# [INTRODUCTION]

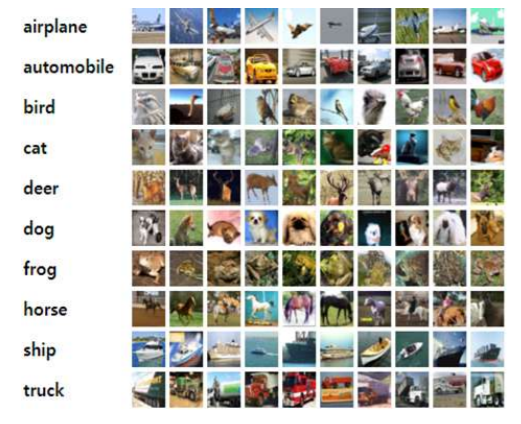
4차 산업혁명의 AI(Artificial Intellectual) 분야라고 할 수 있는 딥러닝(Deep Learning)이 많은 분야에서 다양한 알고리즘과 함께 개발되어 왔다. 인간은 자연스럽게 할 수 있었던 이미지 인식, 음성 인식, 글자체 인식 등을 딥러닝을 통해 컴퓨터에서 인식할 수 있게 되었다. 이 논문은 딥러닝 연구 대상의 기초라고 할 수 있는 이미지 인식에 대한 것이다. 컴퓨터에서 이미지 인식은, RGB의 Pixel들로 이루어진 이미지를 작은 단위로 쪼개어 규칙성을 학습하고 어떤 class인지 판단하여 분류하는 것이다.

Lecun 교수가 오래전에 고안했던 Convolutional Neural Network(CNN) 알고리즘이 이미지 인식 분야에서 각광 받고 있다. 이 CNN을 통하여 이미지 특징의 인식 및 분류가 점점 고도화 되기 시작하였고, 컴퓨터비젼 및 기계학습 영역에서 큰 영향력을 미치게 되었다.

이번 논문에서는 여러 층의 CNN 알고리즘을 통해 컴퓨터가 어떻게 10개의 카테고리를 가지는 이미지를 분류하는지, 분류하는 정확도가 Activation function 종류에 따라 어떻게 달라지는지, Pooling 방식을 Average 혹은 Maximum 방법으로 하였을 때 정확도 차이를 확인하고, 최고의 정확도를 도출하기 위한 방법을 분석해 볼 것이다.

# [The DATASET]

CIFAR10 데이터 세트는 일반적으로 기계학습 및 컴퓨터 비젼 알고리즘을 훈련하는데 사용되는 이미지의 모음이다. CIFAR10 데이터세트에는 10개의 서로 다른 클래스에서 60,000개의 32\*32Pixel 컬러 이미지가 포함되어 있다. 10개의 다른 종류는 비행기, 차, 고양이, 사슴, 개 및 트럭 등이 있다. 각 클래스에는 6,000개의 이미지가 있다. CIFAR10의 이미지는 저해상도이므로 이 데이터세트를 사용하면 다른 알고리즘을 적용하였을 때 신속하게 어떤 것이 어떻게 작동되는지 확인할 수 있다. 만약 CIFAR100 등 클래스가 100개로 많거나 이미지 사이즈가 크면 계산이 커지고 Gradient vanishing 현상이 일어날 수 있어 CIfar10으로 결정하였다. 이런 Vanishing 문제는 Activation function의 변경을 통해 어느정도 극복할 수 있다.



# CNN 개요

## CNN은 전통적인 뉴럴 네트워크에 여러 계층의 컨볼루셔널 계층을 붙인 모델을 사용한다. CNN은 앞의 컨볼루셔널 계층을 통해서 입력 받은 이미지에 대한 특징을 추출하고, 추출된 특징을 기반으로 기존의 뉴럴 네트워크를 이용하여 이미지를 분류한다. 이 컨벌루션 신경망들은 학습 가능한 가중치와 바이어스로 구성되어 있는데 각 뉴런의 입력을 받아 내적연산(dot product)를 한 뒤 비선형 연산(non-linear)을 한다. 마지막 레이어에는 Softmax와 같은 손실 함수 (loss function)을 가지며, 우리가 일반 신경망을 학습시킬 때 사용하던 각종 기법들을 동일하게 적용할 수 있다. 신경망은 입력받은 벡터를 일련의 히든 레이어 (hidden layer) 를 통해 변형 (transform) 시킨다. 각 히든 레이어는 뉴런들로 이뤄져 있으며, 각 뉴런은 앞쪽 레이어 (previous layer)의 모든 뉴런과 연결되어 있다 (fully connected). 같은 레이어 내에 있는 뉴런들 끼리는 연결이 존재하지 않고 서로 독립적이다. 마지막 Fully-connected 레이어는 출력 레이어라고 불린다. 2차원의 이미지는 히든 레이어와 출력 레이어를 통해 3차원의 볼륨을 갖는 값으로 변환되고, (Cifar10의 이미지는) 10개 class 중 하나로 구별된다. 이미 Google에서 제공하는 Tensorflow 나 Keras를 통하여 개발자가 아닌 개인들도 딥 러닝과 CNN을 이용한 이미지 분류를 실험할 수 있게 되었다. CNN은 각 층을 통하여 이미지의 특징을 추출해 내는 것으로 정확도를 높여 나간다.



