최종 발표

7조

김희균, 송준규, 정승연, 정하연

목차

1. Goal of the Project

"Hand-made dataset의 성능이 저하된 원인을 분석" + "개선된 ML model을 학습 및 최적화"

+

“ 높은 점수보다 Overfitting을 막는 최대한 내성이 강한 모델을 만들자”

(노이즈, 테두리, shifted)

2. 데이터 분석 및 전처리

- Discover and Visualize the data

(1) 15개(0~9, +, -, /, x, =) 클래스 이외의 데이터 → **제거**

	숫자	기호
Training	(15119, 28, 28)	(15329, 28, 28)
Test	(2160, 28, 28)	(2190, 28, 28)

〈Data Cleaning 이전 dataset의 shape〉

	숫자 + 기호
Training	(30448, 28, 28)
Test	(4350, 28, 28)

〈15개 클래스 분류기를 위한 데이터 통합〉

최종 데이터셋 shape	
Training	(27389, 784)
Test	(4003, 784)

〈가공할 Handmade 데이터셋〉

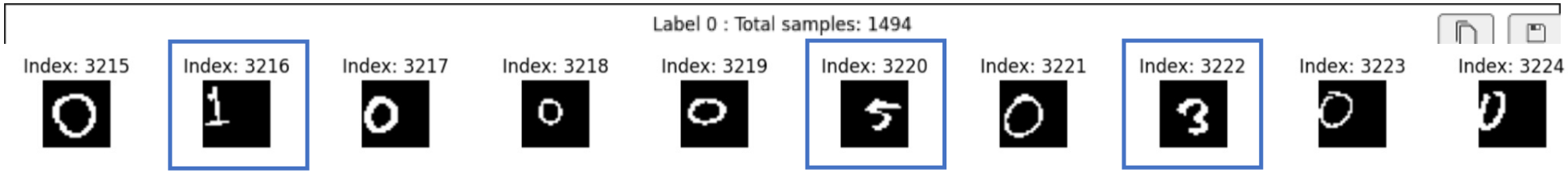
제거된 데이터셋 개수	
Training	3059 (10.0%)
Test	347 (7.9%)

〈15개 이외 라벨 제거〉

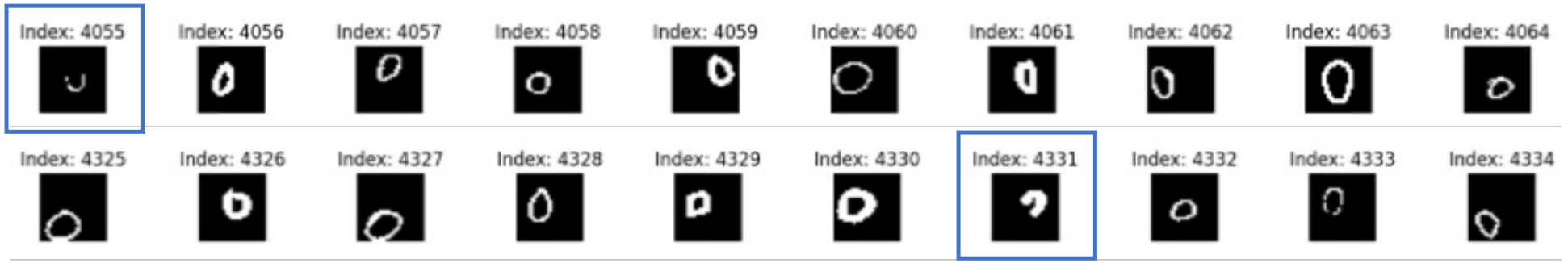
2. 데이터 분석 및 전처리

- Discover and Visualize the data

(2) 잘못된 라벨링 → **육안으로 판별 후 제거**



(3) 학습에 애매한 데이터 → **육안으로 판별 후 제거** (2, 3번 과정에서 1910(train)+209(test)개 삭제됨)




// 라벨 'x'


// 라벨 '9'

2. 데이터 분석 및 전처리

- Discover and Visualize the data

(4) Handmade dataset 데이터 수 : 약 30000개 → **Combined Dataset 구성**

: (Class10: 총 60,000개, trainset : 라벨별 각 6000개, testset : 각 1000개)

: (Class15: 총 60,000개, trainset : 라벨별 각 4000개, testset : 각 660개)

(5) 픽셀값(feature 값)의 차이 → **파이프라인에 Normalizer( ) 추가**

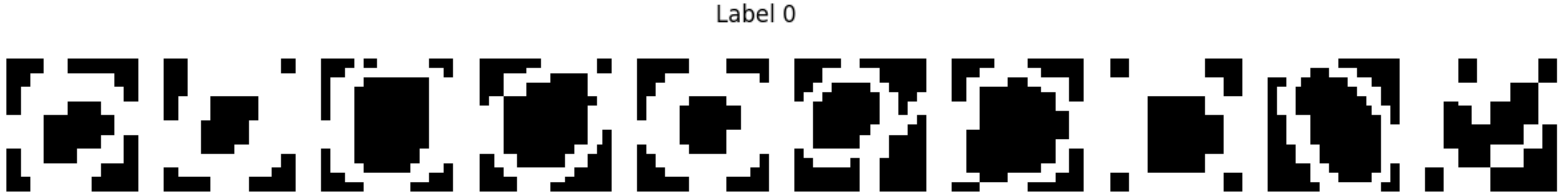
- Original Dataset : 0~255 // Handmade Dataset : 0~1

(6) 노이즈가 있는 데이터 → **노이즈 추가하여 trainset, testset 구축 + 파이프라인에 denoising 함수 추가**

2. 데이터 분석 및 전처리

- Discover and Visualize the data

(7) shift된 데이터, 테두리가 남은 데이터 → **파이프라인에 중앙화, 최대화 과정 추가**



(8) 비슷한 숫자 및 기호의 존재 → **삭제, 향후 Confusion Matrix 분석 계획**

(9) 데이터 크기 : 28*28 데이터만 있음 → **현재 가진 데이터를 확대, 축소해서 새로운 데이터를 만드는 데 한계가 있음**

2. 데이터 분석 및 전처리

- Pipeline: 0 개요

1. **Normalize:** Feature 값 상이
2. **Denosing:** Noise 존재할 경우 없애기
3. **Centering:** 중앙에 위치하도록 수정하기
4. **Enlargning:** 작은 데이터 크게 만들기

2. 데이터 분석 및 전처리

- Pipeline: **1. Normalize**

KNN: 0.83 // 0.88

```
--- No scaling ---
Training time: 0.0281 seconds
Testing time: 0.8710 seconds
Accuracy: 0.8394
----------------------------------------
--- StandardScaler ---
Training time: 0.2734 seconds
Testing time: 1.0164 seconds
Accuracy: 0.8026
----------------------------------------
--- RobustScaler ---
Training time: 0.5483 seconds
Testing time: 0.9632 seconds
Accuracy: 0.8394
----------------------------------------
--- MinMaxScaler ---
Training time: 0.1712 seconds
Testing time: 0.9132 seconds
Accuracy: 0.8394
----------------------------------------
--- Normalizer ---
Training time: 0.1604 seconds
Testing time: 1.0511 seconds
Accuracy: 0.8878
----------------------------------------
```

1. No Scaler
2. Standard Scaler
3. Robust Scaler
4. MinMax Scaler
5. Normalizer

대부분 모델:
Normalizer 사용시 성능 향상 최대
(소수점 두자리 값 차이 정도)

