Today: Outline

- **Computing #parameters in Network Architectures**

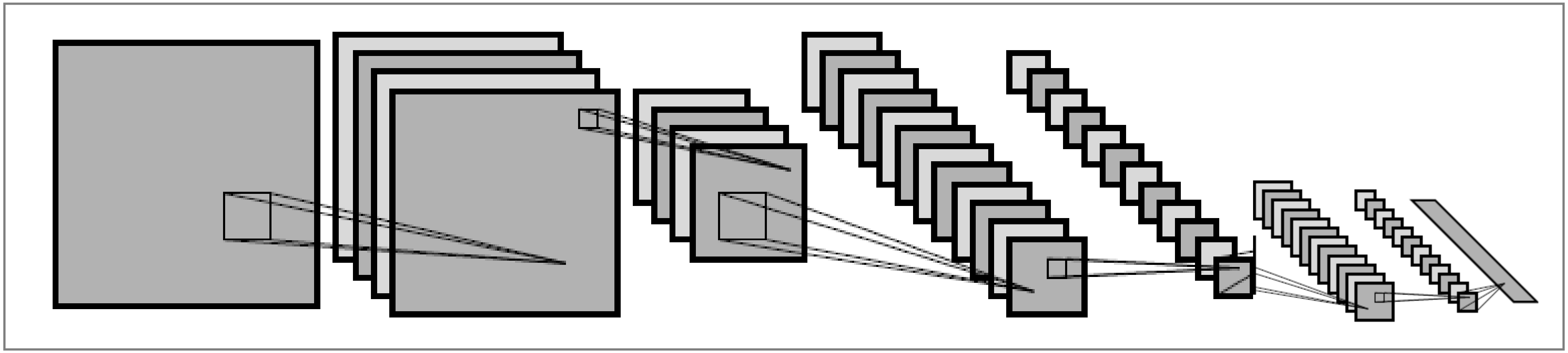
- **Pre-lecture Material:**
  - **GPUs: Divide and Conquer**
  - **Dropout**
  - **Data Augmentation**

