MIDDLE EAST TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

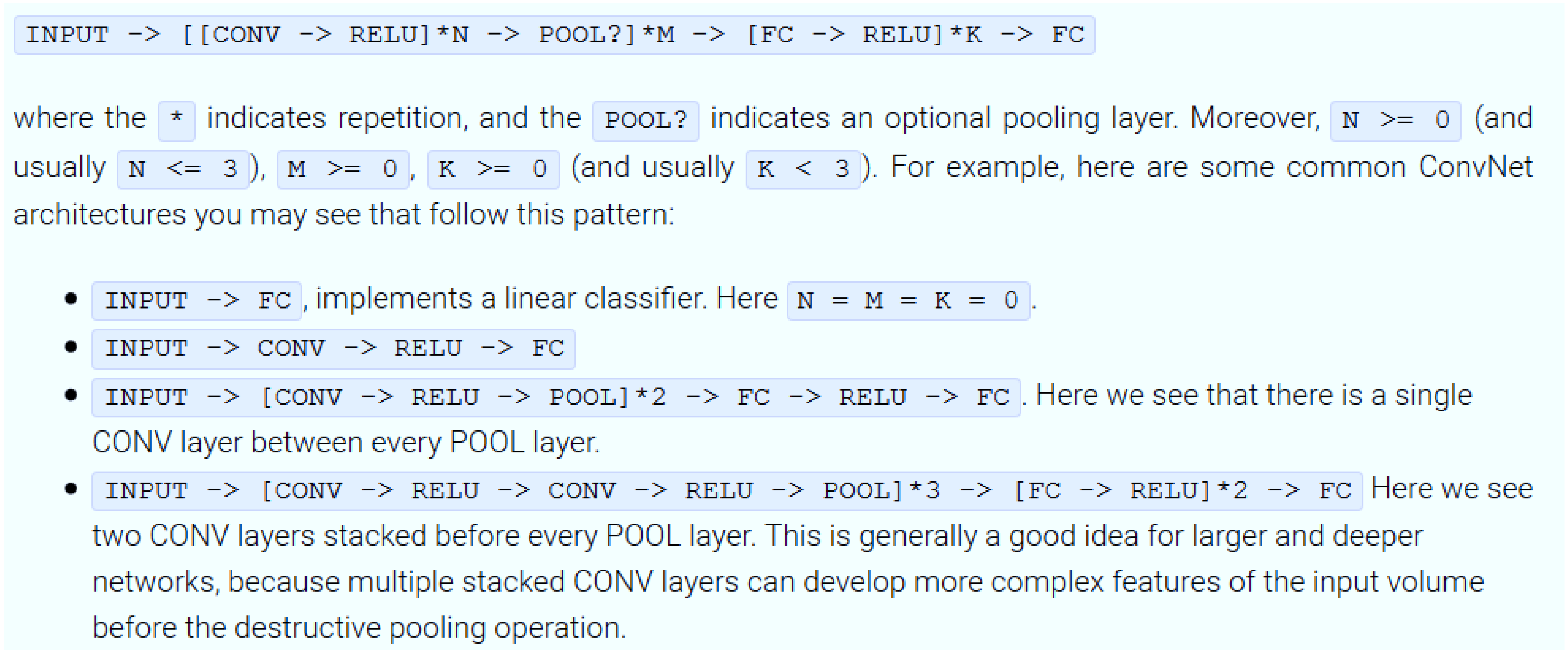
CENG 403
Introduction to Deep Learning

Week 11a

Sinan Kalkan

Previously on CENG403

A Blueprint for CNNs



`INPUT -> [[CONV -> RELU]*N -> POOL?]*M -> [FC -> RELU]*K -> FC`

where the `*` indicates repetition, and the `POOL?` indicates an optional pooling layer. Moreover, `N >= 0` (and usually `N <= 3`), `M >= 0`, `K >= 0` (and usually `K < 3`). For example, here are some common ConvNet architectures you may see that follow this pattern:

- `INPUT -> FC`, implements a linear classifier. Here `N = M = K = 0`.
- `INPUT -> CONV -> RELU -> FC`
- `INPUT -> [CONV -> RELU -> POOL]*2 -> FC -> RELU -> FC`. Here we see that there is a single CONV layer between every POOL layer.
- `INPUT -> [CONV -> RELU -> CONV -> RELU -> POOL]*3 -> [FC -> RELU]*2 -> FC` Here we see two CONV layers stacked before every POOL layer. This is generally a good idea for larger and deeper networks, because multiple stacked CONV layers can develop more complex features of the input volume before the destructive pooling operation.

http://cs231n.github.io/convolutional-networks/

Previously on CENG403

Fully Convolutional Networks (FCNs)

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. CVPR.

- Fully-connected layers limit the input size
- We can convert FC layers to convolution

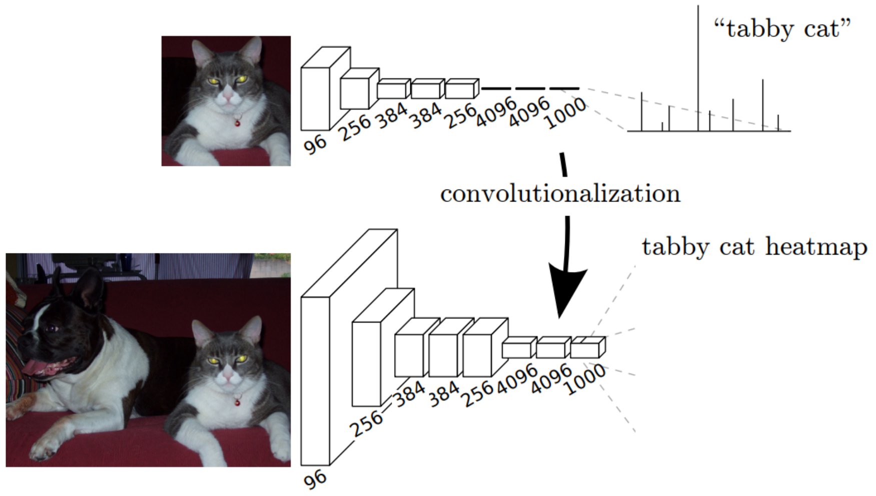


Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.

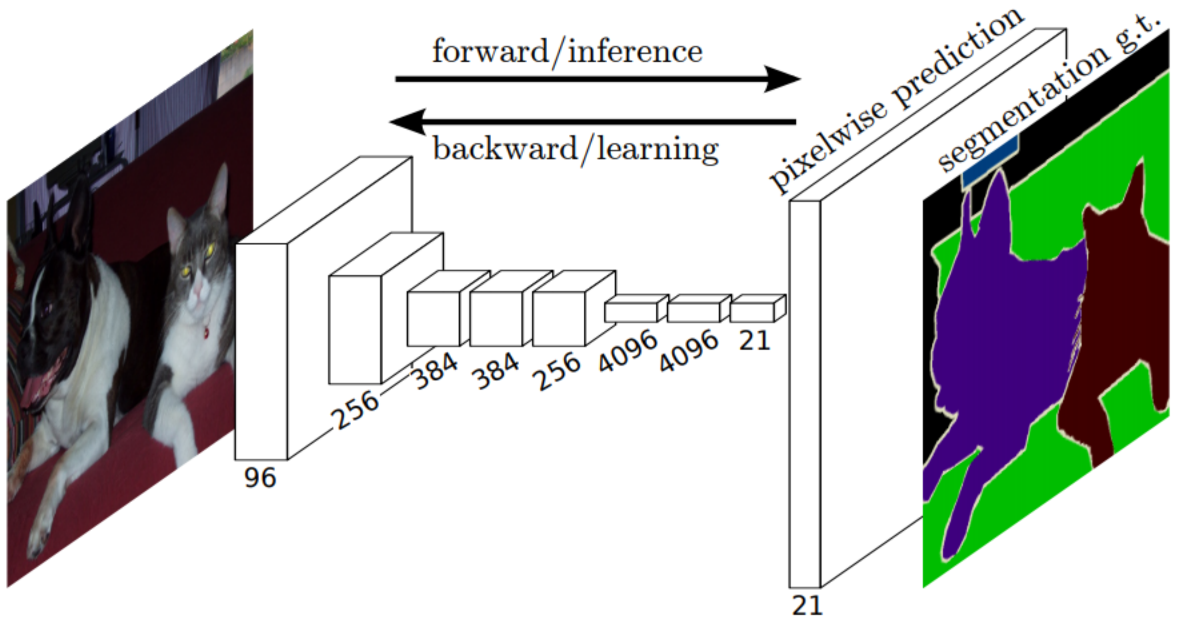
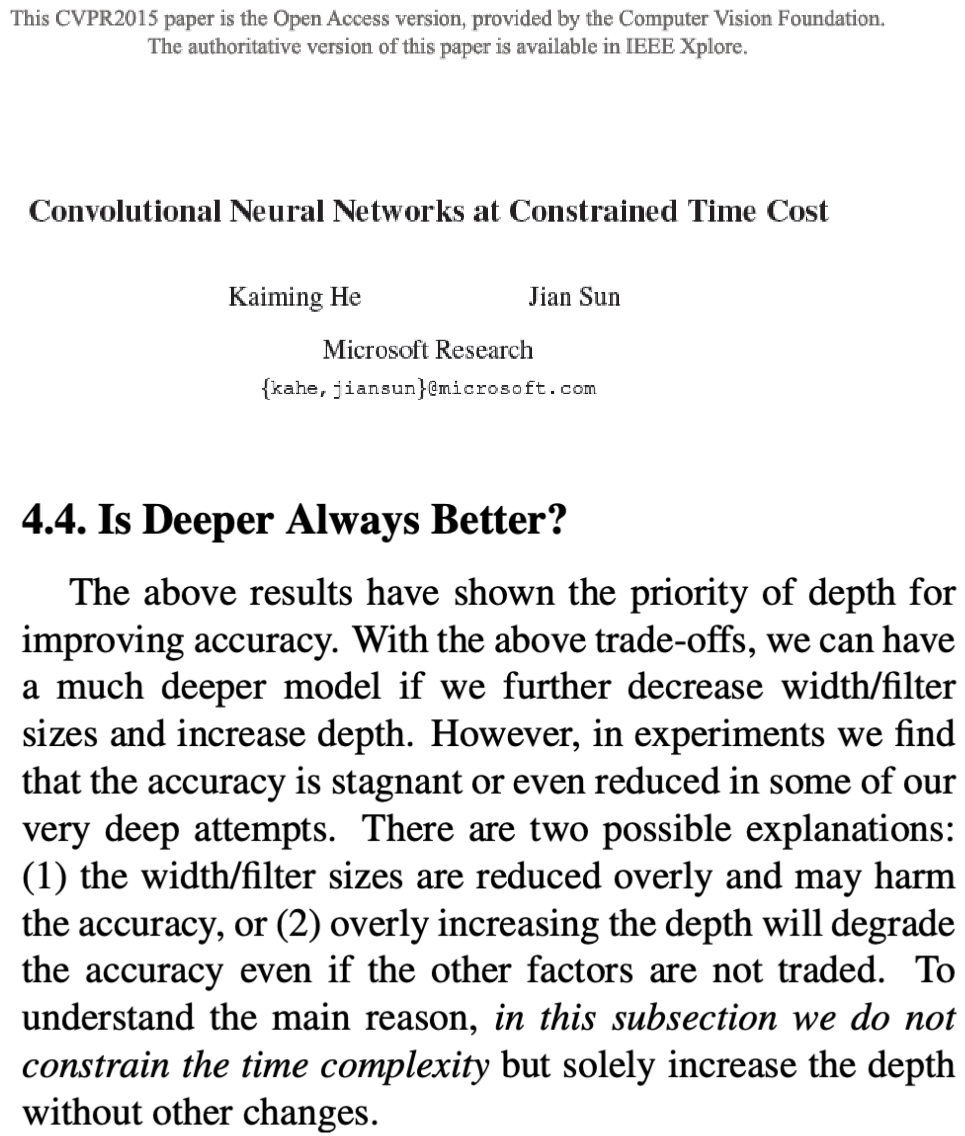