MLP: 0.80 // 0.84

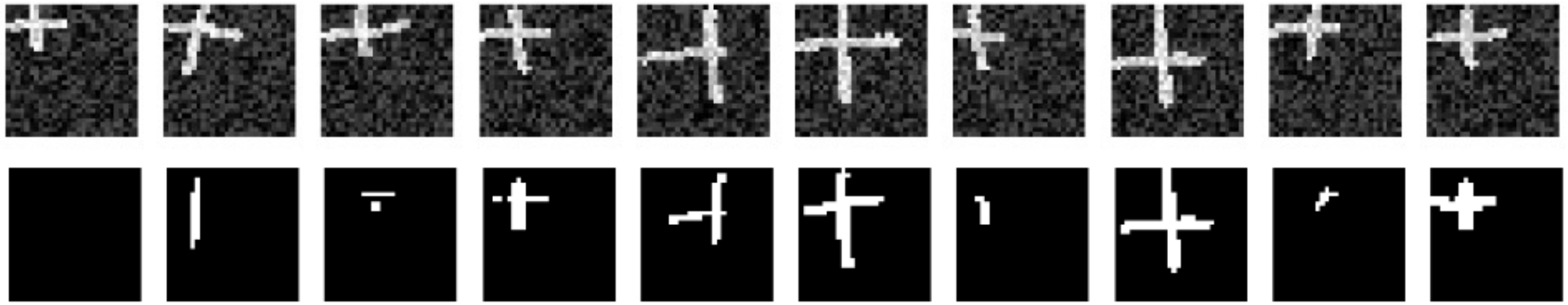
```
--- StandardScaler ---
Training time: 52.8641 // Accuracy: 0.822828
--- RobustScaler ---
Training time: 74.4527 // Accuracy: 0.845253
--- MinMaxScaler ---
Training time: 74.9011 // Accuracy: 0.843333
--- Normalizer ---
Training time: 125.7301 // Accuracy: 0.848788
```

2. 데이터 분석 및 전처리

- Pipeline: 2-1. **Denoising with model (시행착오)**

1. (노이즈가 추가된 이미지, 노이즈 없는 이미지) 학습

2. 해당 모델로 노이즈 제거 -> but 잘 안됨



2. 데이터 분석 및 전처리

- Pipeline: **2-2. Denoising with Function**

1. 표기값 상이함 (0~1 , 0~255)

2. 노이즈 정의: 최대 표기값 A 기준, 0.5 * A 이하

3. **모든 Feature 해당 이미지 최대 표기값 (1A ~ 1.5A)로 나눴을 때 범위 (1.0 ~ 0.66)**

4. **Strict Threshold: 0.66** but **0.5** 일때 최고 score

2. 데이터 분석 및 전처리

- Pipeline: **3. Centering**

2. 데이터 분석 및 전처리

- Pipeline: **4. Enlarging**

- Label

Original

```
def enlarge_image_with_boundary_box(image):
```

1. Tight Boundary Box

```
def enlarge_image_to_square(image):
```

2. Square Boundary Box

2. 데이터 분석 및 전처리

- Pipeline: **5. 최종 Pipeline Set, 순서 확정**

1. Denoise, Normalier, Centering은 확정
2. Centering, Tight Zoom , Square Zoom Combination

```
Pipeline([
    ('denoise', FunctionTransformer(denoise_with_max)),
    ('image_centering', ImageCenteringTransformer()),
    ('image_enlarging', ImageEnlargingTransformer()),
    ('normalizer', Normalizer()),
    ('학습할 모델', 학습할 모델())
])
```

ex) MLP: Square zoom 최적

```
Transformers: image_centering, tight_image_enlarging // Training time: 158.4643 // Accuracy: 0.921010
Transformers: image_centering, square_image_enlarging // Training time: 130.9702 // Accuracy: 0.925960
```

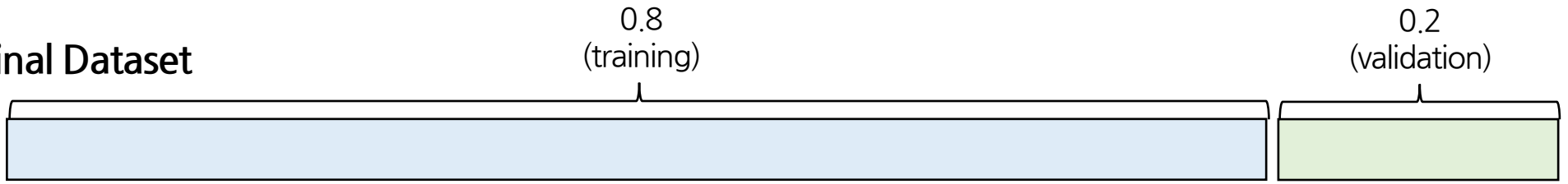
ex) Extra-Trees: Tight zoom 최적

```
Transformers: image_centering, tight_image_enlarging // Training time: 46.5714 // Accuracy: 0.930606
Transformers: image_centering, square_image_enlarging // Training time: 46.1861 // Accuracy: 0.923737
```

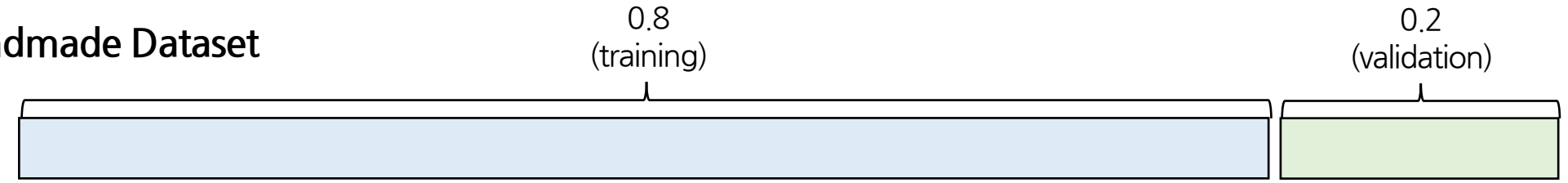
2. 데이터 분석 및 전처리

- Prepare the data (모델 선정을 위함)

