

# ImageNet Classification with Deep Convolutional Neural Networks

## - 2021년 P-sat 논문스터디 6주차

김민정, 김원구, 김재희

2021년 2월 22일

'ImageNet Classification with Deep Convolutional Neural Networks' 논문을 바탕으로 Alex Net에 대한 전반적인 내용을 소개하는 자료입니다.

## 1. Introduction

### 1.1 등장배경

- 간단한 객체 인식(object recognition) 모델을 학습하기 위해서는 수 만장의 라벨링된 데이터셋으로도 데이터 증강 등을 이용하면 충분하다. 하지만, 현실의 데이터는 매우 가변적이기에 훨씬 큰 데이터셋은 불가피하다. 최근에는 대용량 이미지 데이터를 수집하기 수월해졌기에 100만 여 장의 LabelMe나 22,000 개 이상의 카테고리로 라벨링된 천 오백만 개의 고해상도 이미지 데이터셋인 ImageNet이 만들어질 수 있었다.
- 그러나 수 백만 장의 이미지에서 수 천 여개의 객체를 학습하기 위해서는 많은 양의 데이터만으로는 그 복잡성을 특정하기에 역부족이다. 따라서, 이미지 데이터에 대한 사전 지식을 녹여낼 수 있는 모델이 필요하다. 그리고 CNN은 통계량들의 정상성과 픽셀 간의 지역적 의존성을 고려할 수 있으므로 이에 적합한 모델이다. 이론적으로, 비슷한 크기의 FFNN에 비해 CNN은 파라미터 개수와 셀 간 연결이 훨씬 적어 학습이 쉽다.
- 물론 CNN 구조가 간단하더라도 대용량 고해상도 이미지를 다루는 것은 역시나 차원이 많이 소모된다. 본 논문에서는 이 문제를 GPU 병렬 처리를 통해 해결한다. 최근 GPU는 2D convolution 처리에 최적화 되었기에 매우 큰 CNN 모델을 심각한 오버피팅 없이 학습하기에 충분히 강력화하기 때문이다.

## 2. The Dataset

### 2.1 ILSVRC 개요

- ImageNet은 22,000여 개의 카테고리로 라벨링된 천 오백만 여개의 고해상도 이미지로 구성되며, AWS의 크라우드 소싱 툴(AMT)을 이용해 수집됐으며 사람이 라벨링했다. 이 논문에서는 ILSVRC(ImageNet Large-Scale Visual Recognition Challenge)-2010과 ILSVRC-2012로 실험을 했다. 전자는 test 셋을 사용할 수 있었으며, 후자는 대회 중이었기에 사용할 수 없었다고 한다. ILSVRC은 1000여 개의 카테고리가 각각 1000여 개의 이미지로 분류된 데이터에 대한 대회이다.
- Data size of ILSVRC-2010 Train size : 1,200,000 Validation size : 50,000 Test size : 150,000
- ILSVRC은 top-1과 top-5라는 두 가지 error를 사용한다. top-1은 가장 가능성이 큰 레이블에 대한 에러이며, top-5는 상위 5개 예측 레이블에 대한 에러이다.

### 2.2 데이터 전처리

- ILSVRC 이미지는 해상도가 다양하다. 본 논문에서는 사진 해상도를 256 by 256로 통일하였다. 이미지에서 더 짧은 쪽의 길이가 256이 되도록 리스케일링한 후 중앙을 cropping했다. 이후 각 픽셀에서 training set에 대한 mean을 빼 센터링하여 센터링된 RGB를 사용했다.

### 3. The Architecture

- 모델 구조는 크게 8개의 layer이며, 5개의 convolutional layer와 3개의 FC layer로 구성된다. 이후 나올 하위 4개 절은 중요도 순이다!

