TensorFlow Keras를 활용한 컨볼루션 신경망 실습하기

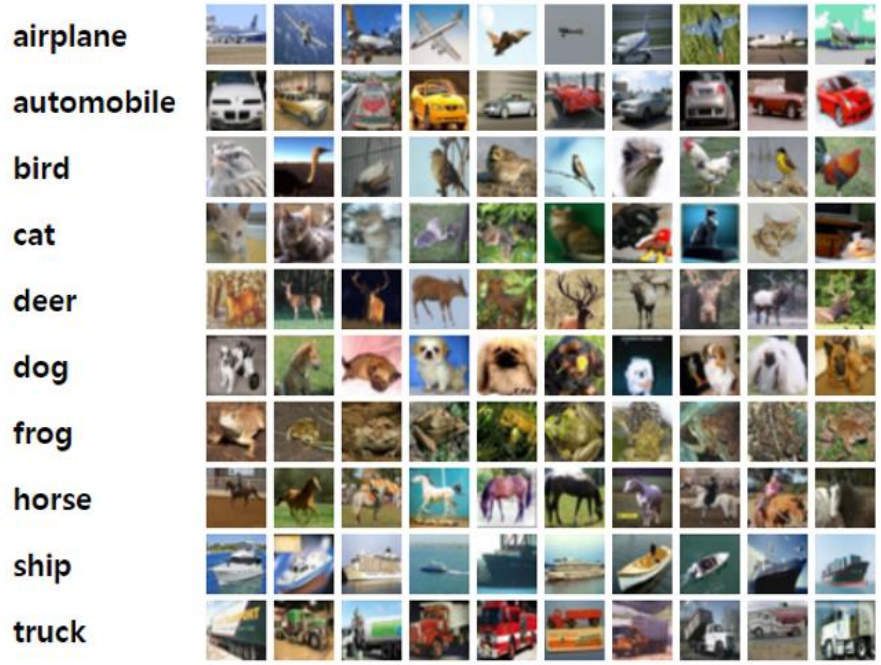
- CIFAR-10 살펴보기
- 설명 가능한 AI(XAI)
- 신경망 시각화
- 과대적합 피하기
- 데이터 증식 사용하기

Wrap Up

1. 컨볼루션 신경망은 이미지 데이터에 특화되어 있지만, 음성 인식이나 비디오, 텍스트 데이터에도 사용됩니다.
2. **완전연결층은 공간 정보를 손실**하는 반면, **컨볼루션층은 공간정보를 유지**합니다.
3. 컨볼루션층은 컨볼루션 필터를 통해 이미지의 특징을 인식할 수 있게 됩니다. 또한, **컨볼루션 필터가 가지는 파라미터**는 이미지 필터와 다르게 직접 정의해주지 않고, **학습을 통해 조정**됩니다.
4. 컨볼루션층에서는 주요한 인자로서 **컨볼루션 필터 개수, 스트라이드 크기, 패딩 여부**를 사용하고, 풀링층에서는 주요한 인자로서 **커널 크기, 스트라이드 크기**를 사용합니다.
5. 컨볼루션층은 1x1 크기를 사용하여 최대한 공간정보를 손실하지 않도록 하며, 다운샘플링이 필요할 경우 최대 풀링층을 사용합니다.
6. model.summary(), plot_model() 함수는 모델 구조를 확인하기에 유용합니다.

CIFAR-10 살펴보기

- CIFAR-10 데이터셋
  - 10개 클래스로 이루어져 있으며, CIFAR-100은 100개의 클래스로 이루어져 있음
  - MNIST 데이터셋과 함께 기본적으로 사용되는 데이터셋이지만, MNIST 데이터셋만큼의 성능을 기대하기 어려움
  - 50,000개 학습 데이터와 10,000 테스트 데이터



그림·5-15·CIFAR-10의·클래스·유형

CIFAR-10 살펴보기

- 신경망 모델의 입력으로 사용하기 위한 전처리 과정 수행
  - 채널별로 평균과 표준편차를 구함

```
01  # 평균과 표준 편차는 채널별로 구해줍니다.
02  x_mean = np.mean(x_train, axis = (0, 1, 2))
03  x_std = np.std(x_train, axis = (0, 1, 2))
04
05  x_train = (x_train - x_mean) / x_std
06  x_test = (x_test - x_mean) / x_std
```