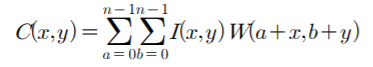
# 컨벌루션 신경망

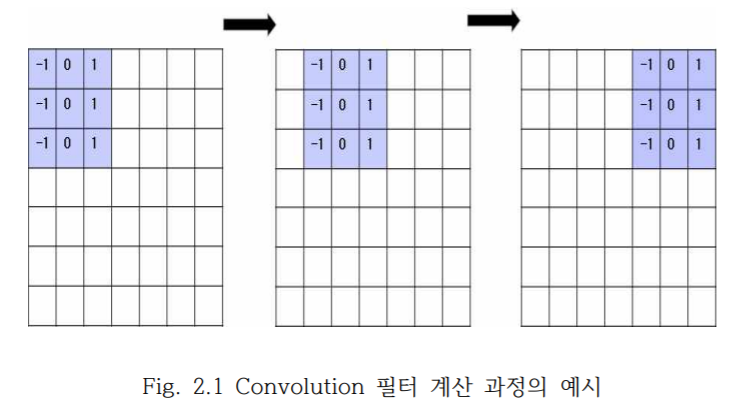
## 입력 데이터로 부터 특징을 추출하는 역할을 하는 컨볼루셔널 레이어는 특징을 추출하는 기능을 하는 필터와 이 필터의 값을 비선형 값으로 바꾸어 주는 Activation function 으로 이루어져 있다. 일반 신경망은 이미지를 다루기에 적절하지 않다. CIFAR-10 데이터의 경우 각 이미지가 32x32x3 (가로,세로 32, 3개 컬러 채널)로 이뤄져 있어서 첫 번째 히든 레이어 내의 하나의 뉴런의 경우 32x32x3=3072개의 가중치가 필요하고, 더 큰 이미지의 경우 필요한 가중치도 기하급수적으로 커진다. 더욱이, 이런 뉴런이 레이어 내에 여러개 존재하므로 모수의 개수가 크게 증가하게 된다. 이와 같이 Fully-connectivity는 심한 낭비이며 많은 수의 모수는 곧 오버피팅(overfitting)으로 귀결된다. 컨벌루션 신경망은 입력이 이미지로 이뤄져 있다는 특징을 살려 좀 더 합리적인 방향으로 모델을 구성할 수 있다. 특히 일반 신경망과 달리, 컨벌루션 신경망의 레이어들은 가로,세로,깊이의 3개 차원을 갖게 된다. 하나의 레이어에 위치한 뉴런들은 일반 신경망과는 달리 앞 레이어의 전체 뉴런이 아닌 일부에만 연결이 되어 있다. 처음 인풋 데이터는 이미지로서 32pixel\*32pixel\*3(Red, Blue, Green) 이고, 컨벌루션 신경망 모델은 전체 이미지를 여러 레이어층을 거쳐 하나의 벡터값을 가지는 요소들로 만들어주기 때문에 마지막 출력 레이어는 1x1x10(10은 CIFAR-10 데이터의 클래스 개수)의 차원을 가지게 된다. Convolutional net을 이루는 layer들을 각각 살펴본다.