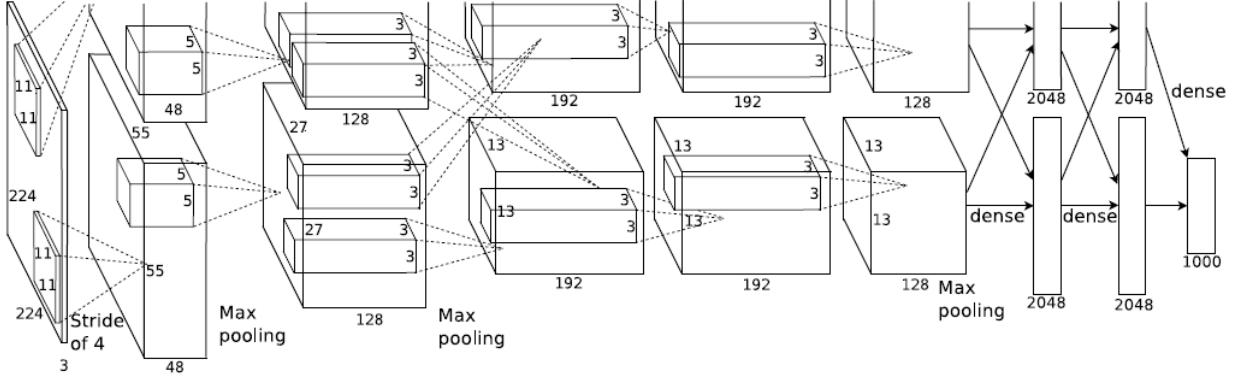
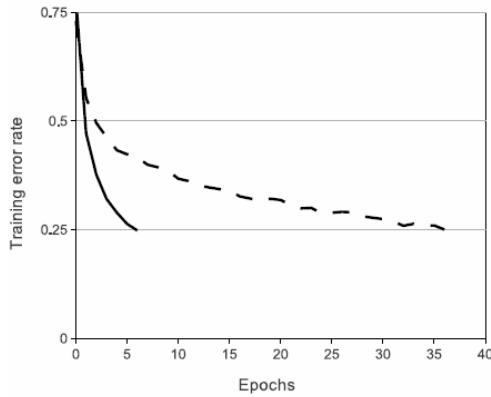


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

#### 3.1 ReLU Nonlinearity

- 출력층 활성화 함수의 표준적인 방법은 tanh나 sigmoid를 사용하는 것이었지만, (이러한 saturating non-linearity는) 경사 하강법에 대한 시간이 오래 걸린다. 따라서, relu(와 같은 non-saturating nonlinearity)를 사용하는 것이 효율적이다.  
cf) saturating nonlinearity 함수는 어떤 입력  $x$ 가 무한대로 갈 때 함수값이 어떤 범위 내에서만 움직이는 것을 의미  
non-saturating nonlinearity 함수는 어떤 입력  $x$ 가 무한대로 갈 때 함수값이 무한대로 가는 것을 의미
- 본 논문에서는 4개의 convolutional layer를 갖는 모델로 간단한 실험을 했는데, error가 25% 이하로 떨어질 때까지 걸리는 iteration이 relu가 훨씬 적었다.



- 물론 기존 활성화 함수를 개선한  $|tanh(x)|$  같은 비선형 함수에 대해서도 실험해봤지만, ImageNet 데이터의 관건 중 하나는 오버피팅을 방지하는 것이므로 relu가 이에 효과적이기에 relu를 사용한다.

### 3.2 Training on Multiple GPUs

- 본 실험에서 사용한 GPU는 GTX 580이며 3GB 메모리를 갖는다. 하나의 GPU로 120만 개의 사진을 다루기에는 버거웠다. 따라서 GPU 2개를 병렬 처리했으며, 각 GPU가 커널의 절반씩 처리하도록 했다. 결과적으로, 시간도 약간 감소했다. 한편, 여기에 하나의 트릭을 더했는데, 두 GPU는 하나의 레이어에서만 커널을 공유한다. 예컨대, 레이어3이 레이어2의 모든 kernel에서 인풋을 받지만, 레이어4는 레이어3의 같은 GPU 상의 kernel만을 인풋을 받는다. 공유 커널을 정하는 것은 CV로 정했지만, 비용이 많이 들었다...

