분포와 표준 정규분포도를 그렸다. <표 3-38>은 데이터베

이스 강화 전(Loop 1)과 후(Loop 20)의 최소 테스트 오차의 최소, 최대, 평균값과 표준편차 값을 비교한 것이다. <그림 3-49>와 <그림 3-50>은 최소 테스트 오차의 정규분포와 표준 정규분포를 나타낸 것이다. <표 3-39>는 데이터베이스 강화 전(Loop 1)과 후(Loop 20)의 최종 테스트 오차(at Epoch 20,000)의 최소값, 평균값, 최대값과 표준편차를 비교한 것이며, <그림 3-51>과 <그림 3-52>는 데이터베이스 강화 Loop 20 최종 테스트 오차의 정규분포도와 표준 정규분포도다. IHO와 마찬가지로, 정규분포와 표준정규분포 모두 15번의 학습 데이터만으로도 충분히 정규화 분포를 나타낼 수 있으며, 예시 표준정규분포 역시 잘 따름을 알 수 있다.

<표 3-38>과 <표 3-39>에서 볼 수 있듯이, 학습 오차의 분포가 전반적으로 작아졌는데, 줄어든 수치가 약 95~97%에 달한다. 이는 제한된 데이터의 수로는 학습 오차를 줄이는데 한계가 있으며, 인공신경망을 잘 학습시키기 위해서는 하이퍼 파라미터 최적화보다 충분한 데이터를 확보할 방법이 필요하다는 것을 알 수 있다.

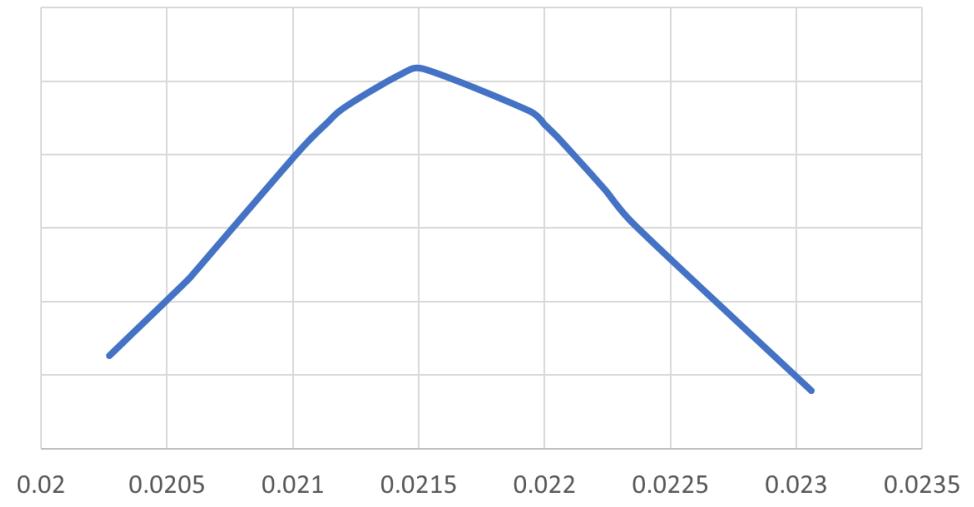
<표 3-38> 데이터베이스 강화 전(Loop 1)과 후(Loop 20)의 최소 테스트 오차의 최소값, 평균값, 최대값과 표준편차 비교

	Min.	Average	Max.	σ
Loop 1	0.56734	0.60554	0.65651	0.02764
Loop 20	0.02027	0.02156	0.02306	0.00077
Difference	-0.54707	-0.58398	-0.63345	-0.02687

<표 3-39> 데이터베이스 강화 전(Loop 1)과 후(Loop 20)의 최종 테스트 오차의 최소값, 평균값, 최대값과 표준편차 비교

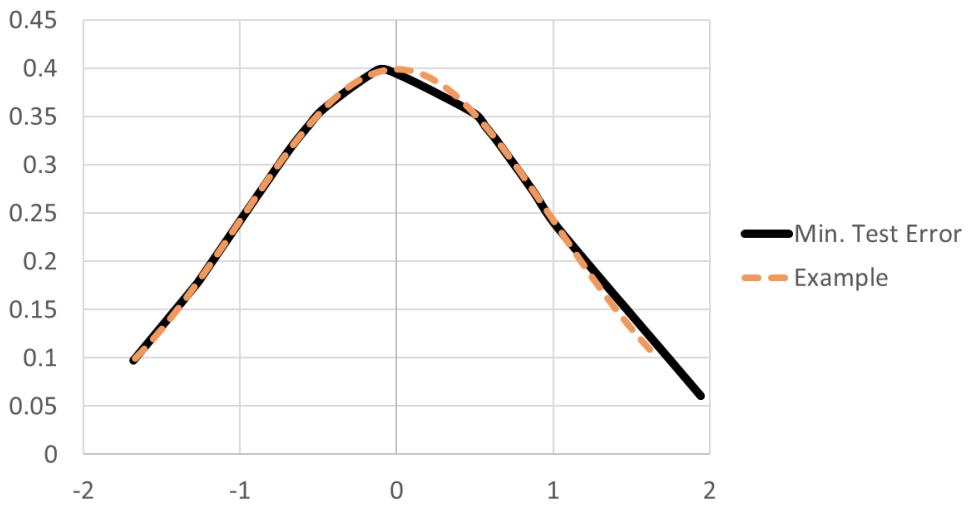
	Min.	Average	Max.	σ
Loop 1	0.61797	0.68125	0.71308	0.02462
Loop 20	0.02135	0.02307	0.02468	0.00103
Difference	-0.59662	-0.65818	-0.68840	-0.02359

Normal Distribution: Min. Test Error



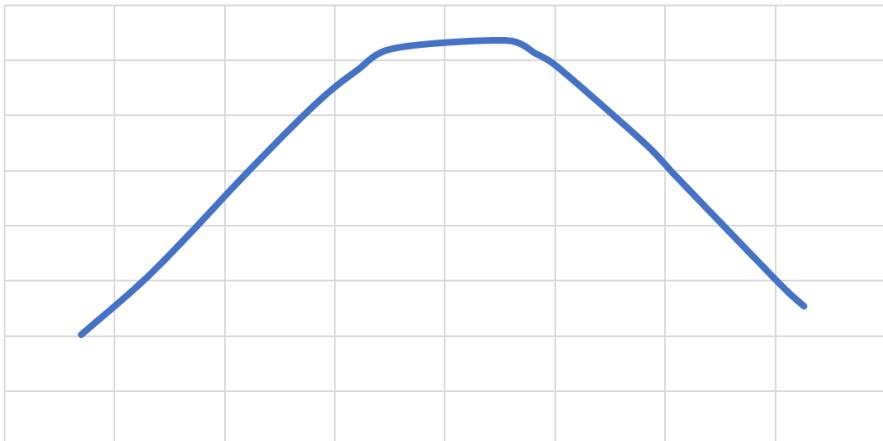
<그림 3-49> Loop 20의 최소 테스트 오차 정규분포도

Standard Normal Distribution: Min. Test Error



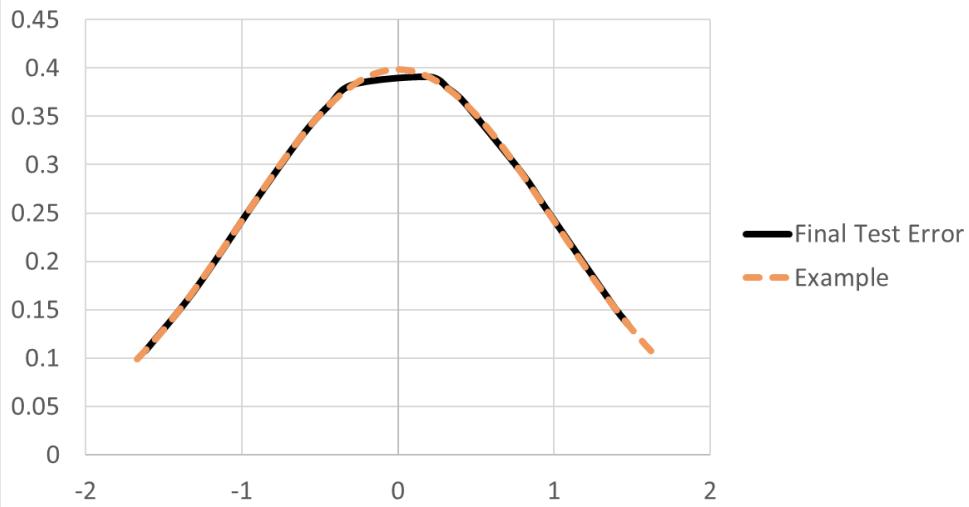
<그림 3-50> Loop 20의 최소 테스트 오차 표준 정규분포도

Normal Distribution: Final Test Error



<그림 3-51> Loop 20의 최종 테스트 오차 정규분포도

Standard Normal Distribution: Final Test Error



<그림 3-52> Loop 20의 최종 테스트 오차 표준 정규분포도

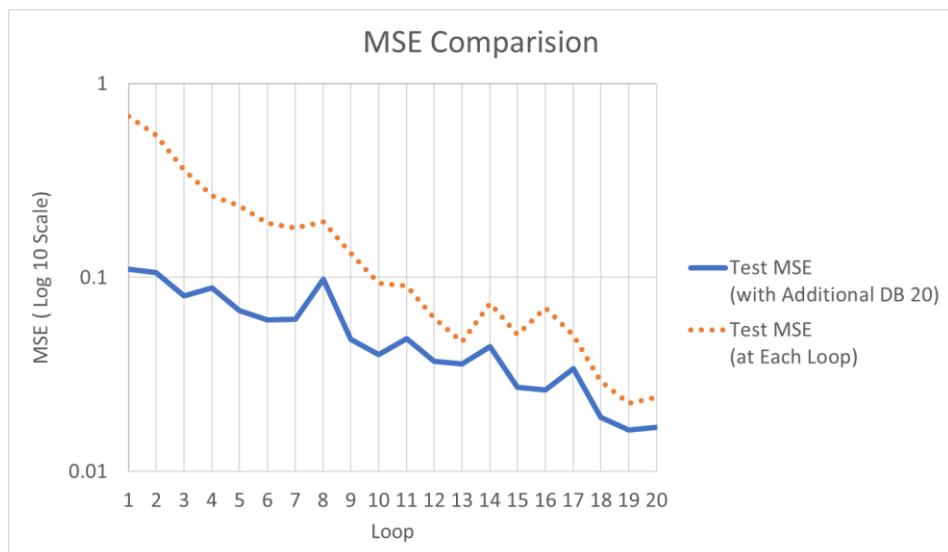
C. 동일한 테스트 DB를 이용한 Loop별 인공신경망 오차 확인

데이터베이스 강화 루프를 통한 오차의 감소는 데이터 수가 특별히 오차가 적은 형상이 오차가 큰 형상에 비해 매우 많아져서 일어나는 것일 수도 있다. 이를 확인하기 위해 Loop 20의 테스트 결과로 만든 데이터 (Additional DB 20)로 각 Loop의 인공신경망 모델의 오차를 하였다.

<표 3-40> 각 Loop의 인공신경망 모델 테스트 결과

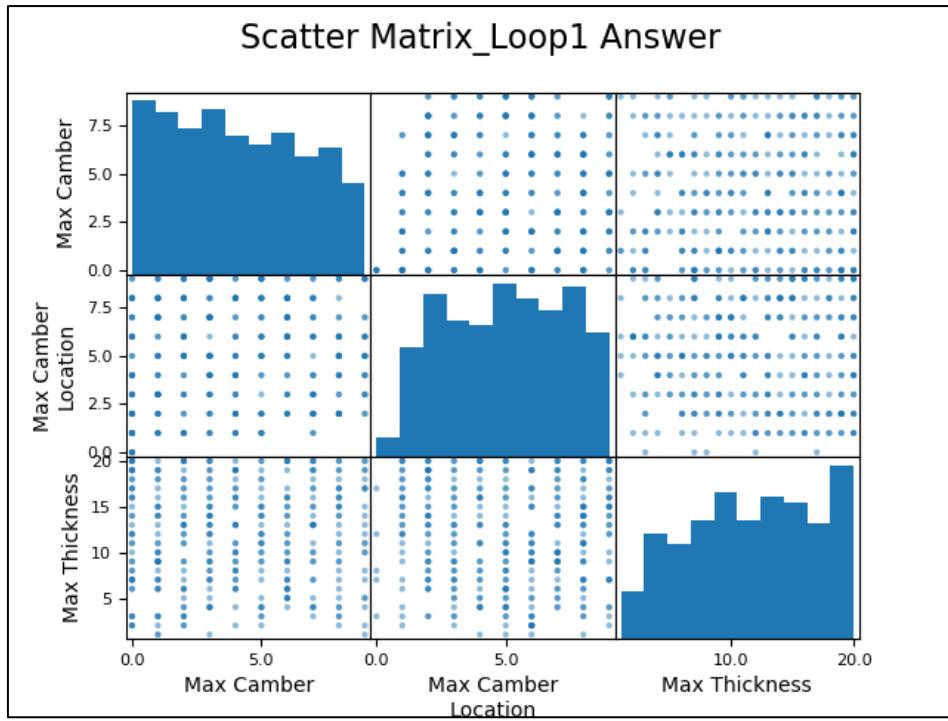
Model Number (Loop)	Test MSE (with Additional DB 20)	Test MSE (at Each Loop)
1	0.109935	0.67359
2	0.105609	0.53951
3	0.08037	0.35763
4	0.087922	0.26344
5	0.067213	0.23411
6	0.060477	0.18962
7	0.060692	0.18021
8	0.098018	0.19382
9	0.047916	0.13277
10	0.040002	0.09348
11	0.048251	0.09004
12	0.036951	0.06165
13	0.035656	0.04648
14	0.043956	0.07371
15	0.027017	0.05079
16	0.026251	0.06971
17	0.033893	0.0502
18	0.018951	0.02904
19	0.016331	0.02241
20	0.016911	0.02405

<표 3-40>은 Additional DB 20의 데이터를 동일하게 이용하여 테스트한 결과와, 앞서 각 데이터베이스 강화 루프를 진행하면서 테스트한 결과를 비교한 것이다. Additional DB 20은 Loop 20에서 인공신경망을 테스트한 형상들을 XFOIL로 다시 해석하여 만든 에어포일 데이터베이스이다. 데이터베이스 강화루프 오차는 0.67359에서 0.02405로 약 96% 감소하였으나, 동일한 DB(Additional DB 20)을 사용하여 테스트한 결과는 0.109935에서 0.016911로 약 85% 감소한 것을 확인할 수 있다.

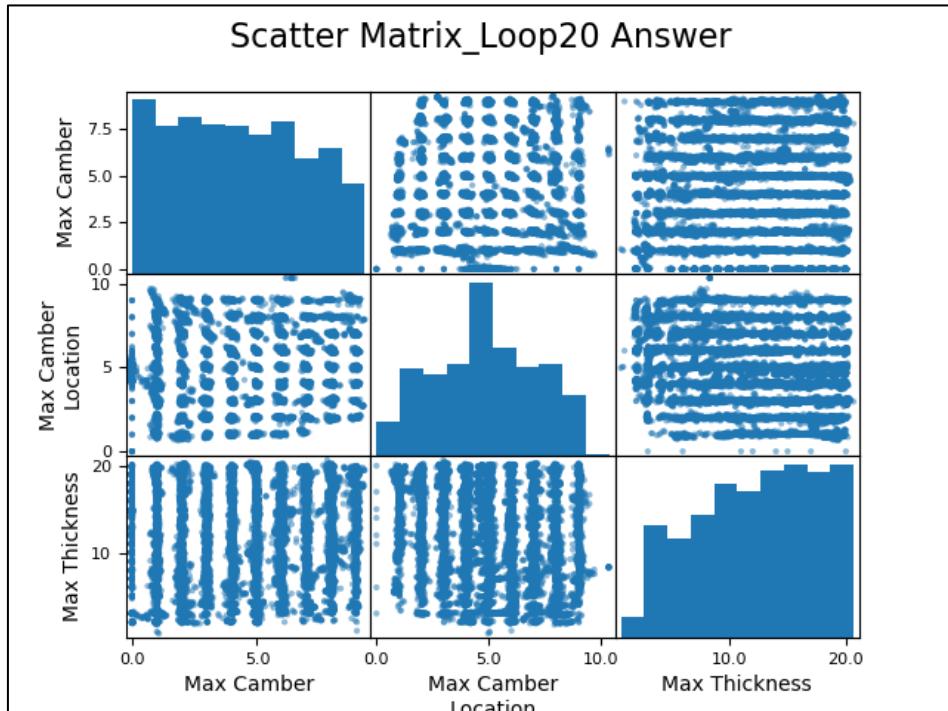


<그림 3-53> 각 Loop의 인공신경망 모델 테스트 결과

<그림 3-53>은 <표 3-40>을 그래프로 나타낸 것이다. 동일한 DB로 테스트 한 결과가 각 루프에서 테스트 한 결과보다 낮은 오차를 보이는 것을 확인할 수 있으며, 테스트 오차가 감소하는 양상이 비슷한 것을 볼 수 있다. 데이터베이스 강화의 효과는 있으나, 전반적으로 오차가 작다는 것은 데이터가 학습이 잘 되는 구간의 비율이 많아지는 것이라 예측할 수 있다. 데이터 분포의 변화를 살펴보기 위해 <그림 3-54>, <그림 3-55>와 같이 테스트에 사용한 데이터의 정답 데이터를 Scatter matrix로 나타냈다. 본 단락에는 Loop 1과 Loop 20의 Scatter matrix만 첨부하였으며, 전체 Loop의 Scatter matrix는 Appendix B에 정리하였다.



<그림 3-54> DEL Loop 1 정답 데이터의 Scatter Matrix



<그림 3-55> DEL Loop 20 정답 데이터의 Scatter Matrix

Scatter matrix는 대각선으로는 각 파라미터들의 빈도수를 나타내는 히스토그램이 있고, 나머지 영역에는 두 파라미터들의 분포도가 그려져 있다.

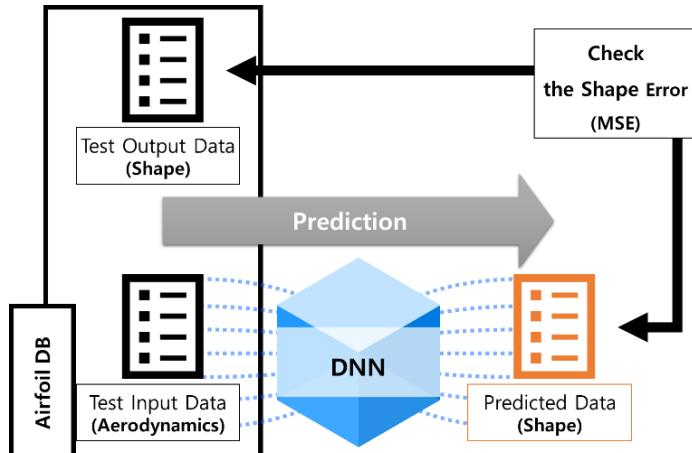
<그림 3-54>, <그림 3-55>, 그리고 Appendix B를 보면 데이터의 증가 양상을 볼 수 있다. 세 가지 형상 파라미터 중 최대 캠버의 분포는 큰 변화가 없지만, 최대 캠버 위치는 5 근처 값의 빈도가 크게 증가하였으며 최대 두께는 19~20 이하 값들의 빈도가 크게 증가하였음을 볼 수 있다.

앞서 제 1절 단락 마지막에 [4]의 결론 부분을 정리하였는데, 이에 따르면 NACA 4-digit의 일반적으로 사용하는 에어포일 형상의 파라미터 범위를 알 수 있다. [4]에 따르면 일반적으로 사용하는 NACA 4-digit 에어포일은 3~5 사이의 최대 캠버 위치값과 9~15 사이의 최대 두께를 갖는다. 이를 Scatter matrix의 변화와 비교하여 보면, 데이터베이스 강화 루프가 진행될수록 학습 데이터에는 일반적인 NACA 4-digit 데이터가 많이 추가되었음을 알 수 있다.

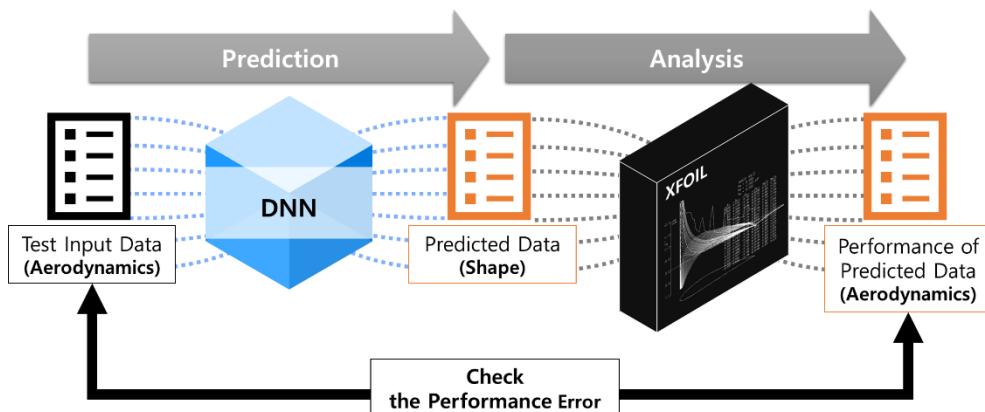
D. 예측 형상의 성능 오차 확인

공학적 설계의 궁극적인 목적은 원하는 성능의 형상을 얻어내는 것이다. 다르게 말하면, 설계한 형상이 원하는 성능을 내야 한다. 인공신경망의 학습에 사용되는 오차는 출력의 오차이다. 본 연구에서 사용한 DLED의 경우는 예측한 NACA 4-digit의 세 가지 형상 파라미터인 최대 캠버, 최대 캠버 위치, 그리고 최대 두께 MSE 오차를 <그림 3-56>과 같이 측정하였다. 만약 학습 오차와 테스트 오차 모두가 0 이라면 인공신경망은 매우 이상적으로 학습을 마쳤고, 실 사용에서도 매우 이상적으로 정확하게 형상을 예측할 것이다. 그러나 이러한 이상적인 상황은 현실에서는 일어나기 매우 힘들며, 시간과 자원의 한계, 그리고 회귀 기법과 메타-모델링 기법의 특성으로 인해 학습 오차는 항상 존재하며, 테스트 오차는 이보다 크게 나타난다. 그러므로 본 연구에서는 <그림 3-57>과 같이, 예측한 형상을 XFOIL [5]을 이용하여 다시 한번 해석하여 입력 성능과 예측한 형상의 성능 사이의 오차를 구하였다. 데이터베이스 강화 루프의 기본적인 원리가

예측한 형상의 성능을 계산하여 이전 데이터베이스에 추가하는 것이므로, 오차계산에는 추가적인 해석이 필요하지 않다.



<그림 3-56> 인공신경망 학습 단계에서의 오차 측정 방법



<그림 3-57> 학습된 인공신경망의 예측 형상 성능 측정 방법

본 연구에서는 앞서 학습 편차를 살펴본 것처럼 성능 오차의 편차와 분포를 확인하였다. <표 3-41>과 <표 3-42>는 각 루프에서 실시한 테스트 결과들로 계산한 오차의 평균이며, <표 3-43>과 <표 3-44>는 테스트 오차의 표준편차이다. 본 단락에는 Loop 1과 Loop 20의 결과만 간략히 기술하였으며, 모든 데이터는 Appendix C.1과 Appendix C.2에 첨부하였다.

**<표 3-41> DEL: Loop에 따른 인공지능 테스트 오차 평균: 형상 파라미터
(Loop 1과 Loop 20)**

DEL Loop	Max Camber	Max Camber Location	Max Thickness
Loop 1	-0.00347680	0.116517	0.0167733
Loop 20	0.00923081	-0.00257035	-0.0229289

**<표 3-42> DEL: Loop에 따른 인공지능 테스트 오차 평균: 성능 파라미터
(Loop 1과 Loop 20)**

DEL Loop	Cl0	Cla0	Cd0	Cm0	Cma0
Loop 1	0.000110	-4.3E-04	-5.09E-05	-2.27E-05	1.3E-04
Loop 20	0.001237	-8.62E-06	5.72E-06	-2.40E-04	2.09E-06

**<표 3-43> DEL: Loop에 따른 인공지능 테스트 오차 표준편차: 형상 파라미터
(Loop 1과 Loop 20)**

DEL Loop	Max Camber	Max Camber Location	Max Thickness
Loop 1	0.132697	1.20072	0.733579
Loop 20	0.0240654	0.141308	0.189559

**<표 3-44> DEL: Loop에 따른 인공지능 테스트 오차 표준편차: 성능 파라미터
(Loop 1과 Loop 20)**

DEL Loop	Cl0	Cla0	Cd0	Cm0	Cma0
Loop 1	0.01914	0.011536	0.000986	0.004500	0.002754
Loop 20	0.004137	0.010011	0.000370	0.001001	0.002239

1) 3σ , 6σ 오차

정규분포에서 표준편자는 데이터가 얼마나 흩어져 있는지를 나타내는 기준이다. 표준편자가 크면 데이터는 평균을 기준으로 넓게 분포할 것이며, 표준편자가 작다면 데이터는 평균 주변에 모여 있음을 예측할 수 있다.

또한 통계학에서는 표준편차를 이용하여 임의의 데이터의 분포 확률을 계산하는데 사용할 수 있다. 잘 알려진 품질 경영 기법인 6σ 는 생산한 제품들 중, 양품을 생산할 확률을 6σ 이내까지 끌어올리겠다는 것이다. 반대로 말하면 불량률을 6σ 바깥으로 내몰겠다는 뜻으로, 백만개의 제품을 생산할 때 불량품을 0.002개 내로 하겠다는 것이다. [58]

시그마에 따라 기대할 수 있는 데이터의 분포 확률은 아래 수식과 같이 표현할 수 있다. 데이터가 1 시그마 이내에 있을 확률은 68.27%이며, 2 시그마 이내에 있을 확률은 95.45%, 3 시그마 이내에 있을 확률은 99.73%이다. <표 3-45>는 시그마 범위에 따라 데이터가 밖에 있을 확률과 안에 있을 확률을 나타낸 것이다. [59]

$$Pr(\mu - 1\sigma \leq X \leq \mu + 1\sigma) \approx 0.6827 \quad (81)$$

$$Pr(\mu - 2\sigma \leq X \leq \mu + 2\sigma) \approx 0.9545 \quad (82)$$

$$Pr(\mu - 3\sigma \leq X \leq \mu + 3\sigma) \approx 0.9973 \quad (83)$$

- μ : 평균
- σ : 표준편차

<표 3-45> σ 에 따른 내부 범위에 있을 확률과 바깥 범위에 있을 확률 [59]

σ	Probability of Inside Range	Frequency of Outside Range
$\mu \pm 1\sigma$	0.682 689 492	1/3
$\mu \pm 2\sigma$	0.954 499 736	1/22
$\mu \pm 3\sigma$	0.997 300 203	1/370
$\mu \pm 4\sigma$	0.999 936 657	1/15,787
$\mu \pm 5\sigma$	0.999 999 426	1/1,744,278
$\mu \pm 6\sigma$	0.999 999 998	1/506,797,346

본 연구에서는 각기 다른 수의 테스트 데이터들의 오차를 비교하기 위해, 각 Loop에 따른 3σ 와 6σ 의 변화를 확인하였다. 이는 임의의 테스트 데이터를 넣었을 때, 99.730 %의 확률과(3σ), 99.999 999 8 %의 확률로 (평균 ± 3 or 6σ) 범위 내에 존재한다는 것이다. 전체 Loop별 3σ 와 6σ , 그리고 각각의 범위는 Appendix C.3에 정리하였으며, <표 3-46>~<표 3-53>에는 Loop 1과 Loop 20의 값들만 간략히 정리하여 기술하였다.

**<표 3-46> DEL: Loop에 따른 인공지능 테스트 오차 3σ : 형상 파라미터
(Loop 1과 Loop 20)**

DEL Loop	Max Camber	Max Camber Location	Max Thickness
Loop 1	0.398091	3.602160	2.200737
Loop 20	0.072196	0.423924	0.568677

**<표 3-47> DEL: Loop에 따른 인공지능 테스트 오차 3σ : 성능 파라미터
(Loop 1과 Loop 20)**

DEL Loop	C _{l0}	C _{la0}	C _{d0}	C _{m0}	C _{ma0}
Loop 1	0.057421	0.034607	0.002958	0.013499	0.008263
Loop 20	0.012411	0.030034	0.001109	0.003003	0.006717

**<표 3-48> DEL: Loop에 따른 인공지능 테스트 오차 ($\mu \pm 3\sigma$): 형상 파라미터
(Loop 1과 Loop 20)**

DEL Loop		Max Camber	Max Camber Location	Max Thickness
Loop 1	max	0.3946142	3.7186770	2.2175103
	min	-0.4015678	-3.4856430	-2.1839637
Loop 20	max	0.0814270	0.4213537	0.5457481
	min	-0.0629654	-0.4264944	-0.5916059

<표 3-49> DEL: Loop에 따른 인공지능 테스트 오차 ($\mu \pm 3\sigma$): 성능 파라미터
(Loop 1과 Loop 20)

DEL Loop		Cl0	Cla0	Cd0	Cm0	Cma0
Loop 1	max	0.057530	0.034177	0.002907	0.013477	0.008393
	min	-0.057311	-0.035037	-0.003009	-0.013522	-0.008133
Loop 20	max	0.013648	0.030026	0.001115	0.002765	0.006719
	min	-0.011174	-0.030043	-0.001103	-0.003241	-0.006715

<표 3-50> DEL: Loop에 따른 인공지능 테스트 오차 6σ : 형상 파라미터
(Loop 1과 Loop 20)

DEL Loop	Max Camber	Max Camber Location	Max Thickness
Loop 1	0.796182	7.20432	4.401474
Loop 20	0.1443924	0.847848	1.137354

<표 3-51> DEL: Loop에 따른 인공지능 테스트 오차 6σ : 성능 파라미터
(Loop 1과 Loop 20)

DEL Loop	Cl0	Cla0	Cd0	Cm0	Cma0
Loop 1	0.114841	0.069215	0.005916	0.026998	0.016526
Loop 20	0.024822	0.060068	0.002218	0.006005	0.013434

<표 3-52> DEL: Loop에 따른 인공지능 테스트 오차 ($\mu \pm 6\sigma$): 형상 파라미터
(Loop 1과 Loop 20)

DEL Loop		Max Camber	Max Camber Location	Max Thickness
Loop 1	max	0.7927052	7.3208370	4.4182473
	min	-0.7996588	-7.0878030	-4.3847007
Loop 20	max	0.1536232	0.8452777	1.1144251
	min	-0.1351616	-0.8504184	-1.1602829

<표 3-53> DEL: Loop에 따른 인공지능 테스트 오차 ($\mu \pm 6\sigma$): 성능 파라미터
(Loop 1과 Loop 20)

DEL Loop		Cl0	Cla0	Cd0	Cm0	Cma0
Loop 1	max	0.114951	0.068785	0.005865	0.026976	0.016656
	min	-0.114732	-0.069645	-0.005967	-0.027021	-0.016396
Loop 20	max	0.026059	0.060060	0.002223	0.005767	0.013436
	min	-0.023585	-0.060077	-0.002212	-0.006243	-0.013432

2) 정리 및 분석

Appendix C.3와 <표 3-46>~<표 3-53>을 살펴보면, 데이터베이스 강화 루프를 통해 표준편차의 크기가 작아지는 것을 확인할 수 있으며, 특히, Cl0와 Cd0, Cm0는 매우 크게 개선되었다. 하지만, Appendix C.1의 평균 값과 그 그래프에서 볼 수 있듯이, Loop가 진행될수록 오차들의 평균이 커진 결과를 보여주었다.

E. 형상 범위 그룹에 따른 오차

NACA 4-digit 에어포일의 기술문서인 [4]을 보면 일반적으로 사용되는 NACA 4-digit 에어포일의 형상 파라미터 범위가 나와있다. XFOIL은 형상이 극단적인 모양새를 취할 경우, 데이터가 도출되지 않거나 풍동실험, 혹은 더 정밀한 CFD와 오차가 크게 나타나므로, 일반적이지 않은 형상의 데이터가 학습을 저하시킬 수 있다. 본 단락에서는 해당 범위에 속하는 에어포일들과 해당 범위 바깥의 에어포일들의 예측 오차를 정리하였다.

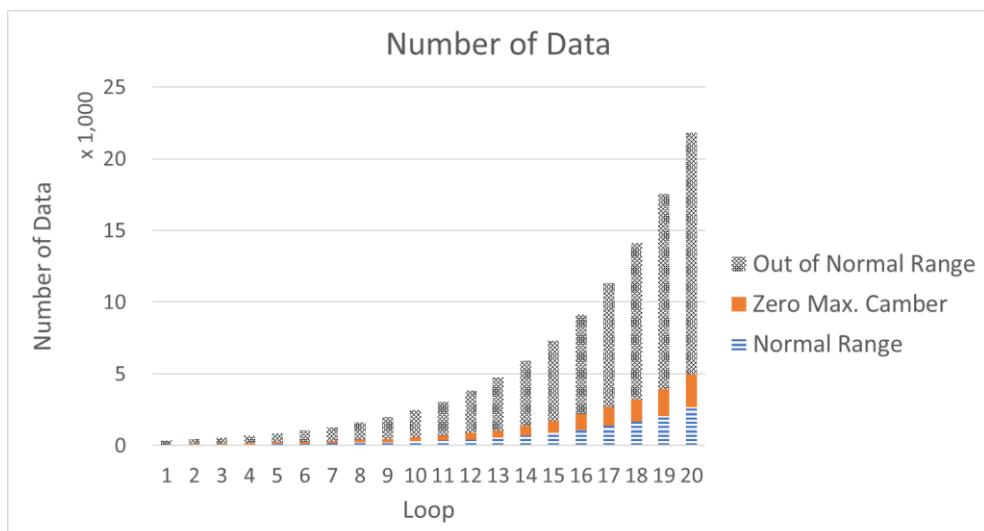
[4]에서 언급한 일반적인 형상 범위는 최대 캠버의 위치가 3~5, 최대 두께가 9~15인 에어포일이었다. 본 연구에서는 정수형 NACA 4-digit 에어포일뿐만 아니라 실수형 에어포일도 다루었으므로, 일반적인 형상의 범위를 <표 3-54>의 Normal Range와 같이 정하였다. 여기서 최대 캠버가 0이 아닌 비대칭형 에어포일만을 사용하였는데, 이는 <그림 3-1>과 그 아래의 식에서 나타나듯이, 최대 캠버가 0이면 최대 캠버의 위치가 어떤 숫자일지라도 영향을 미치지 못하기 때문이다. 그러므로 최대 캠버가 0인 에어

포일은 Zero Max. Camber로 분류하였다. 이 외의 범위에 속하는 에어포일은 Out of Normal Range로 분류하였다.

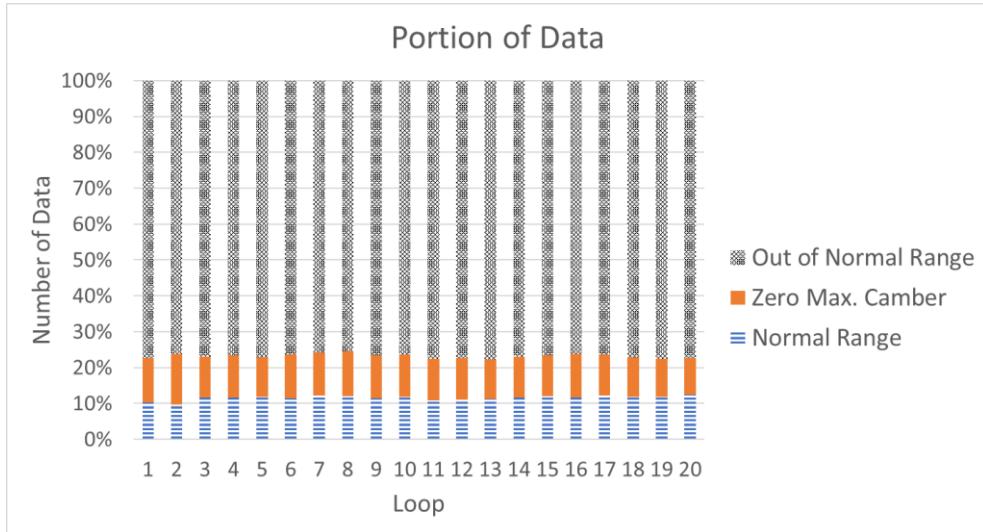
<표 3-54> 형상 범위 설정

	Max. Camber (C)	Location of Max. Camber(LC)	Max. Thickness (T)
Normal Range	$C \neq 0$	$2.5 \leq LC < 5.5$	$8.5 \leq T < 15.5$
Zero Max. Camber	$C = 0$	All Range	All Range
Out of Normal Range	$C \neq 0$	Out of Normal Range	Out of Normal Range

<그림 3-58>과 <표 3-55>는 형상 범위 그룹에 따른 데이터의 수를 나타낸 것이며, <그림 3-59>는 각 그룹이 전체 데이터에서 차지하는 비율을 나타낸 것이다. Loop가 진행될수록 데이터 수가 증가하나, 각각의 그룹의 비율은 일정 수준을 유지하는 것을 확인할 수 있다. Normal Range는 전체 데이터의 약 10~12%를 차지하고, Zero Max. Camber 그룹은 약 10~14%, Out of Normal Range 그룹은 약 75~78%를 차지한다.



<그림 3-58> 형상 범위 그룹에 따른 데이터 수



<그림 3-59> 형상 범위 그룹에 따라 데이터가 차지하는 비율

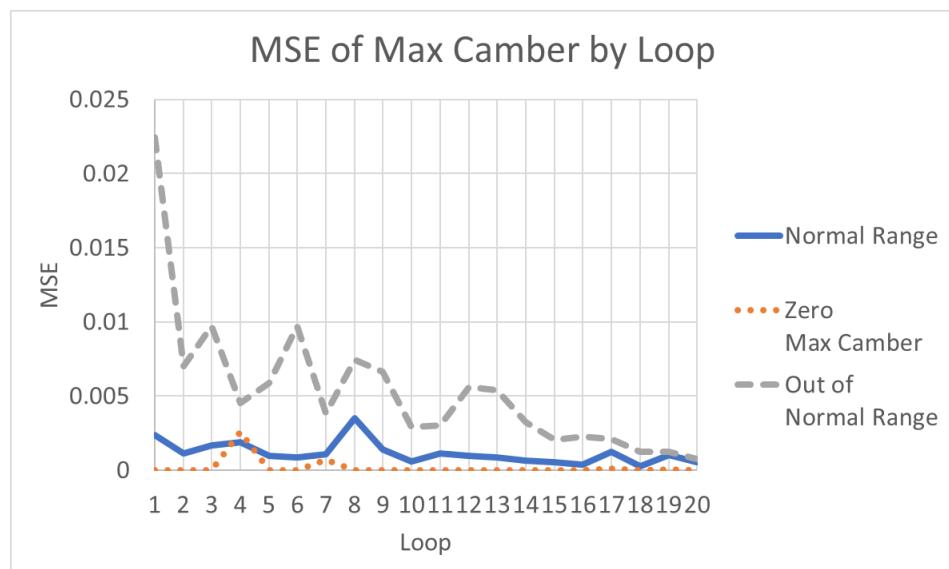
<표 3-55> 형상 범위에 따른 데이터 수

Loop	Normal Range	Zero Max. Camber	Out of Normal Range	Total
1	36	44	273	353
2	42	62	333	437
3	63	62	416	541
4	78	79	515	672
5	100	93	649	842
6	118	126	787	1,031
7	157	153	976	1,286
8	191	202	1,205	1,598
9	230	241	1,533	2,004
10	292	296	1,900	2,488
11	332	353	2,372	3,057
12	418	446	2,947	3,811
13	534	520	3,678	4,732
14	690	664	4,536	5,890

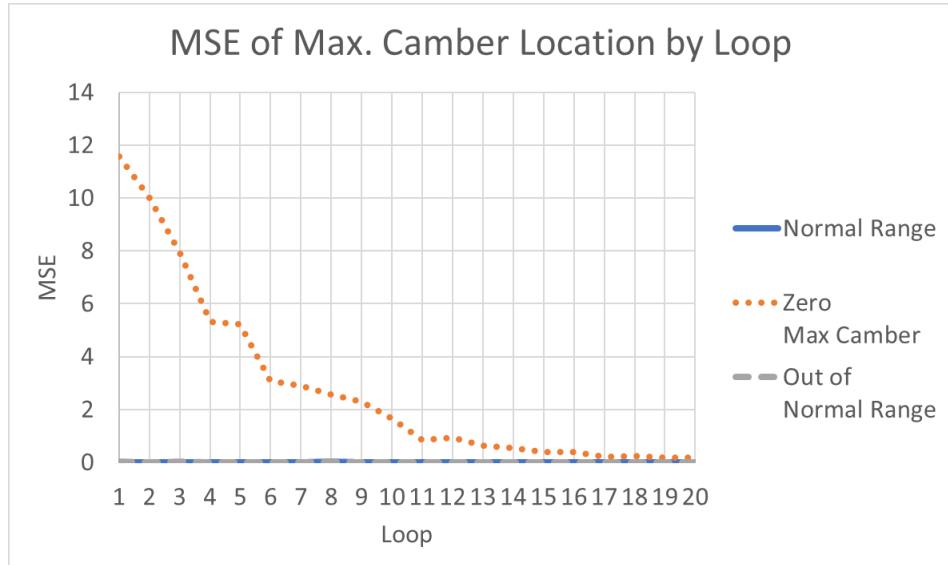
15	882	830	5,592	7,304
16	1,070	1,098	6,966	9,134
17	1,386	1,300	8,654	11,340
18	1,664	1,550	10,921	14,135
19	2,067	1,878	13,606	17,551
20	2,691	2,231	16,869	21,791

1) 형상 그룹 별 MSE

<그림 3-60>부터 <그림 3-63>은 각 루프에서 형상 파라미터 별로 측정한 MSE값을 나타낸 것이다. 형상 그룹에 관계없이, Loop가 진행될수록 MSE가 줄어드는 것을 볼 수 있다. 그래프들을 살펴보면 세 가지 파라미터들 중에서 테스트 MSE에 크게 영향을 미치는 것은 최대 캠버의 위치임을 확인할 수 있으며, 특히 Zero Max. Camber 그룹의 오차가 다른 그룹들과 다른 형상 파라미터들에 비해 매우 큰 것을 알 수 있다. 각 루프의 형상 그룹에 따른 형상/성능 파라미터 MSE는 Appendix D.1에서 확인할 수 있다. <표 3-56>~<표 3-61>에는 Loop 1과 20의 MSE만 간추렸다.

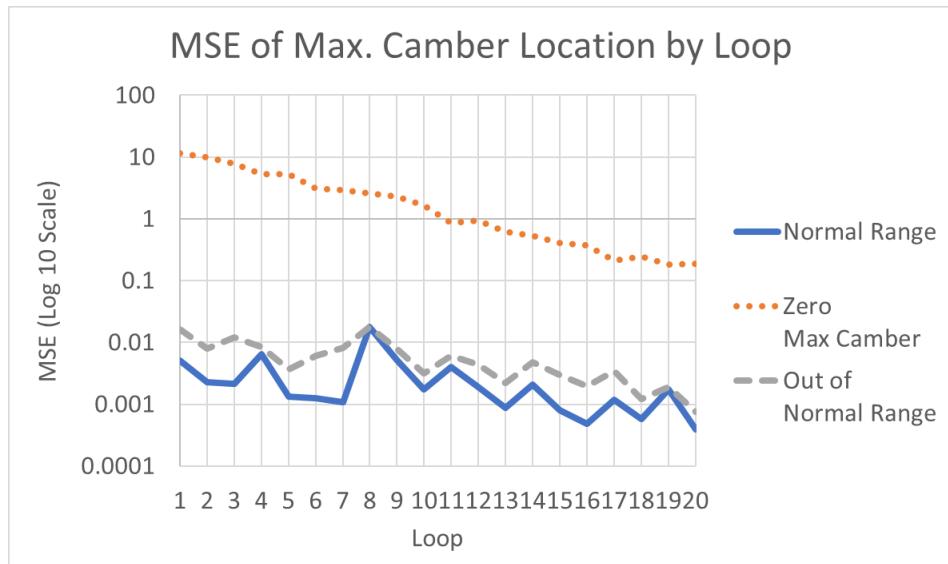


<그림 3-60> Loop에 따른 그룹 별 최대 캠버 MSE 변화

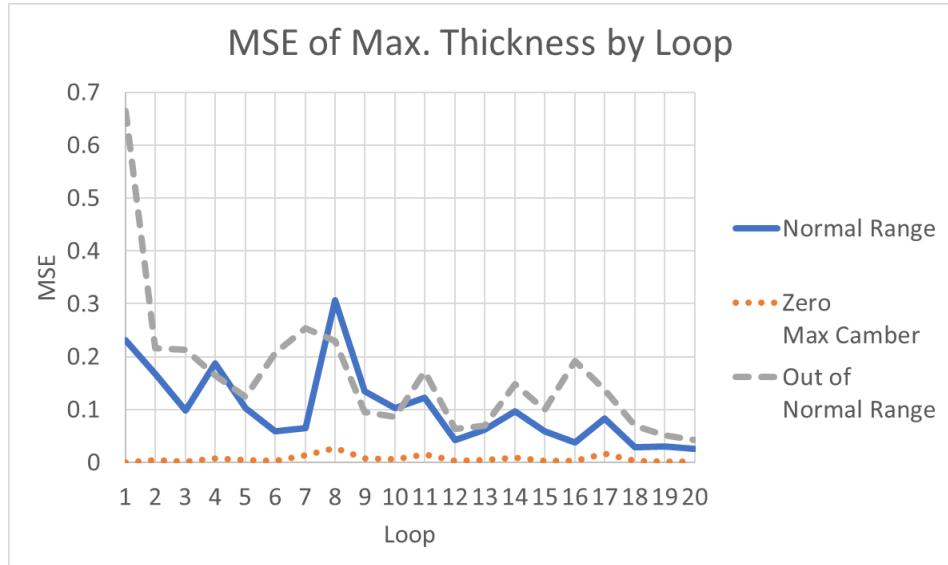


<그림 3-61> Loop에 따른 그룹 별 최대 캠버 위치 MSE 변화

<그림 3-61>은 Zero Max Camber 그룹의 오차가 다른 그룹에 비해 크기에 비교하기 어렵다. 아래의 <그림 3-62>는 <그림 3-61>의 오차를 log10으로 변환하여 비교한 것이다.



<그림 3-62> Loop에 따른 그룹 별 최대 캠버 위치 MSE 변화(Log10 Scale)



<그림 3-63> Loop에 따른 그룹 별 최대 두께 MSE 변화

<표 3-56> Normal Range: 형상 파라미터 MSE

Loop	Max. Camber	Location of Max. Camber	Max. Thickness
1	0.002379	0.005026	0.231741
20	0.000510	0.000392	0.025055

<표 3-57> Zero Max. Camber: 형상 파라미터 MSE

Loop	Max. Camber	Location of Max. Camber	Max. Thickness
1	0	11.569419	0.000719
20	0	0.188927	0.002246

<표 3-58> Out of Normal Range: 형상 파라미터 MSE

Loop	Max. Camber	Location of Max. Camber	Max. Thickness
1	0.02247	0.016426	0.665523
20	0.000777	0.000754	0.042802

<표 3-59> Normal Range: 성능 파라미터 MSE

Loop	Cl0	Cla0	Cd0	Cm0	Cma0
Loop 1	4.0.E-05	1.0.E-06	0.0.E+00	3.0.E-06	0.0.E+00
Loop 20	5.0.E-06	0.0.E+00	0.0.E+00	0.0.E+00	0.0.E+00

<표 3-60> Zero Max. Camber: 성능 파라미터 MSE

Loop	Cl0	Cla0	Cd0	Cm0	Cma0
Loop 1	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00
Loop 20	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00

<표 3-61> Out of Normal Range: 성능 파라미터 MSE

Loop	Cl0	Cla0	Cd0	Cm0	Cma0
Loop 1	4.68.E-04	1.72.E-04	1.00.E-06	2.60.E-05	1.00.E-05
Loop 20	2.30.E-05	1.29.E-04	0.00.E+00	1.00.E-06	6.00.E-06

2) 형상 그룹 별 형상/성능 오차의 평균과 표준편차

앞서 비교한 MSE는 인공신경망의 훈련에 사용되는 오차 측정 방법으로서, 물리적 의미를 갖는 오차와 성격이 다르다. 물리적 오차들은 평균적으로 <표 3-62>~<표 3-67>과 같이 나타난다. 전체 Loop에 대한 오차 평균은 254p의 Appendix D.2에 수록되어 있다.

Normal Range의 경우, Max. Camber의 오차 평균이 크게 증가하였으나 나머지 형상 오차의 평균이나 성능 오차의 평균은 큰 차이를 보이지 않는 것을 볼 수 있다. Zero Max. Camber 그룹은 최대 캠버 위치와 최대 두께가 개선되는 것을 볼 수 있으나, 성능 오차의 차이는 크지 않다.

