**과목명: 시스템프로그래밍**

**분반 01**

**<<Project #4>>**

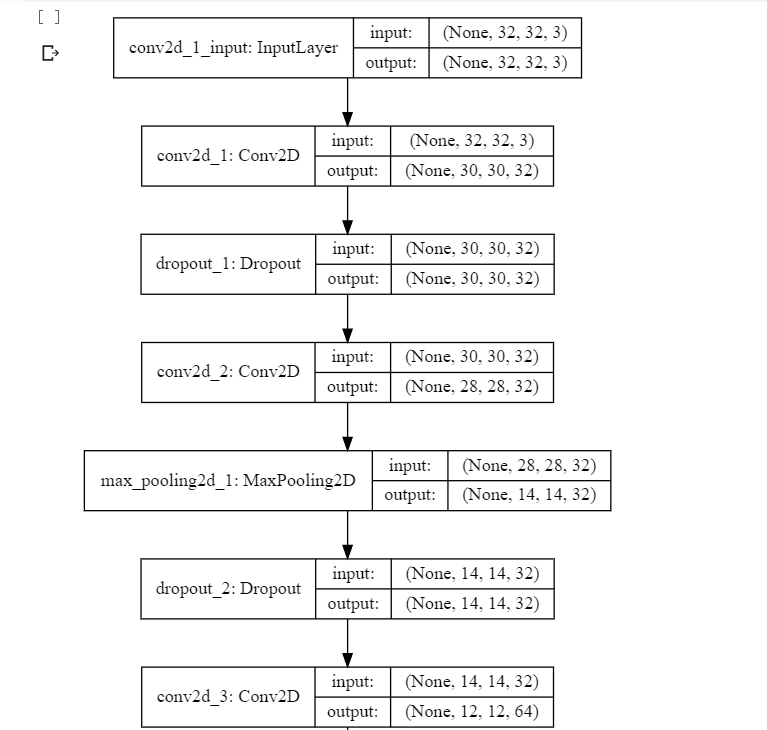
**서강대학교 컴퓨터공학과**

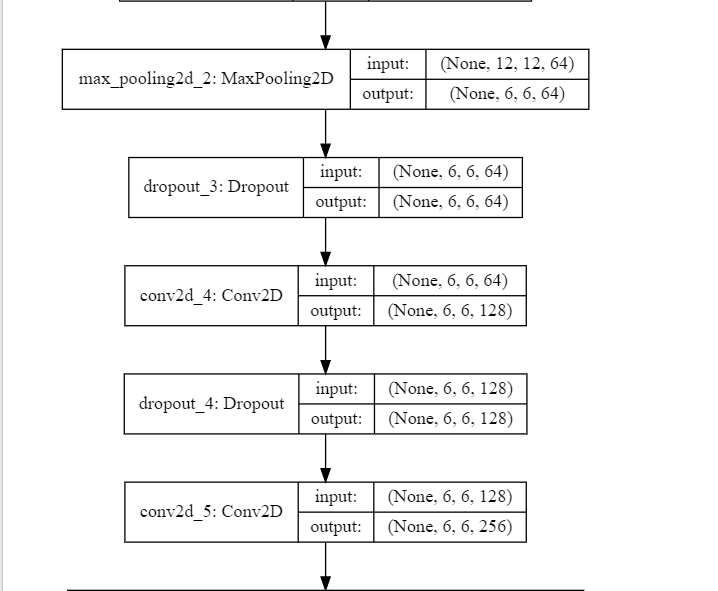
**20181679**

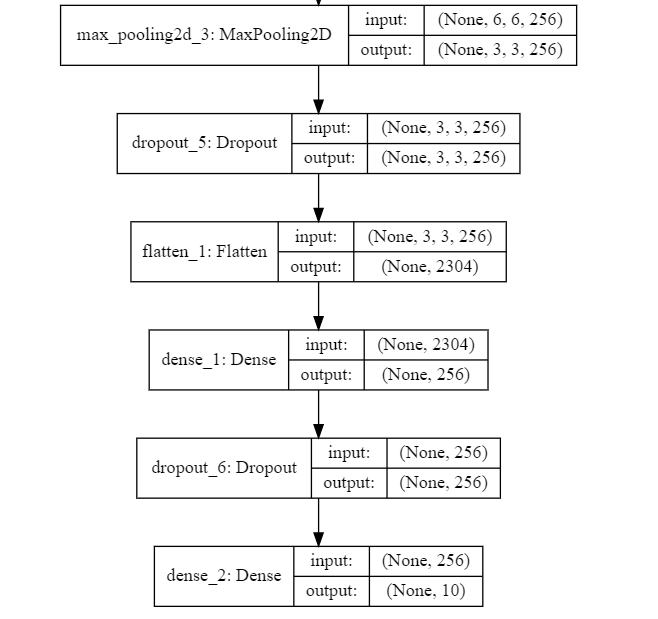
**장서우**

목 차

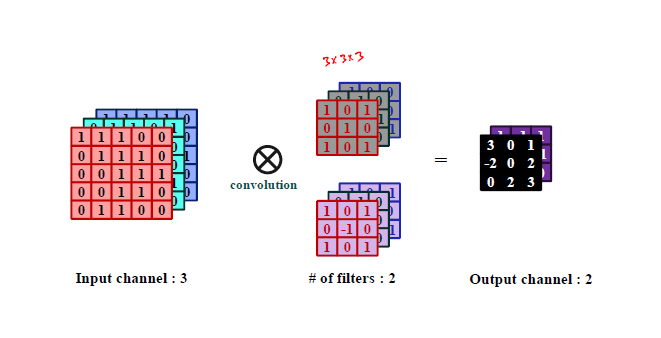
1. **프로그램 개요**
   1. 목표
   2. 실행방법
2. **Import modules and tuning parameters**
3. **Loading the data**
4. **Reshaping the data**
5. **Creating the model**
6. **Compiling and training the model**
7. **Evaluating and prediction the model**
8. **프로그램 개요**
   1. 목표
      1. Google Colab에서 Keras를 이용하여 이미지 분류를 하는 머신러닝 모델을 개발한다. CIFAR-10이라는 dataset을 이용하여 이미지 분류 성능을 측정하고 직접 모델의 파라미터들을 바꿔가면서 테스트 셋에 대한 정확도 80% 이상을 얻는다.
      2. CIFAR-10 데이터 셋은 60000개의 32\*32 픽셀의 컬러이미지로 총 10개의 클래스가 있다.
   2. 실행 방법
      1. 구글 로그인을 한 후 Google Colaboratory를 연다. 왼쪽 상단의 파일을 클릭한 후 노트 열기를 사용하여 들어가고 싶은 노트를 불러올 수 있다.
      2. 런타임에서 모두 실행을 클릭하면 Code Cell 전체 실행을 할 수 있다.
      3. 파일 탭의 .ipynb 다운로드를 하면 작업한 notebook이 .ipynb형식으로 저장된다.
