THE SCALE FOR PIGS

- BY ITS IMAGE WITH DEEPLEARNING

EcoSystem(주)

* Contents

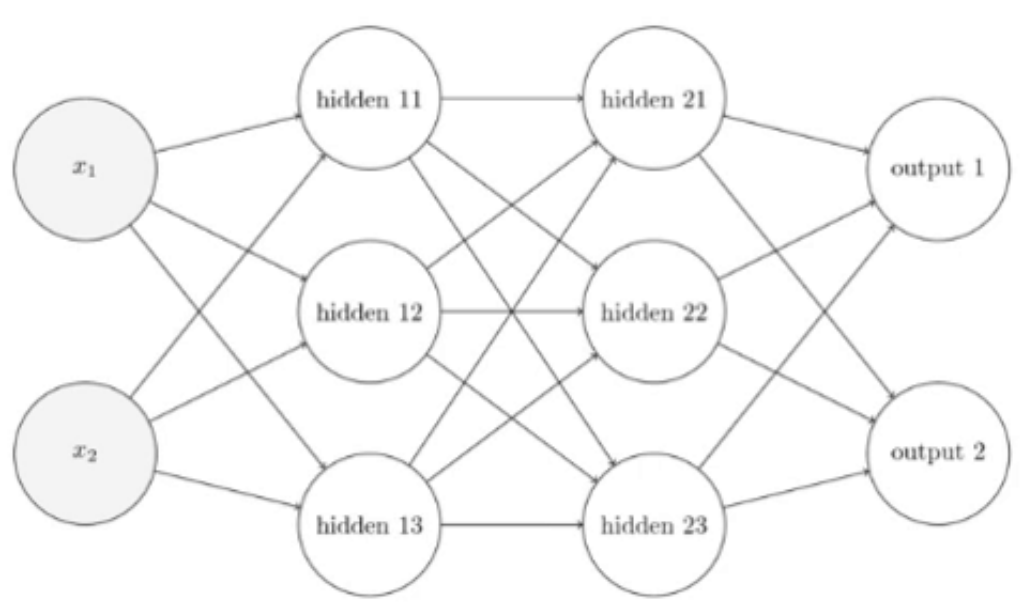
몇 년 전부터, Deeplearning 은 주목을 받았으며, 이를 이용한 많은 연구와 적용 사례가 쏟아 졌습니다. Deeplearning 구조는 인공신경망(ANN, artificial neural networks)에 기반하여 설계된 개념으로 역사를 따지자면 최소 1980년 Kunihiko Fukushima에 의해 소개 된 Neocognitron 까지 거슬러 올라갑니다. 최근 까지 Deeplearning이 관심을 받을 수 없었던 이유는 막대한 연산량을 처리할 컴퓨터가 없었고, 설사 하더라도, 많은 데이터를 사용하지 않으면 성과가 좋지 않고, 학습이 잘 안되었고, 너무 많은 시간을 필요로 했었기 때문입니다. 이 이외에도, 많은 이유로 Neural Network는 외면 받아 왔으나, 이 후, 컴퓨터의 발전과 더불어, GPU를 이용한 복잡한 행렬 연산이 가능해졌고, 데이터가 폭발적으로 증가 했으며, 제프리 힌튼을 필두로 구조의 문제점들이 해결되면서 지금과 같은 많은 관심을 받게 되었습니다.

컴퓨터 이미지 처리 분야에 초점을 맞추면, 이미지 분류, 물체 인식, 이미지의 의미 분할(Semantic Image Segmentation) 등 거의 모든 이미지 관련 문제를 해결하는데 Deeplearning이 사용 되고 있습니다. 그리고 기존에는 분석하는데에 있어, 이미지 데이터를 이용하지 않던 분야에서도, Deeplearning을 적용해보려는 움직임이 있습니다. 저희가 연구한 돼지의 **이미지**를 통해서 **돼지의 무게**를 예측하는 일 또한 그 움직임의 일환이라고 볼 수 있습니다. 이제 Deep learning의 기본 구조와 CNN, 그리고 저희가 연구한 모델에 대한 설명을 이어가겠습니다.