반면, Out of Normal Range 그룹은 형상 오차 평균의 절대적인 크기가 증가하였으나, Cm,0만 오차의 절대적 크기가 증가하였으며, 나머지 성능 오차의 평균은 오히려 줄어들어 개선된 결과를 나타냈다. <그림 3-64>부터 <그림 3-66>은 형상 오차의 평균의 Loop 별 변화를 나타낸 것이다.

<표 3-62> Normal Range: 형상 파라미터 오차 평균

Loop	Max. Camber	Location of Max. Camber	Max. Thickness
1	-0.003183	-0.021154	-0.053103
20	0.013791	-0.008943	-0.01217

<표 3-63> Zero Max. Camber: 형상 파라미터 오차 평균

Loop	Max. Camber	Location of Max. Camber	Max. Thickness
1	0.000000	0.957772	0.021078
20	0.000015	0.012899	0.001608

<표 3-64> Out of Normal Range: 형상 파라미터 오차 평균

Loop	Max. Camber	Location of Max. Camber	Max. Thickness
1	-0.004076	-0.000916	0.025294
20	0.009722	-0.003600	-0.027890

<표 3-65> Normal Range: 성능 파라미터 오차 평균

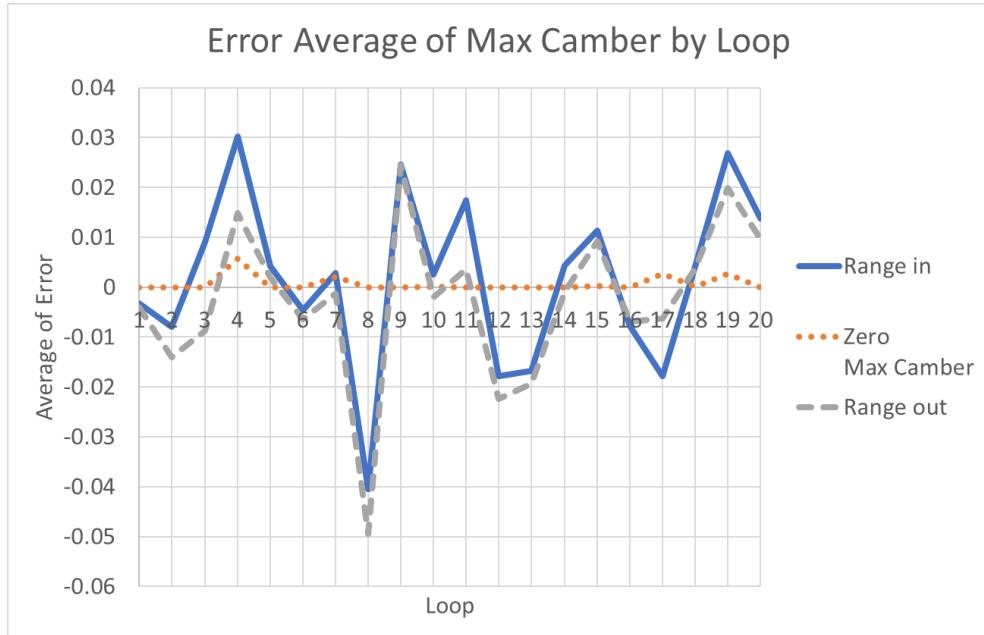
Loop	Cl0	Cla0	Cd0	Cm0	Cma0
Loop 1	-1.41.E-03	-1.06.E-04	6.00.E-06	4.50.E-04	3.10.E-05
Loop 20	1.25.E-03	-1.00.E-06	8.00.E-06	-1.85.E-04	0.00.E+00

<표 3-66> Zero Max. Camber: 성능 파라미터 오차 평균

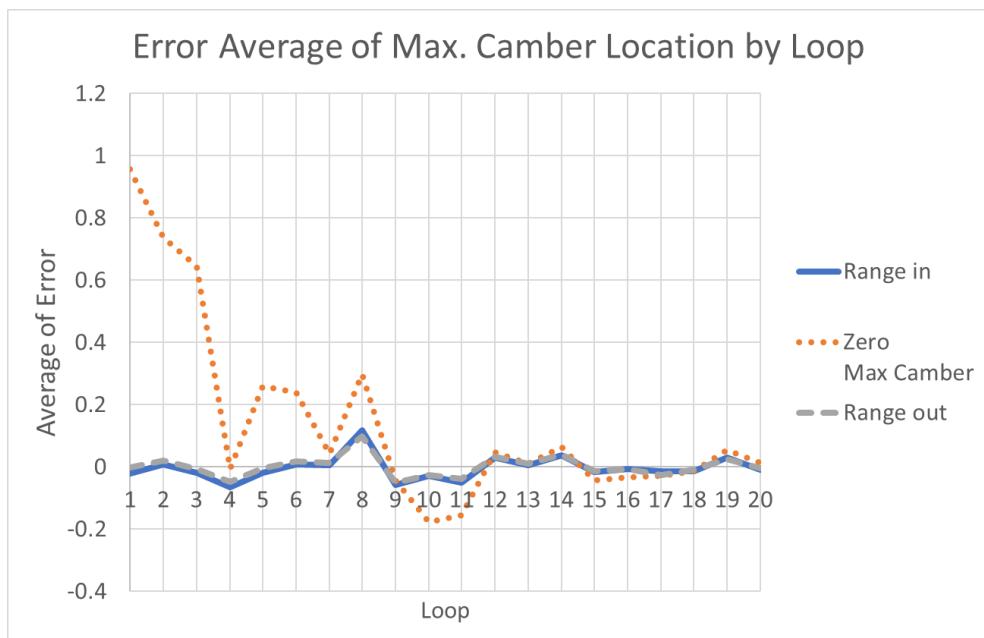
Loop	Cl0	Cla0	Cd0	Cm0	Cma0
Loop 1	-7.0.E-06	7.7.E-05	4.0.E-06	0.0.E+00	-1.1.E-05
Loop 20	0.0.E+00	-2.4.E-05	-2.7.E-05	0.0.E+00	2.0.E-06

<표 3-67> Out of Normal Range: 성능 파라미터 오차 평균

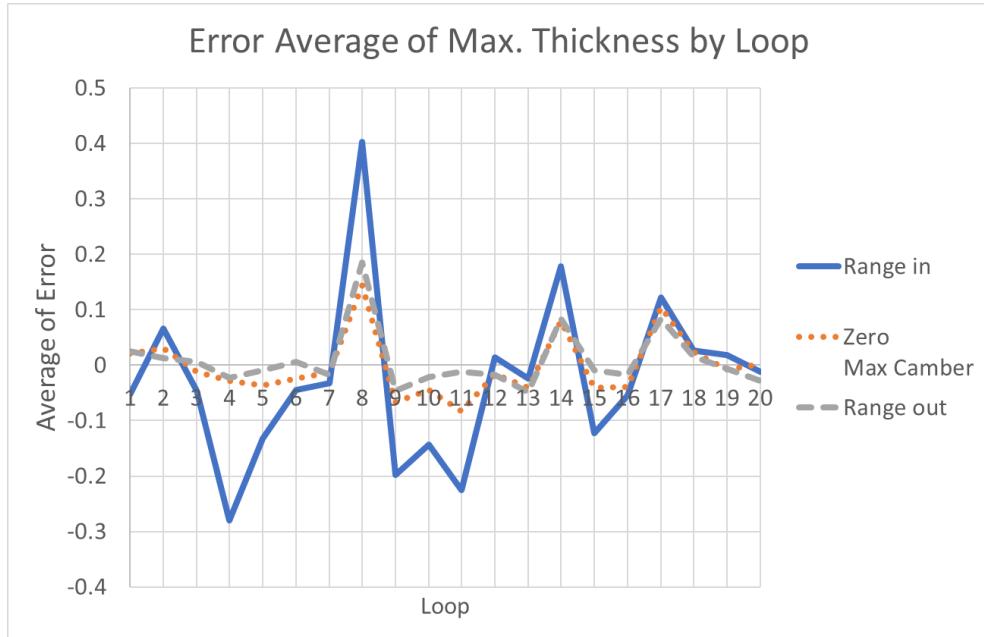
Loop	Cl0	Cla0	Cd0	Cm0	Cma0
Loop 1	3.29.E-04	-5.55.E-04	-6.70.E-05	-8.90.E-05	1.66.E-04
Loop 20	1.40.E-03	-8.00.E-06	1.00.E-05	-2.78.E-04	2.00.E-06



<그림 3-64> Loop에 따른 그룹 별 오차의 평균 변화: 최대 캠버



<그림 3-65> Loop에 따른 그룹 별 오차의 평균 변화: 최대 캠버 위치



<그림 3-66> Loop에 따른 그룹 별 오차의 평균 변화: 최대 두께

오차는 하나의 값으로 특정되는 것이 아니라 어떤 영역 안에 산포되어 있다. 그러므로 앞서 살펴본 오차의 평균만으로는 인공신경망 예측 성능의 개선 여부를 판가름하기 어렵다. 일반적으로 통계적 분석에는 평균과 표준편차가 같이 사용되는데, 각 그룹 별 형상/성능 오차의 표준편차는 <표 3-68>~<표 3-73>과 같다. 전체 Loop에 따른 오차의 표준편차는 260p, Appendix D.3에 수록되어 있다.

Normal Range 그룹의 오차 표준편차는 형상/성능 파라미터 모두 줄어들었음을 확인할 수 있다. 이는 오차가 평균에 밀집되어 있다는 것을 의미한다. 반면, Zero Max. Camber 그룹은 최대 캠버 위치의 표준편차는 개선되었으나, 다른 형상 파라미터의 표준편차가 증가하였다. 이는 성능 표준편차에도 영향을 미쳐 증가시키는 결과를 낳았다. Out of Normal Range 그룹은 형상 오차의 표준편차는 모두 개선되었으며, 성능 파라미터 역시 표준편차가 줄어든 것을 확인할 수 있었다. <그림 3-67>부터 <그림 3-70>은 Loop 별 형상 오차의 표준편차의 변화를 나타낸 것이다.

<표 3-68> Normal Range: 형상 파라미터 오차의 표준편차

Loop	Max. Camber	Location of Max. Camber	Max. Thickness
1	0.048676	0.067661	0.478457
20	0.017869	0.017653	0.157820

<표 3-69> Zero Max. Camber: 형상 파라미터 오차의 표준편차

Loop	Max. Camber	Location of Max. Camber	Max. Thickness
1	0.000000	3.263754	0.016585
20	0.000422	0.434466	0.047364

<표 3-70> Out of Normal Range: 형상 파라미터 오차의 표준편차

Loop	Max. Camber	Location of Max. Camber	Max. Thickness
1	0.149846	0.128161	0.815404
20	0.026122	0.027221	0.204999

<표 3-71> Normal Range: 성능 파라미터 오차의 표준편차

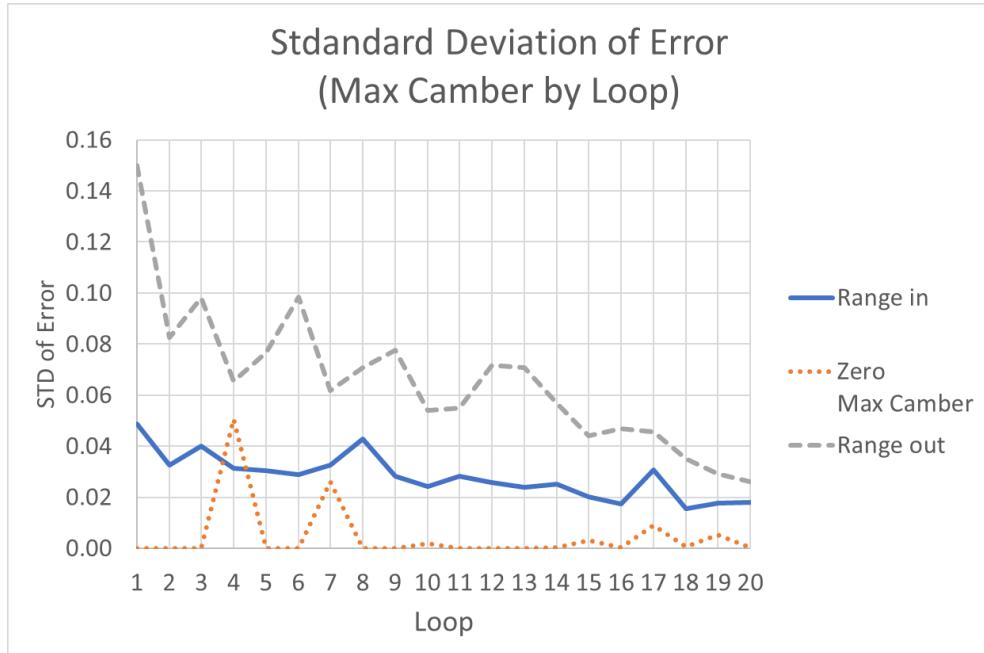
Loop	Cl0	Cla0	Cd0	Cm0	Cma0
Loop 1	6.19.E-03	7.37.E-04	5.60.E-05	1.69.E-03	1.60.E-04
Loop 20	1.74.E-03	4.01.E-04	1.60.E-05	4.56.E-04	9.20.E-05

<표 3-72> Zero Max. Camber: 성능 파라미터 오차의 표준편차

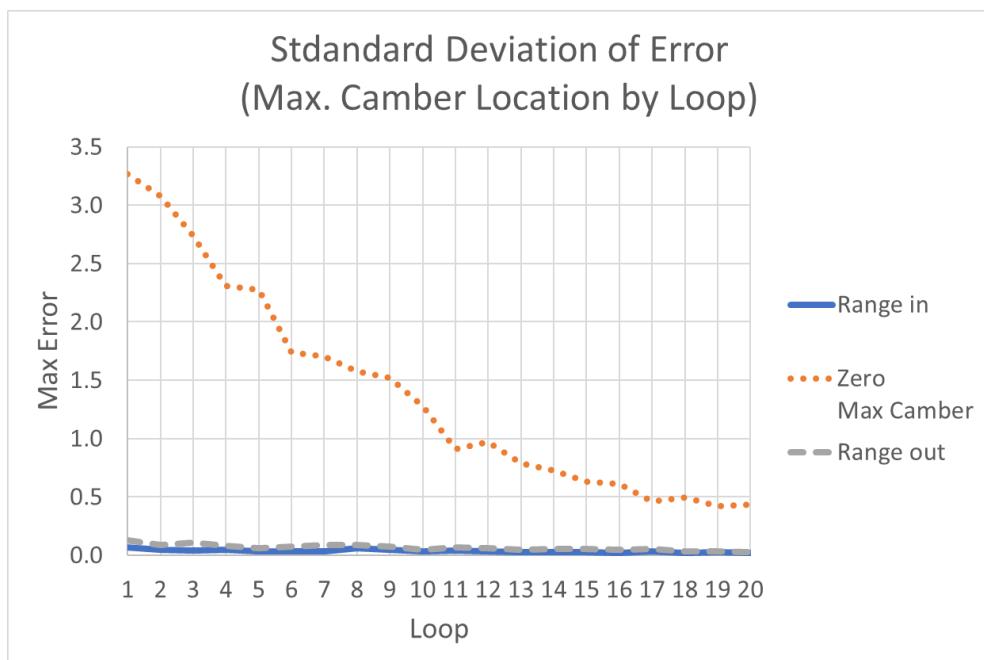
Loop	Cl0	Cla0	Cd0	Cm0	Cma0
Loop 1	3.90.E-05	9.70.E-05	1.70.E-05	0.00.E+00	3.20.E-05
Loop 20	7.30.E-05	2.31.E-04	4.70.E-04	1.30.E-05	3.40.E-05

<표 3-73> Out of Normal Range: 성능 파라미터 오차의 표준편차

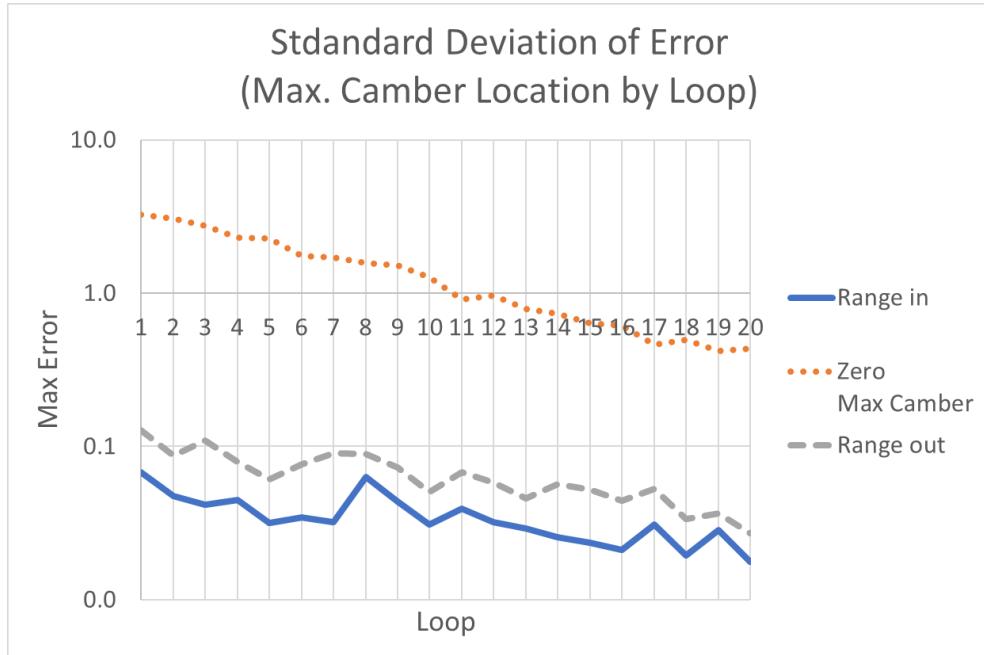
Loop	Cl0	Cla0	Cd0	Cm0	Cma0
Loop 1	2.16.E-02	1.31.E-02	1.12.E-03	5.08.E-03	3.13.E-03
Loop 20	4.63.E-03	1.14.E-02	3.84.E-04	1.12.E-03	2.54.E-03



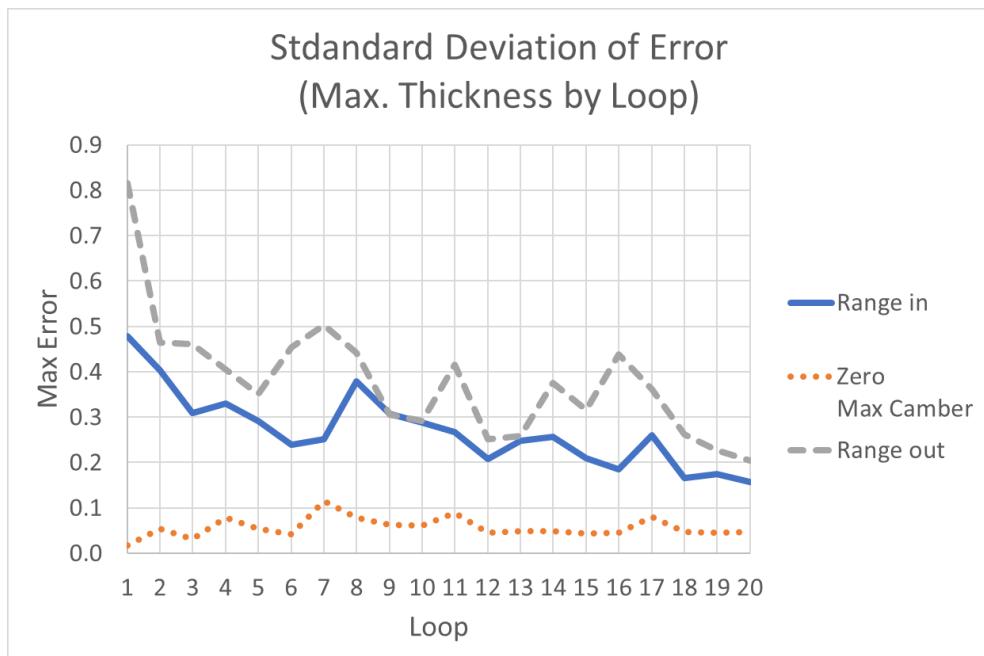
<그림 3-67> Loop에 따른 그룹 별 오차의 표준편차 변화: 최대 캠버



<그림 3-68> Loop에 따른 그룹 별 오차의 표준편차 변화: 최대 캠버 위치



<그림 3-69> Loop에 따른 그룹 별 오차의 표준편차 변화: 최대 캠버 위치
(Log10 Scale)



<그림 3-70> Loop에 따른 그룹 별 오차의 표준편차 변화: 최대 두께

F. 정리

- 데이터베이스 강화 루프는 인공신경망의 학습 테스트 오차를 줄이는데 효과가 있다.
- 동일한 에어포일 데이터로 각 Loop에서 학습된 모델을 테스트를 실시해도 오차가 감소하는 것을 확인할 수 있다.
- NACA 4-digit은 형상 파라미터가 불연속적인 지점이 있으며, 이는 최대 캠버가 0인 경우 발생한다. 최대 캠버가 0이면 이는 대칭형 익형이 되며, 최대 캠버 위치값이 어떤 값이 온다 하더라도 의미가 없어진다.
- 대칭형 에어포일의 최대 캠버 위치로 인해 초기 에어포일 데이터를 이용한 학습에서 MSE가 크게 나타났다. 그러나 데이터의 증가가 이를 완화시킴을 확인할 수 있다.
- 데이터베이스 강화 루프는 형상 오차의 평균과 표준편차를 줄이는데 효과적이었으며, 형상 파라미터의 불연속점 문제를 해결할 수 있었다. 성능 오차의 평균은 증가하거나 감소하는 혼합적인 양상을 보이나, 그 표준편차를 줄이는데 효과적이었다.

6. 최종 하이퍼 파라미터 최적화

(Final Hyperparameter Optimization, FHO)

하이퍼 파라미터 최적화는 모델의 학습 양상과 결과를 결정짓는다. 앞서 데이터베이스 강화 루프를 통해 에어포일 데이터는 그 양이 기하급수적으로 증가하였다. 이미 앞서 초기 하이퍼파라미터 최적화 단계에서 두 가지 하이퍼 파라미터에 대한 적절한 값을 찾았으나, 데이터 수와 그 구성이 달라졌으므로 다시 한번 최적화를 할 필요가 있다.

이 단계에서는 최종적으로 획득한 Loop 20의 데이터베이스를 이용하여 하이퍼 파라미터 최적화를 실시하였다. 앞서 실시한 초기 하이퍼 파라미터 최적화(IHO) 단계와 구분하기 위해 이를 최종 하이퍼 파라미터(Final

Hyperparameter Optimization, FHO)라 하였으며, 다음과 같은 항목에 대한 최적화를 진행하였다.

- 최종 하이퍼 파라미터 단계 최적화 목록
 - Epoch
 - Number of Hidden Layers
 - Number of Neurons per Hidden Layer

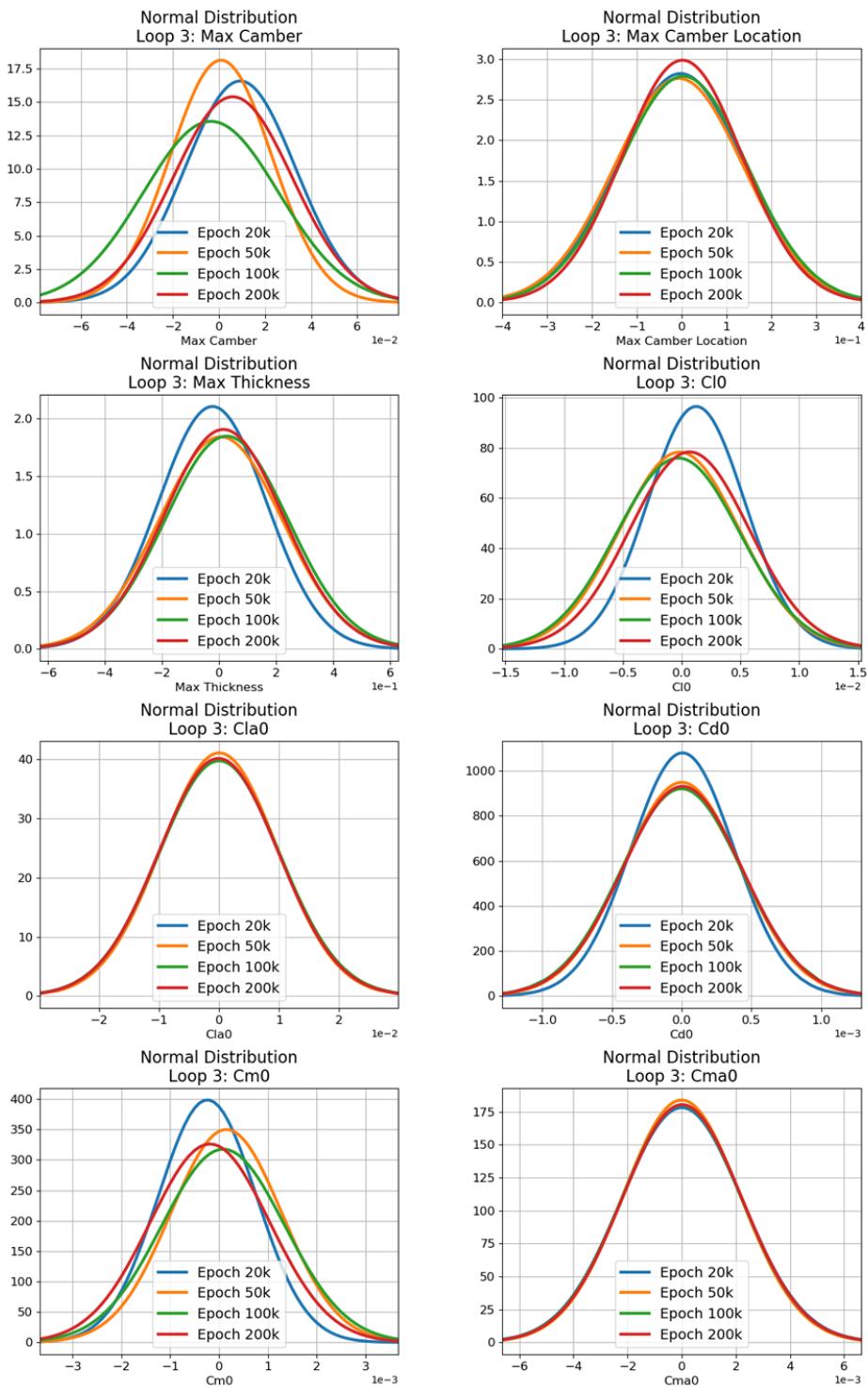
A. Epoch 최적화

Epoch는 전체 학습 데이터를 몇 번이나 학습하였는지를 나타낸다. 혹은 몇 번을 학습할지를 정하는 하이퍼 파라미터다. 이는 mini-batch를 사용할 때 쓰이는 Iteration과 구분하여 사용해야 하는데, Iteration은 파라미터 업데이트가 일어난 횟수를 의미하기 때문이다. Epoch와 Iteration의 구분은 제 1 절 3. B. 1)에서 설명하였다. 본 논문에서는 <표 3-74>와 같이 Epoch를 네 가지 Case로 나누어 최적 값을 찾기 위한 테스트를 진행하였다.

<표 3-74> 최종 하이퍼 파라미터 최적화: Epoch Test Case

Epoch	Case 1 (Baseline)	Case 2	Case 3	Case 4
	20,000 (20k)	50,000 (50k)	100,000 (100k)	200,000 (200k)

다음 페이지에 있는 <그림 3-71>은 Epoch test case에 따른 형상/성능 오차의 정규분포도다. 성능 오차들 중, 기울기 파라미터(Cla0, Cma0)는 Epoch의 변화에도 오차 분포의 양상이 크게 다르지 않다. Cd0의 경우, 평균은 비슷하나 20k의 표준편차가 다른 Case들보다 작은 것을 볼 수 있다. 양력과 모멘트 계수는 Epoch에 따라 오차의 양상이 크게 달라졌는데, 표준편자는 20k가 가장 작으나, 평균은 100k가 0에 가장 근접하였다. 즉, 100k에서 더 작은 오차 범위를 중심으로 몰려 있다고 볼 수 있다.



<그림 3-71> FHO: Epoch 수에 따른 형상/성능 오차의 정규분포

어떤 모델이 더 낮은 오차를 많이 내는지 더 확실하게 보기 위해, 각 파라미터 오차를 절대값을 취한 뒤, 이들을 각각 내림차순으로 정렬하였다. 다음 표들은 내림 차순으로 정렬한 데이터들(오차) 중, 상위 1%, 5%, 10%, 15%, 20%에 위치한 데이터의 오차를 나타낸 것이다. 그러므로 각각의 오차 밑에 있는 99%, 95%, 90%, 85%, 80%의 데이터들은 그보다 낮은 오차를 갖는다.

<표 3-77>~<표 3-84>는 파라미터 별 각 지점(Percentile Rank)의 오차를 나타낸 것이다. Case 1(Epoch 20k)의 최대 캠버 99% 지점 오차는 0.07409이고, 95% 지점 오차는 0.04170이다. 99% 지점은 당연히 오차 상위 1% 지점이므로, 오차 상위 5% 지점인 95% 지점보다 더 높은 오차를 갖게 된다. 95% 지점 아래에 있는 데이터들은 전체 데이터의 95%를 차지하며, 0.04170보다 작은 오차를 갖는다.

<표 3-76>은 각 파라미터와 Percentile rank에서 최소 오차가 나타난 Test case와 가장 빈도수가 높은 Test case(Mode)를 정리한 것이다. 전체 파라미터에서는 Case 2와 3이 40개의 지점 중, 16번의 최소 지점을 가진다. 형상 파라미터에서는 Case 2가 15개의 지점 중 6개의 지점에서 최소값을 가졌으며, 성능 파라미터에서는 Case 3가 25개의 지점 중 13개의 지점에서 최소값을 가졌다. 설계의 목표는 요구한 성능을 내는 형상을 만들어 내는 것이므로, 요구한 성능과 설계한 형상의 성능 오차가 적을수록 잘 만든 것이다. 그러므로 Epoch test에서는 성능 오차의 빈도가 더 높은 **Test case 3**을 최적 하이퍼 파라미터로 선정하였다.

<그림 3-72>와 <그림 3-73>은 각 파라미터들의 Test case를 Percentile rank 별로 묶어 나타낸 그래프이다. 각 Rank에는 Case1부터 Case4까지 차례대로 배치되어 있다. 그래프 상에서도 Test case 2와 3이 전반적으로 좋은 결과를 보여주는 것을 볼 수 있다. Epoch test의 모든 평균과 표준편차 데이터는 266p의 Appendix E.1에서 확인할 수 있다.

<표 3-75> FHO – Epoch Test: 소요 시간

Time(Hr)	Case			
	1	2	3	4
Training	2.07	6.52	10.34	20.73
Analysis	4.18	4.24	4.21	4.23
Total	6.25	10.76	14.55	24.96

<표 3-76> FHO - Epoch Test: 최소 오차가 나타난 Test Case와
가장 많이 나타난 Test Case(Mode)

Parameter	Case of Minimum Error					Mode
	99%	95%	90%	85%	80%	
Max. Camber	3	2	2	2	2	2
Max. Camber Location	2	4	4	3	3	4
Max. Thickness	2	4	4	4	4	4
Cl0	2	3	2	2	2	2
Cla0	2	4	3	3	3	3
Cd0	2	3	3	2	2	2
Cm0	3	3	3	3	3	3
Cma0	2	2	3	1	3	2
Mode	2	4	3	2	3	2, 3

<표 3-77> FHO - Epoch Test Percentile Rank: Max. Camber

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	7.409.E-02	4.170.E-02	3.253.E-02	2.731.E-02	2.335.E-02
2	5.543.E-02	2.705.E-02	2.003.E-02	1.623.E-02	1.398.E-02
3	5.434.E-02	2.775.E-02	2.062.E-02	1.719.E-02	1.482.E-02
4	6.071.E-02	2.991.E-02	2.250.E-02	1.880.E-02	1.630.E-02

<표 3-78> FHO - Epoch Test Percentile Rank: Max. Camber Location

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	1.690.E-01	6.070.E-02	3.846.E-02	3.043.E-02	2.565.E-02
2	1.664.E-01	5.589.E-02	3.815.E-02	3.114.E-02	2.677.E-02
3	1.816.E-01	5.429.E-02	3.455.E-02	2.591.E-02	2.116.E-02
4	1.729.E-01	5.302.E-02	3.378.E-02	2.625.E-02	2.141.E-02

<표 3-79> FHO - Epoch Test Percentile Rank: Max. Thickness

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	5.012.E-01	2.503.E-01	1.781.E-01	1.437.E-01	1.225.E-01
2	4.722.E-01	2.226.E-01	1.524.E-01	1.207.E-01	1.005.E-01
3	4.767.E-01	2.166.E-01	1.495.E-01	1.175.E-01	9.670.E-02
4	4.763.E-01	2.127.E-01	1.421.E-01	1.105.E-01	9.139.E-02

<표 3-80> FHO - Epoch Test Percentile Rank: Cl0

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	1.42.E-02	5.10.E-03	3.80.E-03	3.10.E-03	2.70.E-03
2	1.13.E-02	3.60.E-03	2.40.E-03	2.00.E-03	1.70.E-03
3	1.13.E-02	3.50.E-03	2.40.E-03	2.00.E-03	1.70.E-03
4	1.22.E-02	4.10.E-03	2.90.E-03	2.30.E-03	2.00.E-03

<표 3-81> FHO - Epoch Test Percentile Rank: Cla0

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	2.33.E-02	3.80.E-03	1.80.E-03	1.10.E-03	7.00.E-04
2	1.94.E-02	3.50.E-03	1.80.E-03	1.10.E-03	8.00.E-04
3	1.99.E-02	3.50.E-03	1.50.E-03	9.00.E-04	6.00.E-04
4	2.12.E-02	3.40.E-03	1.60.E-03	9.00.E-04	6.00.E-04

<표 3-82> FHO - Epoch Test Percentile Rank: Cd0

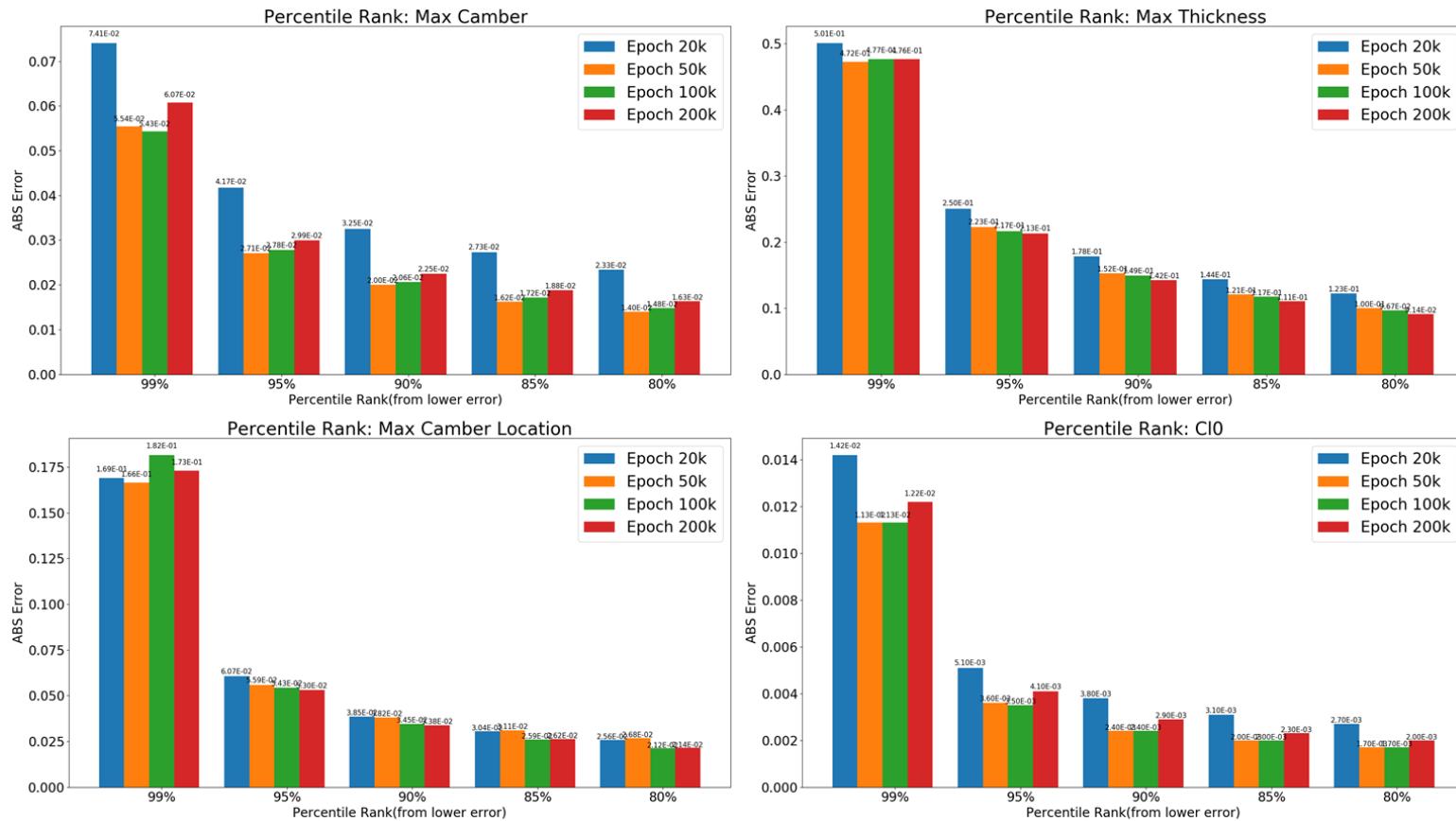
Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	6.1.E-04	1.5.E-04	7.0.E-05	5.0.E-05	4.0.E-05
2	4.3.E-04	1.1.E-04	6.0.E-05	4.0.E-05	3.0.E-05
3	4.4.E-04	1.0.E-04	5.0.E-05	4.0.E-05	3.0.E-05
4	5.2.E-04	1.3.E-04	7.0.E-05	4.0.E-05	3.0.E-05

<표 3-83> FHO - Epoch Test Percentile Rank: Cm0

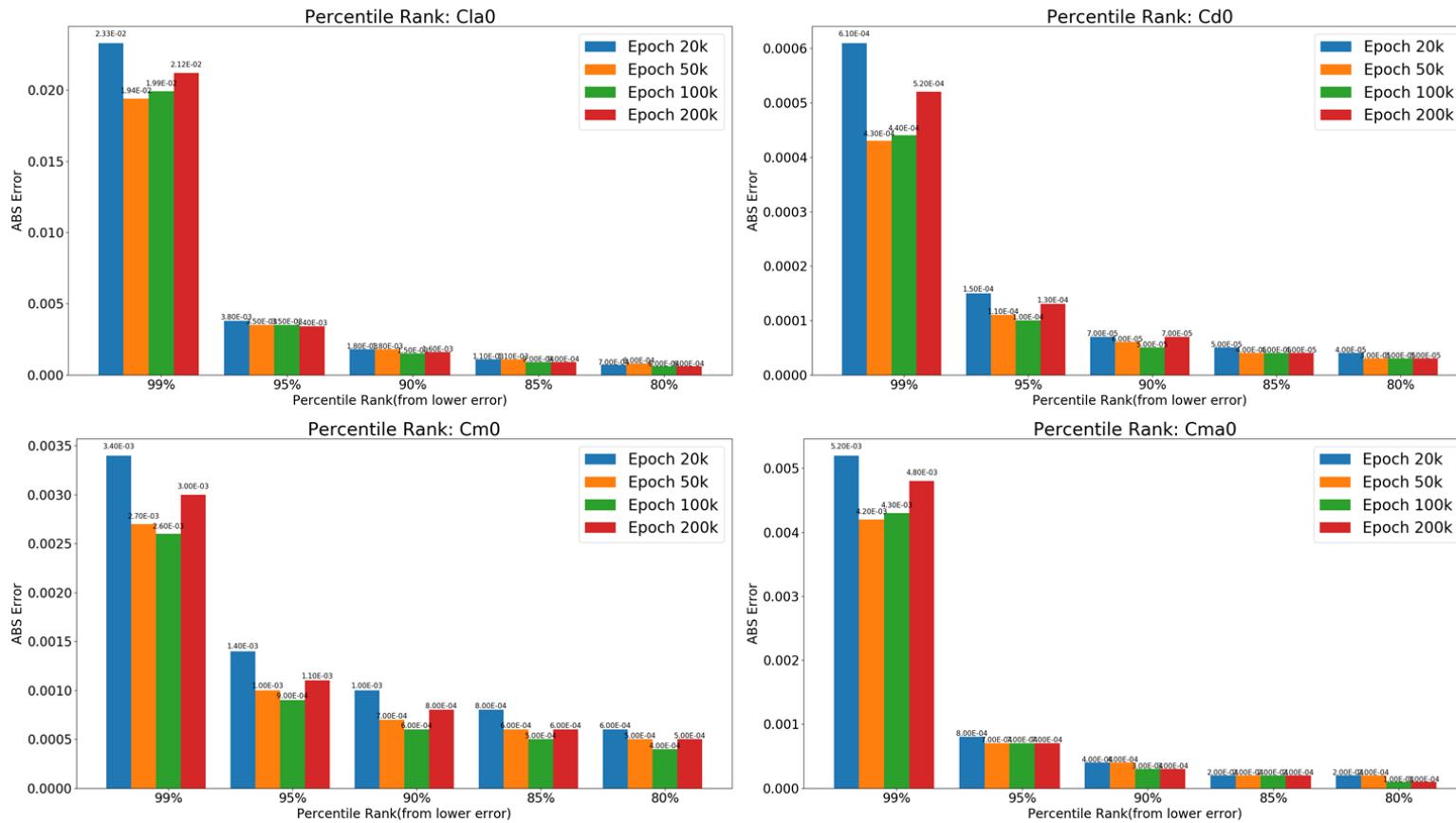
Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	3.4.E-03	1.4.E-03	1.0.E-03	8.0.E-04	6.0.E-04
2	2.7.E-03	1.0.E-03	7.0.E-04	6.0.E-04	5.0.E-04
3	2.6.E-03	9.0.E-04	6.0.E-04	5.0.E-04	4.0.E-04
4	3.0.E-03	1.1.E-03	8.0.E-04	6.0.E-04	5.0.E-04

<표 3-84> FHO - Epoch Test Percentile Rank: Cma0

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	5.2.E-03	8.0.E-04	4.0.E-04	2.0.E-04	2.0.E-04
2	4.2.E-03	7.0.E-04	4.0.E-04	2.0.E-04	2.0.E-04
3	4.3.E-03	7.0.E-04	3.0.E-04	2.0.E-04	1.0.E-04
4	4.8.E-03	7.0.E-04	3.0.E-04	2.0.E-04	1.0.E-04



<그림 3-72> FHO - Epoch Test Percentile Rank:
Max. Camber, Max. Camber Location, Max. Thickness와 C10



<그림 3-73> FHO - Epoch Test Percentile Rank:
Cla0, Cd0, Cm0와 Cma0

B. Number of Hidden Layers

은닉층의 수는 인공신경망의 모델을 더 복잡하게 만들어, 더 어려운 문제도 풀 수 있도록 해준다. 하지만 은닉층 수가 깊어지면 학습이 어려워지고, 모델이 무거워진다. 본 연구에서는 <표 3-85>와 같이 6가지 Case에 대한 테스트를 진행하였다.

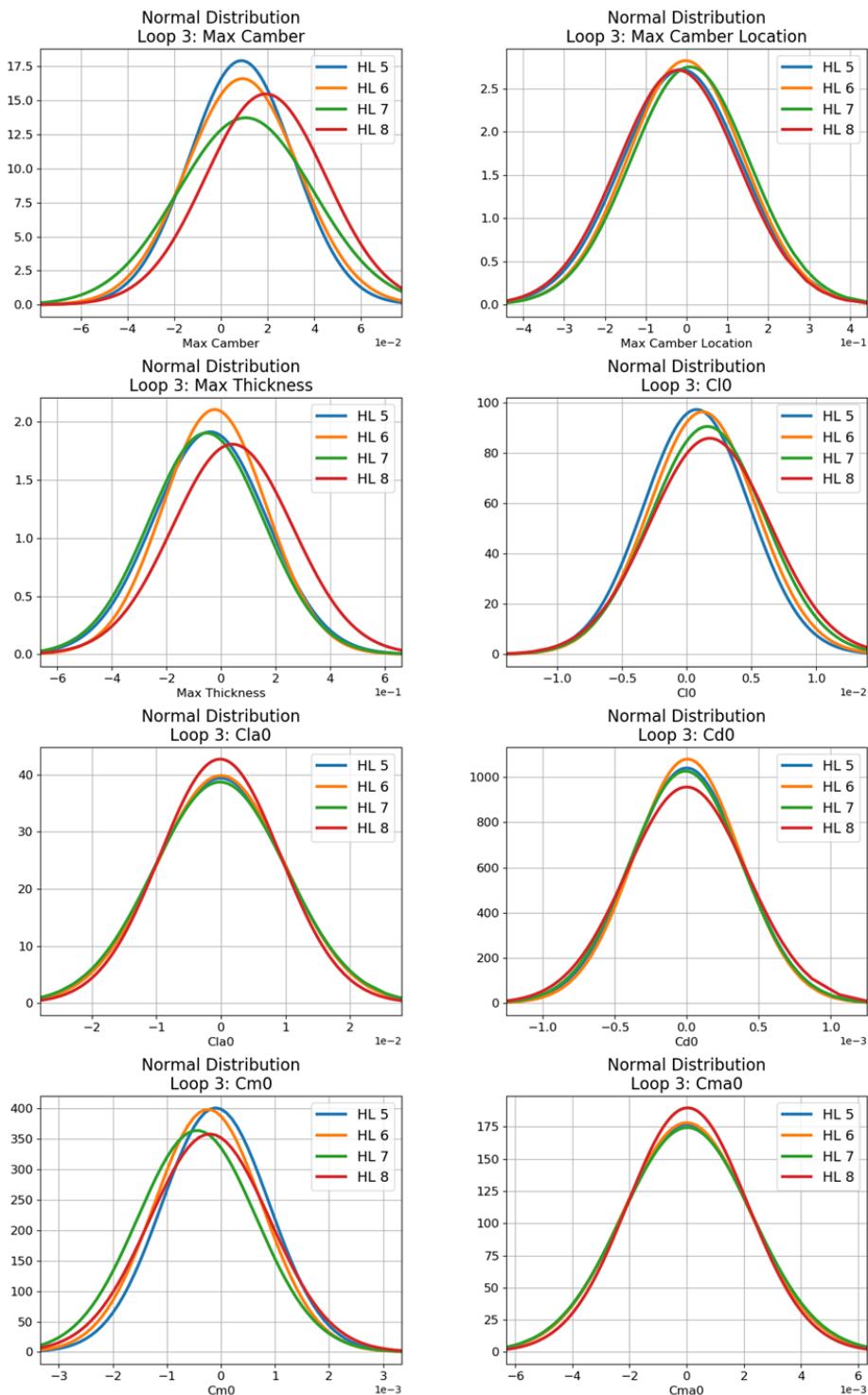
<표 3-85> 최종 하이퍼 파라미터 최적화(FHO): # of Hidden Layers

# of Hidden Layers	Case 1	Case 2 (Baseline)	Case 3	Case 4
	5	6	7	8

<그림 3-74>는 Hidden Layer test case에 따른 형상/성능 오차의 정규분포도다. 전반적으로 Case 1이 좋은 결과를 보여주는 것을 확인할 수 있으며, 특히 성능 파라미터의 오차에서는 Case 1이 평균과 표준편차 모두 좋은 모습을 보여준다.

앞서 Epoch 최적화와 마찬가지로 Hidden layer 테스트 결과를 바탕으로 파라미터 별 각 지점의 오차를 측정하여 최소 오차가 나타난 Test case 및 빈도수를 <표 3-87>에 정리하였다. 가장 낮은 오차를 기록한 Test case는 은닉층 수가 5개인 Case 1이었으며, 총 40개 중 28개 지점에서 가장 낮은 오차를 기록하였다. 전체 형상/성능 파라미터들의 각 Percentile rank 오차는 <표 3-88>~<표 3-95>에 정리되어 있다.

<그림 3-75>와 <그림 3-76>은 각 파라미터들의 Test case를 Percentile rank 별로 묶어 나타낸 그래프이다. 마찬가지로 전반적으로 Case 1이 좋은 결과를 보여주는 것을 알 수 있다. 본 연구에서는 **Case 1(5 Hidden Layer)**을 최적 하이퍼 파라미터로 선정하였으며, Hidden layer 테스트의 모든 평균과 표준편차 데이터는 269p의 Appendix E.2에서 확인할 수 있다.