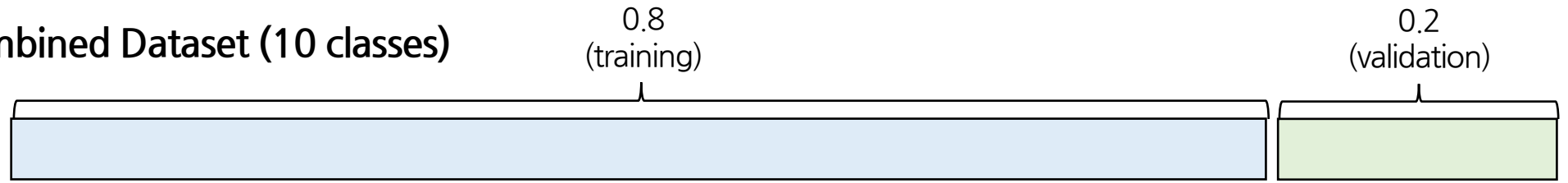
- **Original Dataset**



- **Handmade Dataset**



- **Combined Dataset (10 classes)**

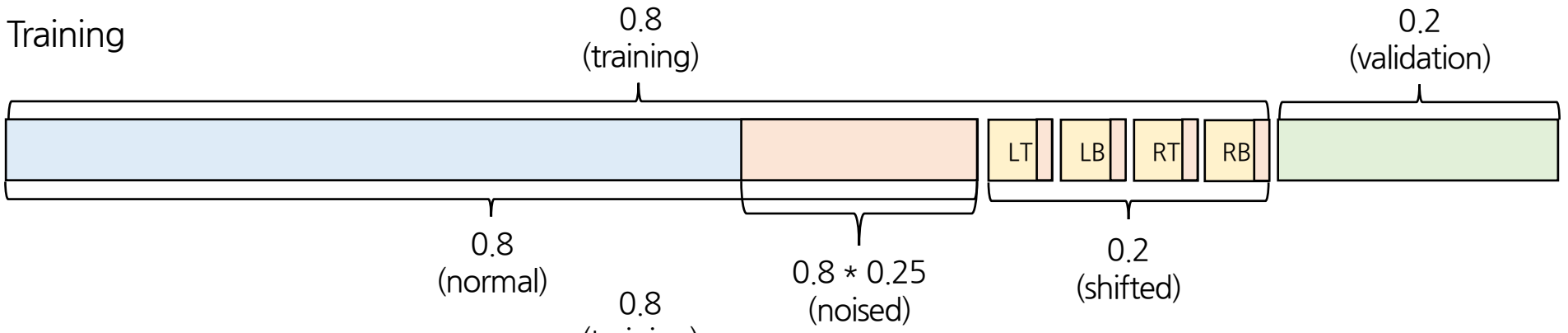


2. 데이터 분석 및 전처리

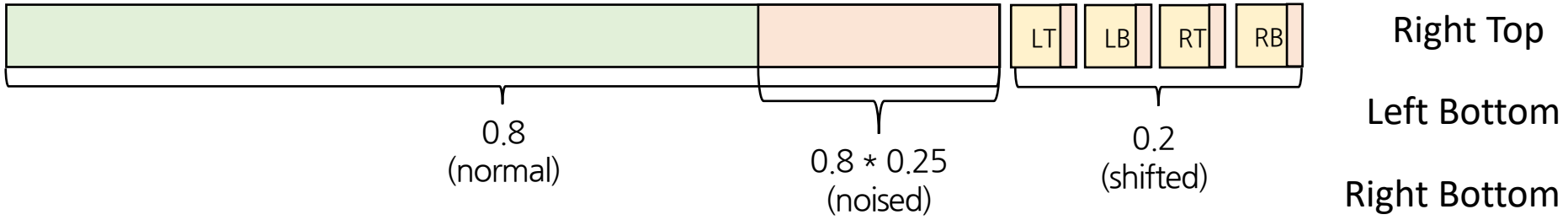
- Prepare the data (최종 모델 학습 / Pipeline을 테스트 위함)

```
def noise_add(X):
def shift_image(image, direction):
```

- **Combined Dataset (15 classes)**
  - Training



  - Validation



Left Top

Right Top

Left Bottom

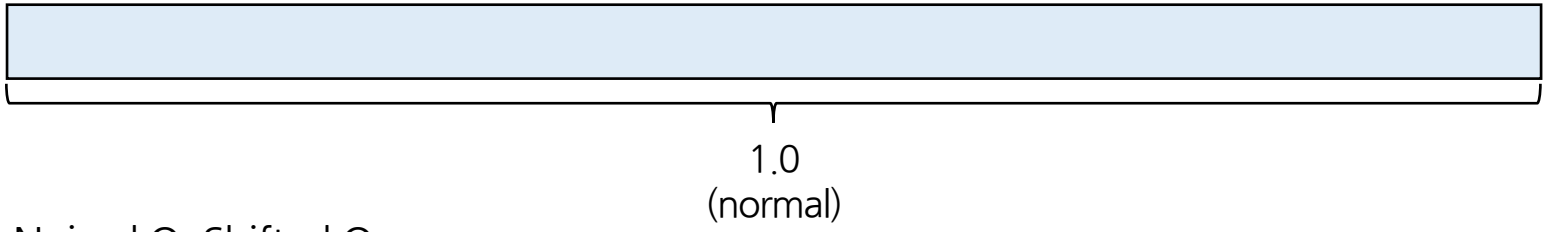
Right Bottom

2. 데이터 분석 및 전처리

- Prepare the data (최종 모델 학습 / Pipeline을 테스트 위함)