### 3.3 Local Response Normalization

- 물론 ReLU 함수는 normalization 없이도 학습이 잘 이루어지게 됩니다. 만약 특정 weight의 기울기가 0에 근접해버리면 더이상 학습이 이루어지지 않을 것이다(논문에서는 saturating이라고 표현). 하지만 ReLU 함수의 특성 상 그럴 일이 잘 없기 때문에 입력값을 normalization할 필요가 없다. 하지만 논문에서는 여전히 일반화를 잘 시키기 위해 local normalization을 시도했다. 그 식은 다음과 같다.

$$b_{x,y}^i = a_{x,y}^i / \left( k + \alpha * \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} a_{x,y}^j \right)^\beta$$

- 위 식에서 alpha, beta, n, k는 모두 하이퍼 파라미터이다. 그리고

$$a_{x,y}^i$$

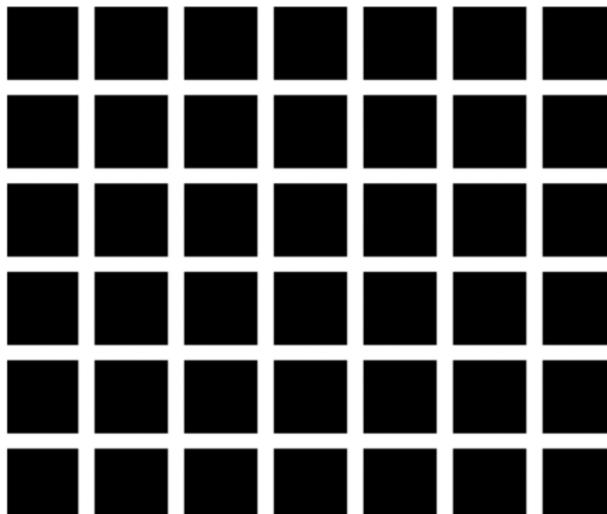
이 의미하는 것은  $(x,y)$  위치의  $i$ 번째 커널의 값이 ReLU 함수를 통과한 이후를 의미한다.  $j$ 는  $i$ 번째 커널과 인접한 다른 커널을 의미한다(앞 뒤 커널).

- 즉, ReLU함수를 통과시킨 후, 다시 한번 위의 식을 통과시키는 것이다. 위 식을 통과하면 해당 커널의  $x,y$ 의 값은 결국 근처 커널의 동일 위치의 값들의 합들로 해당 값을 나누는 것이 된다.
- 이렇게하면 각 커널의 값은 인접 커널의 값과 어느정도 유사해지게 된다. 그리고 학습 과정에서 lateral inhibition(측면 억제, 신경 생리학의 용어로, 이웃 신경 세포가 너무 뛰지 않도록 만드는 현상. 실제로 위의 식은 신경생리학에서 영감을 받았다고 합니다.)을 이를 수 있다고 한다.
- ReLU 함수는  $x$ 값이 크면 대체로 그대로  $y$ 로 전달한다( $x>0$ 일때 항등함수이기 때문에). 이는 특정 커널, 특정 위치의

$$a_{x,y}^i$$

가 너무 클 경우 다른 커널들이 제대로 학습되지 못하게 만든다. 주변 커널의 값은 상대적으로 작기 때문이다. 하지만 위와 같은 식을 이용하면 커널을 통과한 값들이 어느정도 normalization 되어 이러한 현상을 억제할 수 있다.

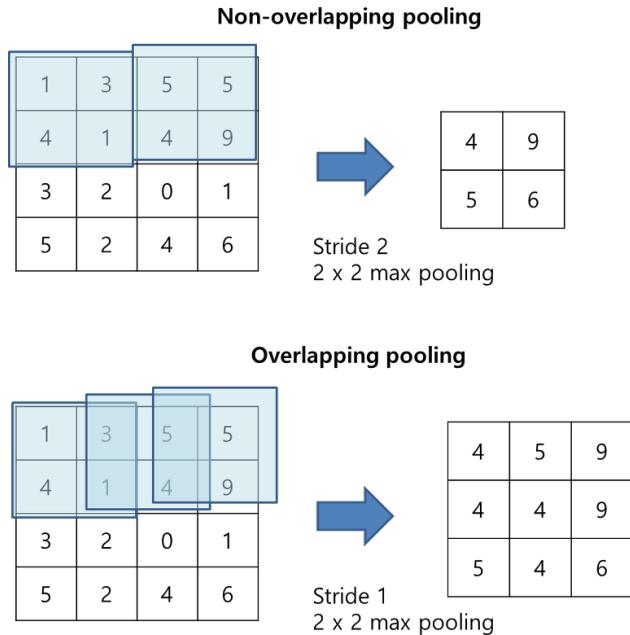
- 실제로 논문에선 이를 통해 top-1, top-5 error를 각각 1.4%, 1.2% 감소시켰다고 한다.



