Assignment 2

Siwon Kim

Electrical and Computer Engineering
Seoul National University

http://data.snu.ac.kr

Assignment Objectives

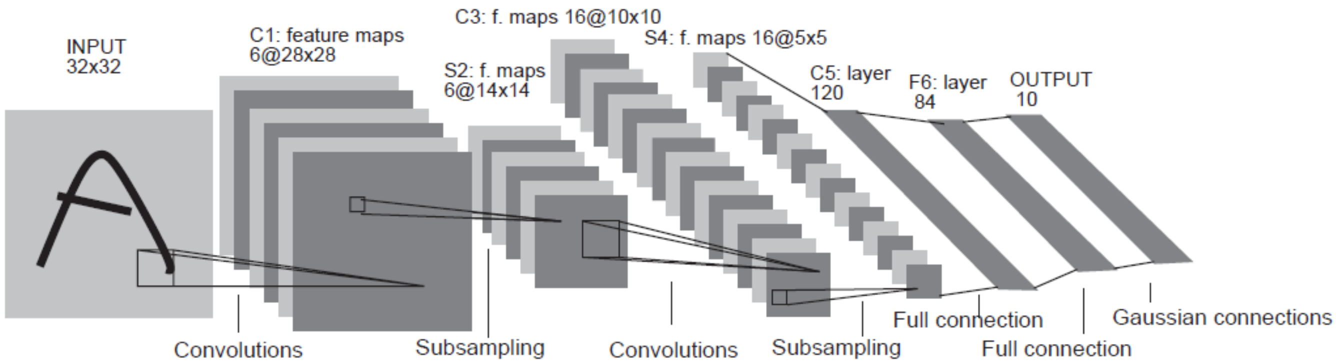
- Part 1: Implementing CNN
  - To understand CNN architecture before using the TensorFlow
  - Implement forward / backward passes for (1) convolution layer and (2) max pooling layer
- Part 2: Training CNN
  - Learn how to define, train, and evaluate CNNs with TensorFlow
  - Explore various hyperparameters to design a better CNN model
- Part 3: Visualizing CNN
  - Learn how to visualize and interpret a trained CNN model
  - Implement the codes for generating (1) image-specific class saliency maps, (2) class representative images, and (3) adversarial examples

CIFAR10 Dataset

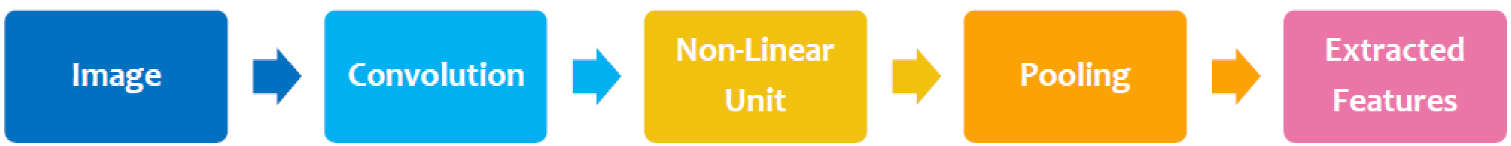
- Consists of 32x32x3 color images in 10 classes
- Training set: 40k instances — *Model training
- Validation set: 10k instances — *Model evaluation and selection
- Test set: 10k instances — *Final model testing

Convolutional Neural Networks (CNNs)

- Simply neural networks that use convolution in their layers



source: [LeCun et al., 2015b]



- Three key ideas behind CNN
  - Local connectivity
  - Invariance to location
  - Invariance to local transition