Final Test Dataset

- Noised X, Shifted X

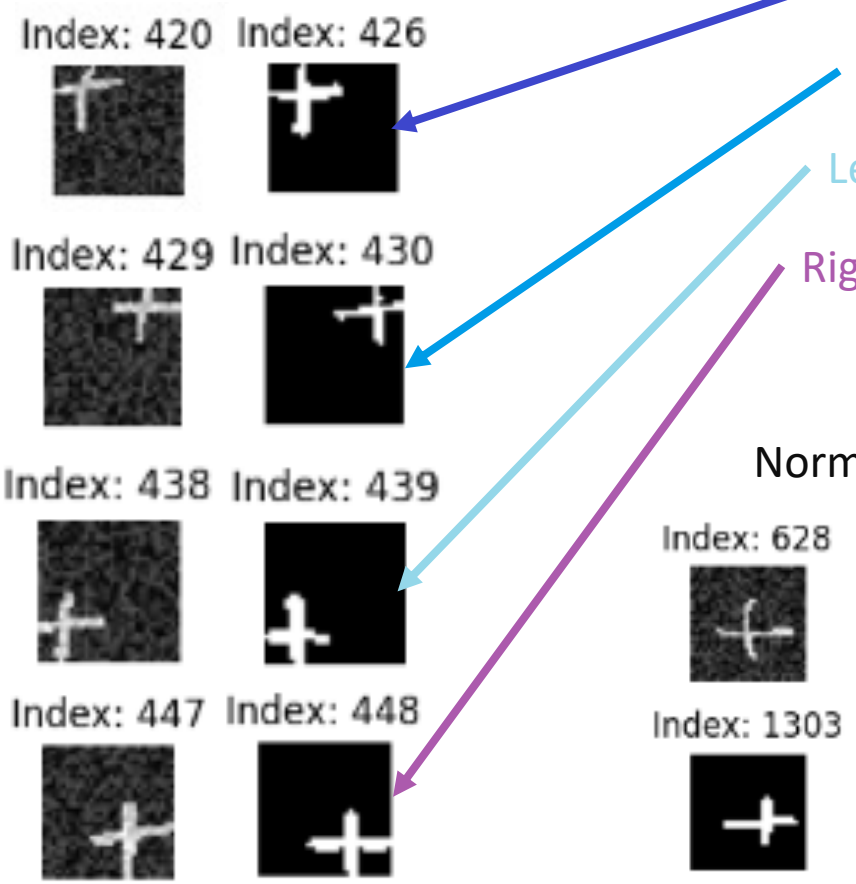


- Noised O, Shifted O

2. 데이터 분석 및 전처리

- Prepare the data

- **Noised, Shifted Final Test Dataset**



Left Top

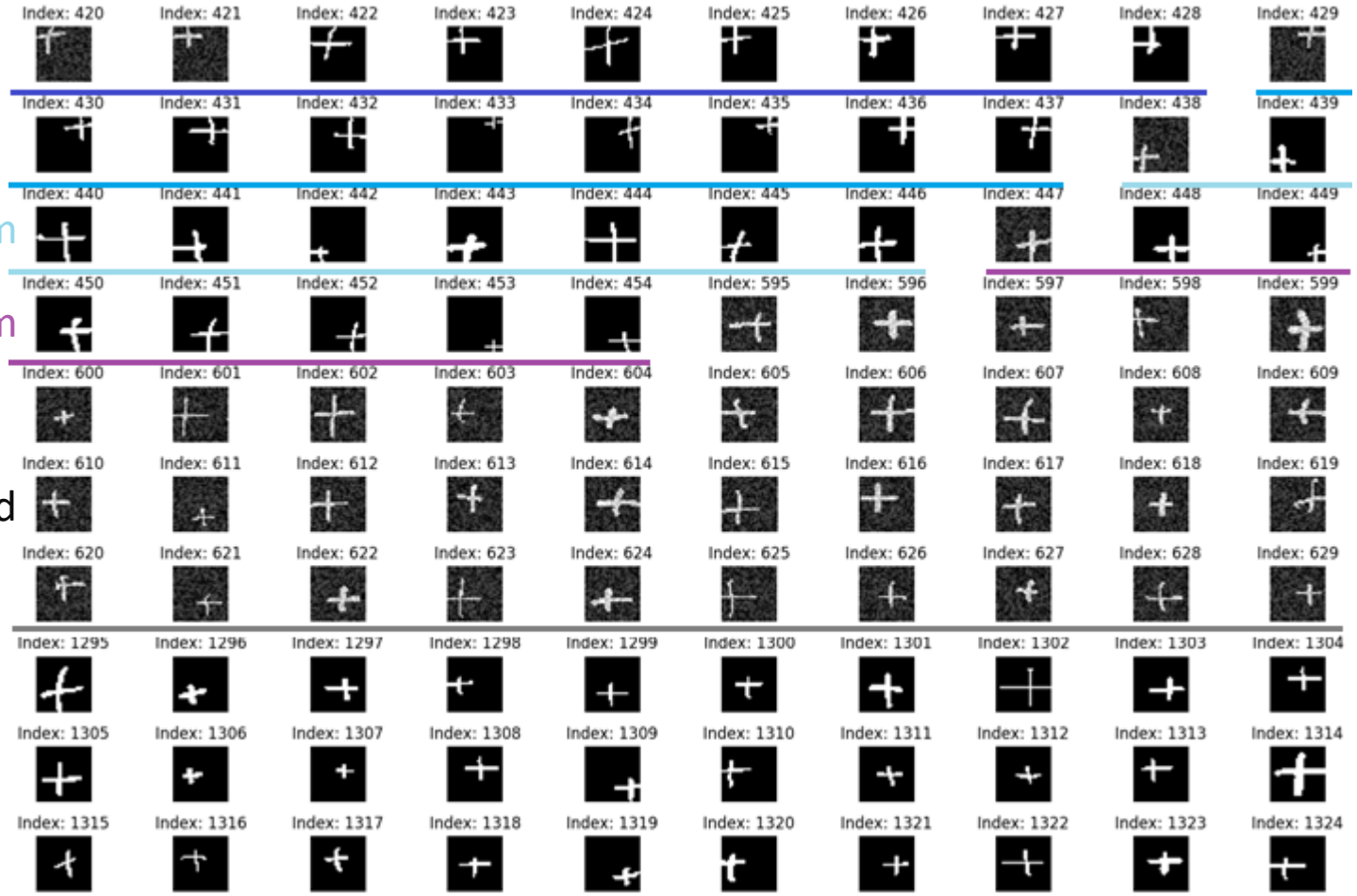
Right Top

Left Bottom

Right Bottom

Normal+Noised

Normal



3. Select and Train Models

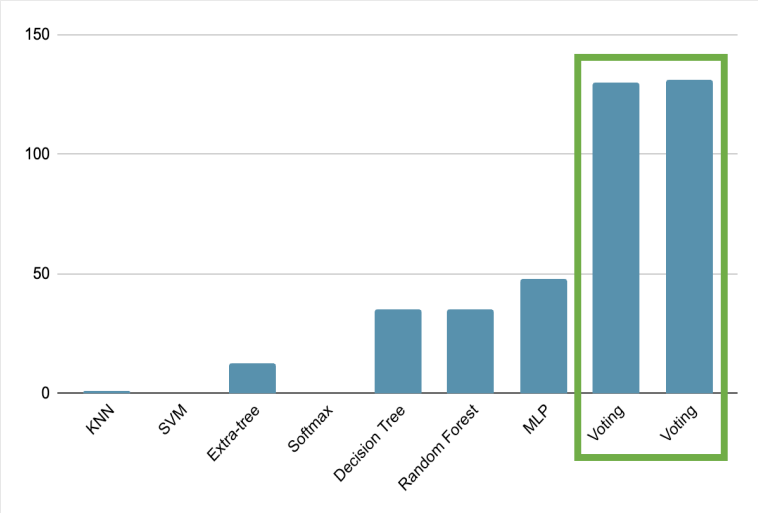
(1) Original Dataset → 학습, 예측 시간이 너무 긴 SVM, VotingClassifier를 사용하지 말자.

Model	KNN	SVM	Extra-tree	Softmax	Decision Tree	Random Forest	MLP	Voting (hard)	Voting (soft)
Training Time (sec)	0.9169		12.76156	0.05638	35.14291	35.14291	47.88176	~~129.84410~~	~~131.20013~~
Original Testset 예측시간	0.97564		0.29157	0.05638	0.02232	0.22734	0.05838	~~8.99141~~	~~9.08884~~
Accuracy on Original Testset	0.97564		0.97000	0.91979	0.87364	0.96636	0.97400	0.97336	0.97479
Team Testset 예측시간	0.23000		0.01019	0.00128	0.16250	0.00857	0.20750	0.77231	0.76504
Accuracy on Team Testset	0.23000		0.18000	0.18500	0.16250	0.19500	0.20750	0.19500	0.19750

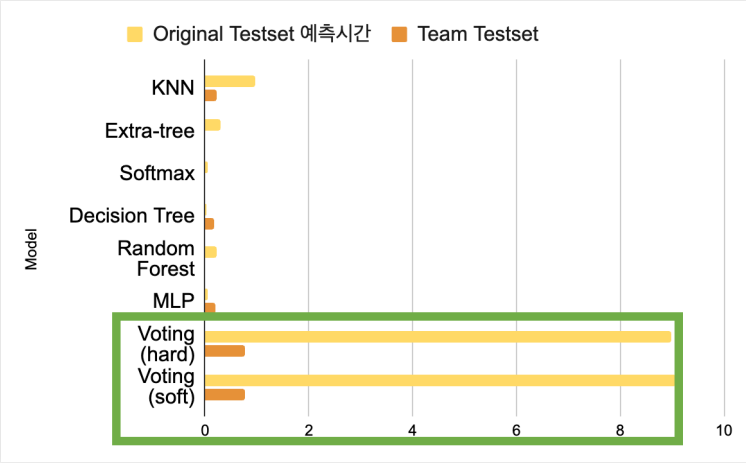
3. Select and Train Models

(1) Original Dataset → 학습, 예측 시간이 너무 긴 SVM, VotingClassifier를 사용하지 말자.

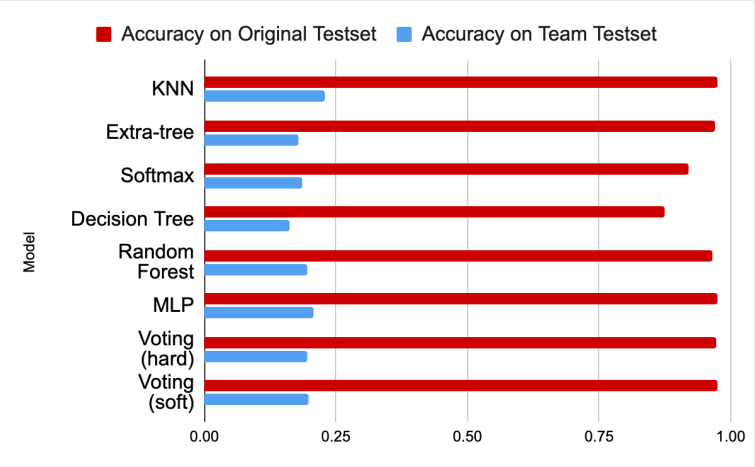
- Training Time (sec)



- Inference Time (Original, Team)



- Accuracy (Original, Team)



Original Testset : 예측이 잘 되었다. (약 90%) ←

Team Testset : 거의 예측이 안됨 (약 20%) ←

3. Select and Train Models

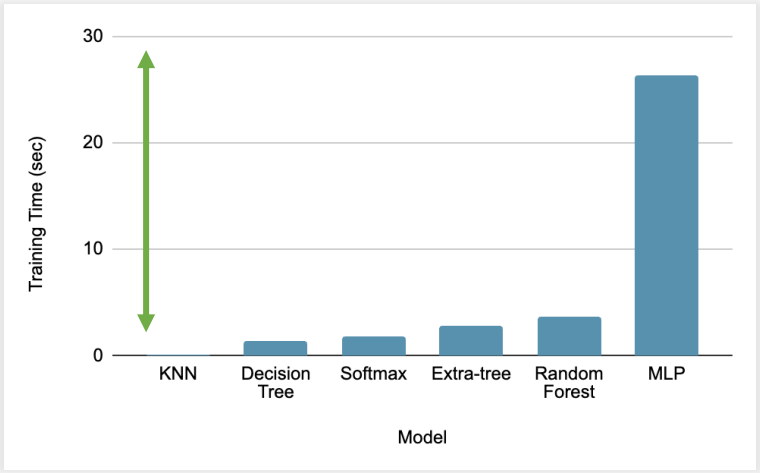
(2) Handmade Dataset → Softmax, DecisionTree의 성능이 낮으니 사용하지 말자.

