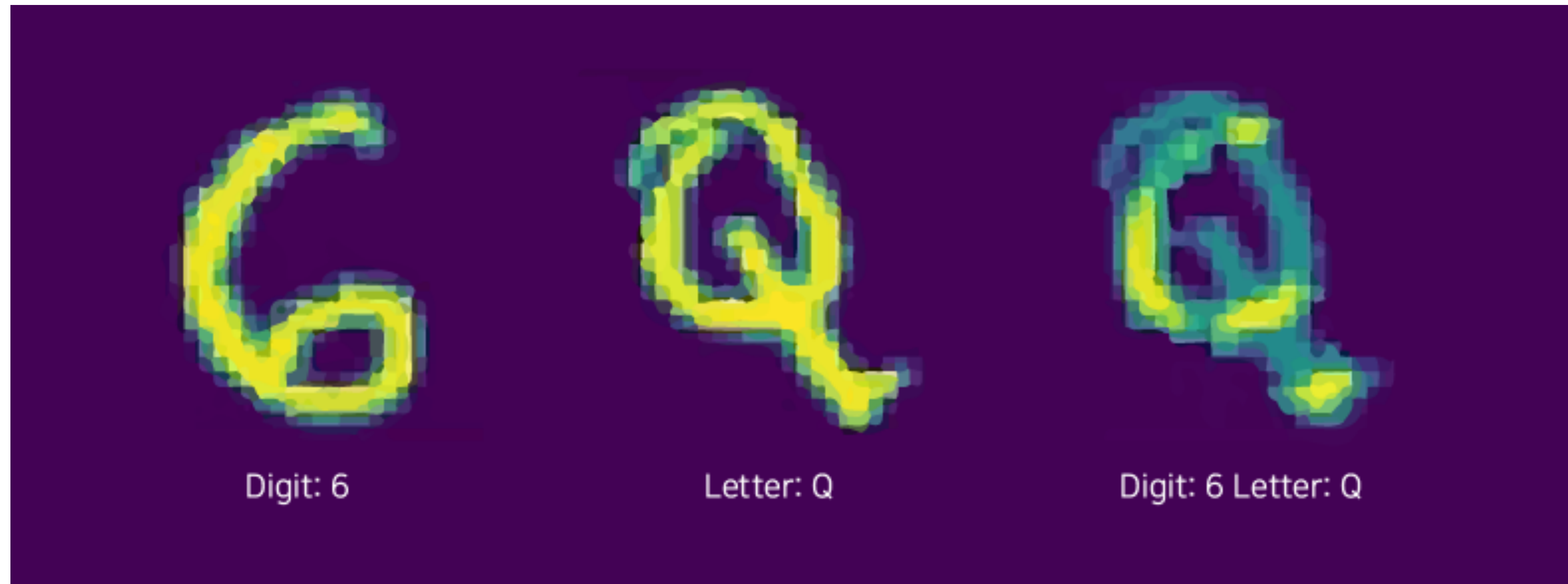


# Computer Vision Task EDA

DACON 컴퓨터 비전 학습 경진대회

# What is the problem?

## 1. 어떠한 문제인가?



- Digit : 원본 숫자 (0~9까지 총 10개)
- Letter : 원본 숫자를 가리는 문자 (알파벳 대소문자 포함 총 52개)

# What is the problem?

## 2. 규칙

- Public Score: 전체 테스트 데이터 중 1%로 채점
- Private Score: 나머지 테스트 데이터로 채점
- 외부데이터 사용 불가, 사전학습모델 (pretrained model) 사용 불가

# What is the problem?

## 3. 제공되는 데이터의 예시

Airtable MD7\_DESCRIPTION

train\_sample test\_sample submission\_sample

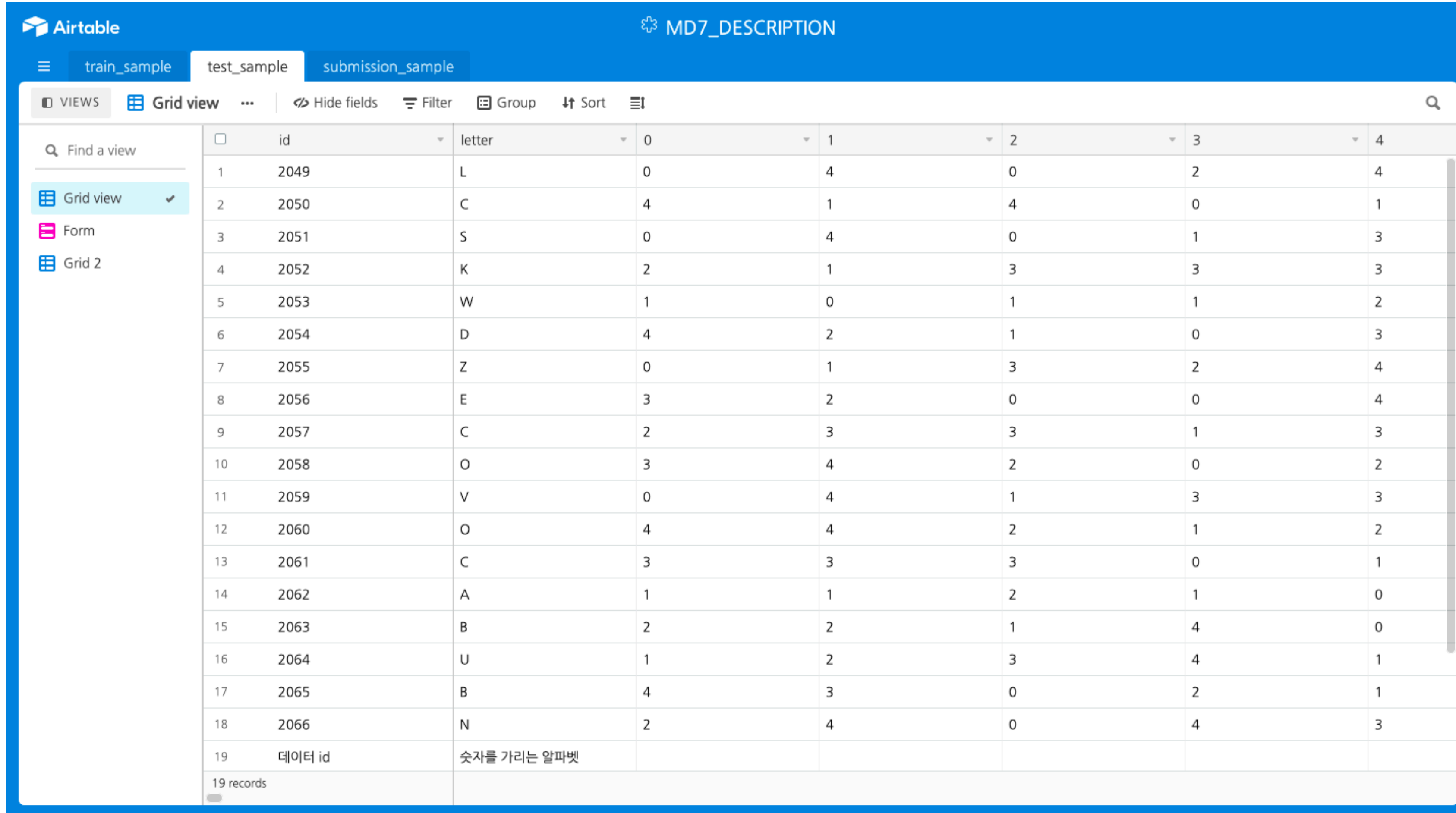
VIEWS Grid view Hide fields Filter Group Sort

	id	digit	letter	0	1	2	3
1	데이터 id	가려진 숫자	숫자를 가리는 알파벳	28x28 이미지의 각 픽셀 값	28x28 이미지의 각 픽셀 값	28x28 이미지의 각 픽셀 값	28x28 이미지의 각 픽셀 값
2	1	5	L	1	1	1	4
3	2	0	B	0	4	0	0
4	3	4	L	1	1	2	2
5	4	9	D	1	2	0	2
6	5	6	A	3	0	2	4
7	6	8	C	4	3	0	3
8	7	1	Q	0	0	4	2
9	8	3	M	1	0	3	4
10	9	6	F	0	1	0	4
11	10	8	J	4	3	4	0
12	11	8	H	3	4	4	1
13	12	2	N	0	4	0	2
14	13	3	C	2	4	4	1
15	14	3	X	4	2	2	4
16	15	9	H	3	2	2	3
17	16	0	I	1	3	2	3
18	17	7	R	1	4	4	3
19	18	7	A	2	3	0	4

20 records

# What is the problem?

## 3. 제공되는 데이터의 예시



Airtable MD7\_DESCRIPTION

train\_sample test\_sample submission\_sample

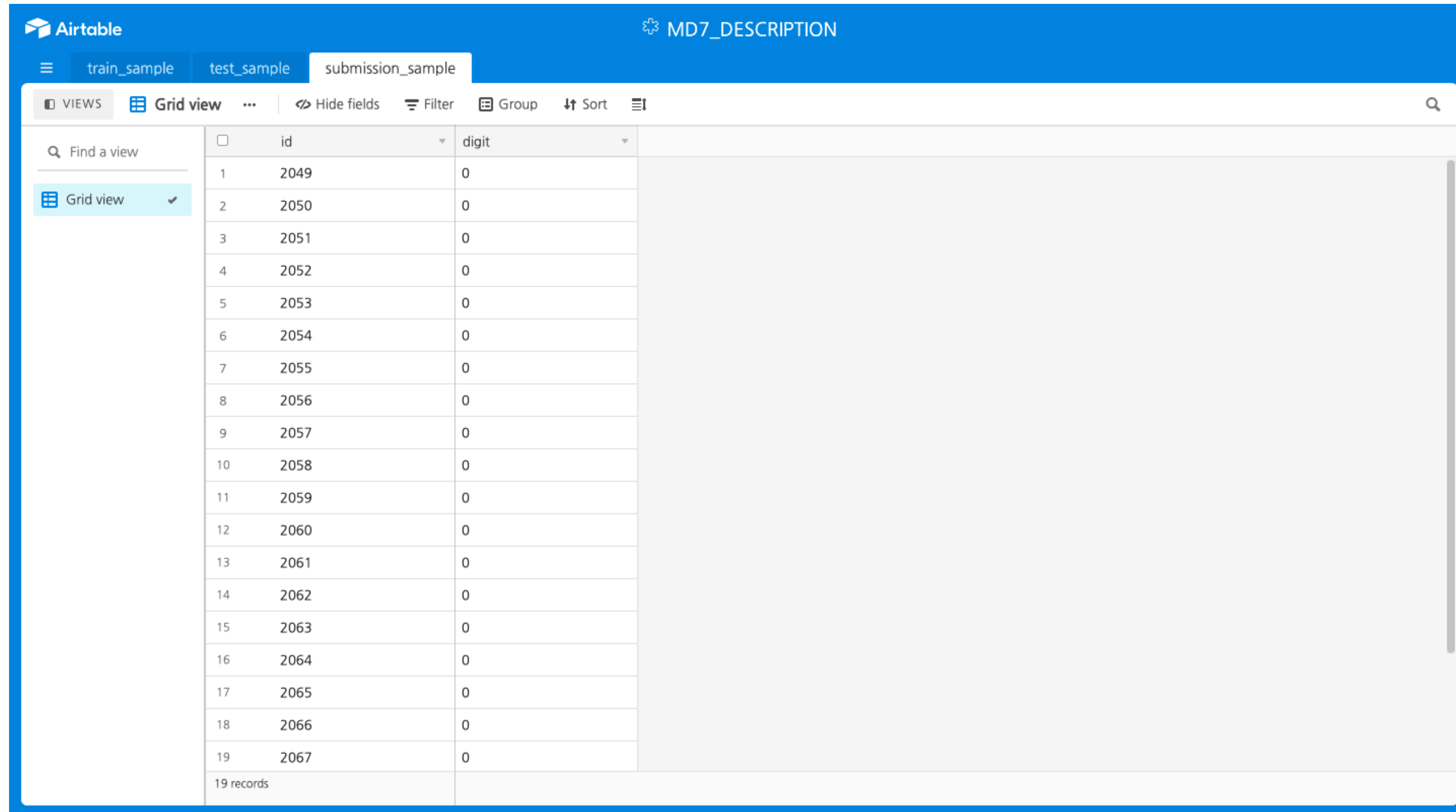
VIEWS Grid view Hide fields Filter Group Sort

	id	letter	0	1	2	3	4
1	2049	L	0	4	0	2	4
2	2050	C	4	1	4	0	1
3	2051	S	0	4	0	1	3
4	2052	K	2	1	3	3	3
5	2053	W	1	0	1	1	2
6	2054	D	4	2	1	0	3
7	2055	Z	0	1	3	2	4
8	2056	E	3	2	0	0	4
9	2057	C	2	3	3	1	3
10	2058	O	3	4	2	0	2
11	2059	V	0	4	1	3	3
12	2060	O	4	4	2	1	2
13	2061	C	3	3	3	0	1
14	2062	A	1	1	2	1	0
15	2063	B	2	2	1	4	0
16	2064	U	1	2	3	4	1
17	2065	B	4	3	0	2	1
18	2066	N	2	4	0	4	3
19	데이터 id	숫자를 가리는 알파벳					