<그림 3-74> FHO: Hidden Layer 수에 따른 형상/성능 오차의 정규분포

<표 3-86> FHO – Hidden Layer Test: 소요 시간

Time(Hr)	Case			
	1	2	3	4
Training	1.73	2.07	2.4	2.73
Analysis	4.12	4.18	4.14	4.2
Total	5.85	6.25	6.54	6.93

<표 3-87> FHO – Hidden Layer Test: 최소 오차가 나타난 Test Case와
가장 많이 나타난 Test Case(Mode)

Parameter	Case of Minimum Error					Mode
	99%	95%	90%	85%	80%	
Max. Camber	1	1	1	1	1	1
Max. Camber Location	2	2	2	2	2	2
Max. Thickness	1	1	2	1	1	1
Cl0	3	1	1	1	1	1
Cla0	3	3	1	1	2	3
Cd0	1	3	1	1	1	1
Cm0	3	1	1	1	1	1
Cma0	1	1	1	1	1	1
Mode	1	1	1	1	1	1

<표 3-88> FHO – Hidden Layer Test Percentile Rank: Max. Camber

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	6.548.E-02	3.711.E-02	2.838.E-02	2.356.E-02	2.034.E-02
2	7.409.E-02	4.170.E-02	3.253.E-02	2.731.E-02	2.335.E-02
3	7.454.E-02	4.689.E-02	3.766.E-02	3.194.E-02	2.785.E-02
4	7.009.E-02	4.695.E-02	3.929.E-02	3.477.E-02	3.150.E-02

<표 3-89> FHO – Hidden Layer Test Percentile Rank: Max. Camber Location

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	1.692.E-01	6.461.E-02	4.375.E-02	3.562.E-02	3.085.E-02
2	1.690.E-01	6.070.E-02	3.846.E-02	3.043.E-02	2.565.E-02
3	1.779.E-01	7.135.E-02	4.619.E-02	3.535.E-02	2.874.E-02
4	1.762.E-01	7.350.E-02	5.235.E-02	4.379.E-02	3.847.E-02

<표 3-90> FHO – Hidden Layer Test Percentile Rank: Max. Thickness

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	4.725.E-01	2.465.E-01	1.791.E-01	1.429.E-01	1.211.E-01
2	5.012.E-01	2.503.E-01	1.781.E-01	1.437.E-01	1.225.E-01
3	4.962.E-01	2.583.E-01	1.853.E-01	1.503.E-01	1.274.E-01
4	4.973.E-01	2.584.E-01	1.868.E-01	1.497.E-01	1.266.E-01

<표 3-91> FHO – Hidden Layer Test Percentile Rank: Cl0

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	1.33.E-02	4.70.E-03	3.20.E-03	2.60.E-03	2.20.E-03
2	1.42.E-02	5.10.E-03	3.80.E-03	3.10.E-03	2.70.E-03
3	1.27.E-02	5.70.E-03	4.60.E-03	3.90.E-03	3.50.E-03
4	1.62.E-02	5.70.E-03	4.00.E-03	3.30.E-03	2.90.E-03

<표 3-92> FHO – Hidden Layer Test Percentile Rank: Cla0

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	2.18.E-02	3.70.E-03	1.80.E-03	1.10.E-03	8.00.E-04
2	2.33.E-02	3.80.E-03	1.80.E-03	1.10.E-03	7.00.E-04
3	2.17.E-02	3.60.E-03	1.80.E-03	1.10.E-03	7.00.E-04
4	2.31.E-02	4.50.E-03	2.20.E-03	1.40.E-03	9.00.E-04

<표 3-93> FHO – Hidden Layer Test Percentile Rank: Cd0

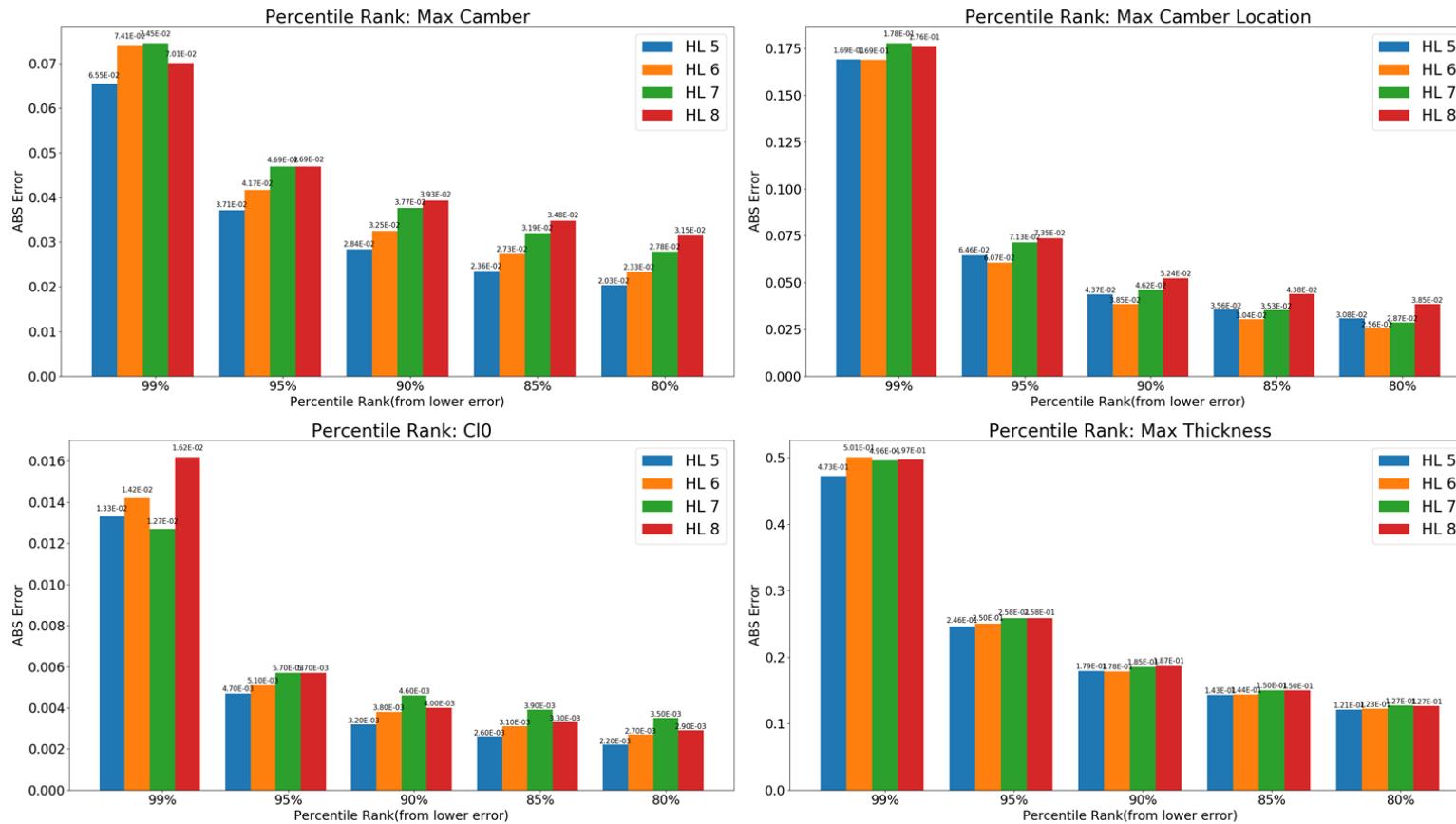
Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	4.6.E-04	1.2.E-04	6.0.E-05	4.0.E-05	3.0.E-05
2	6.1.E-04	1.5.E-04	7.0.E-05	5.0.E-05	4.0.E-05
3	4.9.E-04	1.1.E-04	6.0.E-05	4.0.E-05	3.0.E-05
4	6.1.E-04	1.3.E-04	7.0.E-05	5.0.E-05	4.0.E-05

<표 3-94> FHO – Hidden Layer Test Percentile Rank: Cm0

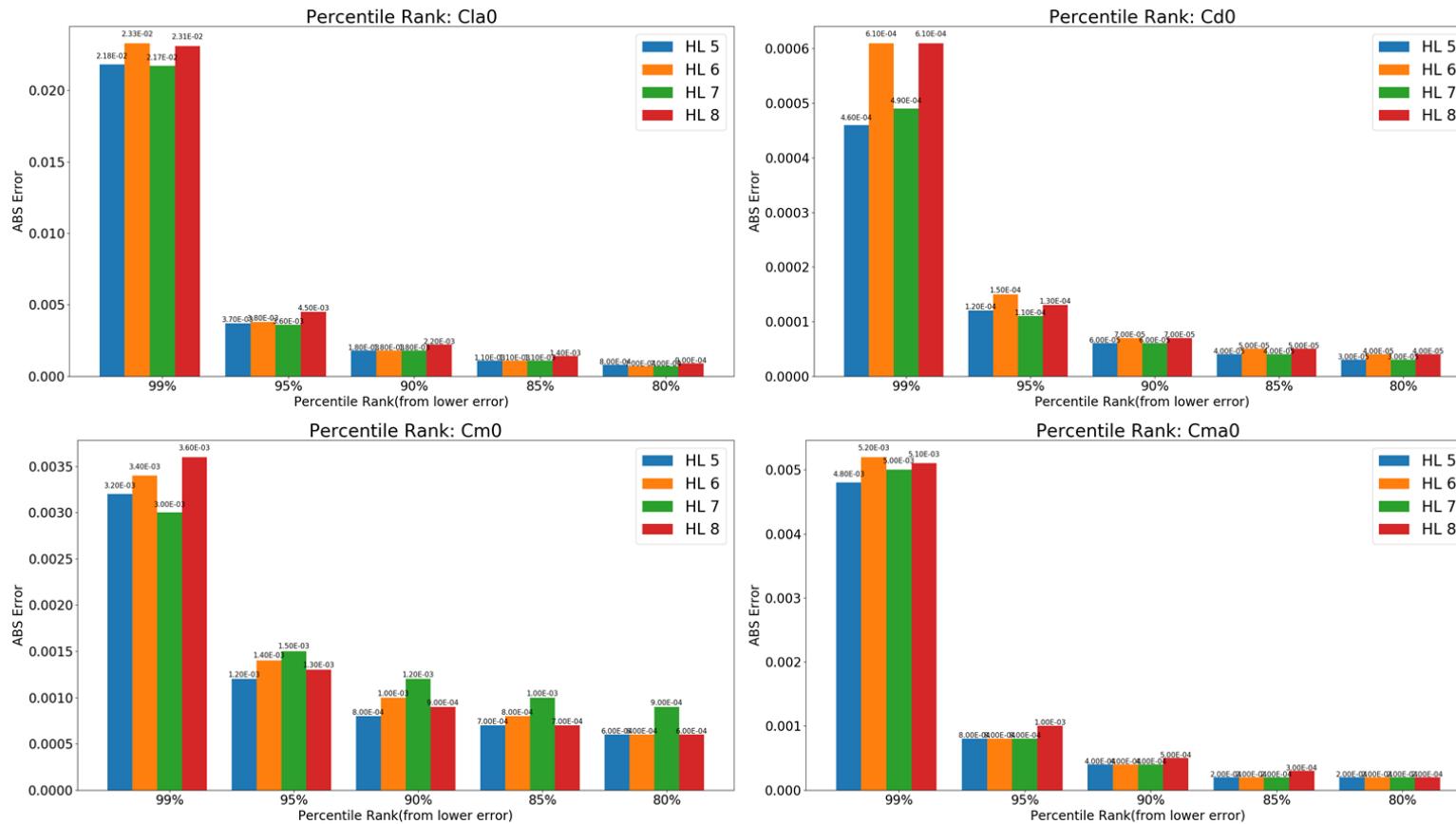
Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	3.2.E-03	1.2.E-03	8.0.E-04	7.0.E-04	6.0.E-04
2	3.4.E-03	1.4.E-03	1.0.E-03	8.0.E-04	6.0.E-04
3	3.0.E-03	1.5.E-03	1.2.E-03	1.0.E-03	9.0.E-04
4	3.6.E-03	1.3.E-03	9.0.E-04	7.0.E-04	6.0.E-04

<표 3-95> FHO – Hidden Layer Test Percentile Rank: Cma0

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	4.8.E-03	8.0.E-04	4.0.E-04	2.0.E-04	2.0.E-04
2	5.2.E-03	8.0.E-04	4.0.E-04	2.0.E-04	2.0.E-04
3	5.0.E-03	8.0.E-04	4.0.E-04	2.0.E-04	2.0.E-04
4	5.1.E-03	1.0.E-03	5.0.E-04	3.0.E-04	2.0.E-04



<그림 3-75> FHO – Hidden Layer Test Percentile Rank:
Max. Camber, Max. Camber Location, Max. Thickness와 C10



<그림 3-76> FHO – Hidden Layer Test Percentile Rank:
Cla0, Cd0, Cm0와 Cma0

C. Number of Neurons per Hidden Layer

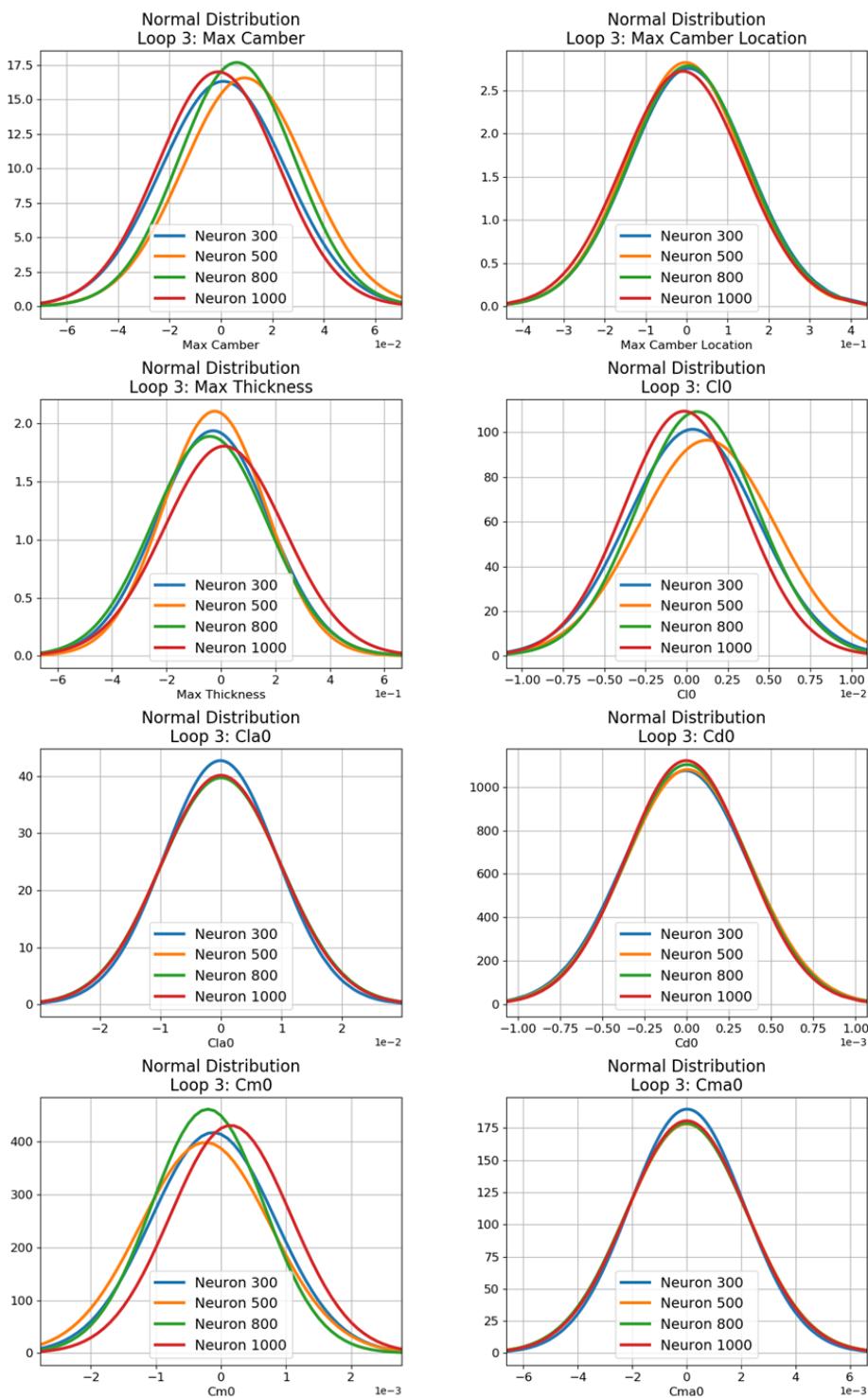
은닉층 내부의 뉴런 수는 모델의 차원을 높이며, 은닉층 수와 마찬가지로 인공신경망 모델의 복잡도를 결정한다. 뉴런의 수가 많아지면 파라미터 수가 증가하므로, 학습 시 업데이트를 해야 할 변수의 수가 많아지게 된다. 하지만 일반적으로 뉴런 수가 증가하면 모델이 더 복잡한 문제를 풀어낼 수 있다. 본 연구에서는 <표 3-96>과 같이 네 가지 Case에 대하여 테스트를 진행하였다.

<표 3-96> 최종 하이퍼 파라미터 최적화: # of Neurons per Hidden Layer

# of Neurons	Case 1	Case 2 (Baseline)	Case 3	Case 4
	300	500	800	1,000

<그림 3-77>은 Neuron test case에 따른 형상/성능 오차의 정규 분포도이며, <표 3-98>은 테스트 결과를 바탕으로 파라미터 별 각 지점의 오차를 측정하여 최소 오차가 나타난 Test case 및 빈도수를 정리한 것이다. 형상 오차의 경우 Case 4가 가장 낮은 오차를 많이 기록하였으나, 성능 오차의 경우 Case 1이 좋은 분포를 보여주었다. Epoch Test와 마찬가지로, 성능 파라미터 오차가 준수한 **Case 1(300 Neurons per Hidden Layer)**을 최적 파라미터로 선정하였다. 전체 형상/성능 파라미터들의 각 Percentile rank 오차는 <표 3-99>~<표 3-106>에 정리되어 있다.

<그림 3-78>, <그림 3-79>는 각 파라미터들의 Test case를 Percentile rank 별로 묶어 나타낸 그래프이다. 형상 오차는 Case 4가 준수한 분포를 보여주고, Case 1은 성능 파라미터에서 좋은 모습을 보이는 것을 알 수 있다. 다만, Cl0는 <표 3-98>에서 Case 4가 좋은 모습을 보이나, <그림 3-78>을 보면 그 차이는 크지 않다. Neuron 테스트의 모든 평균과 표준편차 데이터는 272p의 Appendix E.3에서 확인할 수 있다.



<그림 3-77> FHO: Neuron 수에 따른 형상/성능 오차의 정규분포

<표 3-97> FHO – Neuron Test: 소요 시간

Time(Hr)	Case			
	1	2	3	4
Training	1.3	2.07	3.53	4.56
Analysis	4.15	4.18	4.13	4.14
Total	5.45	6.25	7.66	8.7

<표 3-98> FHO – Neuron Test: 최소 오차가 나타난 Test Case와
가장 많이 나타난 Test Case(Mode)

Parameter	Case of Minimum Error					Mode
	99%	95%	90%	85%	80%	
Max. Camber	4	4	4	4	4	4
Max. Camber Location	3	4	2	2	3	3
Max. Thickness	1	4	4	4	4	4
Cl0	1	3	4	4	4	4
Cla0	1	1	1	1	1	1
Cd0	3	1	1	1	1	1
Cm0	4	1	1	1	1	1
Cma0	1	1	1	1	1	1
Mode	1	1	1	1	1	1

<표 3-99> FHO – Neuron Test Percentile Rank: Max. Camber

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	6.215.E-02	3.377.E-02	2.492.E-02	2.036.E-02	1.723.E-02
2	7.409.E-02	4.170.E-02	3.253.E-02	2.731.E-02	2.335.E-02
3	5.750.E-02	3.254.E-02	2.515.E-02	2.093.E-02	1.803.E-02
4	5.678.E-02	2.891.E-02	2.142.E-02	1.757.E-02	1.493.E-02

<표 3 -100> FHO – Neuron Test Percentile Rank: Max. Camber Location

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	1.731.E-01	6.421.E-02	4.190.E-02	3.338.E-02	2.791.E-02
2	1.690.E-01	6.070.E-02	3.846.E-02	3.043.E-02	2.565.E-02
3	1.614.E-01	6.239.E-02	3.930.E-02	3.047.E-02	2.511.E-02
4	1.769.E-01	5.980.E-02	4.201.E-02	3.406.E-02	2.920.E-02

<표 3 -101> FHO – Neuron Test Percentile Rank: Max. Thickness

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	4.541.E-01	2.303.E-01	1.633.E-01	1.301.E-01	1.090.E-01
2	5.012.E-01	2.503.E-01	1.781.E-01	1.437.E-01	1.225.E-01
3	4.722.E-01	2.426.E-01	1.723.E-01	1.399.E-01	1.200.E-01
4	4.797.E-01	2.241.E-01	1.591.E-01	1.244.E-01	1.031.E-01

<표 3 -102> FHO – Neuron Test Percentile Rank: Cl0

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	1.18.E-02	4.60.E-03	3.20.E-03	2.60.E-03	2.20.E-03
2	1.42.E-02	5.10.E-03	3.80.E-03	3.10.E-03	2.70.E-03
3	1.28.E-02	4.30.E-03	3.20.E-03	2.60.E-03	2.30.E-03
4	1.26.E-02	4.30.E-03	3.10.E-03	2.50.E-03	2.10.E-03

<표 3 -103> FHO – Neuron Test Percentile Rank: Cla0

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	2.04.E-02	3.40.E-03	1.70.E-03	1.00.E-03	7.00.E-04
2	2.33.E-02	3.80.E-03	1.80.E-03	1.10.E-03	7.00.E-04
3	2.23.E-02	3.50.E-03	1.70.E-03	1.10.E-03	7.00.E-04
4	2.34.E-02	3.60.E-03	1.80.E-03	1.20.E-03	8.00.E-04

<표 3 -104> FHO – Neuron Test Percentile Rank: Cd0

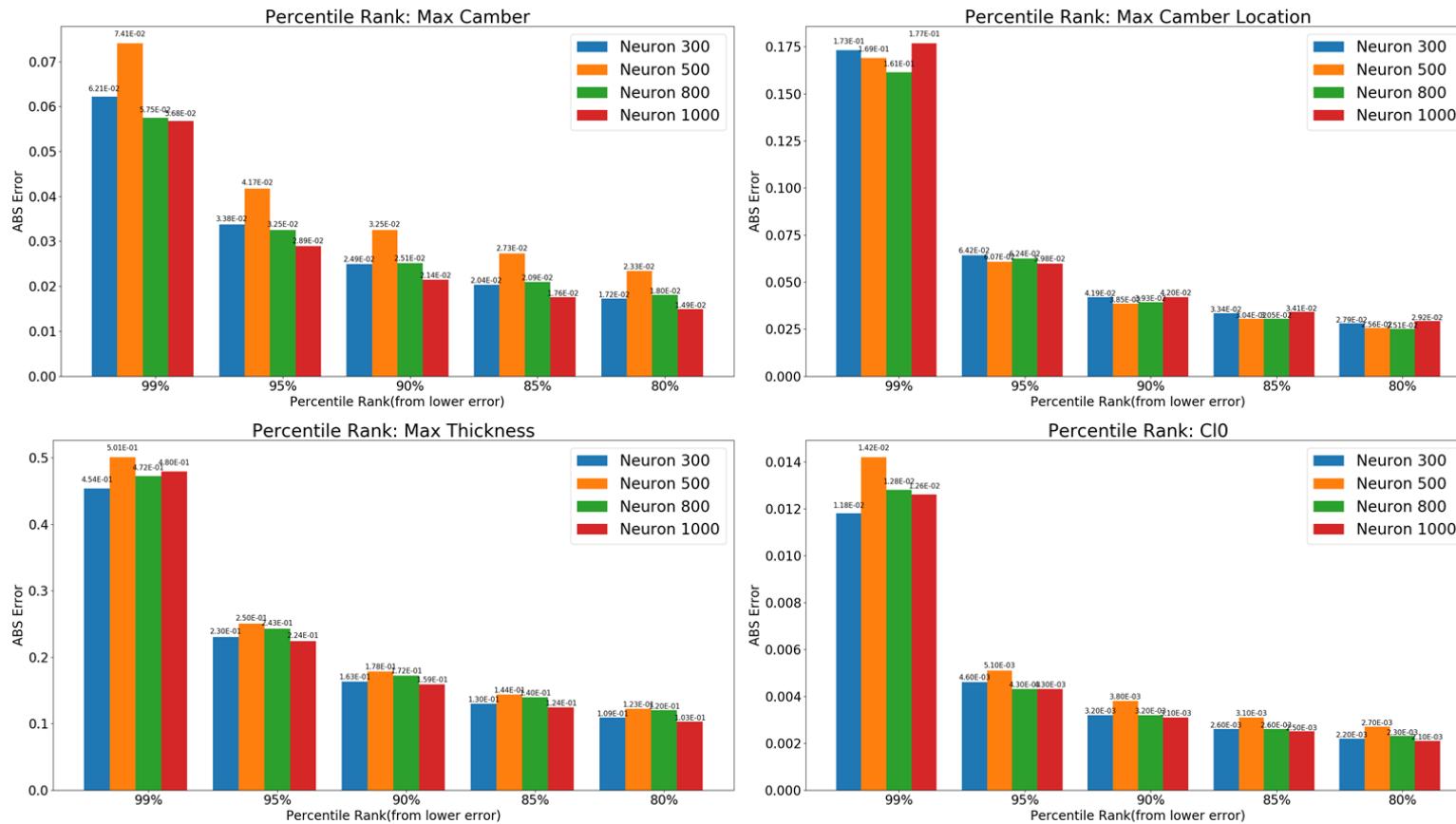
Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	4.9.E-04	1.1.E-04	6.0.E-05	4.0.E-05	3.0.E-05
2	6.1.E-04	1.5.E-04	7.0.E-05	5.0.E-05	4.0.E-05
3	4.6.E-04	1.2.E-04	7.0.E-05	4.0.E-05	3.0.E-05
4	4.7.E-04	1.2.E-04	6.0.E-05	4.0.E-05	3.0.E-05

<표 3 -105> FHO – Neuron Test Percentile Rank: Cm0

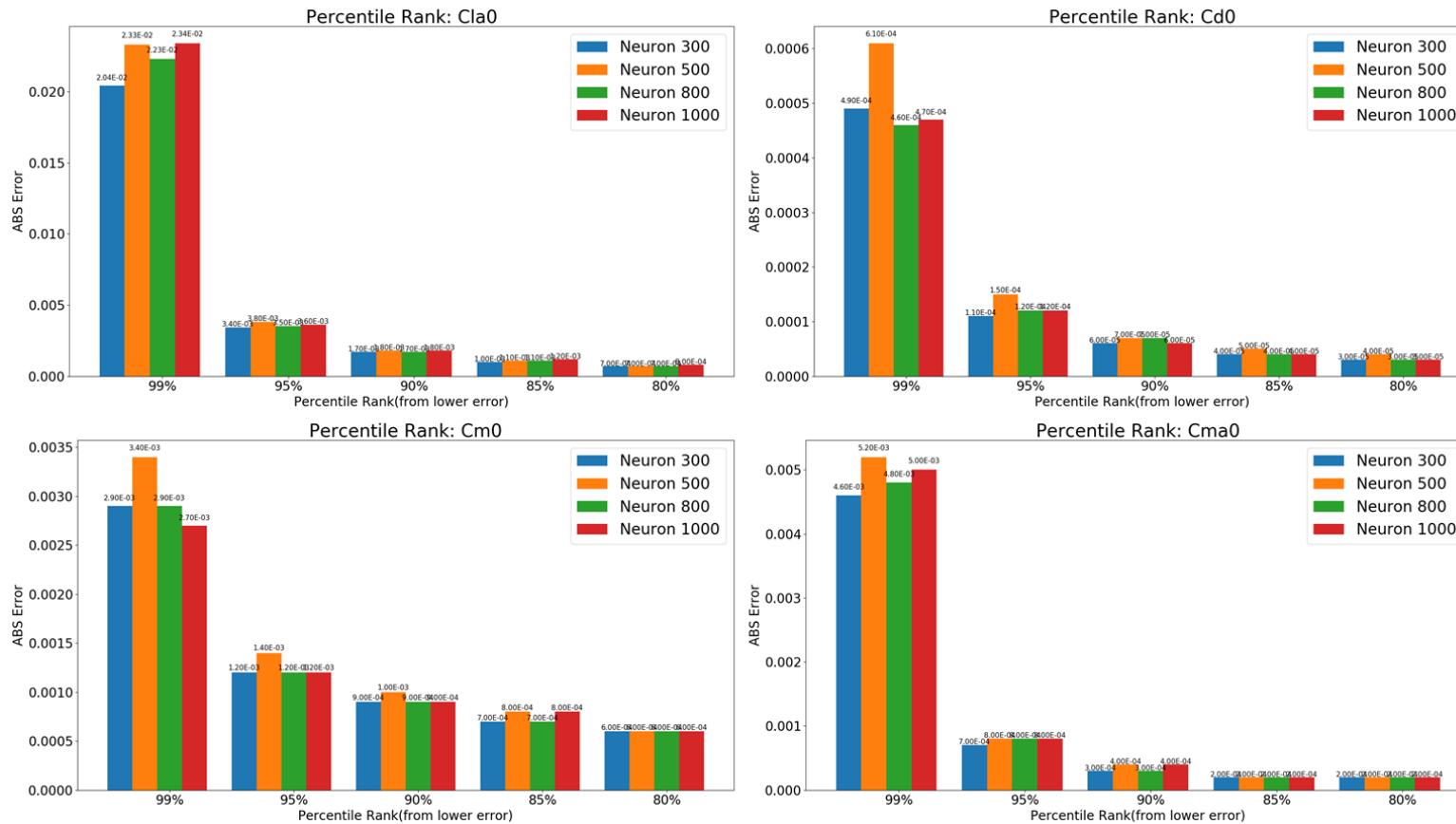
Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	2.9.E-03	1.2.E-03	9.0.E-04	7.0.E-04	6.0.E-04
2	3.4.E-03	1.4.E-03	1.0.E-03	8.0.E-04	6.0.E-04
3	2.9.E-03	1.2.E-03	9.0.E-04	7.0.E-04	6.0.E-04
4	2.7.E-03	1.2.E-03	9.0.E-04	8.0.E-04	6.0.E-04

<표 3 -106> FHO – Neuron Test Percentile Rank: Cma0

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	4.6.E-03	7.0.E-04	3.0.E-04	2.0.E-04	2.0.E-04
2	5.2.E-03	8.0.E-04	4.0.E-04	2.0.E-04	2.0.E-04
3	4.8.E-03	8.0.E-04	3.0.E-04	2.0.E-04	2.0.E-04
4	5.0.E-03	8.0.E-04	4.0.E-04	2.0.E-04	2.0.E-04



<그림 3-78> FHO – Neuron Test Percentile Rank:
Max. Camber, Max. Camber Location, Max. Thickness와 C10



<그림 3-79> FHO – Neuron Test Percentile Rank:
Cla0, Cd0, Cm0와 Cma0

7. 최적 하이퍼 파라미터를 이용한 학습

앞서 최종 하이퍼 파라미터 최적화 단계에서 Epoch, number of hidden layers, number of neurons per hidden layer에 대한 학습 최적 값을 구하였으며, 최적 값은 <표 3-107>과 같다. 나머지 하이퍼 파라미터들은 81p의 <표 3-8> 값을 사용하였다.

<표 3-107> 최적 하이퍼 파라미터

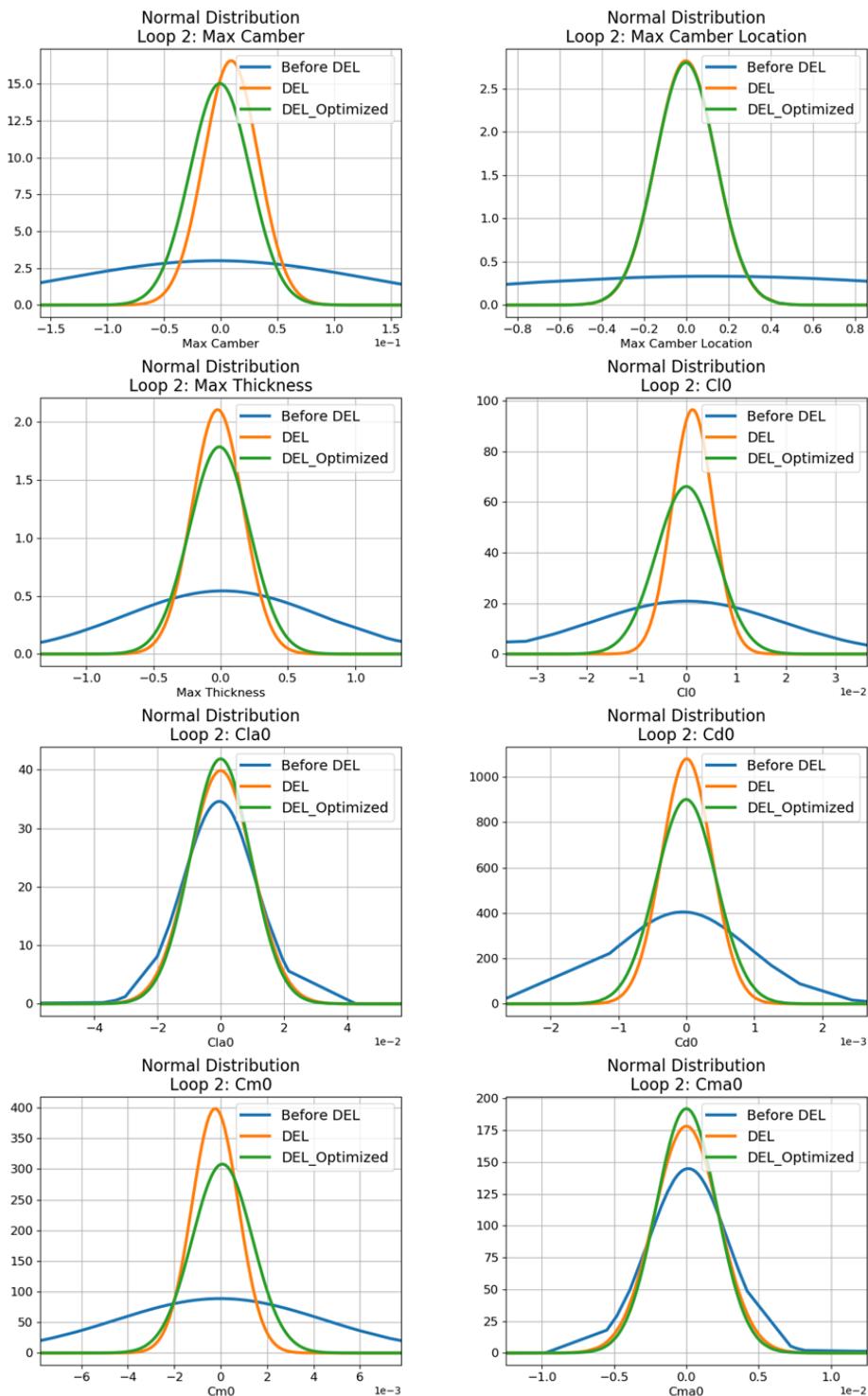
Hyper Parameter	Epoch	# of Hidden Layer	# of Neurons per Hidden Layer
	100,000	5	300

다음 페이지의 <그림 3-80>은 데이터베이스 강화 전인 Loop 1의 학습 테스트 결과(Before DEL), 데이터베이스 강화 후인 Loop 20의 학습 테스트 결과(DEL), 그리고 데이터베이스 강화 후에 하이퍼 파라미터들을 최적화한 뒤 학습 테스트를 한 결과(DEL_Optimized)들의 오차 정규분포이다.

데이터베이스 강화를 실시하면 오차의 분포가 0 근처에 모이게 되며, 하이퍼 파라미터를 최적화한 뒤에는 오차들의 평균이 0에 더 가까워지는 것을 알 수 있다. 그러나 하이퍼 파라미터 최적화는 많은 형상/성능 지표의 표준편차를 키우는 결과를 낳았다. 자세한 평균과 표준편차는 <표 3-118>~<표 3-121>에 형상 그룹별로 정리되어 있다.

앞서 최종 하이퍼 파라미터 최적화와 마찬가지로, 각 Test case 별 Percentile Rank를 측정했으며, 이는 <표 3-110>~<표 3-117>과 같다. 가장 낮은 값을 기록한 Test case는 <표 3-98>과 같다. Case 3(데이터베이스 강화 + 하이퍼 파라미터 최적화)가 전체 40개의 측정 지점 중 33개의 지점에서 가장 낮은 오차를 기록했다.

데이터베이스 강화 루프는 Percentile rank 오차를 최소 50%, 최대 98%까지 줄였으며, 하이퍼 파라미터 최적화는 오차를 최대 40% 줄였다. 데이터베이스 강화 루프 이후, 하이퍼 파라미터 최적화 실시한 결과는 데이터베이스 강화 이전보다 오차를 50~98%까지 줄인 것을 볼 수 있었다.



<그림 3-80> DEL 전, DEL 후, DEL과 파라미터 최적화 후의 정규분포

<표 3-108> Final Training Case 분류

Case 1	Case 2	Case 3
Before DEL	After DEL	DEL + Optimization

<표 3-109> Final Training: 최소 오차가 나타난 Test Case와
가장 많이 나타난 Test Case(Mode)

Parameter	Case of Minimum Error					Mode
	99%	95%	90%	85%	80%	
Max. Camber	3	3	3	3	3	3
Max. Camber Location	2	3	3	3	3	3
Max. Thickness	3	3	3	3	3	3
Cl0	3	3	3	3	3	3
Cla0	3	3	3	3	2	3
Cd0	3	3	3	3	3	3
Cm0	3	3	3	3	2	3
Cma0	3	2	2	2	2	2
Mode	3	3	3	3	3	3

<표 3-110> Final Training Percentile Rank: Max. Camber

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	4.921.E-01	1.689.E-01	1.231.E-01	9.305.E-02	7.463.E-02
2	7.409.E-02	4.170.E-02	3.253.E-02	2.731.E-02	2.335.E-02
3	5.631.E-02	2.903.E-02	2.111.E-02	1.689.E-02	1.401.E-02

<표 3-111> Final Training Percentile Rank: Max. Camber Location

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	4.998.E+00	3.414.E+00	1.167.E+00	2.557.E-01	1.563.E-01
2	1.690.E-01	6.070.E-02	3.846.E-02	3.043.E-02	2.565.E-02
3	1.707.E-01	5.396.E-02	3.449.E-02	2.729.E-02	2.288.E-02

<표 3 -112> Final Training Percentile Rank: Max. Thickness

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	2.447.E+00	9.435.E-01	6.021.E-01	4.657.E-01	3.813.E-01
2	5.012.E-01	2.503.E-01	1.781.E-01	1.437.E-01	1.225.E-01
3	4.493.E-01	2.083.E-01	1.428.E-01	1.122.E-01	9.388.E-02

<표 3 -113> Final Training Percentile Rank: Cl0

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	9.570.E-02	2.280.E-02	1.490.E-02	9.900.E-03	8.400.E-03
2	1.420.E-02	5.100.E-03	3.800.E-03	3.100.E-03	2.700.E-03
3	1.400.E-02	4.800.E-03	3.200.E-03	2.500.E-03	2.000.E-03

<표 3 -114> Final Training Percentile Rank: Cla0

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	6.04.E-02	1.18.E-02	5.50.E-03	3.00.E-03	2.10.E-03
2	2.33.E-02	3.80.E-03	1.80.E-03	1.10.E-03	7.00.E-04
3	2.01.E-02	3.60.E-03	1.70.E-03	1.00.E-03	7.00.E-04

<표 3 -115> Final Training Percentile Rank: Cd0

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	3.2.E-03	6.1.E-04	3.1.E-04	1.8.E-04	1.2.E-04
2	6.1.E-04	1.5.E-04	7.0.E-05	5.0.E-05	4.0.E-05
3	5.5.E-04	1.2.E-04	6.0.E-05	4.0.E-05	3.0.E-05

<표 3-116> Final Training Percentile Rank: Cm0

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	2.1.E-02	6.0.E-03	3.8.E-03	2.9.E-03	2.4.E-03
2	3.4.E-03	1.4.E-03	1.0.E-03	8.0.E-04	6.0.E-04
3	3.2.E-03	1.3.E-03	9.0.E-04	7.0.E-04	6.0.E-04

<표 3-117> Final Training Percentile Rank: Cma0

Case	Percentile Rank				
	99%	95%	90%	85%	80%
1	1.4.E-02	2.8.E-03	1.3.E-03	6.0.E-04	4.0.E-04
2	5.2.E-03	8.0.E-04	4.0.E-04	2.0.E-04	2.0.E-04
3	4.5.E-03	8.0.E-04	4.0.E-04	2.0.E-04	2.0.E-04

<표 3-118> Final Test: 형상 오차의 평균

Shape Group	Case	Max. Camber	Max. Camber Location	Max Thickness
Total	1	-0.0034770	0.1165170	0.016773
	2	0.0092310	-0.0025700	-0.022929
	3	-0.0006340	-0.0024230	-0.009435
Normal Range	1	-0.003183	-0.021154	-0.053103
	2	0.013791	-0.008943	-0.012170
	3	-0.002180	-0.003665	-0.013528
Zero Max. Camber	1	0.000000	0.957772	0.021078
	2	0.000015	0.012899	0.001608
	3	0.000094	0.012020	-0.023530
Out of Normal Range	1	-0.004076	-0.000916	0.025294
	2	0.009722	-0.003600	-0.02789
	3	-0.000483	-0.004144	-0.006905

<표 3-119> Final Test: 성능 오차의 평균

Shape Group	Case	Cl0	Cla0	Cd0	Cm0	Cma0
Total	1	1.10.E-04	-4.30.E-04	-5.10.E-05	-2.30.E-05	1.30.E-04
	2	1.24.E-03	-9.00.E-06	6.00.E-06	-2.38.E-04	2.00.E-06
	3	-2.30.E-05	-2.00.E-05	-3.00.E-06	7.20.E-05	4.00.E-06
Normal Range	1	-1.41.E-03	-1.06.E-04	6.00.E-06	4.50.E-04	3.10.E-05
	2	1.25.E-03	-1.00.E-06	8.00.E-06	-1.85.E-04	0.00.E+00
	3	-5.66.E-04	-4.00.E-06	0.00.E+00	1.92.E-04	3.00.E-06
Zero Max. Camber	1	-7.0.E-06	7.7.E-05	4.0.E-06	0.0.E+00	-1.1.E-05
	2	0.0.E+00	-2.4.E-05	-2.7.E-05	0.0.E+00	2.0.E-06
	3	1.0.E-05	-1.6.E-05	-1.0.E-05	-3.0.E-06	0.0.E+00
Out of Normal Range	1	3.29.E-04	-5.55.E-04	-6.70.E-05	-8.90.E-05	1.66.E-04
	2	1.40.E-03	-8.00.E-06	1.00.E-05	-2.78.E-04	2.00.E-06
	3	6.00.E-05	-2.30.E-05	-2.00.E-06	6.30.E-05	4.00.E-06

<표 3-120> Final Test: 형상 오차의 표준편차

Shape Group	Case	Max. Camber	Max. Camber Location	Max Thickness
Total	1	0.1326970	1.2007170	0.733579
	2	0.0240650	0.1413080	0.189559
	3	0.0265440	0.1424500	0.223517
Normal Range	1	0.048676	0.067661	0.478457
	2	0.017869	0.017653	0.157820
	3	0.014324	0.018343	0.153167
Zero Max. Camber	1	0.000000	3.263754	0.016585
	2	0.000422	0.434466	0.047364
	3	0.001018	0.435425	0.042117

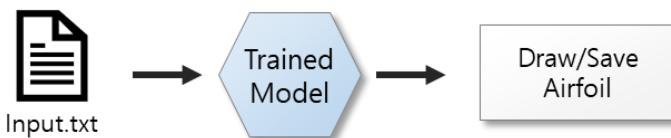
Out of Normal Range	1	0.149846	0.128161	0.815404
	2	0.026122	0.027221	0.204999
	3	0.029624	0.030813	0.246114

<표 3-121> Final Test: 성능 오차의 표준편차

Shape Group	Case	Cl0	Cla0	Cd0	Cm0	Cma0
Total	1	1.914.E-02	1.154.E-02	9.860.E-04	4.500.E-03	2.754.E-03
	2	4.137.E-03	1.001.E-02	3.700.E-04	1.001.E-03	2.239.E-03
	3	6.035.E-03	9.529.E-03	4.430.E-04	1.295.E-03	2.079.E-03
Normal Range	1	6.19.E-03	7.37.E-04	5.60.E-05	1.69.E-03	1.60.E-04
	2	1.74.E-03	4.01.E-04	1.60.E-05	4.56.E-04	9.20.E-05
	3	1.71.E-03	4.24.E-04	1.14.E-04	4.89.E-04	9.60.E-05
Zero Max. Camber	1	3.9.E-05	9.7.E-05	1.7.E-05	0.0.E+00	3.2.E-05
	2	7.3.E-05	2.3.E-04	4.7.E-04	1.3.E-05	3.4.E-05
	3	1.3.E-04	1.7.E-04	3.0.E-04	2.9.E-05	3.3.E-05
Out of Normal Range	1	2.16.E-02	1.31.E-02	1.12.E-03	5.08.E-03	3.13.E-03
	2	4.63.E-03	1.14.E-02	3.84.E-04	1.12.E-03	2.54.E-03
	3	6.83.E-03	1.08.E-02	4.90.E-04	1.46.E-03	2.36.E-03

제 3절 DLED NACA 4-digit Design 사용자 알고리즘

학습이 완료된 인공신경망 모델은 언제든지 다시 불러들여 사용할 수 있다. 앞서 데이터의 확보와 학습, 그리고 모델의 최적화에는 매우 많은 시간이 소요되었지만, 학습이 완료된 모델은 실시간으로 예측이 가능하다. 본 연구에서 사용한 설계 모델의 경우, 입력값을 넣으면 형상을 만들어내는데 2ms(2/1,000 seconds) 내외가 소요된다. 이는 전통적인 최적화 알고리즘이 수 시간에서 수일이 걸리던 것과 비교하면 획기적으로 줄어든 것이다. 그러나 사용자가 이를 쉽게 사용하기 위해서는 입출력 파라미터만 입력하면 알고리즘을 사용할 수 있도록 추가적인 개발이 필요하다.



<그림 3-81> 학습이 완료된 알고리즘을 사용하는 방법

DLED 알고리즘은 사용자 편의성을 높이기 위해 Python script 기반의 사용자 알고리즘과 여기에 사용자 인터페이스(User Interface, UI)를 결합한 프로그램을 개발하였다. 사용자 인터페이스는 어떤 물건과 사람이 상호작용을 할 수 있도록 하는 공간/환경을 말하며 [60], 키보드나 마우스와 같은 물리적인 인터페이스만 아니라, 프로그램과 사용자를 이어주는 그래픽 기반의 GUI(Graphical User Interface)나 텍스트 기반의 TUI(Text-based User Interface), Command-line Interface 역시 사용자 인터페이스에 속한다.