9. **Import modules and tuning parameters**
   1. 모델을 개발하는데 필요한 모듈을 import한다.
   2. 전체 학습 데이터 셋에 대해서 몇 번 학습시킬 것인지를 의미하는 epochs는 50으로 고정한다.
   3. learning rate는 학습 시 가중치가 업데이트 되는 정도를 나타내는 수치로 0.01이 적절한 값이다.
   4. batch size는 입력 데이터로 사용되는 소그룹에 포함되는 샘플의 개수를 나타낸다. 하나의 batch에 있는 샘플들로 forward 연산을 수행하고, 에러의 평균치를 가지고 backprob 연산을 수행한다.
   5. num classes는 분류해야 하는 클래스의 개수로 10이다.
10. **Loading the data**
    1. from keras.datasets import cifar10 을 통해서 cifar10 모듈을 가져오고, cifar.load\_data()를 통해 데이터를 가져올 수 있다.
11. **Reshaping the data**
    1. 학습을 위해 데이터를 전처리한다.
    2. mnist의 경우, 이미지를 담은 배열의 rank를 4로 맞춰준다. 이미지 데이터는 전형적으로 높이, 너비, 컬러 채널의 3차원으로 이루어진다. 이미지 텐서는 항상 3D로 저장된다. 흑백 이미지의 경우 컬러 채널의 차원 크기는 1이다. .shape의 parameter는 순서대로 이미지 개수, 높이, 너비, 컬러 채널이다. reshape 전후의 x\_train.shape를 비교해보면 전은 (60000, 28, 28)로 28\*28 이미지가 60000개 있음을 의미하고 후는 (60000, 28, 28, 1)로 컬러 채널이 추가되어 4차원으로 나타남을 알 수 있다.
    3. cifar10의 rank는 이미 4이므로 넘어간다.
12. **Creating the model**