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

Previously on CENG403

Trade-offs in architecture

- Between filter size and number of layers (depth)
  - Keep the layer widths fixed.
  - *“When the time complexity is roughly the same, the deeper networks with smaller filters show better results than the shallower networks with larger filters.”*
- Between layer width and number of layers (depth)
  - Keep the size of the filters fixed.
  - *“We find that increasing the depth leads to considerable gains…”*
- Between filter size and layer width
  - Keep the number of layers (depth) fixed.
  - No significant difference



This CVPR2015 paper is the Open Access version, provided by the Computer Vision Foundation.
The authoritative version of this paper is available in IEEE Xplore.

Convolutional Neural Networks at Constrained Time Cost

Kaiming He Jian Sun

Microsoft Research

{kahe, jiansun}@microsoft.com

4.4. Is Deeper Always Better?

The above results have shown the priority of depth for improving accuracy. With the above trade-offs, we can have a much deeper model if we further decrease width/filter sizes and increase depth. However, in experiments we find that the accuracy is stagnant or even reduced in some of our very deep attempts. There are two possible explanations: (1) the width/filter sizes are reduced overly and may harm the accuracy, or (2) overly increasing the depth will degrade the accuracy even if the other factors are not traded. To understand the main reason, *in this subsection we do not constrain the time complexity* but solely increase the depth without other changes.

Memory constraints

- Using smaller RFs with more layers means more memory since you need to store more activation maps
- In such memory-scarce cases,
  - the first layer may use bigger RFs with S>1
  - information loss from the input volume may be less critical than the following layers
- E.g., AlexNet uses RFs of 11x11 and S = 4 for the first layer.

Finetuning

1. If the new dataset is small and similar to the original dataset used to train the CNN:
   - Finetuning the whole network may lead to overfitting
   - Just train the newly added layer

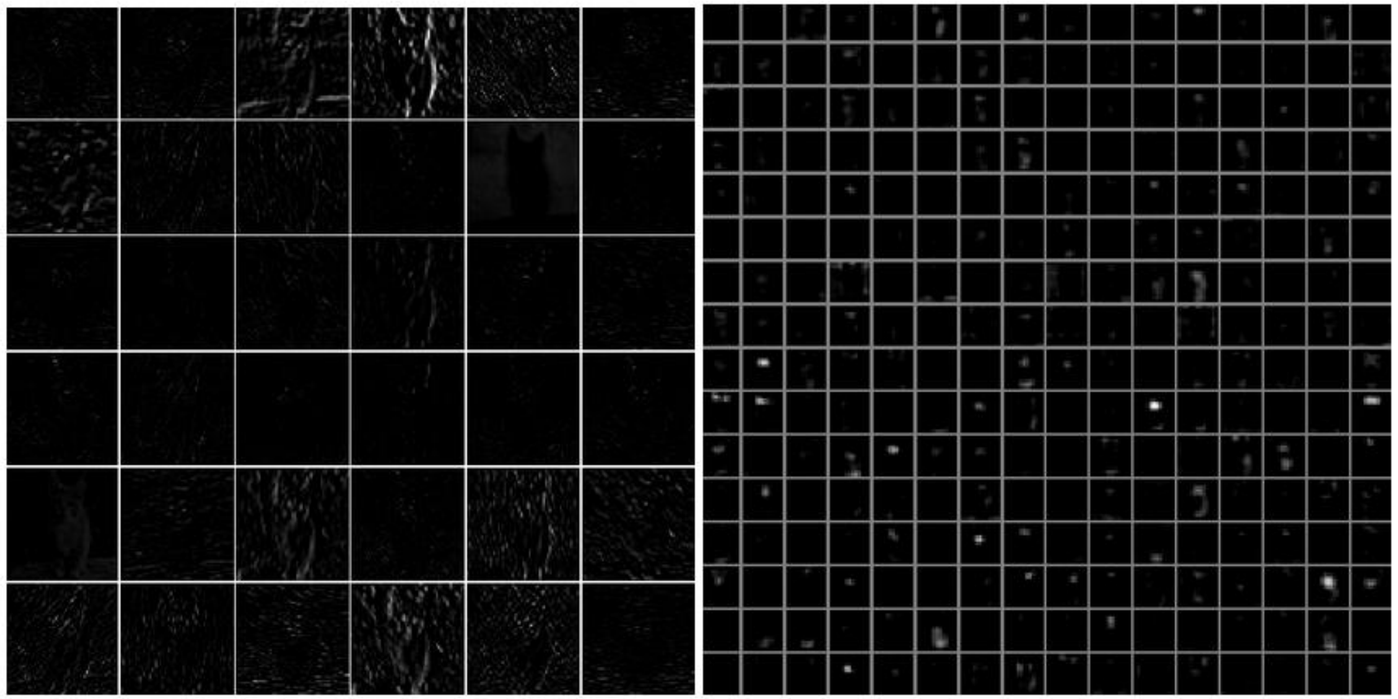
2. If the new dataset is big and similar to the original dataset:
   - The more, the merrier: go ahead and train the whole network

3. If the new dataset is small and different from the original dataset:
   - Not a good idea to train the whole network
   - However, add your new layer not to the top of the network, since those parts are very dataset (problem) specific
   - Add your layer to earlier parts of the network

4. If the new dataset is big and different from the original dataset:
   - We can "finetune" the whole network
   - This amounts to a new training problem by initializing the weights with those of another network

Visualize activations during training

- Activations are dense at the beginning.
  - They should get sparser during training.
- If some activation maps are all zero for many inputs, dying neuron problem => high learning rate in the case of ReLUs.



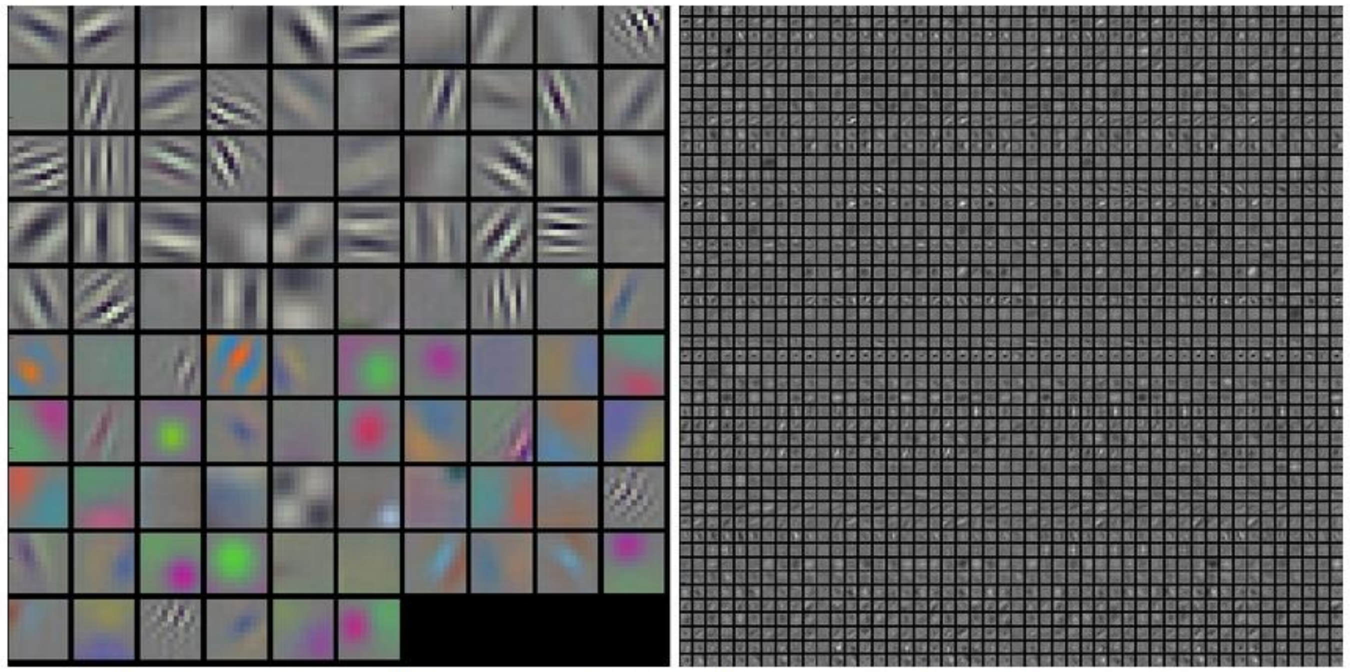
Typical-looking activations on the first CONV layer (left), and the 5th CONV layer (right) of a trained AlexNet looking at a picture of a cat. Every box shows an activation map corresponding to some filter. Notice that the activations are sparse (most values are zero, in this visualization shown in black) and mostly local.