Model	KNN	SVM	Extra-tree	Softmax	Decision Tree	Random Forest	MLP	Voting (hard)	Voting (soft)
Training Time (sec)	0.09063		2.82671	1.77935	1.44450	3.69965	26.31937		
Original Testset 예측시간	3.95527		0.23225	0.04901	0.02034	0.21984	0.08072		
Accuracy on Original Testset	0.68907		0.68571	~~0.35757~~	~~0.32886~~	0.62143	0.71271		
Team Testset 예측시간	0.05734		0.01308	0.00144	0.00124	0.01171	0.00197		
Accuracy on Team Testset	0.41500		0.37250	~~0.21000~~	~~0.22750~~	0.33000	0.38250		

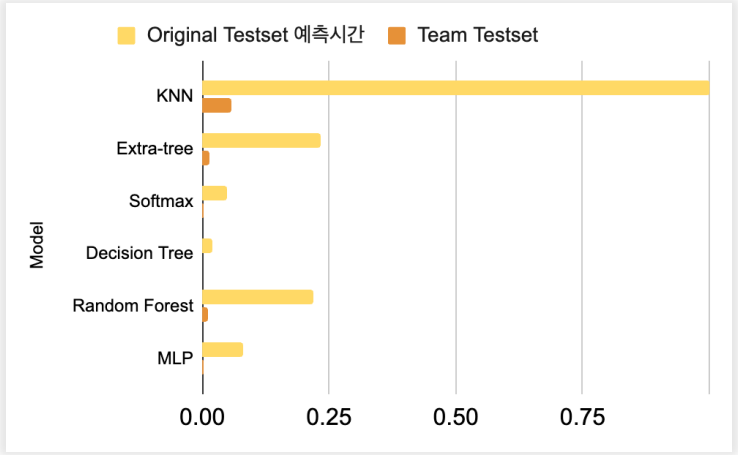
3. Select and Train Models

(2) Handmade Dataset → Softmax, DecisionTree의 성능이 낮으니 사용하지 말자.

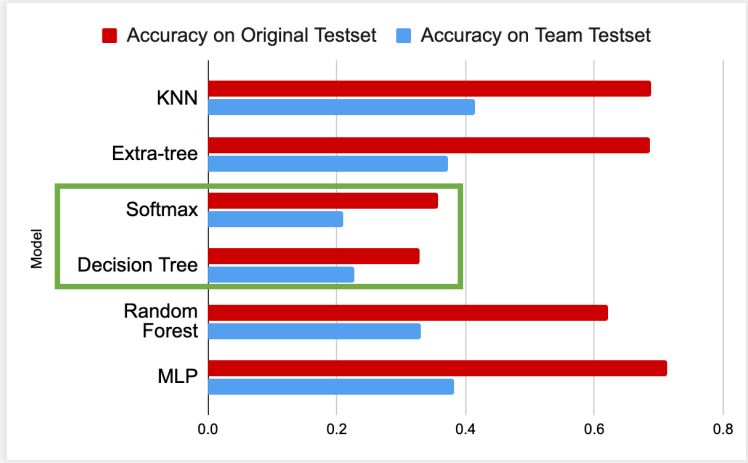
- Training Time (sec)



- Inference Time (Original, Team)



- Accuracy (Original, Team)



Original Testset : 예측 잘 안됨 (약 40% ~ 70%) ←

Team Testset : 예측 잘 안되지만 이전보다 성능 향상됨 (약 30~40%) ←

3. Select and Train Models

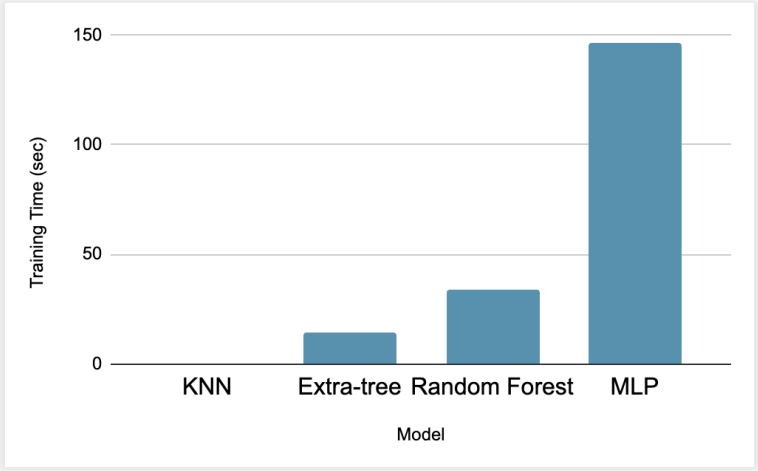
- **Combined Dataset (10 classes)** → KNN은 예측 시간이 길고, Random은 성능이 낮으니 사용하지 말자.

Model	KNN	SVM	Extra-tree	Softmax	Decision Tree	Random Forest	MLP	Voting (hard)	Voting (soft)
Training Time (sec)	0.29248		14.58727			33.62962	146.48267		
Original Testset 예측시간	~~19.26764~~		0.31900			0.24504	0.0561		
Accuracy on Original Testset	0.97514		0.96914			0.96543	0.9764		
Team Testset 예측시간	0.24254		0.01468			0.01233	0.00418		
Accuracy on Team Testset	0.43000		0.39000			~~0.34500~~	0.38250		

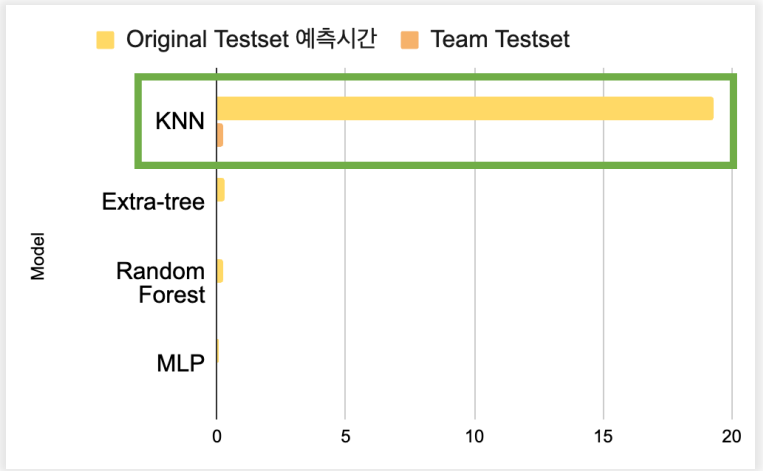
3. Select and Train Models

- Combined Dataset (10 classes) → KNN은 예측 시간이 길고, Random은 성능이 낮으니 사용하지 말자.