19 records

# What is the problem?

## 3. 제공되는 데이터의 예시



Airtable MD7\_DESCRIPTION

train\_sample test\_sample submission\_sample

VIEWS Grid view Hide fields Filter Group Sort

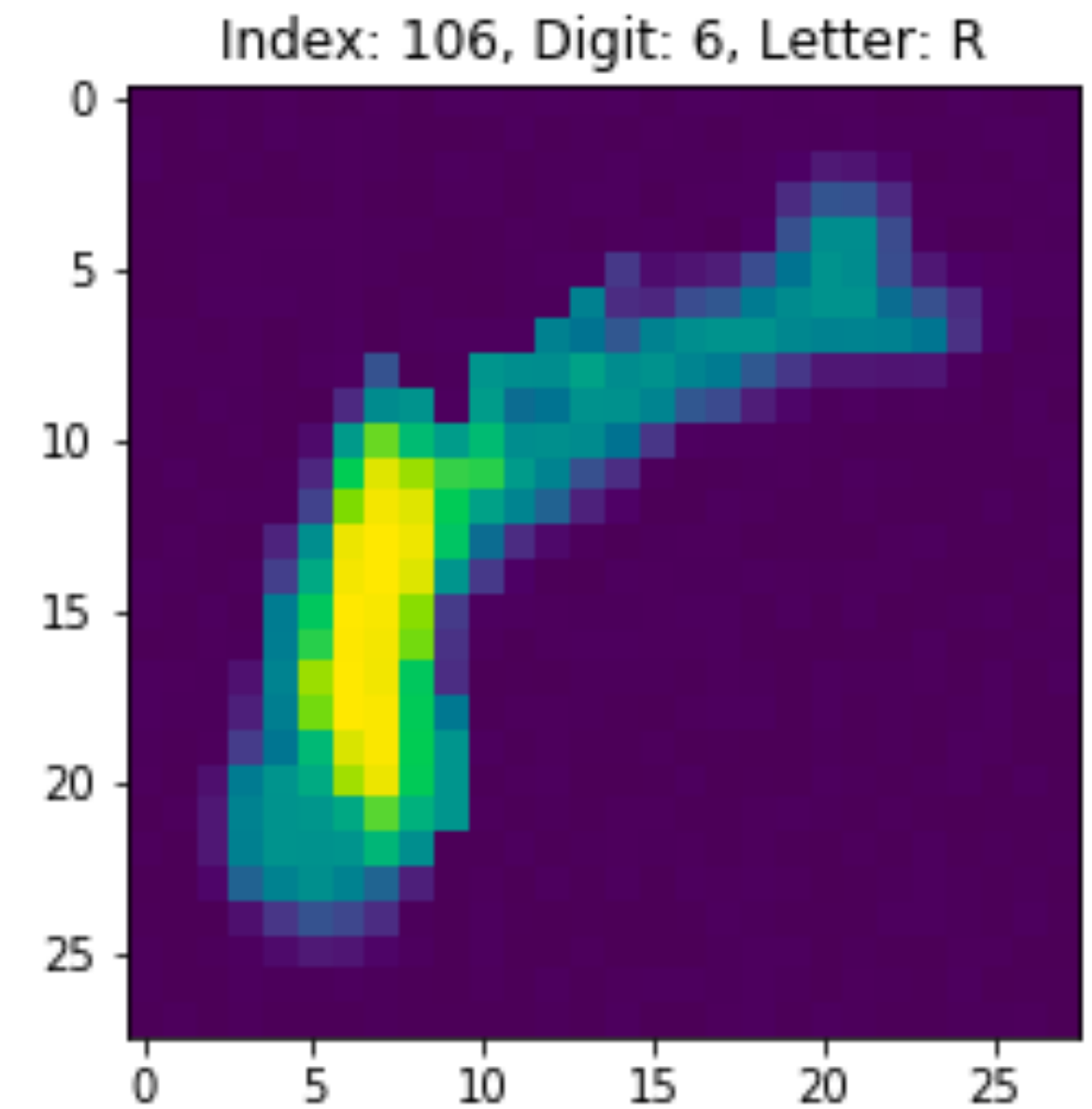
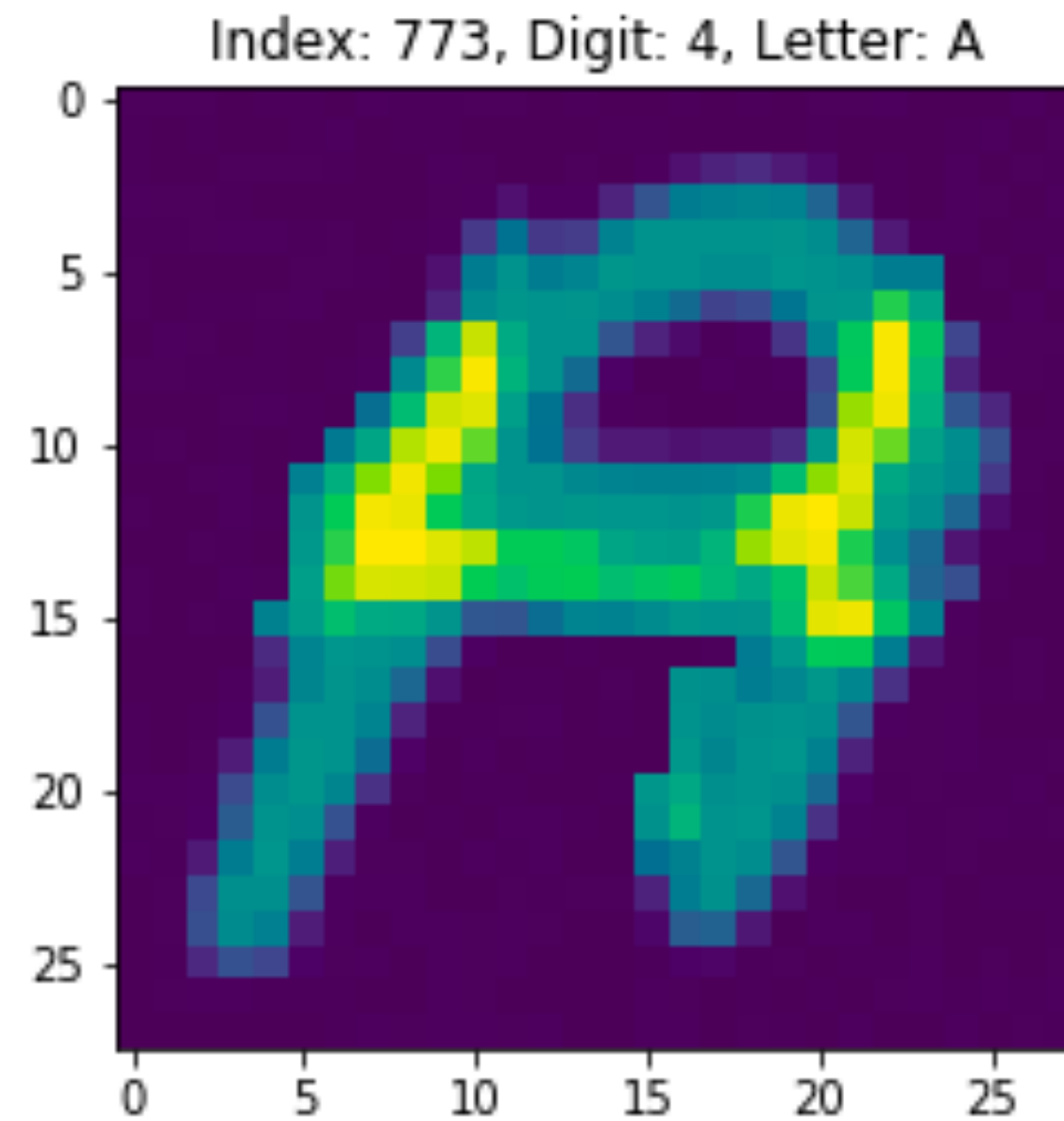
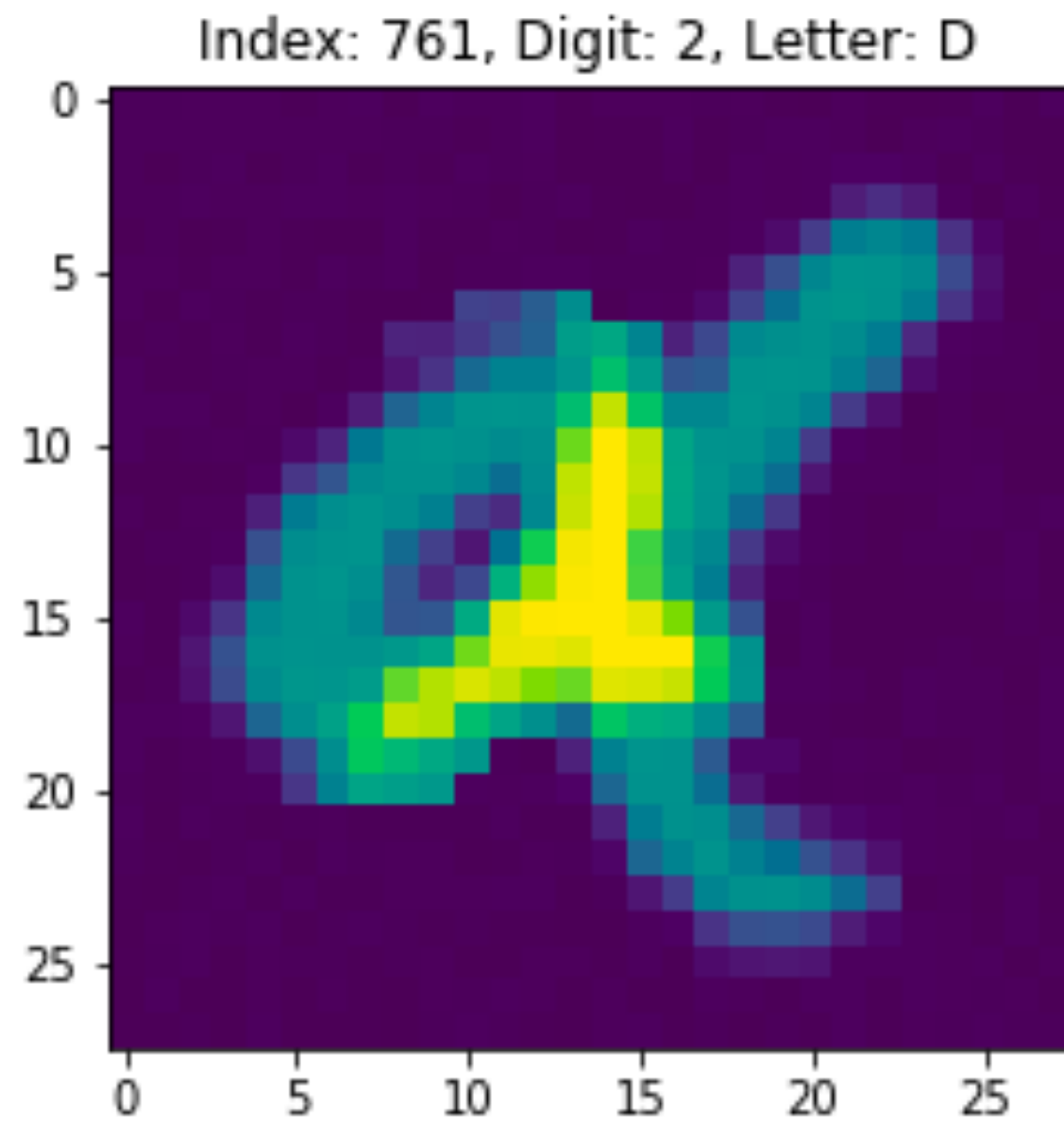
Find a view

Grid view

	id	digit
1	2049	0
2	2050	0
3	2051	0
4	2052	0
5	2053	0
6	2054	0
7	2055	0
8	2056	0
9	2057	0
10	2058	0
11	2059	0
12	2060	0
13	2061	0
14	2062	0
15	2063	0
16	2064	0
17	2065	0
18	2066	0
19	2067	0
19 records		

