SELF-LEARNING

이미지 노이즈 제거 클래스 분류기

>> 목차

1 설계 과제 소개

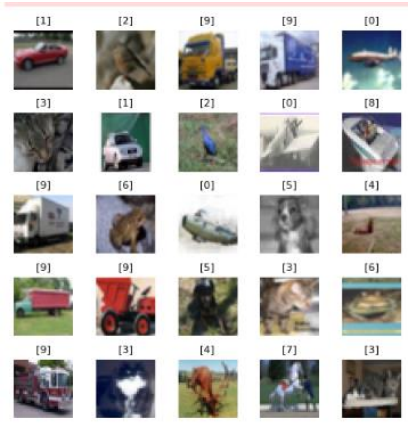
2 오토 인코더 - 이미지 노이즈 제거

3 합성곱 신경망(CNN) - 이미지 분류기

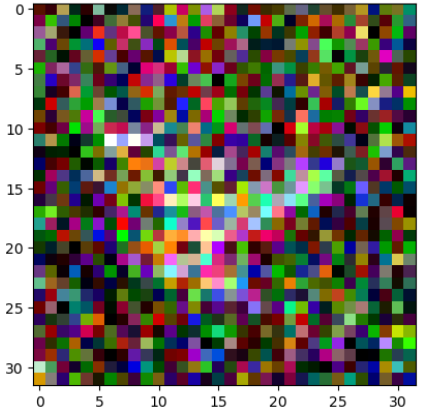
4 프로젝트 진행 현황

Part 1 >> 설계 과제 소개

노이즈가 포함된 이미지를 분류하는 모델!