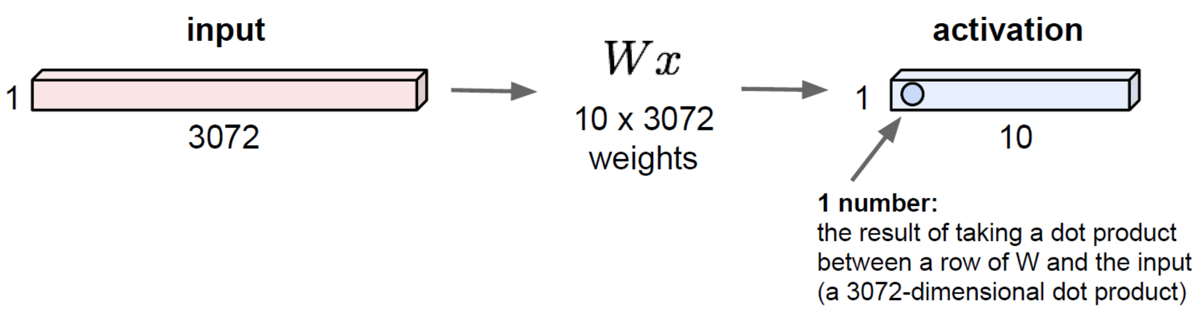
Convolution layer

Pooling layer

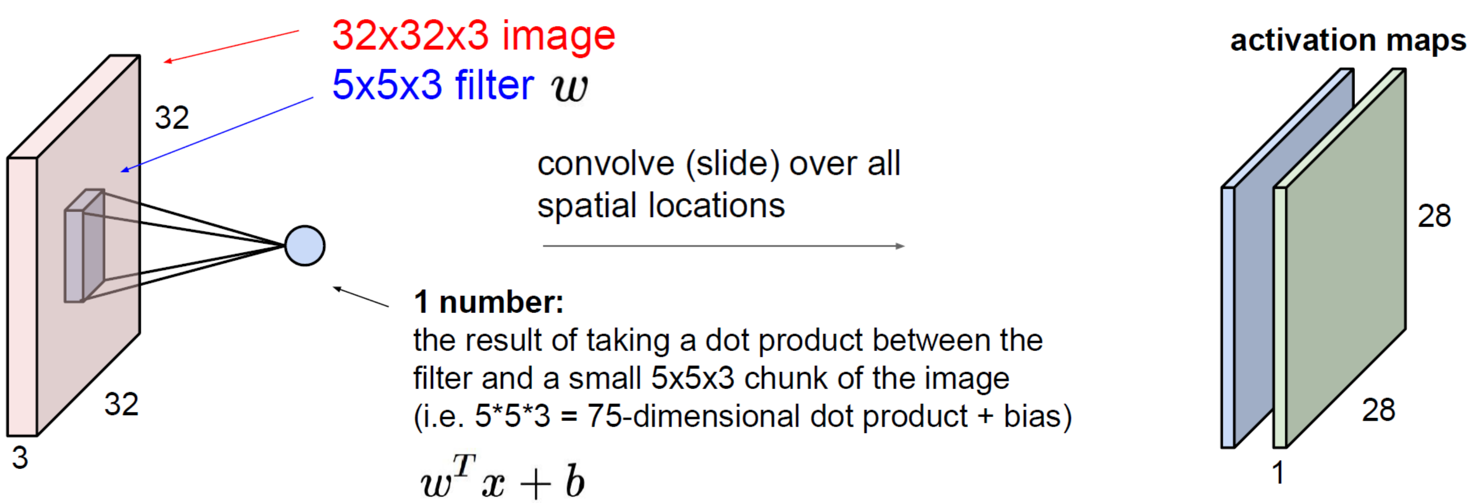
Fully Connected Layer and Convolution Layer