- **Reminder:** PS2 Self Score due Mar 3

Neural Networks IV

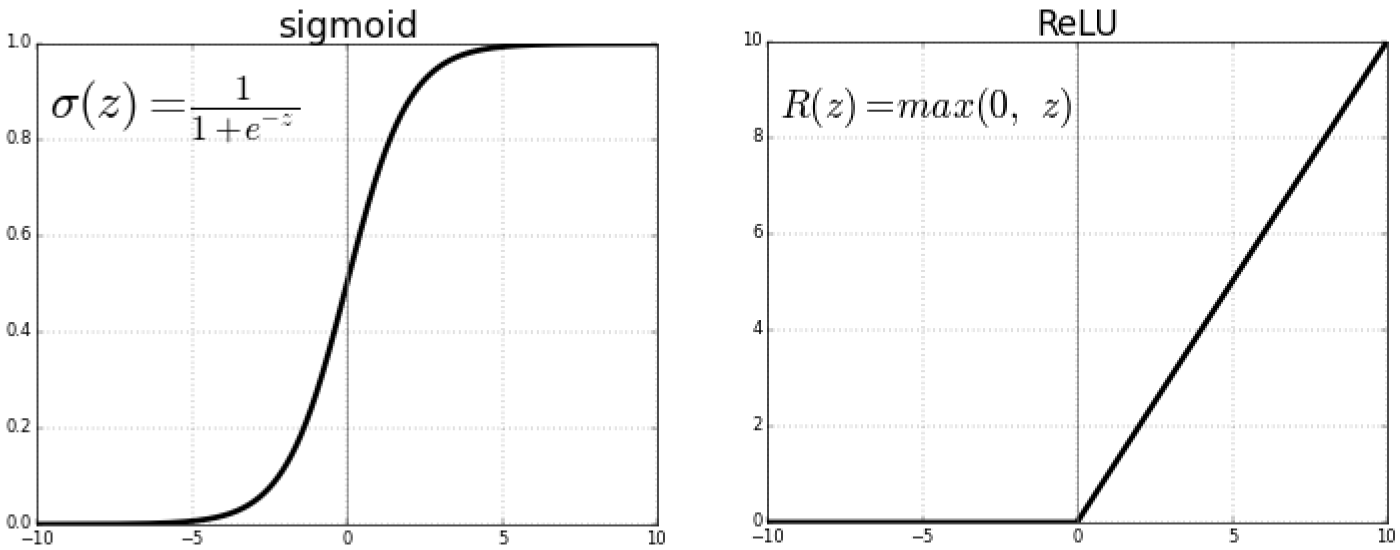
Network Architectures

CIFAR-10 Demo ConvJS Network

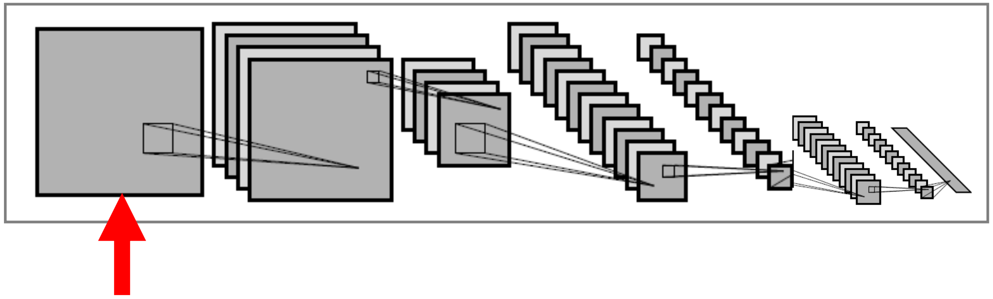


input image conv RELU pool conv RELU pool conv RELU pool fc

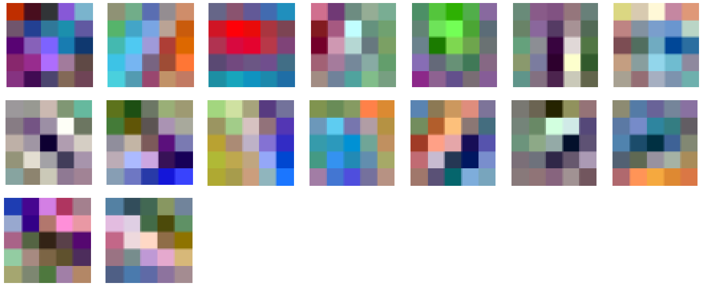
RELU: rectified linear unit



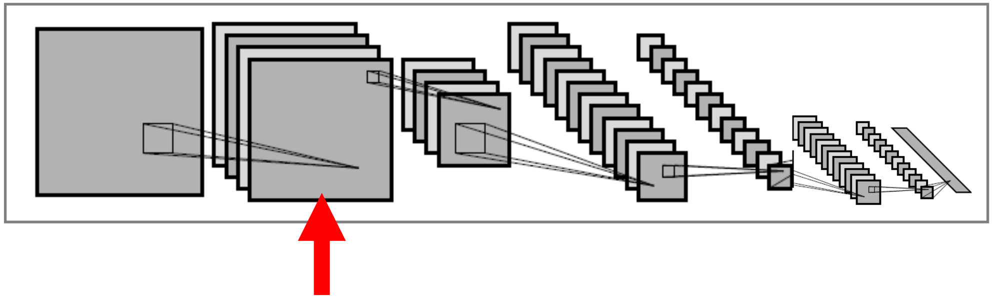
RELU function $g(x) = \max(0, x)$



filter size 5x5x3, stride 1



input (32x32x3)