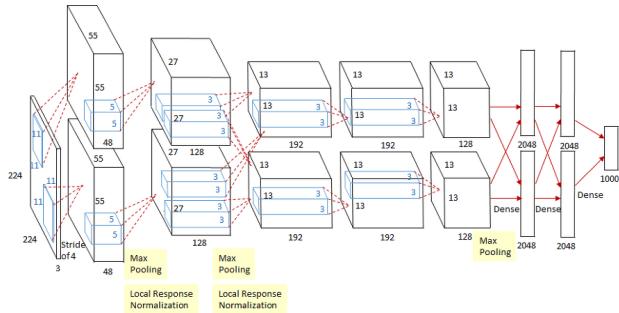
- 심심해서 넣어본 측면 예제 현상 예시. 흰색을 받아들이는 시신경들의 값이 너무 커서 normalization되니 겹은 점들이 보인다고 한다.

### 3.4 Overlapping Pooling

- 본래 Pooling Layer는 서로 겹치지 않는다. 각자 격자 형태로 값을 산출하는데, 이 논문에선 서로 겹쳤다. 아래 그림에서 위에 것이 일반적인 pooling이고 아랫것이 논문에서 사용한 overlapping pooling이다. 논문에선 overlapping pooling을 통해 top-1, top-5 error가 각각 0.4%, 0.3% 줄었다고 했다. 그리고 학습 과정을 관찰한 결과 오버피팅이 잘 걸리지 않는 모습 또한 보였다고 한다.



### 3.5 Overall Architecture



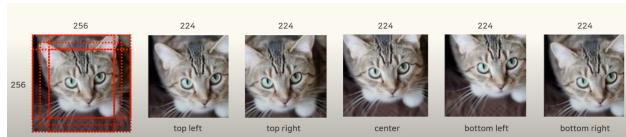
- 지금까지 살펴본 모델의 전체 구조를 살펴보면 다음과 같다. 모델은 두개의 GPU를 사용하여 위와 아래의 layer가 각각의 GPU에 의해 연산된다. 모델은 총 8개의 레이어로 구성되어 있다. 초기 레이어 5개는 Convolution layer이고 뒤의 3개는 Fully-Connected layer이다. 마지막 Fully-Connected layer를 통과한 값은 1000-way softmax를 거쳐 1000개의 클래스 중 하나로 분류한다.
- 2번째, 4번째, 5번째 convolution layer는 특이하게도 동일한 GPU에서 작업한 직전 convolution layer와 연결되어 있다. 3번째 convolution layer는 GPU에 관계없이 모든 직전 layer와 연결되어 있다. 또한 Local Response Normalization과 max pooling은 1번째, 2번째 convolution layer 뒤에 붙어있다.

## 4. 오버피팅 줄이기

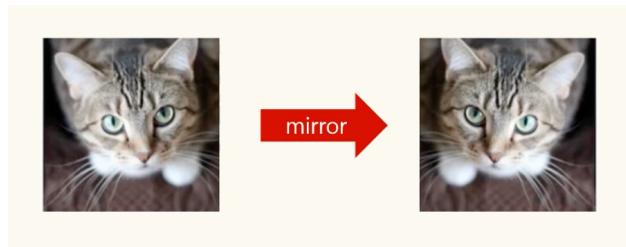
- 이 논문에서 소개하는 모델은 6,000만 개의 파라미터를 가지는데, 이렇게 많은 파라미터들을 오버피팅 없이 학습하기란 쉽지 않다. 따라서 이 논문에서는 오버피팅을 줄이는 2가지 방법을 소개한다.