1. 원본 이미지



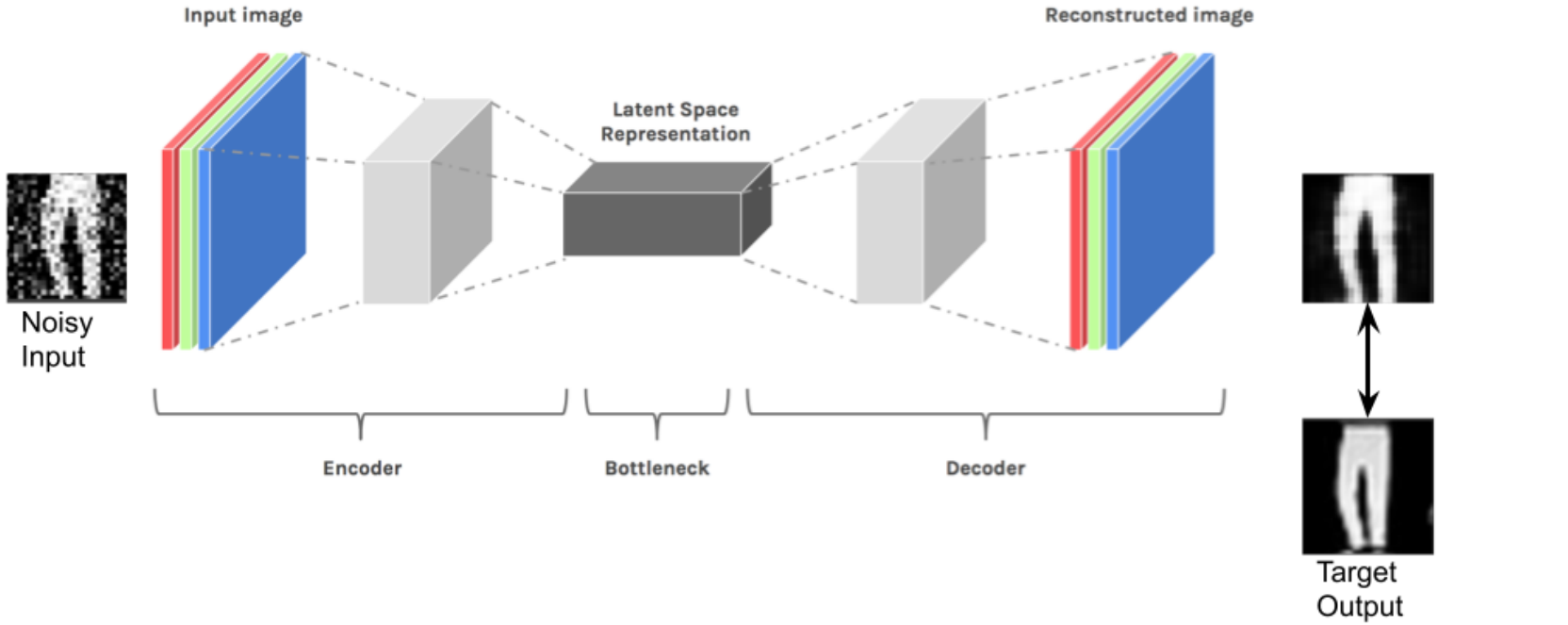
2. 노이즈 추가

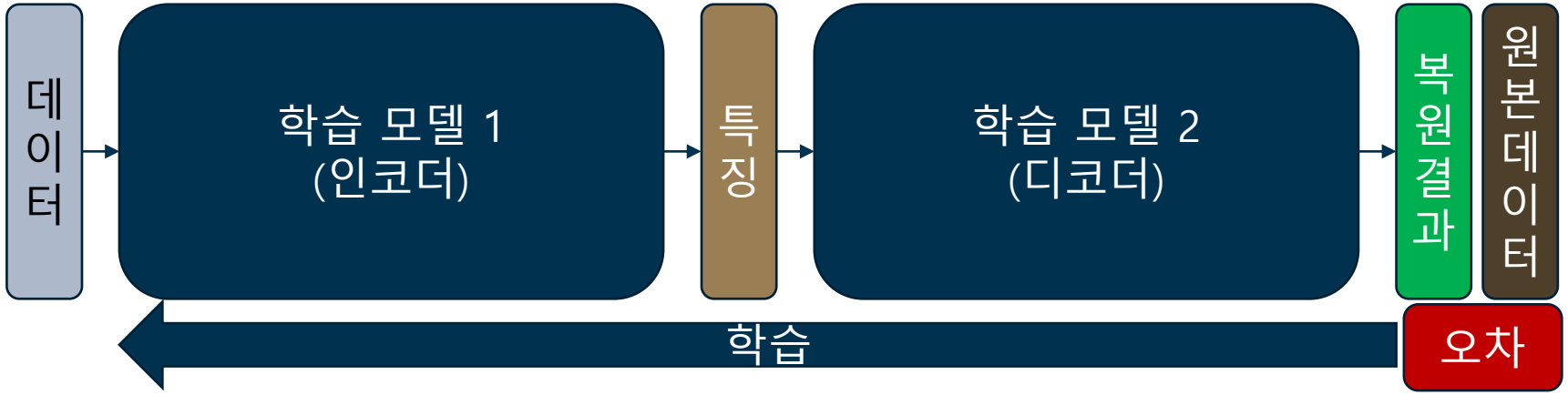
3. 오토인코더 : 노이즈 제거

어떤 종류의 이미지인지 classification

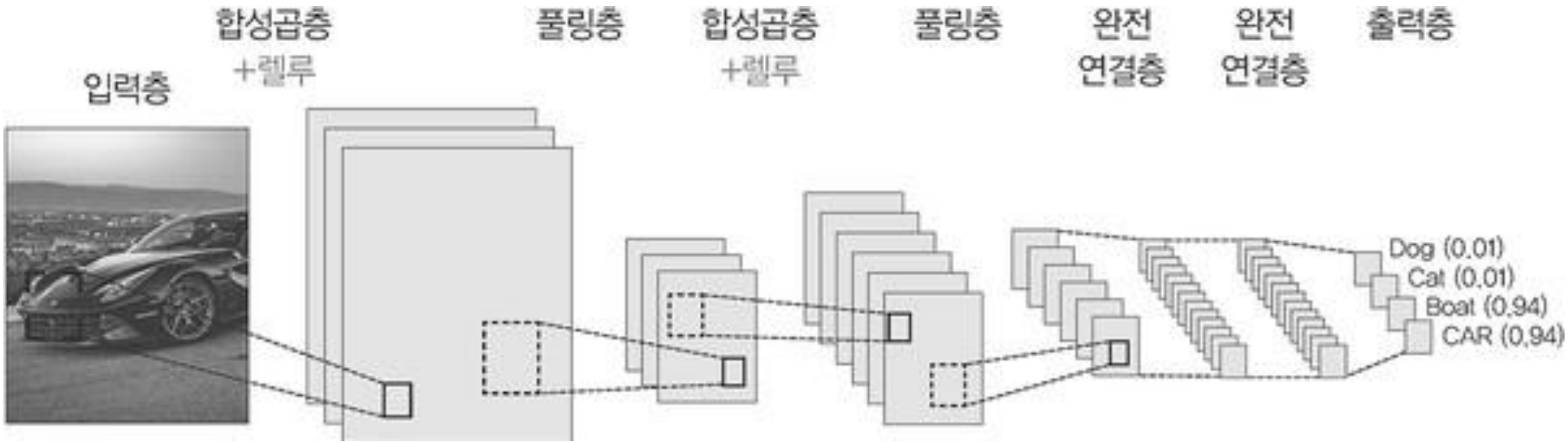
4. CNN을 통한 분류

Part 2 >> 오토인코더를 통한 이미지 노이즈 제거





Part 3 >> 합성곱 신경망(CNN) 을 통한 이미지 분류



Part 4 >> 프로젝트 진행 현황 (오토인코더)

```
# 이미지 분류기를 위해 다양한 카테고리를 가진 cifar10 데이터셋 사용
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.cifar10.load_data()
```

✓ 1.6s

➡ **이미지 분류를 위해 다양한 카테고리를 가진 이미지 dataset인 cifar10을 사용**
훈련용/검증용 데이터로 구분하였다.

