Chapter

2

Model Setting

1. Data augmentation
2. CIFAR-10 dataset

강의에 앞서서..

❖ 본 문서는 아래의 자료들을 활용하여 만들어 졌음을 알립니다

❖ 모두를 위한 딥러닝 강좌

- 네이버 Search & Clova AI 부분 리더 김성훈 교수님
- https://www.youtube.com/playlist?list=PLlMkM4tgfjnLSOjrEJN31gZATbcj_MpUm
- https://www.edwith.org/boostcourse-dl-tensorflow/lecture/43739/

❖ 스탠포드 대학 CNN 강좌

- Fei-Fei Li & Andrej Karpathy & Justin Johnson
- http://cs231n.stanford.edu/slides/2020/

CS231n: Convolutional Neural Networks for Visual Recognition
- This course, Prof. Fei-Fei Li & Justin Johnson & Serena Yeung
- Focusing on applications of deep learning to computer vision

강의에 앞서서..

❖ **TensorFlow Image Classification**

- https://www.tensorflow.org/tutorials/images/classification

❖ **Hands-On Machine Learning**

- https://github.com/ExcelsiorCJH/Hands-On-ML

Data augmentation

❖ 데이터 증강(data augmentation)

- 딥러닝 모델을 충분히 훈련하는데 필요한 데이터를 확보하는 기법
- 적은 양의 훈련 데이터에 인위적인 변화를 가해 새로운 훈련 데이터를 대량 확보하는 방법론
- 이미지를 상하좌우로 뒤집거나(flipping) 자르는(cropping) 방식으로 새로운 이미지 데이터를 확보

Data augmentation

❖ tf.keras.ImageDataGenerator

수평 뒤집기(horizontal flip) 적용

In [37]:
```
# Apply horizontal flip
#수평뒤집기 처리된 이미지 생성을 위한 객체 생성
image_gen = ImageDataGenerator(rescale=1./255, horizontal_flip=True)
#image_gen = ImageDataGenerator(rescale=1./255, vertical_flip=True)
```

In [38]:
```
#train_dir로 부터 이미지를 읽어서 image_gen의  horizontal flip 처리를 포함한 이미지 생성
train_data_gen = image_gen.flow_from_directory(batch_size=batch_size,
                                               directory=train_dir,
                                               shuffle=True,
                                               target_size=(IMG_HEIGHT, IMG_WIDTH))
```

```
Found 2000 images belonging to 2 classes.
```

In [39]:
```
#첫번째 샘플에 대한 5개 증강 이미지를 리스트로 만들어 변수에 저장
#[train_data_gen[0][0][0][0] train_data_gen[0][0][0][1] ... train

augmented_images = [train_data_gen[0][0][0] for i in range(5)]
```

In [40]:
```
#증강 이미지 확인
plotImages(augmented_images)
```

Data augmentation

❖ tf.keras.ImageDataGenerator

회전 이미지 (Randomly rotate)

In [50]:
```
#회전 처리된 이미지 생성을 위한 객체 생성
image_gen = ImageDataGenerator(rescale=1./255, rotation_range=45)
```

In [51]:
```
#train_dir로 부터 이미지를 읽어서 image_gen의 이미지 생성
train_data_gen = image_gen.flow_from_directory(batch_size=batch_size,
                                               directory=train_dir,
                                               shuffle=True,
                                               target_size=(IMG_HEIGHT, IMG_WIDTH))
```

```
Found 2000 images belonging to 2 classes.
```

In [52]:
```
#첫번째 샘플에 대한 5개 증강 이미지를 리스트로 만들어 변수에 저장
#[train_data_gen[0][0][0][0] train_data_gen[0][0][0][1] ... train_data_gen[0][0][0][4]]
augmented_images = [train_data_gen[0][0][0] for i in range(5)]
```

In [53]:
```
#증강 이미지 확인
plotImages(augmented_images)
```

Data augmentation

❖ tf.keras.ImageDataGenerator

확대 이미지 (Randomly rotate)

In [58]:
```
#zoom_range (0 ~ 1), 1 = 100%,
image_gen = ImageDataGenerator(rescale=1./255, zoom_range=0.5) #  50% 확대
```

In [59]:
```
#train_dir로 부터 이미지를 읽어서 image_gen의 이미지 생성
train_data_gen = image_gen.flow_from_directory(batch_size=batch_size,
                                               directory=train_dir,
                                               shuffle=True,
                                               target_size=(IMG_HEIGHT, IMG_WIDTH))
```

```
Found 2000 images belonging to 2 classes.
```