TUI는 Command-line Interface와 헷갈리기 쉬운 인터페이스로, 과거에 많이 사용된 DOS 환경이나 컴퓨터의 BIOS 설정 화면에서 찾아볼 수 있다. 라인을 한 줄씩 써가며 진행해야 하는 1차원적인 Command-line Interface와는 다르게, TUI는 <그림 3-82>처럼 텍스트 기반이긴 해도 2차원적인 평면에 기능들을 넣을 수 있었다. 이후, GUI가 주류가 되고 마우스가 새로운 입력 인터페이스로 자리잡으면서 TUI도 마우스를 이용하거나 버튼처럼 동작하도록 하는 등 GUI의 특징을 들여오기도 하였다. [61]

Left	File	Command	Options	Right		
Name	Size	MTime		Name	Size	MTime
/software				/etc		
..	4096	Oct 2 04:02		..	4096	Oct 2 04:02
/ICAClient-3.0	2048	Jan 6 2003		./java	30	May 13 2004
/aida-2.1.1	2048	Apr 28 2003		/ada	4096	Aug 9 2001
/amber-6.0	2048	Feb 27 2004		/conf	151	Jul 19 2000
/amber-7.0	2048	Mar 5 2004		/config	4096	Dec 13 2004
/amber-7.0p	2048	Apr 16 2004		/cron.d	133	Sep 29 20:23
/amber-8	2048	Dec 22 2004		/default	75	Aug 12 2004
~ansys61	34	Jan 7 2003		/dt	27	Apr 5 2003
~ansys71	34	Nov 28 2003		/fscklogs	39	Aug 3 2000
/ant-1.6	2048	Aug 10 13:26		~fstyp.d	15	Apr 25 2000
/apache-1.3.27	2048	Dec 16 2002		~httpd	20	Jul 19 2000
/apache-1.3.28	2048	Jan 6 2004		/init.d	4096	Sep 21 15:45
/apache-1.3.33	2048	Feb 7 2005		/js	4096	Aug 9 2001
/autoconf-2.57	2048	May 27 2004		/lost+found	4096	Oct 8 2004
/autodock-305	2048	Jan 5 2001		/mail	4096	May 2 10:04
/ICAClient-3.0				/cron.d		

Hint: Keys not working in xterms? Use our xterm.ad, .ti and .tcap files.
aisa:/software> \$ [^]
1Help 2Menu 3View 4Edit 5Copy 6RenMov 7Mkdir 8Delete 9PullDn 10Quit

<그림 3-82> TUI 예시(파일관리자) [61]

XFOIL [5]은 본 연구에서도 사용한 Command-line Interface 프로그램으로, 매 단계를 명령어를 사용해서 진행해야 한다. <그림 3-83>은 XFOIL의 실행화면으로, 명령어가 화면에 나열되어 있는 것을 볼 수 있다. 해당 명령어들은 일반적인 GUI나 TUI처럼 선택이 가능한 것이 아니라, 직접 사용자가 형식에 맞추어 입력하여 실행시켜야 한다. 만약 입력해야 할 명령어가 파일 이름과 같은 내용이 포함되어 길어지게 되면 오타가 나기 쉬운데, 이 경우 오류가 발생하므로 사용자는 명령어를 다시 입력해야 한다.

```
BEND    Display structural properties of current airfoil
PCOP    Set current-airfoil panel nodes directly from buffer airfoil points
PANE   Set current-airfoil panel nodes ( 160 ) based on curvature
.PPAR   Show/change paneling
.PLOP   Plotting options
WDEF f  Write current-settings file
RDEF f  Reread current-settings file
NAME s  Specify new airfoil name
NINC   Increment name version number
Z       Zoom      | (available in all menus)
U       Unzoom   |
XFOIL  c> NACA 2412
```

<그림 3-83> Command-line Interface인 XFOIL [5]

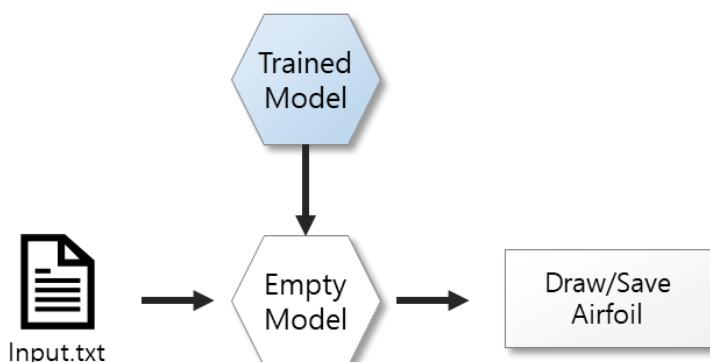
본 연구에서는 사용 편의성을 위해 GUI를 최종적으로 사용하였다. GUI는 개발하는 과정이 TUI나 Command-line Interface보다 어렵지만, 이들보다 직관적이며 사용하기 편리하다. GUI 개발 도구로는 Python의 라이브러리 중 하나인 PyQt5 [62]를 사용하였다.

1. Python Script 기반 알고리즘

Tensorflow로 학습된 모델을 저장하면 <그림 3-84>와 같이 네 개의 파일이 생성된다. 그러나 이 파일들만으로는 학습된 모델을 사용할 수 없다. <그림 3-84>의 파일들에는 모델의 정보만 담겨있기 때문에, 이들을 사용하기 위해서는 <그림 3-85>처럼 정보를 담을 그릇을 만들어 주어야 한다. 주의할 점은 모델의 구조는 학습된 모델과 동일하게 만들어주어야 하며, 모양이 다르면 오류가 발생하여 모델을 만들지 못하거나, 만들어지더라도 제대로 된 기능을 하지 못한다.

- ❑ checkpoint
- ❑ model20.data-00000-of-00001
- ❑ model20.index
- ❑ model20.meta

<그림 3-84> 학습이 완료된 인공신경망 모델 파일

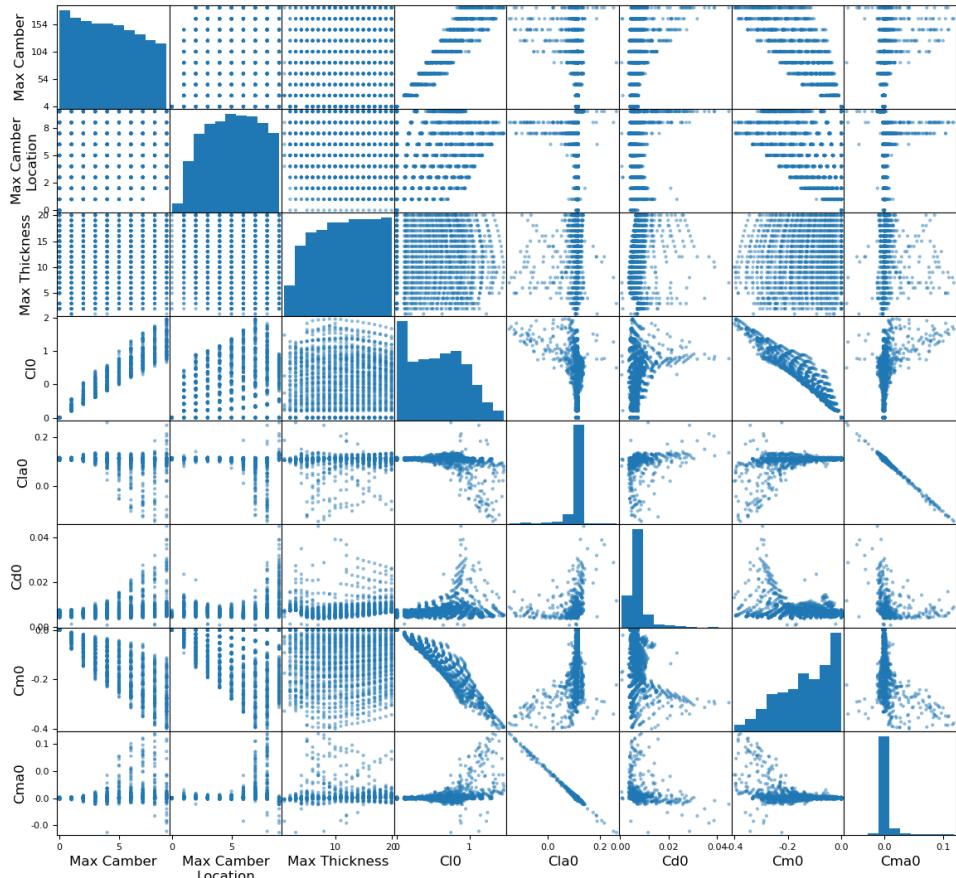


<그림 3-85> 학습이 완료된 모델을 사용하는 방법

2. 에어포일 데이터베이스 분석

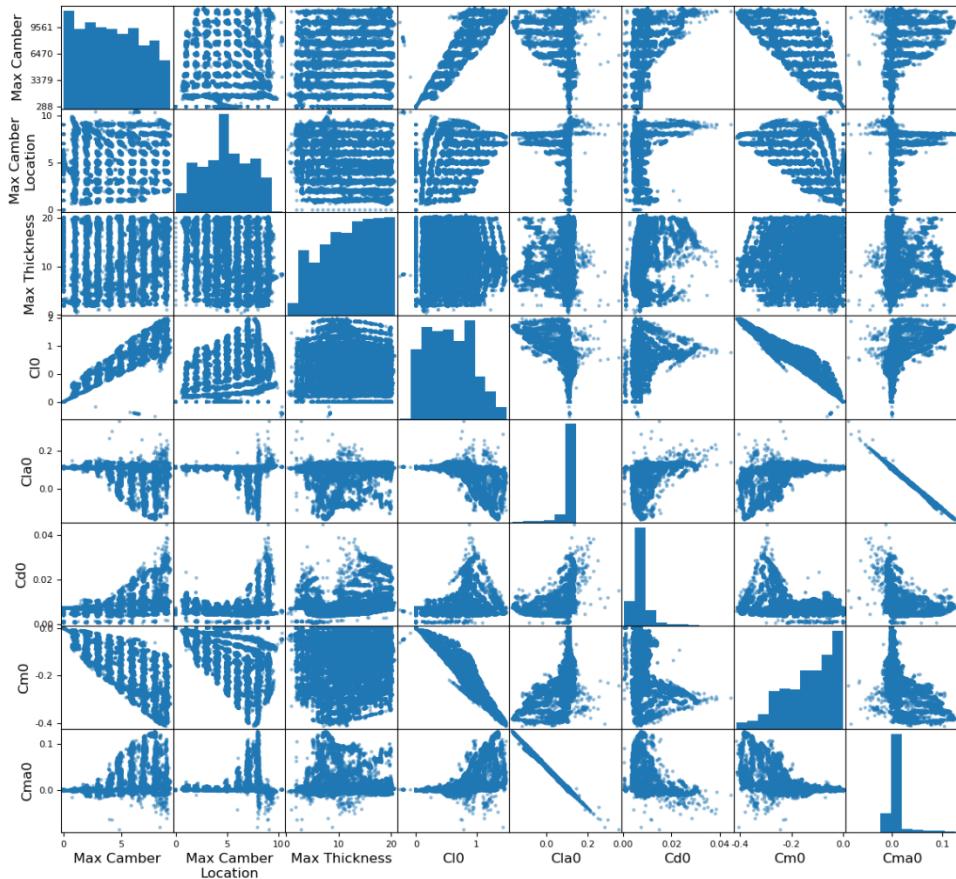
앞서 DLED: NACA 4-digit 학습 과정에서 NACA 4-digit 에어포일 데이터를 111,494개까지 모았다. 이들의 분포를 Scatter matrix를 이용하여 살펴보면 각 파라미터 사이의 관계를 알 수 있다. 아래의 <그림 3-86>은 Loop 1의 에어포일 DB의 Scatter matrix이며, <그림 3-87>은 20th Loop의 Scatter matrix이다.

Scatter Matrix_Loop1 Airfoil DB



<그림 3-86> Loop 1 에어포일 DB의 Scatter Matrix

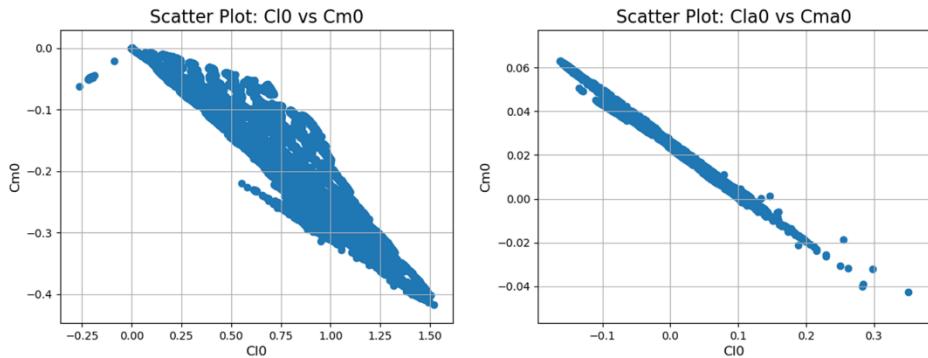
Scatter Matrix: Loop20 Airfoil DB



<그림 3-87> Loop 20 에어포일 DB의 Scatter Matrix

형상 데이터는 초기 DB 생성시, 균등한 분포로 만들었기에 전 영역에 걸쳐 고르게 분포하는 것을 알 수 있다. 그러나 성능 파라미터의 경우, 특정 영역에만 분포되어 있는 것을 볼 수 있다. 이는 NACA 4-digit으로 낼 수 있는 성능은 제약이 있다는 것이며, 이는 본 연구에서 학습시킨 인공 신경망의 예측 성능에 크게 영향을 미친다.

본 연구에서 인공신경망은 성능 파라미터로 NACA 4-digit 형상을 설계하도록 하였다. 그러나 입력 받은 성능이 NACA 4-digit으로는 낼 수 없는 조건이라면, 인공신경망은 능력 밖의 일을 받게 된 것이다.



<그림 3-88> NACA 4-digit Cl_0 vs Cm_0 관계(좌)

Cla_0 vs Cma_0 관계(우)

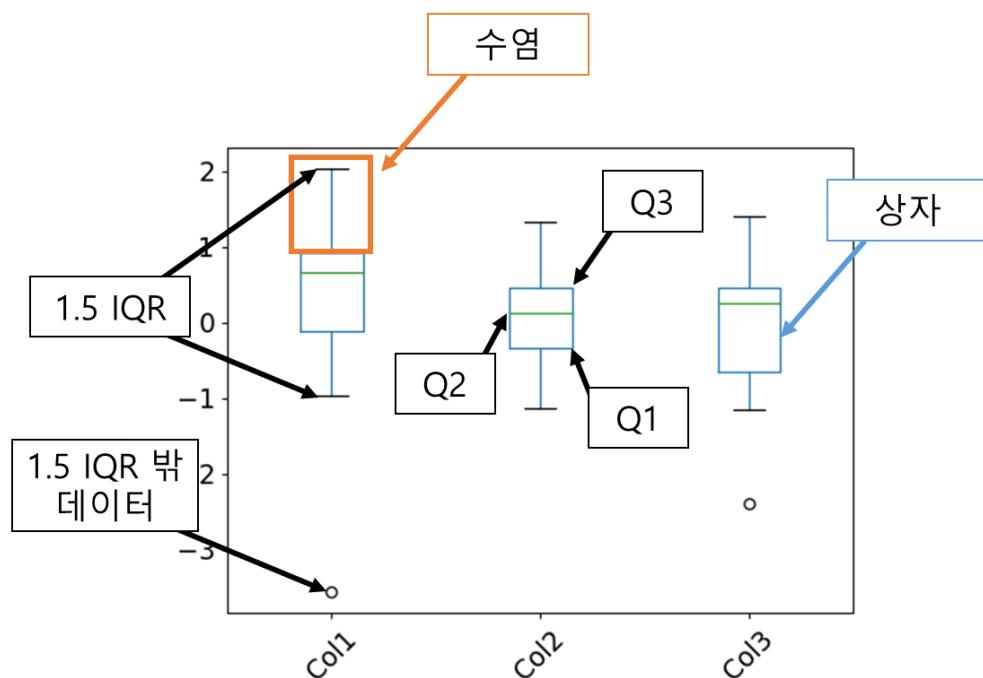
예를 들어 NACA 4-digit 익형들의 Cl_0 , Cm_0 는 <그림 3-88>과 같은 관계를 갖는다. 달리 말하자면 아무리 NACA 4-digit의 세 가지 형상 파라미터를 조합한다 할지라도 위 그래프의 영역 내의 성능밖에 낼 수 없다. 만약 사용자가 Cm_0 는 0, Cl_0 는 1인 NACA 4-digit 에어포일을 찾아내라 한다면 어떤 형상을 예측은 하겠지만, 해석을 하여도 성능은 요구조건과 동떨어질 것이다.

이에 대한 조치로 두 가지를 생각할 수 있다. 하나는 사용자의 파라미터 설정 자유도를 높여 NACA 4-digit으로는 낼 수 없는 성능도 입력받을 수 있도록 하는 것이며, 다른 하나는 입력값의 가이드라인을 만들어 특정 영역 내의 성능 값들만 입력하도록 유도하는 것이다. 자유도를 높이면 프로그램은 더 간단하고 사용자는 마음대로 입력값을 넣을 수 있겠지만, 설계 결과의 정확도는 담보하기 어렵다. 반대로 가이드라인을 만들면 프로그램은 더 복잡해지고 파라미터의 입력 가능한 범위가 줄어들지만, 설계의 정확도는 높일 수 있다. 본 연구에서 최종적으로 개발한 DLED 프로그램은 가이드라인을 만들어 설계 정확도를 높이도록 하였다.

3. 설계 가능 영역 선정

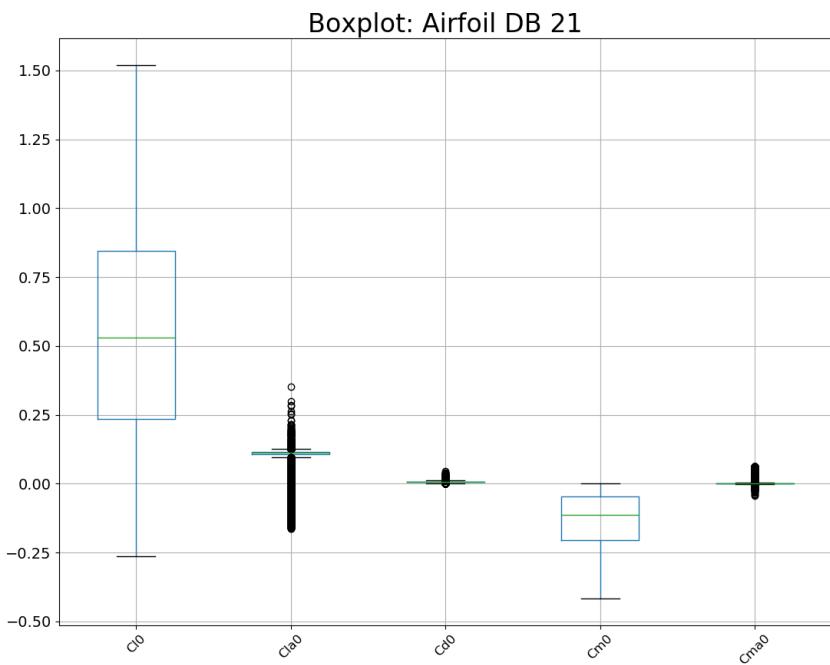
본 연구에서는 21번째 NACA 4-digit 에어포일 데이터베이스로 상자 수염 그래프(상자 그래프)를 그려, 데이터의 분포를 분석하였다. 상자 수염 그

그래프는 전체 데이터에서 1/4번째 있는 데이터(Q1)와 3/4번째에 있는 데이터(Q3)로 직사각형의 상자를 그리고, 중간인 2/4번째에 있는 데이터에 직선을 그는다. 그리고 Q3의 데이터 값과 Q1의 데이터 값의 차인 IQR을 구하고, IQR의 1.5배 지점까지 수직으로 직선을 그고, 1.5IQR 지점에서 가로로 직선을 그린다. 이 선들을 수염이라고 하며, 만약 데이터의 최대값과 최소값이 1.5IQR을 넘지 않는다면 최대 최소값 까지만 수염을 그린다. 그러나 데이터들 중, 1.5IQR을 넘는 데이터들이 있다면 이들은 원으로 표시한다. 아래 <그림 3-89>는 상자수염 그래프의 구성을 설명한 것이다. [63]



<그림 3-89> 상자수염 그래프 구성 [63]

본 연구에서는 각 성능 파라미터들의 1.5 IQR 값들로 21번째 에어포일 DB를 걸러내어 데이터베이스의 범위를 줄였다. 단, Cl0는 음수일 경우 그 의미가 크지 않아 0 이하의 값을 갖는 경우 제외시켰다. 각 성능 파라미터들의 상자수염 그래프는 <그림 3-90>과 같으며, 각 성능 파라미터의 1.5 IQR의 최대/최소 값은 <표 3-122>와 같다. 분석에 사용된 21번째 에어포일 DB는 111,493개의 NACA 4-digit 에어포일이 있다.



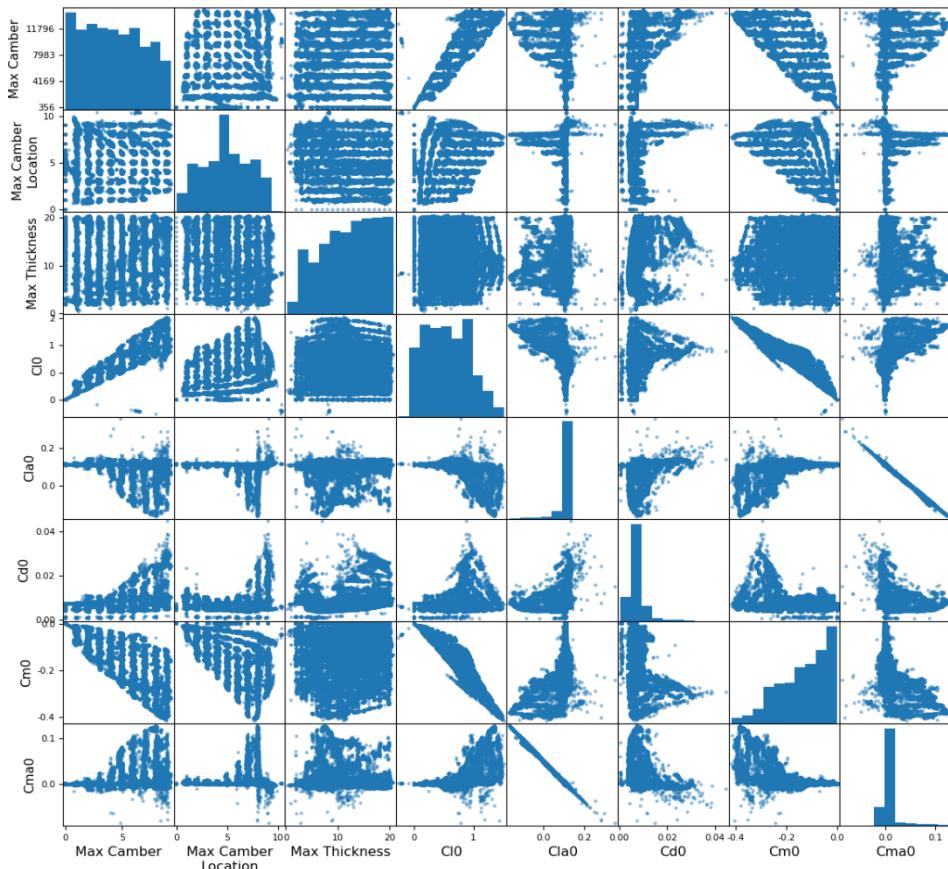
<그림 3-90> 21번째 데이터베이스의 성능 파라미터 상자수염 그래프

<표 3-122> 21번째 데이터베이스 성능 파라미터의 1.5 IQR 최대/최소값

	Cl0	Cla0	Cd0	Cm0	Cma0
1.5IQR	1.5195	0.1268	0.01168	0.0001	0.0036
3Q	0.8461	0.1147	0.00785	-0.0456	0.0014
2Q	0.5314	0.112	0.00637	-0.1131	0.0003
1Q	0.2364	0.1066	0.00529	-0.2054	-0.0001
-1.5IQR	-0.2628	0.0945	0.00146	-0.4169	-0.0023
Number of Outliers	0	17,888	5983	0	19,936
Portion of Outliers	0%	16.04%	5.37%	0%	17.88%

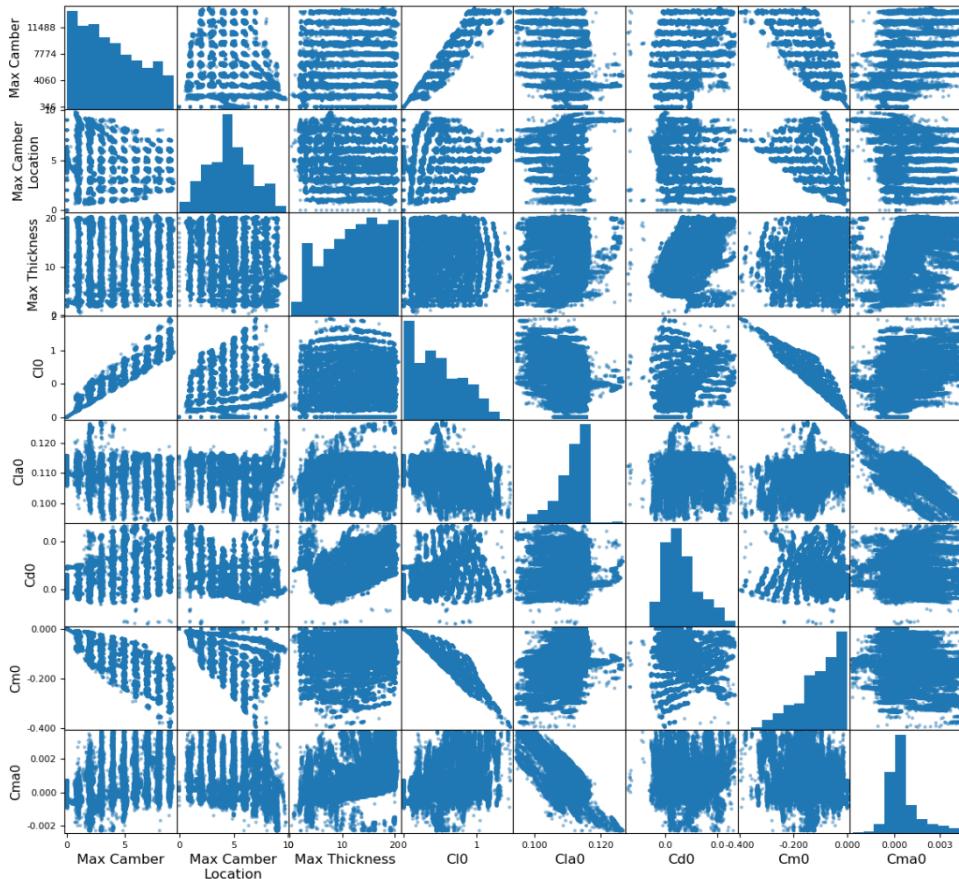
아래의 <그림 3-91>은 앞서 보인 20번째 에어포일 DB와 20번째 학습 테스트 결과를 재해석한 데이터들을 합친 데이터베이스인, 21번째 에어포일 DB Scatter matrix이다. 그리고 다음 페이지의 <그림 3-92>는 앞서 설명한 조건에 따라 걸러낸 데이터들의 Scatter matrix이다. 데이터를 골라 내기 전에도 C_{l0} 와 C_{m0} 사이의 관계나 C_{la0} 와 C_{ma0} 의 관계는 우측 아래로 내려가는 모습을 보이는 것을 알 수 있다. 그러나 C_{d0} 나 이 외의 관계들은 특별한 모양새를 찾기 어려웠다. 그러나 IQR 범위를 통해 데이터를 걸러내자, C_{d0} 의 영역을 사각형 모양으로 제한 지을 수 있게 되었다.

Scatter Matrix_Airfoil DB21



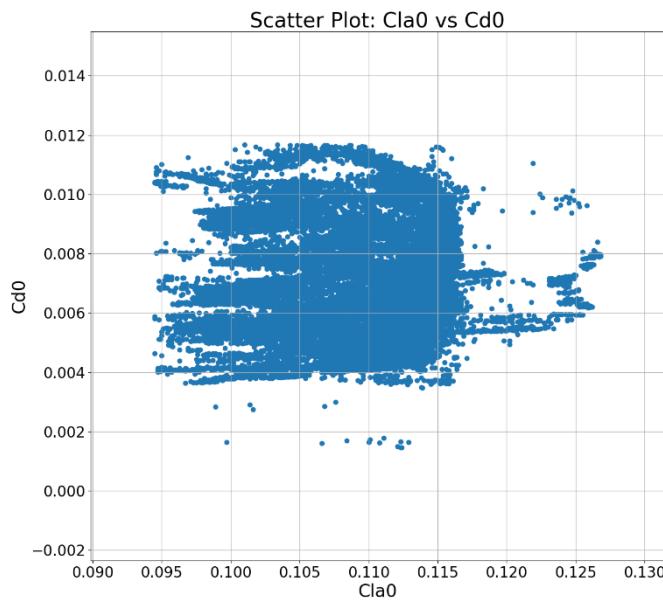
<그림 3-91> Airfoil DB 21의 Scatter Matrix

Scatter Matrix_Airfoil DB21 renew

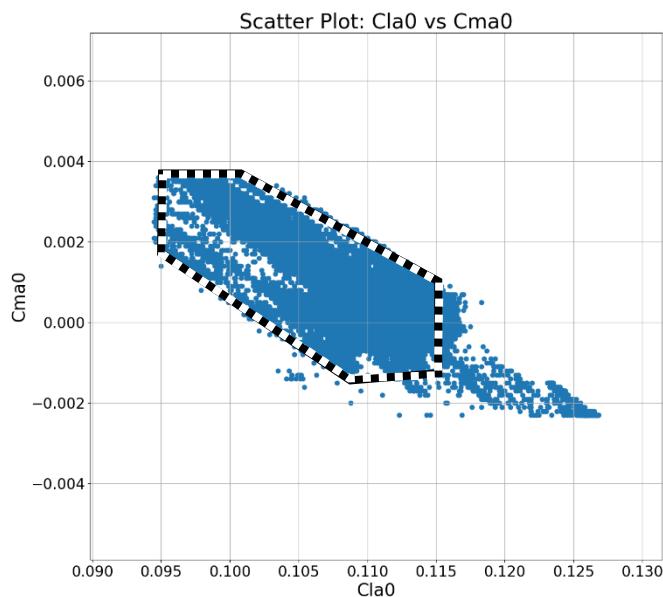


<그림 3-92> 조건에 맞추어 걸러낸 Airfoil DB 21 Scatter Matrix

위 그림에서 Cla_0 와 Cd_0 , 그리고 Cma_0 와 Cd_0 의 관계를 보면 세 성능 파라미터들의 경계 영역을 어느 정도 제한 지을 수 있다는 것을 알 수 있다. 예를 들어 아래의 <그림 3-93>을 보면 Cla_0 와 Cd_0 의 관계를 볼 수 있는데, 대부분의 데이터는 0.115 이하의 Cla_0 를 갖는다. 그리고 0.115 이하의 Cla_0 에서 <그림 3-94>처럼, Cla_0 와 Cma_0 의 관계는 보다 명료하게 나타낼 수 있다. 이처럼 분포 그래프를 통해 각 성능 파라미터의 최대와 최소값을 줄일 수 있으며, 반복적으로 시행하여 최종적으로는 <표 3-123>과 같이 설정하였다.



<그림 3-93> Cla0 vs Cd0 Scatter Plot



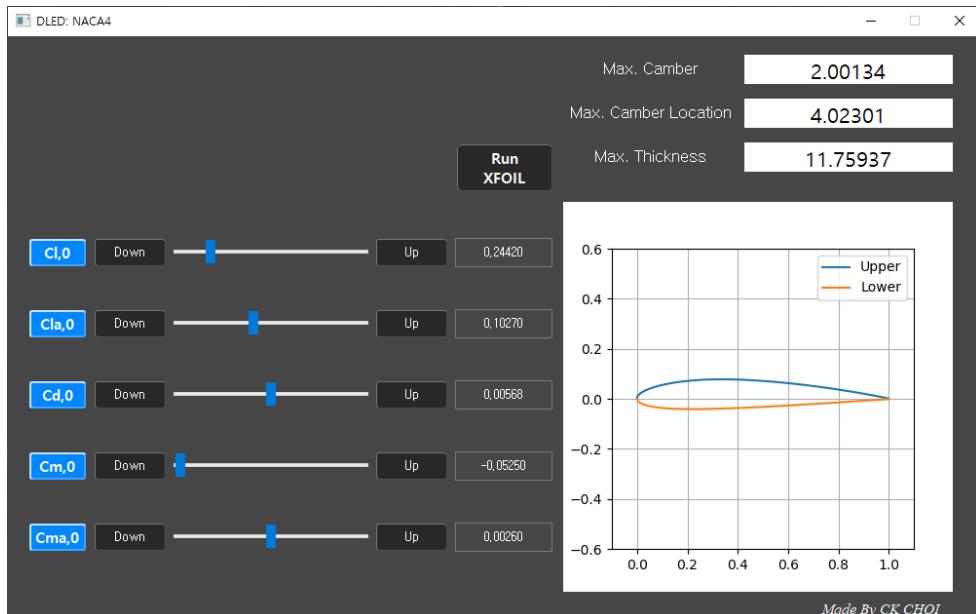
<그림 3-94> Cla0 vs Cma0 Scatter Plot

<표 3-123> 성능 파라미터의 최대 최소값

	Cl0	Cla0	Cd0	Cm0	Cma0
Max	1.5195	0.115	0.01168	0.0	0.0036
Min	0.0	0.095	0.004	-0.035	0.0020

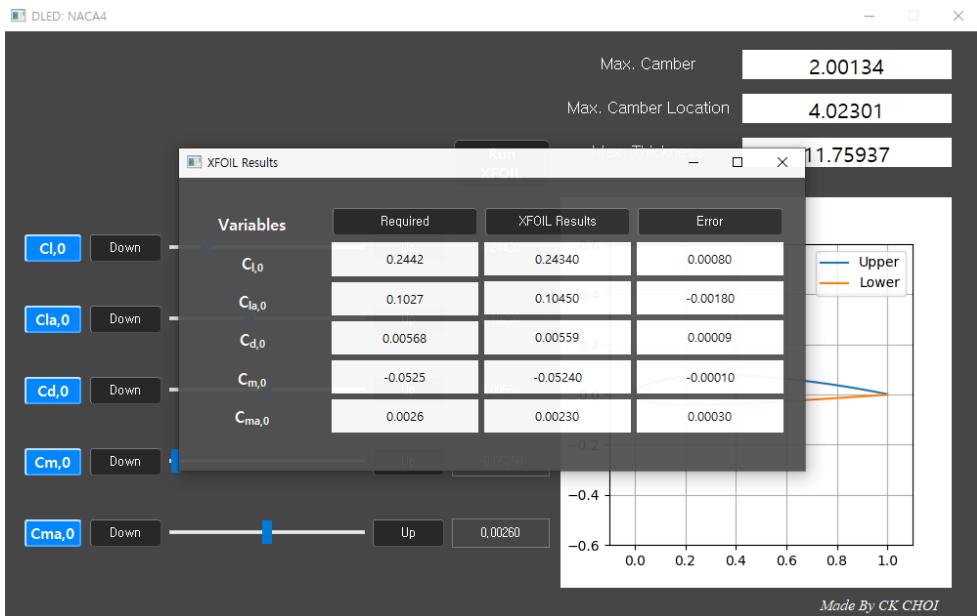
4. GUI 및 실행파일 생성

본 연구의 최종 결과물인 DLED: NACA 4-digit 알고리즘은 PyQt5 [62]로 GUI를 만들고, Pyinstaller [64]를 이용하여 실행가능한 exe 파일로 만들었다. GUI의 구성은 <그림 3-95>와 같다. 다섯가지 파라미터는 버튼과 슬라이드로 조절할 수 있으며, Up 버튼 옆의 입력칸에 사용자가 직접 값을 입력할 수도 있다. 성능 파라미터에 대한 설계 결과는 세 가지 파라미터가 우측 상단에 실시간으로 나타나며, 이에 따라 그 아래의 에어포일 모양도 역시 실시간으로 변하게 된다. 사용자 입력칸 위에는 ‘Run XFOIL’ 버튼이 있는데, 이를 누르면 현재 만들어진 에어포일의 성능을 XFOIL로 해석하여 볼 수 있다. 또한 해석을 해야만 저장이 되도록 하여, 설계 형상의 성능을 사용자가 사용하기 전에 항상 확인하도록 하였다.



<그림 3-95> DLED: NACA 4-digit (기본 화면)

<그림 3-96>은 Run XFOIL 버튼을 누르면 나타나는 창이며, 사용자 요구 사항과 XFOIL 해석 결과, 그리고 오차를 확인할 수 있다.



<그림 3-96> DLED: NACA 4-digit (Run XFOIL을 누른 화면)

제 4 장 결론 및 향후 계획

본 연구는 딥러닝 기법과 인공신경망을 이용하여 항공공학 설계의 기초인 에어포일 설계를 빠르고 직관적으로 할 수 있는 방법을 다루었다. 간단한 예시로 본 논문에서는 NACA 4-digit 에어포일을 사용하였으며, 인공신경망을 학습시키기 위해 XFOIL을 이용하여 형상-성능 데이터베이스를 구축하였다. 또한 설계 분야의 한계인 데이터 부족을 돌파하기 위해 인공신경망의 학습 결과를 XFOIL로 재해석하는 데이터베이스 강화 루프를 구현하였다. 데이터를 보다 잘 얻기 위해 데이터베이스 강화 루프 전에 한번 하이퍼 파라미터 최적화를 실시하였고, 데이터를 충분히 모은 뒤에는 최종 튜닝을 위해 다시 한번 더 하이퍼 파라미터를 최적화하였다. 최종적으로 만들어진 모델은 에어포일 데이터 분석을 통해 제약 조건을 설정였으며, GUI를 구현하여 사용자가 빠르고 쉽게 에어포일을 설계할 수 있도록 하였다. 이번 연구를 진행하며, 인공신경망과 NACA 4-digit 에어포일, 그리고 인공신경망을 이용한 에어포일 설계에 관해 다음과 같은 결론들을 낼 수 있었다.

- 인공신경망의 학습과 테스트
 - 본 연구에서 초기 학습 데이터를 사용하여 CPU 와 GPU 를 각각 사용하여 학습하였을 때, **GPU**를 사용하는 것이 8.5 배 빨랐다.
 - 본 연구에서는 낮은 유동조건($Re=1.0 \times 10^6$, $M=0.1$)과 높은 유동조건($Re=3.0 \times 10^6$, $M=0.2$)으로 나누어 학습을 실시하였다. 두 상태 중, 높은 유동조건을 이용하였을 때 XFOIL 해석을 통한 DB 확보가 용이할 뿐만 아니라 학습의 결과도 좋았다.
 - 인공신경망의 학습 테스트 오차는 15 개의 모델만으로도 충분히 정규분포를 그릴 수 있으며, 이를 바탕으로 인공신경망 테스트 오차의 기대값과 획득할 수 있는 오차의 확률을 예측할 수 있다.

- 데이터베이스 강화 루프
 - 공학적 설계 문제는 이미지 분류 등 다른 인공신경망 학습 문제에 비해 그 데이터가 절대적으로 부족하다. 본 연구에서는 부족한 설계 데이터를 모으기 위해 성능 해석 도구만 있으면 사용할 수 있는 데이터베이스 강화 루프를 도입하였다.
 - NACA 4-digit 은 대칭형과 비대칭형 에어포일로 나눌 수 있는데, 대칭형 에어포일은 최대 캠버의 값이 0 이면 최대 캠버 위치의 값은 어떤 숫자를 사용해도 형상에 영향을 미치지 못한다. 그러므로 대칭형/비대칭형 에어포일 사이에 형상-성능 파라미터가 불연속적이며, 이는 인공신경망의 학습을 어렵게 만든다.
 - 대칭형 에어포일의 최대 캠버 위치의 불연속적 특성으로 인해 초기 에어포일 데이터를 이용한 학습에서 MSE 가 크게 나타났다. 그러나 데이터의 증가가 불연속 지점에서의 학습 오차를 줄이는 것을 확인할 수 있다.
 - 비대칭형 에어포일은 일반적인 형상은 초기 데이터만으로 학습이 잘 되었다. 그러나 일반적이지 않은 형상은 초기 데이터를 사용한 학습 테스트 오차가 컸으며, 데이터베이스 강화로 오차를 줄어드는 것을 확인할 수 있었다.
 - 또한 데이터베이스 강화 루프는 형상 오차의 평균과 표준편차를 줄이는데 효과적이었으며, 성능 오차의 표준편차를 줄이는데 효과적이었다.
- 하이퍼 파라미터 최적화
 - 하이퍼 파라미터 최적화는 주어진 데이터로 더 나은 예측을 하는 모델을 만들 수 있는 방법이다.
 - 데이터베이스 강화 루프가 인공신경망의 오차의 표준편차를 줄이는데 효과가 있다면, 하이퍼 파라미터 최적화는 인공신경망의 예측 오차의 평균이 0에 가까워지도록 한다.

- 딥러닝 기반 NACA 4-digit 에어포일 설계 알고리즘
 - 딥러닝을 사용하면 에어포일의 성능 파라미터로 형상을 빠르게 도출할 수 있다.
 - 본 연구에서 최종적으로 학습한 모델은 약 2ms (0.002 초)만에 형상을 예측하였다.
 - NACA 4-digit 에어포일이 낼 수 있는 성능은 특정 영역에 제한되어 있다. 그러므로 사용자가 성능을 입력하여 형상을 얻어낼 때, 특정 영역을 넘어서는 조건을 부여하면 NACA 4-digit으로는 사용자의 요구를 들어줄 수 없다.

본 연구는 NACA 4-digit 형상을 이용하여 인공신경망을 통해 에어포일 설계가 가능함을 보였다. 그러나 더 범용적인 에어포일 설계를 하기 위해서는 형상 생성이 더 자유로운 에어포일 생성 기법을 사용하여야 한다.

현재 후속 연구로 CST Curve 에어포일에 대한 연구를 진행중이며, 초기 학습 데이터 획득 및 학습을 실시하였다. CST Curve는 윗면의 파라미터가 아랫면 파라미터보다 커야 하지만, NACA 4-digit의 대칭형 에어포일의 최대 캠버와 최대 캠버 위치와 같이 동떨어진 특성을 보이지 않는다. 이로 인해 초기 데이터만으로도 인공신경망을 학습하여도 테스트 결과가 매우 양호한 결과를 보였다. 하지만 데이터베이스 강화 루프는 일반적이지 않은 영역의 NACA 4-digit 에어포일 학습 개선 효과도 존재하며, CST Curve에서도 데이터의 강화로 테스트 오차가 줄어드는 것을 확인할 수 있었다.

CST Curve에 관한 연구는 본 논문과 같이 체계적인 과정으로 이루어지진 않아 단기적인 계획으로 본 논문에서 정립한 프로세스에 기반하여 CST Curve 에어포일을 설계할 수 있는 딥러닝 기반 인공지능 알고리즘을 구현하고자 한다. 더 나아가 중장기적으로는 FEM 해석 프로그램을 이용하여 날개 구조물 설계 인공지능 등 더 다양한 설계 문제에 인공신경망과 딥러닝 기법을 도입하고자 한다.

참 고 문 헌

- [1] Airfoil Tools, "Airfoil Tools," [Online]. Available: <http://airfoiltools.com>. [Accessed 22 4 2019].
- [2] UIUC Applied Aerodynamics Group, "UIUC Airfoil Data Site," UIUC, 2019. [Online]. Available: <https://m-selig.ae.illinois.edu/ads.html>. [Accessed 22 4 2019].
- [3] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, no. 131, pp. 504-507, 2006.
- [4] E. N. Jacobs, K. E. Ward and R. M. Pinerton, "The Characteristics of 78 Related Airfoil Sections from Tests in the Variable-Density Wind Tunnel," National Advisory Committee for Aeronautics, 1935.
- [5] M. Drela, "Xfoil - Subsonic Airfoil Development System," MIT, 23 12 2013. [Online]. Available: <http://web.mit.edu/drela/Public/web/xfoil/>. [Accessed 14 12 2017].
- [6] C. Ortega, "AI & Machine Learning Impact on FP&A," 23 8 2017. [Online]. Available: <https://www.linkedin.com/pulse/ai-machine-learning-impact-fpa-chris-ortega-mba>.
- [7] A. Geron, Hands-On Machine Learning with Scikit-Learn & Tensorflow, Sebastopol, CA: O'Reilly Media, 2019.
- [8] P. Werbos, Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, PhD. thesis, Harvard University, 1974.
- [9] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533-536, 1986.

- [10] Python Software Foundation, "Python," Python Software Foundation, 25 3 2019. [Online]. Available: <https://www.python.org/downloads/>. [Accessed 24 4 2019].
- [11] J. M. Hong, "Deep Learning 3 Programming One Neuron C++," Youtube, 2016.
- [12] Wikipedia, "Perceptron," 10 5 2019. [Online]. Available: <https://en.wikipedia.org/wiki/Perceptron>. [Accessed 1 6 2019].
- [13] G. E. Hinton and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 7, no. 18, pp. 1527-1554, 2006.
- [14] G. E. Hinton and T. J. Sejnowski, "Chapter 7. Learning and Relearning in Boltzmann Machines," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, PDP Research Group, 1986, pp. 283-317.
- [15] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *the 13th International Conference on Artificial Intelligence and Statistics*, Sardinia, Italy, 2010.
- [16] K. He, X. Zhang, S. Ren and J. Sun, "Delving Deep int Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," Microsoft, 6 Feb 2015.
- [17] Google, "tf.contrib.layer.xavier_initializer," [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/contrib/layers/xavier_initializer. [Accessed 9 4 2019].
- [18] Wikipedia, "Logistic function," 8 4 2019. [Online]. Available: https://en.wikipedia.org/wiki/Logistic_function. [Accessed 12 4 2019].
- [19] Wikipedia, "Sigmoid function," 22 2 2019. [Online]. Available: https://en.wikipedia.org/wiki/Sigmoid_function. [Accessed 12 4

- 2019].
- [20] X. Glorot, A. Bordes and Y. Bengio, "Deep Sparse Rectifier Neural Networks," in *14th International Conference on Artificial Intelligence and Statistics*, Fort Lauderdale, FL, USA, 2011.
 - [21] Wikipedia, "Rectifier (neural networks)," 2 4 2019. [Online]. Available: [https://en.wikipedia.org/wiki/Rectifier_\(neural_networks\)](https://en.wikipedia.org/wiki/Rectifier_(neural_networks)). [Accessed 18 4 2019].
 - [22] Wikipedia, "Mean squared error," 30 3 2019. [Online]. Available: https://en.wikipedia.org/wiki/Mean_squared_error. [Accessed 18 4 2019].
 - [23] Wikipedia, "Root-mean-square-deviation," 6 4 2019. [Online]. Available: https://en.wikipedia.org/wiki/Root-mean-square_deviation. [Accessed 18 4 2019].
 - [24] S. Ruder, "An overview of gradient descent optimization algorithms," 9 2 2018. [Online]. Available: <http://ruder.io/optimizing-gradient-descent/index.html#momentum>. [Accessed 18 4 2019].
 - [25] Stanford University, "CS231n, neural networks 3," [Online]. Available: <http://cs231n.github.io/neural-networks-3/>. [Accessed 17 4 2019].
 - [26] J. Duchi, E. Hazan and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, no. 12, pp. 2121-2159, 11.7.2011.
 - [27] G. E. Hinton, N. Srivastava and K. Swersky, "Lecture 6a - Overview of mini-batch gradient descent," in *Neural Networks for Machine Learning*, 2012, p. 29.
 - [28] D. P. Kingma and J. L. Ba, "ADAM: A Method for Stochastic Optimization," in *ICLR*, 2015.
 - [29] Wikipedia, "Airfoil," 7 4 2019. [Online]. Available:

- <https://en.wikipedia.org/wiki/Airfoil>. [Accessed 19 4 2019].
- [30] J. D. Anderson, "Incompressible Flow over Airfoils," in *Fundamentals of Aerodynamics*, NY, USA, McGraw-Hill Higher Education, 2001, pp. 279-281.
 - [31] R. W. Derksen and T. Rogalsky, "Bezier-PARSEC: An optimized aerofoil parameterization for design," *Advances in Engineering Software*, no. 41, pp. 923-930, 2010.
 - [32] B. M. Kulfan, "A Universal Parametric Geometry Representation Method - "CST"," in *45th AIAA*, Reno, Nevada, 2007.
 - [33] A. B. Dinh, K. H. NGO and N. V. Nguyen, "An efficient low-speed airfoil design optimization process using multi-fidelity analysis for UAV flying wing," *Science & Technology Development*, vol. 19, pp. 43-52, 2016.
 - [34] S. L. Lee and M. Damodaran, "Airfoil Design Optimization Using Navier-Stokes Equations and Simulated Annealing," *ICAS*, pp. 2112.1-2112.9, 2000.
 - [35] Wikipedia, "NACA airfoil," 13 1 2019. [Online]. Available: https://en.wikipedia.org/wiki/NACA_airfoil. [Accessed 19 4 2019].
 - [36] ImageNet, "ImageNet," Stanford University, Princeton University, [Online]. Available: <http://image-net.org/index>. [Accessed 22 4 2019].
 - [37] V. Sekar, M. Zhang, C. Shu and B. C. Khoo, "Inverse Design of Airfoil Using a Deep Convolutional Neural Network," *AIAA*, vol. 57, no. 3, pp. 993-1003, 2019.
 - [38] A. Kharal and A. Saleem, "Neural networks based airfoil generation for a given Cp using Besier-PARASEC parameterization," *Aerospace Science and Technology*, no. 23, pp. 330-344, 2012.
 - [39] K. Thinakaran and R. Rajasekar, "Different Hybrid Neural Network in Inverse Design," *International Journal of Engineering and*

Applied Sciences, vol. 2, no. 4, pp. 65-68, 2015.

