Computer Vision_HW

B_3조 김동현

[기존 CNN 모델]

```
In [13]: y_train = np_utils.to_categorical(y_train,100)
         y_test = np_utils.to_categorical(y_test,100)

         x_train = x_train.astype('float32')/255.0
         x_test = x_test.astype('float32')/255.0

         model = Sequential()

         model.add(Conv2D(32,(2,2),padding='same',input_shape=(32,32,3)))
         model.add(LeakyReLU(alpha=0.1))
         model.add(Conv2D(32,(2,2),padding='same'))
         model.add(LeakyReLU(alpha=0.1))
         model.add(MaxPool2D(pool_size=(2,2)))
         model.add(Dropout(0.25))

         model.add(Conv2D(64,(2,2),padding='same'))
         model.add(LeakyReLU(alpha=0.1))
         model.add(Conv2D(64,(2,2),padding='same'))
         model.add(LeakyReLU(alpha=0.1))
         model.add(MaxPool2D(pool_size=(2,2)))
         model.add(Dropout(0.25))

         model.add(Conv2D(128,(2,2),padding='same'))
         model.add(LeakyReLU(alpha=0.1))
         model.add(Conv2D(128,(2,2),padding='same'))
         model.add(LeakyReLU(alpha=0.1))
         model.add(MaxPool2D(pool_size=(2,2)))
         model.add(Dropout(0.35))

         model.add(Conv2D(256,(2,2),padding='same'))
         model.add(LeakyReLU(alpha=0.1))
         model.add(Conv2D(256,(2,2),padding='same'))
         model.add(LeakyReLU(alpha=0.1))
         model.add(MaxPool2D(pool_size=(2,2)))
         model.add(Dropout(0.5))

         model.add(Flatten())
         model.add(Dense(512))
         model.add(LeakyReLU(alpha=0.1))
         model.add(Dropout(0.5))
         model.add(Dense(100,activation='softmax'))

         model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

         model.summary()
```

Model: "sequential_3"

Layer (type)	Output Shape	Param #
conv2d_20 (Conv2D)	(None, 32, 32, 32)	416
leaky_re_lu_22 (LeakyReLU)	(None, 32, 32, 32)	0
conv2d_21 (Conv2D)	(None, 32, 32, 32)	4128
leaky_re_lu_23 (LeakyReLU)	(None, 32, 32, 32)	0
max_pooling2d_10 (MaxPooling	(None, 16, 16, 32)	0
dropout_13 (Dropout)	(None, 16, 16, 32)	0
conv2d_22 (Conv2D)	(None, 16, 16, 64)	8256
leaky_re_lu_24 (LeakyReLU)	(None, 16, 16, 64)	0
conv2d_23 (Conv2D)	(None, 16, 16, 64)	16448
leaky_re_lu_25 (LeakyReLU)	(None, 16, 16, 64)	0
max_pooling2d_11 (MaxPooling	(None, 8, 8, 64)	0
dropout_14 (Dropout)	(None, 8, 8, 64)	0
conv2d_24 (Conv2D)	(None, 8, 8, 128)	32896
leaky_re_lu_26 (LeakyReLU)	(None, 8, 8, 128)	0
conv2d_25 (Conv2D)	(None, 8, 8, 128)	65664
leaky_re_lu_27 (LeakyReLU)	(None, 8, 8, 128)	0
max_pooling2d_12 (MaxPooling	(None, 4, 4, 128)	0
dropout_15 (Dropout)	(None, 4, 4, 128)	0
conv2d_26 (Conv2D)	(None, 4, 4, 256)	131328
leaky_re_lu_28 (LeakyReLU)	(None, 4, 4, 256)	0
conv2d_27 (Conv2D)	(None, 4, 4, 256)	262400
leaky_re_lu_29 (LeakyReLU)	(None, 4, 4, 256)	0
max_pooling2d_13 (MaxPooling	(None, 2, 2, 256)	0
dropout_16 (Dropout)	(None, 2, 2, 256)	0
flatten_3 (Flatten)	(None, 1024)	0
dense_6 (Dense)	(None, 512)	524800
leaky_re_lu_30 (LeakyReLU)	(None, 512)	0
dropout_17 (Dropout)	(None, 512)	0
dense_7 (Dense)	(None, 100)	51300

Total params: 1,097,636
Trainable params: 1,097,636
Non-trainable params: 0

```
In [15]: loss_and_accuracy = model.evaluate(x_test, y_test, batch_size=64)
         print('loss : %.4f, accruracy : %.4f'%(loss_and_accuracy[0],loss_and_accuracy[1]))

         157/157 [==============================] - 6s 38ms/step - loss: 1.8239 - accuracy: 0.5162
         loss : 1.8239, accruracy : 0.5162
```

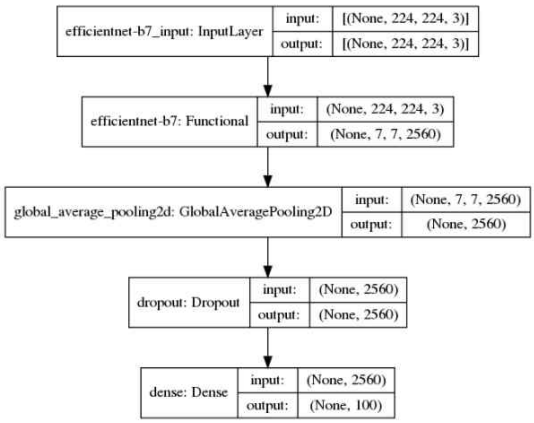
- kernel 사이즈를 2x2로 설정하여 섬세한 CNN 시도
- 사진의 크기가 32x32임을 알고 pooling을 3번 정도로 하기로 결정
- Activation 함수로 구글링하여 찾아본 결과 기본 Relu보다 LeakyRelu를 사용했을 때 성능이 일반적으로 좋다는 말을 듣고 LeakhyRelu 함수를 사용
- 성능 향상을 위해 적당한 Dropout을 사용
- optimizer 함수로 Adam 함수를 사용

=> 이 밖에 파라마티를 조정하는 등 여러 시도를 해보았으나 성능이 51.62% 까지 밖에 오르지않음.

다음장 계속=>

[EfficientNet 모델] - 최종 모델

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
efficientnet-b7 (Functional) (None, 7, 7, 2560)        64097680
_________________________________________________________________
global_average_pooling2d (Gl (None, 2560)              0
_________________________________________________________________
dropout (Dropout)            (None, 2560)              0
_________________________________________________________________
dense (Dense)                (None, 100)               256100
=================================================================
Total params: 64,353,780
Trainable params: 64,043,060
Non-trainable params: 310,720
```



The accuracy on the testing data : 62.58%

- Cifar100 데이터에 대한 여러 CNN 모델을 찾다가 Kaggle에서 1등을 차지한 'EfficientNet'이라는 모델을 발견
- Kaggle에서 이용한 모델은 EfficientNet 모델 중 B0 모델을 사용하였으나 B7 모델의 성능이 대체적으로 더 좋다는 말을 듣고 B7 모델로 변경하여 모델링을 시도
- 현재 B8까지 나왔지만 Keras에서는 B7 모델까지 지원
- optimizer 함수를 SGD에서 ADAM으로 변경
- batch size를 8로 설정하고 epoch 수를 25에서 50으로 두 배 늘림으로써 성능을 향상
- epoch이 증가할수록 validation accuracy가 조금씩 증가하는 것으로 보아, epoch 수를 100을 준다면 test accuracy가 65%까지 증가할 것으로 예상됨.