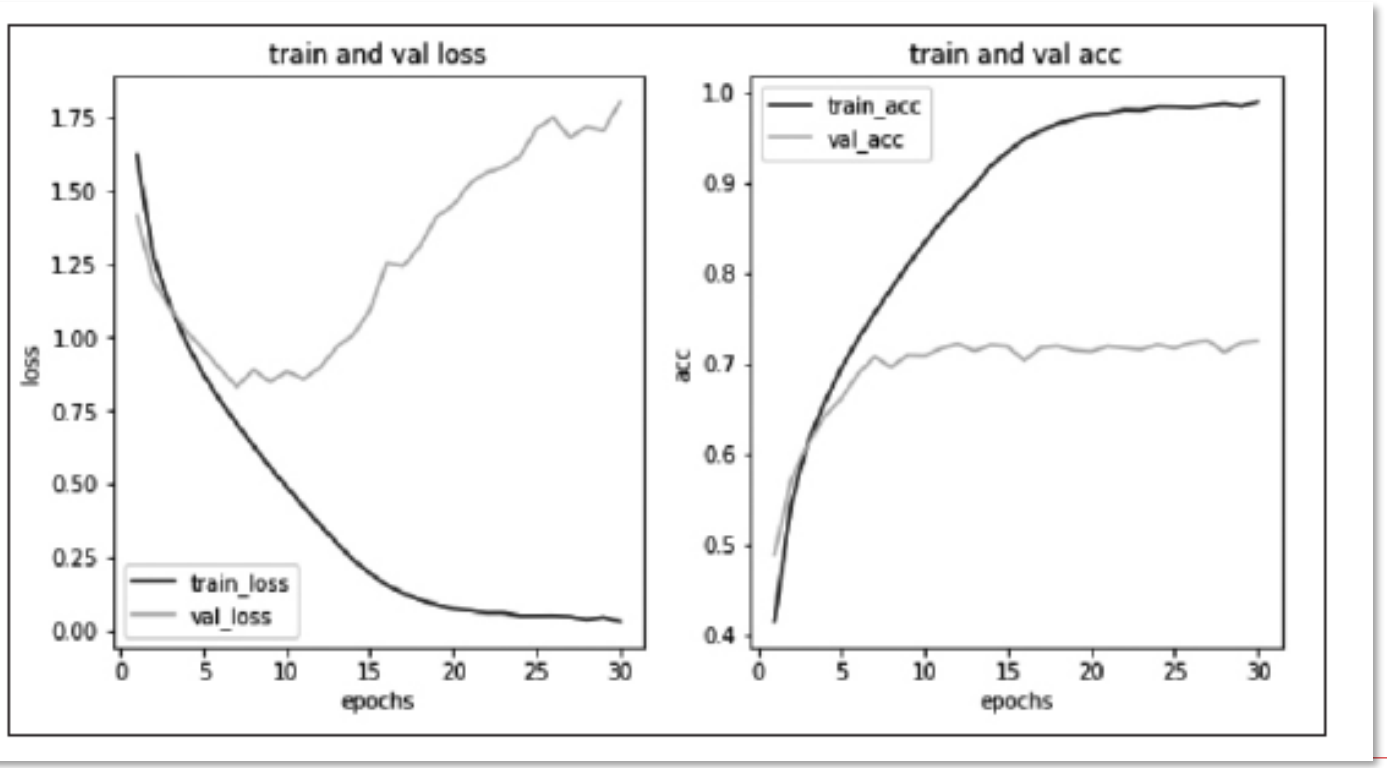
- 모델 구성
  - Conv2D, MaxPool2D
  - Adam(0.0001), sparse_categorical_crossentropy
  - **sparse_categorical_crossentropy**는 0~9 형태로 되어있는 레이블을 그대로 사용할 수 있게 해줌

```
08  model.add(Conv2D(filters = 32, kernel_size = 3, padding = 'same',
                     activation = 'relu'))
09  model.add(MaxPool2D(pool_size = (2, 2), strides = 2, padding = 'same'))
```

CIFAR-10 살펴보기(실습)

- 제공되는 코드를 통해 모델을 학습시켜보세요.

- 결과 확인
  - 과대적합 문제 발생!
  - 예방할 수 있는 방법을 알아보자



설명 가능한 AI(XAI)

- 신경망의 가장 큰 단점 중 하나는 **블랙박스인 모델을 쉽게 해석할 수 없다는 것**
  - **(고민)** 굳이 해석해야 하는가?
  - **(고민)** 자연의 이치와 같이 해석하지 않아도 좋은 성능 그대로를 사용한다면?
  - 그래도 해석해보자!