http://cs231n.github.io/convolutional-networks/

Previously on CENG403

Visualize the weights

- We can directly look at the filters of all layers
- First layer is easier to interpret
- Filters shouldn't look noisy



Typical-looking filters on the first CONV layer (left), and the 2nd CONV layer (right) of a trained AlexNet. Notice that the first-layer weights are very nice and smooth, indicating nicely converged network. The color/grayscale features are clustered because the AlexNet contains two separate streams of processing, and an apparent consequence of this architecture is that one stream develops high-frequency grayscale features and the other low-frequency color features. The 2nd CONV layer weights are not as interpretable, but it is apparent that they are still smooth, well-formed, and absent of noisy patterns.

http://cs231n.github.io/convolutional-networks/

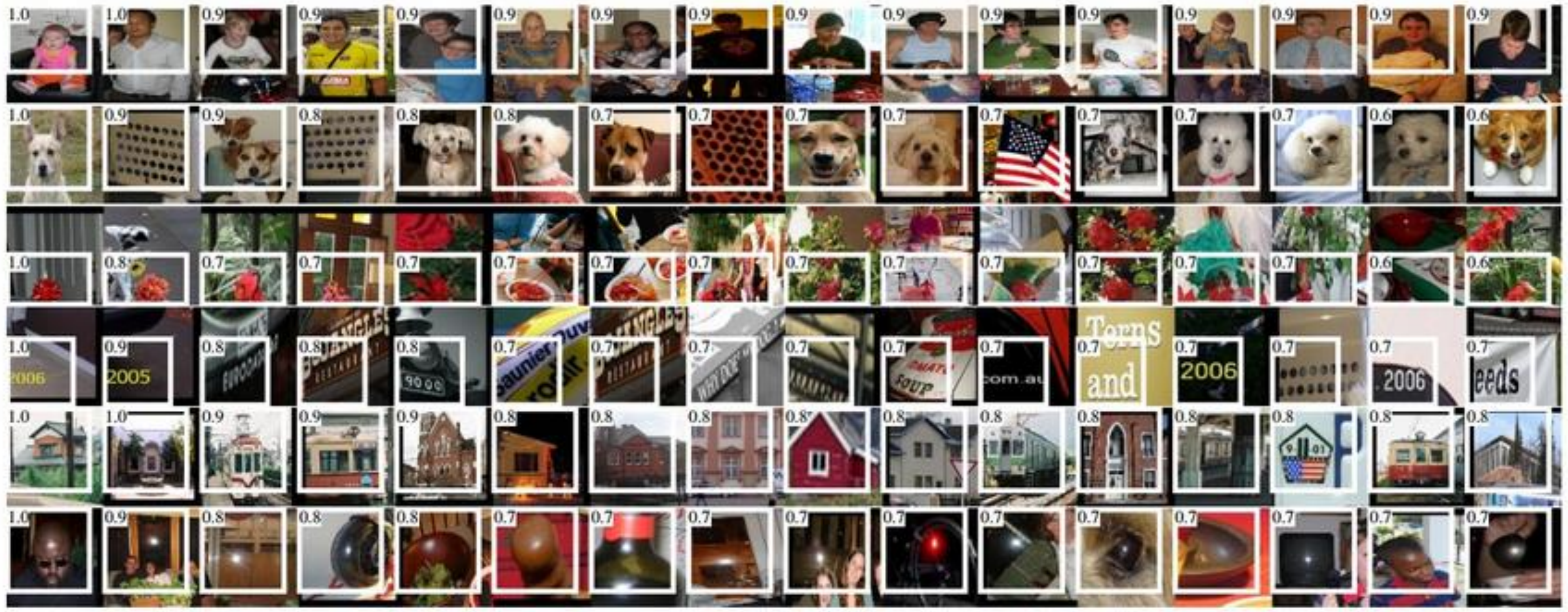
Previously on CENG403

Visualize the inputs that maximally activate a neuron

- Keep track of which images activate a neuron most

Rich feature hierarchies for accurate object detection and semantic segmentation
Tech report (v5)

Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik
UC Berkeley
{rbg,jdonahue,trevor,malik}@eecs.berkeley.edu



Maximally activating images for some POOL5 (5th pool layer) neurons of an AlexNet. The activation values and the receptive field of the particular neuron are shown in white. (In particular, note that the POOL5 neurons are a function of a relatively large portion of the input image!) It can be seen that some neurons are responsive to upper bodies, text, or specular highlights.

http://cs231n.github.io/convolutional-networks/

Previously on CENG403

Embed the codes in a lower-dimensional space

- Place images into a 2D space such that images which produce similar CNN codes are placed close.
- You can use, e.g., t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE embedding of a set of images based on their CNN codes. Images that are nearby each other are also close in the CNN representation space, which implies that the CNN "sees" them as being very similar. Notice that the similarities are more often class-based and semantic rather than pixel and color-based. For more details on how this visualization was produced the associated code, and more related visualizations at different scales refer to t-SNE visualization of CNN codes.

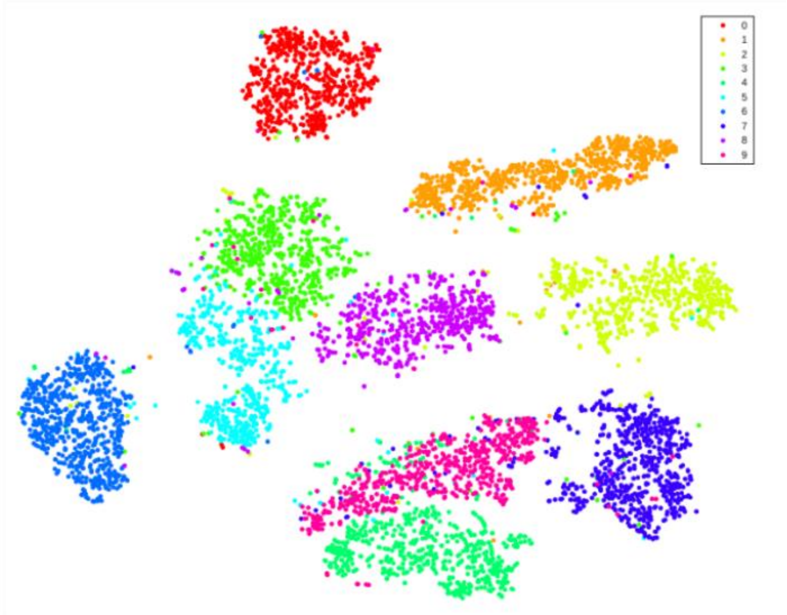


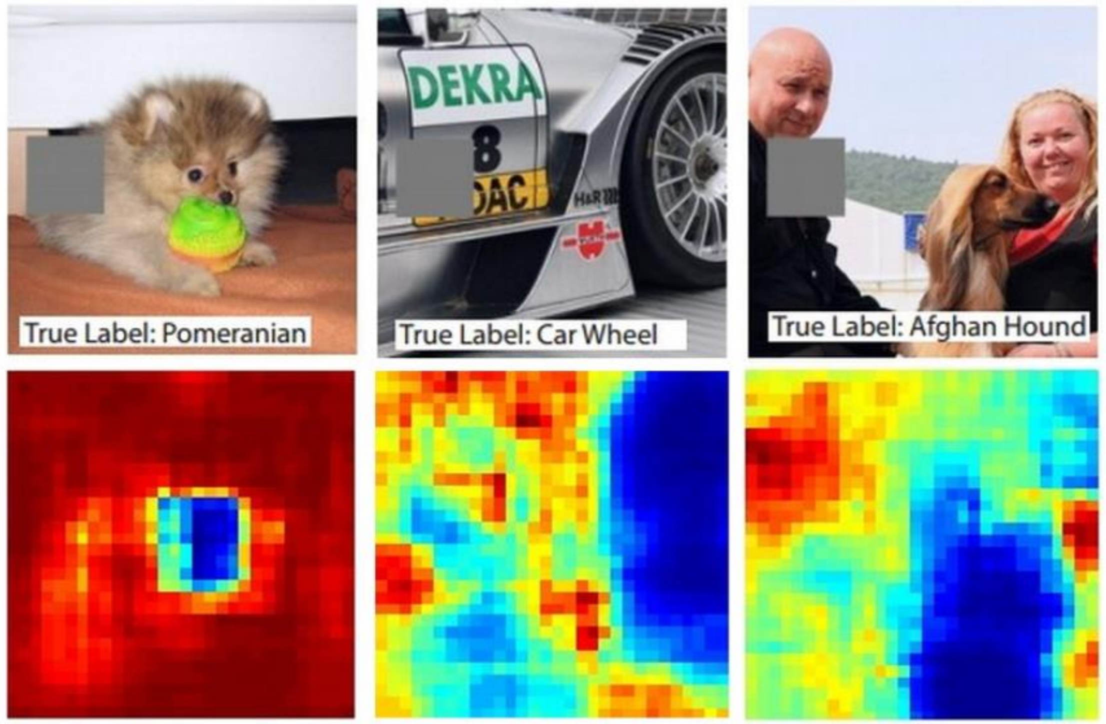
Figure 1 : Illustration of t-SNE on MNIST dataset

Figure: Laurens van der Maaten and Geoffrey Hinton

Previously on CENG403

Occlude parts of the image

- Slide an “occlusion window” over the image
- For each occluded image, determine the class prediction confidence/probability.