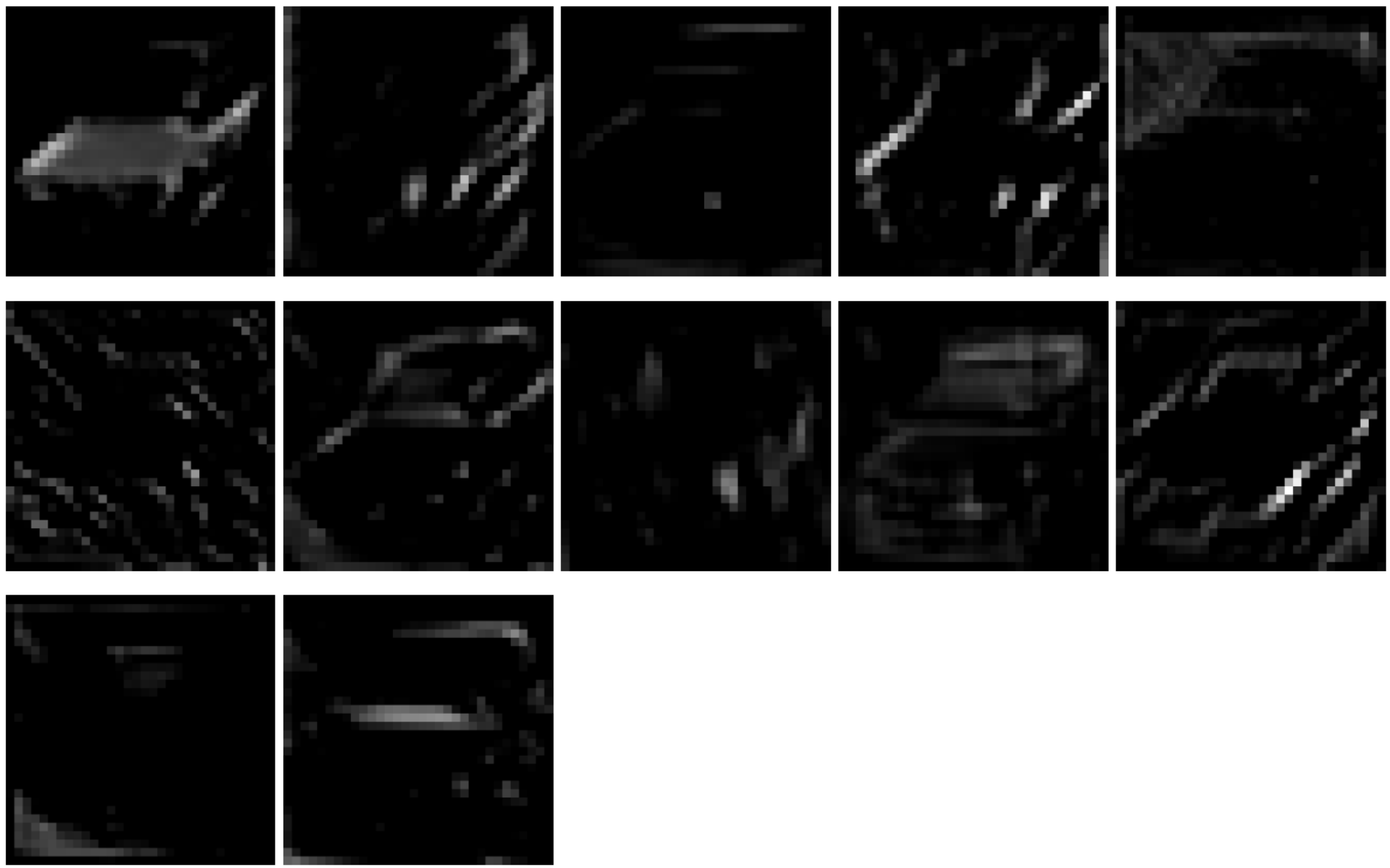


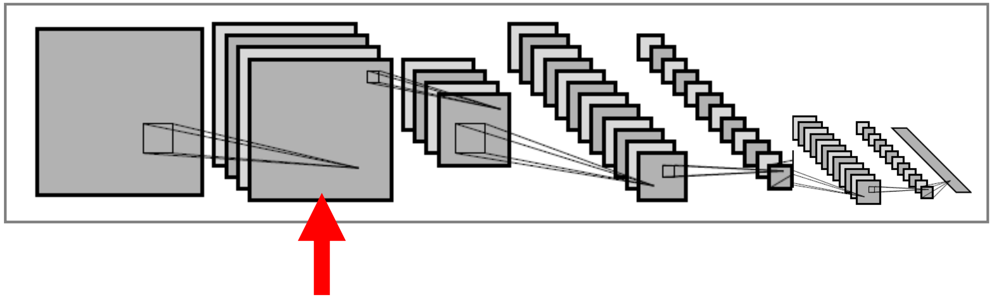
filter size 5x5x3, stride 1

input (32x32x3)