2. Work Explanation

- 심층 신경망 (Deep Neural Network)

심층 신경망은 여러 개의 층으로 이루어진 신경망을 의미합니다. 한 층은 여러 개의 노드로 이루어져 있고, 이 노드에서는 실제로 연산이 일어납니다. 이 연산 과정은 인간의 신경망을 구성하는 뉴런에서 일어나는 과정을 모사하도록 설계되어있습니다. 노드는 일정 크기 이상의 자극을 받으면 반응을 하는데, 그 반응의 크기는 입력 값과 노드의 계수를 곱한 값과 대략적으로 비례합니다. 일반적으로 노드는 여러 개의 입력을 받으며 입력의 개수 만큼 계수를 가지고 있습니다. 따라서 이 계수를 조절함으로써 여러 입력에 다른 가중치를 부여할 수 있습니다. 최종적으로 곱한 값들은 전부 더해지고 그 합은 활성 함수(비선형 함수)의 입력으로 들어가게 됩니다. 활성 함수의 결과가 노드의 출력에 해당하며 이 출력값이 궁극적으로 분류나 회귀 분석에 쓰이게 됩니다.



- CNN (Convolutional Neural Network)

위의 신경망 모형을 기초로 하여, 각 데이터의 특성에 따라 많은 아키텍쳐가 존재합니다. 이미지 처리 분야에서는 Convolutional Neural Network 가 가장 좋은 성능을 내고, 지배적으로 사용되고 있습니다.

- **구조**

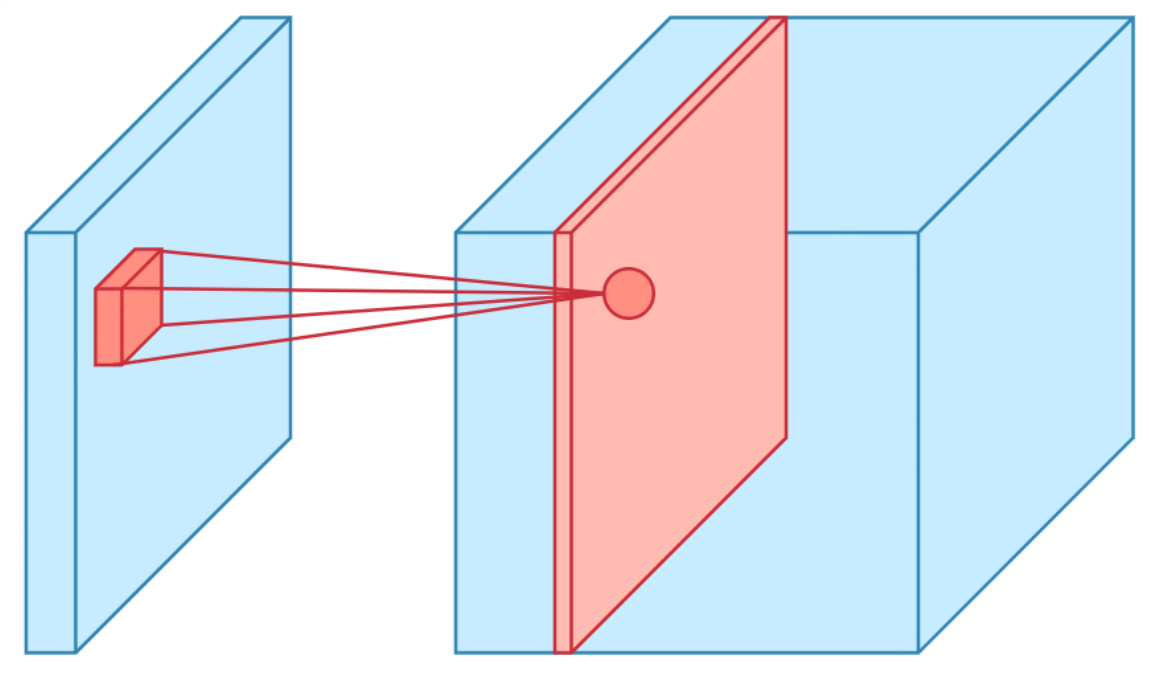
CNN의 기본적인 구조는 크게 **Convolutional Layers** 와 **Classification Layers**로 나눌 수 있습니다. Classification Layers는 실제로 분류 역할을 수행하는 layers를 말하는 것으로, 기본적으로는 일반적인 심층 신경망과 같은 구조를 가지고 있습니다.