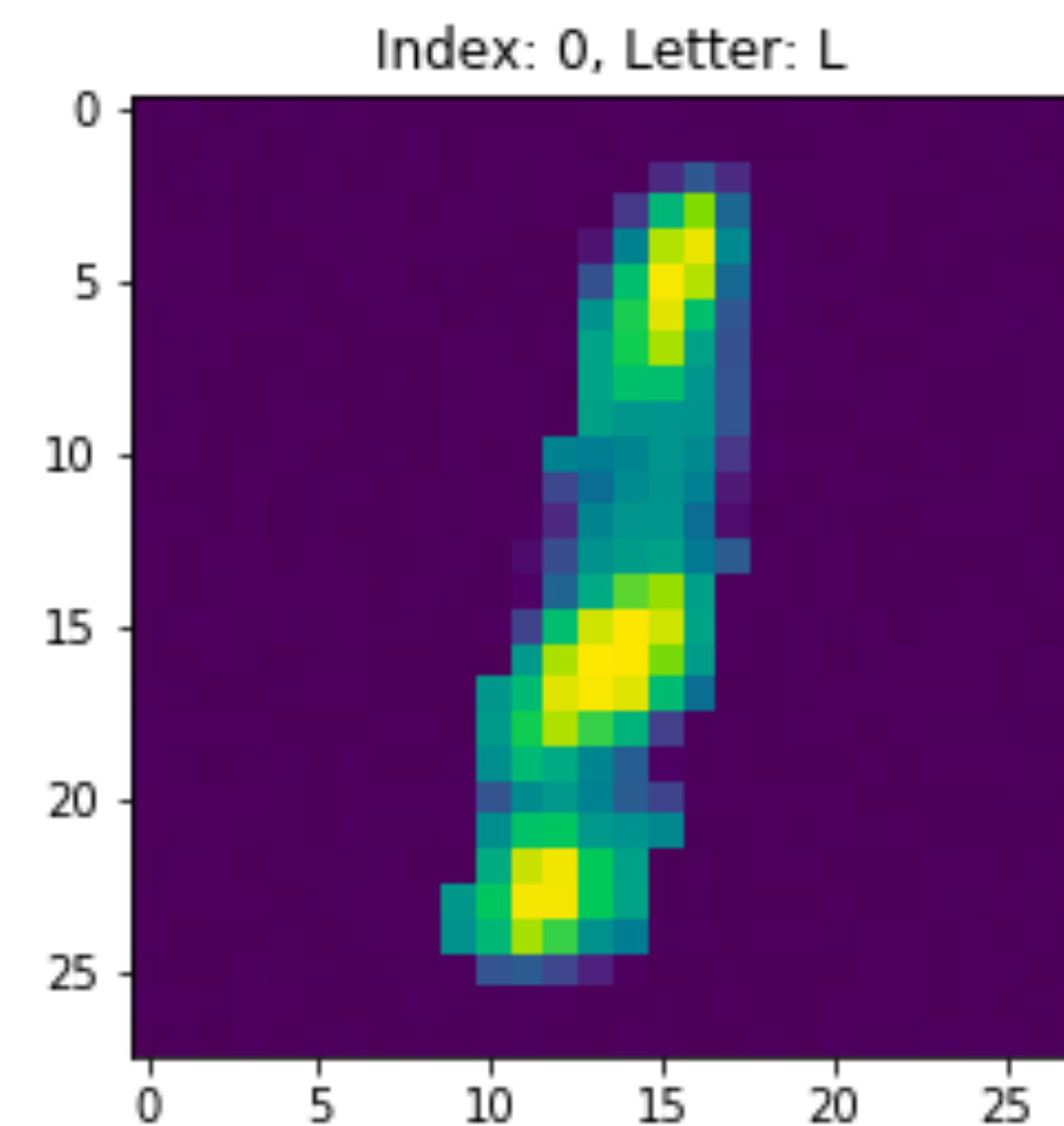
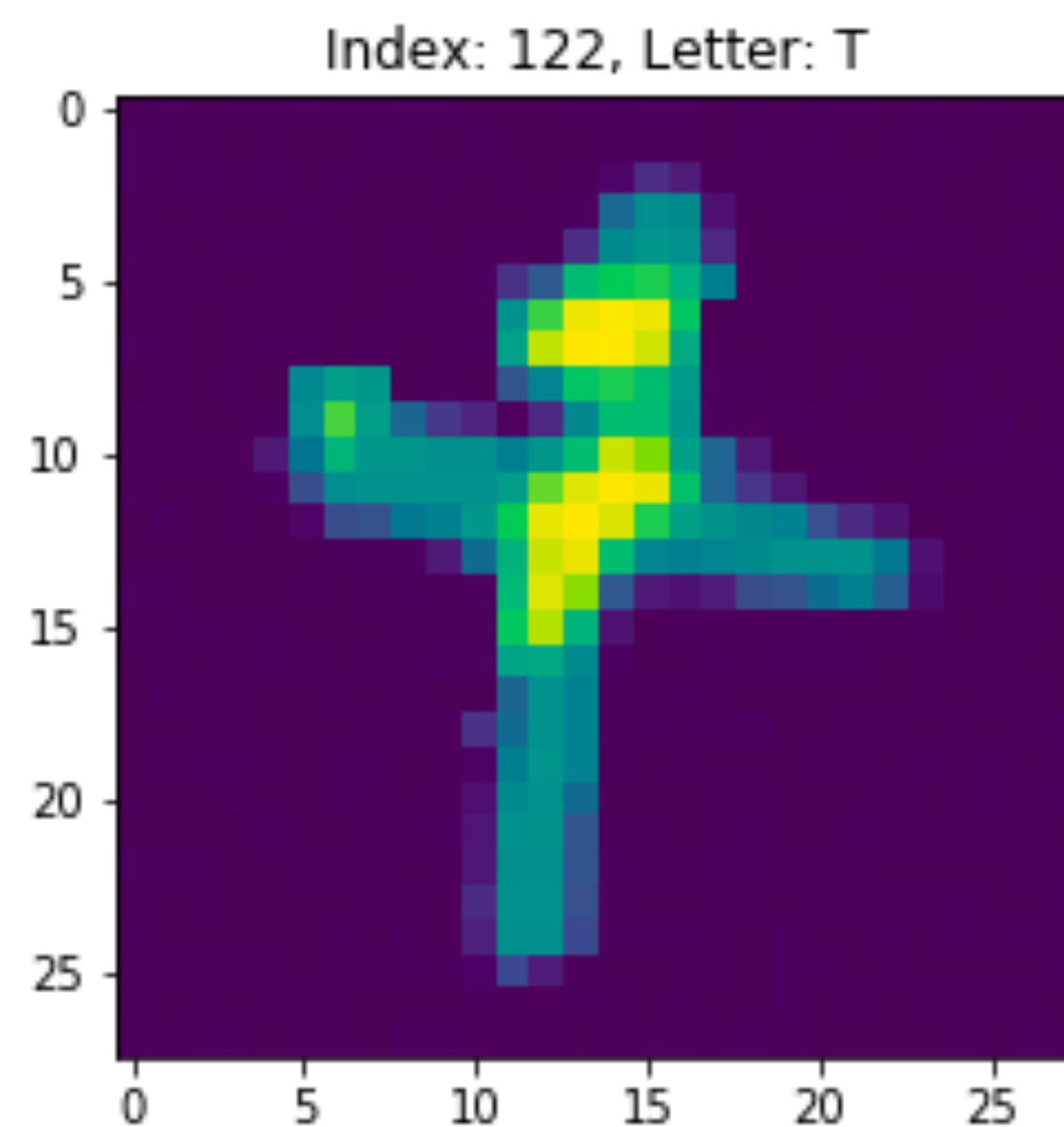
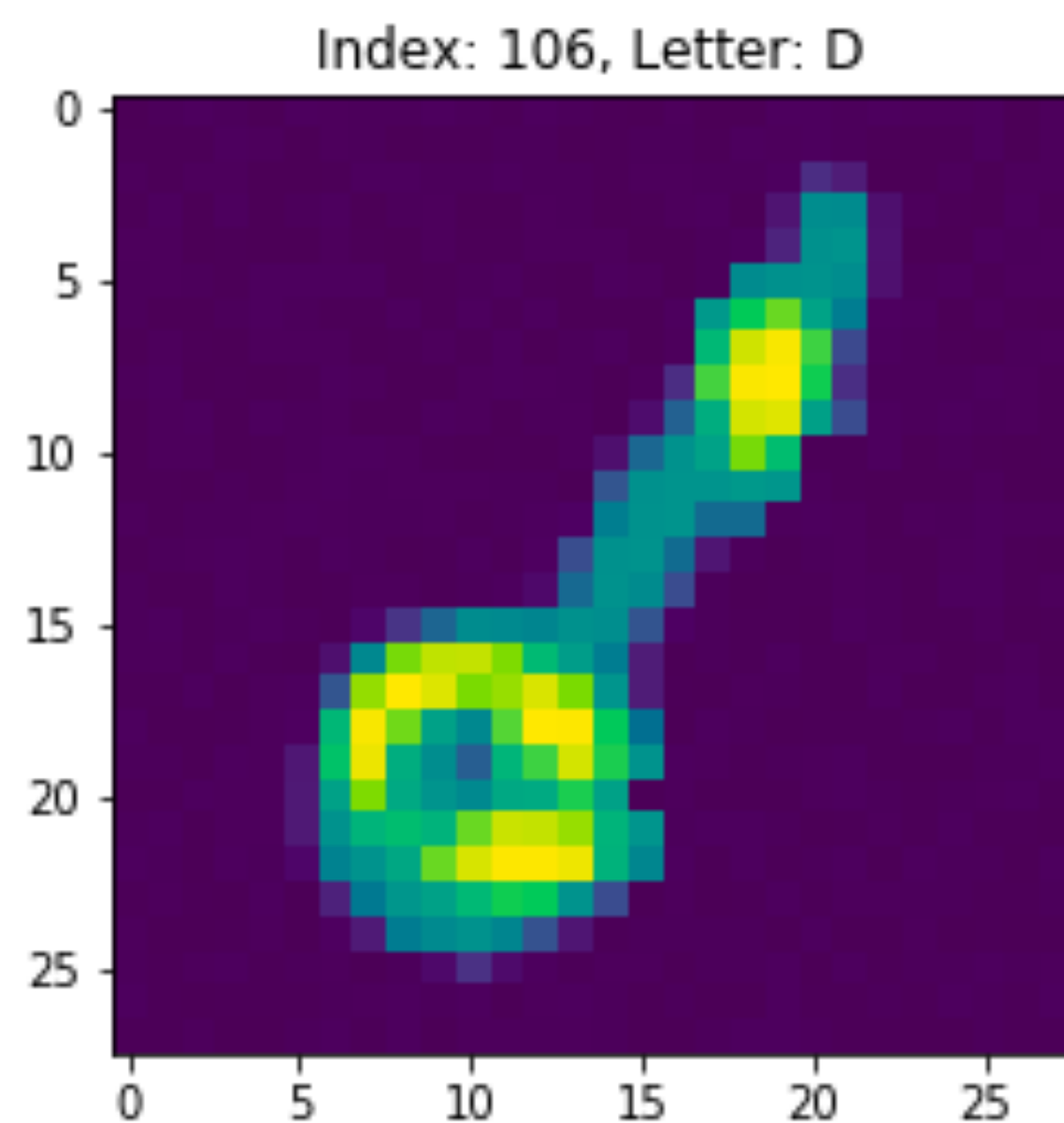
# What is the problem?

## 4. Data Visualization을 통한 확인(Train Data set).



# What is the problem?

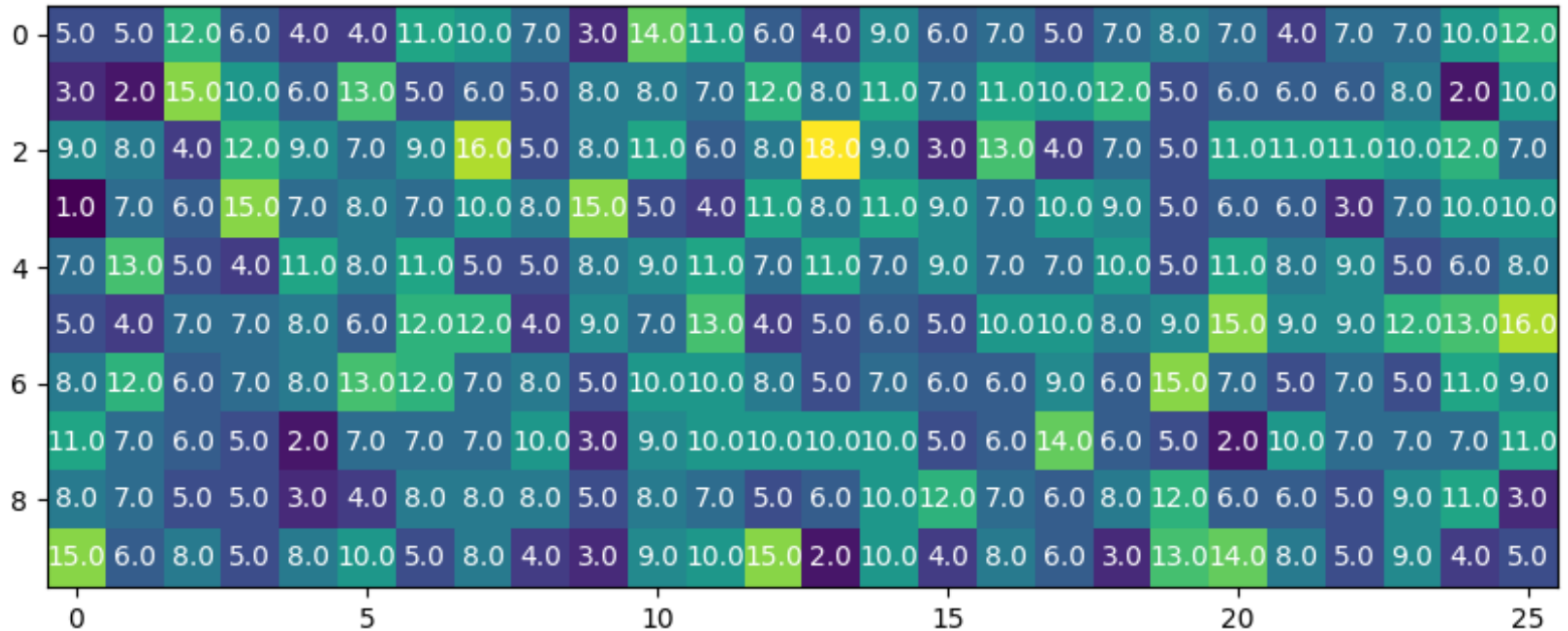
## 4. Data Visualization을 통한 확인(Test Data set).