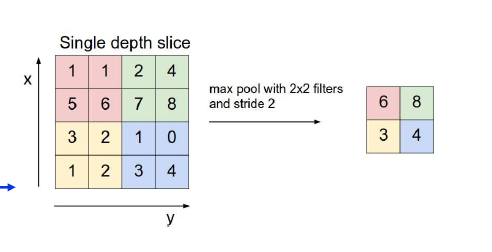




* 1. Conv2D의 첫번째 인자는 convolution filter 개수이고, 두번째 인자는 convolution kernel의 (행,열)을 의미한다. 이번 프로젝트에서는 3\*3의 filter를 사용하였고, activation 함수는 활성화 함수를 설정하는 것으로, relu(rectified linear unit)를 사용하였다. 이는 filter에 따라 input과 output dimension이 바뀌게 된다.
  2. 맨 처음 input layer는 32\*32 이미지의 channel 3개로 이루어져 있다. 이 input layer가 첫번째 convolution **conv2d\_1**을 거치면 3\*3 filter 32개를 사용하였기 때문에 output dimension은 30\*30 32개(=filter의 개수)가 나올 것이다. **conv2d\_3**을 보면 3\*3 filter 64개를 사용하였기 때문에 14\*14가 12\*12가 되고 32가 64로 바뀜을 알 수 있다. **conv2d\_2**도 동일한 방식이다.
  3. 일부 padding값을 설정하였는데 이는 경계 처리방법을 정의하며 ‘same’은 출력 이미지 사이즈가 입력 이미지 사이즈와 동일하게 설정하는 것이다. 이는 **conv2d\_4**와 **conv2d\_5**에서 살펴볼 수 있는데 input과 output의 행렬 dimension 차이 없이 channel 수만 바뀜을 확인할 수 있다.

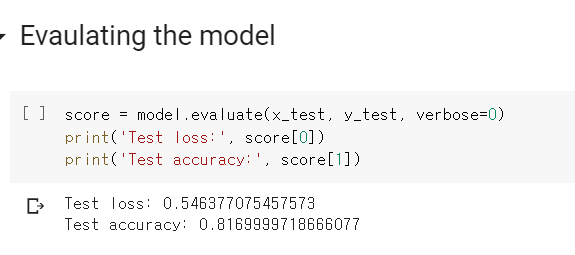


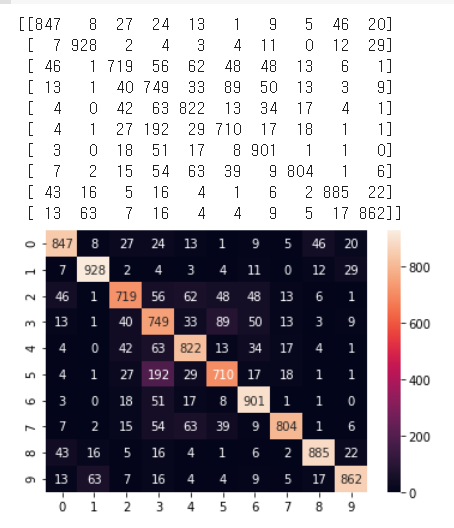
* 1. Dropout은 input, output dimension에 영향을 끼치지 않는다. dropout은 전체 weight을 계산에 참여시키는 것이 아니라 layer에 포함된 weight 중에서 일부만 참여시키는 기능을 한다.
  2. MaxPooling2D는 pool size(2,2)를 사용하여 2\*2 kernel size의 maxpooling하는 layer를 추가한다. maxpooling은 다음과 같이 적용된다.