- Convolution Layers

Convolution layers 는 인풋의 특징값들을 **여러개의 채널로 쌓는 작업**을 수행하여, 이미지가 담고 있는 가장 중요한 특징을 추출하는 단계입니다. 인간이 어떤 자극을 받을 때, 그 자극이 주어지는 점에서 가까울수록 그 자극은 크게 느끼는 것을 차용하여 만들어졌습니다. Convolution layers로 추출된 특징값들을 Classification Layer의 인풋을 넘겨주어 더 정교한 분류를 수행할 수 있게됩니다. 주요 개념으로는 Filter, Feature maps , Pooling 이 있습니다.

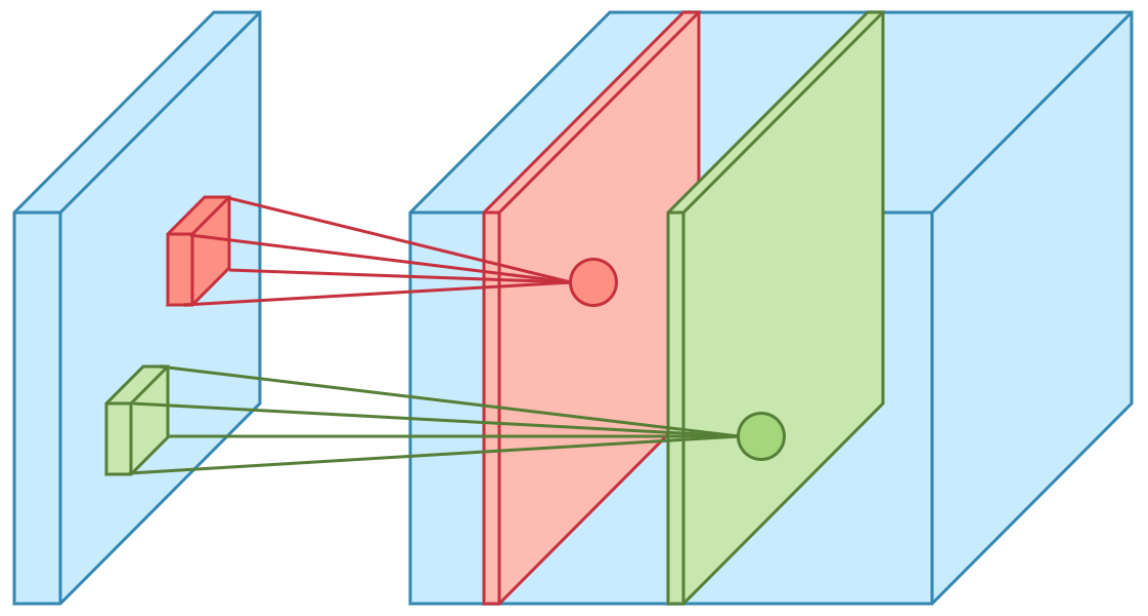
- **Filter**

  Filter는 이미지 크기보다 작은, 공유된 가중치를 가지는 정방행렬을 말합니다. 아래의 그림처럼 이 Filter가 이미지를 돌아다니면서, **한개의 숫자**를 만들고, 이 숫자가 그 부분의 이미지를 대표 할 수 있도록 학습하게 됩니다.