### 4.1 Data Augmentation(데이터 증가시키기)

- 이미지 데이터에서 오버피팅을 줄이는 가장 쉬운 방법은 label은 보존하면서 데이터셋을 인공적으로 증가시키는 것이다. 이 논문에서는 2가지 방법을 적용을 했는데, 두 가지 방법 모두 원래 이미지에 약간의 연산만으로 변형을 가하는 것이다. 변형된 이미지를 따로 디스크에 저장할 필요 없이 GPU가 이전의 배치 이미지에 대해서 학습을 하는 동안 CPU에서 Python 코드로 구현된다. 따라서 이 방식은 효율적이면서 연산의 부담이 없다.
- 첫 번째 Data Augmentation 방법은 cropping과 reflection이다.



- 먼저 cropping은 이미지를 자르는 것이다. Training을 할 때는 256x256 이미지에서 224x224만큼씩 랜덤하게 잘라내어 데이터를 생성하였고, 이를 통해 한 장의 학습 이미지로부터 2048장의 다른 이미지를 얻을 수 있었다.



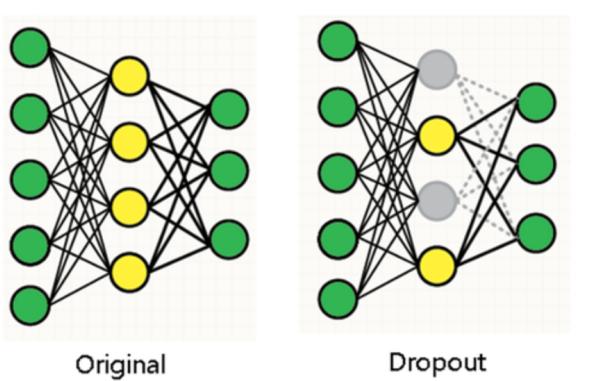
- Test를 할 때는 상,하,좌,우, 및 중앙을 224x224로 잘라내어 5장의 이미지를 생성하고 위의 그림처럼 그 5개의 이미지를 좌우 반전 하여 총 10장의 이미지를 생성한 다음 softmax 출력을 평균하는 방법을 채택하였다.
- 두 번째 Data Augmentation 방법은 RGB channel의 값을 변화시키는 것이다. 이를 위하여 학습 데이터의 RGB값에 대한 주성분 분석을 시행하였으며, 거기에 평균은 0, 표준편차는 0.1을 갖는 랜덤 변수  $\alpha$ 를 곱하고 그것을 원래 픽셀 값에 더해주는 방법을 시행하였다. 따라서 RGB channel의 값은 다음과 같다.

$$I_{xy} = [I_{xy}^R, I_{xy}^G, I_{xy}^B] + [p_1, p_2, p_3] [\alpha_1 \lambda_1, \alpha_2 \lambda_2, \alpha_3 \lambda_3]$$
$$\alpha_i \sim N(\mathbf{0}, \mathbf{1})$$

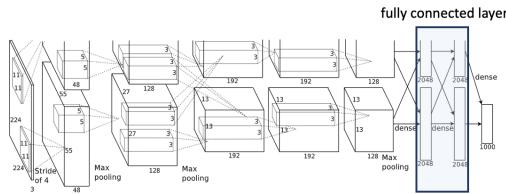
- 여기서 주황색 부분은 원래 RGB값이며 노란 부분이 주성분 분석에 랜덤 변수를 곱해준 것이다.  $p_i$ 는  $3 \times 3$  공분산 행렬의 고유벡터이고,  $\lambda_i$ 는 고유값이다. 초록색으로 표시한  $\alpha_i$ 는 앞서 언급한 랜덤 변수이며 매 epoch 마다 바뀐다. 이러한 방식은 내추럴 이미지의 중요한 특징을 가져온다. 즉, 이미지의 아이덴티티는 조명에 의한 강도나 색깔의 변화와는 무관한 것이다. 이 방식을 통해 top-1 error를 1%까지 줄인다.

### 4.2 Dropout

- 여러 모델을 결합하는 것은 test error를 줄이는 성공적인 방법이지만 큰 neural network에서는 학습시키는데만 며칠이 걸리기 때문에 비용이 너무 크다. 여기서 소개하고자 하는 “dropout”은 모델을 조합하는데 비용이 딱 2배만 드는 아주 효율적인 방법이다.