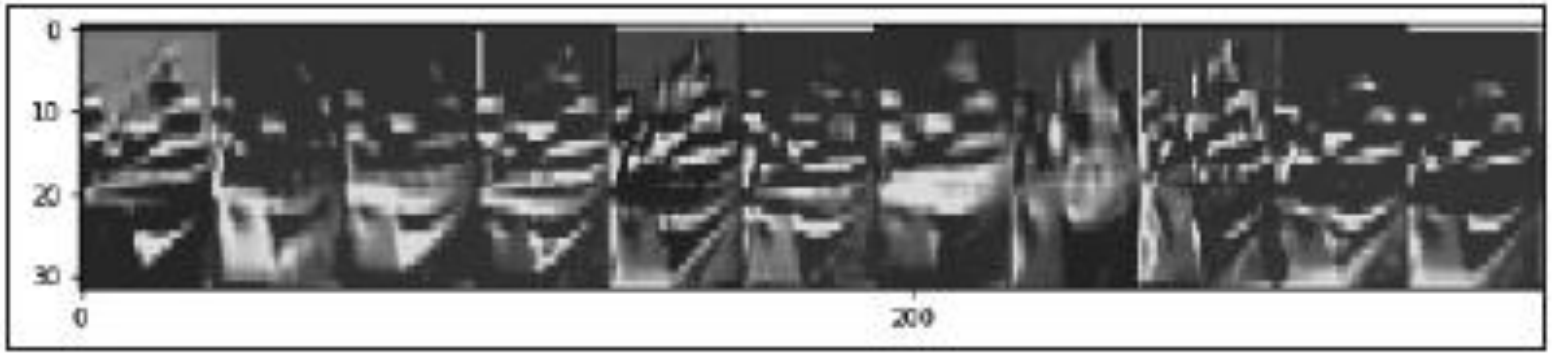
- 아무리 모델 성능이 좋다고 할지라도 왜 성능이 좋은지를 알지 못하면, 향후 모델의 견고함과 일반화를 위한 실험 방향 설정과 실제 서비스나 연구에서 신뢰성이 떨어지는 결과를 제공할 수 있음
  - ex) 환자가 병원에서 AI를 통해 진료받을 때, 의사가 결과를 설명하지 않고 단순히 AI의 판독 결과만 통보한다면?
  - 신뢰할 수 없는 결과: 이를 설명하지 못한 의사의 잘못? 해석 불가능한 AI의 잘못?
  - 이러한 문제를 해결하기 위해 연구되고 있는 분야: **설명 가능한 AI(XAI; Explainable AI)**

신경망 시각화

- 이미지에서 모델이 인식하는 특징을 확인하는 방법
  - 구체적인 것은 코드를 참고

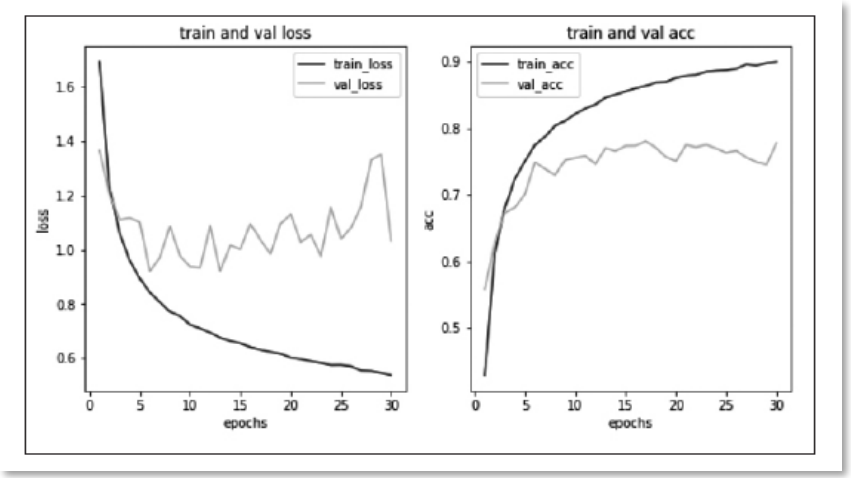
```
06  # 모델 전체에서 output을 가져올 수 있습니다.
07  visual_model = tf.keras.models.Model(inputs = model.input, outputs = get_output)
```

  - 모델이 포함하고 있는 Conv2D, MaxPool2D층에서 특징을 뽑아서 확인할 수 있음
  - ex) 배



과대적합 피하기

- 과대적합을 방지할 수 있는 2+1가지 방법
  - **여기서 설명할 방법은 예방책일 뿐, 100% 해결해주지 않음**
  - 오캄의 면도날(Occam's Razor) 이론 기반: 어떤 것을 설명하는 두 가지 방법이 있다면 더 정확한 설명은 **최소한의 가정이 필요한 가장 "간단한" 설명**
  - 규제화 함수(Regularizer)
  - 드롭아웃(Dropout)
  - 배치정규화(BatchNormalization)