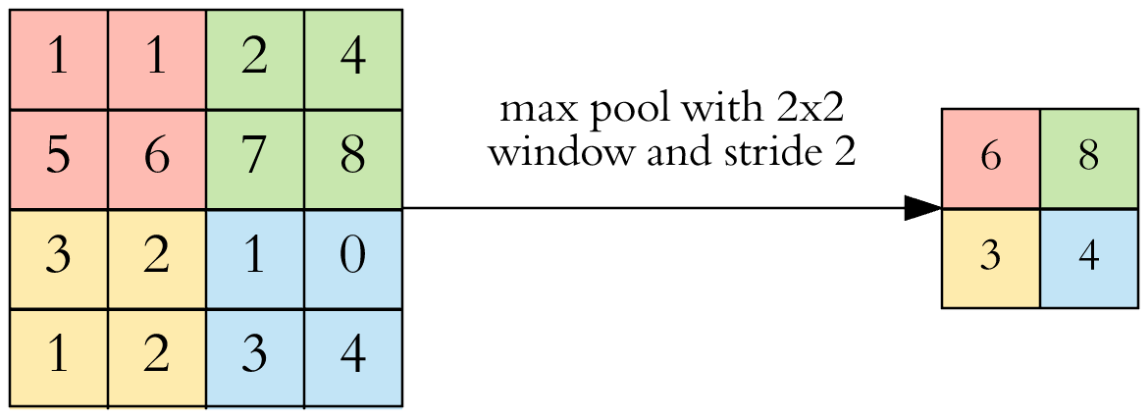
- **Feature maps**

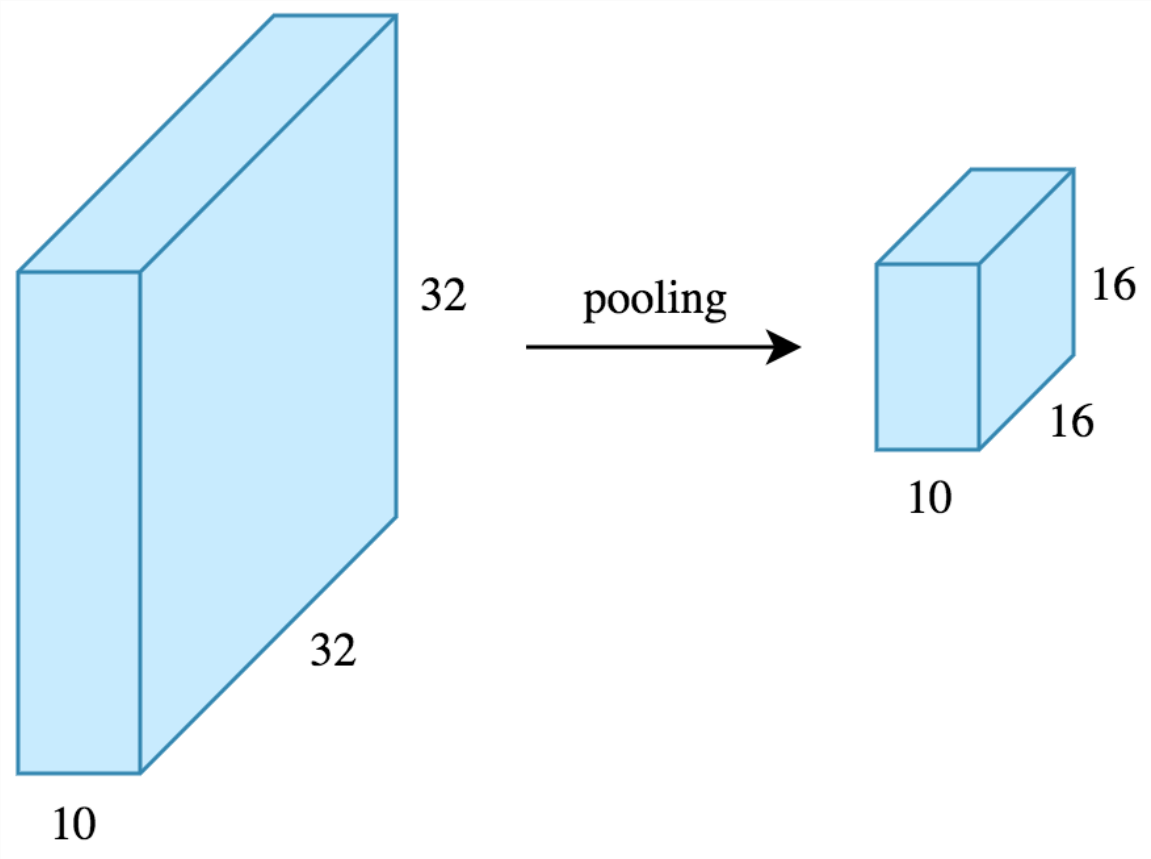
  Filter가 돌아다니며 추출한 숫자가 쌓여서 만든 또다른 이미지 한장을 Feature map이라고 합니다. Filter 하나가 하나의 Feature map을 만듭니다. 하나의 Filter를 통해 만들어진 하나의 Feature map은 이미지에서 어떠한 특징 하나가 강조되어 있는 그림이라고 할 수 있겠습니다. 여러개의 Filter가 만드는 여러 개의 Feature maps 들을 통해 이 이미지가 가지고 있는 중요한 몇 가지의 특징 정보를 수집할 수 있습니다.