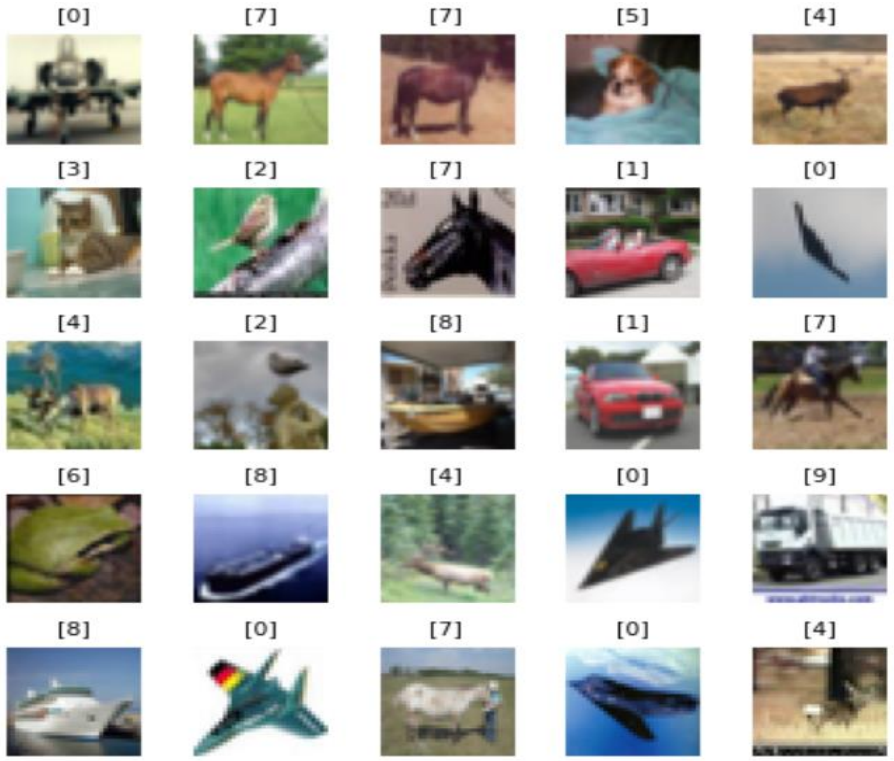
```
# subplots()함수 fig, axes 객체 리턴
fig, axes = plt.subplots(5, 5, figsize = (17,17))

# 15 * 15 행렬을 255크기의 배열로 바꿈
axes = axes.ravel()

# 훈련용 데이터셋 크기 반환
n_training = len(X_train)

# 0 ~ n_training 까지 중 랜덤한 인덱스의 데이터 이미지 출력
for i in np.arange(0, 5*5):
    # 랜덤한 인덱스 지정
    index = np.random.randint(0, n_training)
    # 인덱스에 해당하는 데이터 이미지 출력
    axes[i].imshow( X_train[index] )
    axes[i].set_title(y_train[index], fontsize = 8)
    axes[i].axis('off')

plt.subplots_adjust(hspace=0.4)
```