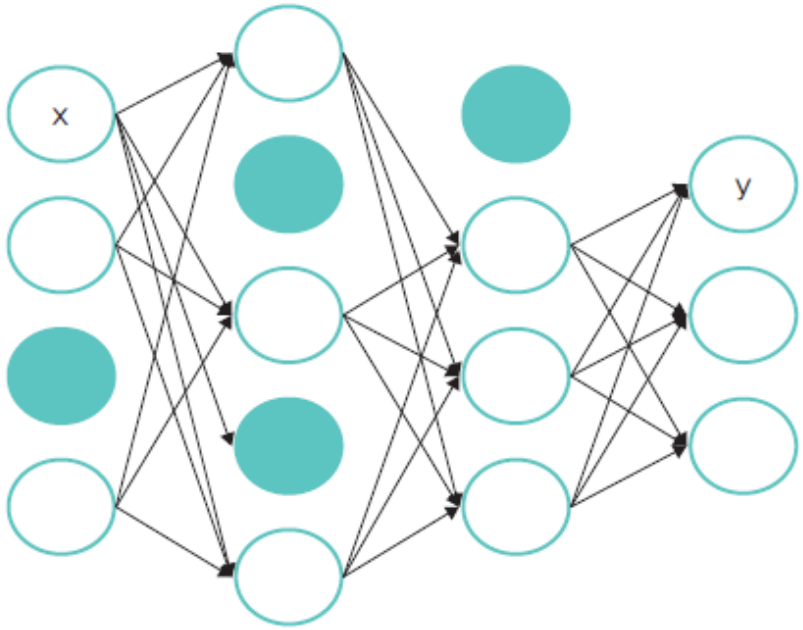


- 규제화 함수
  - **임의로 모델 복잡도를 제한**시키는 방법
  - L1 노름(Norm), L2 노름, 엘라스틱넷(ElasticNet)이 존재, 가중치 감쇠(Weight Decay)라고도 표현
  - 규제화 함수는 기능에 맞게 가중치의 합을 구하여 손실함수에 더해줌
  - 사용하지 않은 것보다 안정적으로 그래프가 그려짐

```
08  model.add(Conv2D(filters = 32, kernel_size = 3, padding = 'same',
                     activation = 'relu', kernel_regularizer = l2(0.001)))
```

과대적합 피하기

- 드롭아웃
  - 드롭아웃(dropout)은 신경망에서 가장 효과적이고 널리 사용하는 규제 기법 중 하나
  - 토론토(Toronto) 대학의 힌튼(Hinton)과 그의 제자들이 개발
  - **학습이 진행되는 동안 신경망의 일부 유닛을 제외(드롭)** 즉, 0으로 만듬
  - 테스트 시에는 드롭아웃이 작동하지 않고 모든 유닛이 활성화되는 대신, 출력값을 드롭아웃 비율만큼 줄여줌
  - 드롭아웃 비율(Dropout Rate)는 일반적으로 0.2~0.5를 사용