RELU

conv (32x32x16) params: 16x5x5x3+16 = 1216

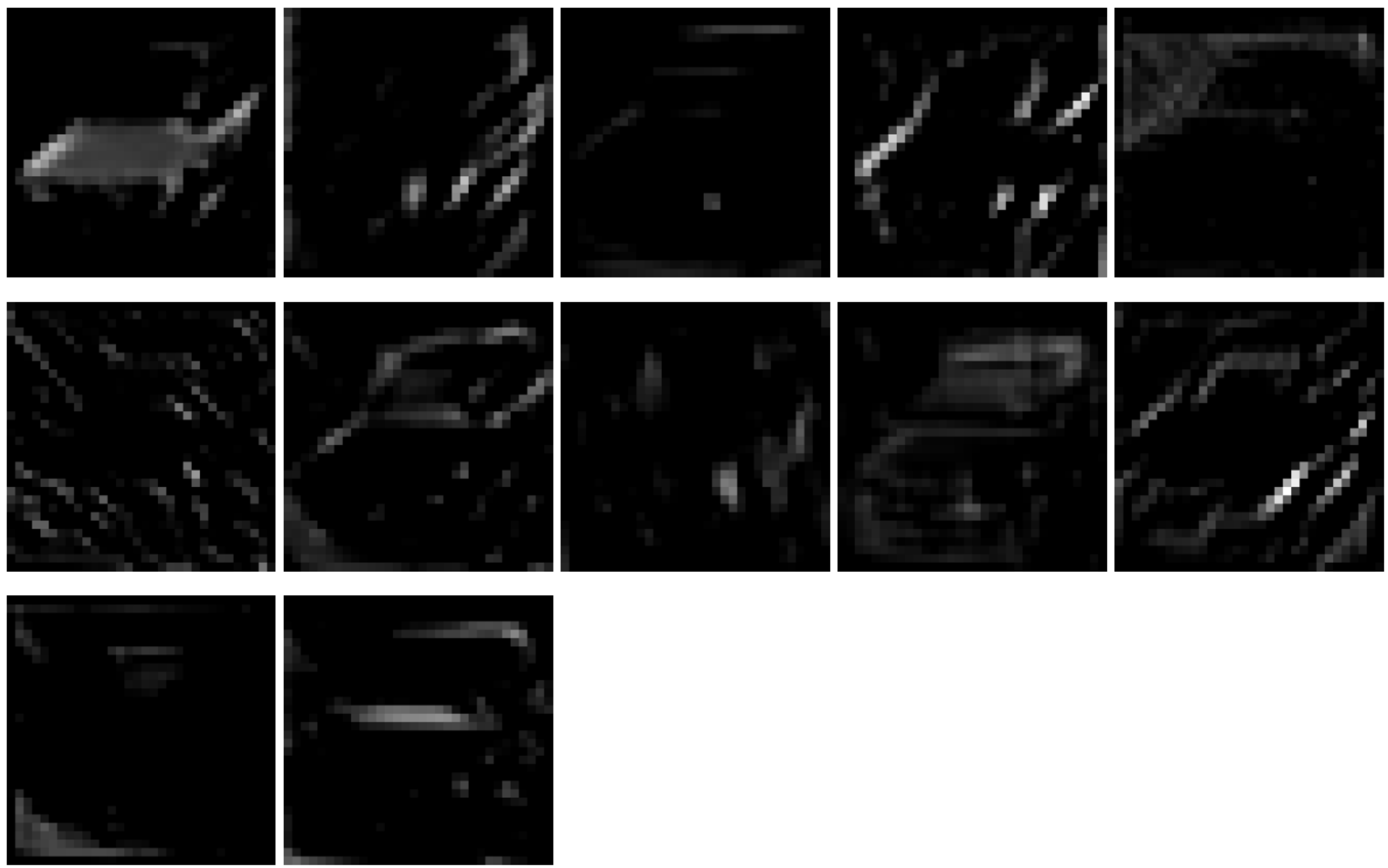


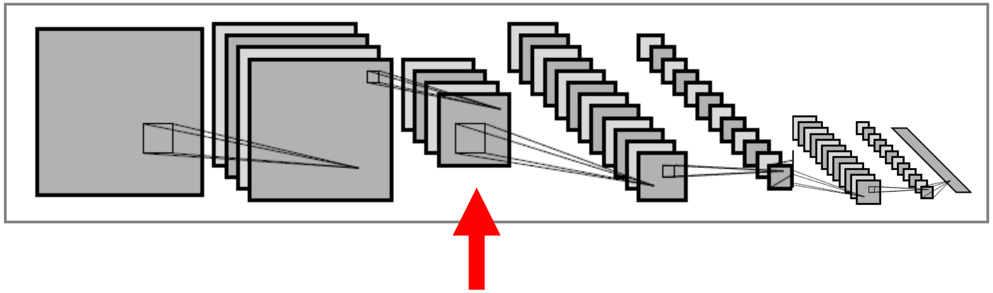


filter size 5x5x3, stride 1

input (32x32x3)