- 그림처럼 50%의 확률로 은닉층 뉴런을 “dropout”시키면 이 뉴런은 출력값이 0으로 설정되면서 순전파, 역전파에서 모두 아무 기여를 하지 못한다. 따라서 매번 input값이 들어갈 때마다 신경망 네트워크의 구조가 바뀐다. 이러한 기술은 뉴런의 co-adaptation 문제를 줄여주는데 co-adaptation이란 어떤 뉴런이 다른 특정 뉴런에 의존적으로 변하는 현상을 말한다. 구체적으로 설명하자면 신경망의 학습 중 어느 시점에서 두 개 이상의 노드의 입력 및 출력 연결강도가 같아지면 아무리 학습이 진행되어도 그 노드들은 같은 일을 수행하게 되어 불필요한 중복이 생기는 문제를 말하는 것이다. 결국 여러 개의 노드가 하나의 노드로 작동함으로써 낭비가 발생하게 된다. 하지만 dropout을 적용하면 한 뉴런이 다른 특정 뉴런에 의존할 수 없게 되어 co-adaptation 문제를 줄일 수 있는 것이다.



- 이 논문에서는 첫 번째, 두 번째 fully-connected layer에 dropout을 적용했고, dropout을 적용하지 않았을 때는 상당한 오버피팅이 발생했다고 한다.

## 5. Details of learning

- 본 실험에서는 optimizer로 SGD를 사용했고, 모멘텀은 0.9이며, weight decay는 0.0005이다. batch size는 128로 정했다. 실험에서 이렇게 작은 weight decay가 중요함을 발견했는데, regularizer 역할뿐만 아니라 training error 또한 감소시킨다. 개선된 식은 다음과 같다.

$$v_{i+1} := 0.9 \cdot v_i - 0.0005 \cdot \epsilon \cdot w_i - \epsilon \cdot \left\langle \frac{\partial L}{\partial w} \mid w_i \right\rangle_{D_i}$$

$$w_{i+1} := w_i + v_{i+1}$$

$i$  : iteration index

$v$  : momentum variable

$\epsilon$  : learning rate

$\left\langle \frac{\partial L}{\partial w} \mid w_i \right\rangle_{D_i}$  :  $w_i$ 에서 평가된  $i$ 번째 batch  $D_i$ 의  $w$ 에 대한 objective 미분값의 평균

- 실험에서 각 레이어의 가중치를  $\mathcal{N}(0, 0.01^2)$ 를 따르도록 초기화했고, 2, 4, 5번째 convolutional layer와 모든 FC layer의 bias를 1로 고정했다. 이러한 초기화는 학습 초기 단계에서 relu에 양수를 input으로 제공하도록

한다. (그래야 역전파 잘 돼서 학습 잘 됨) 이외의 레이어에서는 bias를 0으로 고정했다.

- 또한 모든 레이어에서 lr를 0.01로 통일했으며, validation error가 개선되지 않을 때 lr를 1/10하는 휴리스틱한 방법을 사용했다. 종료 전에는 3배 감소시킨 lr을 적용했다. epoch는 90회 반복했다.

## 6. Results

- ILSVRC-2010에 대한 실험 결과는 아래와 같다. top-1 error는 37.5%, top-5 error는 17%로 해당 대회의 1등이었던 Sparse coding 보다 결과가 좋다.

Model	Top-1	Top-5
<i>Sparse coding [2]</i>	47.1%	28.2%
<i>SIFT + FVs [24]</i>	45.7%	25.7%
CNN	<b>37.5%</b>	<b>17.0%</b>

Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.

- ILSVRC-2012의 결과는 아래와 같다. ILSVRC-2012는 실험 당시에 test set이 공개되지 않았었다. 하지만 경험에 비추어 볼 때 val error와 test error가 0.1% 이상 차이가 나지 않기 때문에 둘을 서로 교환해서 사용하도록 한다. 5개의 CNN 평균이 16.4%, ImageNet2011 Fall을 pre-train model로 사용한 7개 CNN 평균이 15.3%으로 나왔다.

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
<i>SIFT + FVs [7]</i>	—	—	26.2%
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	<b>16.4%</b>
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	<b>15.3%</b>

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In *italics* are best results achieved by others. Models with an asterisk\* were “pre-trained” to classify the entire ImageNet 2011 Fall release. See Section 6 for details.