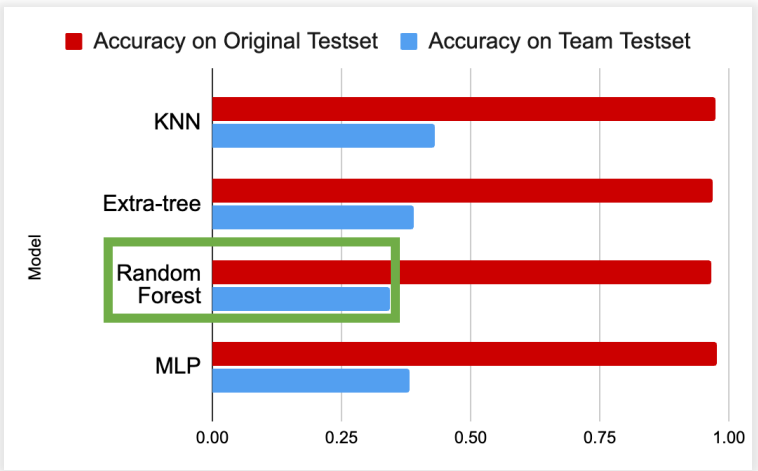
- Training Time (sec)



- Inference Time (Original, Team)



- Accuracy (Original, Team)



→ 후보 1 : Extra-Tree

→ 후보 2 : MLP

4. Fine-Tune the Model

- 15 classes 분류기 : GridSearch로 최적의 파라미터 찾기

(1) Extra-Tree

- **n_estimators** : [100, 200, **300**]
- **max_depth** : [10, **20**]

- Search (Training) Time : 415(sec)
- Validation Dataset - Accuracy : 0.9282
- Validation Dataset - Inference Time : 0.87(sec)

(2) MLP

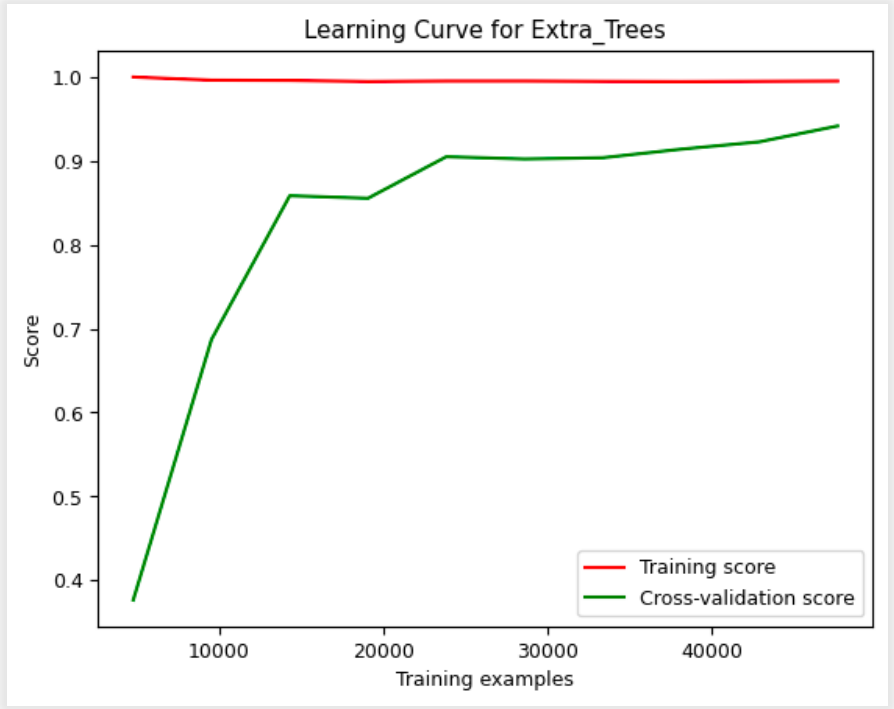
- max_iter : [500, 1000, **2000**]
- alpha : [0.0001, 0.001, **0.01**, 0.1]

- Search (Training) Time : 8017(sec)
- **Validation Dataset - Accuracy** : 0.9387
- **Validation Dataset - Inference Time** : 0.47(sec)