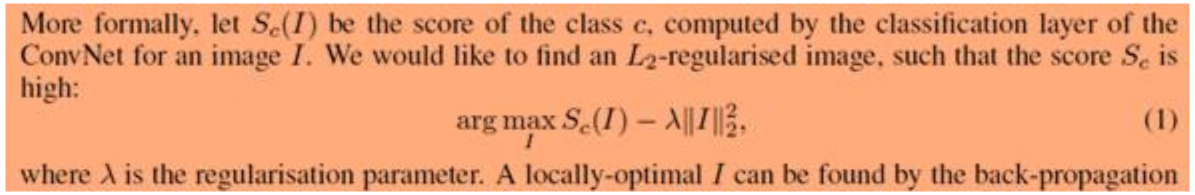
Three input images (top). Notice that the occluder region is shown in grey. As we slide the occluder over the image we record the probability of the correct class and then visualize it as a heatmap (shown below each image). For instance, in the left-most image we see that the probability of Pomeranian plummets when the occluder covers the face of the dog, giving us some level of confidence that the dog's face is primarily responsible for the high classification score. Conversely, zeroing out other parts of the image is seen to have relatively negligible impact.

http://cs231n.github.io/convolutional-networks/

Data gradients

Previously on CENG403

- Generate an image that maximizes the class score.

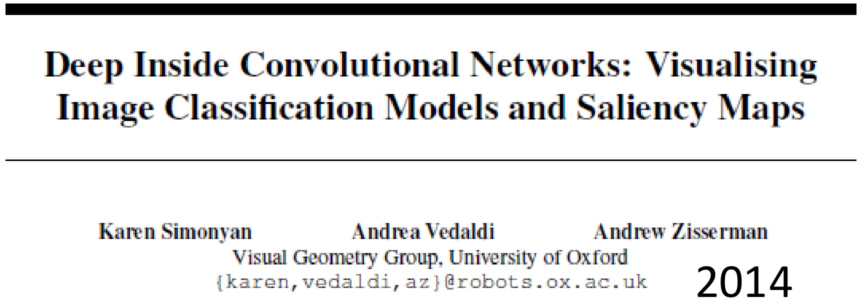


More formally, let $S_c(I)$ be the score of the class c, computed by the classification layer of the ConvNet for an image I. We would like to find an L_2-regularised image, such that the score S_c is high:

$$\arg\max_I S_c(I) - \lambda \|I\|_2^2, \qquad (1)$$

where λ is the regularisation parameter. A locally-optimal I can be found by the back-propagation

- Use: Gradient ascent!



Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps

Karen Simonyan Andrea Vedaldi Andrew Zisserman
Visual Geometry Group, University of Oxford
{karen,vedaldi,az}@robots.ox.ac.uk

2014

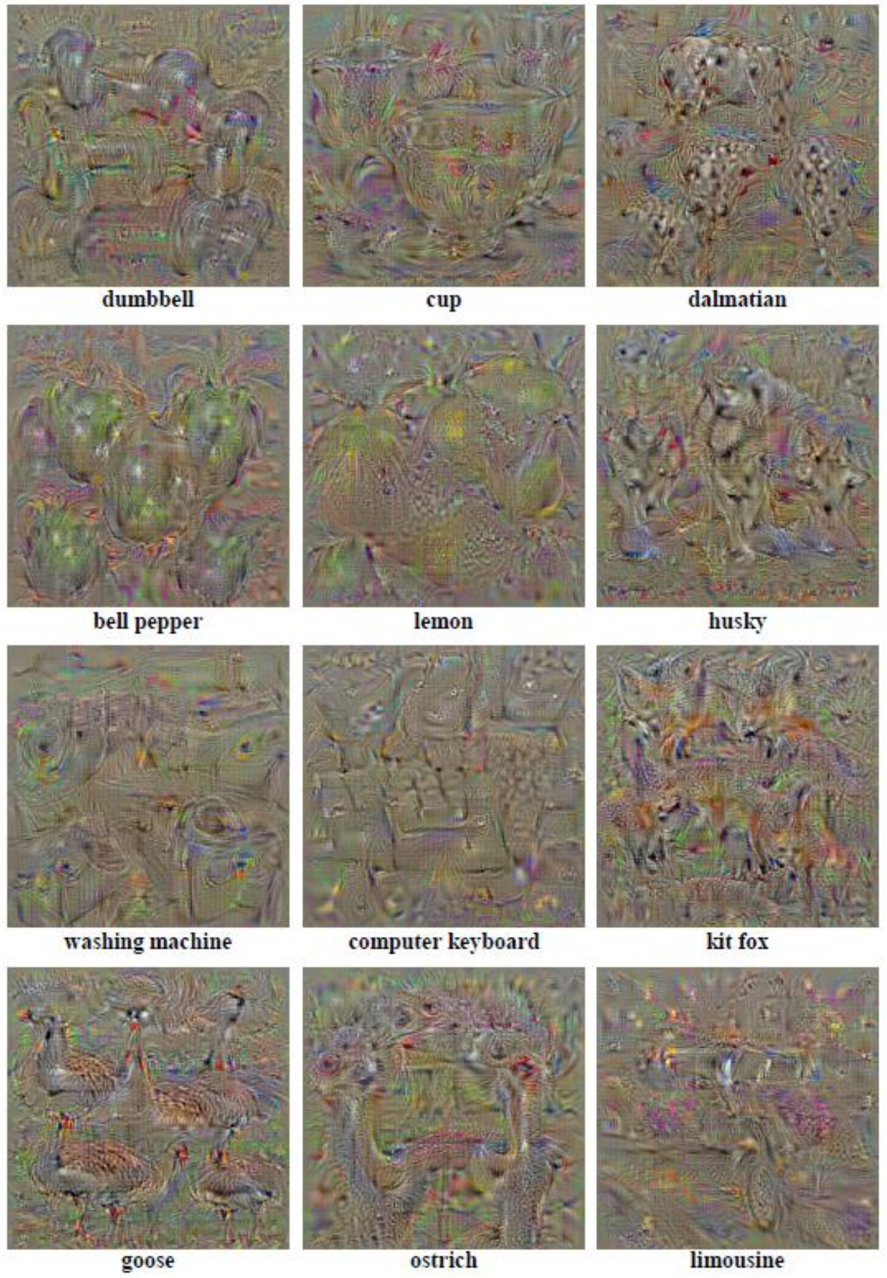
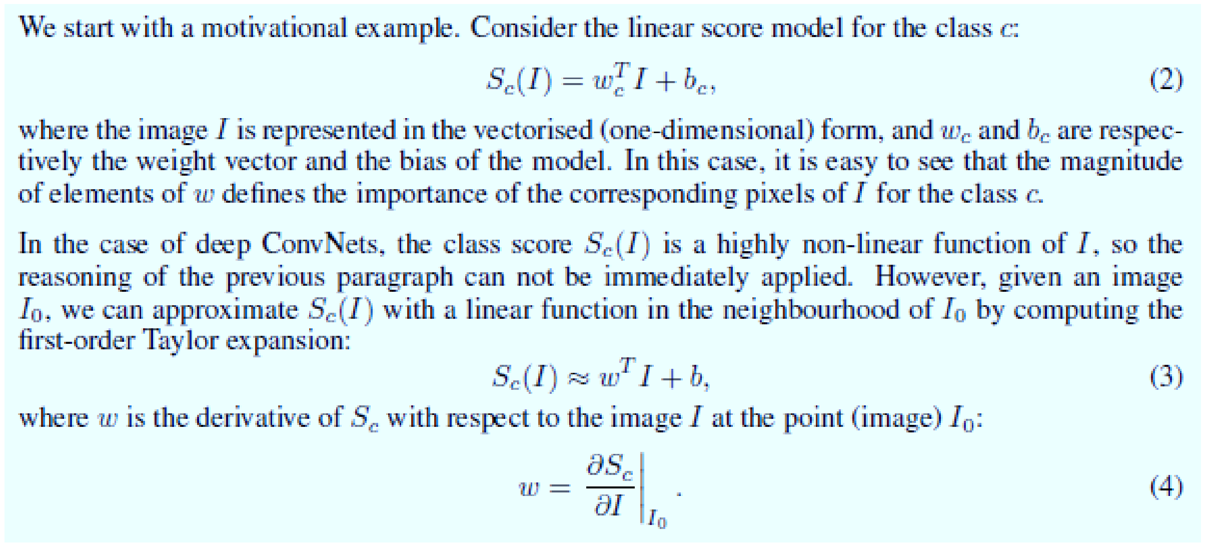


Figure 1: Numerically computed images, illustrating the class appearance models, learnt by a ConvNet, trained on ILSVRC-2013. Note how different aspects of class appearance are captured in a single image. Better viewed in colour.

Previously on CENG403

Data gradients

- The gradient with respect to the input is high for pixels which are on the object


We start with a motivational example. Consider the linear score model for the class c:

$$S_c(I) = w_c^T I + b_c, \quad (2)$$

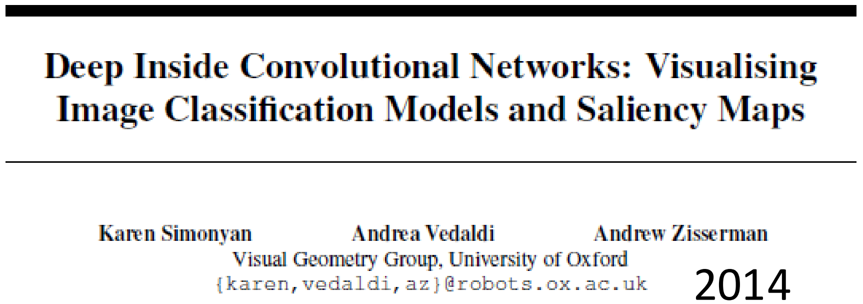
where the image I is represented in the vectorised (one-dimensional) form, and w_c and b_c are respectively the weight vector and the bias of the model. In this case, it is easy to see that the magnitude of elements of w defines the importance of the corresponding pixels of I for the class c.