conv (32x32x16) params: 16x5x5x3+16 = 1216

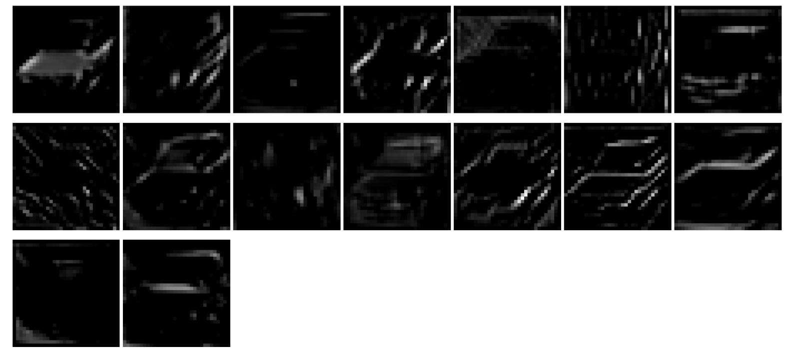




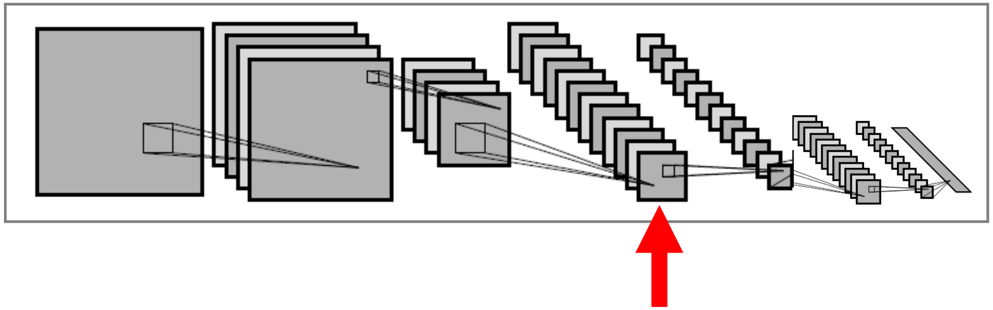
filter size 5x5x3, stride 1

input (32x32x3)

conv (32x32x16) params: 16x5x5x3+16 = 1216



pool (16x16x16)
pooling size 2x2, stride 2

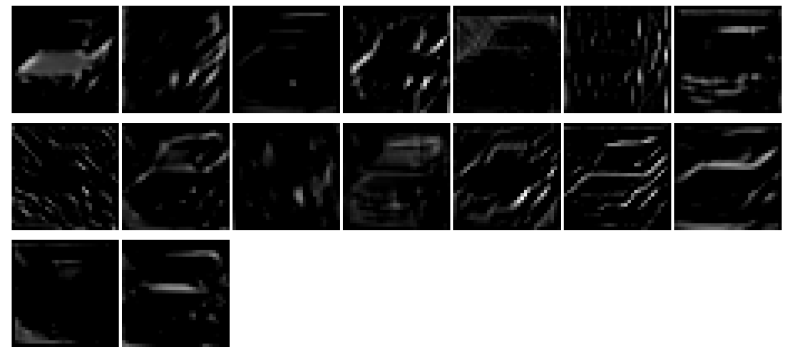


filter size 5x5x3, stride 1

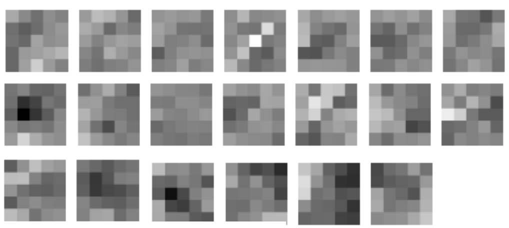
input (32x32x3)