In [60]:
```
#첫번째 샘플에 대한 5개 증강 이미지를 리스트로 만들어 변수에 저장
#[train_data_gen[0][0][0][0] train_data_gen[0][0][0][1] ... train_data_gen[0][0][0][4]]
augmented_images = [train_data_gen[0][0][0] for i in range(5)]
```

In [61]:
```
#증강 이미지 확인
plotImages(augmented_images)
```

Data augmentation

❖ tf.keras.ImageDataGenerator

다양한 증강 방법을 한번에 적용

In [68]:
```
#회전, 폭시프트,높이 시프트, 수평FLIP, 확대 50%
image_gen_train = ImageDataGenerator(
                    rescale=1./255,
                    rotation_range=45,
                    width_shift_range=.15,
                    height_shift_range=.15,
                    horizontal_flip=True,
                    zoom_range=0.5
                    )
```

In [69]:
```
train_data_gen = image_gen_train.flow_from_directory(batch_size=batch_size,
                                                     directory=train_dir,
                                                     shuffle=True,
                                                     target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                     class_mode='binary')
```

```
Found 2000 images belonging to 2 classes.
```

In [70]:
```
augmented_images = [train_data_gen[0][0][0] for i in range(5)]
plotImages(augmented_images)
```

Data augmentation

❖ tf.keras.ImageDataGenerator

Create validation data generator

In [71]:
```
image_gen_val = ImageDataGenerator(rescale=1./255)
```

In [72]:
```
val_data_gen = image_gen_val.flow_from_directory(batch_size=batch_size,
                                                 directory=validation_dir,
                                                 target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                 class_mode='binary')
```

```
Found 1000 images belonging to 2 classes.
```

Train the model (증강된 데이터 포함)

In [73]:
```
history = model.fit_generator(
    train_data_gen,
    steps_per_epoch=total_train // batch_size,
    epochs=epochs,
    validation_data=val_data_gen,
    validation_steps=total_val // batch_size
)
```

QUIZ!

cat & dog 예제(Lab11_1_CAT_DOG_classification_Keras _dataAugmentation.ipynb)에서 tf.keras.ImageDataGenerator의 width_shift_range, height_shift_range 로 증강된 이미지를 확인해보세요

9

Data augmentation

❖ tf.keras.ImageDataGenerator

데이터 증강 전

데이터 증강 후

CIFAR dataset

- **https://www.cs.toronto.edu/~kriz/cifar.html**

- **CIFAR-10 : 60000개의 32 x 32 컬러 이미지, 10개의 레이블**
  - 한 개의 레이블당 6000개의 이미지
  - 50000개의 Training Image, 10000개의 Test Image

- **CIFAR-100 : 60000개의 32 x 32 컬러 이미지, 100개의 레이블**
  - 클래스당 500개의 Training Image, 100개의 Test Image
  - 100개의 클래스는 20개의 슈퍼클래스에 속함

CIFAR dataset for Keras

❖ https://keras.io/datasets/

CIFAR10 적용

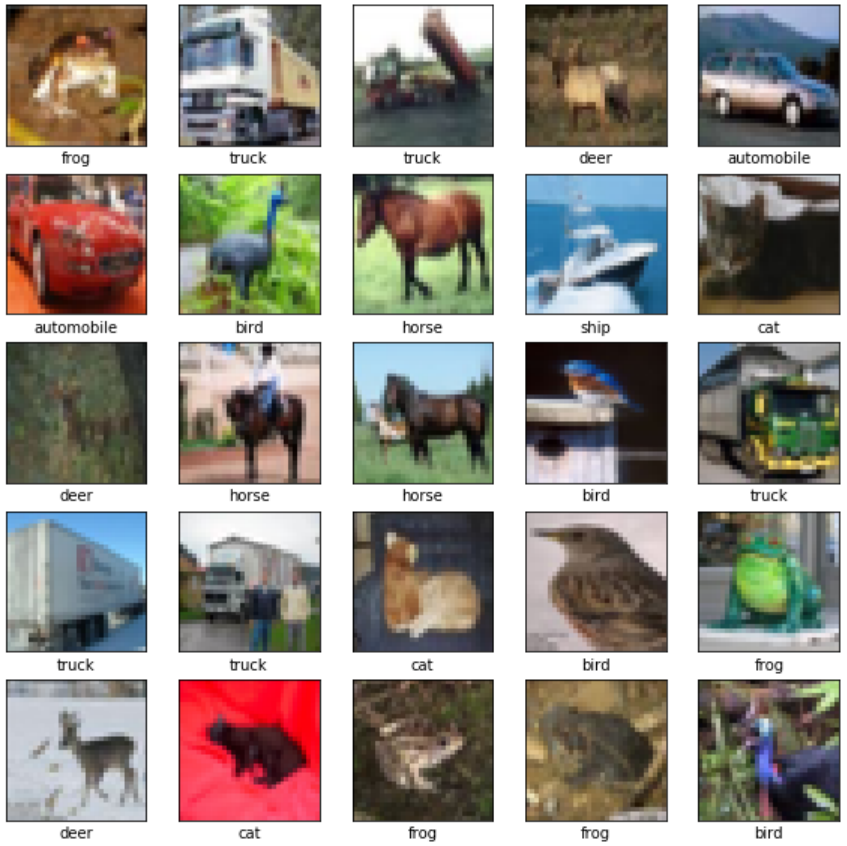
refered https://www.cs.toronto.edu/~kriz/cifar.html