- Fully connected layer

  32x32x3 image -> stretch to 3072 x 1



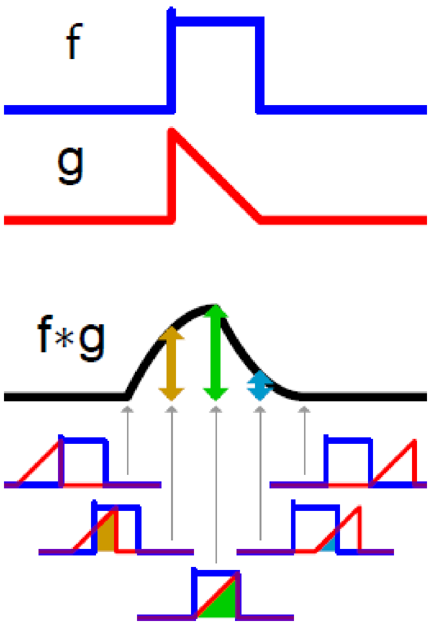
- Convolution layer



Convolution and Cross-correlation

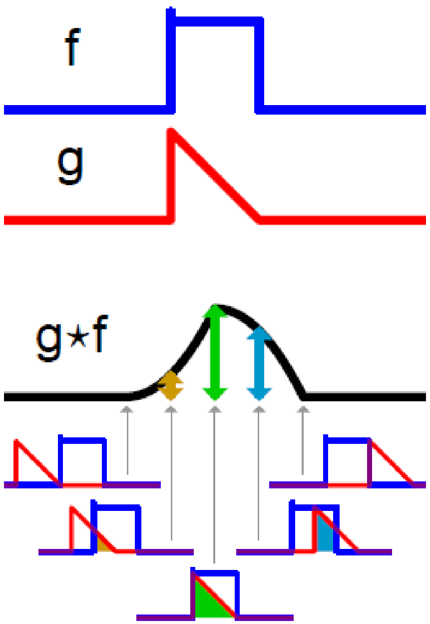
- Convolution

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, i-n)K(m,n)$$



- Cross-correlation

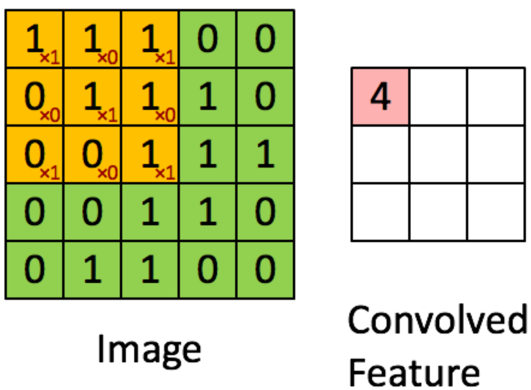
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

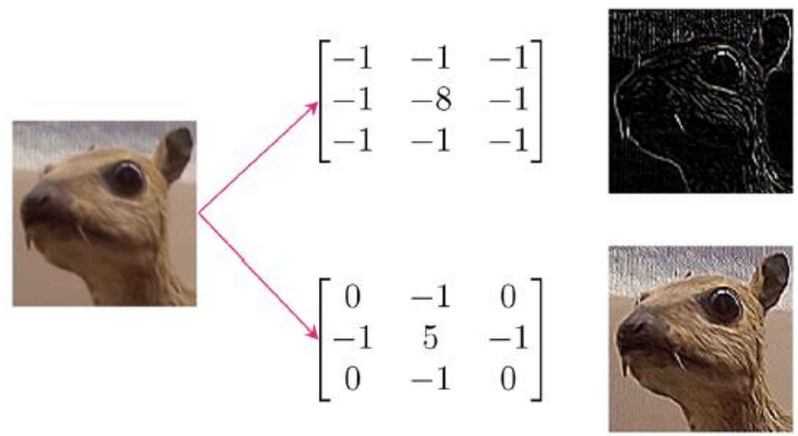


- Many NN libraries implement cross-correlation but call it convolution (without kernel flipping)

Convolution Layer

- Input is convolved with a set of filters, giving feature maps (without kernel flipping)