pool (16x16x16)
pooling size 2x2, stride 2

conv (32x32x16) params: 16x5x5x3+16 = 1216

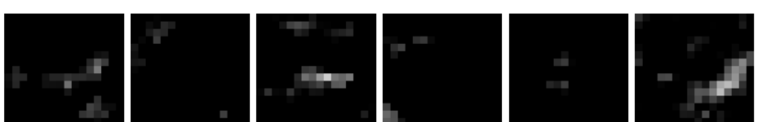


filter size 5x5x16, stride 1

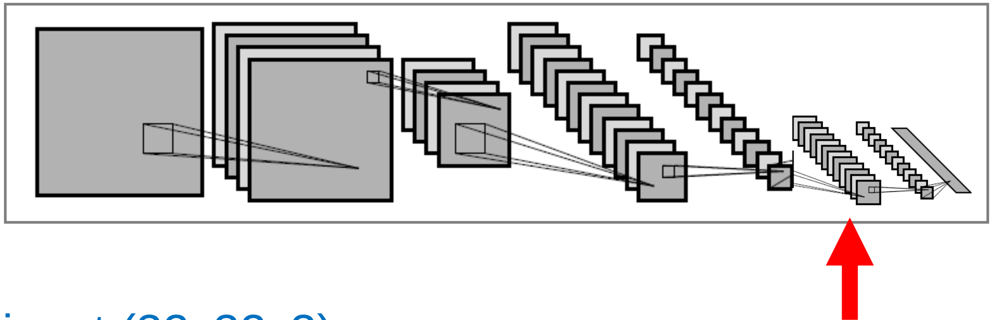


RELU

conv (16x16x20) params: 20x5x5x16+20 = 8020



pool (8x8x20)
pooling size 2x2, stride 2



One more conv+RELU+pool:

conv (8x8x20)
filter size 5x5x20, stride 1
relu (8x8x20)
pool (4x4x20)
pooling size 2x2, stride 2
parameters: 20x5x5x20+20 = 10020

input (32x32x3)

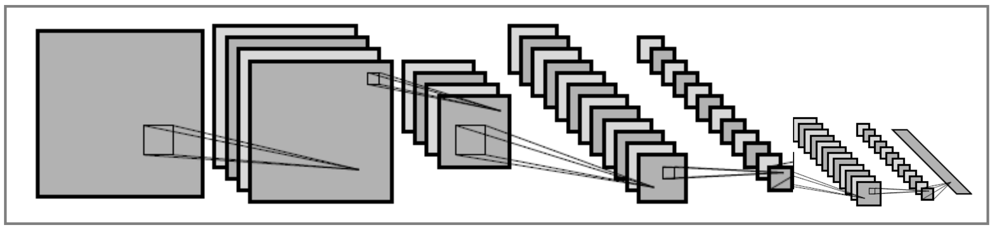
fc (1x1x10); parameters: 10x320+10 = 3210

softmax (1x1x10)

Dog cat Car …

Testing the network

- Show top three most likely classes



http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

Neural Networks IV

Pre-Lecture Material

Alex Krizhevsky

Alex Krizhevsky

Dessa

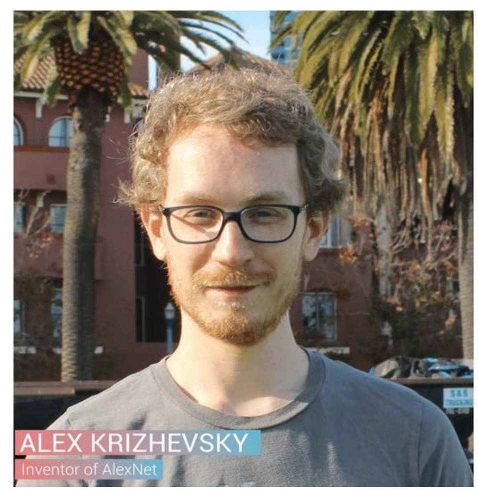
Verified email at dessa.com

Machine Learning

FOLLOW

TITLE	CITED BY	YEAR
Imagenet classification with deep convolutional neural networks A Krizhevsky, I Sutskever, GE Hinton Advances in neural information processing systems, 1097-1105	57520	2012
Dropout: a simple way to prevent neural networks from overfitting N Srivastava, G Hinton, A Krizhevsky, I Sutskever, R Salakhutdinov The journal of machine learning research 15 (1), 1929-1958	18013	2014

Hence the name ***AlexNet***



ACM Turing Award (2019)

- Three ‘Godfathers of Deep Learning’ Selected for Turing Award

- ***Geoff Hinton***, an emeritus professor at the University of Toronto and a senior researcher at Alphabet Inc.’s Google Brain

- ***Yann LeCun***, a professor at New York University and the chief AI scientist at Facebook Inc.

- ***Yoshua Bengio***, a professor at the University of Montreal as well as co-founder of AI company Element AI Inc.