- [40] R. Santhanakrishnan, J. Upendira and R. Manikandan, "Inverse Design Methodology of Transonic Airfoil by Artificial Neural Network Technoqe," *International Journal of Mechanical and Production*, vol. 3, no. 1, pp. 95-100, 2013.
- [41] J. Morgado, R. Vizinho, M. Silvestre and J. Pascoa, "XFOIL vs CFD performance predictions for high lift low Reynolds number airfoil," *Aerospace Science and Technology*, no. 52, pp. 207-214, 2016.
- [42] J. D. Anderson, Jr, "Incompressible Flow over Finite Wings," in *Fundamentals of Aerodynamics*, McGraw-Hill, 2001, pp. 351-356.
- [43] I. H. Abbott and A. E. von Doenhoff, Theory of Wing Section, New York: Dover Publications and McGraw-Hill Company, 2959.
- [44] J. L. Ba, J. R. Kiros and G. E. Hinton, "Layer Normalization," Toronto University, Toronto, Canada, 21 Jul. 2016.
- [45] Google, "Tensorflow," Google, [Online]. Available: <https://www.tensorflow.org/>. [Accessed 24 4 2019].
- [46] Anaconda, "Anaconda," Anaconda, 2019. [Online]. Available: <https://www.anaconda.com/>. [Accessed 24 4 2019].
- [47] scikit-learn, "scikit-learn: Machine Learning in Python," 3 2019. [Online]. Available: <https://scikit-learn.org/stable/>. [Accessed 24 4 2019].
- [48] NVIDIA, "CUDA Toolkit 10.1 Download," NVIDIA, 2019. [Online]. Available: <https://developer.nvidia.com/cuda-downloads>. [Accessed 24 4 2019].
- [49] NVIDIA, "About CUDA," [Online]. Available: <https://developer.nvidia.com/about-cuda>. [Accessed 24 4 2019].
- [50] NVIDIA, "CUDA Toolkit 10.0 Archive," NVIDIA, [Online]. Available: <https://developer.nvidia.com/cuda-10.0-download-archive>.

- [Accessed 24 4 2019].
- [51] NVIDIA, "NVIDIA cuDNN," NVIDIA, [Online]. Available: <https://developer.nvidia.com/cudnn>. [Accessed 24 4 2019].
 - [52] C.-K. Choi, A. A. Maw, M.-K. Kim, M. Tyan, K.-i. Kim and J.-W. Lee, "Training Artificial Neural Network for Low Speed Airfoil Design," in *Korea Society for Aviation and Aeronautics*, Korea, Nov. 2017.
 - [53] C.-K. Choi, A. A. Maw, M. Tyan and J.-W. Lee, "NACA 4 digit Airfoil Inverse Design with Deep Learning," in *GPU Technology Conference*, San Jose, CA, USA, Mar. 2019.
 - [54] Wikipedia, "Normal distribution," 3 4 2019. [Online]. Available: https://en.wikipedia.org/wiki/Normal_distribution. [Accessed 29 4 2019].
 - [55] Microsoft, "NORM.DIST 함수," 2019. [Online]. Available: <https://support.office.com/ko-kr/article/norm-dist-%ED%95%A8%EC%88%98-edb1cc14-a21c-4e53-839d-8082074c9f8d>. [Accessed 29 4 2019].
 - [56] Wikipedia, "Standard deviation," 23 4 2019. [Online]. Available: https://en.wikipedia.org/wiki/Standard_deviation. [Accessed 29 4 2019].
 - [57] Microsoft, "STDEVS 함수," 2019. [Online]. Available: <https://support.office.com/ko-kr/article/stdevs-%ED%95%A8%EC%88%98-7d69cf97-0c1f-4acf-be27-f3e83904cc23>. [Accessed 29 4 2019].
 - [58] Wikipedia, "Six Sigma," 9 5 2019. [Online]. Available: https://en.wikipedia.org/wiki/Six_Sigma. [Accessed 10 5 2019].
 - [59] Wikipedia, "68-95-99.7 rule," 7 4 2109. [Online]. Available: https://en.wikipedia.org/wiki/68%E2%80%9395%E2%80%9399.7_rule. [Accessed 10 5 2019].
 - [60] Wikipedia, "User interface," 6 5 2019. [Online]. Available:

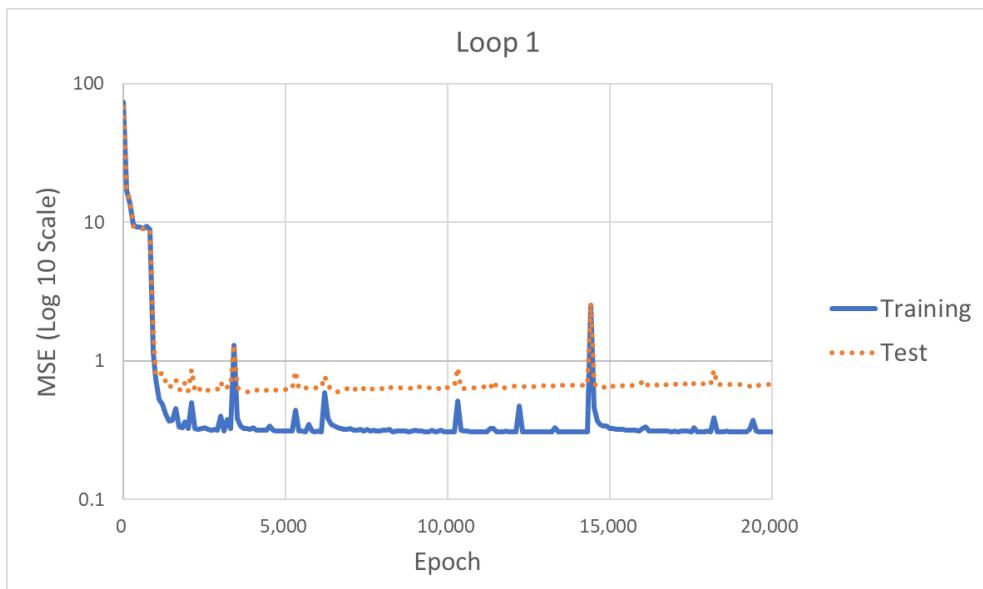
- https://en.wikipedia.org/wiki/User_interface. [Accessed 13 5 2019].
- [61] Wikipedia, "Text-based user interface," 26 4 2019. [Online]. Available: https://en.wikipedia.org/wiki/Text-based_user_interface. [Accessed 13 5 2019].
- [62] Python Software Foundation, "PyQT5," Python Software Foundation, 2019. [Online]. Available: <https://pypi.org/project/PyQt5/>. [Accessed 13 5 2019].
- [63] Pandas, "pandas.DataFrame.boxplot," Pandas, [Online]. Available: <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.boxplot.html>. [Accessed 23 5 2019].
- [64] Pyinstaller Developer Team, "Pyinstaller Quickstart," Pyinstaller Developer Team, 21 2 2019. [Online]. Available: <https://www.pyinstaller.org/>. [Accessed 24 5 2019].
- [65] C.-K. Choi, A. A. Maw, M.-K. Kim, M. Tyan, J. Chung and J.-W. Lee, "A Deep Learning Based Inverse Airfoil Design Algorithm with Database Update," in *the Asian Congress of Structural and Multidisciplinary Optimization*, Dalian, China, 2018.
- [66] C.-K. Choi, A. A. Maw, M. Tyan and J.-W. Lee, "Deep Learning based NACA 4-digit Airfoil Design Algorithm Hyperparameter Optimization," in *Korea Society for Aviation and Aeronautics*, Dec. 2018.
- [67] C.-K. Choi and J.-W. Lee, "Performance Comparison of the Deep Learning based Airfoil Inverse Design Algorithm with the Training Epoch," in *KAI Paper Competition*, 2018.
- [68] S. Gudmundsson, "Airfoil Selection How-to," in *General Aviation Aircraft Design: Applied Methods and Procedures*, Walthan, MA, Elsevier, 2014, pp. 289-294.
- [69] A. Muller, Introduction to Machine Learning with Python, 한빛미디

어, 2017.

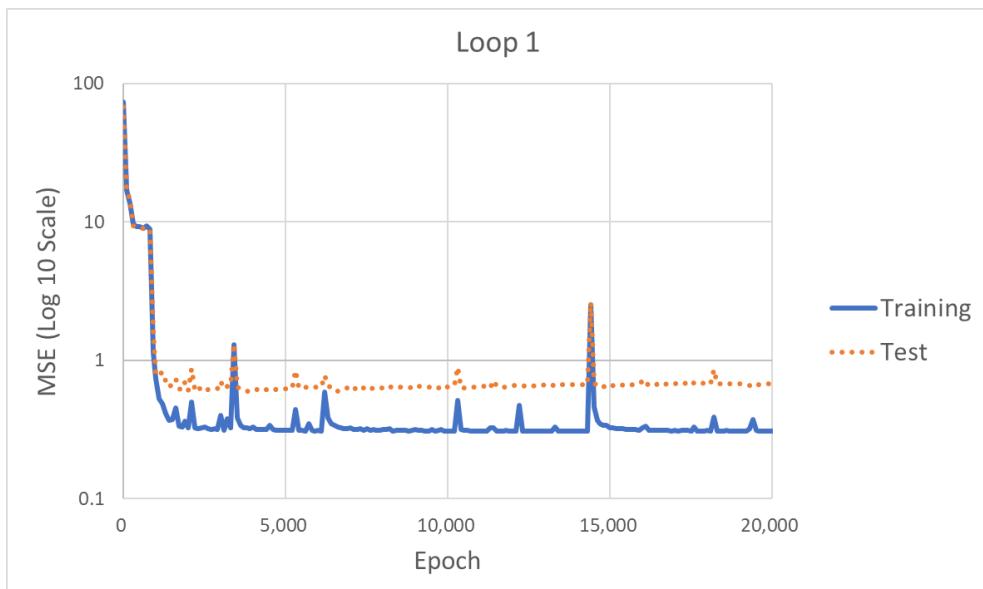
- [70] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *27th International Conference on Machine Learning*, Haifa, Israel, 2010.

Appendix A. Database Enhancement Loop(DEL)

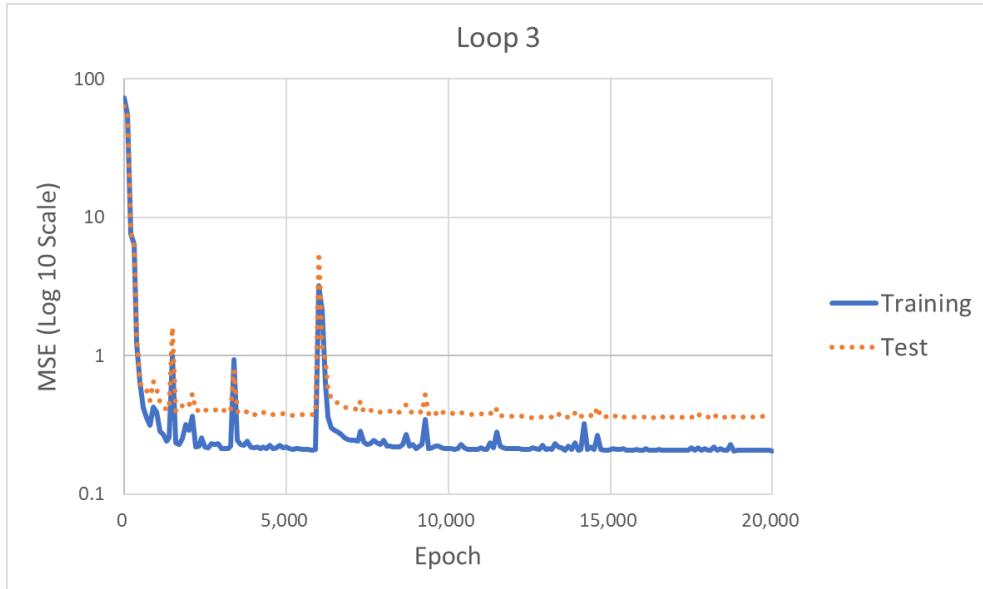
Training Result: NACA 4-digit Airfoil Case



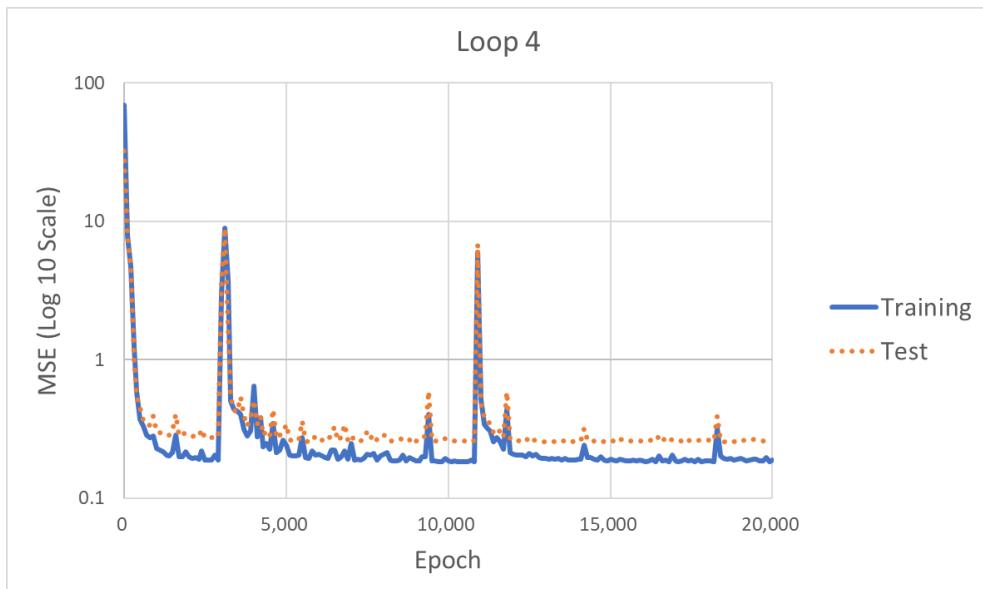
<Appendix A. 그림- 1> DEL: NACA 4-digit Airfoil – Loop 1



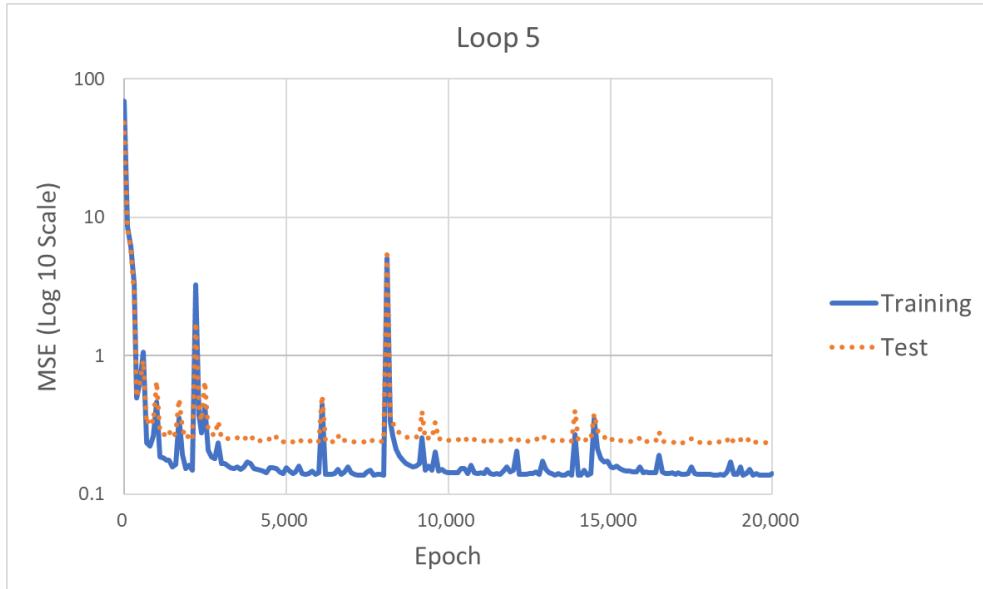
<Appendix A. 그림- 2> DEL: NACA 4-digit Airfoil – Loop 2



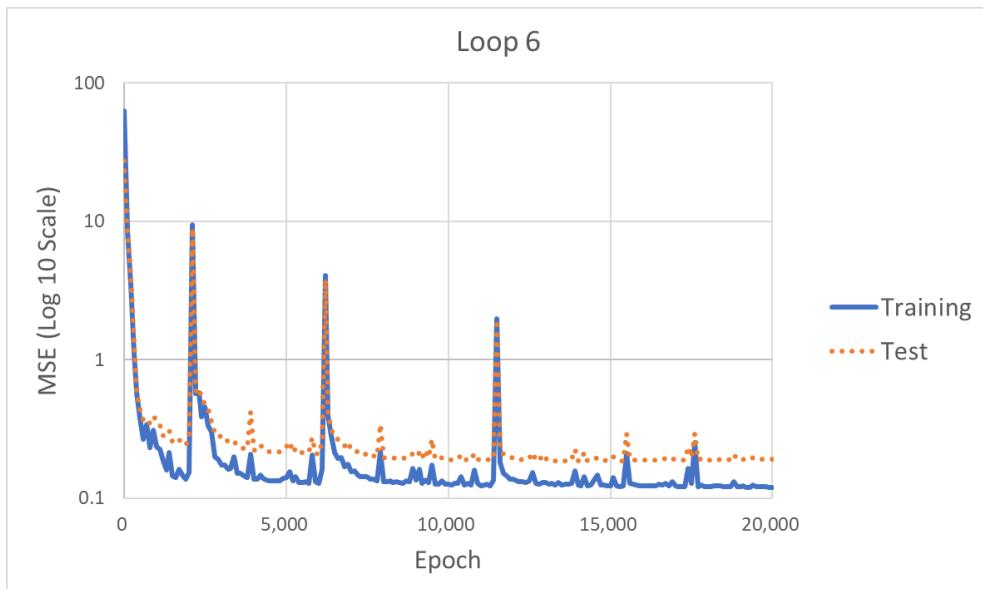
<Appendix A. 그림- 3> DEL: NACA 4-digit Airfoil – Loop 3



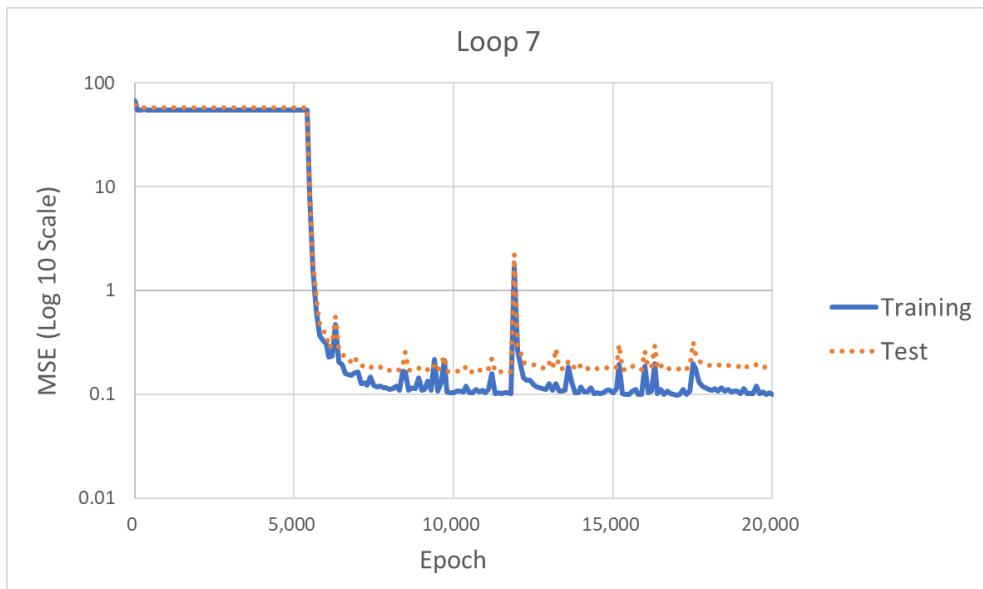
<Appendix A. 그림- 4> DEL: NACA 4-digit Airfoil – Loop 4



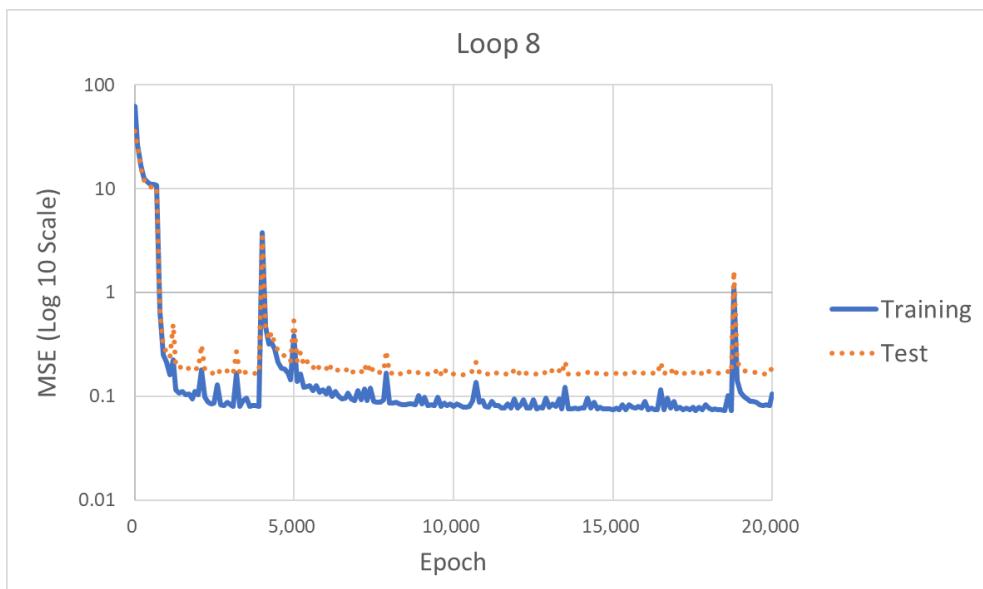
<Appendix A. 그림- 5> DEL: NACA 4-digit Airfoil – Loop 5



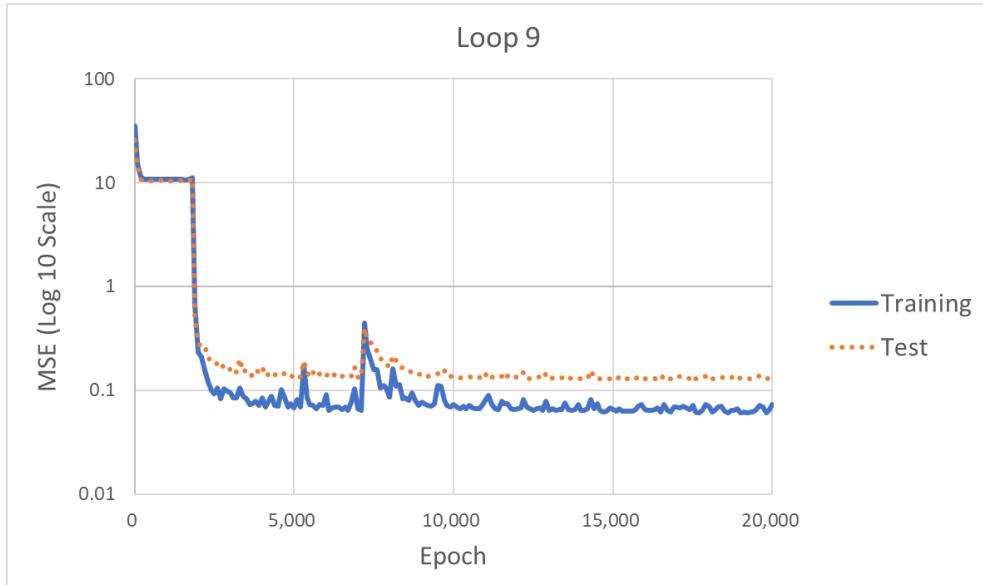
<Appendix A. 그림- 6> DEL: NACA 4-digit Airfoil – Loop 6



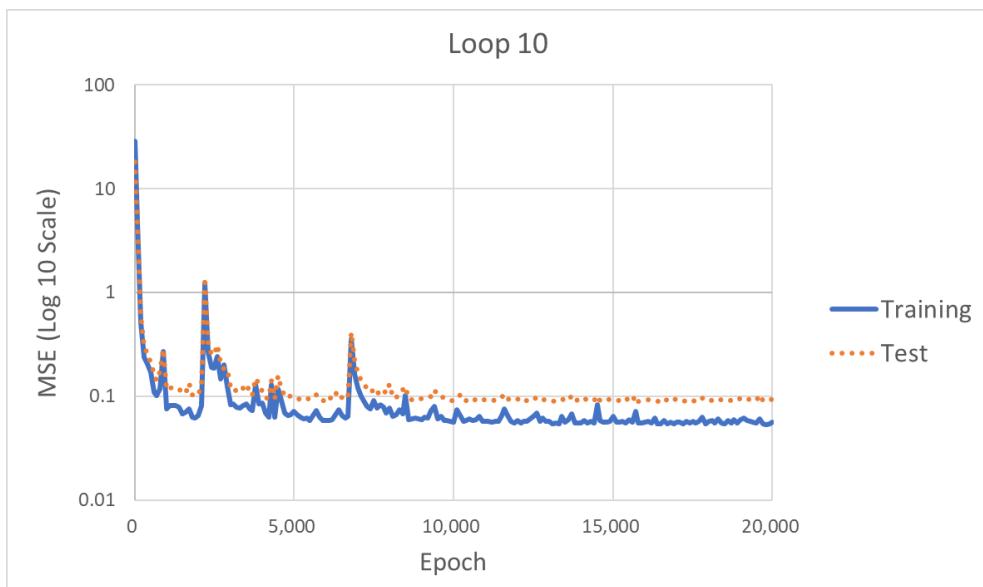
<Appendix A. 그림- 7> DEL: NACA 4-digit Airfoil – Loop 7



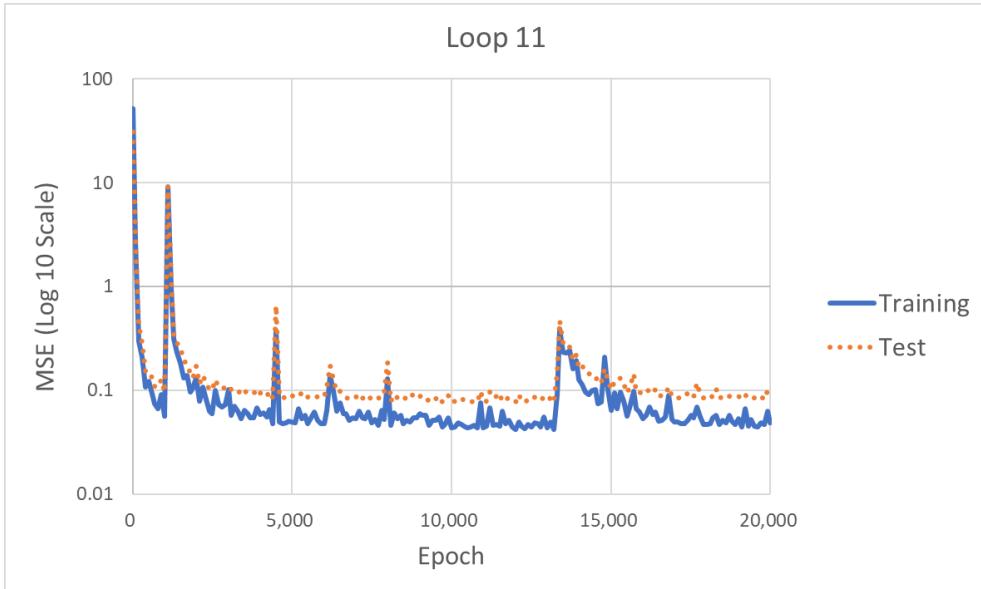
<Appendix A. 그림- 8> DEL: NACA 4-digit Airfoil – Loop 8



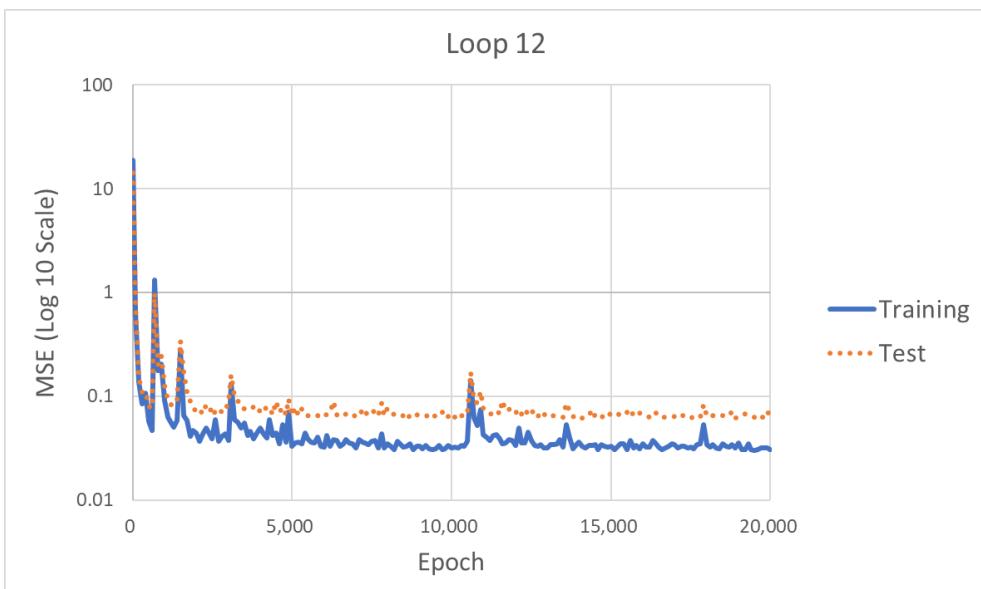
<Appendix A. 그림- 9> DEL: NACA 4-digit Airfoil – Loop 9



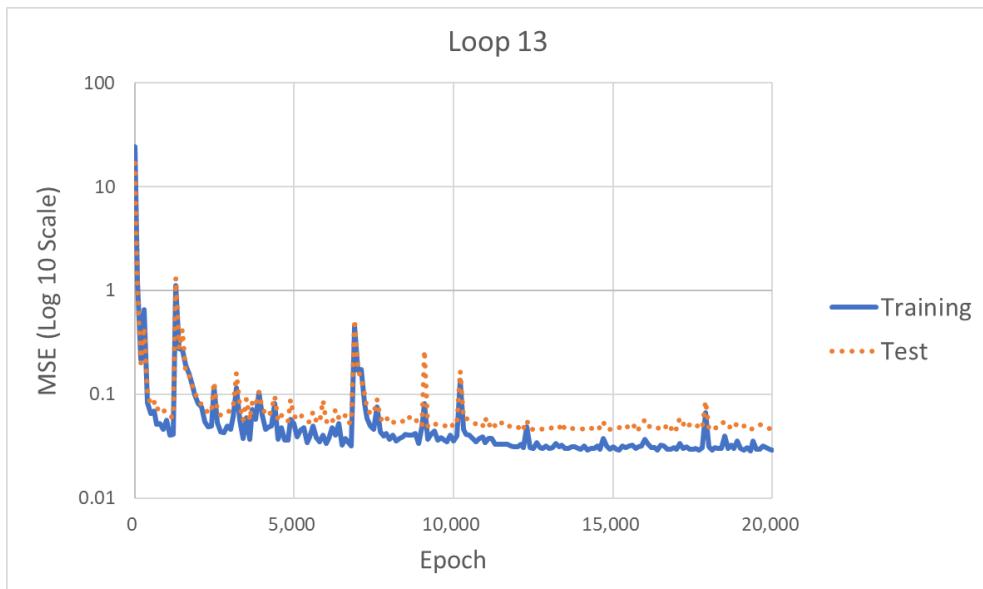
<Appendix A. 그림- 10> DEL: NACA 4-digit Airfoil – Loop 10



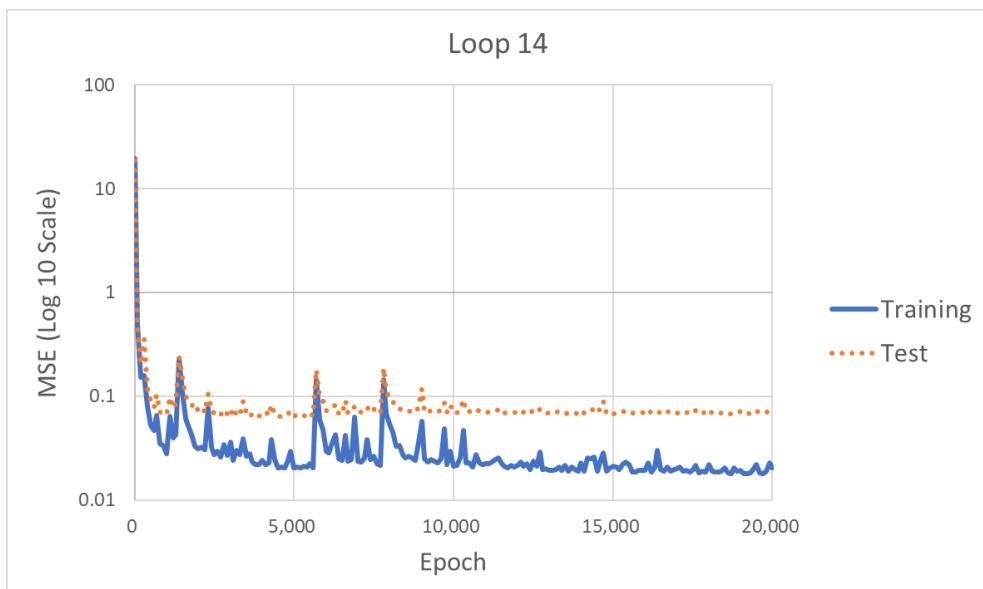
<Appendix A. 그림- 11> DEL: NACA 4-digit Airfoil – Loop 11



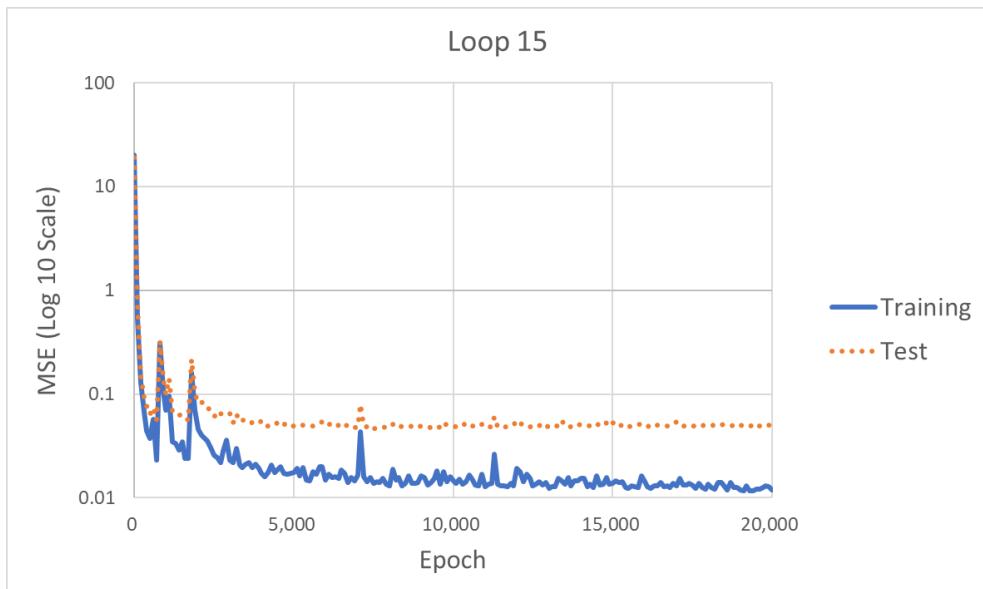
<Appendix A. 그림- 12> DEL: NACA 4-digit Airfoil – Loop 12



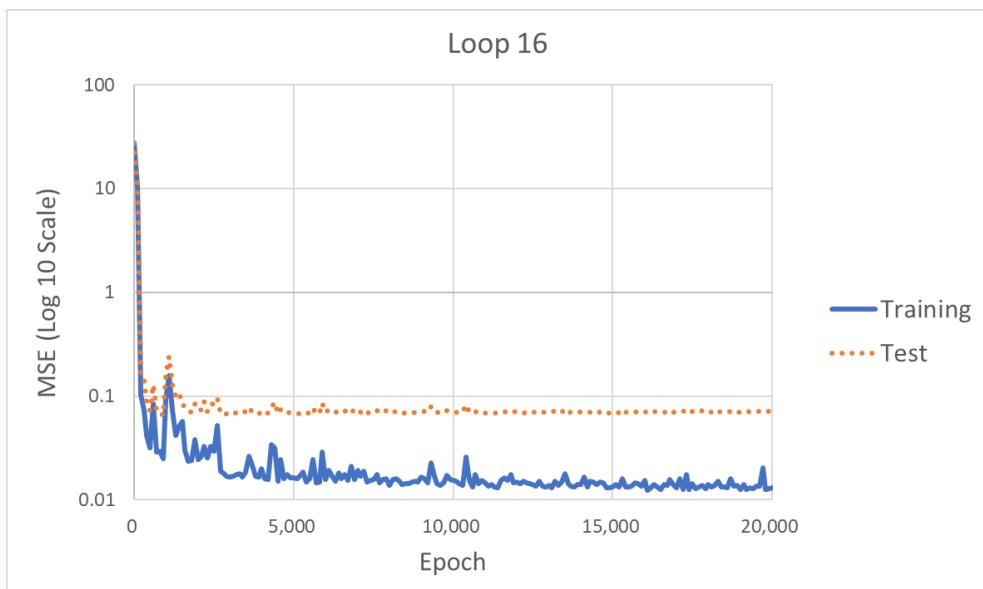
<Appendix A. 그림- 13> DEL: NACA 4-digit Airfoil – Loop 13



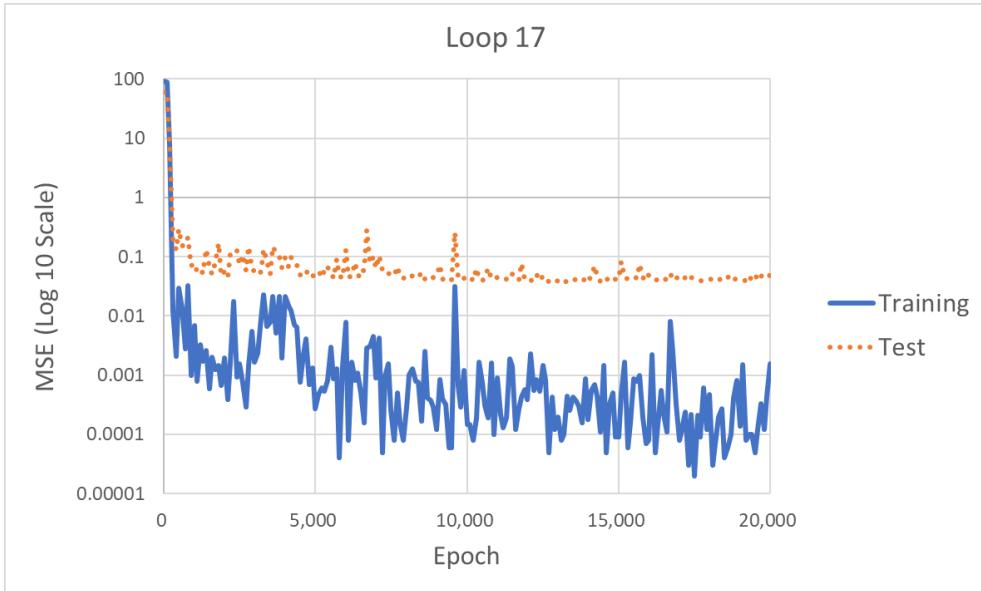
<Appendix A. 그림- 14> DEL: NACA 4-digit Airfoil – Loop 14



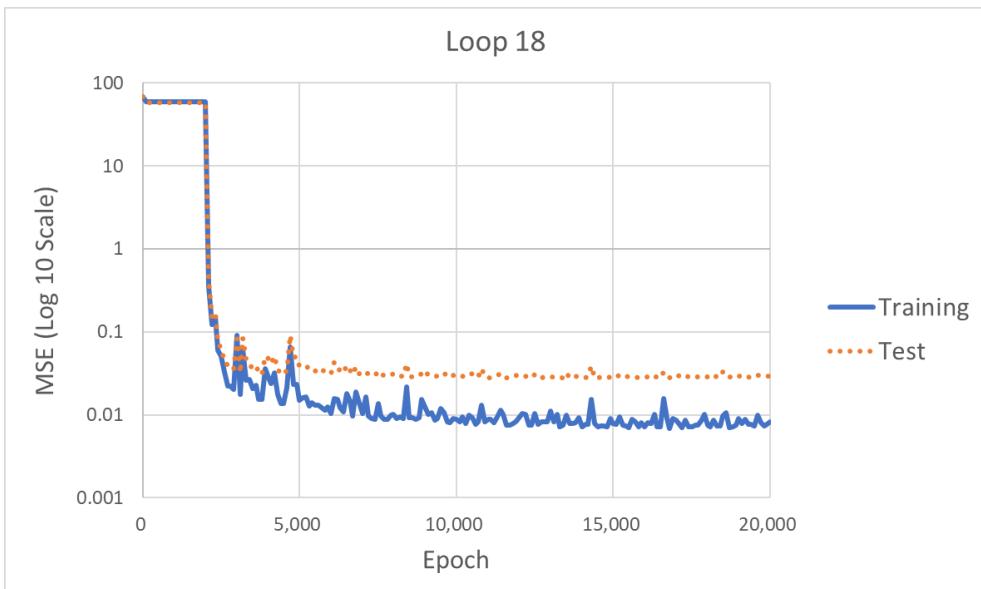
<Appendix A. 그림- 15> DEL: NACA 4-digit Airfoil – Loop 15



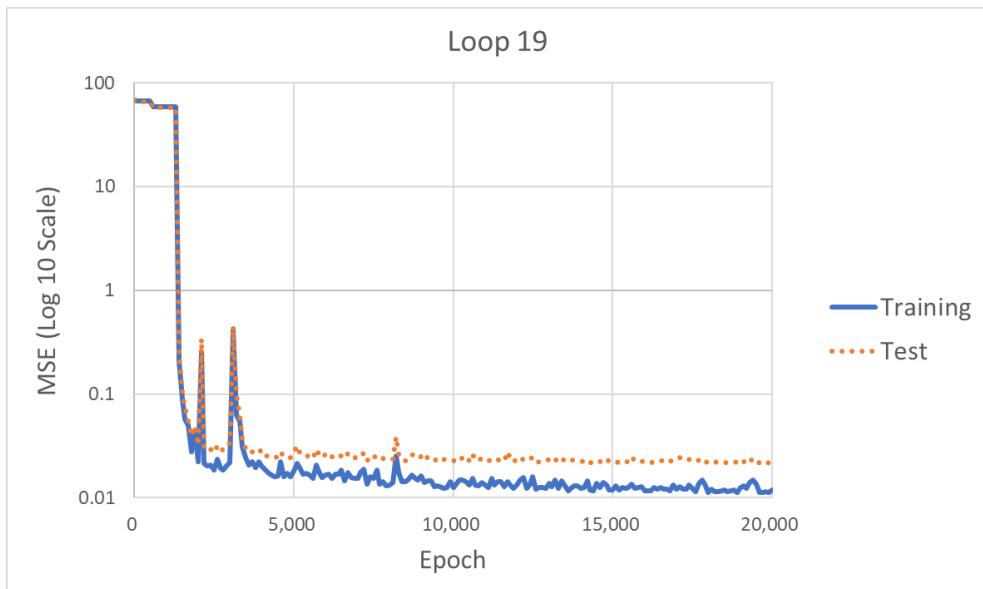
<Appendix A. 그림- 16> DEL: NACA 4-digit Airfoil – Loop 16



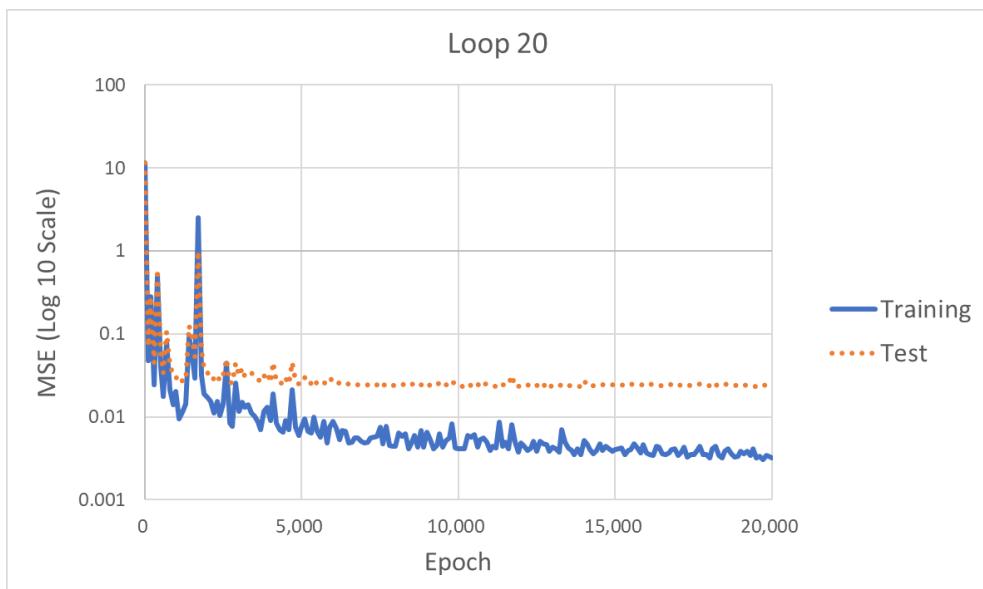
<Appendix A. 그림- 17> DEL: NACA 4-digit Airfoil – Loop 17



<Appendix A. 그림- 18> DEL: NACA 4-digit Airfoil – Loop 18



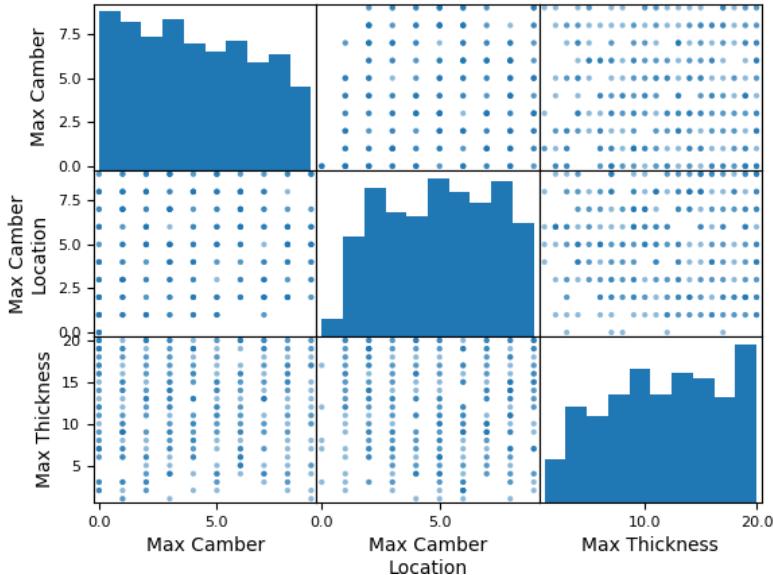
<Appendix A. 그림- 19> DEL: NACA 4-digit Airfoil – Loop 19



<Appendix A. 그림- 20> DEL: NACA 4-digit Airfoil – Loop 20

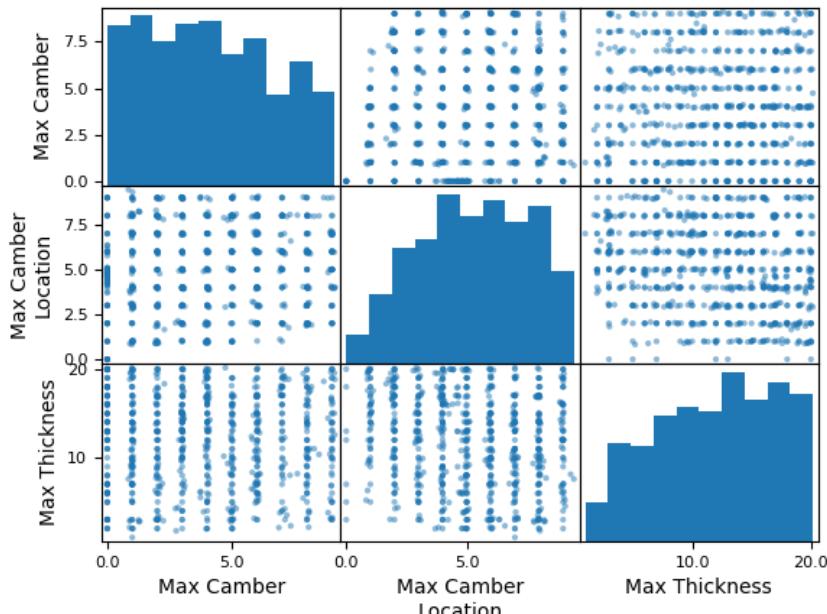
Appendix B. DEL Loop 별 정답 데이터의 Scatter Matrix