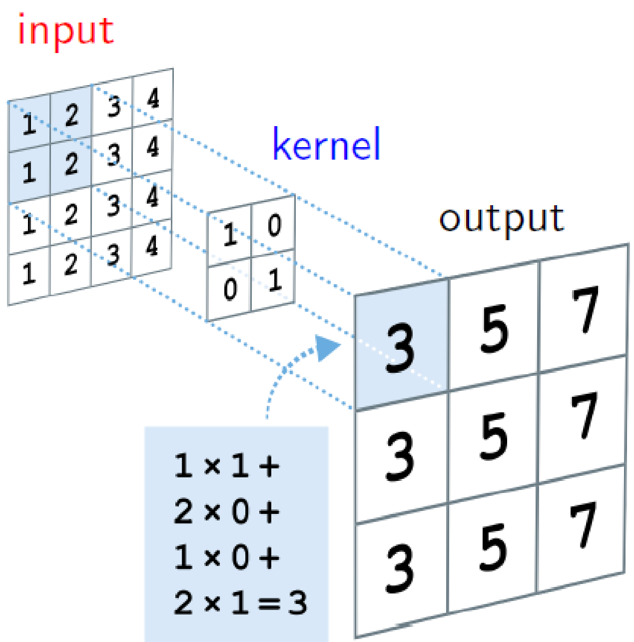




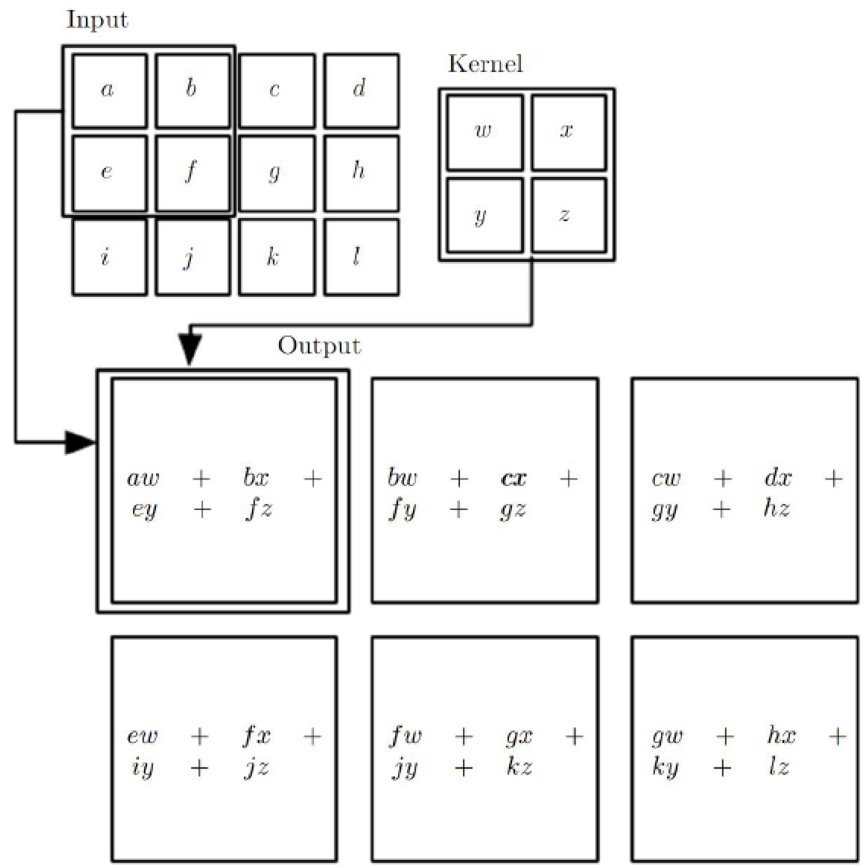
source: www.wikipedia.org

- Filter (aka kernel or convolution matrix)
  - Different filters extract different features (e.g. edges)
  - Filter weights: trained with data
  - Weight sharing & local connectivity