input과 output dimension을 보면 다른 값의 변경 없이 행과 열만 1/2이 됨을 확인할 수 있다.

* 1. Flatten은 CNN에서 convolution layer나 maxpooling layer는 주로 2차원 자료를 다루지만 fully connected layer에 전달하기 위해선 1차원 자료로 바꿔줘야 한다. 이 역할을 하는 것이 Flatten이다.
  2. model.add(Dense(256, activation='relu'))는 256개의 unit을 가진 fully connected layer를 추가한다. 고로 output 뉴런의 개수가 256이 된다. 마지막은 model.add(Dense(num\_classes, activation='softmax'))를 사용하여 num classes만큼의 출력 뉴런 수를 가지며 활성화 함수는 softmax를 사용한다. softmax는 입력 받은 값을 출력으로 0~1사이의 값으로 모두 정규화하며 출력 값들의 총합은 항상 1이 되는 특성을 가진 함수이다

1. **Compiling and training the model**
   1. 입력 샘플을 랜덤하게 선택하는 SGD(Stochastic Gradient Descent) 방식을 사용한다.
   2. Optimizer는 SGD로 설정한다. 이는 확률적 경사 하강법(Stochastic Gradient Descent, SGD) 모멘텀과 네스테로프 모멘텀(Nesterov momentum), 그리고 학습률 감소 기법(learning rate decay)을 지원한다.
   3. 인자는 lr는 learning rate을 의미하고, momentum은 SGD를 적절한 방향으로 가속화하며, 흔들림(진동)을 줄여주는 매개변수이다. decay는 업데이트마다 적용되는 학습률의 감소율이다. nesterov는 네스테로프 모멘텀의 적용 여부이다.
   4. 이렇게 nesterov까지 적용된 optimizer를 Nesterov Accelrated Gradient(NAG, 네스테로프 모멘텀) 라고 한다. 이는 momentum값과 gradient값이 더해져 실제(actual)값을 만드는 기존 모멘텀과 달리 momentum값이 적용된 지점에서 gradient값이 계산된다. 이 단계를 추가함으로써 한단계를 미리 예측하여 불필요한 이동을 줄일 수 있다.
   5. model.compile로 모델을 컴파일한다.
   6. 여기서 loss function은 cross entropy를 사용하는데 클래스 개수가 10개이므로 categorical\_crossentropy를 사용한다. cross entropy에는 binary\_crossentropy와 categorical\_crossentropy가 있는데, binary는 output layer가 sigmoid인 경우에 사용하고, categorical은 output layer가 softmax일 경우에 사용한다. 손실 함수 categorical\_crossentropy의 경우 사용되는 타겟들은 categorical format을 따라야 한다. 예를 들어 10개의 클래스 중 하나에 속하는 데이터에 대하여 각 샘플은 타겟 클래스에 해당하는 하나의 인덱스만 1의 값을 가지고 이외의 값들은 모두 0이어야 한다. ([0,1] or [1,0]의 형태) 이런 형태를 가지고 softmax\_cross\_entropy를 loss로 하고자 할 때는 categorical\_crossentropy를 쓰면 된다.
   7. model.fit를 하면 만든 모델과 설정한 파라미터들을 가지고 학습을 진행하게 된다.
2. **Evaluating and prediction the model**
   1. 이번 프로젝트에서는 다음과 같은 정확도를 얻었다. 
   2. confusion matrix



-아래는 heatmap을 사용하였다.