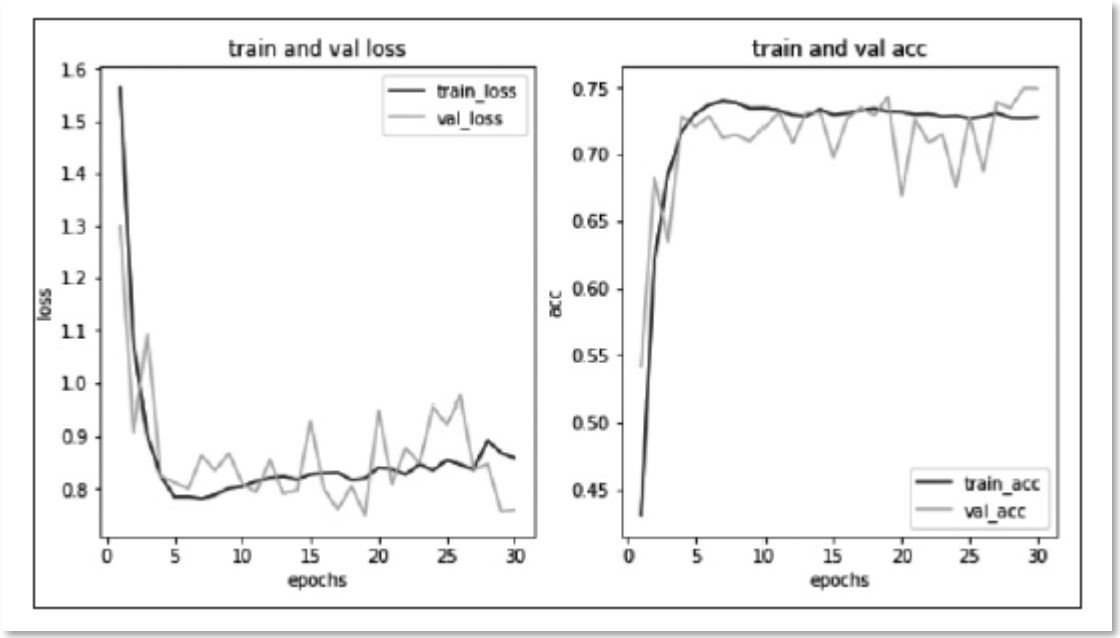
[그림 5-15] 드롭아웃

과대적합 피하기

- 드롭아웃
  - 제공되는 코드 참고

```
09  model.add(Dropout(0.2))  # 드롭아웃을 추가합니다.
```

  - 결과 확인
    - 효과가 매우 강력해보임
    - **학습 속도를 느리게 하는 단점**이 존재



과대적합 피하기

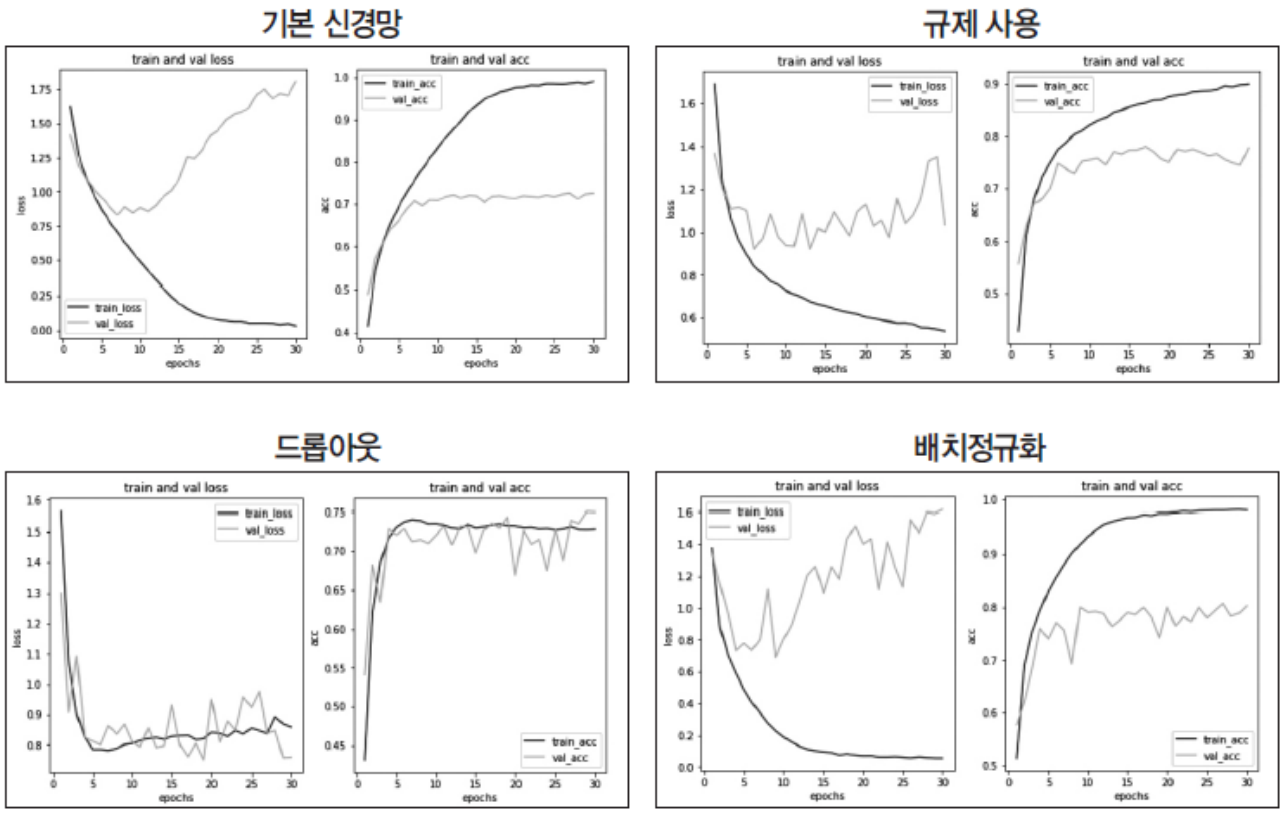
- 배치 정규화
  - 근본적으로 과대적합을 피하기 위한 방법으로 설명되지 않으나 드롭아웃과 비교되어 사용
  - **내부 공선성(Internal Covariance Shift)**를 해결하기 위해 고안된 방법
  - 신경망 층의 출력값은 다양한 입력 데이터에 따라 쉽게 변할 수 있는데, 매우 큰 범위의 출력값은 신경망을 불안정하게 할 수 있음. 따라서, 이 범위를 제한시켜 불확실성을 감소시키는 방법
  - 그래디언트 손실/폭발 없이 **높은 학습률을 사용할 수 있음**
  - **자체적인 규제화 효과가 포함**되어 있음. 보장하진 않으나 "이를 사용하면 별도의 규제화 함수나 드롭아웃을 사용하지 않아도 된다"라는 의견이 다수

- 배치 정규화 사용 순서
  - Dense층 또는 Conv2D층 → BatchNormalization() → Activation()
  - 최근 BatchNormalization() 함수를 층의 가장 앞에 사용되는 패턴도 사용되고 있음

```
09  model.add(Conv2D(filters = 32, kernel_size = 3, padding = 'same'))
10  model.add(BatchNormalization())
11  model.add(Activation('relu'))
```

과대적합 피하기

- 전체 결과
  - 그래프가 벌어지지 않는 것: 드롭아웃
  - 배치 정규화는 과대적합이 발생했지만, 가장 높은 성능을 보여줌
  - **하지만 이를 통해 "드롭아웃은 과대적합에 매~우 강력하다거나 배치 정규화를 사용하면 무조건 높은 성능을 얻을 것이다" 라는 편협적 시각은 절대 금물!**



[그림 5-16] 예제 결과 모음

데이터 증식 사용하기

- 데이터 증식(Data Augmentation)을 사용한 성능 향상
  - 딥러닝의 고질적인 문제: 일반화의 해결책, But 근본적으로는 해결 불가