Convolution Example



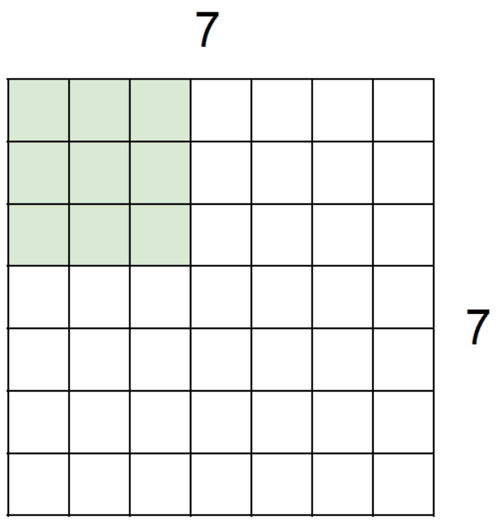
source: [Angermueller et al., 2016]



source: [Goodfellow et al., 2016]

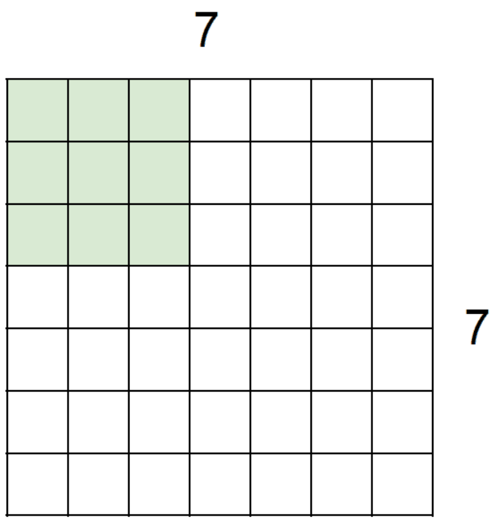
Convolution with Stride

- The number of pixels between adjacent receptive fields
  = down-sampling the output of full convolution function



7x7 input
3x3 filter with stride 1

➔ 5x5 output



7x7 input
3x3 filter with stride 2

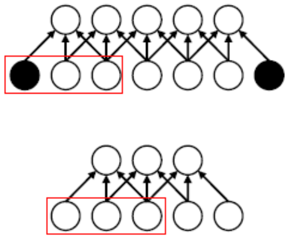
➔ 3x3 output

Zero-padding

- Implicitly zero-pad input to make it wider

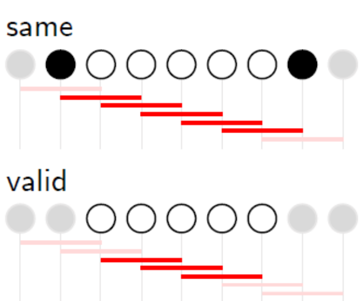
type	output		# zeros padded: left	right	total
same	m	k even	$\lfloor \frac{k-1}{2} \rfloor$	$\lfloor \frac{k-1}{2} \rfloor + 1$	$k-1$
		k odd	$\frac{k-1}{2}$	$\frac{k-1}{2}$	
valid	$m-(k-1)$		0	0	0

(m: input width; k: kernel width; $s=1$)