- **Pooling**

  하나의 Featur map의 크기는 이미지(인풋)와 크기가 비슷합니다. 거기에다 여러개의 Feature map을 쌓아두었기 때문에, 연산해야하는 숫자가 많이 늘어 나게 됩니다. 그리고 Feature maps 에는 아직 그리 중요하지 않은 정보가 있을 가능성이 큽니다. 이 점을 보완하기 위해서, Feature map의 개수는 그대로 두고 그것의 사이즈를 줄이는 작업을 합니다. 이를 Pooling 이라고 합니다. 이 Pooling layer를 거치면 Feature map 하나하나의 크기는 Pooling size 에 반비례 해 작아집니다. 보통은 2x2의 Maxpooling 을 사용합니다. 이는 2x2의 구간에서 가장 큰 값만 남기고 나머지 숫자는 지우는 것을 말합니다. (아래의 그림에서 '32'가 feature map의 크기 이고 '10' 이 Feature map의 개수입니다.)





- Our Model

기본적인 CNN으로도 기존의 어떤 방법보다 이미지 관련 과업에 뛰어난 성능을 얻을 수 있었습니다. 그러나 여전히 문제점이 존재했고 속도나 빠른 연산 같이 개선해야 할 점들도 존재 했습니다. 이를 해결하기 위한 연구가 있었고, 지금도 활발히 이루어지고 있습니다. 그리고, 기존의 연구들은 이미지가 가지고 있는 **형태** 정보를 가지고 이미지를 분류하거나 이미지에서 특정 영역을 인식하는데 집중되어 있습니다. 이와는 다르게 저희가 원하는 Task는 이미지를 통해 어떠한 숫자 예측하는 것이였고, 추후에는 디바이스 레벨에서 실행되는 모델을 만들고 싶었기에 모델을 만들 때, 많은 연구와 실험이 필요했습니다.

- **목표**

모델 엔지니어링의 큰 목표는 **정확한 예측** 뿐만 아니라 **효율성** 또한 고려한 CNN model을 디자인하는 것입니다.