- 데이터 증식의 장점
  - 다양한 데이터를 입력시킴으로써 모델을 더욱 **견고하게** 만들어주기 때문에 테스트 시에 더 높은 성능 기대
  - 수집된 데이터가 적은 경우 강력한 힘 → **일반화**

- 데이터 증식은 모델 성능에 큰 영향을 끼치기 때문에 관련 연구가 활발하게 진행되고 있음
  - Augmentation
  - Auto Augmentation

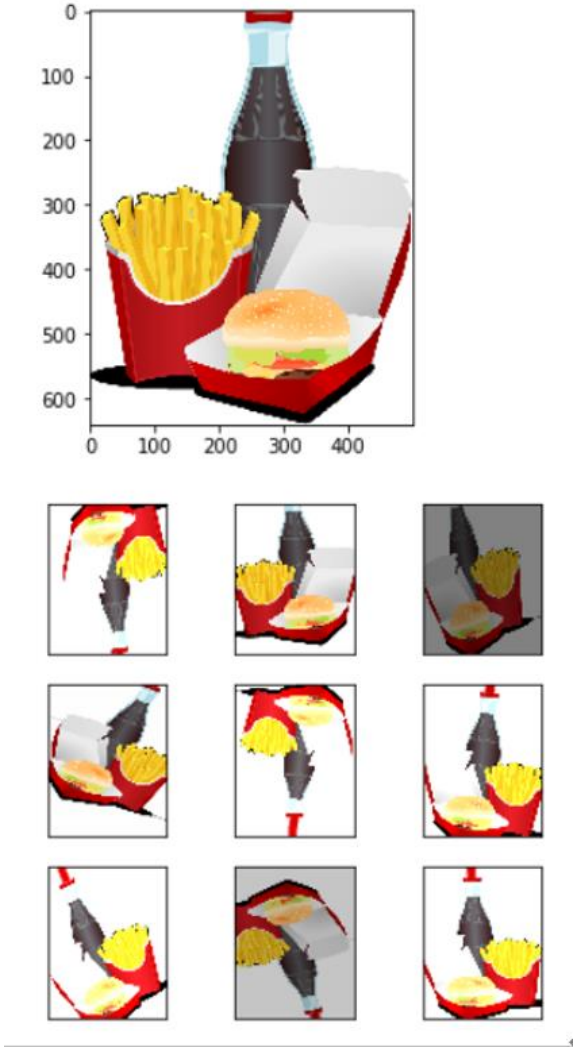
데이터 증식 사용하기

- 케라스는 이를 편리하게 사용할 수 있도록 **이미지 제네레이터(Image Generator)**를 제공하며, 변환 방식은 다음과 같음
  - width_shift_range: 임의의 크기만큼 너비 방향으로 이동시킵니다.
    - 0.2이고 이미지의 너비가 100이라면, -20~+20의 범위에서 너비 방향으로 이동시킵니다.
  - height_shift_range: 임의의 크기만큼 높이 방향으로 이동시킵니다.
    - 0.2이고 이미지의 높이가 100이라면, -20~+20의 범위에서 높이 방향으로 이동시킵니다.
  - brightness_range: 이미지의 밝기 정도를 조정합니다.
    - (0.5, 1.5)이면 원본 대비 최대 50%의 비율로 어둡거나 밝게 조절합니다.
  - shear_range: 시계 반대 방향으로 밀림 강도를 조절합니다.
    - 0.5이면, 최대 50%의 비율로 시계 반대 방향으로 기울어지게 됩니다.
  - zoom_range: 임의의 비율만큼 이미지를 확대/축소시킵니다.
    - 0.5이면, 0.5~1.5배의 범위에서 이미지의 크기를 조절합니다.
  - rotation_range: 이미지를 임의로 회전시킵니다.
    - 180이라면, 0~180의 범위에서 임의로 이미지를 회전시킵니다.
  - rescale: 이미지 픽셀값의 크기를 조절합니다.
    - 1/255이면, 각 픽셀값에 해당 값이 곱해집니다.
  - fill_mode: 이미지 변환 시에 새로 생기는 픽셀을 채울 방법을 결정합니다.
    - ["nearest", "constant", "reflect or wrap"]
  - horizontal_flip: True일 경우, 임의로 이미지를 수평 방향으로 뒤집습니다.
  - vertical_flip: True일 경우, 임의로 이미지를 수직 방향으로 뒤집습니다.
  - preprocessing_function: 사용자 정의 전처리 함수 또는 전처리 함수를 적용합니다.