Scatter Matrix_Loop1 Answer



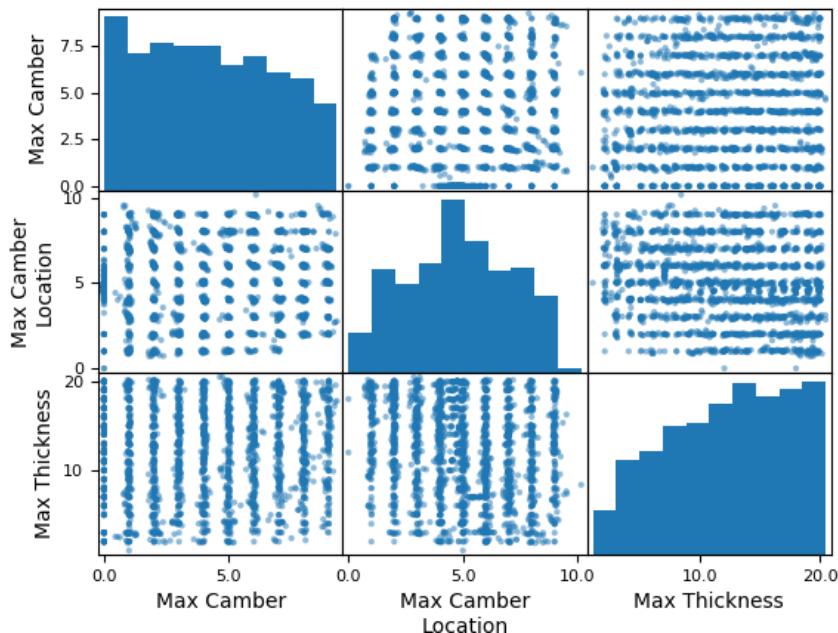
<Appendix B. 그림- 1> 형상 파라미터 Scatter Matrix: Loop 1 Answer DB

Scatter Matrix_Loop5 Answer



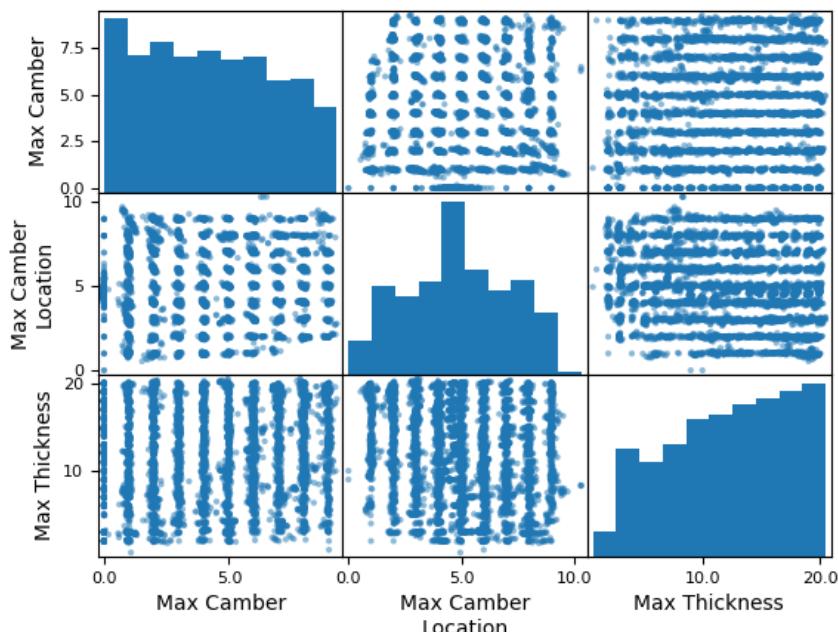
<Appendix B. 그림- 2> 형상 파라미터 Scatter Matrix: Loop 5 Answer DB

Scatter Matrix_Loop10 Answer



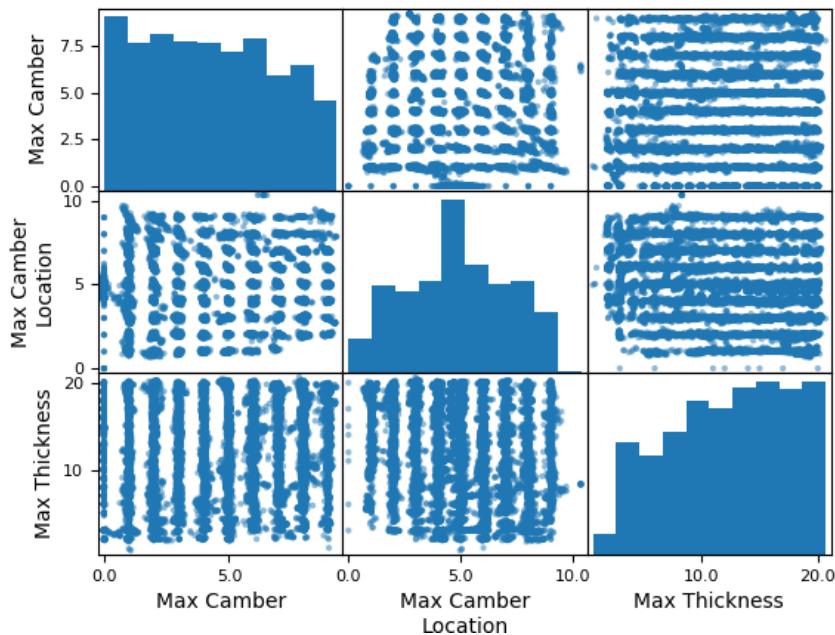
<Appendix B. 그림- 3> 형상 파라미터 Scatter Matrix: Loop 10 Answer DB

Scatter Matrix_Loop15 Answer



<Appendix B. 그림- 4> 형상 파라미터 Scatter Matrix: Loop 15 Answer DB

Scatter Matrix_Loop20 Answer



<Appendix B. 그림- 5> 형상 파라미터 Scatter Matrix: Loop 20 Answer DB

Appendix C. DEL Loop 별 파라미터 오차의 정규분포와 표준정규분포

1. 평균과 표준편차

<Appendix C. 표- 1> DEL - Loop에 따른 테스트 오차 평균: 형상 파라미터

DEL Loop	Max Camber	Max Camber Location	Max Thickness
Loop 1	-0.0034768	0.116517	0.0167733
Loop 2	-0.0114626	0.120003	0.019963
Loop 3	-0.0056763	0.0663081	-0.00221957
Loop 4	0.015585	-0.0454593	-0.0535585
Loop 5	0.00193606	0.0222548	-0.0266243
Loop 6	-0.00538808	0.0426059	-0.00297473
Loop 7	-0.000389199	0.0147416	-0.01837
Loop 8	-0.0421384	0.12633	0.206068
Loop 9	0.0215172	-0.0504511	-0.0660302
Loop 10	-0.00117549	-0.0440781	-0.0385653
Loop 11	0.00463098	-0.0529865	-0.0432381
Loop 12	-0.019303	0.0335363	-0.0142332
Loop 13	-0.0169291	0.00895104	-0.0452355
Loop 14	-0.00017864	0.0425743	0.0942836
Loop 15	0.00842811	-0.0185793	-0.0268796
Loop 16	-0.00612721	-0.00921264	-0.0243764
Loop 17	-0.00672442	-0.0245228	0.0906987
Loop 18	0.00336166	-0.0102514	0.0179624
Loop 19	0.0188608	0.0279578	-0.0031793
Loop 20	0.00923081	-0.00257035	-0.0229289

<Appendix C. 표- 2> DEL - Loop에 따른 테스트 오차 평균: 성능 파라미터

DEL Loop	C_{I0}	C_{la0}	C_{d0}	C_{m0}	C_{ma0}
Loop 1	0.00011	-0.00043	-5.09E-05	-2.27E-05	0.00013
Loop 2	-0.00017	-0.00034	-2.24E-06	-2.47E-05	7.64E-05
Loop 3	0.000793	-0.00016	-7.23E-05	-0.00013	5.08E-05
Loop 4	0.001917	-0.00062	-1.87E-05	-0.00016	0.000145
Loop 5	0.00031	0.00054	3.48E-06	-1.91E-05	-0.00012
Loop 6	5.36E-05	0.00032	-1.25E-05	-0.00011	-7.20E-05
Loop 7	0.00028	6.74E-05	-2.52E-06	-0.00017	-1.50E-05
Loop 8	-0.00346	-0.00034	-6.24E-05	0.000208	9.42E-05
Loop 9	0.00119	-0.0002	-1.38E-06	1.95E-05	5.18E-05
Loop 10	-0.00069	0.000349	-2.11E-05	0.000335	-7.86E-05
Loop 11	7.70E-05	-3.30E-05	-2.63E-05	0.000238	1.57E-05
Loop 12	-0.00088	-0.0002	-2.26E-05	-3.78E-05	5.59E-05
Loop 13	-0.00178	-0.00018	-7.80E-06	0.000312	3.14E-05
Loop 14	0.001462	-0.00015	2.06E-05	-0.00059	4.23E-05
Loop 15	0.000193	-5.80E-05	-7.63E-06	2.52E-05	3.83E-06
Loop 16	-0.00056	-0.00026	-1.73E-05	0.000159	5.73E-05
Loop 17	-0.00222	0.000136	-4.62E-06	0.000713	-3.03E-05
Loop 18	0.000233	-6.25E-05	2.23E-06	8.92E-05	1.59E-05
Loop 19	0.003099	-7.74E-05	-1.07E-05	-0.00089	1.84E-05
Loop 20	0.001237	-8.62E-06	5.72E-06	-0.00024	2.09E-06

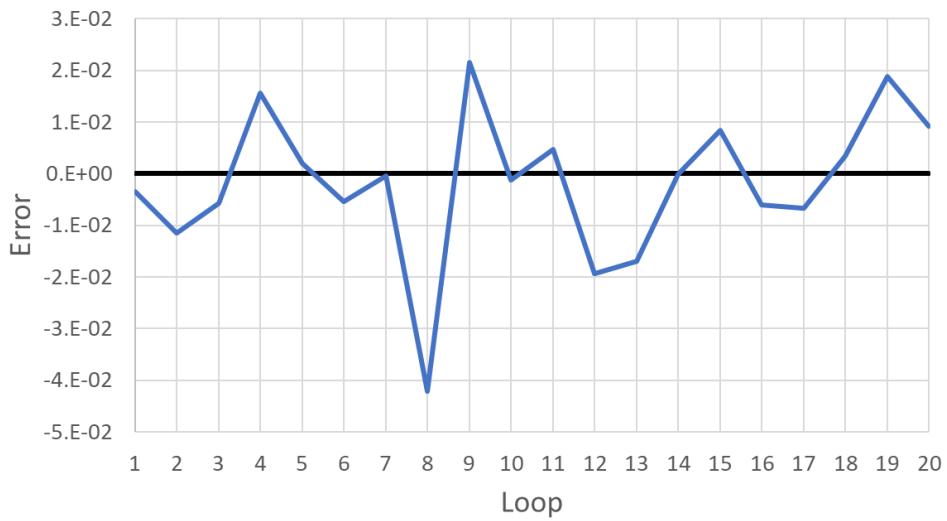
<Appendix C. 표- 3> DEL - Loop에 따른 테스트 오차 표준편차: 형상 파라미터

DEL Loop	Max Camber	Max Camber Location	Max Thickness
Loop 1	0.132697	1.20072	0.733579
Loop 2	0.0729325	1.18835	0.425252
Loop 3	0.0873533	0.955018	0.418145
Loop 4	0.0611733	0.794814	0.382474
Loop 5	0.0680277	0.761492	0.32719
Loop 6	0.0865089	0.616256	0.405989
Loop 7	0.0557969	0.593101	0.449107
Loop 8	0.0653692	0.570384	0.413501
Loop 9	0.0690032	0.530336	0.29164
Loop 10	0.0478176	0.444506	0.27717
Loop 11	0.0495256	0.317452	0.383516
Loop 12	0.0639664	0.335768	0.232656
Loop 13	0.0632093	0.264252	0.243576
Loop 14	0.0505073	0.25016	0.343302
Loop 15	0.0393486	0.218526	0.288431
Loop 16	0.0414379	0.216008	0.387831
Loop 17	0.0416344	0.163264	0.330323
Loop 18	0.0314269	0.166017	0.237918
Loop 19	0.0272384	0.142115	0.20962
Loop 20	0.0240654	0.141308	0.189559

<Appendix C. 표- 4> DEL - Loop에 따른 테스트 오차 표준편차: 성능 파라미터

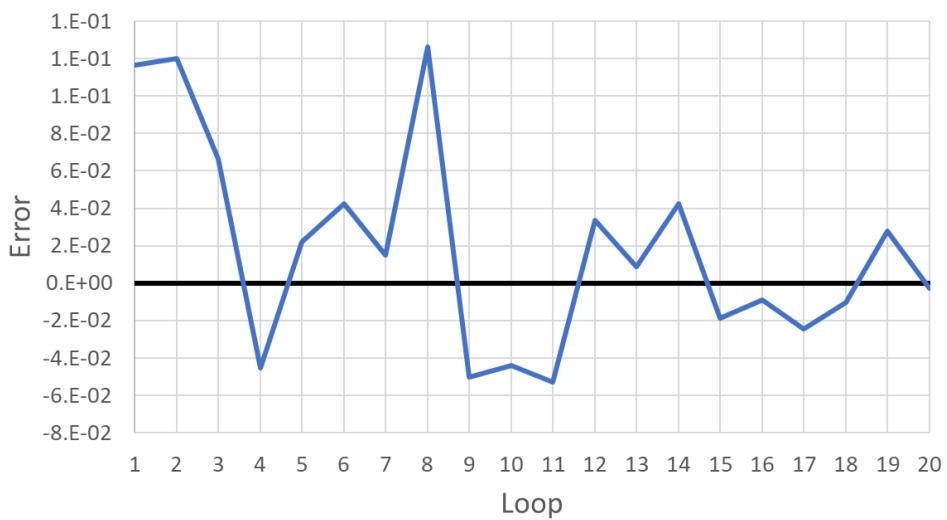
DEL Loop	C_{I0}	C_{la0}	C_{d0}	C_{m0}	C_{ma0}
Loop 1	0.01914	0.011536	0.000986	0.0045	0.002754
Loop 2	0.015905	0.014718	0.000891	0.00326	0.003214
Loop 3	0.017346	0.010356	0.000781	0.004548	0.002294
Loop 4	0.011691	0.008112	0.000435	0.002917	0.001863
Loop 5	0.00972	0.01286	0.000382	0.002264	0.002915
Loop 6	0.008022	0.012288	0.000561	0.002161	0.0028
Loop 7	0.009881	0.01192	0.000691	0.002328	0.002702
Loop 8	0.041235	0.009898	0.000998	0.008802	0.002213
Loop 9	0.00735	0.010928	0.000478	0.001807	0.00242
Loop 10	0.006877	0.012119	0.000445	0.001629	0.002769
Loop 11	0.008223	0.012807	0.000493	0.002094	0.0029
Loop 12	0.011043	0.010967	0.000592	0.002683	0.00246
Loop 13	0.009853	0.010383	0.000413	0.002402	0.00234
Loop 14	0.007044	0.01084	0.000411	0.001585	0.002447
Loop 15	0.010574	0.009807	0.000383	0.002431	0.002158
Loop 16	0.007961	0.011086	0.000407	0.001825	0.002447
Loop 17	0.006172	0.011224	0.00039	0.001695	0.002472
Loop 18	0.00457	0.009276	0.000377	0.001215	0.002073
Loop 19	0.004514	0.009205	0.000444	0.001112	0.002051
Loop 20	0.004137	0.010011	0.00037	0.001001	0.002239

DEL: Mean of Errors by Loop - Max Camber



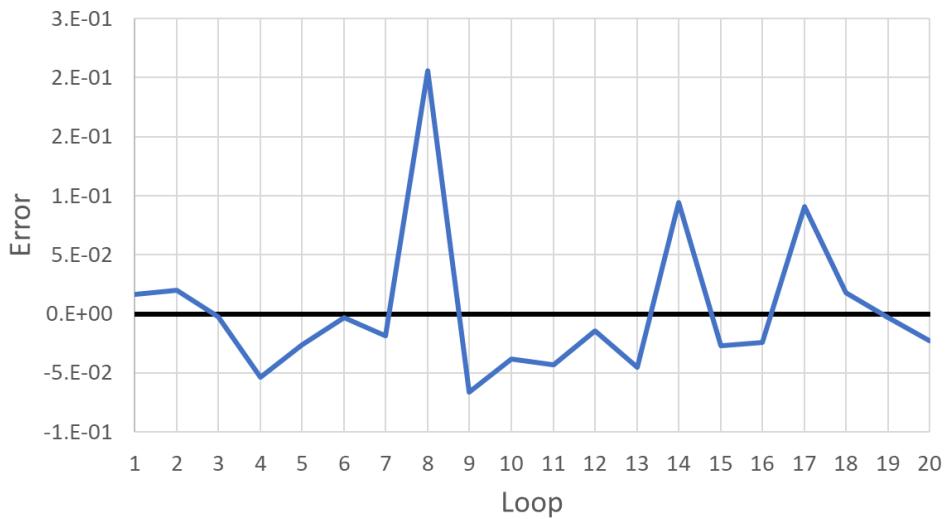
<Appendix C. 그림- 1> Loop에 따른 오차의 평균: 최대 캠버

DEL: Mean of Errors by Loop - Max Camber Location



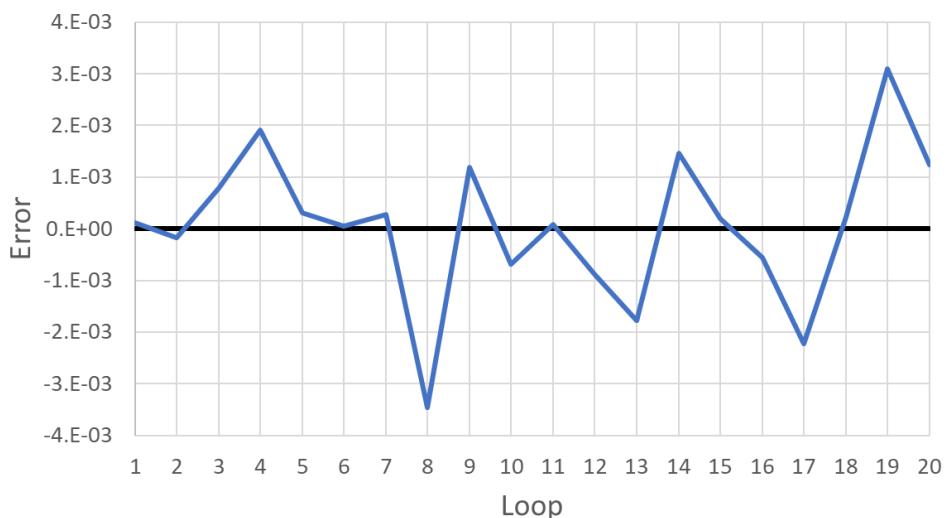
<Appendix C. 그림- 2> Loop에 따른 오차의 평균: 최대 캠버 위치

DEL: Mean of Errors by Loop - Max Thickness



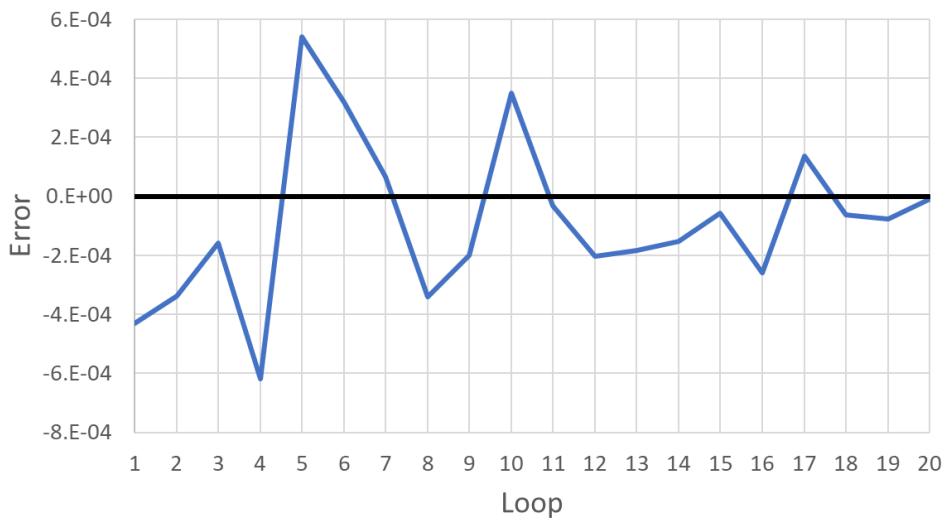
<Appendix C. 그림- 3> Loop에 따른 오차의 평균: 최대 두께

DEL: Mean of Errors by Loop - Cl0



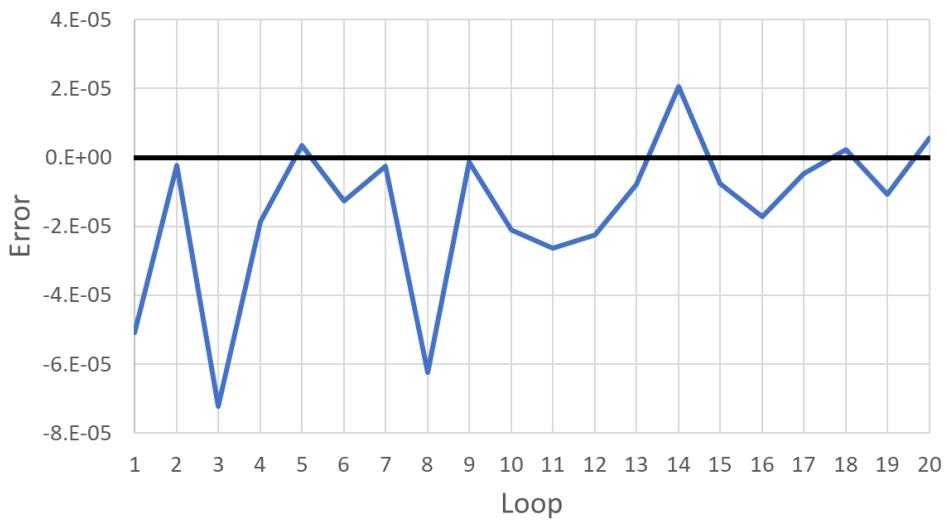
<Appendix C. 그림- 4> Loop에 따른 오차의 평균: 양력계수

DEL: Mean of Errors by Loop - Cla0

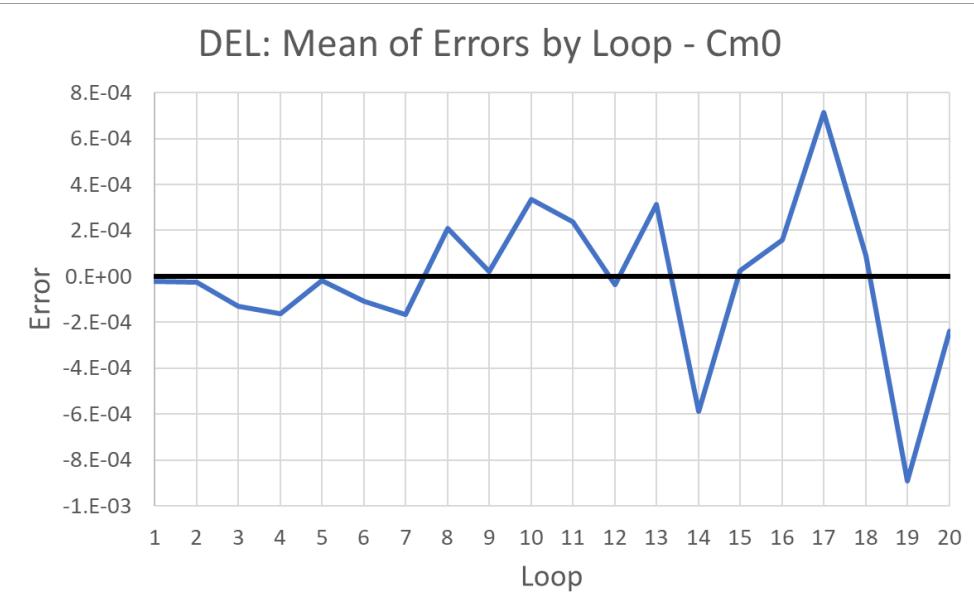


<Appendix C. 그림- 5> Loop에 따른 오차의 평균: 양력계수 기울기

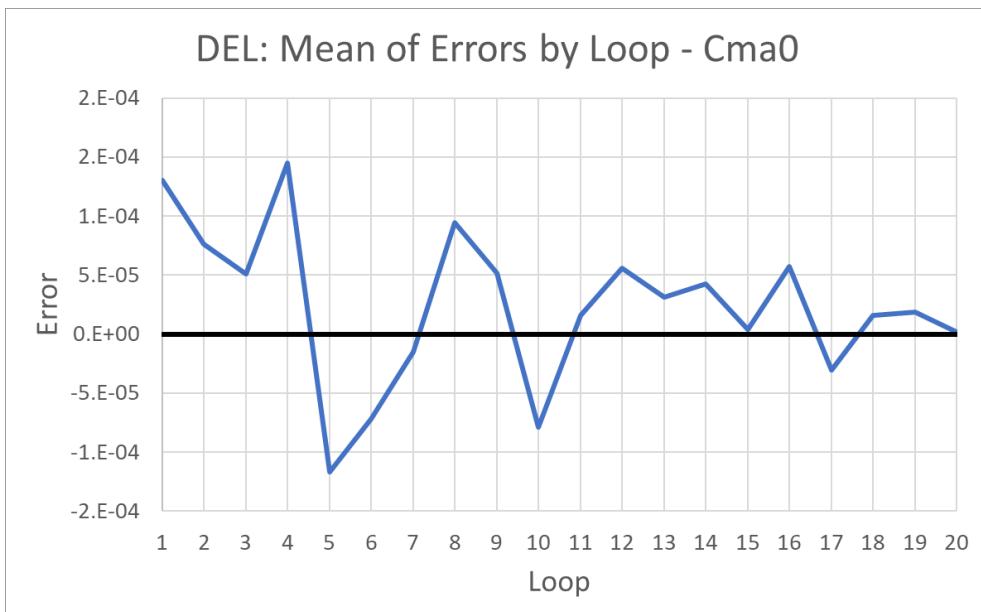
DEL: Mean of Errors by Loop - Cd0



<Appendix C. 그림- 6> Loop에 따른 오차의 평균: 항력계수

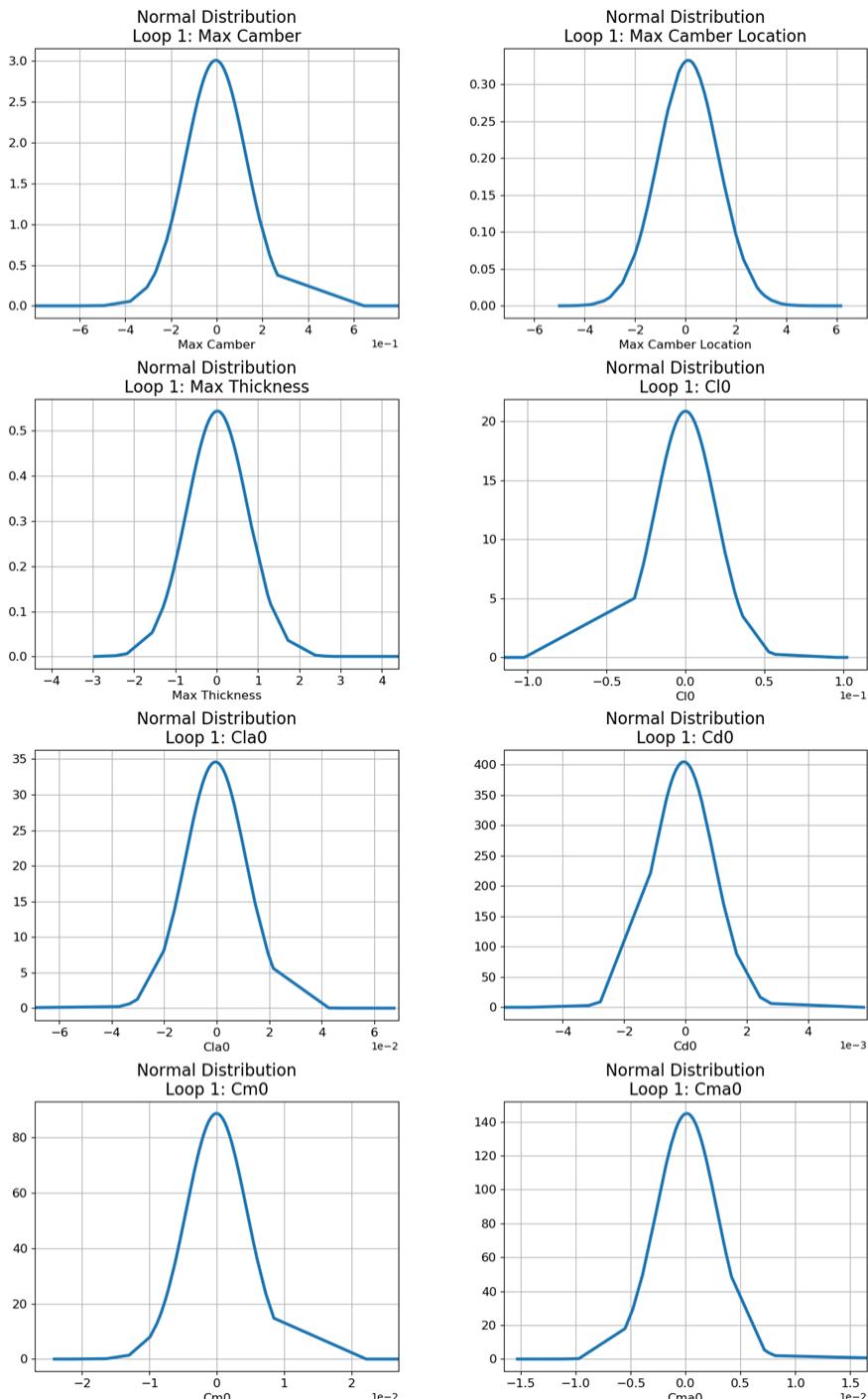


<Appendix C. 그림- 7> Loop에 따른 오차의 평균: 모멘트 계수

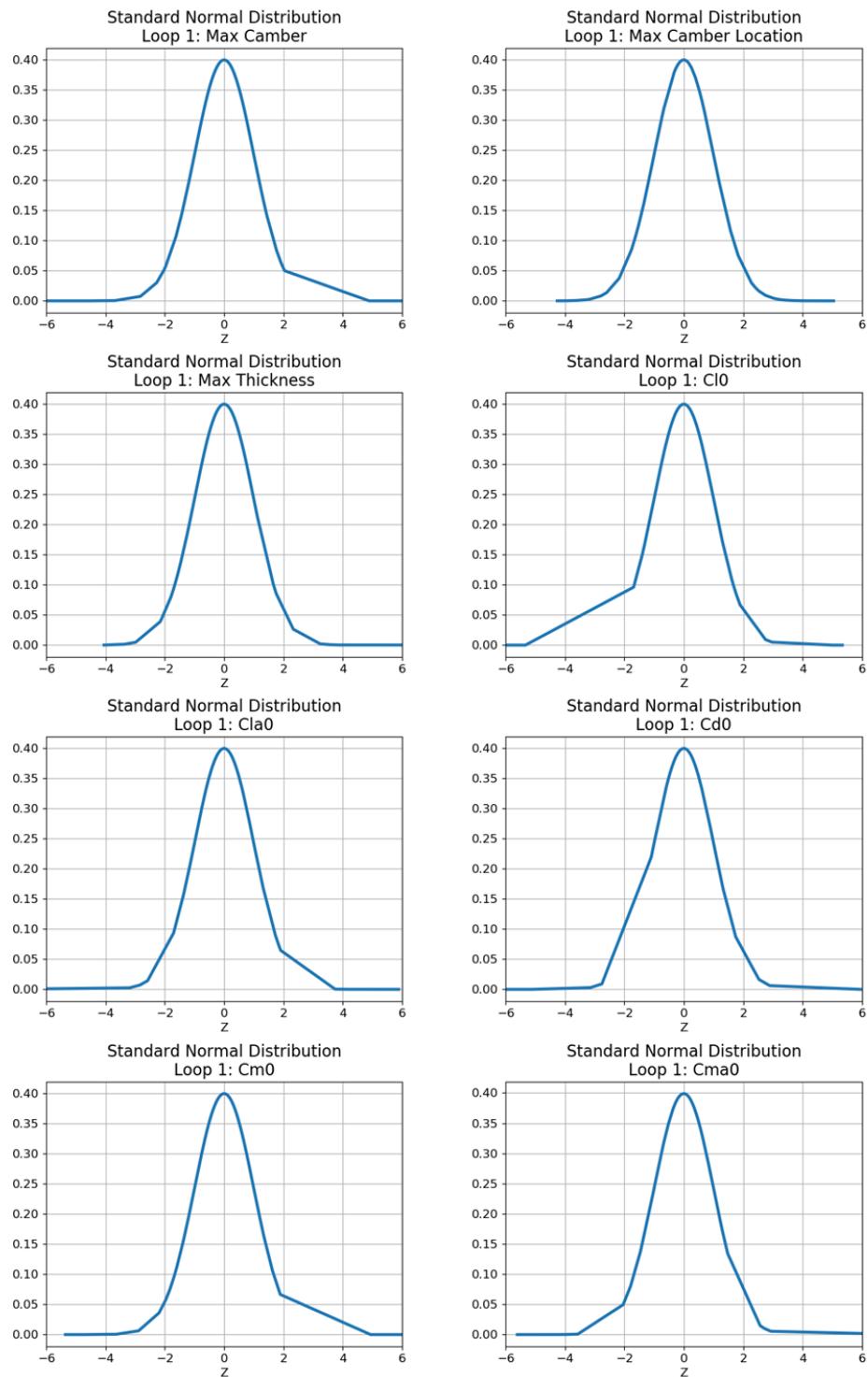


<Appendix C. 그림- 8> Loop에 따른 오차의 평균: 모멘트 계수 기울기

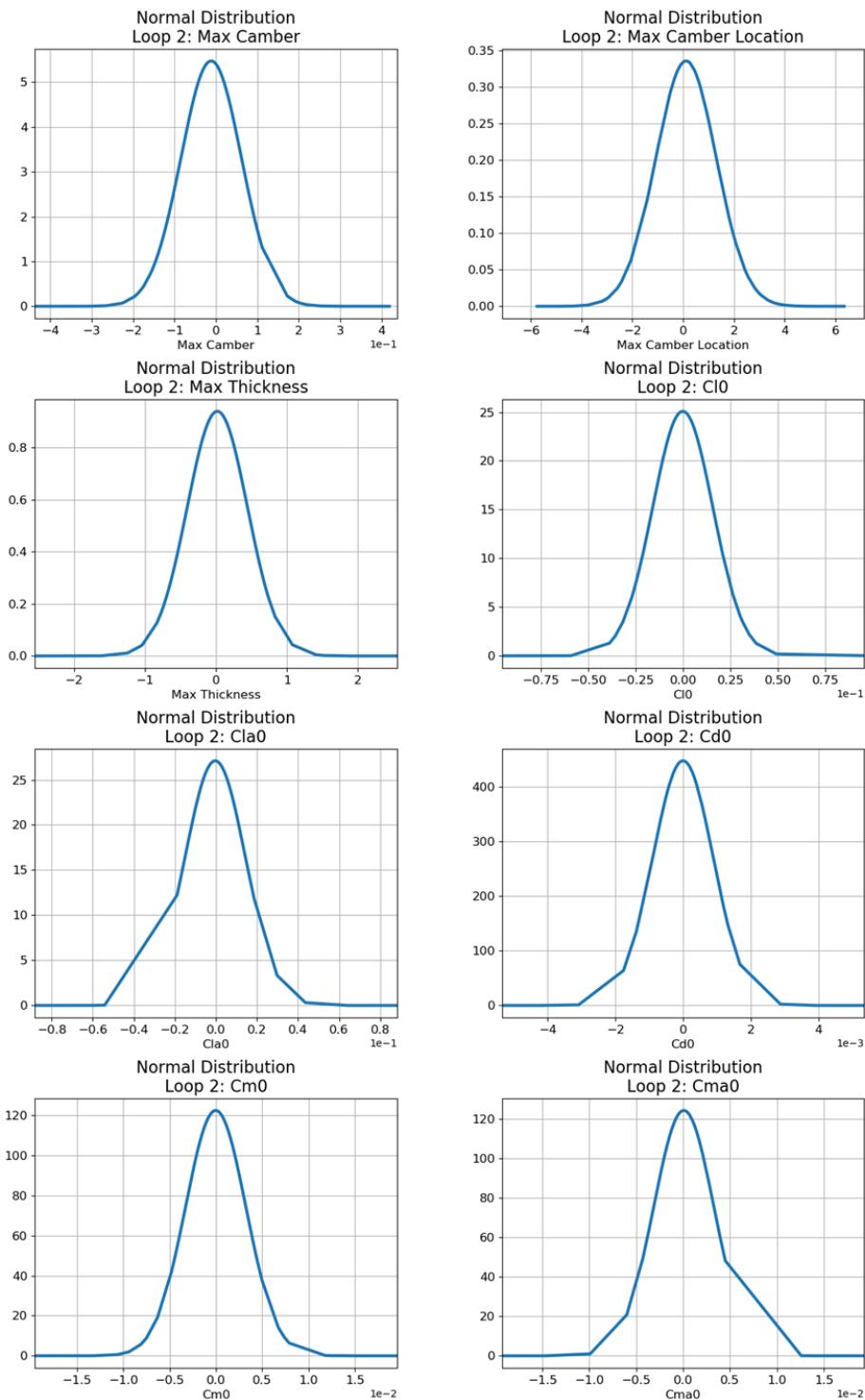
2. 정규분포도와 표준 정규분포도 ($\mu \pm 6\sigma$)



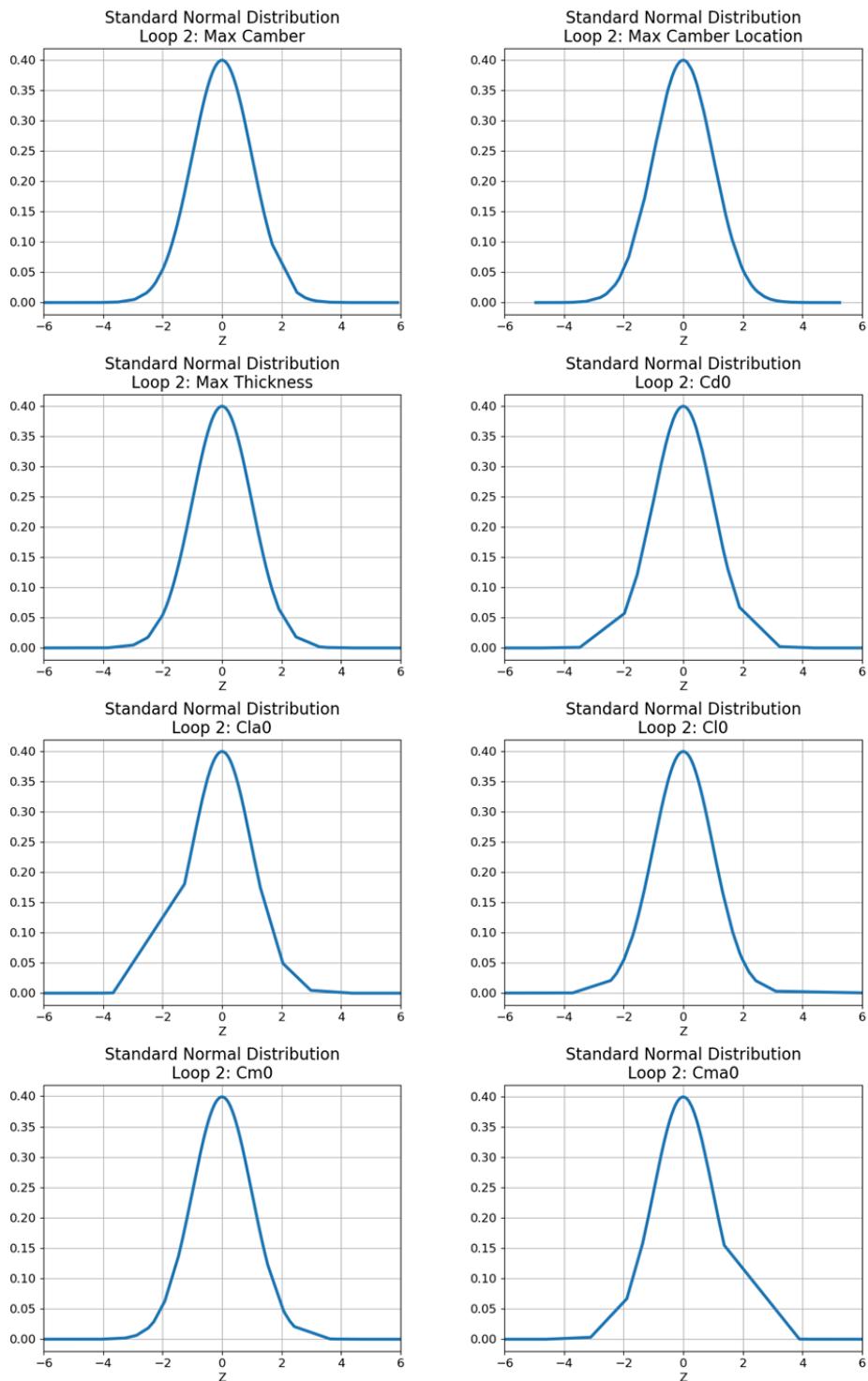
<Appendix C. 그림- 9> DEL Loop 1: 형상/성능 오차 정규분포



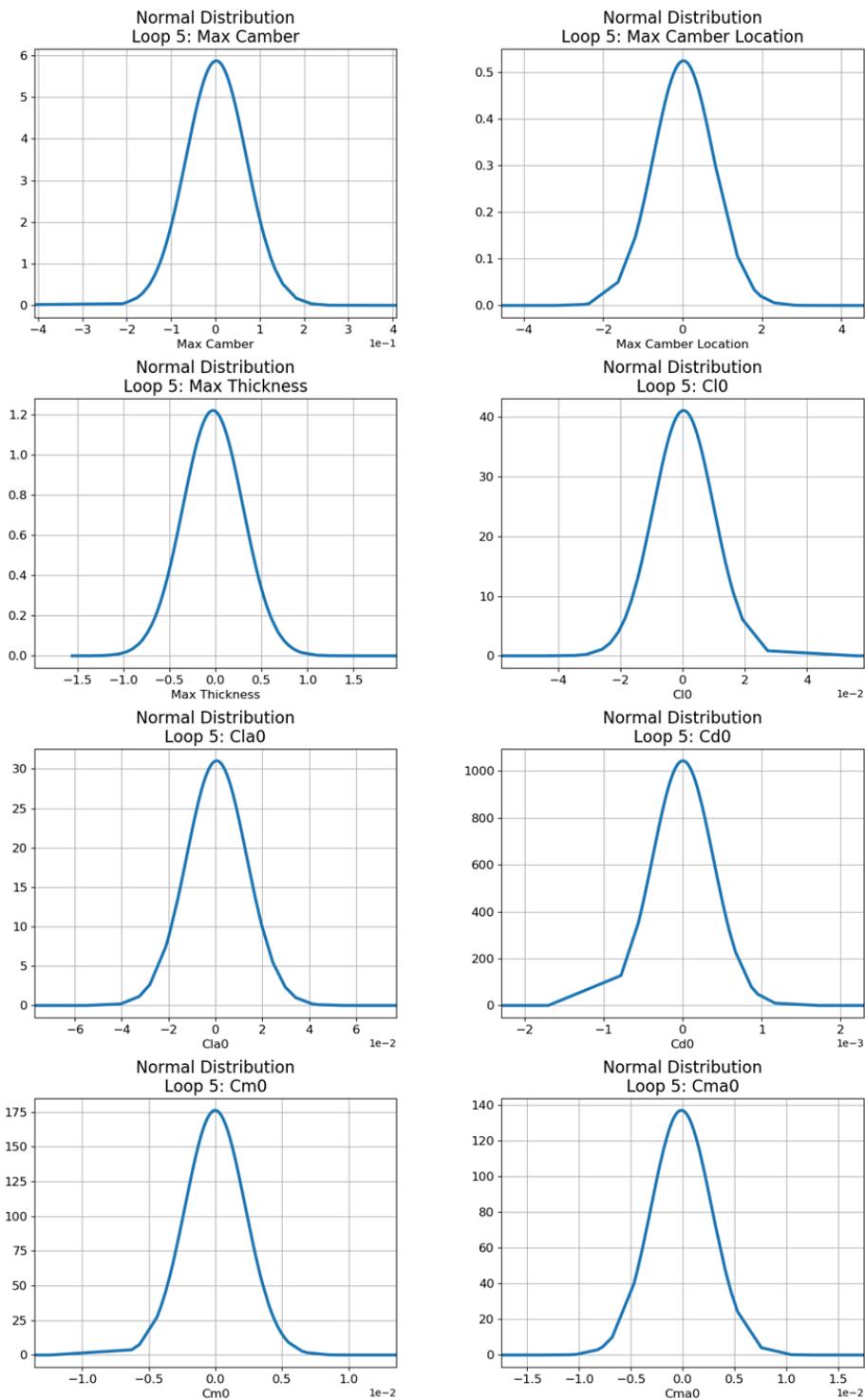
<Appendix C. 그림- 10> DEL Loop 1: 형상/성능 오차 표준정규분포



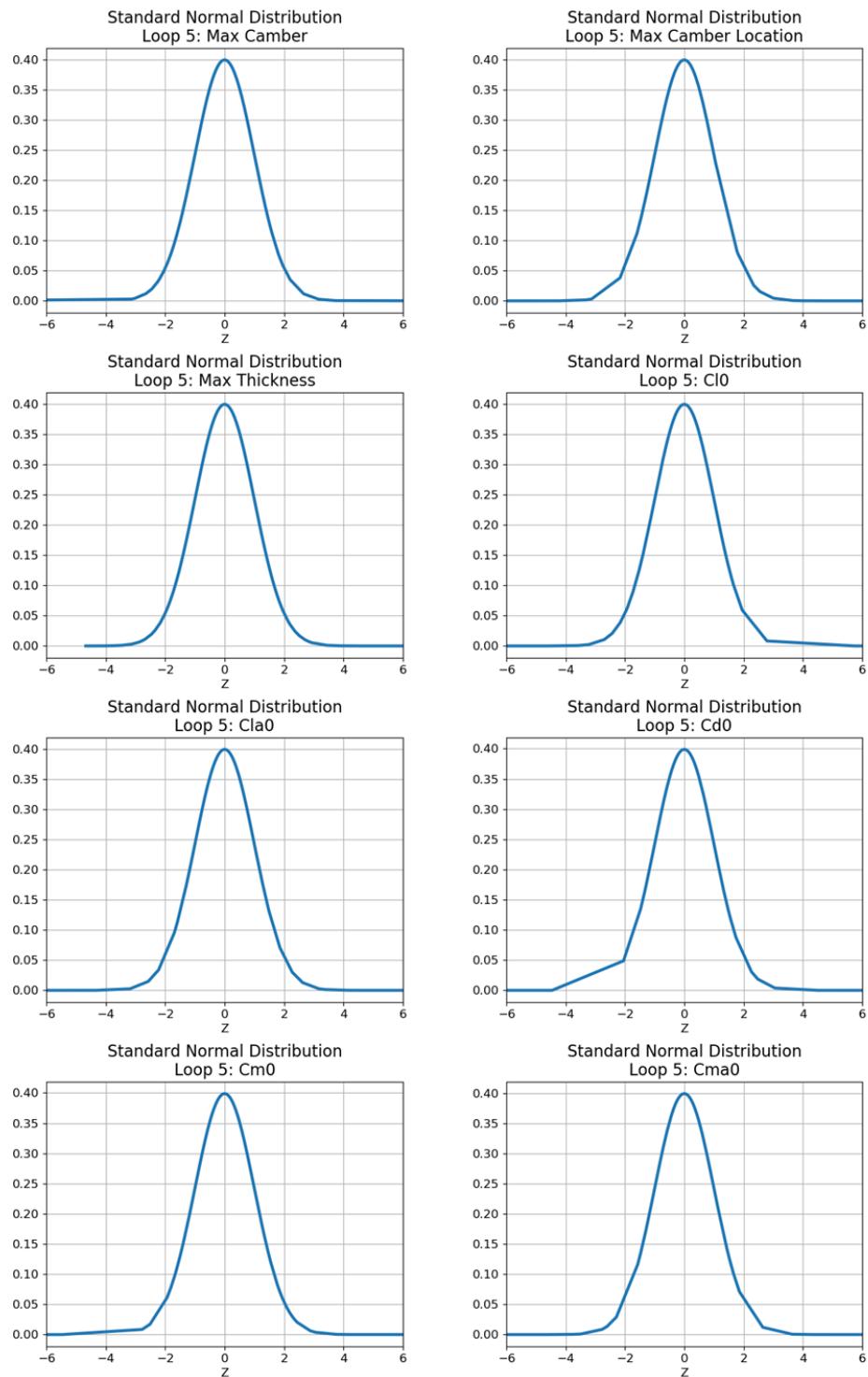
<Appendix C. 그림- 11> DEL Loop 2: 형상/성능 오차 정규분포