Part 4 >> 프로젝트 진행 현황 (오토인코더)

```
# 노이즈 강도 0.3
noise_factor = 0.3

# 노이즈 추가된 이미지 저장할 배열
noise_dataset = []

# 훈련용 데이터 노이즈 추가
for img in X_train:
  noisy_image = img + noise_factor * np.random.randn(*img.shape) # 이미지 픽셀에 노이즈 추가
  noisy_image = np.clip(noisy_image, 0., 1.) # 이미지 클리핑
  noise_dataset.append(noisy_image) # 노이즈 추가된 이미지 noise_dataset 배열에 추가

noise_dataset = np.array(noise_dataset)
```

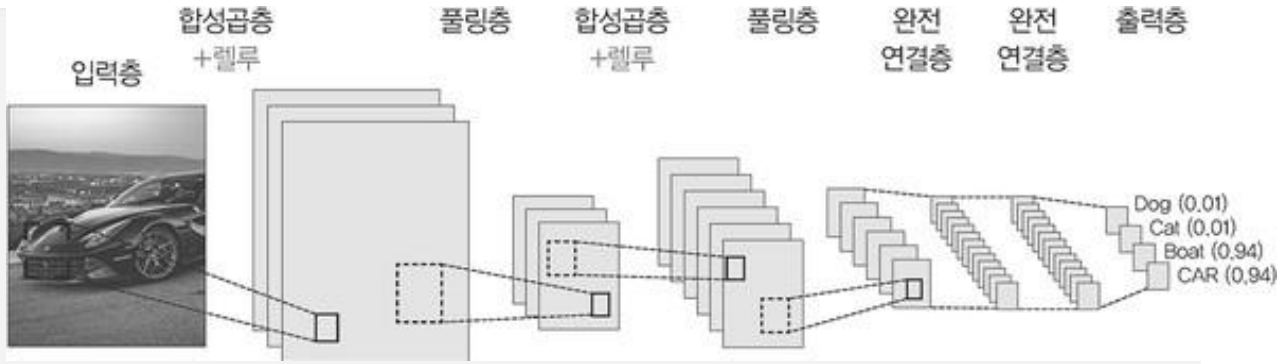
➡ **노이즈 강도 0.3 설정 후 각 이미지 데이터 픽셀에 노이즈 추가 (위 코드 훈련용 데이터)**
검증용 데이터에 대해서도 똑같이 노이즈 추가