- 마지막으로, 10,184개 카테고리의 890만 개 이미지를 가진 ImageNet2011 Fall에 대해서도 실험했다. 여기서는 training과 test를 5:5로 분리했다. top-1과 top-5 결과는 각각 67.4%, 40.9%로 나왔는데 그 당시 최저 error는 78.1%, 60.9%였다.

### 6.1 Qualitative Evaluation

- 아래 그림은 two data-connected layers를 가진 networks로 학습된 convolutional kernels다. 네트워크는 다양한 색 부분뿐만 아니라 다양한 주파수 및 방향 선택에 대한 kernel을 학습했다.

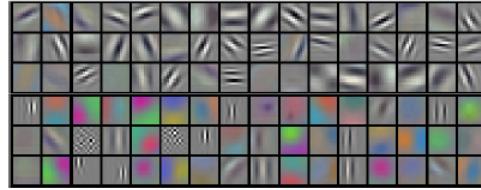


Figure 3: 96 convolutional kernels of size  $11 \times 11 \times 3$  learned by the first convolutional layer on the  $224 \times 224 \times 3$  input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2. See Section 6.1 for details.

- 본 논문에서는 앞서 언급했듯이 2개의 GPU를 사용했는데, GPU1은 색깔과 상관없는 정보, GPU2는 색깔에 관한 정보에 대한 kernel을 담당한다. 이러한 분업화는 매 학습마다 적용되며, 가중치 초기화와는 독립적이다. 이전 그림에서 윗부분의 48개는 GPU1의 결과이고, 아랫부분 48개는 GPU2의 결과다.
- 아래 그림의 왼편은 8개의 test 이미지에 대한 top-5 예측 결과이다. 대부분의 결과가 리즈너블하고, 심지어 진드기(mite, 1행1열)의 경우 일부분임에도 인식에 성공했다.

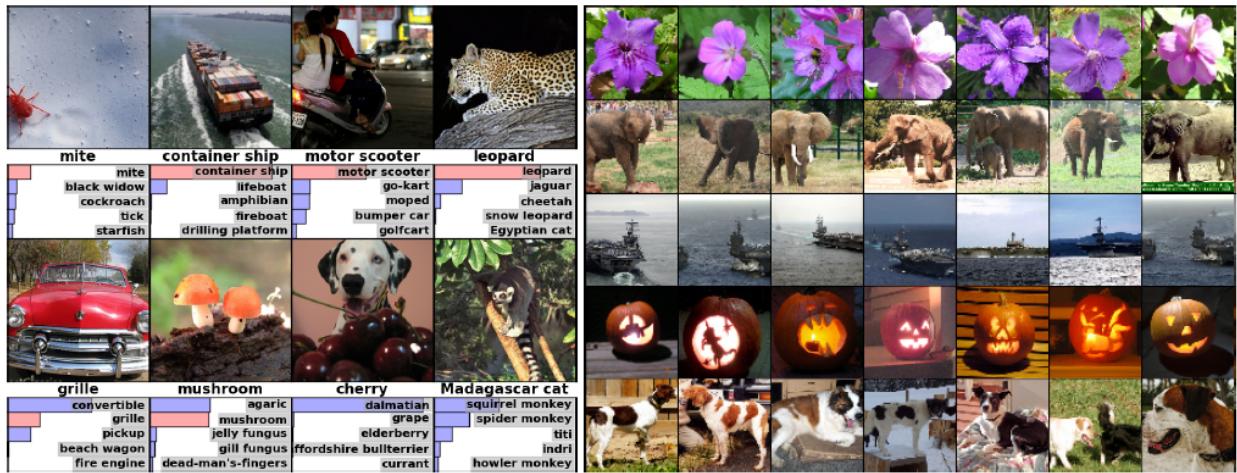


Figure 4: (Left) Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). (Right) Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.

- 이뿐만 아니라 비슷한 사진끼리는 히든 레이어 가중치 벡터 간의 거리가 짧을 것으로 이를 검증했다. 위 그림의 오른편이 이에 대한 결과다. 1열의 5개 사진들은 test set의 사진들이고, 나머지 열들의 각 6장의 사진들은 training set의 사진이며 각 행들이 같은 사진임을 알 수 있다. (그런데 2~7열 사진들은 피셀 수준에서 일반적으로 1열의 이미지와 가깝지는 않다)