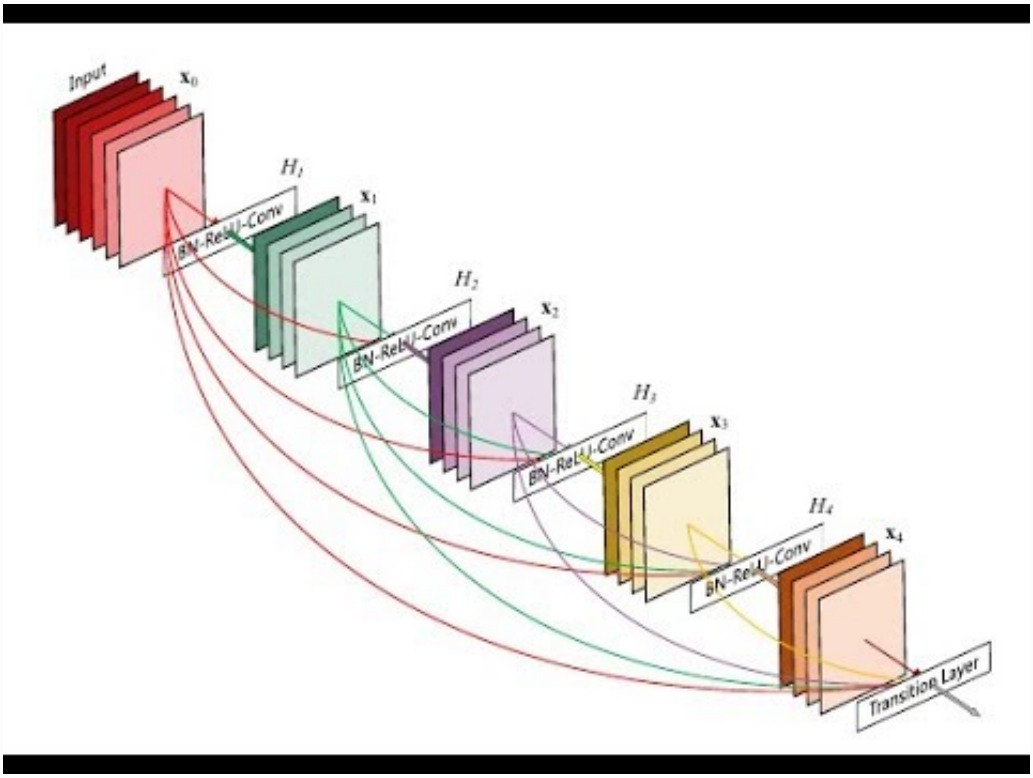
- **입력과 출력**

Input : 돼지의 사진 $\rightarrow$ (250, 550, 3) 의 컬러 이미지

Output : 돼지의 무게 (소숫점 2번째 자리 까지)

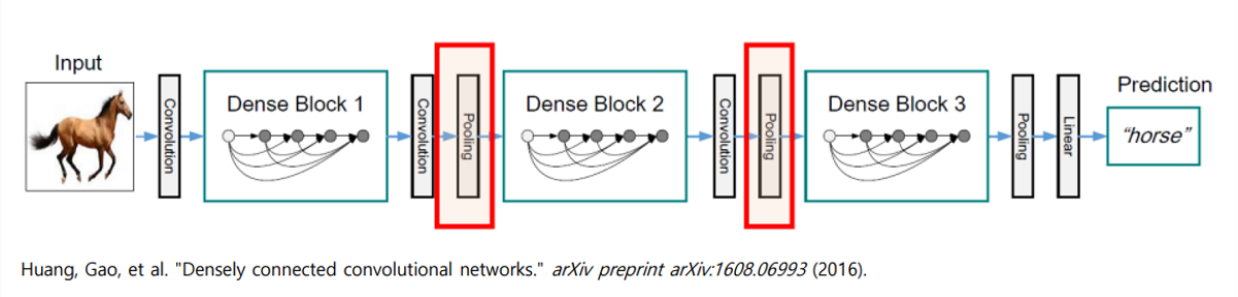
- **구조**

△ Dense Block



Dense block은 여러개의 Convolution layers로 이루어져 있습니다. 그러나 지나가는 동안 feature map의 크기는 동일합니다. 그리고 주목해야하는 점은 Convolution layer의 입력이 되기 전, 그 전의 Convolution layer를 통과 하기 전의 Feature maps와 그 Convolution layer를 통과한 후의 Feature maps를 쌓은 것을 다음 Convolution layer의 입력으로 줍니다. 이는 Dense block 안에서는 Feature maps 들의 사이즈가 동일하기에 가능합니다. 이렇게 하면, 적은 Feature map의 개수로 학습이 잘 되는 효과 있습니다.