데이터 증식 사용하기

- 이미지 제네레이터를 활용한 결과 확인

```
05 train_datagen = ImageDataGenerator(horizontal_flip = True,
06                                    vertical_flip = True,
07                                    shear_range = 0.5,
08                                    brightness_range = [0.5, 1.5],
09                                    zoom_range = 0.2,
10                                    width_shift_range = 0.1,
11                                    height_shift_range = 0.1,
12                                    rotation_range = 30,
13                                    fill_mode = 'nearest')
```



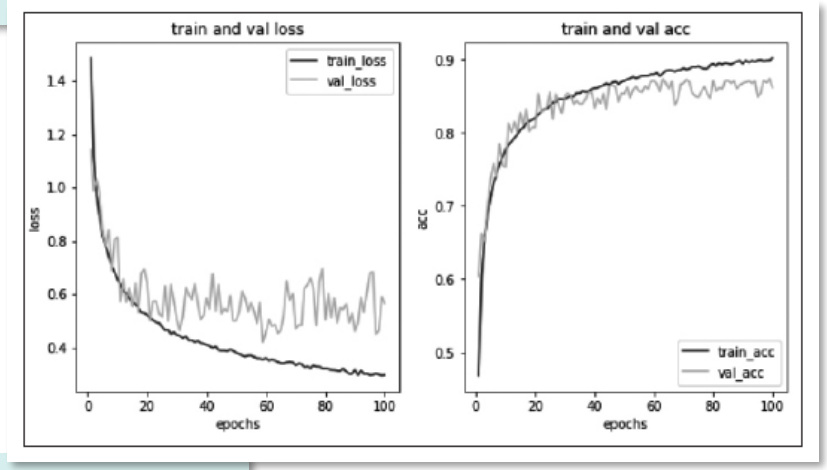
데이터 증식 사용하기

- 이미지 제네레이터 정의
  - 검증 데이터에 활용될 이미지 제네레이터는 증식 옵션을 사용하지 않음

```
03  train_datagen = ImageDataGenerator(horizontal_flip = True,
04                                     zoom_range = 0.2,
05                                     width_shift_range = 0.1,
06                                     height_shift_range = 0.1,
07                                     rotation_range = 30,
08                                     fill_mode = 'nearest')
09
10  val_datagen = ImageDataGenerator()
```

- 모델 학습
  - steps_per_epoch 인자
    - 1 에폭에 배치만큼의 크기를 몇번 전달할 것인지
    - '학습 데이터 개수/배치 크기 ' 를 전달
    - 제공되는 코드는 1094번
    - 해당 횟수보다 적은 값을 전달할 경우, 전체 데이터를 사용하지 않으므로 주의

```
27  history = model.fit(train_generator,
28                      epochs = 100,
29                      steps_per_epoch = get_step(len(x_train), batch_size),
30                      validation_data = val_generator,
31                      validation_steps = get_step(len(x_val), batch_size))
```

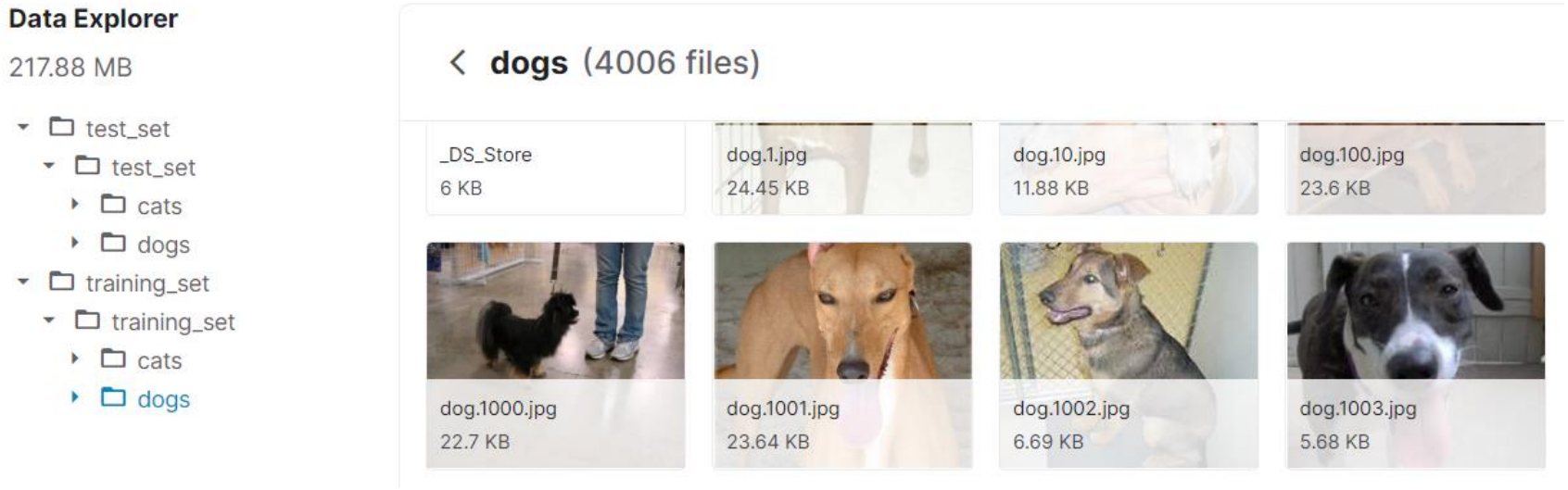


TensorFlow Keras를 활용한 컨볼루션 신경망 실습하기

- 개와 고양이 분류하기

CNN 신경망 모델 만들어보기 – 개와 고양이 분류하기

- **문제 정의하기**
  - CNN(Convolutional Neural Network)를 이용하여 개와 고양이 사진을 분류하는 모델을 만들어보기
  - 훈련데이터셋(training_set: cats; 4001건, dogs; 4006건=총 8007건)
  - 테스트데이터셋(test_set: cats;1012건, dogs; 1013건 = 총 2025건)
  - 강아지 고양이 사진을 학습하여 강아지; 1 고양이;0 으로 분류하는 이중 분류 문제로 정의