In the case of deep ConvNets, the class score $S_c(I)$ is a highly non-linear function of I, so the reasoning of the previous paragraph can not be immediately applied. However, given an image I_0, we can approximate $S_c(I)$ with a linear function in the neighbourhood of I_0 by computing the first-order Taylor expansion:

$$S_c(I) \approx w^T I + b, \quad (3)$$

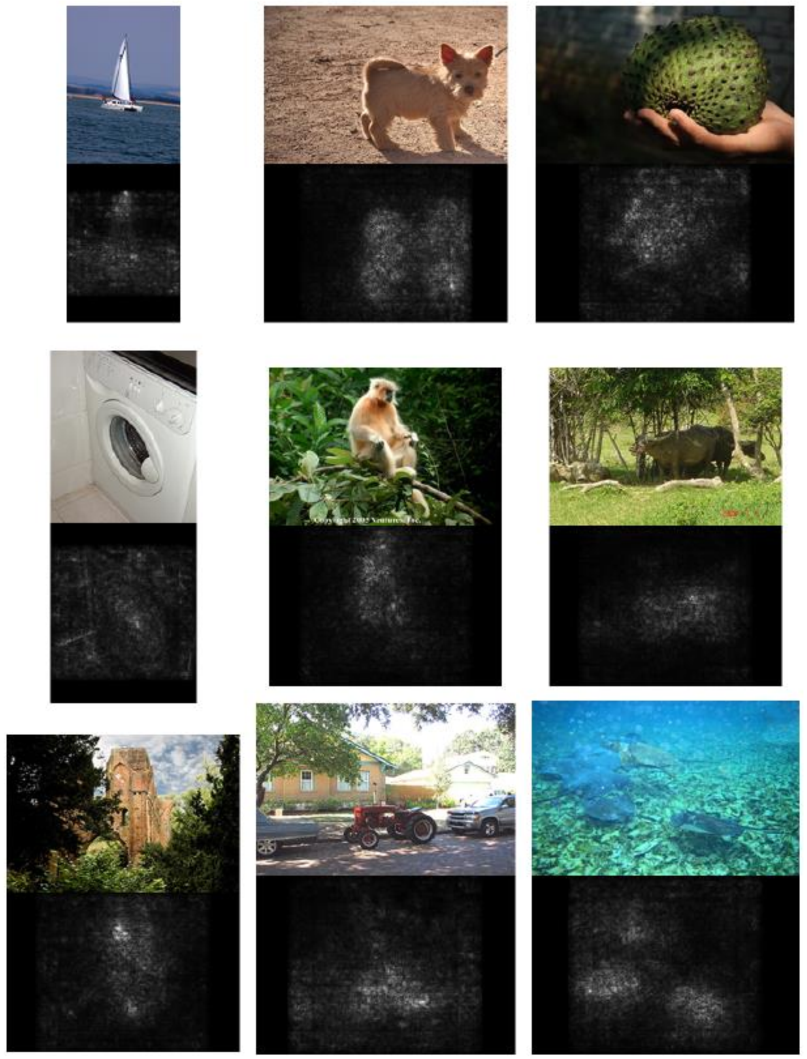
where w is the derivative of S_c with respect to the image I at the point (image) I_0:

$$w = \left.\frac{\partial S_c}{\partial I}\right|_{I_0}. \quad (4)$$


Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps

Karen Simonyan Andrea Vedaldi Andrew Zisserman
Visual Geometry Group, University of Oxford
{karen,vedaldi,az}@robots.ox.ac.uk

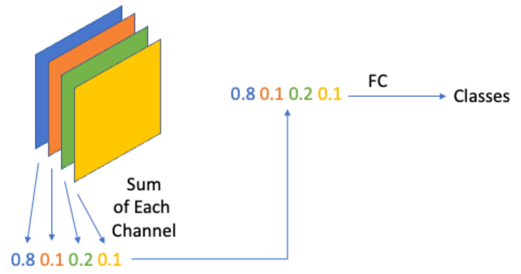
2014



Today

- Convolutional Neural Networks (CNNs)
  - Visualizing/understanding CNNs
  - Well-known CNN architectures

Class Activation Maps



- Weighted combination of the feature maps before GAP:

$$M(x,y) = \sum_k w_k^c f_k(x,y)$$

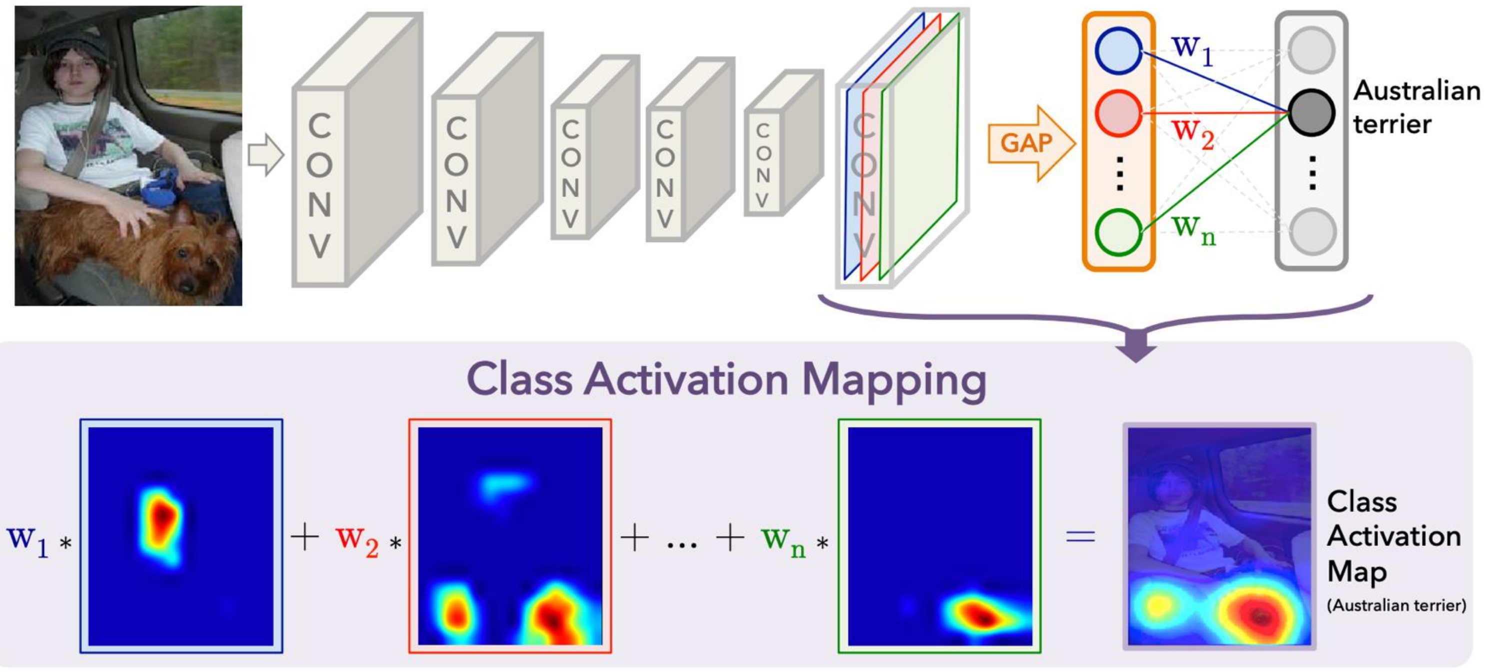


Figure 2. Class Activation Mapping: the predicted class score is mapped back to the previous convolutional layer to generate the class activation maps (CAMs). The CAM highlights the class-specific discriminative regions.

B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2921–2929.

Class Activation Maps

- GradCAM:

$$\alpha_k^c = \sum_{x,y} \frac{\partial S_c}{\partial f_k(x,y)}$$

$$M^c(x,y) = ReLU\left(\sum_k \alpha_k^c \, f_k(x,y)\right)$$

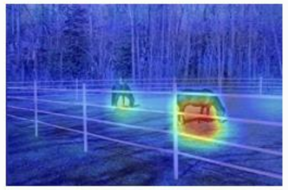
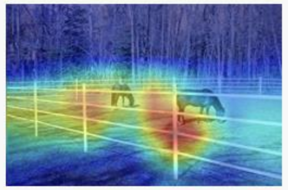
Network	Image	GradCAM
VGG16		
Resnet50		

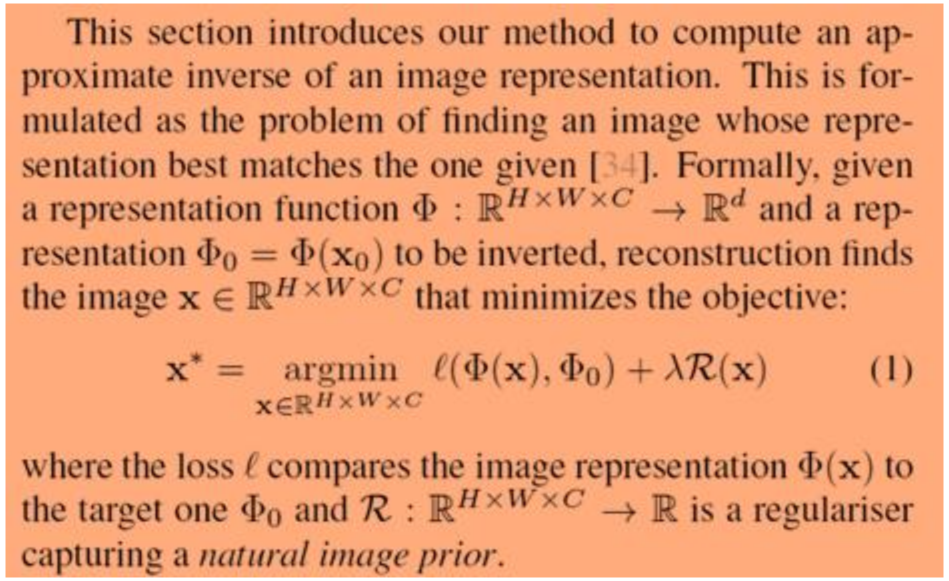
Figure: https://pypi.org/project/grad-cam/

R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, "Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization," arXiv preprint arXiv:1610.02391, 2016.

Chattopadhay, A., Sarkar, A., Howlader, P., & Balasubramanian, V. N. (2018, March). Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 839-847). IEEE.

Feature inversion

- Reconstruct an image from its representation



This section introduces our method to compute an approximate inverse of an image representation. This is formulated as the problem of finding an image whose representation best matches the one given [34]. Formally, given a representation function $\Phi : \mathbb{R}^{H\times W\times C} \to \mathbb{R}^d$ and a representation $\Phi_0 = \Phi(\mathbf{x}_0)$ to be inverted, reconstruction finds the image $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$ that minimizes the objective:

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^{H\times W\times C}} \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda\mathcal{R}(\mathbf{x}) \qquad (1)$$

where the loss ℓ compares the image representation $\Phi(\mathbf{x})$ to the target one Φ_0 and $\mathcal{R} : \mathbb{R}^{H\times W\times C} \to \mathbb{R}$ is a regulariser capturing a *natural image prior*.