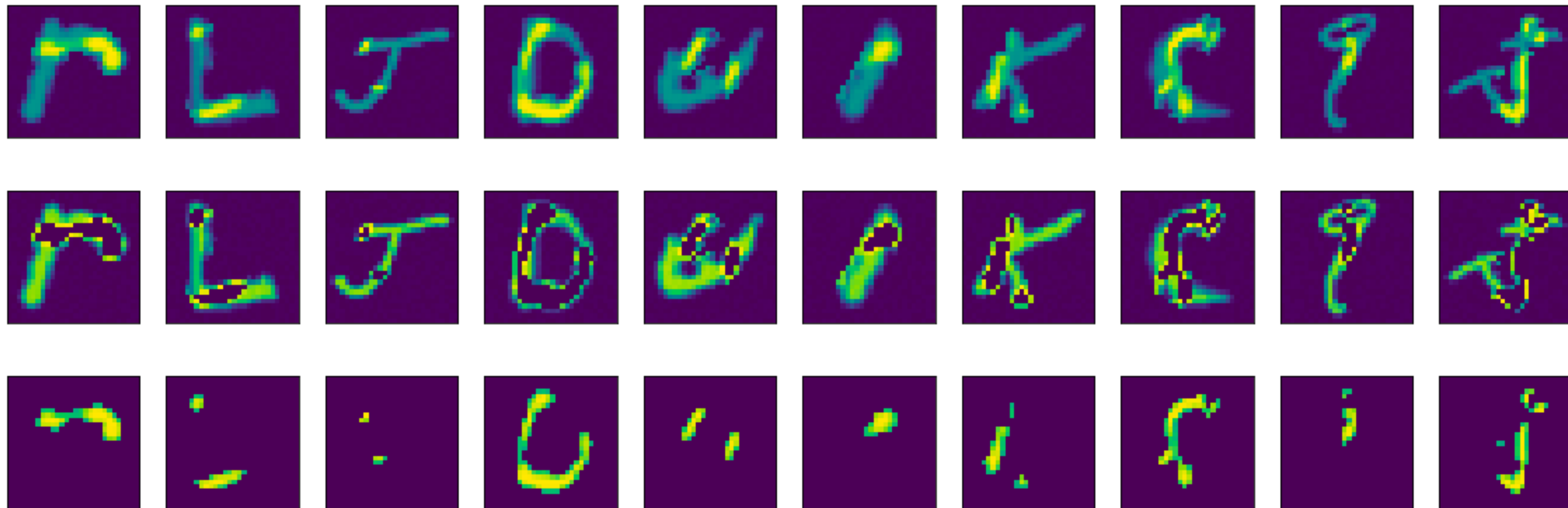
# What is the problem?

5. Data 분포 통계. (0~9, A~Z까지 가능한 경우의 수 260가지)

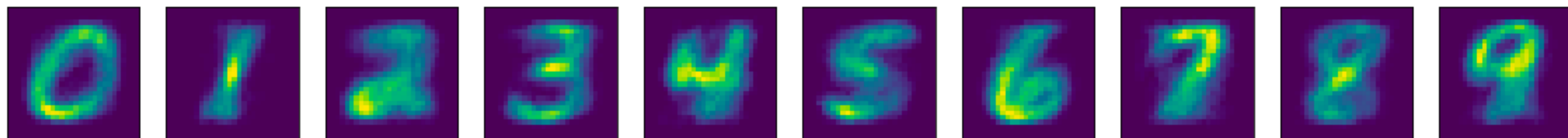


# What is the problem?

6. 원본 숫자 이미지, 숫자가 있어서는 안되는 영역, 숫자가 무조건 있는 영역

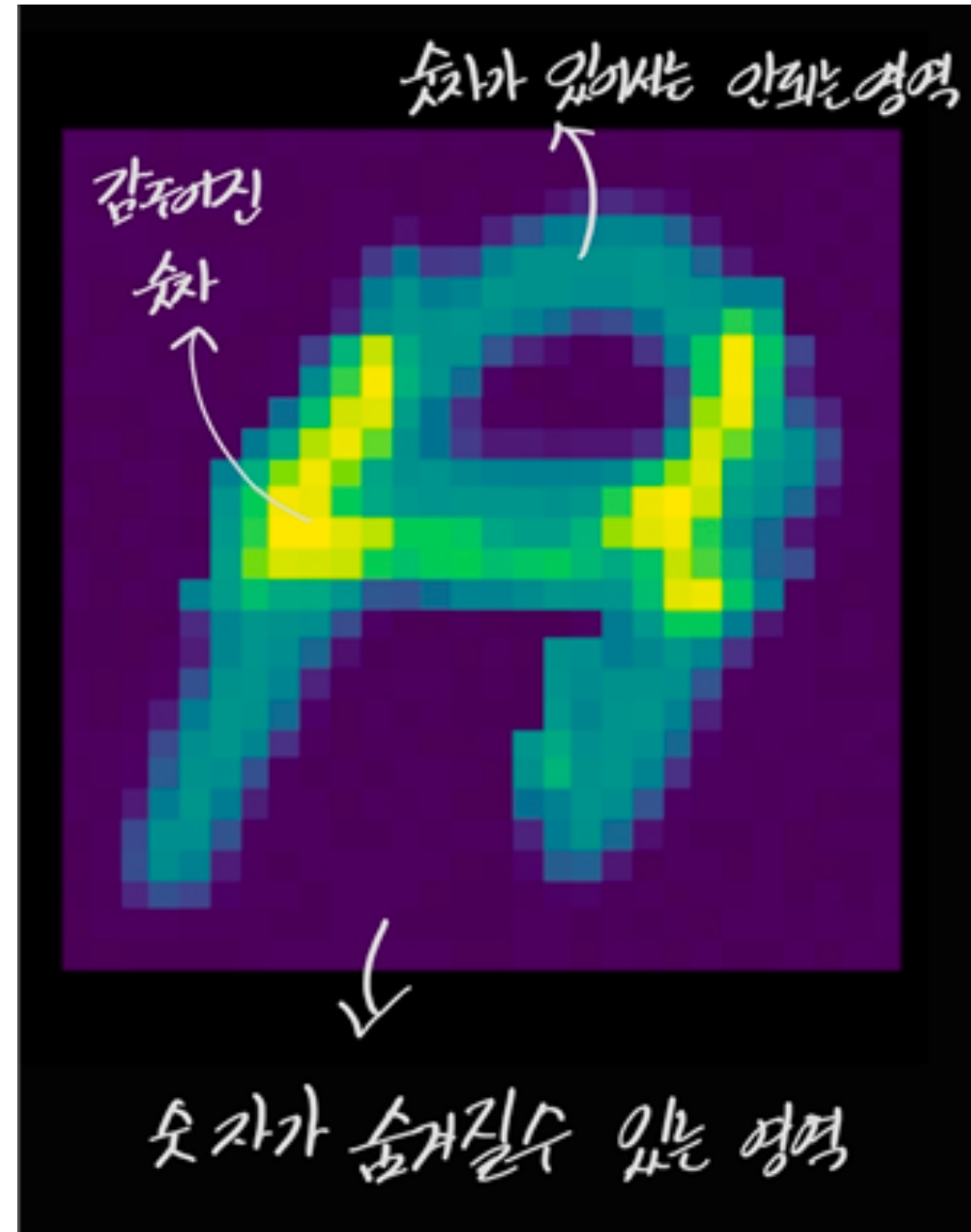


7. 전체 Train Data에서 보이는 숫자 부분의 모든 이미지를 겹친 이미지



# Why is that a problem?

## 1. 해당 문제를 왜 진행 하였는가?



- 숫자의 일부분 정보를 가지고 전체 숫자에 대한 정보를 알아내는 것이 중요 포인트!
- 이와 관련된 숨겨진 영역을 유추해야하는 문제에 대해서 사용 가능한 접근법이라고 생각하였습니다.

# Where did it proceed?

## 1. Google Colaboratory

Tue Aug 24 12:01:04 2021

```
+-----+
| NVIDIA-SMI 470.57.02   Driver Version: 460.32.03   CUDA Version: 11.2   |
+-----+-----+-----+
```

```
| GPU Name          Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|                               | MIG M. |
+=====+=====+=====+=====+=====+=====+=====+=====+=====+
|  0 Tesla P100-PCIE...  Off   | 00000000:00:04.0 Off |                    0 |
|N/A   36C   P0   26W / 250W |  0MiB / 16280MiB |   0%    Default |
|                               |      N/A |
+-----+-----+-----+-----+
```

```
+-----+
| Processes:                                     |
| GPU  GI  CI       PID   Type   Process name                      GPU Memory |
|   ID ID                  Usage   |
+=====+=====+=====+=====+=====+=====+=====+=====+=====+
| No running processes found                                     |
+-----+
```