In [110]:
```
from tensorflow.keras.datasets import cifar10

(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()

print("cifar10 train : ", train_images.shape)
print("cifar10 test  : ", test_images.shape)

train_images = train_images.astype(np.float32) / 255.
test_images = test_images.astype(np.float32) / 255.

class_names = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]
```

```
cifar10 train :  (50000, 32, 32, 3)
cifar10 test  :  (10000, 32, 32, 3)
```

CIFAR dataset for Keras

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i])
    plt.xlabel(class_names[train_labels[i][0]])
```



CIFAR dataset for Keras

```
#Create the model
model = Sequential([
    Conv2D(32, 3, padding='same', activation='relu', input_shape=(IMG_HEIGHT, IMG_WIDTH, 3)),
    MaxPooling2D(),
    Conv2D(32, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Conv2D(64, 3, padding='same', activation='relu'),
    MaxPooling2D(),

    Flatten(),
    Dense(10, activation='relu'),
    Dense(10, activation='relu'),
    Dense(10, activation='softmax')
])

# compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

#
model.summary()
```

```
#Train the model
history = model.fit(train_x, train_y, validation_data=(test_x, test_y), epochs=epochs, batch_size=128)
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/5
50000/50000 [==============================] - 31s 627us/sample - loss: 2.0650 - accuracy: 0.1869 - val_loss:
1.8293 - val_accuracy: 0.2389
Epoch 2/5
50000/50000 [==============================] - 31s 617us/sample - loss: 1.7782 - accuracy: 0.2713 - val_loss:
1.7476 - val_accuracy: 0.2905
Epoch 3/5
50000/50000 [==============================] - 31s 615us/sample - loss: 1.6786 - accuracy: 0.3214 - val_loss:
1.6003 - val_accuracy: 0.3515
Epoch 4/5
50000/50000 [==============================] - 31s 617us/sample - loss: 1.5650 - accuracy: 0.3682 - val_loss:
1.5346 - val_accuracy: 0.3739
Epoch 5/5
50000/50000 [==============================] - 31s 629us/sample - loss: 1.4926 - accuracy: 0.4053 - val_loss:
1.4683 - val_accuracy: 0.4157
```

```
loss, acc = model.evaluate(test_x, test_y,verbose=0)
print("loss=",loss)
print("acc=",acc)
```

```
loss= 1.468318296432495
acc= 0.4157
```

CIFAR dataset for Keras

image augmentation

ImageDataGenerator를 사용

```
# image augmentation
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
    rotation_range=10, # 0 ~ 180
    width_shift_range=0.1,
    height_shift_range=0.1,
    fill_mode='nearest',
    horizontal_flip=True,
    vertical_flip=False
)
# ADDED END
```

```
#Train the model
#history = model.fit(train_x, train_y, validation_data=(test_x, test_y), epochs=epochs, batch_size=128)
history = model.fit_generator(datagen.flow(train_x, train_y,batch_size=batch_size), epochs=epochs)

loss, acc = model.evaluate(test_x, test_y,verbose=0)
print("loss=",loss)
print("acc=",acc)
```

```
Epoch 1/5
250/250 [==============================] - 46s 186ms/step - loss: 2.0884 - accuracy: 0.2103
Epoch 2/5
250/250 [==============================] - 47s 188ms/step - loss: 1.6723 - accuracy: 0.3713
Epoch 3/5
250/250 [==============================] - 47s 190ms/step - loss: 1.5540 - accuracy: 0.4168
Epoch 4/5
250/250 [==============================] - 47s 189ms/step - loss: 1.4873 - accuracy: 0.4412
Epoch 5/5
250/250 [==============================] - 47s 188ms/step - loss: 1.4413 - accuracy: 0.4609
loss= 1.4105569246292113
acc= 0.4728
```

ConvNetJS CIFAR-10 demo

- https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html