Understanding Deep Image Representations by Inverting Them

Aravindh Mahendran
University of Oxford

Andrea Vedaldi
University of Oxford

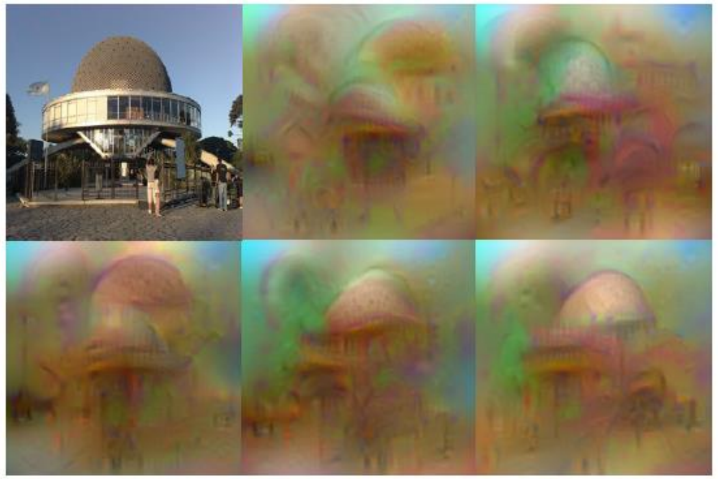


Figure 1. **What is encoded by a CNN?** The figure shows five possible reconstructions of the reference image obtained from the 1,000-dimensional code extracted at the penultimate layer of a reference CNN[13] (before the softmax is applied) trained on the ImageNet data. From the viewpoint of the model, all these images are practically equivalent. This image is best viewed in color/screen.

- Regularization term here is the key factor.
  Example: A combination of the two terms:

$$\mathcal{R}_\alpha(\mathbf{x}) = \|\mathbf{x}\|_\alpha^\alpha \qquad \mathcal{R}_{V^\beta}(\mathbf{x}) = \sum_{i,j}\left((x_{i,j+1} - x_{ij})^2 + (x_{i+1,j} - x_{ij})^2\right)^{\frac{\beta}{2}}$$

Feature inversion with perceptual losses

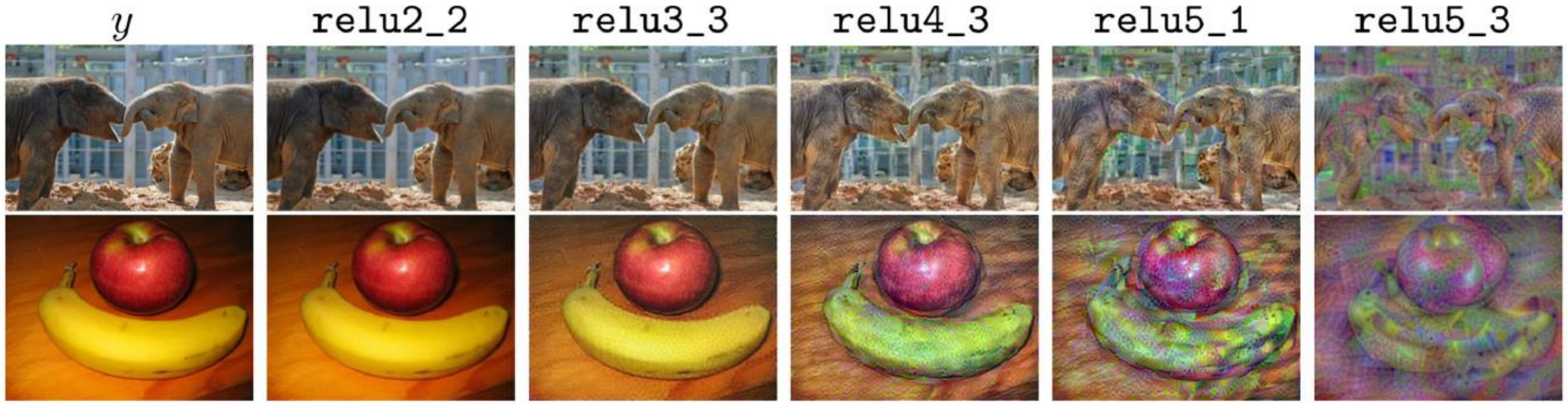


Figure from Johnson, Alahi, and Fei-Fei, “Perceptual Losses for Real-Time Style Transfer and Super-Resolution”, ECCV 2016.

Visualization distill.pub

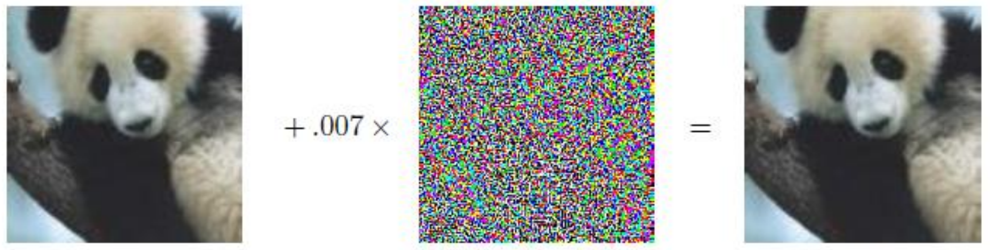
https://distill.pub/2017/feature-visualization/

https://distill.pub/2018/building-blocks/

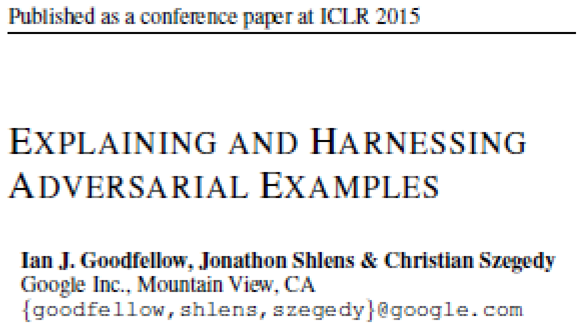
Fooling ConvNets

- Given an image I labeled as y_1, find minimum "r" (noise) such that $I + r$ is classified as a different label, y_2.
- I.e., minimize:

$$\arg\min_{r} loss(I + r, y_2) + c|r|$$



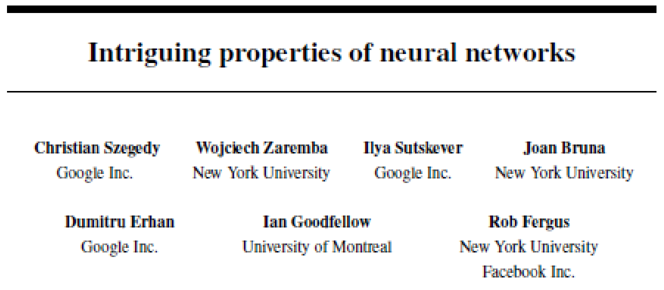
x "panda" 57.7% confidence | $\text{sign}(\nabla_x J(\theta, x, y))$ "nematode" 8.2% confidence | $x + \epsilon\text{sign}(\nabla_x J(\theta, x, y))$ "gibbon" 99.3 % confidence



Published as a conference paper at ICLR 2015

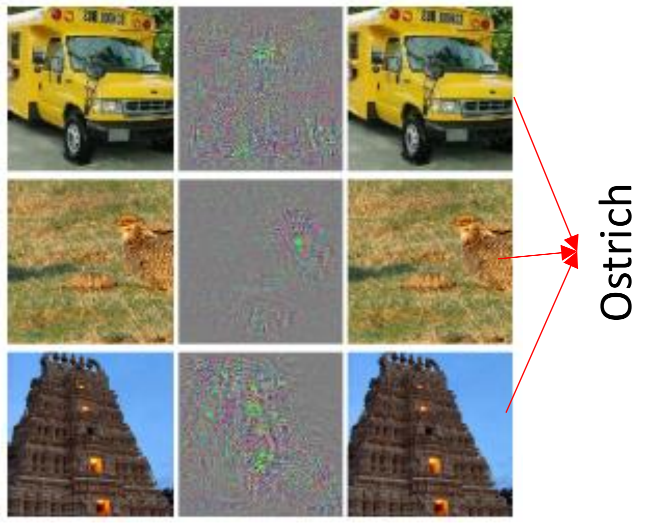
EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES

Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy
Google Inc., Mountain View, CA
{goodfellow,shlens,szegedy}@google.com



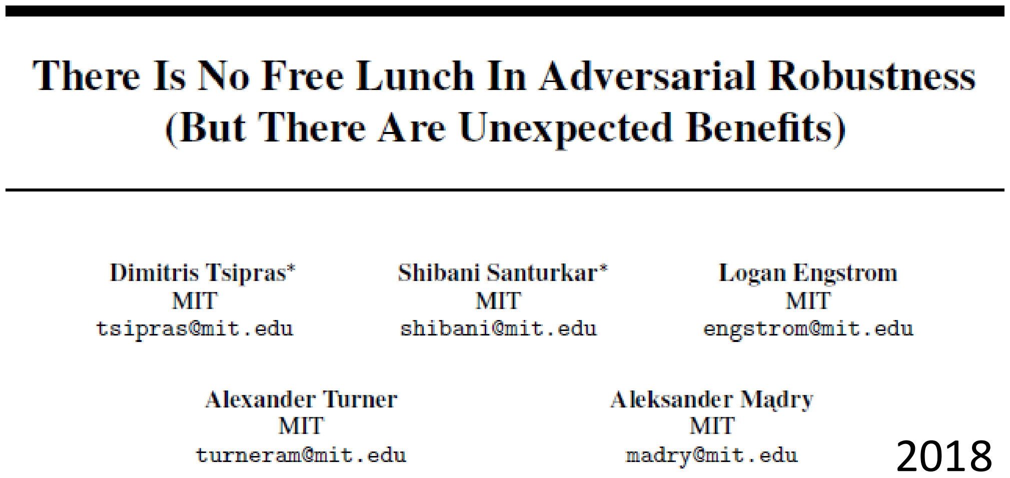
Intriguing properties of neural networks

Christian Szegedy, Google Inc.
Wojciech Zaremba, New York University
Ilya Sutskever, Google Inc.
Joan Bruna, New York University
Dumitru Erhan, Google Inc.
Ian Goodfellow, University of Montreal
Rob Fergus, New York University, Facebook Inc.