Part 4 >> 프로젝트 진행 현황 (오토인코더)

```python
# 입력 이미지의 형태 저장
input_shape = X_train.shape[1:]

# 오토인코더 모델 생성
autoencoder = tf.keras.models.Sequential([
    ### 인코더 부분
    # 이미지 특성 추출
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=input_shape),
    # 2x2 최대 풀링을 사용하여 이미지 크기를 1/2배 (차원 축소 및 압축)
    tf.keras.layers.MaxPooling2D((2, 2), padding='same'),
    # 이미지 특성 추출
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', padding='same'),

    ### 디코더 부분
    # 2x2 업샘플링을 사용하여 이미지 크기를 2배 (차원 복원 및 복원)
    tf.keras.layers.UpSampling2D((2, 2)),
    # 이미지 특성 추출
    tf.keras.layers.Conv2D(3, (3, 3), activation='sigmoid', padding='same')
])
```



➡ **이미지 데이터 특성을 추출하기 위해 Conv2D 함수 이용**
MaxPooling2D()함수를 사용하여 차원 축소 및 압축을 시행
Upsampling2D() 함수를 이용하여 차원 복원 및 압축으로부터의 복원을 시행

Part 4 >> 프로젝트 진행 현황 (오토인코더)

```
# 데이터 훈련
autoencoder.fit(noise_dataset, X_train, epochs=10, batch_size=128, validation_data=(noise_test_set, X_test))
```

➡ **설계한 오토 인코더 모델을 통해 데이터 훈련을 fit() 함수를 통해 진행**
noise_dataset 입력에 대해 정답 레이블인 원본 데이터 X_train과 비교하여 학습
학습 반복 횟수 epochs=10

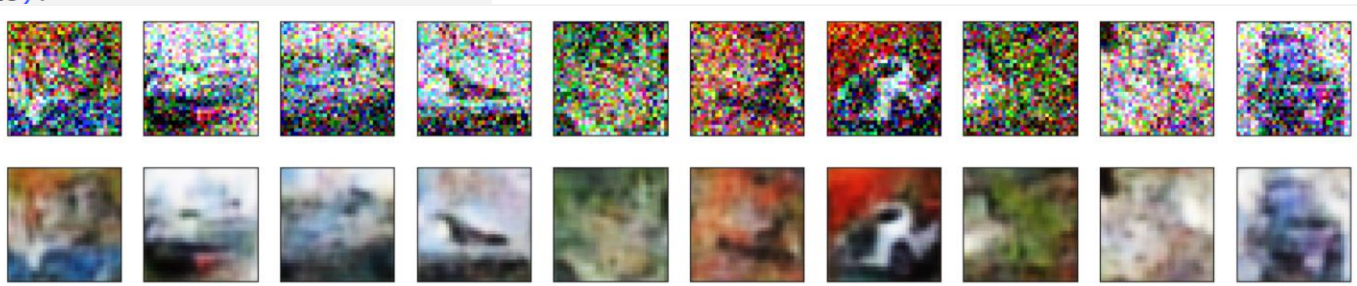
```
# 모델 성능 측정 (예측 정확도 파라미터)
evaluation = autoencoder.evaluate(noise_test_set, X_test)
print('Test Accuracy : {:.3f}'.format(evaluation))
```

➡ **정확도 파라미터로 모델 성능 측정**

```
313/313 [==============================] - 4s 14ms/step - loss: 0.5659
Test Accuracy : 0.566
```

```
# 예측 결과 출력
fig, axes = plt.subplots(nrows=2, ncols=10, sharex=True, sharey=True, figsize=(20,4))
for images, row in zip([noise_test_set[:10], predicted], axes):
    for img, ax in zip(images, row):
        ax.imshow(img.reshape((32, 32, 3)), cmap='Greys_r')
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)
```

➡ **잡음제거(예측) 결과 출력**



QnA

팀 셀프러닝
20190895 김찬영
20190954 허진환