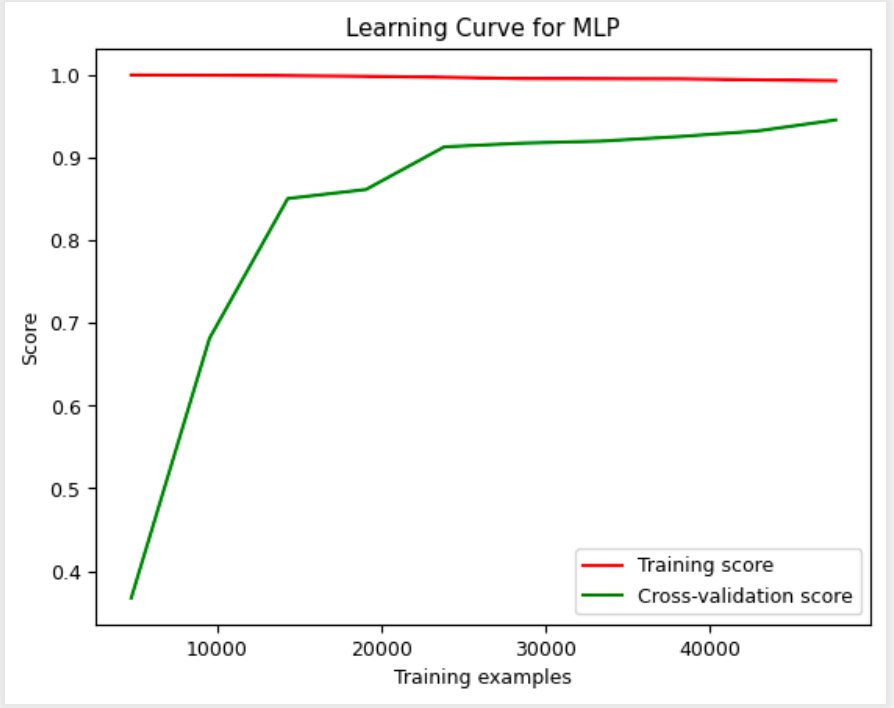
4. Fine-Tune the Model

- 15 classes 분류기 : Learning Curve

(1) Extra-Tree



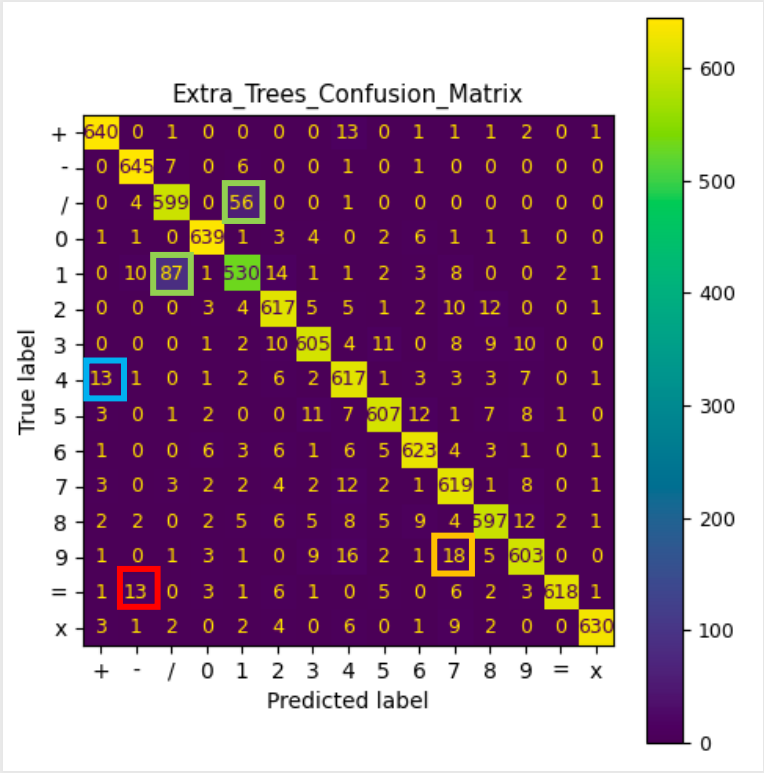
(2) MLP



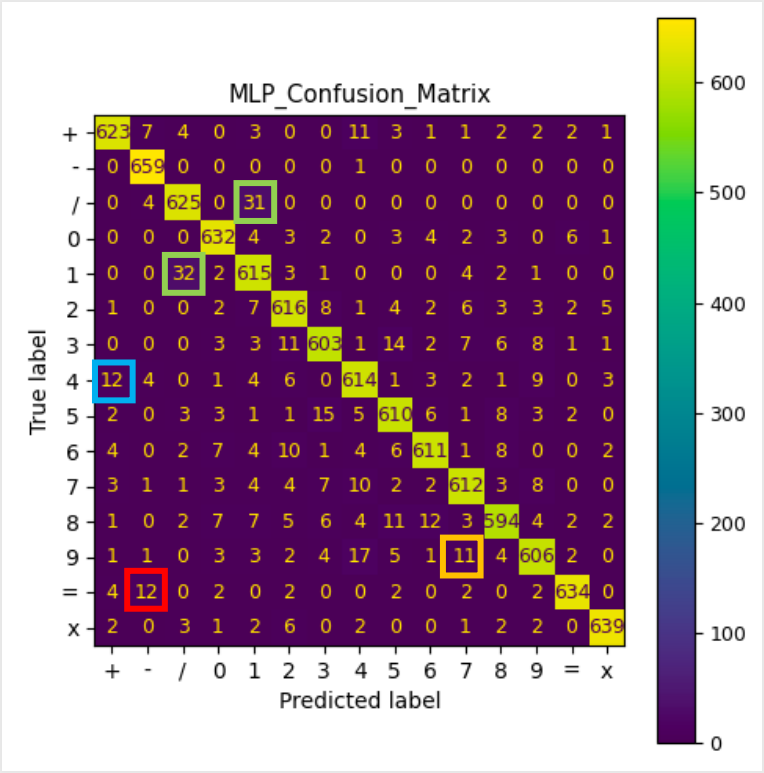
4. Fine-Tune the Model

- 15 classes 분류기 : **Confusion Matrix** (Validation Dataset으로 진행)

(1) Extra-Tree



(2) MLP



→ 최종 : **MLP**가 가장 좋은 성능을 보인다.

4. Fine-Tune the Model

- 15 classes 분류기 : **Voting Classifier**

(1) Voting Classifier (soft)

- Search (Training) Time : 357(sec)
- Validation Dataset - Accuracy : 0.9422
- Validation Dataset - Inference Time : 1.62(sec)

(2) Voting Classifier (hard)

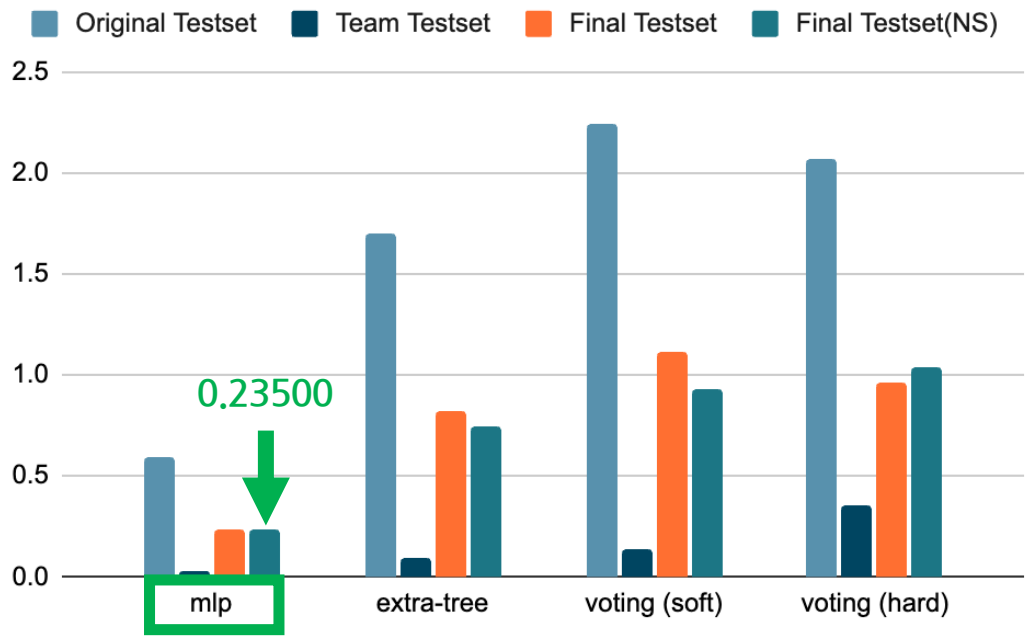
- Search (Training) Time : 393(sec)
- **Validation Dataset - Accuracy** : 0.9305
- Validation Dataset - Inference Time : 1.66(sec)

5. Final Test

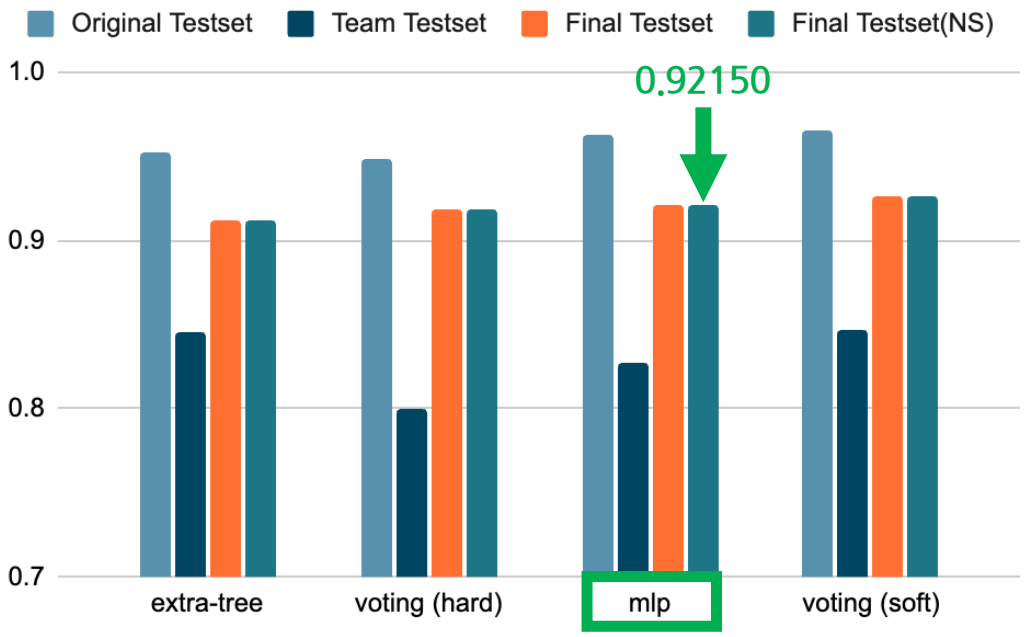
		Extra-Tree	MLP	Voting (hard)	Voting (soft)
Original Testset (10classes, 14,000)	Inference Time	1.69720	0.59328	2.06655	2.25069
	Accuracy	0.95264	0.96271	0.94907	0.96500
Team Testset (10classes, 400)	Inference Time	0.08964	0.02795	0.35179	0.13811
	Accuracy	0.84500	0.82750	0.80000	0.84750
Final Testset (15classes, 5,478)	Inference Time	0.82431	0.23394	0.96035	1.12011
	Accuracy	0.91201	0.92150	0.91822	0.92680
Final Testset (15classes, **Noised + shifted**, 5,478)	Inference Time	0.74779	**0.23500**	1.04238	0.92722
	Accuracy	0.91201	**0.92150**	0.91822	0.92680