# How to solve the problem

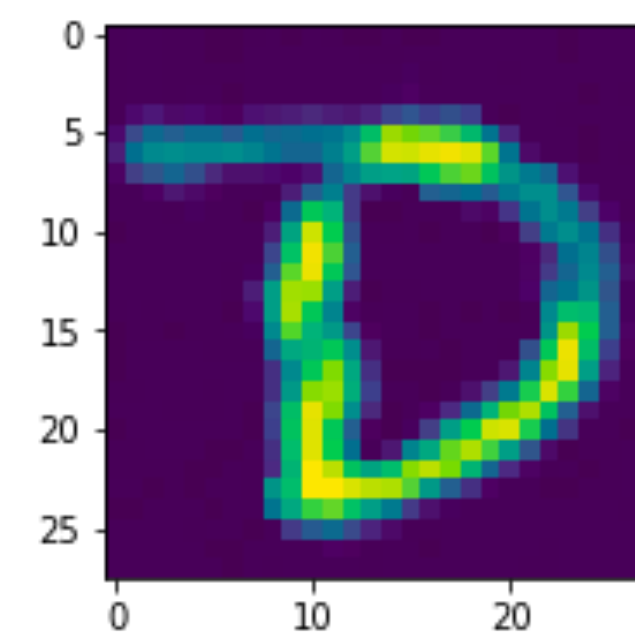
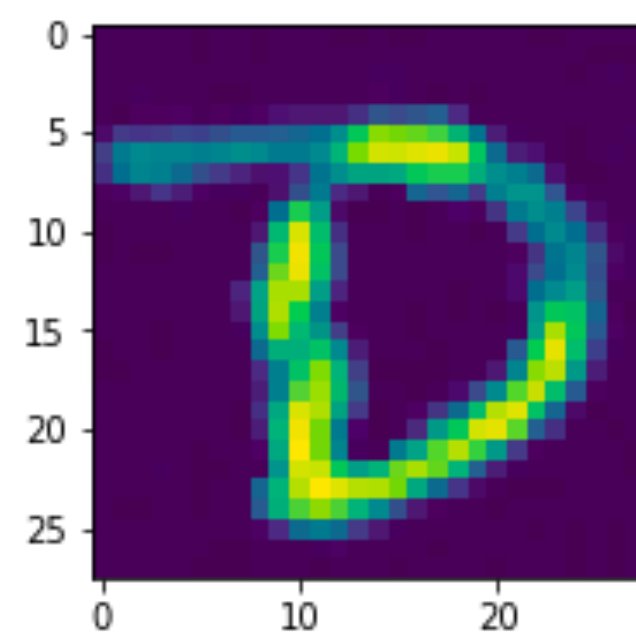
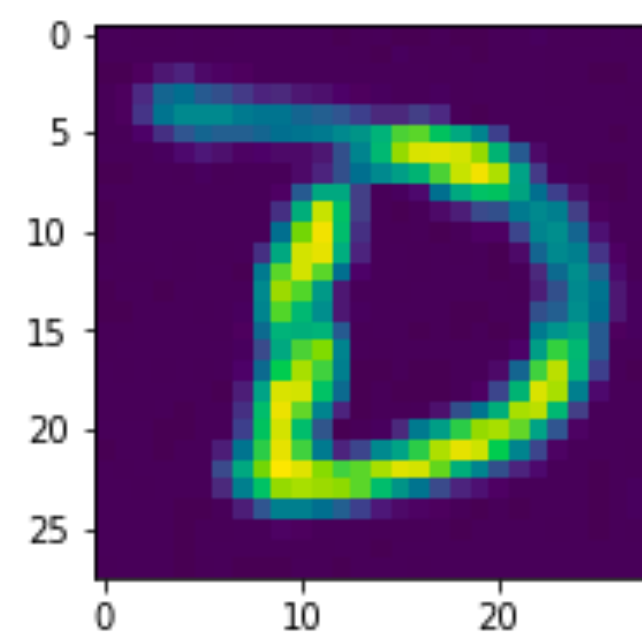
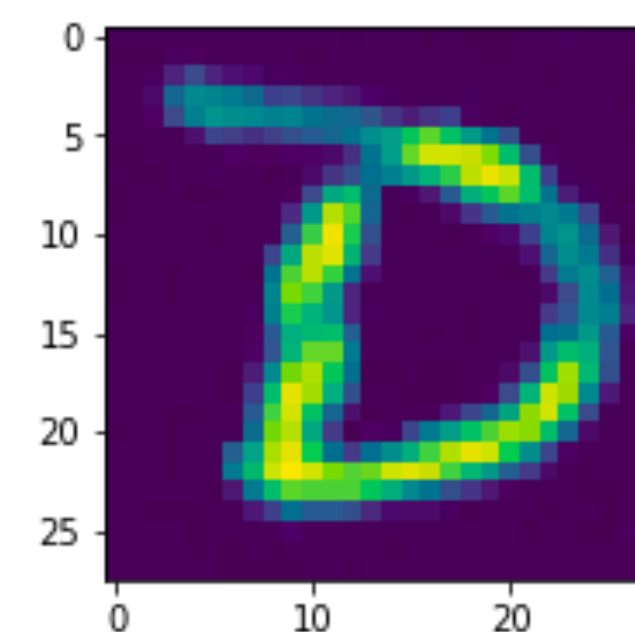
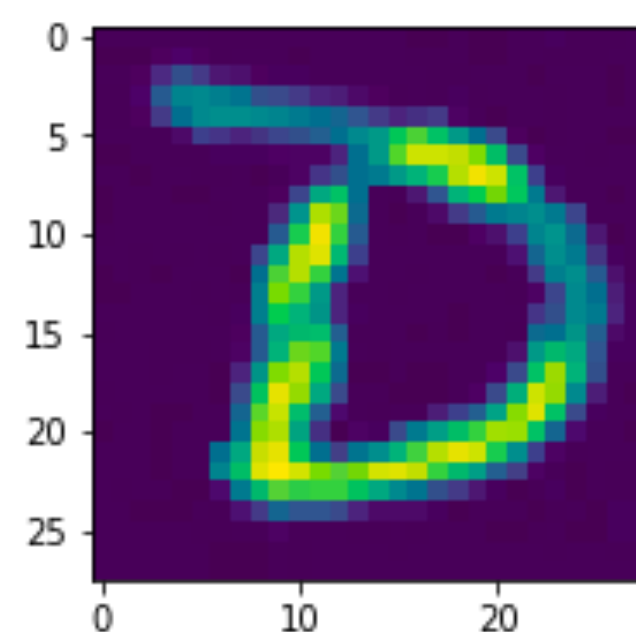
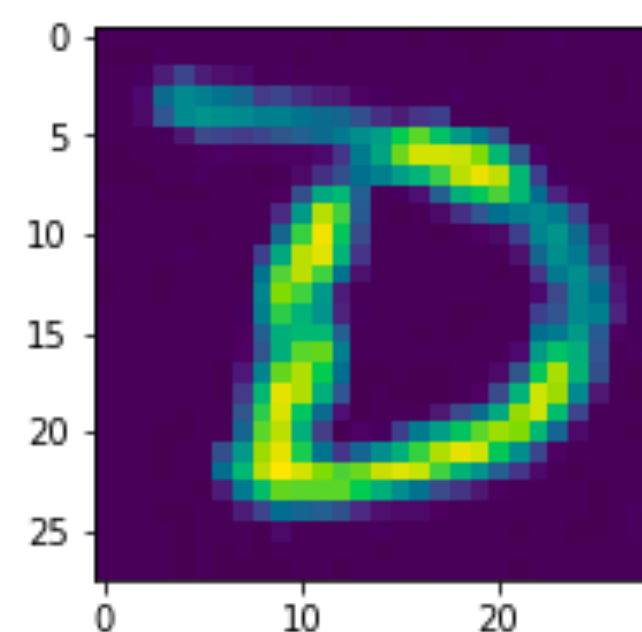
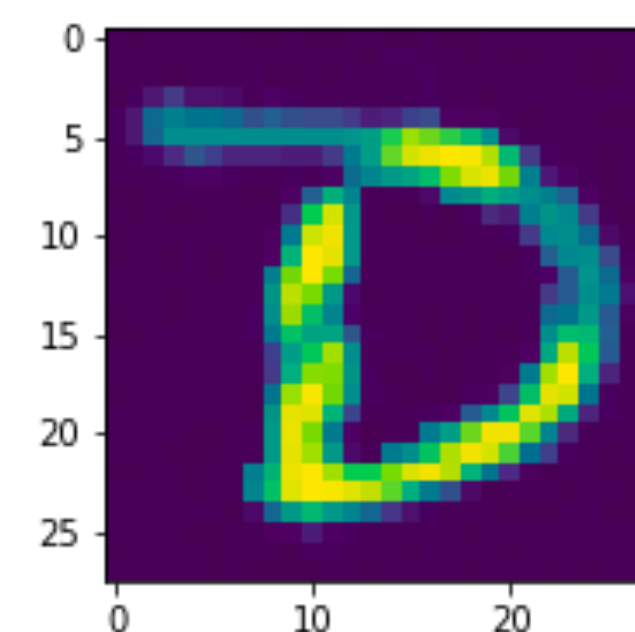
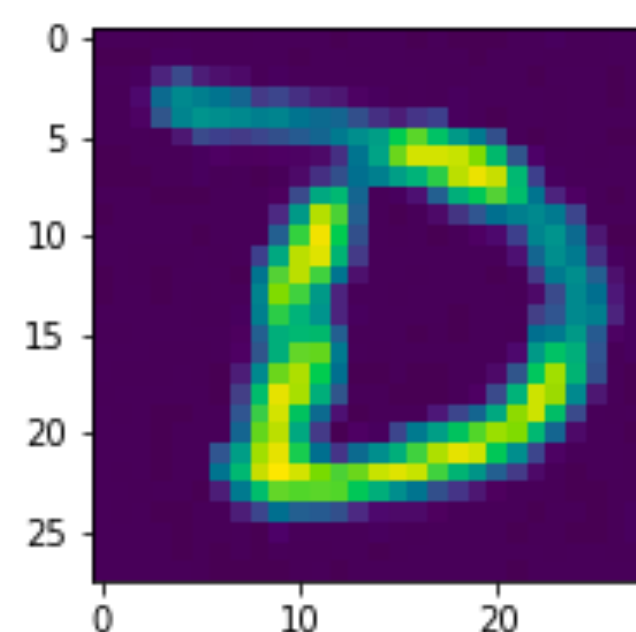
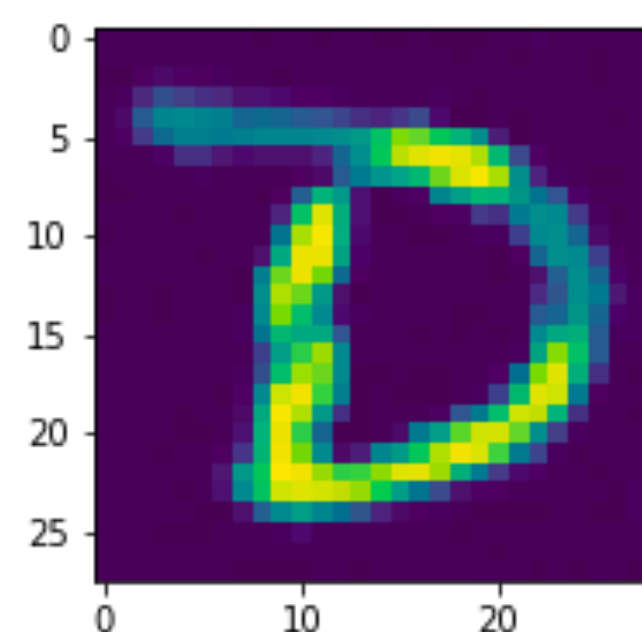
## 1. 기존 Image Classification model들을 사용해보자.

- Train data set 2048개 > 학습데이터(1642개), 검증데이터(406개)로 나누어서 학습
- 2048개로 20480개를 유추해야 하기에 Train Data set의 부족을 해결하기 위해서 ImageDataGenerator을 통해 Data Argmentation을 진행하여 학습에는 Train image = 52544, Validataion image = 12992가 사용되었습니다.
- ImageDataGenerator( rescale=1./255, validation\_split=0.2, rotation\_range=10, width\_shift\_range=0.1, height\_shift\_range=0.1)
- Batch\_size = 32 (dafault)
- optimizer = Adam(lr=0.002, epsilon=None)
- epochs = 500