Geoffrey E Hinton

Yann LeCun

Yoshua Bengio

Neural Networks IV

Research Paper

Elements of a Research Paper

- *Title, Authors, Abstract*

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

Elements of a Research Paper

- *Introduction, Related Works*

- How the proposed approach addresses an important problem that has clear applications?

- How the proposed approach is different from other works in the literature?

Elements of a Research Paper

- *Dataset(s) and Architecture(s)*

- What datasets (and their specs) have been used to demonstrate that the results of this paper generalize to different datasets, and possibly different tasks.

- What neural network architectures (and their specs) have been used to demonstrate the results of this paper generalizes to different architectures.

Elements of a Research Paper

- *Experimental Setup*
  - All aspects of experimental setup should be provided such that the experimental results are ***reproducible***. Code should ideally also be made available.

- *Experimental Results*
  - Typically, research papers will be accepted if they provide a novel contribution and obtain state-of-the-art results on multiple datasets/tasks.

- *Conclusions*

Neural Networks IV

GPUs

GPUs

- NVIDIA TITAN V GPU

Mini-batches

- Gradients could be updated using:
  - One data point (too inaccurate)
  - All data points (too expensive)
  - Mini-batch (a good trade-off)

- The size of the mini-batch depends on:
  - How good of an approximation you need
  - How much GPU memory you have per GPU
  - How many GPUs you have

- GPUs can compute gradients of mini-batches in parallel, *i.e. Training on multiple GPUs*: Divide and Conquer.

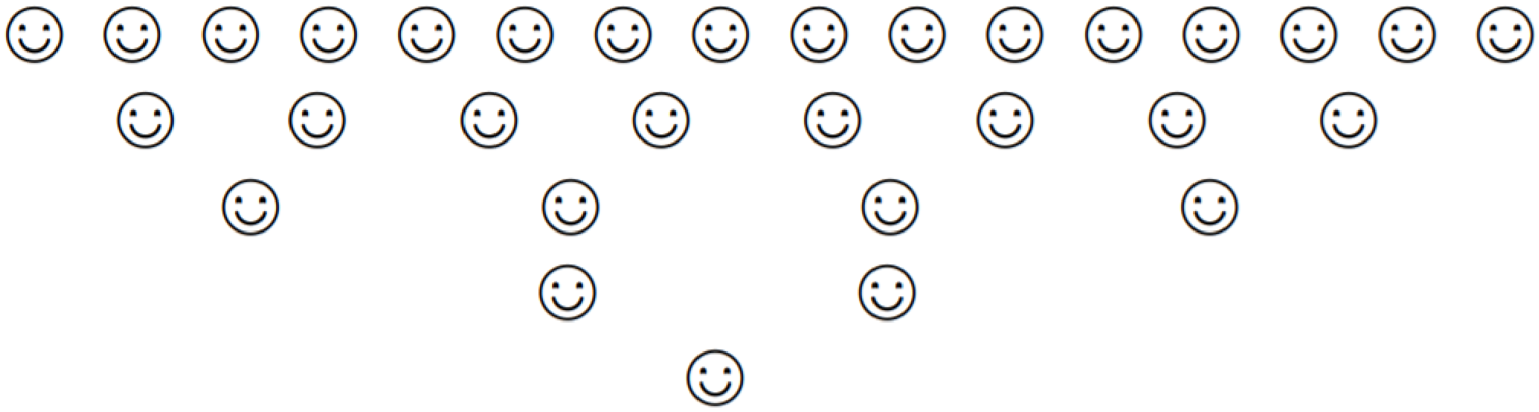
Divide and Conquer

- Everyone stand up.
- You will each carry out the following algorithm:

```
count = 1;
while (you are not the only person standing) {
    find another person who is standing
    if (your first name < other person's first name)
        sit down (break ties using last names)
    else
        count = count + the other person's count
}
if (you are the last person standing)
    report your final count
```

Divide and Conquer

- At each stage of the "joint algorithm", the problem size is divided in half.



- This approach benefits from the fact that you perform the algorithm *in parallel* with each other.

Neural Networks IV

Dropout

Dropout: A Classical Regularization Technique

- Combining the predictions of many different models is a very successful way to reduce test errors.

- But it appears to be too expensive for big neural networks that already take several days to train.

- There is, however, a very efficient version of model combination that only costs about a factor of two during training: ***Dropout***

[Krizhevsky *et al.*]

Dropout: A Classical Regularization Technique

- Setting to zero the output of each hidden neuron with a specific dropout probability, *e.g.* 0.5.