More on adversarial examples

- How to classify adversarial examples correctly?
  - You need to train your network against them!
  - That is very expensive and training against all kinds of adversarial examples is not possible
  - However, training against adversarial examples increases accuracy on non-adversarial examples as well.
- This is still an unsolved issue in neural networks
- Adversarial examples are problems of any learning method
- See I. Goodfellow for more on adversarial examples:
  - http://www.kdnuggets.com/2015/07/deep-learning-adversarial-examples-misconceptions.html



There Is No Free Lunch In Adversarial Robustness (But There Are Unexpected Benefits)

Dimitris Tsipras* MIT tsipras@mit.edu
Shibani Santurkar* MIT shibani@mit.edu
Logan Engstrom MIT engstrom@mit.edu
Alexander Turner MIT turneram@mit.edu
Aleksander Mądry MIT madry@mit.edu

2018

- "We provide a new understanding of the fundamental nature of adversarially robust classifiers and how they differ from standard models. In particular, we show that there provably exists a trade-off between the standard accuracy of a model and its robustness to adversarial perturbations. We demonstrate an intriguing phenomenon at the root of this tension: a certain dichotomy between "robust" and "non-robust" features. We show that while robustness comes at a price, it also has some surprising benefits. Robust models turn out to have interpretable gradients and feature representations that align unusually well with salient data characteristics. In fact, they yield striking feature interpolations that have thus far been possible to obtain only using generative models such as GANs."

Well-known CNN architectures

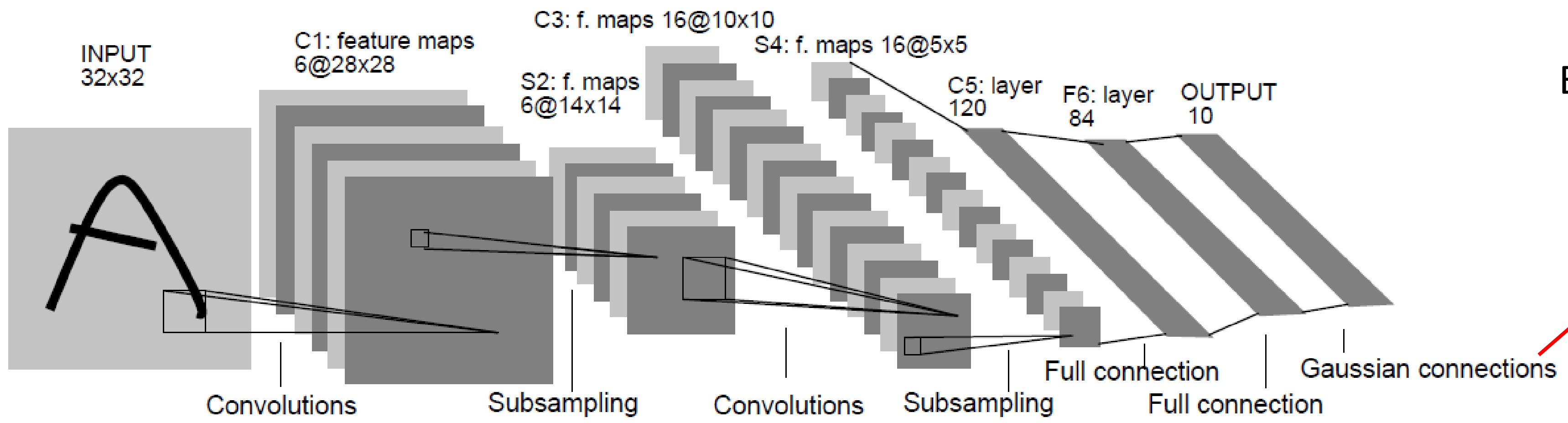
LeNet (1998)

PROC. OF THE IEEE, NOVEMBER 1998

Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

- For reading zip codes and digits



Euclidean RBF:

$$y_i = \sum_j (x_j - w_{ij})^2.$$

F=5x5
S=1
P=0

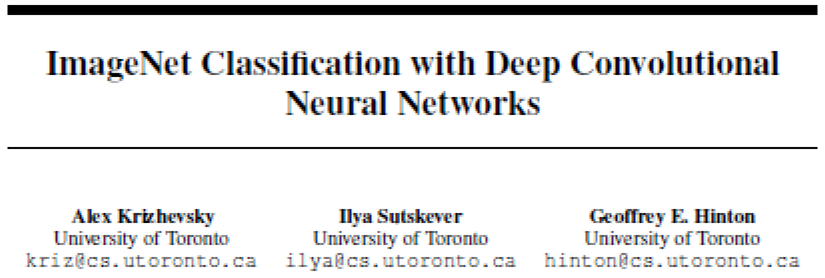
F=2x2
S=2
sigm($\alpha \times avg + b$)
α & b: trainable

F=5x5
S=1
P=0

F=2x2
S=2
sigm($\alpha \times avg + b$)
α & b: trainable

F=5x5
S=1
P=0

AlexNet (2012)



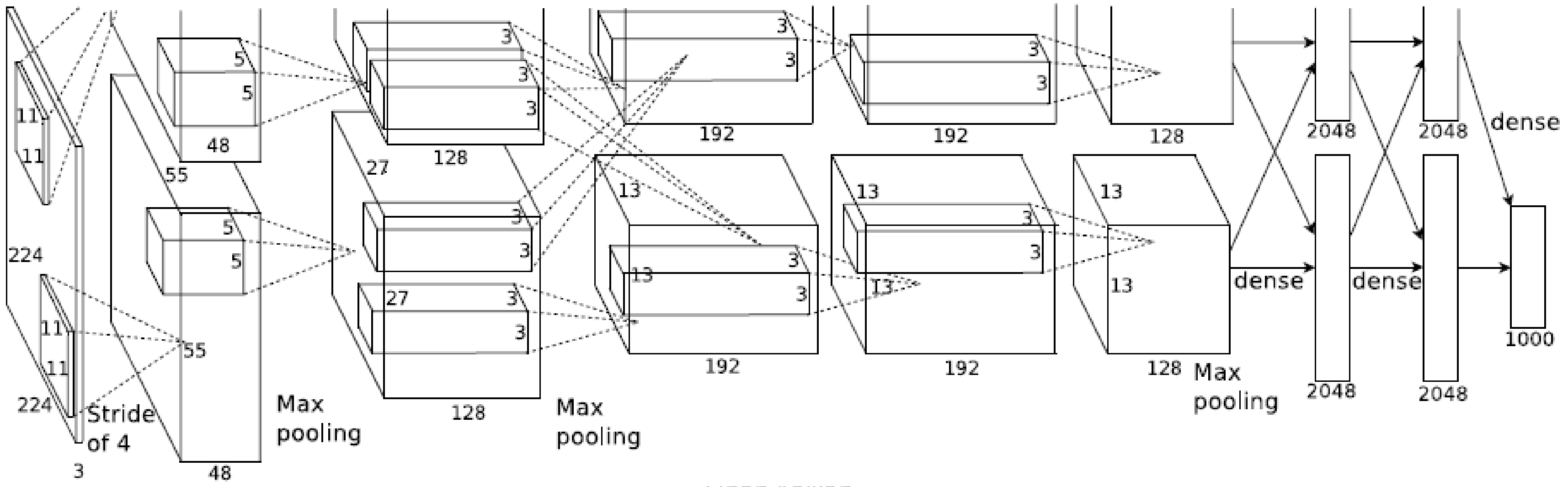
ImageNet Classification with Deep Convolutional Neural Networks

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

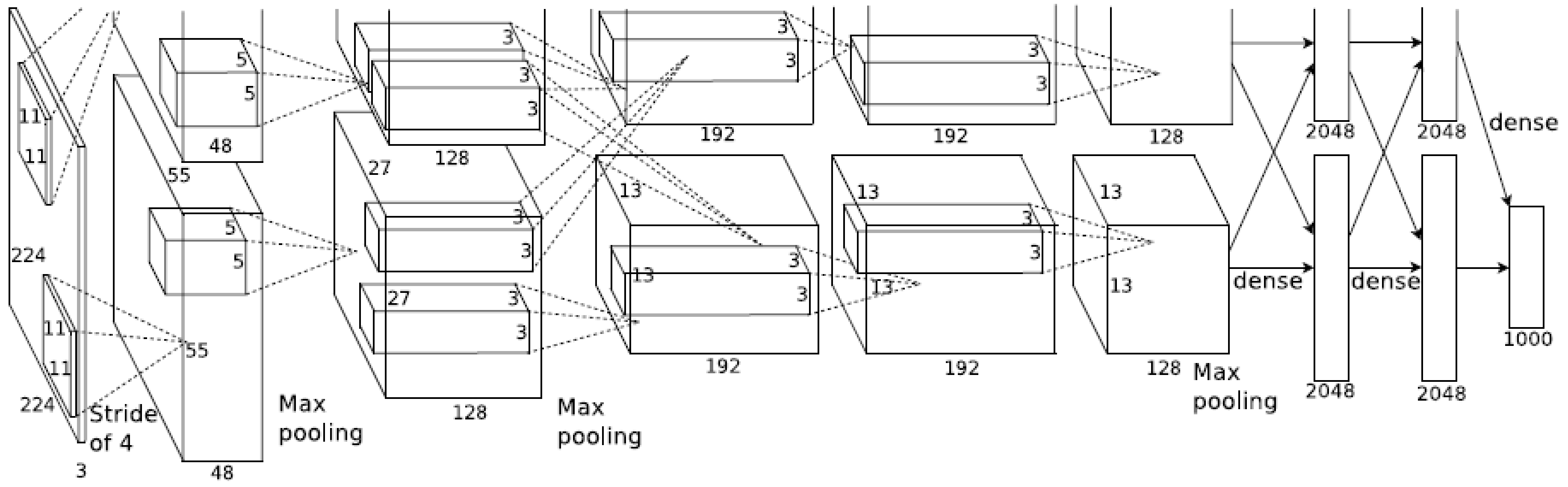
Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

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

- Popularized CNN in computer vision & pattern recognition
- ImageNet ILSVRC challenge 2012 winner
- Similar to LeNet
  - Deeper & bigger
  - Many CONV layers on top of each other (rather than adding immediately a pooling layer after a CONV layer)
  - Uses GPU
- 650K neurons. 60M parameters. Trained on 2 GPUs for a week.