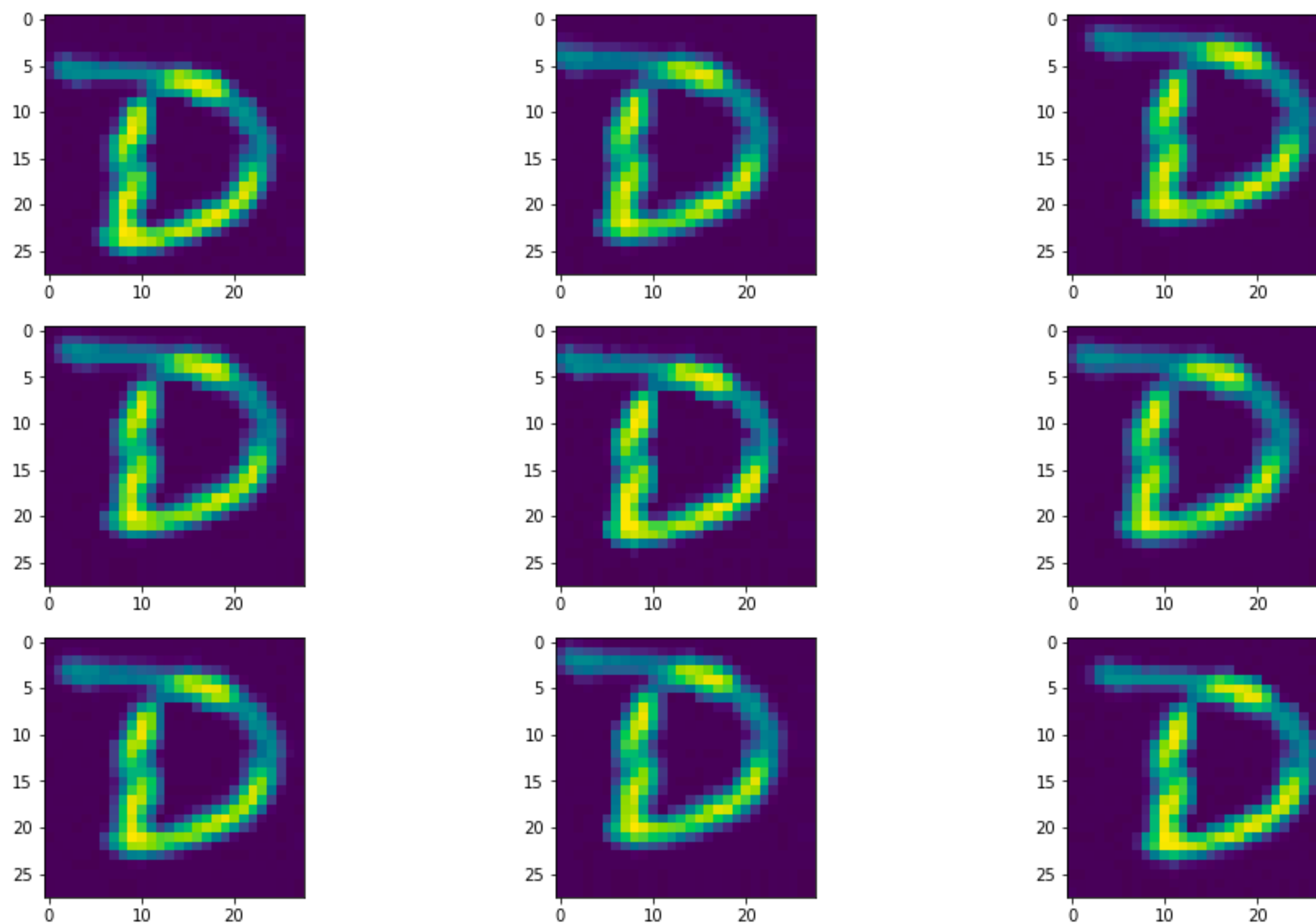
# How to solve the problem

1.1 rotation\_range=10



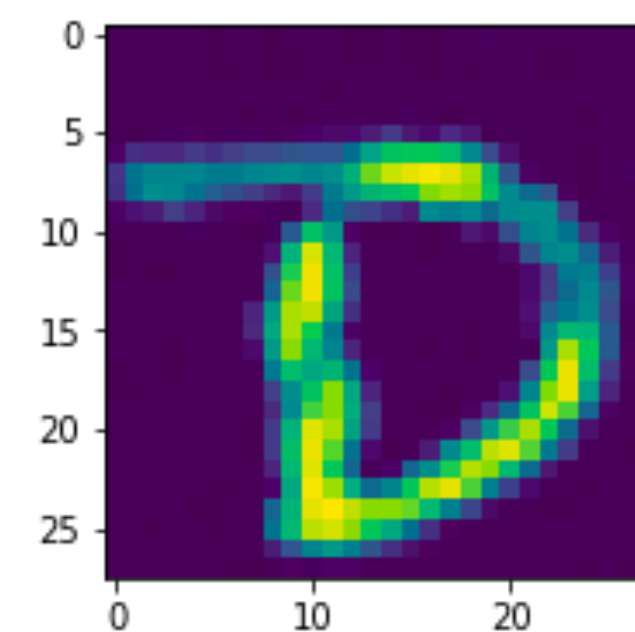
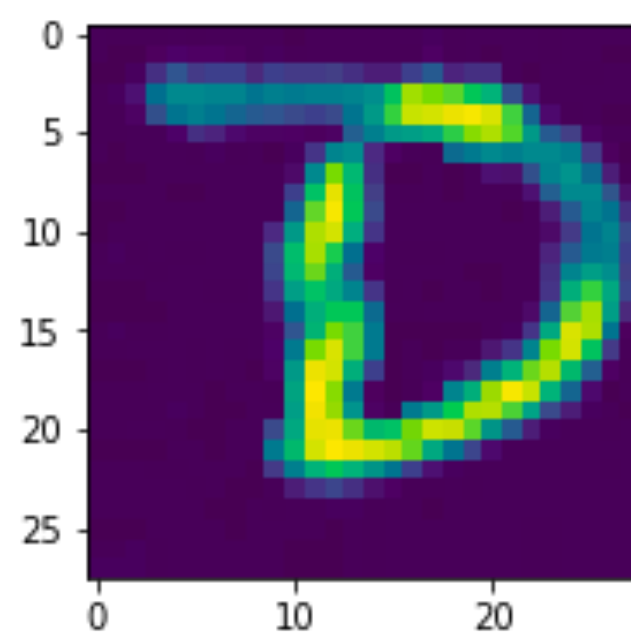
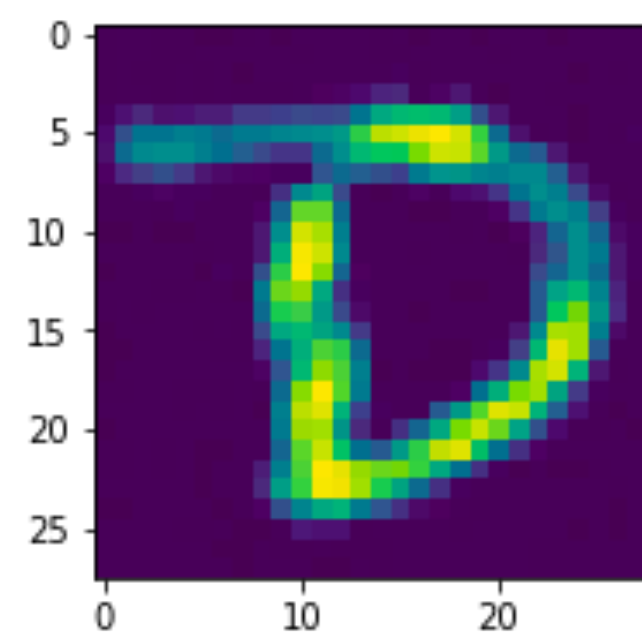
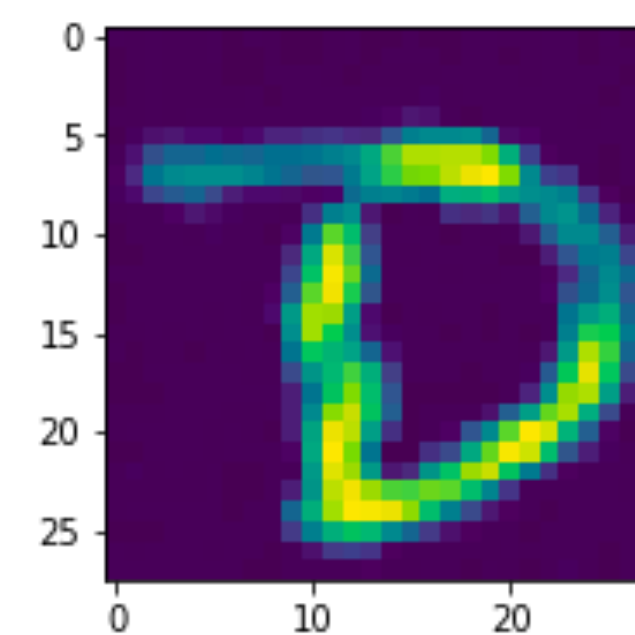
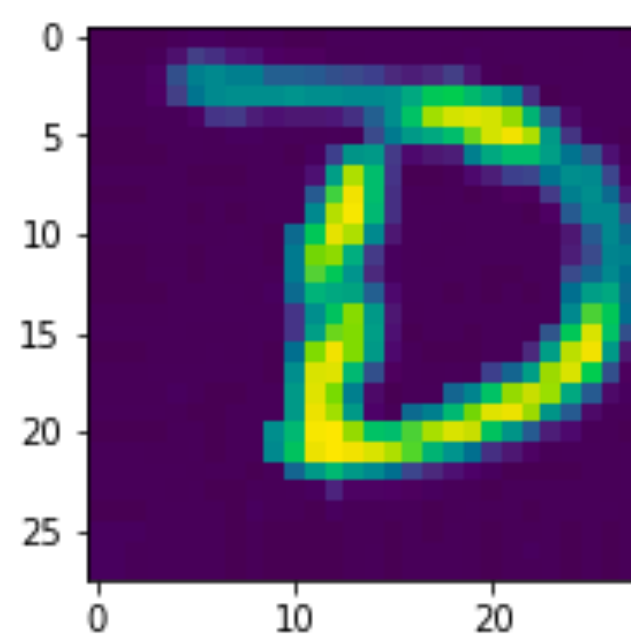
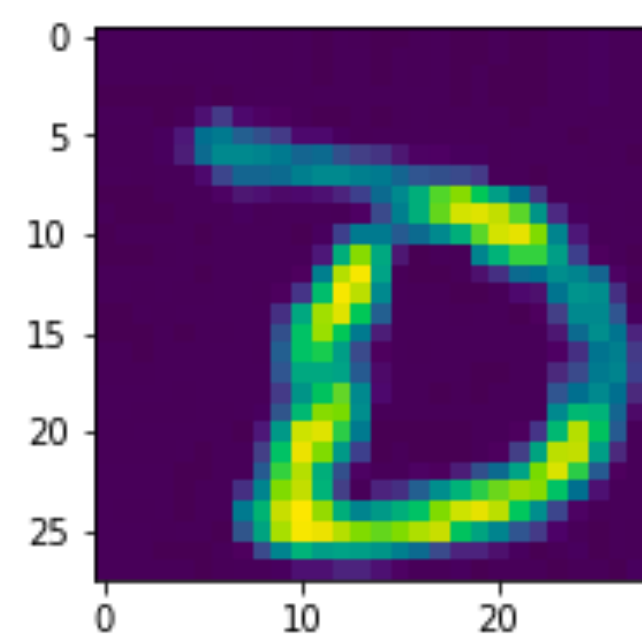
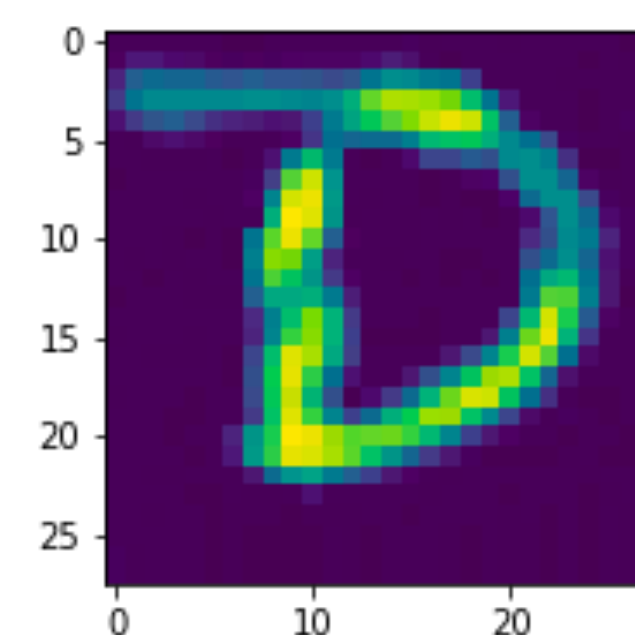
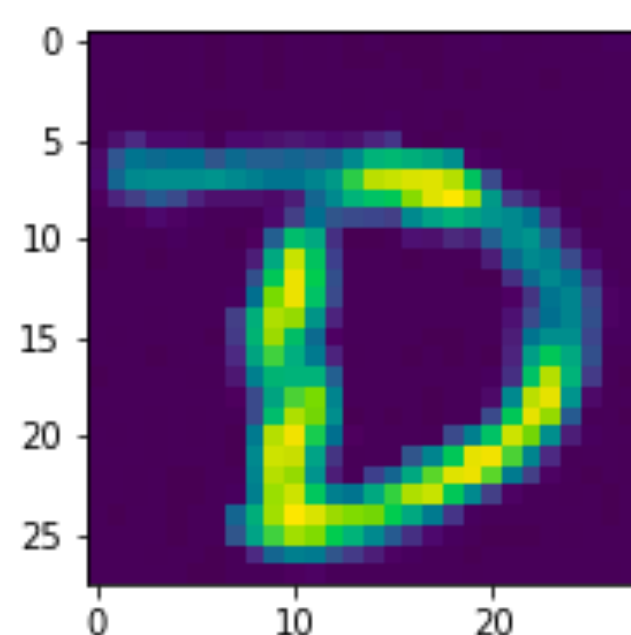
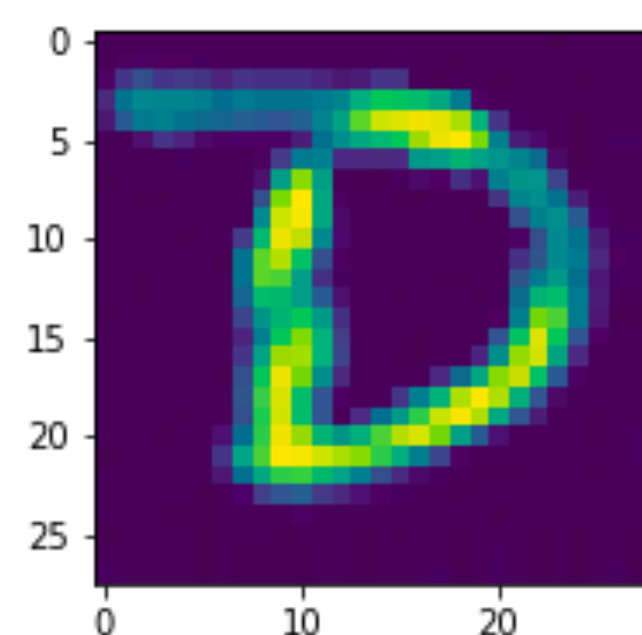
# How to solve the problem

1.2 width\_shift\_range=0.1, height\_shift\_range=0.1



# How to solve the problem

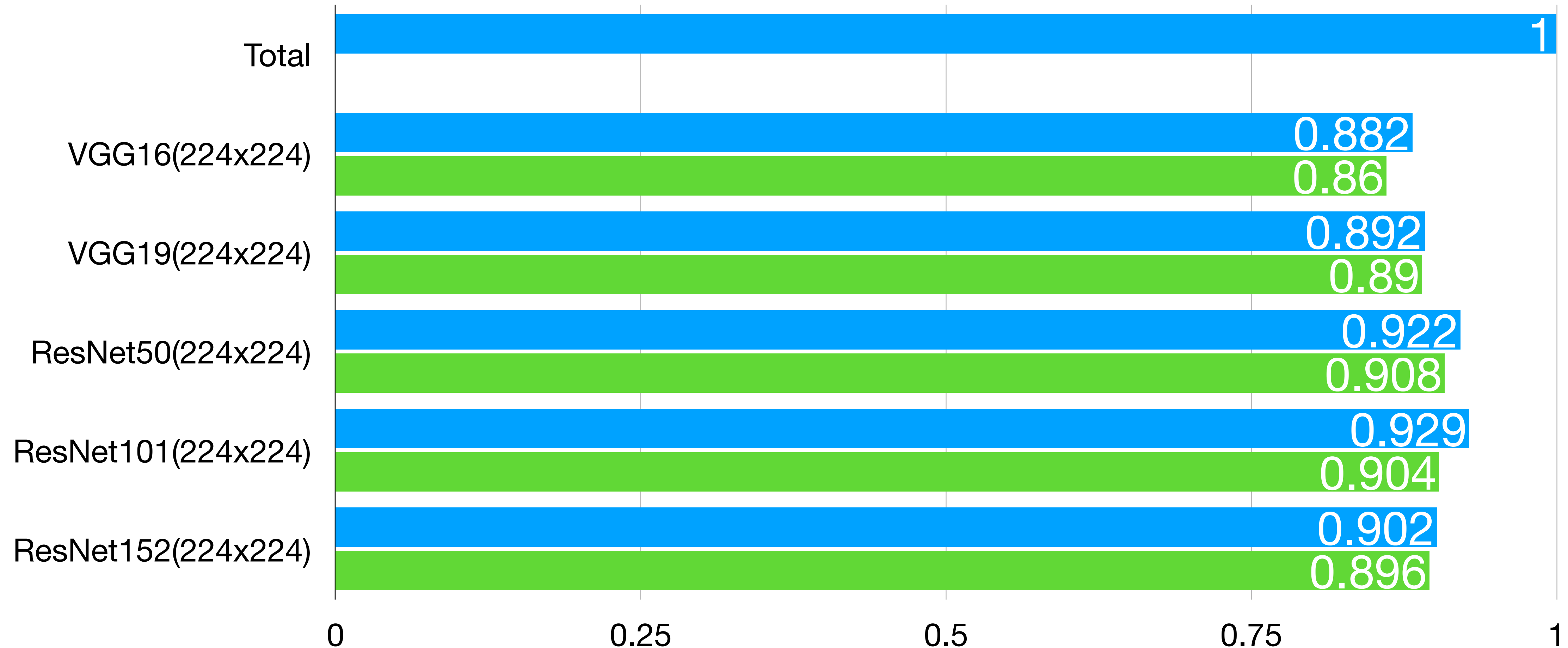
1.3 rotation\_range=10, width\_shift\_range=0.1, height\_shift\_range=0.1