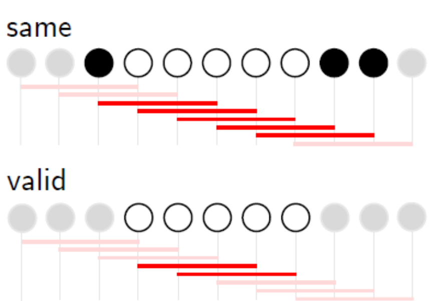


- Examples

$\boxed{m=5, k=3}$

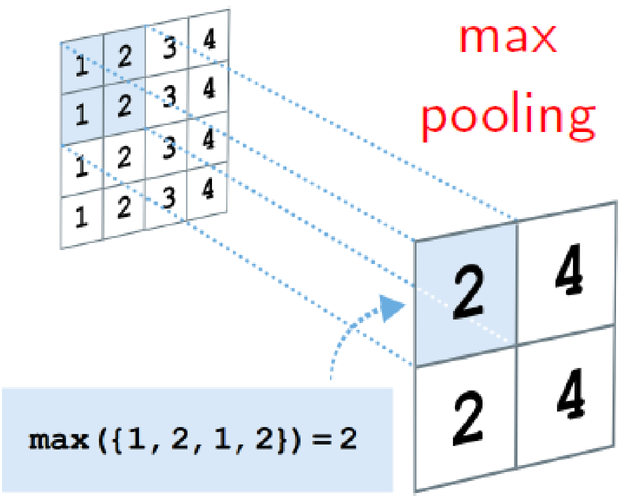


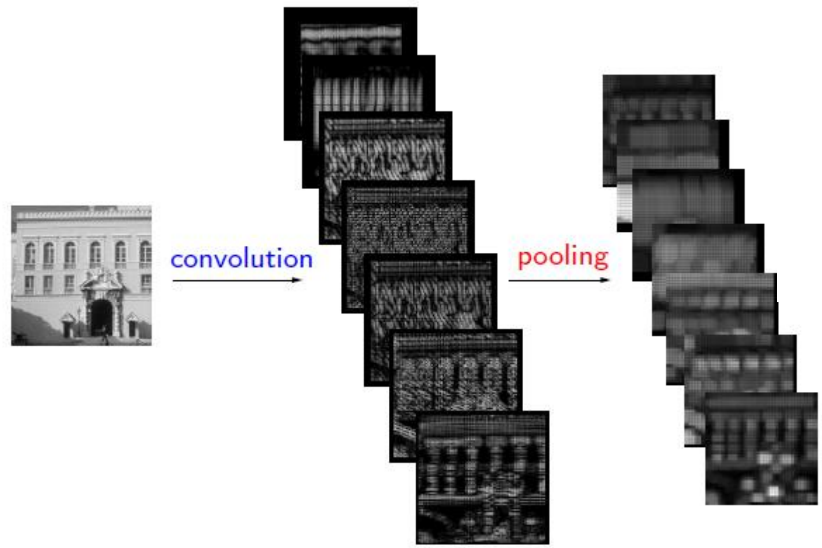
$\boxed{m=5, k=4}$



Pooling Layer

- Nonlinear down-sampling
  - Aggregates statistics of local features (with max or average operation)
  - Reduced variance: provides invariance to local transformations

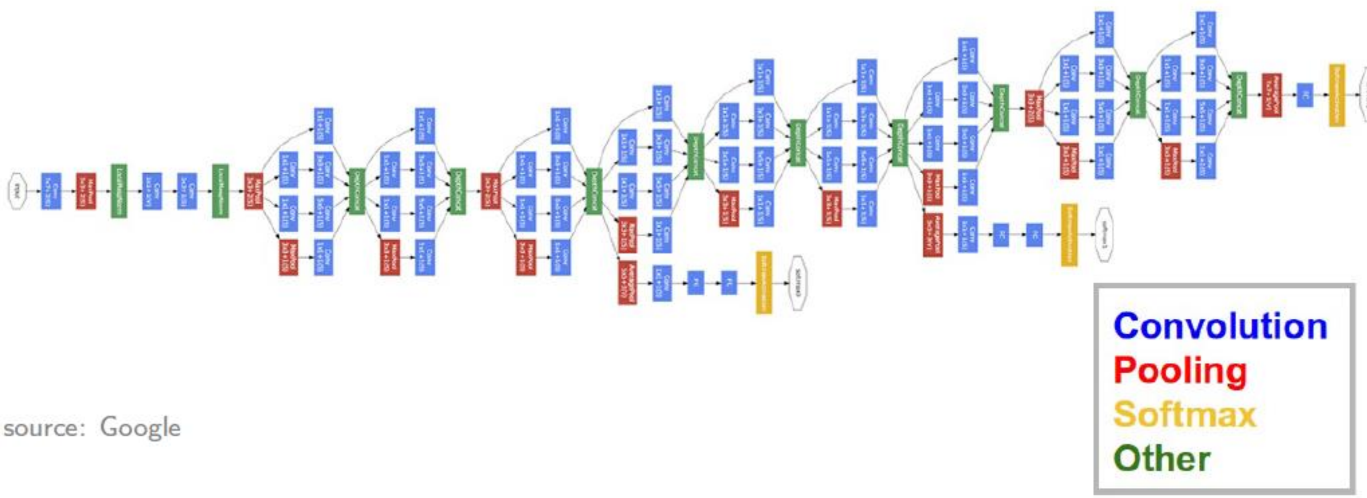




source: [Angermueller et al., 2016, Thériault et al., 2013]