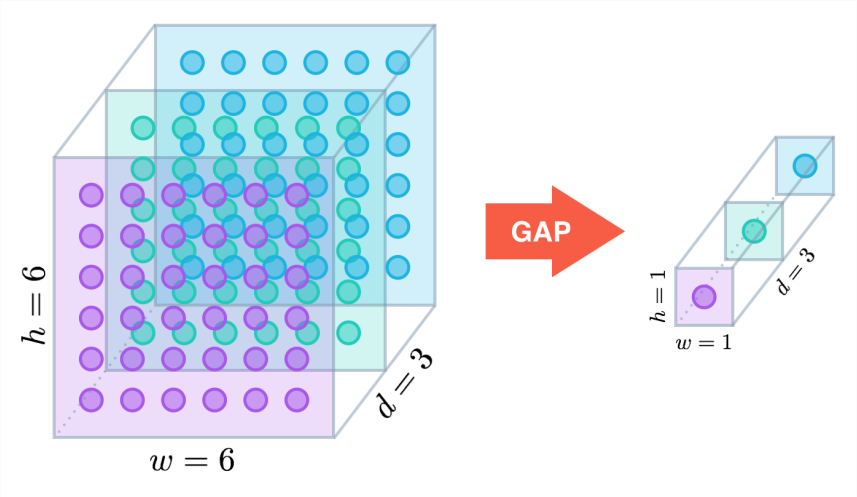
△ Transition Layer



Huang, Gao, et al. "Densely connected convolutional networks." *arXiv preprint arXiv:1608.06993* (2016).

DenseBlock으로 많아진 Feature map의 개수를 다시 더 줄이고, Feature map의 사이즈를 줄이는 단계입니다. 1x1 Convolution 을 통과 시켜, Feature map의 개수를 줄이고, Pooling으로 사이즈를 줄였습니다. 이 때는, 2x2의 Average Pooling을 사용했습니다. (위 그림의 빨간색 블럭)

△ Global Average Pooling



모델을 구성하며 두 개의 Dense Block 과 그 사이에 하나의 Transition Layer를 두어 이미지의 특징값을 효과적이며, 효율적으로 추출 했습니다. 이제는 실제로 무게를 예측하는 Regression layer(위에서는 Classification layer라고 표현 했던 부분 입니다. 이 모델의 목적은 실수값을 예측하는 Regression을 수행하는 문제이고 따라서 임의로 Regression Layer라고 표기 하겠습니다.)를 통과 시켜 실제 무게를 예측하는 부분으로 넘어갑니다. 보통의 경우에는 Feature maps 을 1차원의 벡터로 차원을 축소시켜 Dense layer의 각 노드의 인풋으로 넣어 줍니다.

그러나 저희 모델에서는 여기서 조금더 숫자의 개수를 줄이기 위해서(효율성을 위해서) Global Average Pooling 을 적용했습니다. Feature map 하나를 하나의 숫자로 만드는 방법입니다. 이 방법의 효과성은 여러 논문에서 입증이 되었습니다.

△ Regression layers

- Dense layer (200 node and Kernel regularized (l2=0.001))

- Activation Function : Elu function

⋆ elu(x) $= \exp(x) - 1, \ \{if\ x\ < 0\}$
$\qquad\qquad\quad x, \qquad \{if\ x \geq 0\}$

- **성과**

Test performance : 최소 MAE = 2.3618

2. Simulation

- Link

3. Resources

- Dataset

  - Company owned

- Server

  - Google Cloud Platform(two of Testla k 80 GPUs)

- reference

  - DenseNet

  - Xception

  - MobileNet

  - Global Average Pooling