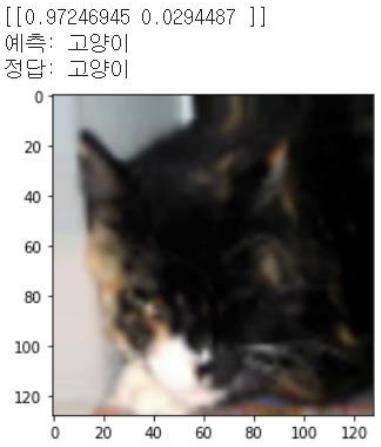
- **데이터 준비하기**
- https://www.kaggle.com/tongpython/cat-and-dog 데이터 확인하기
- **데이터 다운받아, 구글 드라이브에 업로드 진행**

CNN 신경망 모델 만들어보기 – 개와 고양이 분류하기

- **데이터 업로드가 힘든 경우 google api 사용하여 코랩에서 바로 다운 받기**
  - 고양이와 강아지 모두 1,000개의 훈련 이미지와 500개의 테스트 이미지로 구성
- **데이터 증식**
  - 데이터 증식(Data argumentaion)을 위해 케라스에서 제공하는 이미지 제너레이터를 사용
  - 이미지의 위치를 조금 옮긴다거나, 회전, 좌우반전등을 했을 때 컴퓨터가 받아들이는 이미지는 전혀 다른것이 되며, 변형을 줌으로써 학습 데이터를 늘리고, 이러한 변조에 강하게 모델을 학습시킬 수 있음
- **모델 구성하기**
  - Conv2D, MaxPooling2D, Dropout, Dense 레이어만 CNN 신경망 모델을 구성데이터 준비하기
  - 마지막 출력 Dense 레이어 이중 분류이기 때문에 0~1사이의 값을 나타내는 출력 뉴런이 1개 활성화 함수 sigmoid
- **모델 학습과정 설정하기** 손실함수(loss): binary_crossentropy, optimizer(하이퍼파라미터) : Adam , metrics : acc
- **모델 학습시키기** epochs=32
- **모델 평가하기, 모델 예측하기**

```
[42] model.evaluate_generator(validationGen)

     [0.5030390024185181, 0.75]
```



정리해봅시다

1. 컨볼루션 신경망은 이미지 데이터에 특화되어 있지만, 음성 인식이나 비디오, 6장에서 다뤄볼 텍스트 데이터에도 사용됩니다.
2. **완전연결층은 공간 정보를 손실**하는 반면, **컨볼루션층은 공간정보를 유지**합니다.
3. 컨볼루션층은 컨볼루션 필터를 통해 이미지의 특징을 인식할 수 있게 됩니다. 또한, **컨볼루션 필터가 가지는 파라미터**는 이미지 필터와 다르게 직접 정의해주지 않고, **학습을 통해 조정**됩니다.
4. 컨볼루션층에서는 주요한 인자로서 **컨볼루션 필터 개수, 스트라이드 크기, 패딩 여부**를 사용하고, 풀링층에서는 주요한 인자로서 **커널 크기, 스트라이드 크기**를 사용합니다.
5. 컨볼루션층은 1x1 크기를 사용하여 최대한 공간정보를 손실하지 않도록 하며, 다운샘플링이 필요할 경우 최대 풀링층을 사용합니다.
6. model.summary(), plot_model() 함수는 모델 구조를 확인하기에 유용합니다.
7. **규제화 함수, 드롭아웃, 배치 정규화**는 과대적합을 방지할 뿐, 100% 해결해주지 않습니다.

정리해봅시다

8. **데이터 증식 방법**은 다양한 데이터를 모델에 입력해 **더욱 견고한 모델**을 얻을 수 있도록 도와줍니다. 케라스에는 이를 위한 이미지 제네레이터가 준비되어 있습니다.
9. **전이 학습**은 **사전 학습된 가중치를 사용**하여 더욱 빠르게 향상된 성능을 얻을 수 있도록 도와줍니다. 케라스는 수많은 이미지 데이터로 구성된 ImageNet 데이터를 학습한 다양한 모델을 제공하고 있습니다.
10. **모델 상위층은 데이터의 구체적 특징**을 학습하며, **모델 하위층은 단순한 특징을 학습**합니다.
11. **텐서플로우 허브**는 우리가 원하는 모델을 쉽게 찾을 수 있도록 도와줍니다.

Thank you for your attention