5. Final Test

- **Inference Time** (Final Testset(NS) 기준 정렬)

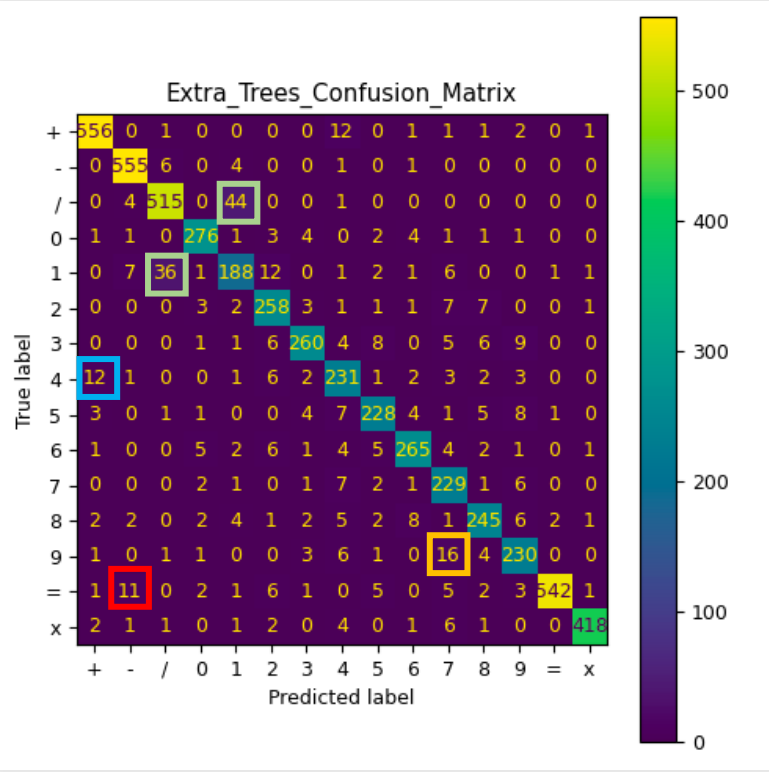


- **Accuracy** (Final Testset(NS) 기준 정렬)

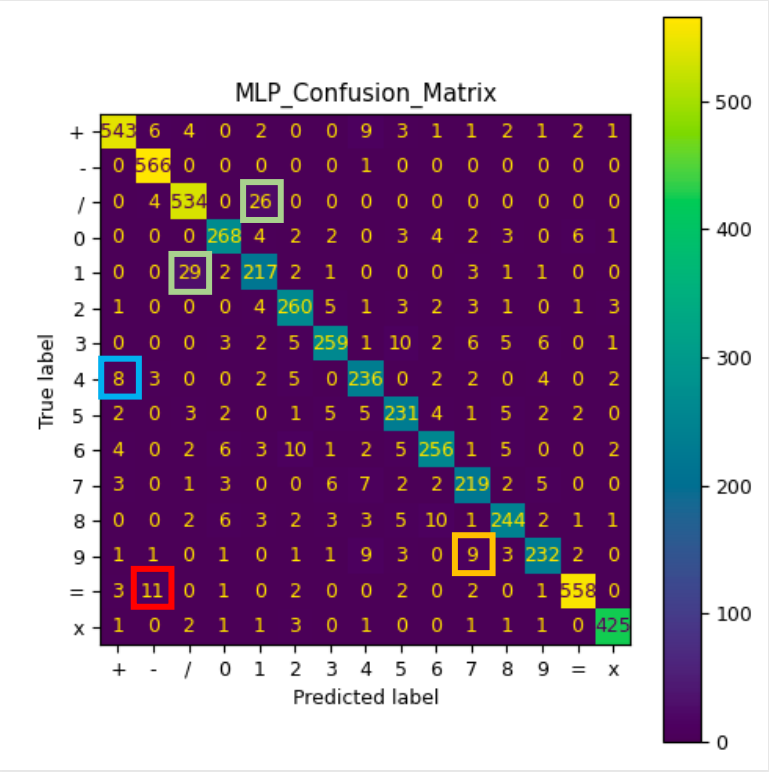


5. Final Test

(1) Extra-Tree

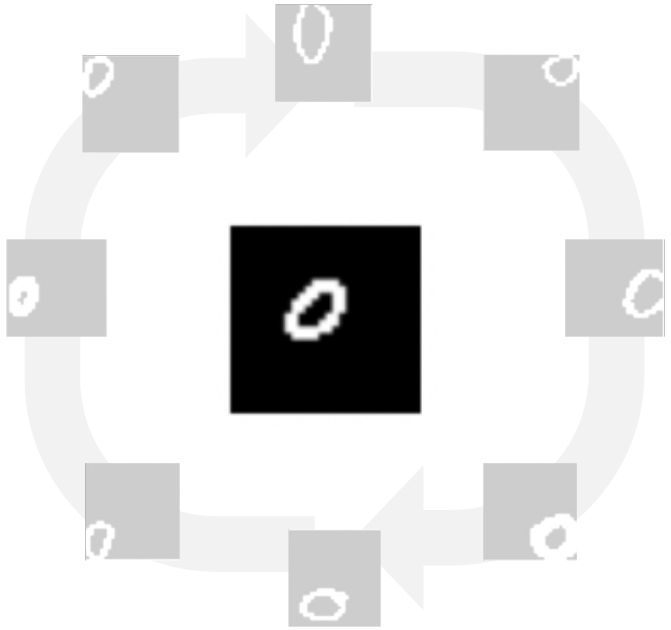


(2) MLP



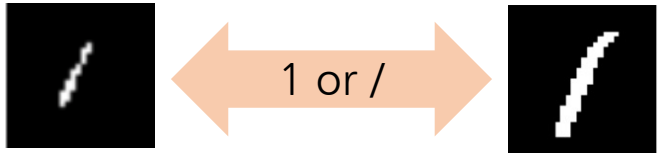
6. Feedback

- Shifted data 처리 방식 결정 과정



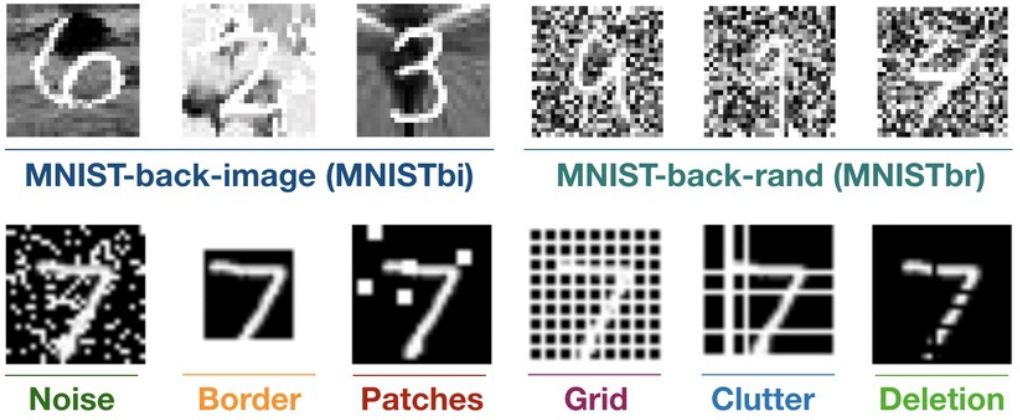
6. Feedback

- 한계 1 : Rotate



- 한계 2 : 테두리

- 한계 3 : 다양한 노이즈



감사합니다