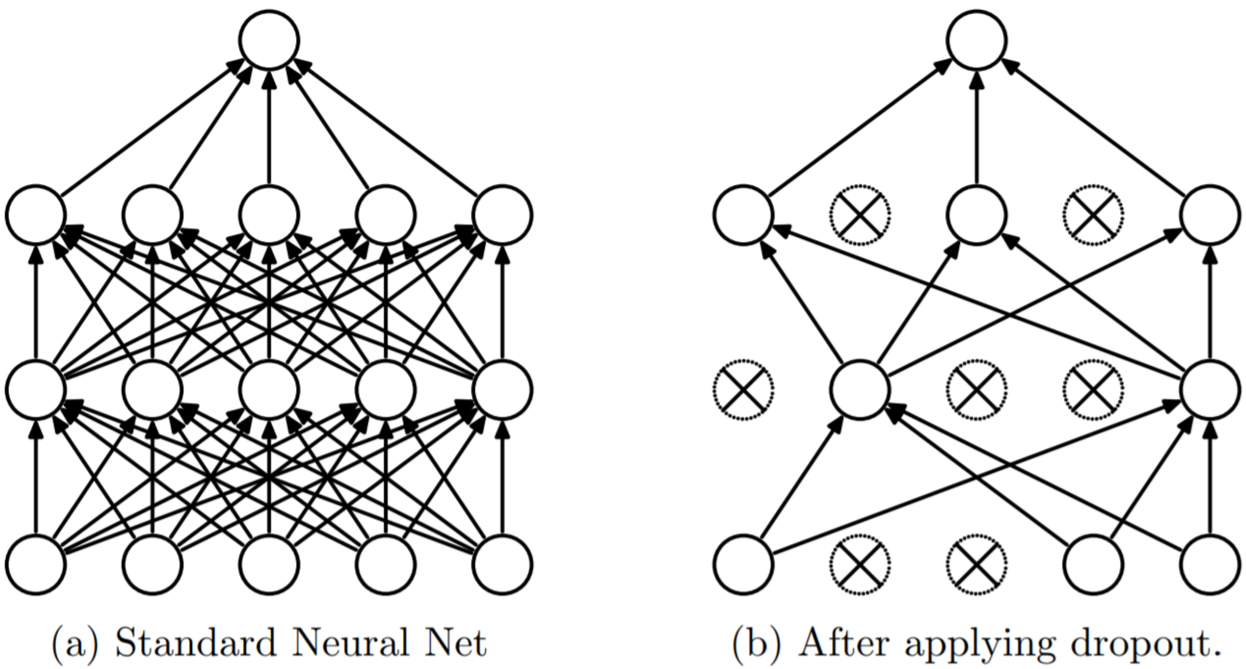
- The neurons which are “dropped out” in this way
  - do not contribute to the forward pass, and
  - do not participate in backpropagation.

- So every time an input is presented, the neural network samples a different architecture, but all these architectures share weights.

[Krizhevsky *et al.*]

Dropout: A Classical Regularization Technique

- Many Deep Models employ dropout at training time to avoid overfitting, allowing for better generalization.



[Srivastava *et al.*]

Dropout: A Classical Regularization Technique

- Dropout can be thought of as a model averaging technique.
- Dropout can be applied to fully-connected layers or convolutional layers.
- It has so far been observed to give higher performance gains when applied to fully-connected layers.

Dropout Variants

- Several variants of dropout have been introduced:
  - How much dropout is applied to neurons/weights?
    - Information Dropout
    - DropConnect
    - Curriculum Dropout
  - Which neurons to drop out?
    - Adaptive Dropout
    - DropBlock
    - Excitation Dropout

Neural Networks IV

Data Augmentation

Data Augmentation

- Another technique that prevents overfitting.

- How?
  By artificially enlarging the dataset using label-preserving transformations.

- Examples:
  - generating image translations and horizontal reflections
  - altering the intensities of the RGB channels in training images: add perturbations to each RGB image pixel $I_{xy} = [I_{xy}^R, I_{xy}^G, I_{xy}^B]$

[Krizhevsky *et al.*]

Data Augmentation

- Could be computed “on the fly,” and do not necessarily need to be stored on disk.

- How?
  The transformed images are generated in Python code on the CPU while the GPU is training on the previous batch of images.

- So these data augmentation schemes can be, in effect, computationally free.

[Krizhevsky *et al.*]