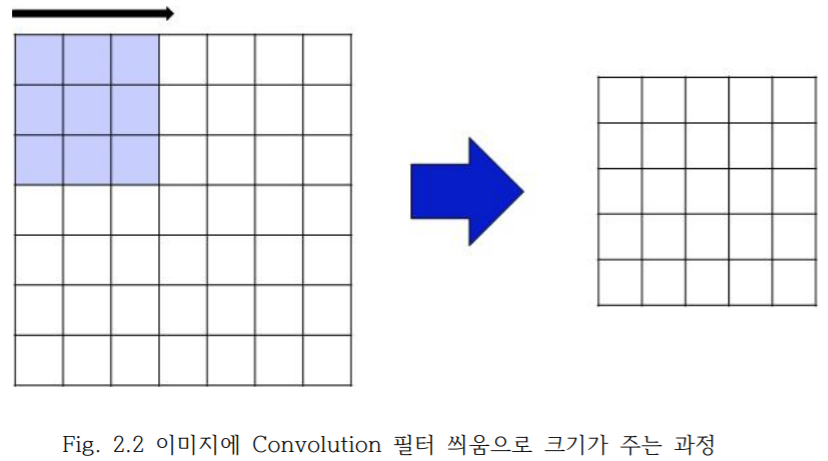
# 필터

## 앞서 특징을 추출함에 있어 특징이 데이터에 있는지 없는지를 검출해 주는 함수이다. 예를들어 글씨체를 구별하는 경우 흰 바탕에 검은 가로줄이 있는지 여부를 판단해주는 함수를 행렬로 나타낼 수 있다. Pixel 로 이루어진 이미지를 필터링 하는 경우 특정 행만 값을 주고 나머지 행의 값을 0 으로 두는 필터를 만들 수 있다. 입력받은 이미지 역시 Pixel 로 변환이 되면서 필터를 적용하면 필터값과 이미지의 값이 곱해져서 0인 부분만 제외하고 특징 부분만 큰 값으로 남게 된다. 즉 필터는 입력받은 데이터에서 그 특성을 가지고 있으면 결과값이 큰 값이 나오고, 특성을 가지고 있지 않으면 0에 가까운 값이 나오게 된다. 이렇듯 데이터가 그 특성을 가지고 있는지 없는지 여부를 알 수 있게 해 주는 것이 Filter 이다. 다른 말로 커널이라고도 한다. 입력 데이터에는 여러 가지 특징이 있기 때문에 여러 개의 필터를 같이 적용하게 된다. 각기 다른 특징을 추출하는 필터를 조합하여 네트워크에 적용하면, 원본 데이터가 어떤 형태의 특징을 가지고 있는지 없는지 판단할 수 있다. 하나의 글씨체 이미지 (인풋 이미지) 에 가로선과 세로선의 필터를 동시에 적용 (Convolutinal filter)하여 글씨체를 판별 할 수 있다.

2D이미지의 Convolutional 식C(x,y)는 입력된 이미지 I(x,y)와 커널필터 w(x,y)의 Convolutional 값이 된다. 

원래 컨벌루션은 W(x,y)를 180도 반전시킨 것으로 곱하지만 (W(-x,-y), 정방향으로 필터를 움직이므로 식은 이 된다. (cross correlation과정) 그러나 어느 방향이든 결과는 같다.





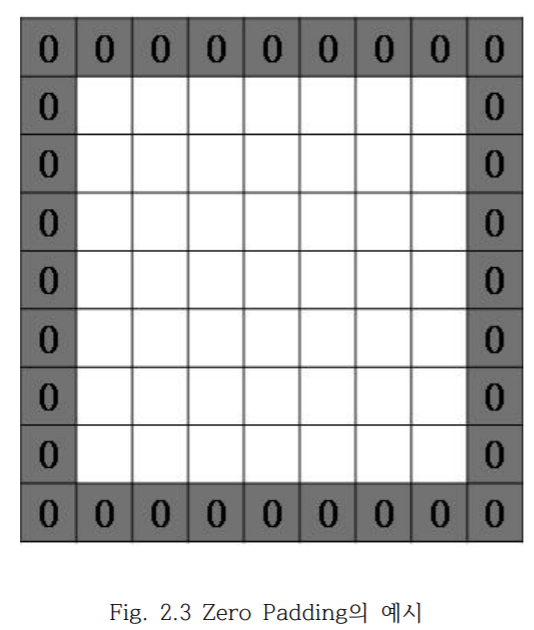
위의 그림과 같이 이미지보다 작은 크기의 필터를 씌우게 되면 원래 이미지가 줄어들게 된다. 줄어드는 정도는 Stride에 의하며, 줄어듦으로 인한 손실을 막기 위해 Padding 을 할 것이다. 아래는 Stride 와 Padding 에 대한 설명이다.