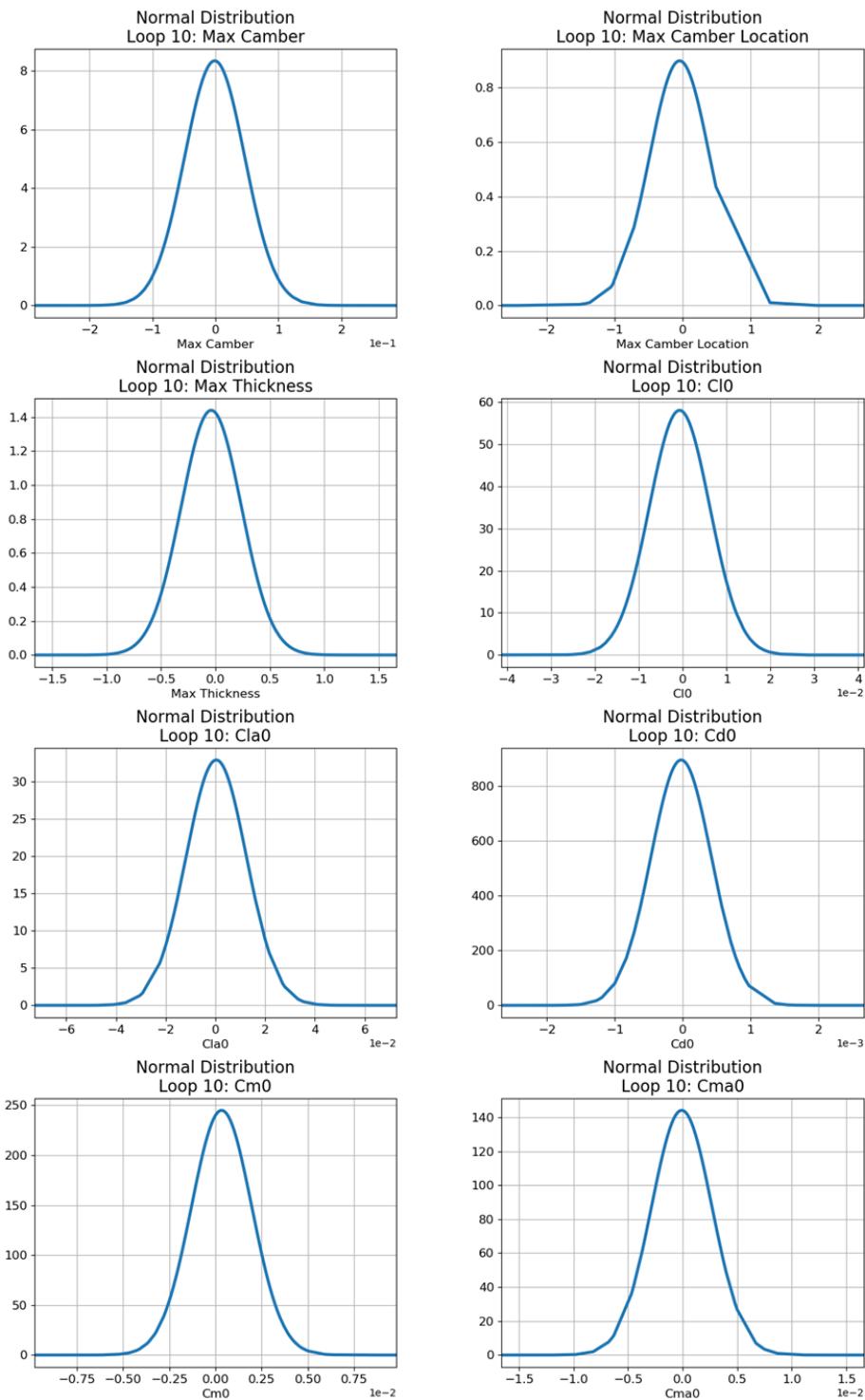
<Appendix C. 그림- 12> DEL Loop 2: 형상/성능 오차 표준정규분포



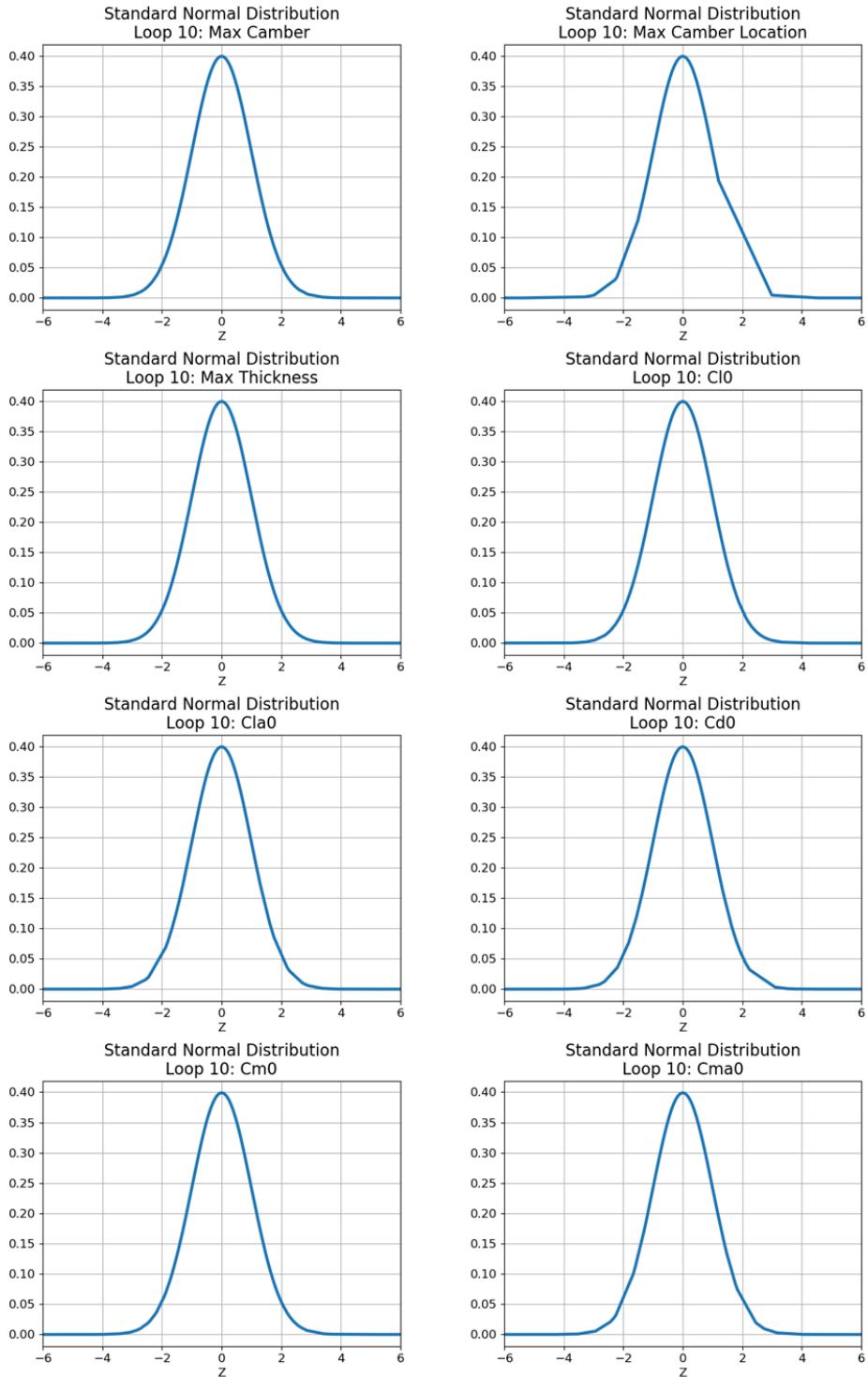
<Appendix C. 그림- 13> DEL Loop 5: 형상/성능 오차 정규분포



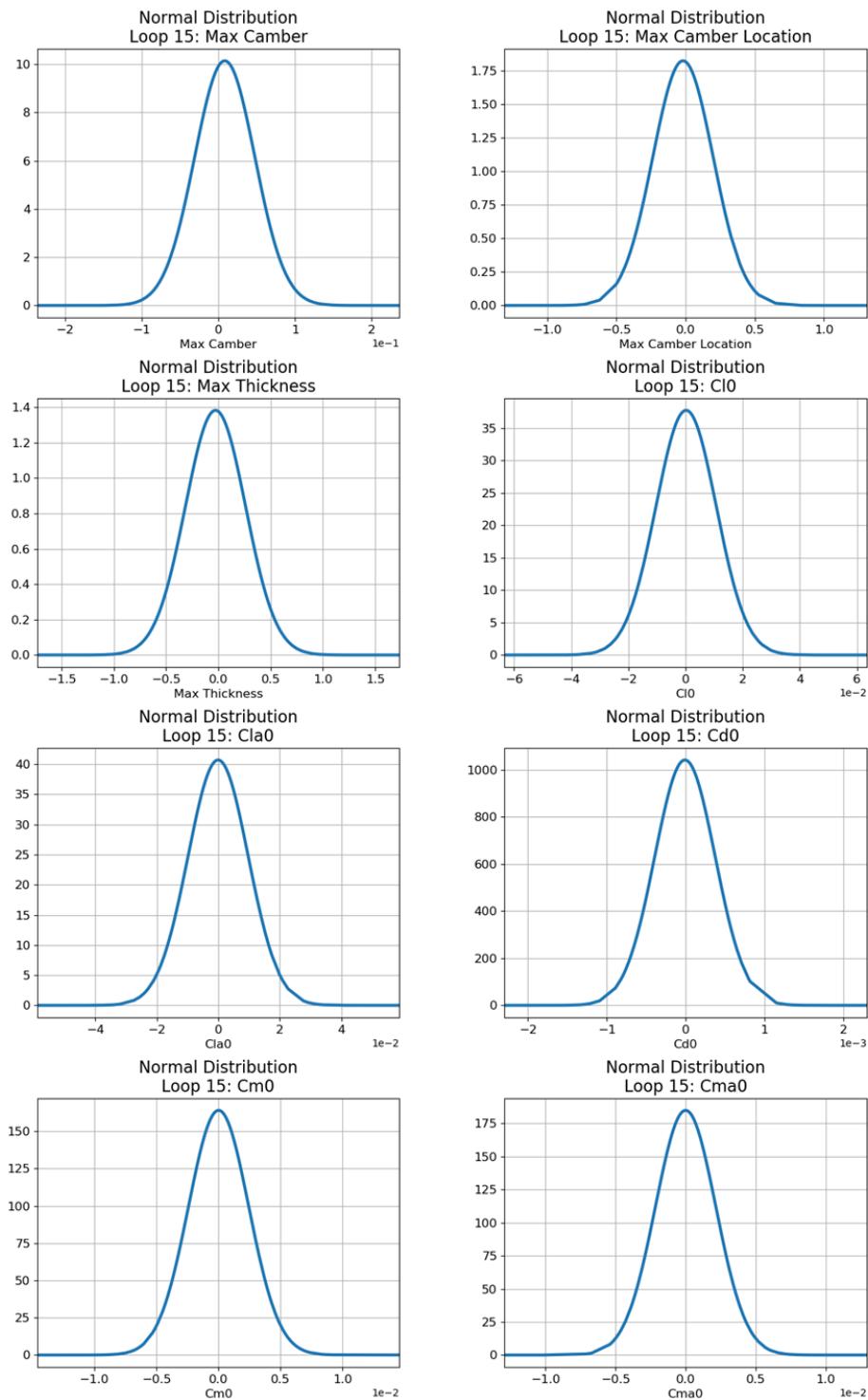
<Appendix C. 그림- 14> DEL Loop 5: 형상/성능 오차 표준정규분포



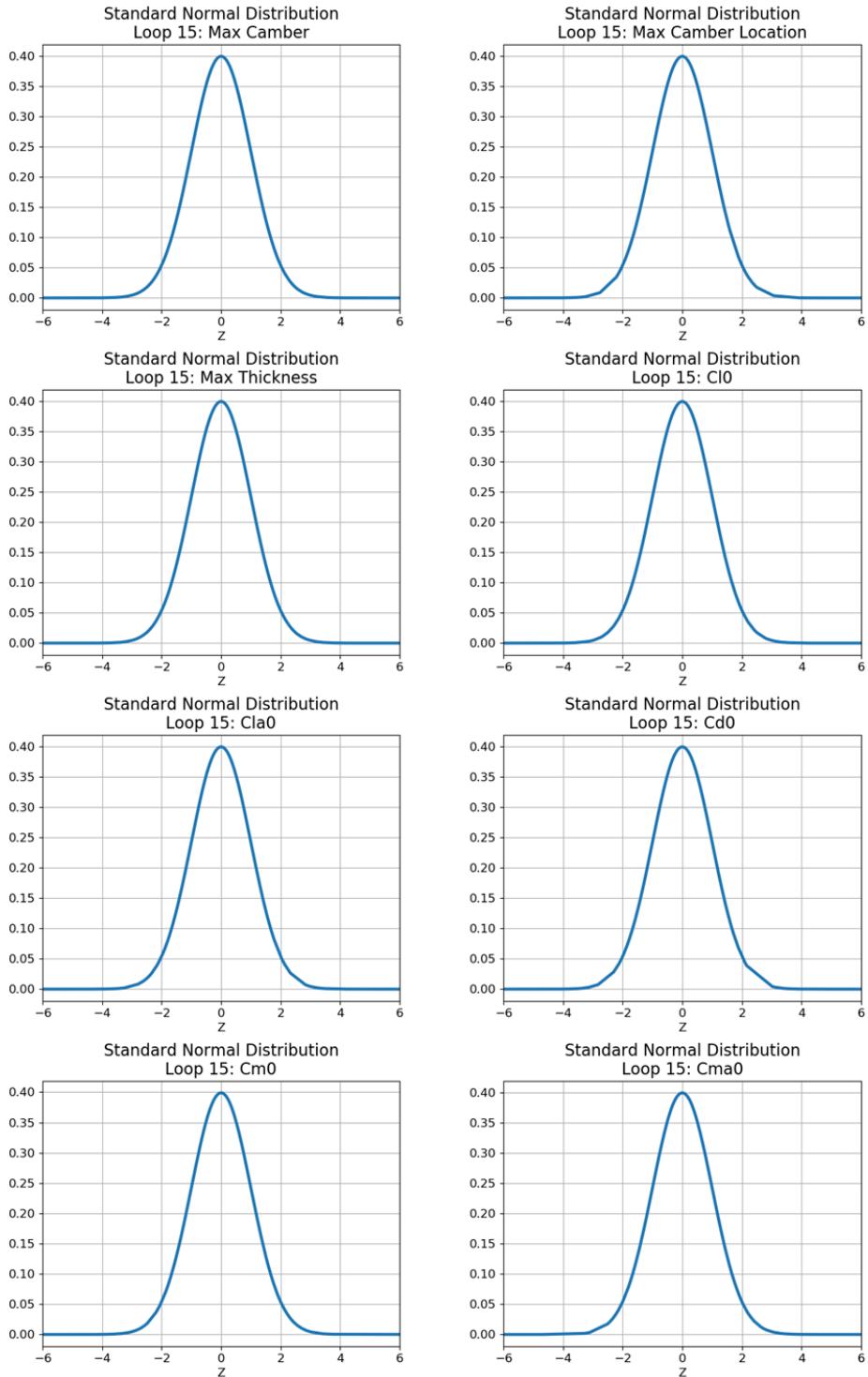
<Appendix C. 그림- 15> DEL Loop 10: 형상/성능 오차 정규분포



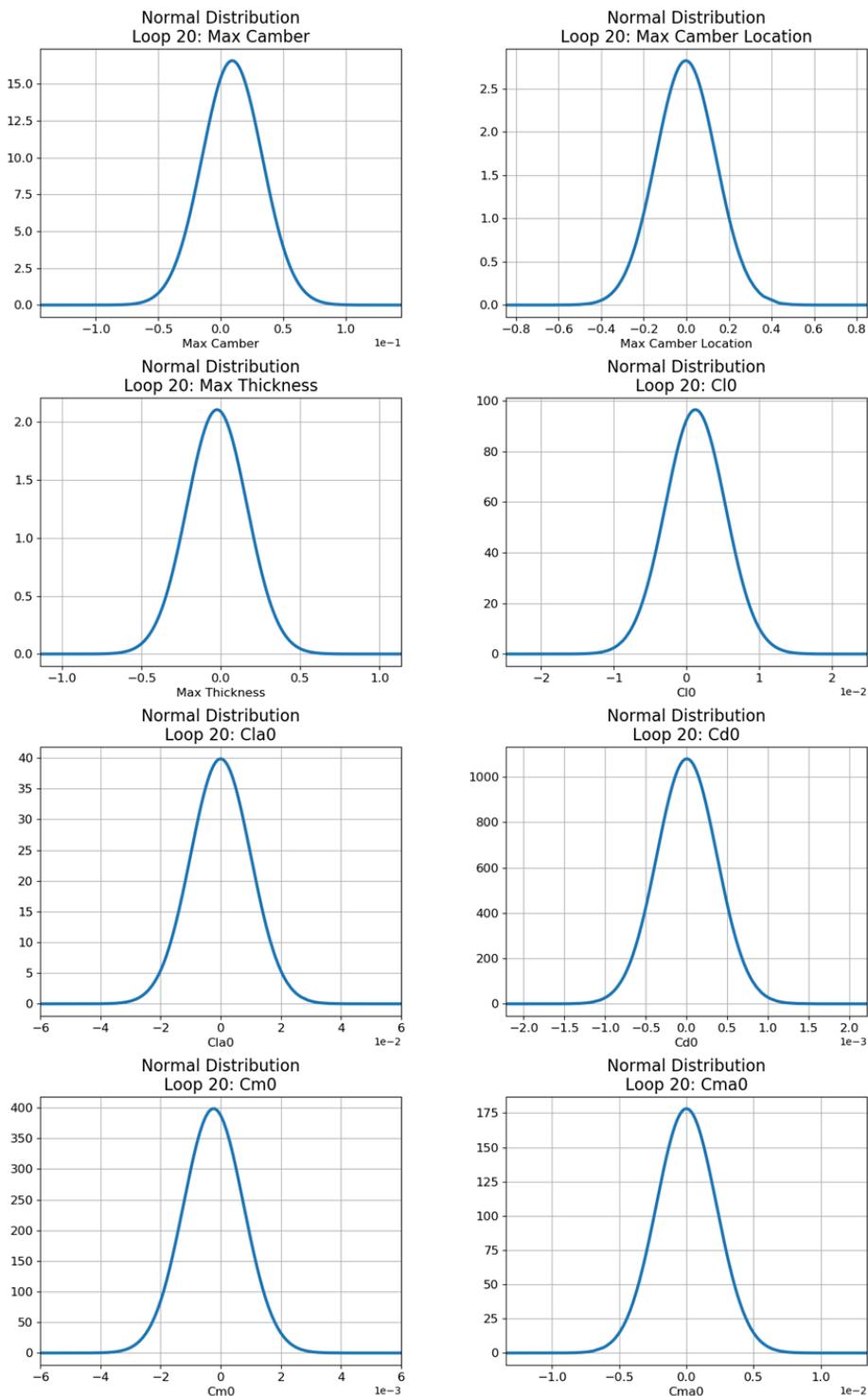
<Appendix C. 그림- 16> DEL Loop 10: 형상/성능 오차 표준정규분포



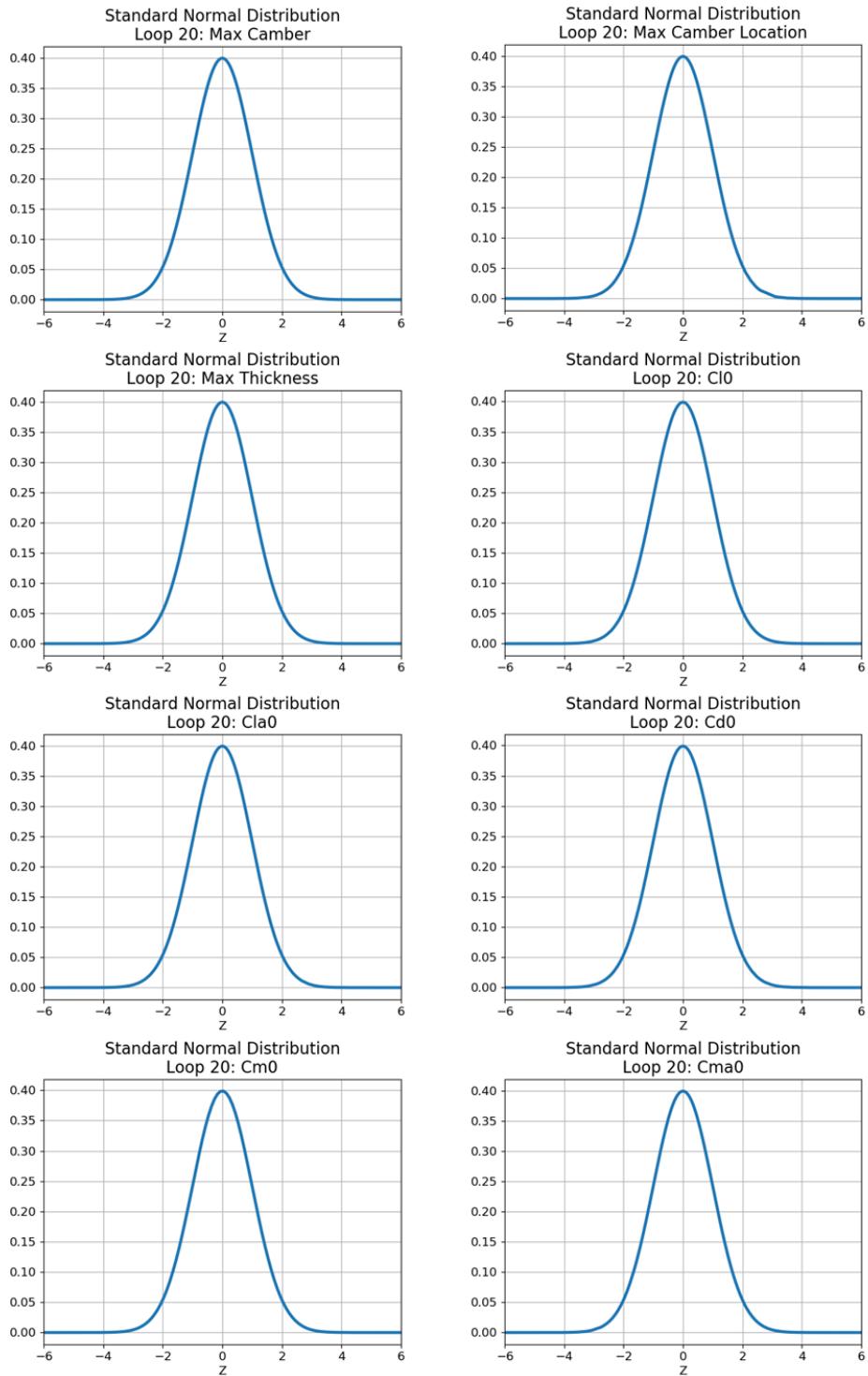
<Appendix C. 그림- 17> DEL Loop 15: 형상/성능 오차 정규분포



<Appendix C. 그림- 18> DEL Loop 15: 형상/성능 오차 표준정규분포



<Appendix C. 그림- 19> DEL Loop 20: 형상/성능 오차 정규분포



<Appendix C. 그림- 20> DEL Loop 20: 형상/성능 오차 표준정규분포

3. 3σ , 6σ

<Appendix C. 표- 5> DEL Loop 별 테스트 오차 3σ : 형상 파라미터

DEL Loop	Max Camber	Max Camber Location	Max Thickness
Loop 1	0.398091	3.602160	2.200737
Loop 2	0.218798	3.565050	1.275756
Loop 3	0.262060	2.865054	1.254435
Loop 4	0.183520	2.384442	1.147422
Loop 5	0.204083	2.284476	0.981570
Loop 6	0.259527	1.848768	1.217967
Loop 7	0.167391	1.779303	1.347321
Loop 8	0.196108	1.711152	1.240503
Loop 9	0.207010	1.591008	0.874920
Loop 10	0.143453	1.333518	0.831510
Loop 11	0.148577	0.952356	1.150548
Loop 12	0.191899	1.007304	0.697968
Loop 13	0.189628	0.792756	0.730728
Loop 14	0.151522	0.750480	1.029906
Loop 15	0.118046	0.655578	0.865293
Loop 16	0.124314	0.648024	1.163493
Loop 17	0.124903	0.489792	0.990969
Loop 18	0.094281	0.498051	0.713754
Loop 19	0.081715	0.426345	0.628860
Loop 20	0.072196	0.423924	0.568677

<Appendix C. 표- 6> DEL Loop 별 테스트 오차 3σ : 성능 파라미터

DEL Loop	C_{l0}	C_{la0}	C_{d0}	C_{m0}	C_{ma0}
Loop 1	0.057421	0.034607	0.002958	0.013499	0.008263
Loop 2	0.047715	0.044153	0.002674	0.009779	0.009641
Loop 3	0.052039	0.031067	0.002342	0.013643	0.006881
Loop 4	0.035072	0.024336	0.001305	0.008751	0.00559
Loop 5	0.029159	0.038581	0.001147	0.006791	0.008746
Loop 6	0.024066	0.036863	0.001684	0.006483	0.008399
Loop 7	0.029644	0.03576	0.002072	0.006984	0.008107
Loop 8	0.123705	0.029695	0.002994	0.026407	0.006638
Loop 9	0.022051	0.032784	0.001435	0.005422	0.007259
Loop 10	0.02063	0.036358	0.001336	0.004886	0.008306
Loop 11	0.02467	0.03842	0.001478	0.006281	0.008699
Loop 12	0.033128	0.032901	0.001777	0.008048	0.00738
Loop 13	0.02956	0.03115	0.001239	0.007207	0.007021
Loop 14	0.021132	0.03252	0.001233	0.004754	0.00734
Loop 15	0.031722	0.029422	0.001149	0.007294	0.006473
Loop 16	0.023883	0.033258	0.00122	0.005475	0.00734
Loop 17	0.018517	0.033673	0.00117	0.005086	0.007417
Loop 18	0.013711	0.027828	0.001131	0.003646	0.006219
Loop 19	0.013541	0.027614	0.001331	0.003335	0.006153
Loop 20	0.012411	0.030034	0.001109	0.003003	0.006717

<Appendix C. 표- 7> DEL Loop 별 테스트 오차 범위($\mu \pm 3\sigma$): 형상 파라미터

DEL Loop		Max Camber	Max Camber Location	Max Thickness
Loop 1	min	0.3946142	3.7186770	2.2175103
	max	-0.4015678	-3.4856430	-2.1839637
Loop 2	min	0.2073349	3.6850530	1.2957190
	max	-0.2302601	-3.4450470	-1.2557930
Loop 3	min	0.2563836	2.9313621	1.2522154
	max	-0.2677362	-2.7987459	-1.2566546
Loop 4	min	0.1991049	2.3389827	1.0938635
	max	-0.1679349	-2.4299013	-1.2009805
Loop 5	min	0.2060192	2.3067308	0.9549457
	max	-0.2021470	-2.2622212	-1.0081943
Loop 6	min	0.2541386	1.8913739	1.2149923
	max	-0.2649148	-1.8061621	-1.2209417
Loop 7	min	0.1670015	1.7940446	1.3289510
	max	-0.1677799	-1.7645614	-1.3656910
Loop 8	min	0.1539692	1.8374820	1.4465710
	max	-0.2382460	-1.5848220	-1.0344350
Loop 9	min	0.2285268	1.5405569	0.8088898
	max	-0.1854924	-1.6414591	-0.9409502
Loop 10	min	0.1422773	1.2894399	0.7929447
	max	-0.1446283	-1.3775961	-0.8700753
Loop 11	min	0.1532078	0.8993695	1.1073099
	max	-0.1439458	-1.0053425	-1.1937861
Loop 12	min	0.1725962	1.0408403	0.6837348
	max	-0.2112022	-0.9737677	-0.7122012
Loop 13	min	0.1726988	0.8017070	0.6854925
	max	-0.2065570	-0.7838050	-0.7759635

Loop 14	min	0.1513433	0.7930543	1.1241896
	max	-0.1517005	-0.7079057	-0.9356224
Loop 15	min	0.1264739	0.6369987	0.8384134
	max	-0.1096177	-0.6741573	-0.8921726
Loop 16	min	0.1181865	0.6388114	1.1391166
	max	-0.1304409	-0.6572366	-1.1878694
Loop 17	min	0.1181788	0.4652692	1.0816677
	max	-0.1316276	-0.5143148	-0.9002703
Loop 18	min	0.0976424	0.4877996	0.7317164
	max	-0.0909190	-0.5083024	-0.6957916
Loop 19	min	0.1005760	0.4543028	0.6256807
	max	-0.0628544	-0.3983872	-0.6320393
Loop 20	min	0.0814270	0.4213537	0.5457481
	max	-0.0629654	-0.4264944	-0.5916059

<Appendix C. 표- 8> DEL Loop 별 테스트 오차 범위($\mu \pm 3\sigma$): 성능 파라미터

DEL Loop		C10	Cla0	Cd0	Cm0	Cma0
Loop 1	min	0.0575302	0.0341774	0.0029071	0.0134766	0.0083928
	max	-0.0573110	-0.0350374	-0.0030089	-0.0135219	-0.0081328
Loop 2	min	0.0475447	0.0438139	0.0026715	0.0097541	0.0097175
	max	-0.0478853	-0.0444917	-0.0026760	-0.0098035	-0.0095646
Loop 3	min	0.0528316	0.0309075	0.0022693	0.0135123	0.0069318
	max	-0.0512456	-0.0312261	-0.0024139	-0.0137744	-0.0068301
Loop 4	min	0.0369892	0.0237184	0.0012862	0.0085877	0.0057352
	max	-0.0331544	-0.0249538	-0.0013235	-0.0089145	-0.0054447
Loop 5	min	0.0294689	0.0391217	0.0011501	0.0067714	0.0086287
	max	-0.0288494	-0.0380407	-0.0011432	-0.0068096	-0.0088624
Loop 6	min	0.0241198	0.0371821	0.0016715	0.0063755	0.0083274
	max	-0.0240125	-0.0365429	-0.0016966	-0.0065901	-0.0084713

Loop 7	min	0.0299237	0.0358274	0.0020693	0.0068161	0.0080916
	max	-0.0293641	-0.0356926	-0.0020743	-0.0071522	-0.0081216
Loop 8	min	0.1202407	0.0293557	0.0029312	0.0266146	0.0067325
	max	-0.1271699	-0.0300349	-0.0030561	-0.0261989	-0.0065441
Loop 9	min	0.0232418	0.0325826	0.0014338	0.0054417	0.0073105
	max	-0.0208612	-0.0329848	-0.0014366	-0.0054028	-0.0072069
Loop 10	min	0.0199442	0.0367067	0.0013146	0.0052201	0.0082270
	max	-0.0213151	-0.0360091	-0.0013568	-0.0045511	-0.0083842
Loop 11	min	0.0247473	0.0383865	0.0014520	0.0065193	0.0087149
	max	-0.0245933	-0.0384525	-0.0015046	-0.0060429	-0.0086836
Loop 12	min	0.0322517	0.0326976	0.0017541	0.0080100	0.0074356
	max	-0.0340039	-0.0331038	-0.0017992	-0.0080856	-0.0073239
Loop 13	min	0.0277815	0.0309671	0.0012310	0.0075191	0.0070523
	max	-0.0313380	-0.0313321	-0.0012466	-0.0068950	-0.0069895
Loop 14	min	0.0225944	0.0323668	0.0012540	0.0041644	0.0073822
	max	-0.0196694	-0.0326732	-0.0012129	-0.0053436	-0.0072976
Loop 15	min	0.0319149	0.0293636	0.0011419	0.0073190	0.0064766
	max	-0.0315285	-0.0294796	-0.0011571	-0.0072686	-0.0064689
Loop 16	min	0.0233244	0.0329977	0.0012024	0.0056338	0.0073976
	max	-0.0244416	-0.0335177	-0.0012369	-0.0053156	-0.0072830
Loop 17	min	0.0162984	0.0338093	0.0011651	0.0057989	0.0073866
	max	-0.0207362	-0.0335371	-0.0011743	-0.0043721	-0.0074471
Loop 18	min	0.0139435	0.0277657	0.0011328	0.0037353	0.0062346
	max	-0.0134782	-0.0278908	-0.0011284	-0.0035569	-0.0062028
Loop 19	min	0.0166399	0.0275370	0.0013203	0.0024423	0.0061713
	max	-0.0104423	-0.0276919	-0.0013418	-0.0042281	-0.0061345
Loop 20	min	0.0136481	0.0300256	0.0011145	0.0027645	0.0067190
	max	-0.0111737	-0.0300428	-0.0011031	-0.0032408	-0.0067148

<Appendix C. 표- 9> DEL Loop 별 테스트 오차 6σ : 형상 파라미터

DEL Loop	Max Camber	Max Camber Location	Max Thickness
Loop 1	0.796182	7.20432	4.401474
Loop 2	0.437595	7.1301	2.551512
Loop 3	0.5241198	5.730108	2.50887
Loop 4	0.3670398	4.768884	2.294844
Loop 5	0.4081662	4.568952	1.96314
Loop 6	0.5190534	3.697536	2.435934
Loop 7	0.3347814	3.558606	2.694642
Loop 8	0.3922152	3.422304	2.481006
Loop 9	0.4140192	3.182016	1.74984
Loop 10	0.2869056	2.667036	1.66302
Loop 11	0.2971536	1.904712	2.301096
Loop 12	0.3837984	2.014608	1.395936
Loop 13	0.3792558	1.585512	1.461456
Loop 14	0.3030438	1.50096	2.059812
Loop 15	0.2360916	1.311156	1.730586
Loop 16	0.2486274	1.296048	2.326986
Loop 17	0.2498064	0.979584	1.981938
Loop 18	0.1885614	0.996102	1.427508
Loop 19	0.1634304	0.85269	1.25772
Loop 20	0.1443924	0.847848	1.137354

<Appendix C. 표- 10> DEL Loop 별 테스트 오차 6σ : 성능 파라미터

DEL Loop	Cl0	Cla0	Cd0	Cm0	Cma0
Loop 1	0.114841	0.069215	0.005916	0.026998	0.016526
Loop 2	0.09543	0.088306	0.005348	0.019558	0.019282
Loop 3	0.104077	0.062134	0.004683	0.027287	0.013762
Loop 4	0.070144	0.048672	0.00261	0.017502	0.01118
Loop 5	0.058318	0.077162	0.002293	0.013581	0.017491
Loop 6	0.048132	0.073725	0.003368	0.012966	0.016799
Loop 7	0.059288	0.07152	0.004144	0.013968	0.016213
Loop 8	0.247411	0.059391	0.005987	0.052814	0.013277
Loop 9	0.044103	0.065567	0.00287	0.010844	0.014517
Loop 10	0.041259	0.072716	0.002671	0.009771	0.016611
Loop 11	0.049341	0.076839	0.002957	0.012562	0.017398
Loop 12	0.066256	0.065801	0.003553	0.016096	0.014759
Loop 13	0.05912	0.062299	0.002478	0.014414	0.014042
Loop 14	0.042264	0.06504	0.002467	0.009508	0.01468
Loop 15	0.063443	0.058843	0.002299	0.014588	0.012946
Loop 16	0.047766	0.066515	0.002439	0.010949	0.014681
Loop 17	0.037035	0.067346	0.002339	0.010171	0.014834
Loop 18	0.027422	0.055657	0.002261	0.007292	0.012437
Loop 19	0.027082	0.055229	0.002662	0.00667	0.012306
Loop 20	0.024822	0.060068	0.002218	0.006005	0.013434

<Appendix C. 표- 11> DEL Loop 별 테스트 오차 범위($\mu \pm 6\sigma$): 형상 파라미터

DEL Loop		Max Camber	Max Camber Location	Max Thickness
Loop 1	min	0.7927052	7.3208370	4.4182473
	max	-0.7996588	-7.0878030	-4.3847007
Loop 2	min	0.4261324	7.2501030	2.5714750
	max	-0.4490576	-7.0100970	-2.5315490
Loop 3	min	0.5184435	5.7964161	2.5066504
	max	-0.5297961	-5.6637999	-2.5110896
Loop 4	min	0.3826248	4.7234247	2.2412855
	max	-0.3514548	-4.8143433	-2.3484025
Loop 5	min	0.4101023	4.5912068	1.9365157
	max	-0.4062301	-4.5466972	-1.9897643
Loop 6	min	0.5136653	3.7401419	2.4329593
	max	-0.5244415	-3.6549301	-2.4389087
Loop 7	min	0.3343922	3.5733476	2.6762720
	max	-0.3351706	-3.5438644	-2.7130120
Loop 8	min	0.3500768	3.5486340	2.6870740
	max	-0.4343536	-3.2959740	-2.2749380
Loop 9	min	0.4355364	3.1315649	1.6838098
	max	-0.3925020	-3.2324671	-1.8158702
Loop 10	min	0.2857301	2.6229579	1.6244547
	max	-0.2880811	-2.7111141	-1.7015853
Loop 11	min	0.3017846	1.8517255	2.2578579
	max	-0.2925226	-1.9576985	-2.3443341
Loop 12	min	0.3644954	2.0481443	1.3817028
	max	-0.4031014	-1.9810717	-1.4101692
Loop 13	min	0.3623267	1.5944630	1.4162205
	max	-0.3961849	-1.5765610	-1.5066915

Loop 14	min	0.3028652	1.5435343	2.1540956
	max	-0.3032224	-1.4583857	-1.9655284
Loop 15	min	0.2445197	1.2925767	1.7037064
	max	-0.2276635	-1.3297353	-1.7574656
Loop 16	min	0.2425002	1.2868354	2.3026096
	max	-0.2547546	-1.3052606	-2.3513624
Loop 17	min	0.2430820	0.9550612	2.0726367
	max	-0.2565308	-1.0041068	-1.8912393
Loop 18	min	0.1919231	0.9858506	1.4454704
	max	-0.1851997	-1.0063534	-1.4095456
Loop 19	min	0.1822912	0.8806478	1.2545407
	max	-0.1445696	-0.8247322	-1.2608993
Loop 20	min	0.1536232	0.8452777	1.1144251
	max	-0.1351616	-0.8504184	-1.1602829

<Appendix C. 표- 12> > DEL Loop 별 테스트 오차 범위($\mu \pm 6\sigma$): 성능 파라미터

DEL Loop		C10	Cla0	Cd0	Cm0	Cma0
Loop 1	min	0.1149508	0.0687848	0.0058651	0.0269758	0.0166556
	max	-0.1147316	-0.0696448	-0.0059669	-0.0270211	-0.0163955
Loop 2	min	0.0952597	0.0879667	0.0053453	0.0195329	0.0193585
	max	-0.0956003	-0.0886445	-0.0053497	-0.0195823	-0.0192056
Loop 3	min	0.1048702	0.0619743	0.0046108	0.0271557	0.0138127
	max	-0.1032842	-0.0622929	-0.0047554	-0.0274178	-0.0137110
Loop 4	min	0.0720610	0.0480544	0.0025910	0.0173387	0.0113251
	max	-0.0682262	-0.0492898	-0.0026283	-0.0176655	-0.0110346
Loop 5	min	0.0586280	0.0777029	0.0022968	0.0135619	0.0173743
	max	-0.0580085	-0.0766219	-0.0022898	-0.0136001	-0.0176080
Loop 6	min	0.0481859	0.0740446	0.0033556	0.0128584	0.0167268
	max	-0.0480786	-0.0734054	-0.0033806	-0.0130729	-0.0168707

Loop 7	min	0.0595676	0.0715874	0.0041411	0.0138003	0.0161981
	max	-0.0590081	-0.0714526	-0.0041462	-0.0141363	-0.0162281
Loop 8	min	0.2439460	0.0590510	0.0059249	0.0530214	0.0133707
	max	-0.2508752	-0.0597303	-0.0060497	-0.0526056	-0.0131824
Loop 9	min	0.0452933	0.0653663	0.0028690	0.0108639	0.0145693
	max	-0.0429126	-0.0657685	-0.0028718	-0.0108250	-0.0144657
Loop 10	min	0.0405738	0.0730646	0.0026503	0.0101058	0.0165326
	max	-0.0419447	-0.0723670	-0.0026925	-0.0094367	-0.0166898
Loop 11	min	0.0494176	0.0768060	0.0029304	0.0128005	0.0174141
	max	-0.0492636	-0.0768720	-0.0029830	-0.0123241	-0.0173828
Loop 12	min	0.0653795	0.0655983	0.0035307	0.0160578	0.0148153
	max	-0.0671317	-0.0660045	-0.0035759	-0.0161334	-0.0147036
Loop 13	min	0.0573413	0.0621167	0.0024698	0.0147262	0.0140732
	max	-0.0608977	-0.0624817	-0.0024854	-0.0141020	-0.0140104
Loop 14	min	0.0437263	0.0648868	0.0024875	0.0089184	0.0147221
	max	-0.0408013	-0.0651932	-0.0024463	-0.0100976	-0.0146375
Loop 15	min	0.0636366	0.0587851	0.0022914	0.0146128	0.0129494
	max	-0.0632502	-0.0589011	-0.0023066	-0.0145624	-0.0129417
Loop 16	min	0.0472074	0.0662554	0.0024220	0.0111085	0.0147378
	max	-0.0483247	-0.0667754	-0.0024565	-0.0107903	-0.0146233
Loop 17	min	0.0348156	0.0674825	0.0023348	0.0108844	0.0148035
	max	-0.0392534	-0.0672103	-0.0023440	-0.0094576	-0.0148640
Loop 18	min	0.0276544	0.0555940	0.0022634	0.0073814	0.0124533
	max	-0.0271891	-0.0557191	-0.0022589	-0.0072030	-0.0124215
Loop 19	min	0.0301811	0.0551514	0.0026513	0.0057774	0.0123242
	max	-0.0239835	-0.0553063	-0.0026728	-0.0075632	-0.0122873
Loop 20	min	0.0260590	0.0600598	0.0022233	0.0057672	0.0134359
	max	-0.0235846	-0.0600770	-0.0022118	-0.0062434	-0.0134317

Appendix D. DEL Loop 별 형상 그룹에 따른 오차

- 형상그룹의 분류는 139p 의 <표 3-54> 형상 범위 설정을 참조

1. MSE

<Appendix D. 표- 1> DEL 형상 그룹 별 MSE:

Normal Range 형상 파라미터

Loop	Max Camber	Max Camber Location	Max Thickness
1	0.002379	0.005026	0.231741
2	0.001135	0.002303	0.167858
3	0.001681	0.002114	0.097951
4	0.001891	0.006379	0.188045
5	0.000947	0.00133	0.102483
6	0.000861	0.001239	0.059562
7	0.001064	0.001072	0.064576
8	0.003485	0.018039	0.306011
9	0.001400	0.005239	0.133806
10	0.000588	0.001757	0.102967
11	0.001111	0.004049	0.121783
12	0.000988	0.001921	0.043079
13	0.000853	0.000881	0.062432
14	0.000645	0.002072	0.097376
15	0.000532	0.000807	0.05858
16	0.000359	0.000484	0.037188
17	0.001259	0.001165	0.082954
18	0.000255	0.00058	0.028056
19	0.001035	0.001744	0.030963
20	0.000510	0.000392	0.025055

<Appendix D. 표- 2> DEL 형상 그룹 별 MSE:

Normal Range 성능 파라미터

Loop	Cl0	Cla0	Cd0	Cm0	Cma0
1	4.0.E-05	1.0.E-06	0.0.E+00	3.0.E-06	0.0.E+00
2	1.5.E-05	1.0.E-06	0.0.E+00	1.0.E-06	0.0.E+00
3	1.9.E-05	1.0.E-06	0.0.E+00	1.0.E-06	0.0.E+00
4	9.0.E-06	1.0.E-06	0.0.E+00	1.0.E-06	0.0.E+00
5	1.0.E-05	0.0.E+00	0.0.E+00	1.0.E-06	0.0.E+00
6	8.0.E-06	0.0.E+00	0.0.E+00	1.0.E-06	0.0.E+00
7	1.2.E-05	0.0.E+00	0.0.E+00	1.0.E-06	0.0.E+00
8	1.9.E-05	1.0.E-06	0.0.E+00	3.0.E-06	0.0.E+00
9	6.0.E-06	1.0.E-06	0.0.E+00	0.0.E+00	0.0.E+00
10	6.0.E-06	0.0.E+00	0.0.E+00	1.0.E-06	0.0.E+00
11	9.0.E-06	0.0.E+00	0.0.E+00	1.0.E-06	0.0.E+00
12	8.0.E-06	0.0.E+00	0.0.E+00	1.0.E-06	0.0.E+00
13	8.0.E-06	0.0.E+00	0.0.E+00	0.0.E+00	0.0.E+00
14	1.5.E-05	0.0.E+00	0.0.E+00	1.0.E-06	0.0.E+00
15	4.0.E-06	0.0.E+00	0.0.E+00	0.0.E+00	0.0.E+00
16	4.0.E-06	0.0.E+00	0.0.E+00	0.0.E+00	0.0.E+00
17	2.0.E-05	0.0.E+00	0.0.E+00	2.0.E-06	0.0.E+00
18	2.0.E-06	0.0.E+00	0.0.E+00	0.0.E+00	0.0.E+00
19	2.2.E-05	0.0.E+00	0.0.E+00	2.0.E-06	0.0.E+00
20	5.0.E-06	0.0.E+00	0.0.E+00	0.0.E+00	0.0.E+00

<Appendix D. 표- 3> DEL 형상 그룹 별 MSE:

Zero Max. Camber 형상 파라미터

Loop	Max Camber	Max Camber Location	Max Thickness
1	0.000000	11.569419	0.000719
2	0.000000	10.011039	0.003851
3	0.000000	7.913803	0.001183
4	0.002578	5.329252	0.006844
5	0.000000	5.227281	0.004296
6	0.000000	3.083359	0.002343
7	0.000678	2.904761	0.013293
8	0.000000	2.577452	0.026825
9	0.000000	2.304870	0.008286
10	0.000004	1.655053	0.005867
11	0.000000	0.852809	0.014820
12	0.000000	0.942243	0.002368
13	0.000000	0.619700	0.004072
14	0.000000	0.535869	0.008750
15	0.000010	0.402391	0.003595
16	0.000000	0.375902	0.003524
17	0.000086	0.213537	0.017195
18	0.000001	0.243116	0.002767
19	0.000034	0.180391	0.002105
20	0.000000	0.188927	0.002246

<Appendix D. 표- 4> DEL 형상 그룹 별 MSE:

Zero Max. Camber 성능 파라미터

Loop	Cl0	Cla0	Cd0	Cm0	Cma0
1	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00
2	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00
3	0.00.E+00	0.00.E+00	1.00.E-06	0.00.E+00	0.00.E+00
4	3.60.E-05	0.00.E+00	0.00.E+00	2.00.E-06	0.00.E+00
5	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00
6	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00
7	1.40.E-05	0.00.E+00	0.00.E+00	1.00.E-06	0.00.E+00
8	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00
9	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00
10	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00
11	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00
12	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00
13	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00
14	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00
15	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00
16	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00
17	1.00.E-06	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00
18	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00
19	1.00.E-06	0.00.E+00	1.00.E-06	0.00.E+00	0.00.E+00
20	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00	0.00.E+00

<Appendix D. 표- 5> DEL 형상 그룹 별 MSE:

Out of Normal Range 형상 파라미터

Loop	Max Camber	Max Camber Location	Max Thickness
1	0.022470	0.016426	0.665523
2	0.007010	0.007907	0.215952
3	0.009711	0.012053	0.212379
4	0.004518	0.008548	0.165095
5	0.005863	0.003694	0.123402
6	0.009713	0.006057	0.206636
7	0.003825	0.008255	0.253733
8	0.007469	0.017679	0.230060
9	0.006620	0.007867	0.095507
10	0.002905	0.003167	0.085807
11	0.003033	0.006015	0.172719
12	0.005633	0.004376	0.063791
13	0.005385	0.002201	0.069324
14	0.003214	0.004856	0.148486
15	0.002030	0.002972	0.099833
16	0.002246	0.001967	0.191737
17	0.002116	0.003452	0.137890
18	0.001254	0.001216	0.069013
19	0.001254	0.001897	0.051700
20	0.000777	0.000754	0.042802

<Appendix D. 표- 6> DEL 형상 그룹 별 MSE:

Out of Normal Range 성능 파라미터

Loop	Cl0	Cla0	Cd0	Cm0	Cma0
1	4.68.E-04	1.72.E-04	1.00.E-06	2.60.E-05	1.00.E-05
2	3.30.E-04	2.84.E-04	1.00.E-06	1.40.E-05	1.40.E-05
3	3.89.E-04	1.39.E-04	1.00.E-06	2.70.E-05	7.00.E-06
4	1.76.E-04	8.60.E-05	0.00.E+00	1.10.E-05	5.00.E-06
5	1.21.E-04	2.15.E-04	0.00.E+00	7.00.E-06	1.10.E-05
6	8.30.E-05	1.98.E-04	0.00.E+00	6.00.E-06	1.00.E-05
7	1.25.E-04	1.87.E-04	1.00.E-06	7.00.E-06	1.00.E-05
8	2.27.E-03	1.30.E-04	1.00.E-06	1.02.E-04	6.00.E-06
9	7.20.E-05	1.56.E-04	0.00.E+00	4.00.E-06	8.00.E-06
10	6.20.E-05	1.92.E-04	0.00.E+00	4.00.E-06	1.00.E-05
11	8.60.E-05	2.11.E-04	0.00.E+00	6.00.E-06	1.10.E-05
12	1.58.E-04	1.56.E-04	0.00.E+00	9.00.E-06	8.00.E-06
13	1.28.E-04	1.39.E-04	0.00.E+00	7.00.E-06	7.00.E-06
14	6.50.E-05	1.53.E-04	0.00.E+00	4.00.E-06	8.00.E-06
15	1.45.E-04	1.26.E-04	0.00.E+00	8.00.E-06	6.00.E-06
16	8.30.E-05	1.61.E-04	0.00.E+00	4.00.E-06	8.00.E-06
17	5.30.E-05	1.65.E-04	0.00.E+00	4.00.E-06	8.00.E-06
18	2.70.E-05	1.11.E-04	0.00.E+00	2.00.E-06	6.00.E-06
19	3.50.E-05	1.09.E-04	0.00.E+00	2.00.E-06	5.00.E-06
20	2.30.E-05	1.29.E-04	0.00.E+00	1.00.E-06	6.00.E-06

2. Average of Errors

<Appendix D. 표- 7> DEL 형상 그룹 별 오차의 평균:

Normal Range 형상 파라미터

Loop	Max Camber	Max Camber Location	Max Thickness
1	-0.003183	-0.021154	-0.053103
2	-0.007982	0.007284	0.066425
3	0.009019	-0.019898	-0.045637
4	0.030235	-0.066123	-0.280697
5	0.004249	-0.018349	-0.132744
6	-0.004596	0.006714	-0.044201
7	0.002803	0.006350	-0.031804
8	-0.040485	0.118649	0.403411
9	0.024564	-0.057573	-0.198291
10	0.002548	-0.028223	-0.142866
11	0.017450	-0.050263	-0.224905
12	-0.017844	0.029872	0.013488
13	-0.016809	0.004827	-0.023513
14	0.004420	0.037715	0.178646
15	0.011274	-0.015923	-0.122471
16	-0.007816	-0.006525	-0.054770
17	-0.017791	-0.014812	0.122826
18	0.004262	-0.014194	0.026566
19	0.026854	0.030741	0.018319
20	0.013791	-0.008943	-0.012170

<Appendix D. 표- 8> DEL 형상 그룹 별 오차의 평균:

Normal Range 성능 파라미터

Loop	Cl0	Cla0	Cd0	Cm0	Cma0
1	-1.41.E-03	-1.06.E-04	6.00.E-06	4.50.E-04	3.10.E-05
2	-3.57.E-04	-3.60.E-05	2.00.E-06	3.30.E-05	1.20.E-05
3	2.68.E-04	7.60.E-05	5.00.E-06	9.80.E-05	-1.70.E-05
4	5.91.E-04	-1.83.E-04	4.00.E-06	2.55.E-04	1.50.E-05
5	-4.94.E-04	-1.19.E-04	-1.00.E-06	1.97.E-04	1.00.E-05
6	-3.83.E-04	4.70.E-05	-9.00.E-06	-3.30.E-05	-1.00.E-05
7	5.66.E-04	0.00.E+00	-6.00.E-06	-1.89.E-04	-1.00.E-06
8	7.51.E-04	-4.60.E-05	-2.00.E-05	-9.83.E-04	4.40.E-05
9	3.72.E-04	-5.20.E-05	9.00.E-06	3.06.E-04	-4.00.E-06
10	-1.08.E-03	3.40.E-05	-1.00.E-06	3.99.E-04	-1.60.E-05
11	-2.31.E-04	0.00.E+00	0.00.E+00	3.48.E-04	-8.00.E-06
12	-8.84.E-04	6.00.E-05	-2.00.E-05	-9.30.E-05	-9.00.E-06
13	-1.98.E-03	4.00.E-06	-7.00.E-06	3.76.E-04	2.00.E-06
14	2.60.E-03	2.00.E-06	1.00.E-06	-8.14.E-04	8.00.E-06
15	3.49.E-04	1.00.E-05	-4.00.E-06	-2.40.E-05	-1.50.E-05
16	-1.38.E-03	-2.00.E-06	-5.00.E-06	3.24.E-04	-4.00.E-06
17	-2.98.E-03	8.00.E-06	2.80.E-05	9.69.E-04	9.00.E-06
18	-3.30.E-05	-1.20.E-05	1.40.E-05	2.07.E-04	5.00.E-06
19	4.23.E-03	1.90.E-05	-4.00.E-06	-1.15.E-03	-2.00.E-06
20	1.25.E-03	-1.00.E-06	8.00.E-06	-1.85.E-04	0.00.E+00

**<Appendix D. 표- 9> DEL 형상 그룹 별 오차의 평균:
Zero Max. Camber 형상 파라미터**

Loop	Max Camber	Max Camber Location	Max Thickness
1	0.000000	0.957772	0.021078
2	0.000000	0.737510	0.030618
3	0.000000	0.643652	-0.012290
4	0.005713	-0.007512	-0.028038
5	0.000000	0.258107	-0.035842
6	0.000000	0.240643	-0.023660
7	0.002105	0.041366	-0.013561
8	0.000000	0.297294	0.144103
9	0.000000	-0.043172	-0.066017
10	0.000164	-0.178434	-0.045334
11	0.000000	-0.155500	-0.083590
12	0.000000	0.046876	-0.015405
13	0.000000	0.009579	-0.040680
14	0.000013	0.063300	0.079981
15	0.000303	-0.044082	-0.041108
16	0.000012	-0.032854	-0.039245
17	0.002647	-0.029319	0.103894
18	0.000027	-0.013064	0.023175
19	0.002688	0.053784	-0.007742
20	0.000015	0.012899	0.001608

<Appendix D. 표- 10> DEL 형상 그룹 별 오차의 평균:

Zero Max. Camber 성능 파라미터

Loop	Cl0	Cla0	Cd0	Cm0	Cma0
1	-7.0.E-06	7.7.E-05	4.0.E-06	0.0.E+00	-1.1.E-05
2	-3.0.E-06	6.0.E-05	-6.0.E-06	0.0.E+00	2.0.E-06
3	6.0.E-06	-4.5.E-05	-9.5.E-05	0.0.E+00	3.0.E-06
4	7.0.E-04	-1.5.E-05	7.6.E-05	-1.6.E-04	4.0.E-06
5	-1.3.E-05	2.0.E-06	-1.0.E-05	0.0.E+00	0.0.E+00
6	7.0.E-06	-3.3.E-05	-1.0.E-06	0.0.E+00	2.0.E-06
7	3.1.E-04	4.0.E-06	4.0.E-05	-8.2.E-05	-1.0.E-06
8	-5.0.E-06	9.9.E-05	1.5.E-05	0.0.E+00	2.0.E-06
9	-1.0.E-06	-4.6.E-05	-6.6.E-05	0.0.E+00	1.0.E-06
10	2.1.E-05	-9.0.E-06	2.4.E-05	-6.0.E-06	-6.0.E-06
11	-3.0.E-06	8.0.E-06	-1.4.E-05	0.0.E+00	-3.0.E-06
12	2.0.E-06	-2.7.E-05	-7.2.E-05	0.0.E+00	6.0.E-06
13	0.0.E+00	-4.7.E-05	-4.2.E-05	0.0.E+00	-2.0.E-06
14	2.0.E-06	4.4.E-05	3.6.E-05	-1.0.E-06	1.0.E-06
15	3.7.E-05	-6.0.E-06	4.5.E-05	-9.0.E-06	-2.0.E-06
16	3.0.E-06	-1.6.E-05	-2.0.E-06	0.0.E+00	3.0.E-06
17	3.3.E-04	3.1.E-05	2.5.E-05	-8.2.E-05	1.0.E-06
18	4.0.E-06	1.0.E-06	2.0.E-06	-1.0.E-06	-1.0.E-06
19	3.5.E-04	-2.8.E-05	-9.6.E-05	-7.9.E-05	-1.0.E-06
20	0.0.E+00	-2.4.E-05	-2.7.E-05	0.0.E+00	2.0.E-06

**<Appendix D. 표- 11> DEL 형상 그룹 별 오차의 평균:
Out of Normal Range 형상 파라미터**

Loop	Max Camber	Max Camber Location	Max Thickness
1	-0.004076	-0.000916	0.025294
2	-0.014036	0.019248	0.012119
3	-0.008748	-0.006683	0.005856
4	0.014881	-0.048151	-0.023072
5	0.001857	-0.005286	-0.008952
6	-0.006369	0.016281	0.006518
7	-0.001294	0.011918	-0.016963
8	-0.049464	0.098888	0.185176
9	0.024443	-0.050527	-0.046189
10	-0.001956	-0.025583	-0.021481
11	0.003526	-0.038112	-0.011806
12	-0.022431	0.032037	-0.017988
13	-0.01934	0.009461	-0.049033
14	-0.000906	0.04028	0.083544
15	0.009185	-0.015213	-0.00969
16	-0.006835	-0.005899	-0.017364
17	-0.00636	-0.025358	0.083571
18	0.003698	-0.009252	0.015912
19	0.019879	0.02397	-0.005815
20	0.009722	-0.0036	-0.02789

<Appendix D. 표- 12> DEL 형상 그룹 별 오차의 평균:
Out of Normal Range 성능 파라미터

Loop	Cl0	Cla0	Cd0	Cm0	Cma0
1	3.29.E-04	-5.55.E-04	-6.70.E-05	-8.90.E-05	1.66.E-04
2	-1.78.E-04	-4.51.E-04	-2.00.E-06	-3.70.E-05	9.80.E-05
3	9.90.E-04	-2.12.E-04	-8.10.E-05	-1.85.E-04	6.80.E-05
4	2.31.E-03	-7.76.E-04	-3.70.E-05	-2.27.E-04	1.87.E-04
5	4.80.E-04	7.19.E-04	6.00.E-06	-5.50.E-05	-1.53.E-04
6	1.27.E-04	4.17.E-04	-1.50.E-05	-1.36.E-04	-9.30.E-05
7	2.30.E-04	8.80.E-05	-9.00.E-06	-1.78.E-04	-1.90.E-05
8	-4.71.E-03	-4.60.E-04	-8.20.E-05	4.32.E-04	1.18.E-04
9	1.50.E-03	-2.48.E-04	7.00.E-06	-2.00.E-05	6.80.E-05
10	-7.35.E-04	4.53.E-04	-3.10.E-05	3.78.E-04	-1.00.E-04
11	1.32.E-04	-4.40.E-05	-3.20.E-05	2.58.E-04	2.20.E-05
12	-1.01.E-03	-2.67.E-04	-1.60.E-05	-3.60.E-05	7.30.E-05
13	-2.00.E-03	-2.29.E-04	-3.00.E-06	3.47.E-04	4.00.E-05
14	1.50.E-03	-2.06.E-04	2.10.E-05	-6.42.E-04	5.40.E-05
15	1.92.E-04	-7.70.E-05	-1.60.E-05	3.80.E-05	8.00.E-06
16	-5.22.E-04	-3.38.E-04	-2.20.E-05	1.59.E-04	7.50.E-05
17	-2.48.E-03	1.72.E-04	-1.40.E-05	7.92.E-04	-4.10.E-05
18	3.05.E-04	-7.90.E-05	0.00.E+00	8.40.E-05	2.00.E-05
19	3.31.E-03	-9.90.E-05	0.00.E+00	-9.66.E-04	2.40.E-05
20	1.40.E-03	-8.00.E-06	1.00.E-05	-2.78.E-04	2.00.E-06

3. Standard Deviation of Errors

<Appendix D. 표- 13> DEL 형상 그룹 별 오차의 평균:
Normal Range 형상 파라미터

Loop	Max Camber	Max Camber Location	Max Thickness
1	0.048676	0.067661	0.478457
2	0.032731	0.047435	0.404285
3	0.039996	0.041452	0.309626
4	0.031259	0.044792	0.330536
5	0.030481	0.031522	0.291311
6	0.028976	0.034550	0.240017
7	0.032505	0.032127	0.252121
8	0.042962	0.062936	0.378510
9	0.028216	0.043862	0.307387
10	0.024119	0.030988	0.287326
11	0.028390	0.039017	0.266834
12	0.025867	0.032080	0.207116
13	0.023894	0.029281	0.248754
14	0.025013	0.025478	0.255855
15	0.020111	0.023521	0.208759
16	0.017268	0.021005	0.184901
17	0.030706	0.030745	0.260515
18	0.015404	0.019449	0.165378
19	0.017719	0.028269	0.175008
20	0.017869	0.017653	0.157820

**<Appendix D. 표- 14> DEL 형상 그룹 별 오차의 평균:
Normal Range 성능 파라미터**

Loop	C_{I0}	C_{la0}	C_{d0}	C_{m0}	C_{ma0}
1	6.19.E-03	7.37.E-04	5.60.E-05	1.69.E-03	1.60.E-04
2	3.80.E-03	7.52.E-04	3.10.E-05	1.14.E-03	1.59.E-04
3	4.30.E-03	7.88.E-04	4.50.E-05	1.06.E-03	1.76.E-04
4	2.86.E-03	7.47.E-04	3.10.E-05	7.34.E-04	1.41.E-04
5	3.18.E-03	5.25.E-04	2.70.E-05	8.29.E-04	1.09.E-04
6	2.77.E-03	5.27.E-04	2.40.E-05	7.75.E-04	1.16.E-04
7	3.36.E-03	4.54.E-04	3.50.E-05	8.35.E-04	1.00.E-04
8	4.30.E-03	8.50.E-04	4.70.E-05	1.25.E-03	1.71.E-04
9	2.32.E-03	7.72.E-04	2.80.E-05	6.02.E-04	1.70.E-04
10	2.15.E-03	4.99.E-04	2.00.E-05	6.19.E-04	1.10.E-04
11	3.02.E-03	6.50.E-04	2.70.E-05	8.20.E-04	1.39.E-04
12	2.75.E-03	5.06.E-04	2.30.E-05	7.76.E-04	1.14.E-04
13	1.98.E-03	4.90.E-04	2.00.E-05	5.54.E-04	1.01.E-04
14	2.95.E-03	6.54.E-04	2.50.E-05	8.01.E-04	1.38.E-04
15	1.89.E-03	4.87.E-04	1.70.E-05	4.86.E-04	1.04.E-04
16	1.43.E-03	4.17.E-04	1.60.E-05	4.10.E-04	9.60.E-05
17	3.29.E-03	5.41.E-04	2.50.E-05	9.09.E-04	1.19.E-04
18	1.44.E-03	4.28.E-04	1.60.E-05	3.81.E-04	9.60.E-05
19	2.04.E-03	4.22.E-04	1.90.E-05	5.11.E-04	9.30.E-05
20	1.74.E-03	4.01.E-04	1.60.E-05	4.56.E-04	9.20.E-05

<Appendix D. 표- 15> DEL 형상 그룹 별 오차의 평균:
Zero Max. Camber 형상 파라미터

Loop	Max Camber	Max Camber Location	Max Thickness
1	0.000000	3.263754	0.016585
2	0.000000	3.076868	0.053979
3	0.000000	2.738524	0.032122
4	0.050455	2.308505	0.077829
5	0.000000	2.271709	0.054880
6	0.000000	1.739382	0.042231
7	0.025957	1.703834	0.114493
8	0.000000	1.577678	0.077840
9	0.000000	1.517566	0.062673
10	0.001987	1.274054	0.061739
11	0.000000	0.910290	0.088500
12	0.000000	0.969559	0.046164
13	0.000000	0.787152	0.049167
14	0.000239	0.729289	0.048506
15	0.003089	0.632809	0.043644
16	0.000403	0.612227	0.044535
17	0.008879	0.461170	0.080004
18	0.000716	0.492895	0.047224
19	0.005135	0.421306	0.045226
20	0.000422	0.434466	0.047364

<Appendix D. 표- 16> DEL 형상 그룹 별 오차의 평균:

Zero Max. Camber 성능 파라미터

Loop	Cl0	Cla0	Cd0	Cm0	Cma0
1	3.90.E-05	9.70.E-05	1.70.E-05	0.00.E+00	3.20.E-05
2	3.60.E-05	1.14.E-04	3.40.E-05	0.00.E+00	3.40.E-05
3	3.50.E-05	2.83.E-04	7.60.E-04	0.00.E+00	4.40.E-05
4	5.96.E-03	4.43.E-04	6.86.E-04	1.42.E-03	3.70.E-05
5	8.10.E-05	1.50.E-04	3.20.E-05	0.00.E+00	3.30.E-05
6	5.80.E-05	1.49.E-04	2.00.E-05	0.00.E+00	2.50.E-05
7	3.78.E-03	3.94.E-04	5.05.E-04	1.01.E-03	3.70.E-05
8	5.70.E-05	2.03.E-04	3.20.E-05	0.00.E+00	4.70.E-05
9	5.70.E-05	2.30.E-04	4.91.E-04	0.00.E+00	3.20.E-05
10	2.92.E-04	1.81.E-04	3.73.E-04	7.40.E-05	3.50.E-05
11	4.50.E-05	1.62.E-04	2.80.E-05	0.00.E+00	3.30.E-05
12	4.10.E-05	2.60.E-04	6.95.E-04	0.00.E+00	3.60.E-05
13	4.50.E-05	2.54.E-04	4.49.E-04	0.00.E+00	3.70.E-05
14	6.10.E-05	2.26.E-04	4.93.E-04	1.10.E-05	3.20.E-05
15	3.97.E-04	1.97.E-04	5.29.E-04	9.80.E-05	3.00.E-05
16	6.10.E-05	1.77.E-04	2.33.E-04	1.20.E-05	3.20.E-05
17	1.12.E-03	1.88.E-04	3.49.E-04	2.79.E-04	3.60.E-05
18	1.08.E-04	1.75.E-04	2.42.E-04	2.10.E-05	3.60.E-05
19	6.52.E-04	2.97.E-04	7.93.E-04	1.53.E-04	4.10.E-05
20	7.30.E-05	2.31.E-04	4.70.E-04	1.30.E-05	3.40.E-05

**<Appendix D. 표- 17> DEL 형상 그룹 별 오차의 평균:
Out of Normal Range 형상 파라미터**

Loop	Max Camber	Max Camber Location	Max Thickness
1	0.149846	0.128161	0.815404
2	0.082539	0.086811	0.464549
3	0.098154	0.109584	0.460809
4	0.065548	0.078928	0.405663
5	0.076547	0.060550	0.351172
6	0.098349	0.076103	0.454525
7	0.061832	0.090070	0.503434
8	0.070869	0.088881	0.442459
9	0.077603	0.072899	0.305571
10	0.053862	0.050129	0.292140
11	0.054963	0.067548	0.415427
12	0.071623	0.057875	0.251928
13	0.070789	0.045955	0.258688
14	0.056688	0.056864	0.376173
15	0.044108	0.052348	0.315815
16	0.046891	0.043961	0.437533
17	0.045559	0.053002	0.361810
18	0.035218	0.033618	0.262222
19	0.029306	0.036366	0.227301
20	0.026122	0.027221	0.204999

<Appendix D. 표- 18> DEL 형상 그룹 별 오차의 평균:
Out of Normal Range 성능 파라미터

Loop	Cl0	Cla0	Cd0	Cm0	Cma0
1	2.16.E-02	1.31.E-02	1.12.E-03	5.08.E-03	3.13.E-03
2	1.82.E-02	1.69.E-02	1.02.E-03	3.71.E-03	3.68.E-03
3	1.97.E-02	1.18.E-02	8.40.E-04	5.17.E-03	2.61.E-03
4	1.31.E-02	9.25.E-03	4.16.E-04	3.27.E-03	2.13.E-03
5	1.10.E-02	1.46.E-02	4.35.E-04	2.56.E-03	3.32.E-03
6	9.12.E-03	1.41.E-02	6.42.E-04	2.45.E-03	3.20.E-03
7	1.12.E-02	1.37.E-02	7.67.E-04	2.62.E-03	3.10.E-03
8	4.74.E-02	1.14.E-02	1.15.E-03	1.01.E-02	2.55.E-03
9	8.33.E-03	1.25.E-02	5.10.E-04	2.05.E-03	2.77.E-03
10	7.82.E-03	1.39.E-02	4.87.E-04	1.84.E-03	3.17.E-03
11	9.27.E-03	1.45.E-02	5.59.E-04	2.36.E-03	3.29.E-03
12	1.25.E-02	1.25.E-02	6.16.E-04	3.04.E-03	2.80.E-03
13	1.11.E-02	1.18.E-02	4.37.E-04	2.71.E-03	2.65.E-03
14	7.91.E-03	1.23.E-02	4.29.E-04	1.76.E-03	2.79.E-03
15	1.21.E-02	1.12.E-02	3.87.E-04	2.77.E-03	2.47.E-03
16	9.09.E-03	1.27.E-02	4.56.E-04	2.08.E-03	2.80.E-03
17	6.85.E-03	1.28.E-02	4.25.E-04	1.87.E-03	2.83.E-03
18	5.17.E-03	1.06.E-02	4.19.E-04	1.37.E-03	2.36.E-03
19	4.93.E-03	1.05.E-02	4.07.E-04	1.20.E-03	2.33.E-03
20	4.63.E-03	1.14.E-02	3.84.E-04	1.12.E-03	2.54.E-03

Appendix E. 최종 하이퍼 파라미터 최적화(FHO):

평균과 표준편차

1. Epoch Test

- 형상그룹의 분류는 139p 의 <표 3-54> 형상 범위 설정을 참조

<Appendix E. 표- 1> FHO Epoch Test: Epoch Test Cases

Epoch	Case 1 (Baseline)	Case 2	Case 3	Case 4
	20,000 (20k)	50,000 (50k)	100,000 (100k)	200,000 (200k)

<Appendix E. 표- 2> FHO Epoch Test: 형상 오차의 평균

Shape Group	Case	Max. Camber	Max. Camber Location	Max Thickness
Total	1	0.009231	-0.002570	-0.022929
	2	0.000875	-0.006285	0.005789
	3	-0.003500	0.003440	0.026724
	4	0.005954	0.001520	0.015503
Normal Range	1	0.013791	-0.008943	-0.012170
	2	0.002742	-0.014701	-0.027872
	3	-0.002366	0.001467	0.045796
	4	0.004006	-0.003405	0.006338
Zero Max. Camber	1	0.000015	0.012899	0.001608
	2	0.000006	0.009345	-0.012839
	3	0.000028	0.029460	0.024276
	4	0.000003	0.009128	0.012914
Out of Normal Range	1	0.009722	-0.0036	-0.02789
	2	0.000689	-0.00696	0.013579
	3	-0.004149	0.000305	0.024005
	4	0.007053	0.001298	0.01731

<Appendix E. 표- 3> FHO Epoch Test: 성능 오차의 평균

Shape Group	Case	Cl0	Cla0	Cd0	Cm0	Cma0
Total	1	1.24.E-03	-9.00.E-06	6.00.E-06	-2.38.E-04	2.00.E-06
	2	-1.83.E-04	3.90.E-05	1.00.E-06	1.50.E-04	-5.00.E-06
	3	-2.88.E-04	2.70.E-05	-1.00.E-06	8.80.E-05	0.00.E+00
	4	6.66.E-04	-8.30.E-05	5.00.E-06	-1.85.E-04	1.70.E-05
Normal Range	1	1.25.E-03	-1.00.E-06	8.00.E-06	-1.85.E-04	0.00.E+00
	2	-3.12.E-04	-7.00.E-06	6.00.E-06	2.14.E-04	2.00.E-06
	3	-1.40.E-04	-9.00.E-06	3.00.E-06	6.10.E-05	7.00.E-06
	4	4.69.E-04	8.00.E-06	-3.00.E-06	-1.05.E-04	0.00.E+00
Zero Max. Camber	1	0.0.E+00	-2.4.E-05	-2.7.E-05	0.0.E+00	2.0.E-06
	2	0.0.E+00	1.2.E-05	-1.0.E-06	0.0.E+00	1.0.E-06
	3	3.0.E-06	1.0.E-05	-7.0.E-06	-1.0.E-06	1.0.E-06
	4	2.0.E-06	-2.3.E-05	-1.8.E-05	0.0.E+00	4.0.E-06
Out of Normal Range	1	1.40.E-03	-8.00.E-06	1.00.E-05	-2.78.E-04	2.00.E-06
	2	-1.86.E-04	5.00.E-05	0.00.E+00	1.59.E-04	-7.00.E-06
	3	-3.50.E-04	3.60.E-05	-1.00.E-06	1.04.E-04	-1.00.E-06
	4	7.86.E-04	-1.05.E-04	1.00.E-05	-2.23.E-04	2.20.E-05

<Appendix E. 표- 4> FHO Epoch Test: 형상 오차의 표준편차

Shape Group	Case	Max. Camber	Max. Camber Location	Max Thickness
Total	1	0.024065	0.141308	0.189559
	2	0.022004	0.144330	0.216631
	3	0.029426	0.143000	0.215953
	4	0.025908	0.133563	0.209360

Normal Range	1	0.017869	0.017653	0.157820
	2	0.014430	0.016972	0.155862
	3	0.013877	0.016283	0.148845
	4	0.015293	0.017943	0.155351
Zero Max. Camber	1	0.000422	0.434466	0.047364
	2	0.000141	0.446855	0.045487
	3	0.000392	0.436272	0.042636
	4	0.000092	0.409927	0.040949
Out of Normal Range	1	0.026122	0.027221	0.204999
	2	0.024294	0.030785	0.236773
	3	0.032952	0.032436	0.237521
	4	0.028697	0.026893	0.229231

<Appendix E. 표- 5> FHO Epoch Test: 성능 오차의 평균

Shape Group	Case	Cl0	Cla0	Cd0	Cm0	Cma0
Total	1	4.14.E-03	1.00.E-02	3.70.E-04	1.00.E-03	2.24.E-03
	2	5.10.E-03	9.72.E-03	4.21.E-04	1.14.E-03	2.17.E-03
	3	5.25.E-03	1.00.E-02	4.34.E-04	1.26.E-03	2.22.E-03
	4	5.09.E-03	9.95.E-03	4.29.E-04	1.22.E-03	2.21.E-03
Normal Range	1	1.74.E-03	4.01.E-04	1.60.E-05	4.56.E-04	9.20.E-05
	2	1.35.E-03	4.04.E-04	1.60.E-05	3.85.E-04	9.30.E-05
	3	1.28.E-03	3.67.E-04	1.22.E-04	3.45.E-04	8.40.E-05
	4	1.61.E-03	4.03.E-04	1.75.E-04	4.43.E-04	9.10.E-05
Zero Max. Camber	1	7.3.E-05	2.3.E-04	4.7.E-04	1.3.E-05	3.4.E-05
	2	5.1.E-05	1.5.E-04	2.1.E-04	8.0.E-06	3.1.E-05
	3	6.0.E-05	1.6.E-04	2.6.E-04	1.2.E-05	3.2.E-05
	4	4.9.E-05	1.9.E-04	3.9.E-04	5.0.E-06	3.6.E-05

Out of Normal Range	1	4.63.E-03	1.14.E-02	3.84.E-04	1.12.E-03	2.54.E-03
	2	5.76.E-03	1.10.E-02	4.72.E-04	1.28.E-03	2.46.E-03
	3	5.95.E-03	1.14.E-02	4.81.E-04	1.42.E-03	2.52.E-03
	4	5.75.E-03	1.13.E-02	4.61.E-04	1.38.E-03	2.51.E-03

2. Number of Hidden Layers Test

- 형상그룹의 분류는 139p 의 <표 3-54> 형상 범위 설정을 참조

<Appendix E. 표- 6> FHO Hidden Layer Test: Test Cases

Epoch	Case 1	Case 2 (Baseline)	Case 3	Case 4
	5	6	7	8

<Appendix E. 표- 7> FHO Hidden Layer Test: 형상 오차의 평균

Shape Group	Case	Max. Camber	Max. Camber Location	Max Thickness
Total	1	0.008773	-0.012298	-0.039243
	2	0.009231	-0.002570	-0.022929
	3	0.010617	0.007926	-0.053277
	4	0.019230	-0.020741	0.042813
Normal Range	1	0.008026	-0.020834	-0.087052
	2	0.013791	-0.008943	-0.012170
	3	0.020555	0.005184	-0.062337
	4	0.016633	-0.020480	0.010503
Zero Max. Camber	1	0.000113	0.003278	-0.030094
	2	0.000015	0.012899	0.001608
	3	0.000004	0.023534	-0.039852
	4	0.000430	-0.009859	0.019182

Out of Normal Range	1	0.010046	-0.012998	-0.032766
	2	0.009722	-0.0036	-0.02789
	3	0.010411	0.006324	-0.053579
	4	0.022139	-0.022226	0.051128

<Appendix E. 표- 8> FHO Hidden Layer Test: 성능 오차의 평균

Shape Group	Case	Cl0	Cla0	Cd0	Cm0	Cma0
Total	1	7.67.E-04	-4.30.E-05	0.00.E+00	-9.80.E-05	7.00.E-06
	2	1.24.E-03	-9.00.E-06	6.00.E-06	-2.38.E-04	2.00.E-06
	3	1.61.E-03	-1.47.E-04	-8.00.E-06	-4.42.E-04	2.40.E-05
	4	1.79.E-03	-1.24.E-04	0.00.E+00	-2.06.E-04	3.50.E-05
Normal Range	1	-1.12.E-04	-1.40.E-05	1.00.E-06	1.65.E-04	-1.00.E-06
	2	1.25.E-03	-1.00.E-06	8.00.E-06	-1.85.E-04	0.00.E+00
	3	2.42.E-03	-1.70.E-05	-4.00.E-06	-6.21.E-04	0.00.E+00
	4	1.15.E-03	-3.00.E-06	1.80.E-05	-1.30.E-05	3.00.E-06
Zero Max. Camber	1	1.4.E-05	-3.3.E-05	-6.0.E-06	-3.0.E-06	2.0.E-06
	2	0.0.E+00	-2.4.E-05	-2.7.E-05	0.0.E+00	2.0.E-06
	3	2.0.E-06	-4.7.E-05	-1.8.E-05	0.0.E+00	0.0.E+00
	4	6.1.E-05	3.0.E-06	-8.0.E-06	-1.4.E-05	2.0.E-06
Out of Normal Range	1	1.01.E-03	-4.90.E-05	1.00.E-06	-1.52.E-04	1.00.E-05
	2	1.40.E-03	-8.00.E-06	1.00.E-05	-2.78.E-04	2.00.E-06
	3	1.69.E-03	-1.81.E-04	-7.00.E-06	-4.71.E-04	3.10.E-05
	4	2.12.E-03	-1.61.E-04	-1.00.E-06	-2.63.E-04	4.40.E-05

<Appendix E. 표- 9> FHO Hidden Layer Test: 형상 오차의 표준편차

Shape Group	Case	Max. Camber	Max. Camber Location	Max Thickness
Total	1	0.022292	0.146572	0.208535
	2	0.024065	0.141308	0.189559
	3	0.029100	0.145028	0.209622
	4	0.025807	0.147062	0.220806
Normal Range	1	0.015887	0.018087	0.159531
	2	0.017869	0.017653	0.157820
	3	0.018308	0.020157	0.161375
	4	0.015438	0.019511	0.153629
Zero Max. Camber	1	0.000950	0.448590	0.043141
	2	0.000422	0.434466	0.047364
	3	0.000109	0.445802	0.051426
	4	0.001755	0.448795	0.044394
Out of Normal Range	1	0.024303	0.030136	0.226963
	2	0.026122	0.027221	0.204999
	3	0.031766	0.032651	0.228424
	4	0.027678	0.034341	0.242244

<Appendix E. 표- 10> FHO Hidden Layer Test: 성능 오차의 표준편차

Shape Group	Case	Cl0	Cla0	Cd0	Cm0	Cma0
Total	1	4.10.E-03	1.01.E-02	3.84.E-04	9.95.E-04	2.26.E-03
	2	4.14.E-03	1.00.E-02	3.70.E-04	1.00.E-03	2.24.E-03
	3	4.40.E-03	1.03.E-02	3.89.E-04	1.10.E-03	2.29.E-03
	4	4.64.E-03	9.34.E-03	4.17.E-04	1.11.E-03	2.10.E-03

Normal Range	1	1.48.E-03	4.39.E-04	1.27.E-04	3.99.E-04	9.80.E-05
	2	1.74.E-03	4.01.E-04	1.60.E-05	4.56.E-04	9.20.E-05
	3	1.76.E-03	3.95.E-04	1.50.E-05	4.67.E-04	9.10.E-05
	4	1.61.E-03	4.60.E-04	1.60.E-05	4.51.E-04	1.03.E-04
Zero Max. Camber	1	1.2.E-04	2.2.E-04	2.2.E-04	3.2.E-05	3.4.E-05
	2	7.3.E-05	2.3.E-04	4.7.E-04	1.3.E-05	3.4.E-05
	3	4.8.E-05	2.0.E-04	3.5.E-04	6.0.E-06	3.1.E-05
	4	2.4.E-04	1.7.E-04	2.6.E-04	5.7.E-05	3.5.E-05
Out of Normal Range	1	4.60.E-03	1.15.E-02	4.26.E-04	1.11.E-03	2.57.E-03
	2	4.63.E-03	1.14.E-02	3.84.E-04	1.12.E-03	2.54.E-03
	3	4.91.E-03	1.17.E-02	4.23.E-04	1.22.E-03	2.60.E-03
	4	5.19.E-03	1.06.E-02	4.65.E-04	1.25.E-03	2.39.E-03

3. Number of Neurons per Hidden Layer Test

- 형상그룹의 분류는 139p 의 <표 3-54> 형상 범위 설정을 참조

<Appendix E. 표- 11> FHO Hidden Layer Test: Test Cases

Epoch	Case 1	Case 2 (Baseline)	Case 3	Case 4
	300	500	800	1,000

<Appendix E. 표- 12> FHO Neuron Test: 형상 오차의 평균

Shape Group	Case	Max. Camber	Max. Camber Location	Max Thickness
Total	1	0.000927	0.005101	-0.028953
	2	0.009231	-0.002570	-0.022929
	3	0.006184	0.003162	-0.040459
	4	-0.001123	-0.009107	0.012175

Normal Range	1	-0.001341	-0.003840	-0.034342
	2	0.013791	-0.008943	-0.012170
	3	0.011314	-0.002202	-0.087329
	4	-0.005612	-0.017601	0.001109
Zero Max. Camber	1	0.000167	0.021718	-0.023315
	2	0.000015	0.012899	0.001608
	3	0.000000	0.012109	-0.058590
	4	0.000017	0.001078	0.001367
Out of Normal Range	1	0.001392	0.004313	-0.028843
	2	0.009722	-0.0036	-0.02789
	3	0.006177	0.002842	-0.030549
	4	-0.000559	-0.009095	0.015352

<Appendix E. 표- 13> FHO Neuron Test: 성능 오차의 평균

Shape Group	Case	C10	Cla0	Cd0	Cm0	Cma0
Total	1	3.69.E-04	-7.80.E-05	-5.00.E-06	-1.20.E-04	1.80.E-05
	2	1.24.E-03	-9.00.E-06	6.00.E-06	-2.38.E-04	2.00.E-06
	3	6.26.E-04	3.80.E-05	2.00.E-06	-2.00.E-04	-1.00.E-05
	4	-1.57.E-04	-1.00.E-06	-3.00.E-06	1.49.E-04	4.00.E-06
Normal Range	1	-5.20.E-04	-2.00.E-06	0.00.E+00	1.47.E-04	0.00.E+00
	2	1.25.E-03	-1.00.E-06	8.00.E-06	-1.85.E-04	0.00.E+00
	3	1.10.E-03	-5.00.E-06	-1.10.E-05	-3.10.E-04	-6.00.E-06
	4	-1.48.E-03	8.00.E-06	1.00.E-05	5.27.E-04	4.00.E-06
Zero Max. Camber	1	2.1.E-05	-5.0.E-06	-8.0.E-06	-4.0.E-06	0.0.E+00
	2	0.0.E+00	-2.4.E-05	-2.7.E-05	0.0.E+00	2.0.E-06
	3	-1.0.E-06	-3.6.E-05	-1.0.E-05	0.0.E+00	0.0.E+00
	4	1.0.E-06	1.3.E-05	-3.0.E-06	0.0.E+00	1.0.E-06

Out of Normal Range	1	5.57.E-04	-1.00.E-04	-6.00.E-06	-1.78.E-04	2.30.E-05
	2	1.40.E-03	-8.00.E-06	1.00.E-05	-2.78.E-04	2.00.E-06
	3	6.33.E-04	5.40.E-05	5.00.E-06	-2.09.E-04	-1.20.E-05
	4	3.20.E-05	-5.00.E-06	-5.00.E-06	1.09.E-04	4.00.E-06

<Appendix E. 표- 14> FHO Neuron Test: 형상 오차의 표준편차

Shape Group	Case	Max. Camber	Max. Camber Location	Max Thickness
Total	1	0.024439	0.144808	0.206026
	2	0.024065	0.141308	0.189559
	3	0.022553	0.143254	0.211278
	4	0.023451	0.146526	0.221123
Normal Range	1	0.016527	0.020717	0.153678
	2	0.017869	0.017653	0.157820
	3	0.016062	0.018070	0.156828
	4	0.013987	0.016599	0.157219
Zero Max. Camber	1	0.001214	0.441780	0.044239
	2	0.000422	0.434466	0.047364
	3	0.000022	0.441147	0.047739
	4	0.000276	0.450286	0.047604
Out of Normal Range	1	0.026976	0.030463	0.22554
	2	0.026122	0.027221	0.204999
	3	0.024631	0.028071	0.230099
	4	0.025975	0.031774	0.242516

<Appendix E. 표- 15> FHO Neuron Test: 성능 오차의 표준편차

Shape Group	Case	Cl0	Cla0	Cd0	Cm0	Cma0
Total	1	3.94.E-03	9.34.E-03	3.71.E-04	9.56.E-04	2.10.E-03
	2	4.14.E-03	1.00.E-02	3.70.E-04	1.00.E-03	2.24.E-03
	3	3.65.E-03	1.01.E-02	3.62.E-04	8.65.E-04	2.23.E-03
	4	3.65.E-03	9.94.E-03	3.56.E-04	9.26.E-04	2.21.E-03
Normal Range	1	1.74.E-03	4.21.E-04	1.60.E-05	4.68.E-04	9.40.E-05
	2	1.74.E-03	4.01.E-04	1.60.E-05	4.56.E-04	9.20.E-05
	3	1.49.E-03	3.94.E-04	1.27.E-04	4.19.E-04	9.10.E-05
	4	1.36.E-03	4.68.E-04	1.60.E-05	4.00.E-04	1.03.E-04
Zero Max. Camber	1	1.5.E-04	1.6.E-04	2.7.E-04	3.3.E-05	3.3.E-05
	2	7.3.E-05	2.3.E-04	4.7.E-04	1.3.E-05	3.4.E-05
	3	4.3.E-05	1.6.E-04	1.9.E-04	4.0.E-06	3.2.E-05
	4	5.3.E-05	1.5.E-04	2.1.E-04	8.0.E-06	3.4.E-05
Out of Normal Range	1	4.41.E-03	1.06.E-02	4.11.E-04	1.06.E-03	2.39.E-03
	2	4.63.E-03	1.14.E-02	3.84.E-04	1.12.E-03	2.54.E-03
	3	4.10.E-03	1.14.E-02	4.02.E-04	9.65.E-04	2.54.E-03
	4	4.07.E-03	1.13.E-02	3.97.E-04	1.03.E-03	2.51.E-03

국문초록

심층 학습 기반 공학 설계

NACA 에어포일 설계 심층신경망 학습

심층 학습, 딥러닝(Deep Learning)은 기존 인공신경망(Artificial Neural Network, ANN)의 한계였던 Gradient vanishing, Overfitting 등의 문제들을 극복한, 대규모의 인공신경망을 말한다.

딥러닝은 Computer Visioning(CV) 분야에서 이전의 Support Vector Machine(SVM, 서포트 벡터 머신)이나 K-NN(K-Nearest Neighbor, K-최근접 이웃)과 같은 다른 기계학습 알고리즘이 넘지 못한 이미지 분류 점수를 획득하면서 각광을 받기 시작했다. 최근에는 거의 모든 이미지 분류 알고리즘이 딥러닝을 사용하며, 이를 바탕으로 이미지 인식 및 처리, 분류, 그리고 자율 주행과 같은 다양한 분야에서 사용되고 있다.

기계학습은 데이터 사이의 관계를 나타내기 위해 사용되는데, 딥러닝은 다른 기계학습 기법들 중에서도 매우 탁월한 성능을 보인다. 본 연구는 딥러닝이 복잡한 데이터 사이의 관계를 탁월하게 나타낸다는 점을 이용하여 공학적 설계를 하도록 하였다. 공학적 설계란 주어진 요구 성능을 만족하는 기계적 장치를 어떻게 만들면 되는지 그 과정을 말한다. 현재 대부분의 설계 기법들과 도구들은 설계 과정에서 성능 해석 도구들을 이용하여 반복적으로 요구도와의 오차를 확인하고 줄여 나가는 과정을 따른다. 해석과 성능오차 확인, 그리고 형상의 수정을 반복할 수밖에 없는 이유는 우리가 가진 방정식들이 자연법칙에 근거하며, 형상에 대해 그 성능을 측정하는 기법들이기 때문이다. 만약, 성능 지표들을 이용하여 형상을 도출해낼 수 있는 식이 만들어진다면, 설계에서 반복적인 과정은 줄어들고, 가속시킬 수 있을 것이다.

본 연구에서는 다양한 설계 예시 중 에어포일(Airfoil) 설계를 예시로 사용하였다. 에어포일은 바람을 이용하여 양력(Lift force)를 만들어내는 기계적 장치이며, 여객기나 헬리콥터와 같은 항공기가 비행할 수 있도록 하는 핵심적인 장치이다. 그러므로 에어포일 설계는 항공기 설계중의 핵심이며, 기초라고 할 수 있다. 항공기의 설계는 요구조건의 분석과, 이를 바탕으로 밑그림을 그리는 개념설계로부터 시작한다. 개념설계는 최신 트렌드, 주어진 요구조건을 만족하는 동체, 날개, 추진시스템과 같이 큰 부분에 대한 설계로부터 출발하며, 항공기의 양력을 발생하는 날개의 설계는 양력, 항력, 모멘트 등의 요구 성능을 만족하도록 형태, 너비, 시위길이, 에어포일 종류 등을 결정한다. 날개의 다양한 형상 요소들 중, 에어포일은 양력을 발생시키기 위해 고안된 장치이므로 가장 큰 영향력을 갖는다. 그러므로 에어포일 설계는 항공공학을 배우는 학생들이 가장 먼저 배우는 항공기 설계의 기본적 요소이며, 중요한 부분이라 할 수 있다.

에어포일을 설계하는 방법들은 다양한데, 가장 기본적이며 간단한 방법은 데이터베이스에서 요구조건과 유사한 에어포일을 검색하는 것이다. 검색은 간단하게 사용할 수 있으나, 요구조건을 정확하게 만족하는 에어포일을 찾아내기 어려우며, 때로는 찾아낼 수 없을 수도 있다.

전문적인 영역에서는 검색과 더불어, 형상을 요구조건에 맞추도록 하는 최적화 기법을 사용한다. 최적화 기법은 요구조건과 유사한 성능을 내는 기본 형상(Baseline)을 선정한 후, 조금씩 형상을 변화시켜가면서 성능을 요구도에 맞추어 나가는 방법이다. 최적화 기법은 검색보다 정확한 결과를 도출할 수 있지만, 최적화를 위해서는 사용자가 최적화 대상을 표현하는 목적 함수, 요구조건을 이용한 제약함수를 만들어내야 하며, 적합한 최적 기법을 선정할 수 있어야 한다. 또한, 최적화 과정은 한 번에 이루어지는 것이 아니라 조금씩 형상을 변경해가며 요구도를 만족시키도록 하므로, 매번 요구도가 만족되는지 확인을 하기 위해 성능해석을 실시해야 한다. 만약 100번만에 최적 형상을 찾았다면, 100번의 성능 해석을 실시한 것이

다. 이는 설계에 시간과 자원이 소모되게 하는 요인들 중 하나이다.

다른 전문적인 방법으로는 성능 파라미터를 바탕으로 형상을 찾아내는 역설계 기법(Airfoil Inverse Design)¹⁰ 있다. 역설계 기법도 요구 성능을 만족하도록 기본 형상의 성능과의 오차를 줄여 나가는 기법이다. 최적 설계와 다른 점은, 최적 설계는 제약 조건 내에서 어떤 목표 성능을 최대화하거나 최소화하지만, 역설계 기법은 명료하게 제시된 요구 성능에 근접하도록 한다는 점에 있다.

본 연구에서도 역설계 기법과 마찬가지로 명료한 성능 지표를 이용하여 이와 매우 근접한 형상을 도출하도록 하지만 두 가지 다른 점이 존재한다. 역설계 기법은 요구 성능으로 사용자가 압력 분포를 그려야 한다. 여기서 역설계 기법의 단점이 드러나는데, 바로 설계자가 사용하기에 직관적이지 못하다는 것이다. 항공기 설계에서 요구되는 성능은 주로 힘과 모멘트로 이루어진 파라미터들이기 때문에 이를 다시 압력 분포도를 그릴 줄 알아야 한다. 비록 성능 해석을 통해 얻어내는 힘과 모멘트가 압력분포를 계산하여 얻어내는 것이라 할지라도, 이를 역으로 적용하여 압력분포를 그려내는 것은 경험이 매우 많아도 어려운 일이다. 다른 단점은 오차를 확인하고 이를 줄여 나가는 과정이 반복적이기 때문에 해석에 오랜 시간이 소요된다는 점이다.

본 연구에서는 최적 설계 기법보다 명료하며, 역설계 기법보다 직관적인 성능 파라미터들을 바탕으로, 두 가지 기법들보다 매우 빠르게 형상을 도출해내기 위해 인공신경망과 딥러닝 기법을 도입하였다. 설계 과정이 오래 걸리는 가장 큰 이유는 성능에 기반하여 형상을 도출해낼 수 있는 명확한 관계 식이 없기 때문인데, 인공신경망을 사용하면 사람이 둘 사이의 관계를 찾아낼 필요 없이 인공신경망 스스로 성능과 형상 데이터 사이에서 그 관계를 찾아낼 것이다.

그러나 인공신경망의 학습은 방대한 양의 데이터를 기반하는데, 설계 문제에서 데이터 사이언스 분야에서 요구하는 수준의 데이터를 얻기는 쉽지

않다. 이는 그동안 설계 기법이 최소한의 데이터를 바탕으로 문제를 해결 할 수 있도록 진화해온 이유이자 원인이다. 본 연구에서는 설계 영역에서 데이터 부족 문제를 해결하기 위해 소규모 데이터를 통한 학습으로 학습 데이터를 강화하는 방법을 개발하였으며, 이에 대해 논하였다. 또한 인공 신경망을 동일한 데이터를 이용하여 여러 번 학습하여 학습 양상을 살펴 보고, 다양한 하이퍼 파라미터에 대하여 최적화를 진행하였다.

설계 예시 형상은 개의 파라미터로 쉽게 형상을 생성할 수 있는 NACA 4-digit 에어포일을 선정하였다. 성능 데이터를 획득하기 위한 해석 도구로는 XFOIL [5]을 사용하여, 인공신경망 학습을 위한 저속 유동 영역의 에어포일 데이터베이스를 구축하였다. 그리고 해당 데이터베이스를 75%, 25%로 나누어 각각 인공신경망 학습과 검증에 사용하였다.

데이터의 증가는 인공신경망이 NACA 4-digit의 대칭형 형상과 비대칭형 형상 사이의 불연속적인 관계를 잘 나타낼 수 있도록 할 뿐만 아니라, 학습 정도를 테스트에서 예측 오차의 표준편차를 크게 줄이는 효과를 낳았다. 이후, 인공신경망이 예측 성능을 충분히 낼 정도로 데이터를 획득한 후에는 하이퍼 파라미터 최적화를 실시하여 인공신경망의 예측 평균이 0에 가까워지도록 하였다.

본 논문 마지막에는 인공신경망의 예측 결과를 재해석하여 입력한 성능 파라미와 예측한 형상의 성능 사이의 오차를 측정하고 가장 높은 1%, 5%, 10%, 15%, 20% 지점에서의 Percentile rank 값을 구하고 비교하였다. 데이터 베이스 강화 루프는 Percentile rank 오차를 최소 50%, 최대 98%까지 줄였으며, 하이퍼 파라미터 최적화는 오차를 최대 40% 줄였다. 데이터베이스 강화 루프 이후, 하이퍼 파라미터 최적화 실시한 결과는 데이터베이스 강화 이전보다 오차를 50~98%까지 줄인 것을 확인할 수 있었다.

주제어: 딥러닝, 심층신경망, 인공신경망, 에어포일 설계, 하이퍼 파라미터 최적화, NACA 4-digit