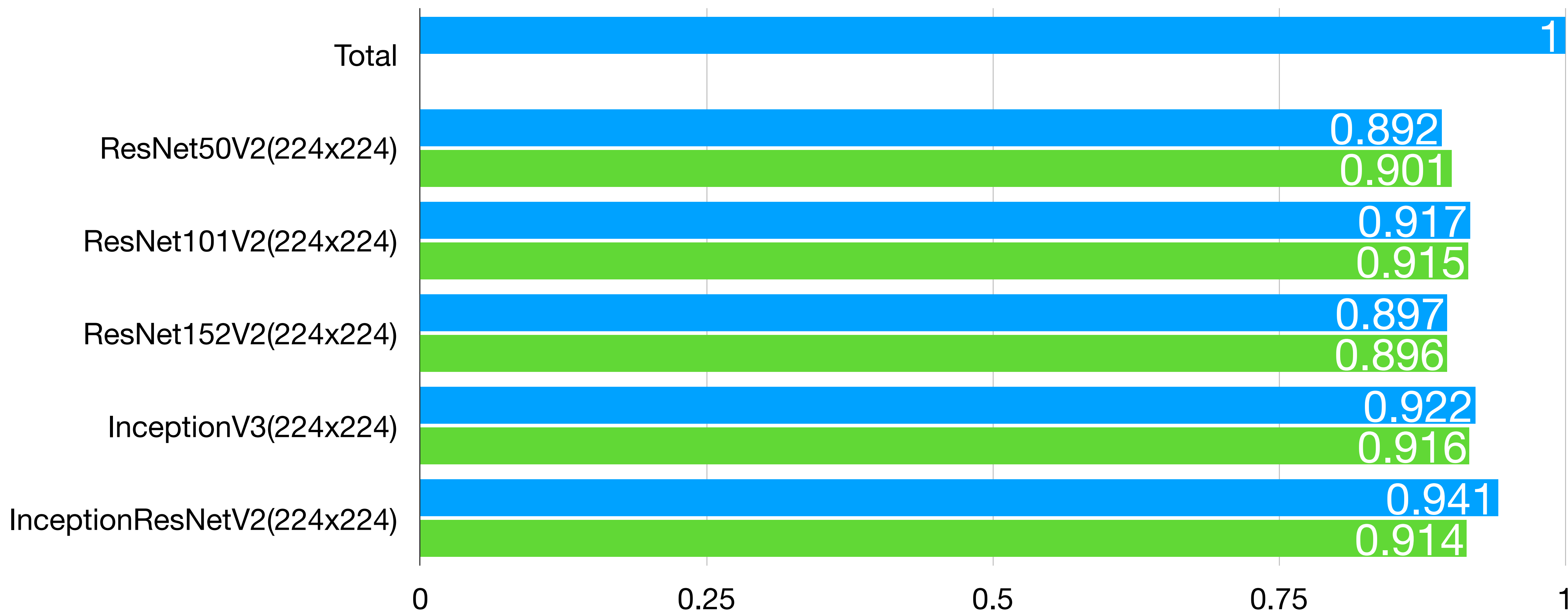
# How to solve the problem

## 2. 학습된 개별 모델들의 정확도.



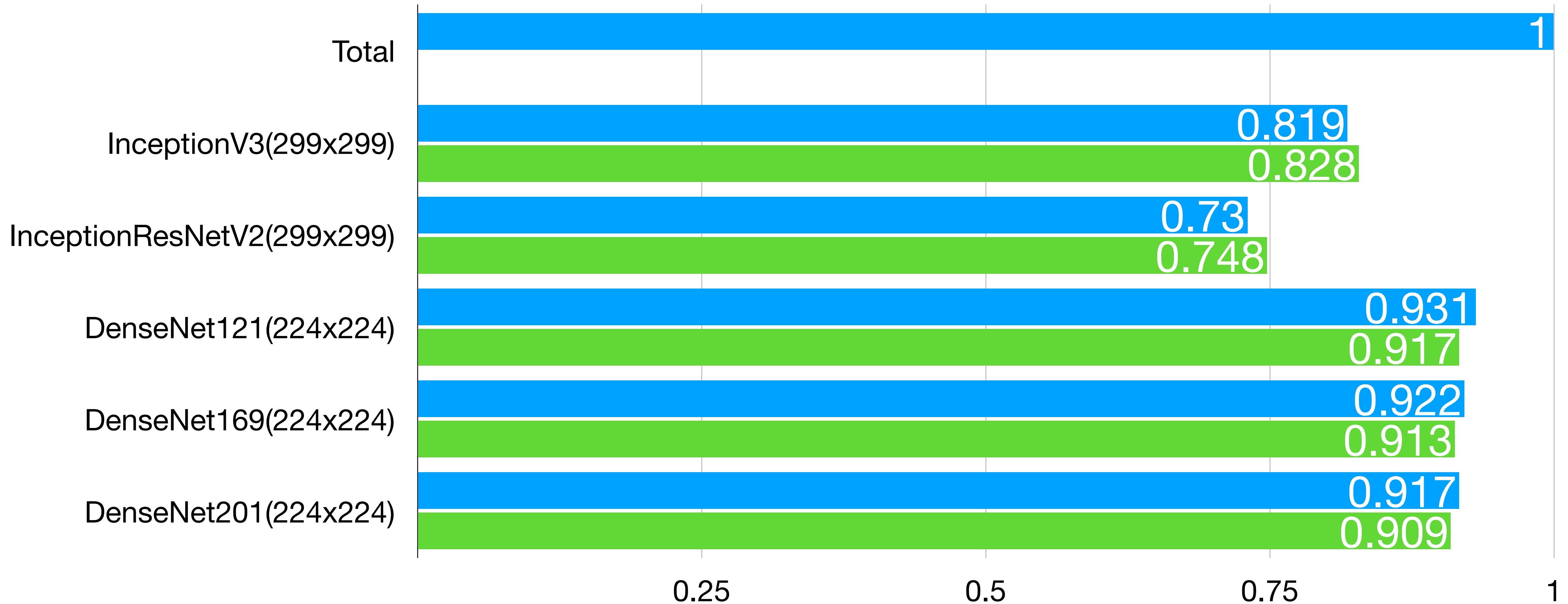
# How to solve the problem

## 2. 학습된 개별 모델들의 정확도.



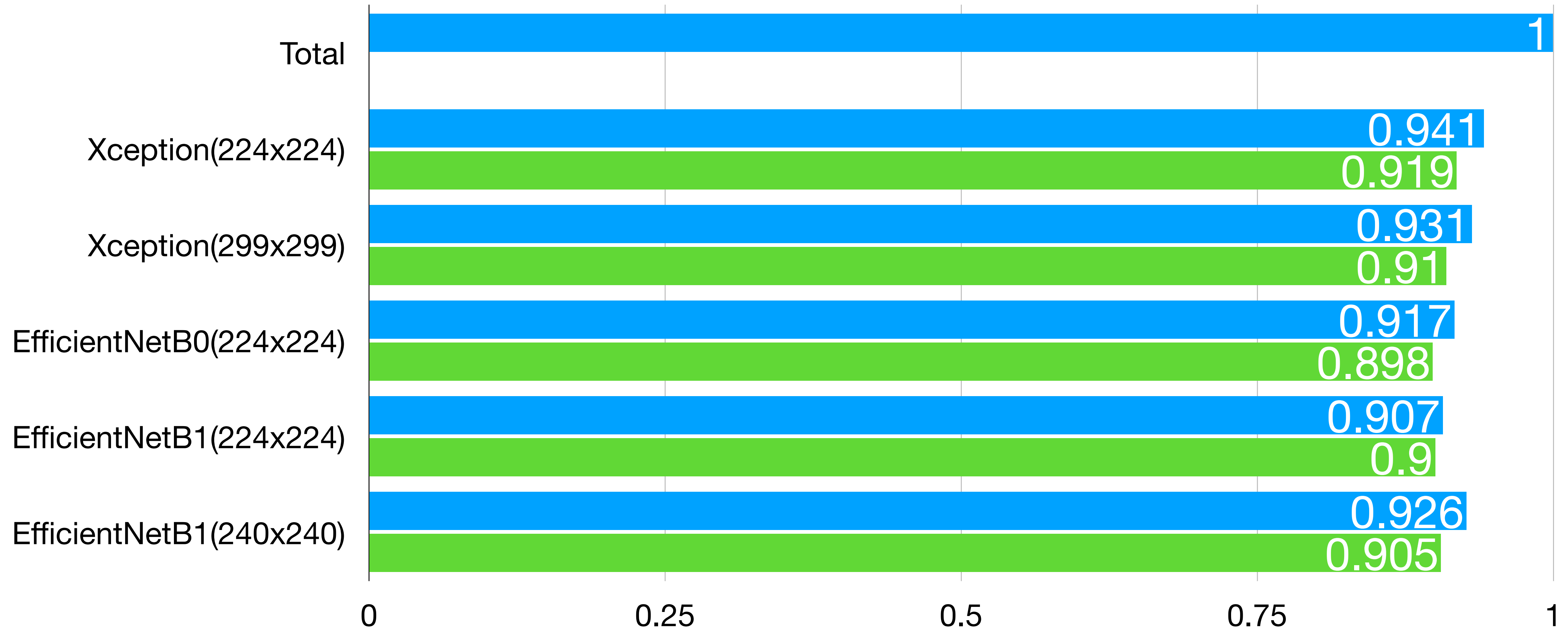
# How to solve the problem

## 2. 학습된 개별 모델들의 정확도.



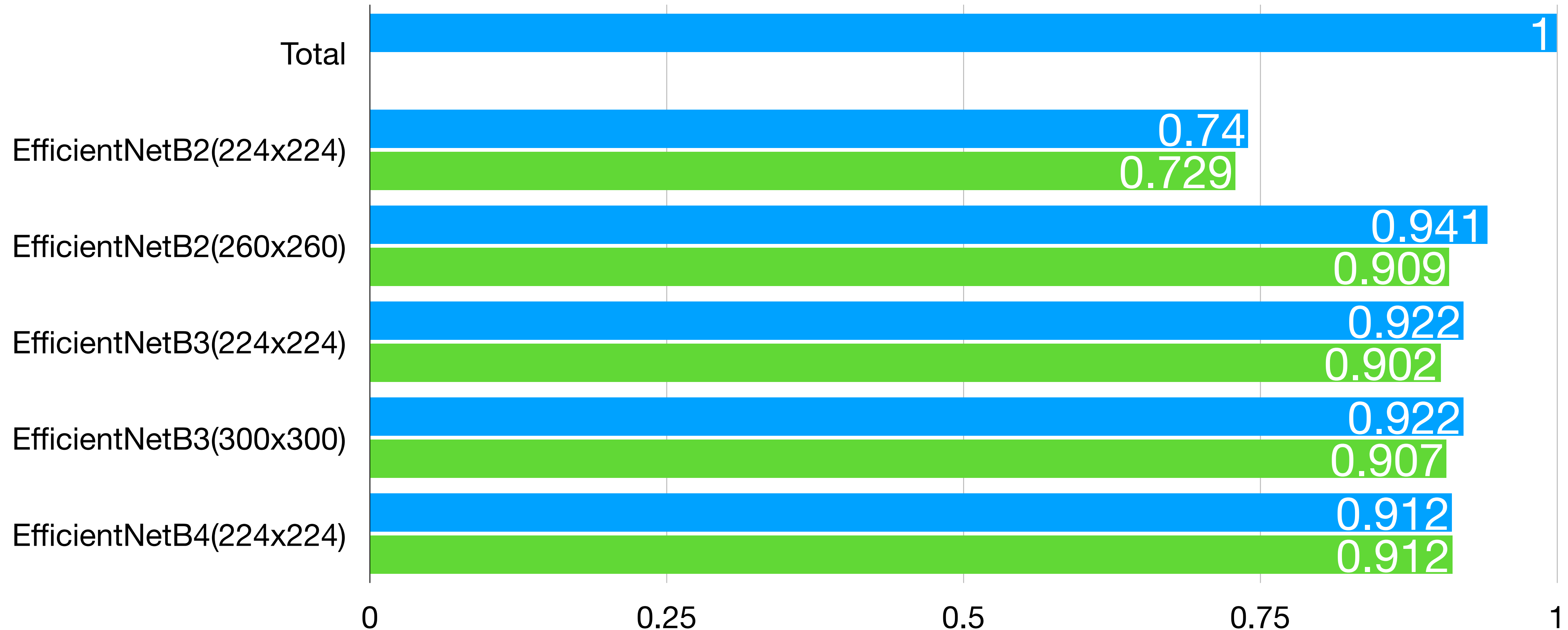
# How to solve the problem

## 2. 학습된 개별 모델들의 정확도.



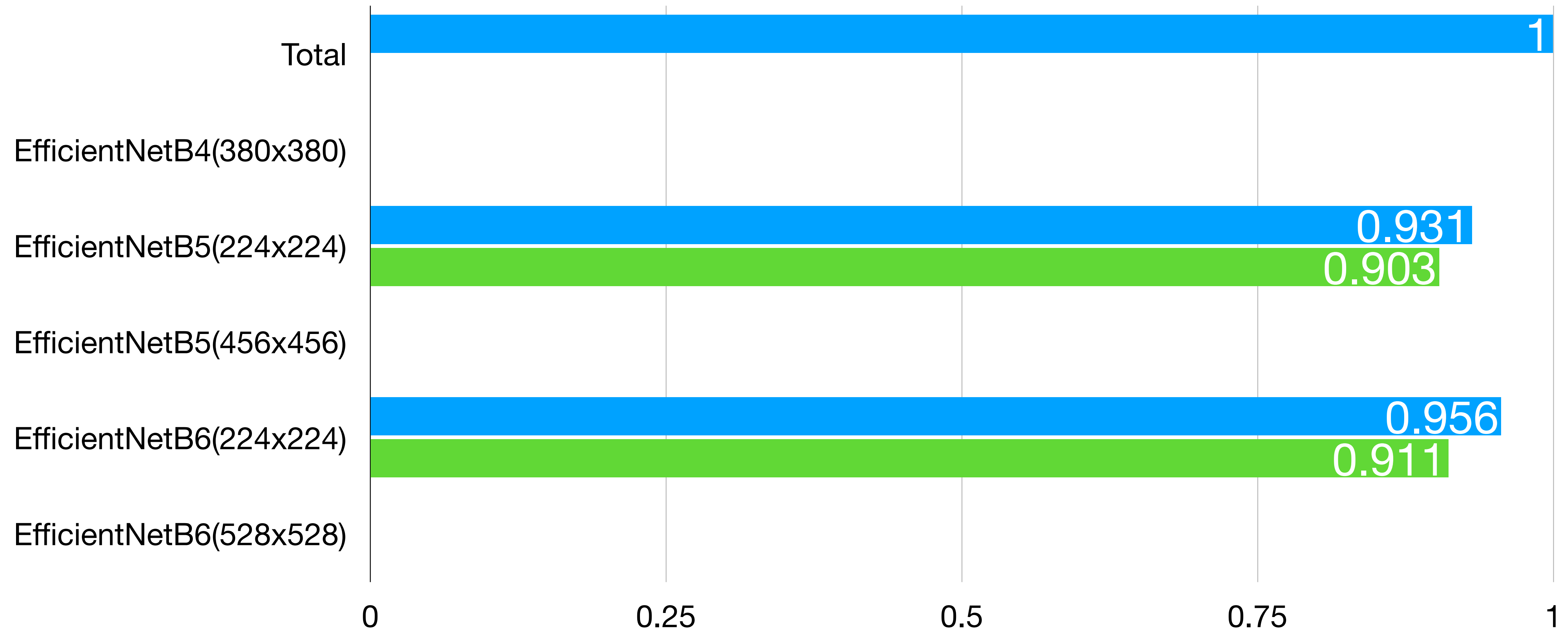
# How to solve the problem

## 2. 학습된 개별 모델들의 정확도.



# How to solve the problem

## 2. 학습된 개별 모델들의 정확도.



# How to solve the problem

## 2. 학습된 개별 모델들의 정확도.

- Model Default Input size와 Input size가 같은 모델의 경우 가장 정확도가 높은 모델은 DenseNet121이었으며, 해당 모델을 기준으로 Parameter optimization(ImageDataGenerator), Optimizer optimization을 진행한 다음 최적을 찾은 후 재학습을 진행할 예정입니다.
- EfficientNetB4(380x380), EfficientNetB5(456x456), EfficientNetB6(528x528), EfficientNetB7(224x224), EfficientNetB7(600x600)의 경우, Colab pro GPU memory 부족으로 인해서 학습이 불가하였습니다.

# How to solve the problem

## 3. Voting ensemble?

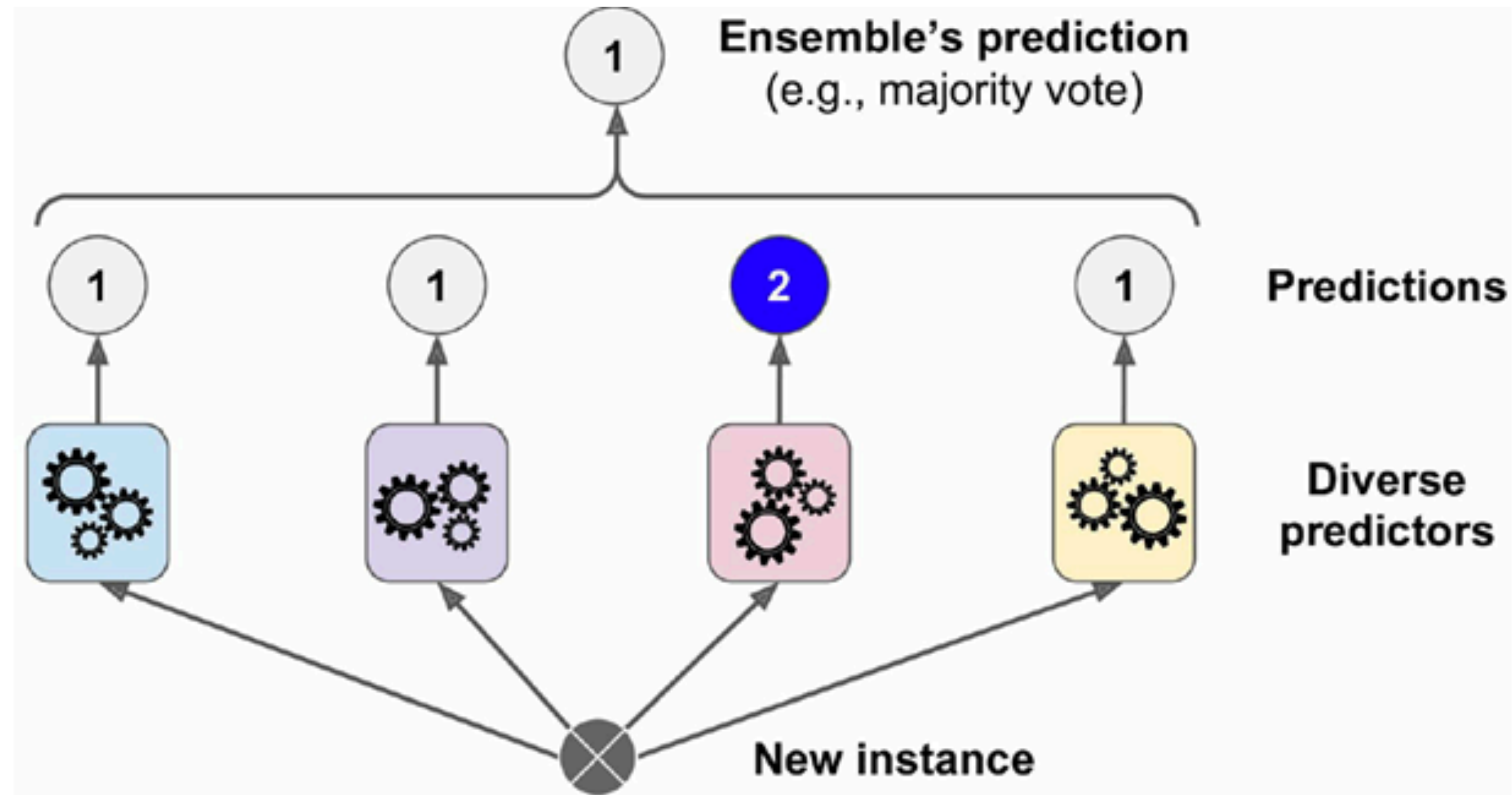


Figure 7-2. Hard voting classifier predictions



# How to solve the problem

## 3. 학습된 개별 모델을 사용한 Voting ensemble.

- 11 model, 25 model Voting ensemble result
- 11개, 25개의 모델의 비교를 진행할 때 VGG16, VGG19 model running error가 있어서 Voting ensemble을 진행하였습니다.
- Individual model(Before parameter optimization)을 통해서 개별적으로 학습된 모델을 이용하여서 Voting ensemble 된 결과 입니다.
- EfficientNetB4(380x380), EfficientNetB5(456x456), EfficientNetB6(528x528), EfficientNetB7(224x224), EfficientNetB7(600x600)의 경우, Colab pro GPU memory 부족으로 인해서 학습이 불가하였습니다.

# How to solve the problem

## 3. 학습된 개별 모델을 사용한 Voting ensemble. ( 11 models )

Model	Validation accuracy	default Input size	Input size
ResNet50	0.92857	224x224	224x224
ResNet101	0.93596	224x224	224x224
ResNet152	0.92857	224x224	224x224
ResNet50V2	0.93350	224x224	224x224
ResNet101V2	0.92118	224x224	224x224
ResNet152V2	0.93596	224x224	224x224
InceptionV3	0.94581	299x299	224x224
InceptionResNetV2	0.93103	299x299	224x224
DenseNet121	0.93842	224x224	224x224
DenseNet169	0.93596	224x224	224x224
DenseNet201	0.93103	224x224	224x224

- Private accuracy - 0.94607, Private accuracy - 0.93317.

# How to solve the problem

## 3. 학습된 개별 모델을 사용한 Voting ensemble. ( 25 models )

Model	Private accuracy	Default input size	Input size
VGG16	X	224x224	224x224
VGG19	X	224x224	224x224
ResNet50	0.90816	224x224	224x224
ResNet101	0.90377	224x224	224x224
ResNet152	0.89568	224x224	224x224
ResNet50V2	0.90076	224x224	224x224
ResNet101V2	0.91512	224x224	224x224
ResNet152V2	0.89647	224x224	224x224
InceptionV3	0.91640	299x299	224x224