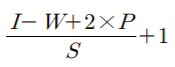
# STRIDE

## 필터를 원본 이미지에 적용하는 방법으로 사진 Pixel에 일대일 대응되는 필터 하나를 적용할 수 있지만, 그렇다면 필터가 너무 커지고 계산해야 하는 값 및 횟수도 한번 뿐이므로 정확도가 떨어진다. 따라서 필터를 작게 만들어 여러 번 겹치면서 스캐닝 하게 되면 더 정확도가 올라가게 된다. 예를 들어 5\*5 원본 이미지에 3\*3 필터를 좌측 상단에서부터 왼쪽으로 하나씩 이동하면서 한줄 이동하고 그 다음줄을 또 왼쪽에서부터 한칸씩 이동하면서 특징을 추출해 낸다. 이렇게 필터를 적용하는 간격(우측으로 한칸, 아래로 한칸) 을 Stride라고 하고 필터를 적용해서 얻어낸 결과를 Feature map (Activation map ) 이라고 한다.

# Padding

## 원본 데이터에 필터를 적용하다보면 필터를 적용한 후의 결과값은 필터 적용전 보다 작아진다. 5\*5 원본 이미지가 3\*3 의 1stride 값을 가지고 적용되었을 때 그 결과값은 3\*3 으로 크기가 작아지기 때문이다. CNN네트워크가 하나의 필터 레이어가 아니라 여러 단계로 계속 필터를 적용하게 되므로 필터후 결과값이 작아지게 되면 그만큼 특징의 유실이 생길 수 있다. 가장자리의 정보 유실은 곧 성능의 하락을 의미한다. 이를 방지하기 위한 방법으로 Padding 이라는 기법을 이용하는 것이다. 입력 이미지 주위로 0 값의 테두리를 만드는 것이다. 입력값의 크기를 인위적으로 키워서 결과값이 작아지는 것을 방지할 수 있다. 5\*5 주위를 테두리에 2줄씩 0값을 넣는다면 7\*7 이 되고, 3\*3 필터를 적용하더라도 결과값은 5\*5 로 유지된다. 즉 특징Feature가 소실되는 것을 막는다. Padding은 결과 값을 작아지는 것을 막아서 특징이 유실되는 것을 막을 뿐 아니라 오버피팅도 방지하게 된다. 원본 데이터에 0값을 넣어 원래 특징을 희석화하고 이 희석화는 특정 값에만 트레이닝 되는 (오버피팅) 현상을 방지할 수 있기 때문이다.



Padding의 유무, Stride 의 개수, Filter의 크기에 의해 이미지의 사이즈는 다음과 같은 식으로 정해지며, 그 값은 정수로 제한된다. (I는 이미지의 크기, W는 필터의 크기, P는 패딩의 존재시1 부존재시 0, S는 stride의 크기) 

# Pooling Layer;

Convolutional layer 을 통해 추출된 특징들을 분류하는 layer 이다. Pooling 과정의 종류로서 Max pooling, Average pooling, Min pooling, Fraciton pooling 등 여러가지 방법이 있다. Figure1과 같이 2\*2 필터를 Stride2 로 한 Pooling 이다. Max Pooling 을 많이 사용하고 SubSamping 이라고 한다. Pooling 은 Convolutional 필터를 통해 얻은 이미지 특징들 중에서 더 중요한 값을 추출해 내는데 필요한다. 하지만 필터를 통해 축소된 이미지의 픽셀을 받아 반으로 줄여 깊은 층의 학습을 하게 되므로 작은 크기의 이미지에는 깊은 층의 학습을 시킬 수 없을 수 있다. 이러한 문제점을 해결하기 위해 정수배가 아닌 실수배로 서서히 픽셀을 줄어들도록 하는 Fractional Max Pooling 알고리즘이 제안되었다. (Graham, Benjamin)

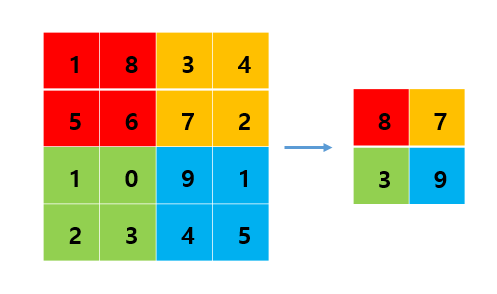


Figure 1. Max Pooling