Inception model (a.k.a GoogLeNet)

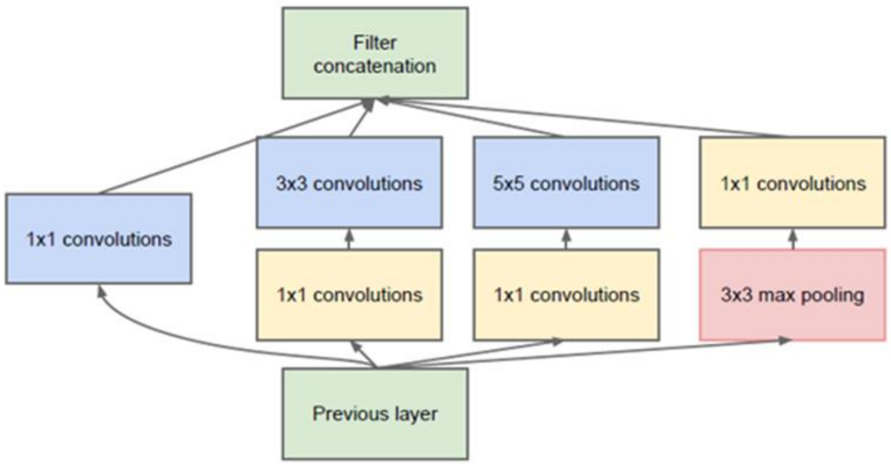
- Deeper network with computational efficiency
  - ImageNet Large Scale Visual Recognition Competition (ILSVRC) 2014 winner (6.7% top 5 error)
  - 22 layers with 5 million parameters (12x less than AlexNet *ILSVRC 2012 winner)
  - Efficient "Inception" module



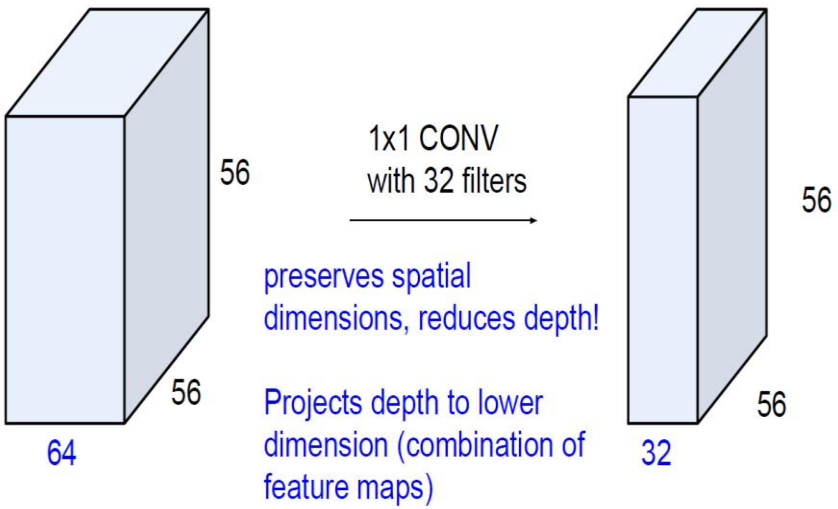
source: Google

Inception module

- Local network topology composing the Inception model
  - Apply parallel filter operations on the input from previous layer
  - Multiple filter sizes for convolution (1x1, 3x3, 5x5)
  - 1x1 convolution for dimensionality reduction