InceptionResNetV2	0.91408	299x299	224x224
InceptionV3	0.82831	299x299	299x299
InceptionResNetV2	0.74758	299x299	299x299
DenseNet121	0.91689	224x224	224x224

# How to solve the problem

## 3. 학습된 개별 모델을 사용한 Voting ensemble. ( 25 models )

Model	Private accuracy	Default input size	Input size
DenseNet169	0.91285	224x224	224x224
DenseNet201	0.90940	224x224	224x224
Xception	0.91862	299x299	224x224
Xception	0.91009	299x299	299x299
EfficientNetB0	0.89830	224x224	224x224
EfficientNetB1	0.90032	240x240	224x224
EfficientNetB1	0.90540	240x240	240x240
EfficientNetB2	0.72913	260x260	224x224
EfficientNetB2	0.90930	260x260	260x260
EfficientNetB3	0.90185	300x300	224x224
EfficientNetB3	0.90693	300x300	300x300
EfficientNetB4	0.91196	380x380	224x224
EfficientNetB4	X	380x380	380x380

# How to solve the problem

## 3. 학습된 개별 모델을 사용한 Voting ensemble. ( 25 models )

Model	Private accuracy	Default input size	Input size
EfficientNetB5	0.90338	456x456	224x224
EfficientNetB5	X	456x456	456x456
EfficientNetB6	0.91122	528x528	224x224
EfficientNetB6	X	528x528	528x528
EfficientNetB7	X	600x600	224x224
EfficientNetB7	X	600x600	600x600

- Private accuracy - 0.94607, Private accuracy - 0.93386.

# What's left?

## 1. Parameter optimization(ImageDataGenerator)

- model = DenseNet121 (input size = 224 x 224)
- Batch\_size = 32 (dafault)
- optimizer = Adam(lr=0.002, epsilon=None)
- epochs = 500
- ImageDataGenerator(
  - rescale = 1./255,
  - validation\_split = 0.X,
  - rotation\_range = X,
  - width\_shift\_range = 0.X,
  - height\_shift\_range = 0.X )

# What's left?

## 2. Optimizer optimization

- class Adadelta: Optimizer that implements the Adadelta algorithm.
- class Adagrad: Optimizer that implements the Adagrad algorithm.
- class Adam: Optimizer that implements the Adam algorithm.
- class Adamax: Optimizer that implements the Adamax algorithm.
- class Ftrl: Optimizer that implements the FTRL algorithm.
- class Nadam: Optimizer that implements the NAdam algorithm.
- class Optimizer: Base class for Keras optimizers.
- class RMSprop: Optimizer that implements the RMSprop algorithm.
- class SGD: Gradient descent (with momentum) optimizer.

# What's left?

## 3. Data input size tendency

Input size	acc(one)	acc(two)	acc(thr)	acc(four)	acc(five)	Avg
28x28						
32x32						
64x64						
128x128						
224x224						
256x256						
299x299						



# What's left?

## 4. Voting ensemble

- Parameter optimization(ImageDataGenerator), Optimizer optimization, Data input size tendency을 진행하여 결정된 최적의 값을 사용하여서 개별적인 모델들의 학습을 진행하여 다시 한번 학습된 모델들의 Voting ensemble을 진행할 예정입니다.