Inception module



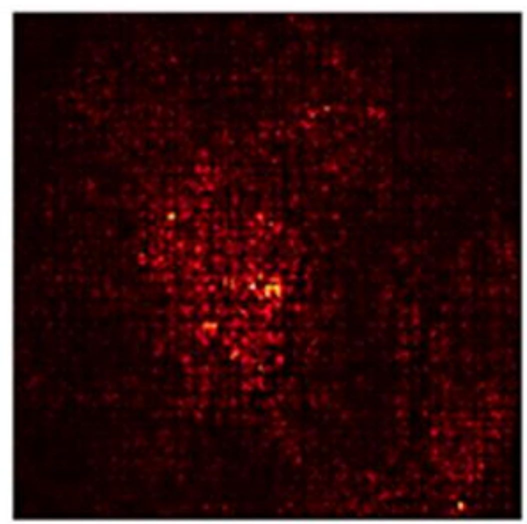
1x1 convolution

Image-specific class saliency maps

- How to tell which pixels matter for classification?
- Visualize the degree to which each pixel affects the classification
- Compute gradient of unnormalized class score with respect to image pixels, take absolute value and max over RGB channels

Given image
Class label: Collie

Saliency map



Class representative images

- Generate an image I* that achieves a high score for the class y

$$I^* = \arg\max_I s_y(I) - R(I)$$

$$R(I) = \lambda \|I\|_2^2$$

- Starting with a random noise, perform gradient ascent on a target class
- **L2 regularization** and **periodic Gaussian blur** regularization

Class representative image
Target class: Collie

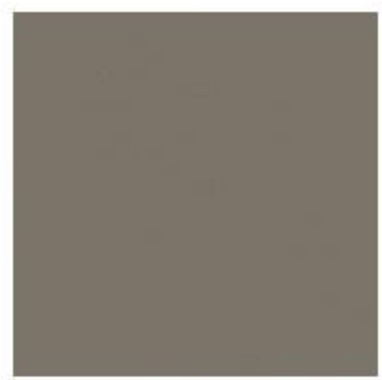
Adversarial examples

- Make the CNN model to miss-classify a given image into a target class
- Starting with a given image (cf. random noise),
  perform gradient ascent over the image to maximize the target class score
- Stop when the network classifies the image as the target class
- **L2 regularization** to normalize the gradients

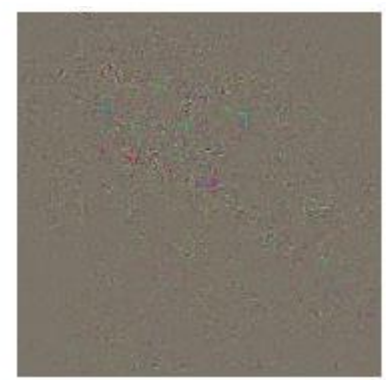
Given image
Class label: Collie

Adversarial example
Target class: Tarantula

Difference



Magnified difference (10x)



How To Install Assignment Files

- Assignment files
  - Utils/*
  - Assignment2-1_Implementing_CNN.ipynb
  - Assignment2-2_Training_CNN.ipynb
  - Assignment2-3_Visualizing_CNN.ipynb
  - CollectSubmission.sh

- Install assignment files
  - tar zxvf assignment2.tar.gz
  - sudo chmod 755 CollectSubmission.sh
  - jupyter notebook

- Open the notebooks on your browser and get started

Submitting your work

- Submitting your work
  - DO NOT clear the final outputs
  - After you are done **all three parts**
    - ✓ $ ./CollectSubmission.sh team_#
    - ✓ Upload the team_#.tar.gz on ETL
    - ✓ Your team_# is in the excel file, http://etl.snu.ac.kr/mod/ubboard/article.php?id=724723&bwid=1535905

Important Notes

- DUE : 10/15/2018, We do not accept late submissions!
- PLEASE read the notes on the notebooks carefully
- Google first before mailing TAs
- TA email: deeplearning.snu@gmail.com


Thank You!