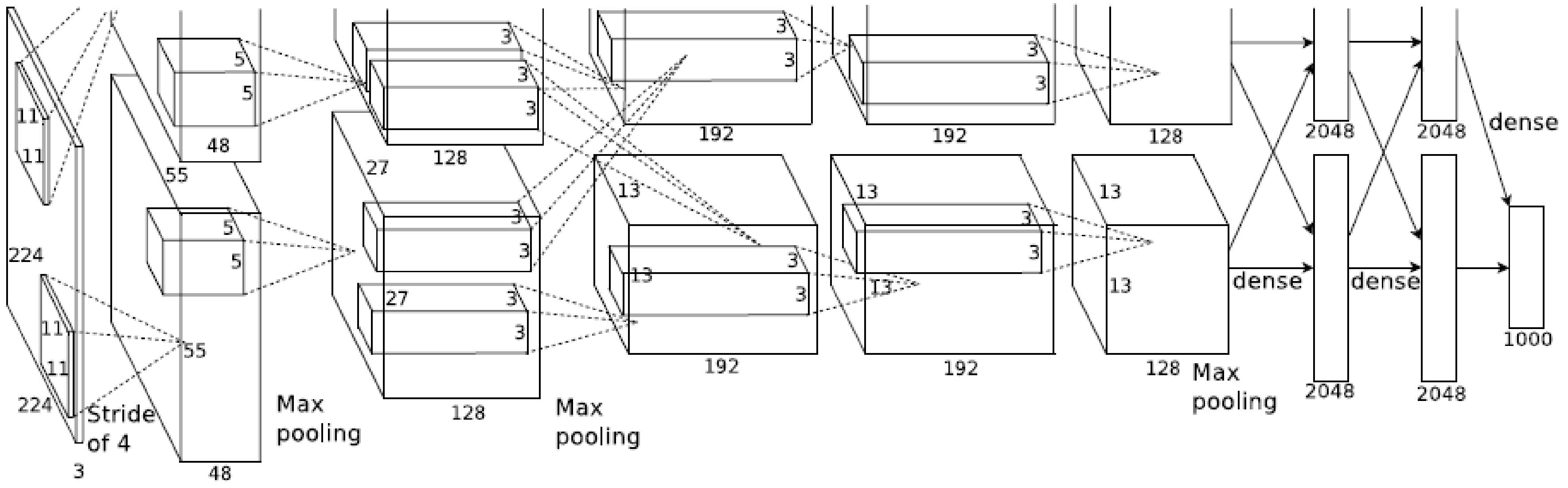


AlexNet (2012) Details



- Since the network is too big to fit in on GPU, it is divided into two.
- Note the cross connections between the “pathways”.
- Uses ReLU as non-linearity after every convolutional and fully-connected layer.
- Normalization layer is placed after the first & the second convolutional layers.
- Max-pooling layer is placed only after the normalization layers & the fifth convolutional layer.
- Last layer is a soft-max.

AlexNet (2012) Training



- Data augmentation & dropout are used during training to avoid overfitting.
- Stochastic Gradient Descent with a batchsize of 128 examples is used.
- Momentum with coefficient 0.9 is employed.
- Weight decay (L2 regularization cost) with factor 0.0005 is also used in the loss function.
- Weights are initialized from a zero-mean Gaussian distribution with 0.01 std.
- Learning rate started with 0.01 and manually divided by 10 when the validation error rate stopped improving.
- Trained on 1.2 million images, which took 5-6 days on two GPUs.

AlexNet (2012): The learned filters

- Do you notice anything strange with the filters?

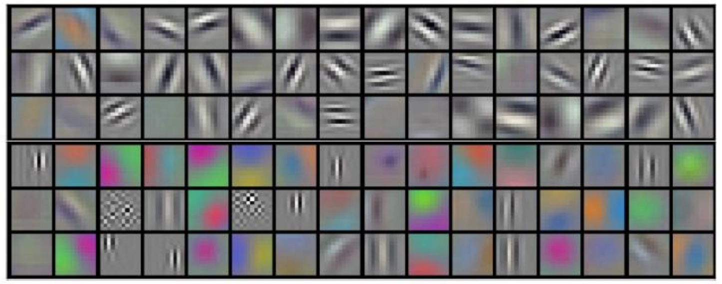


Figure 3: 96 convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer on the $224 \times 224 \times 3$ input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2. See Section 6.1 for details.

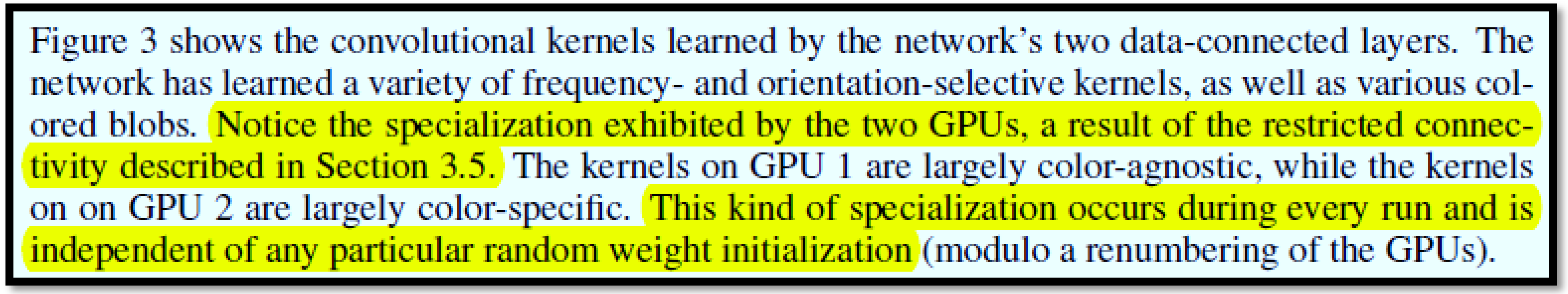
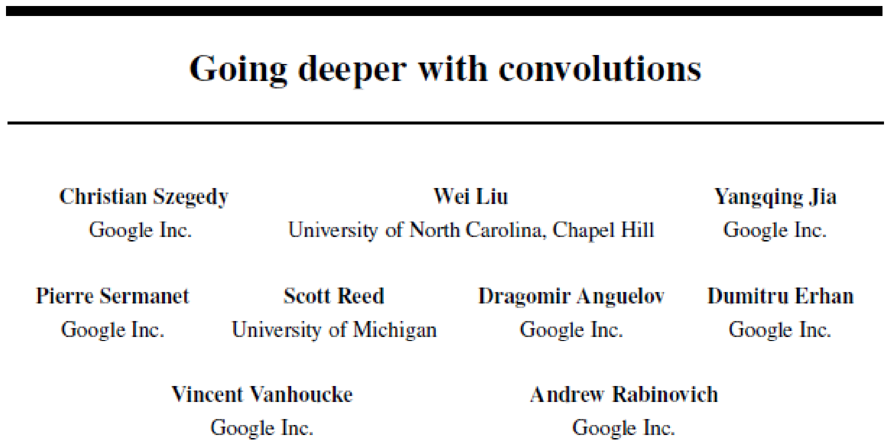


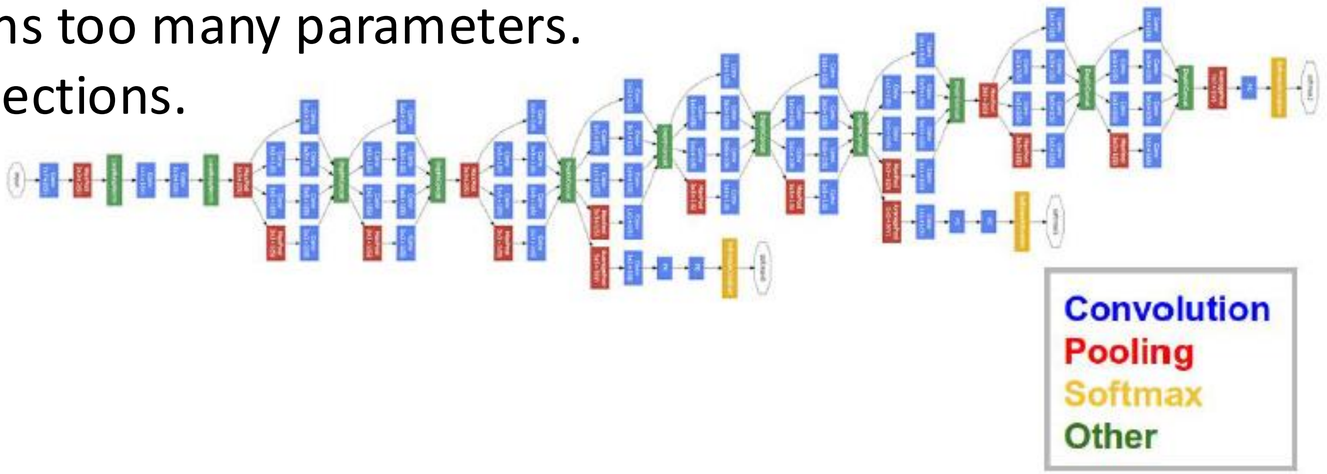
Figure 3 shows the convolutional kernels learned by the network's two data-connected layers. The network has learned a variety of frequency- and orientation-selective kernels, as well as various colored blobs. Notice the specialization exhibited by the two GPUs, a result of the restricted connectivity described in Section 3.5. The kernels on GPU 1 are largely color-agnostic, while the kernels on on GPU 2 are largely color-specific. This kind of specialization occurs during every run and is independent of any particular random weight initialization (modulo a renumbering of the GPUs).

GoogleNet (2014)

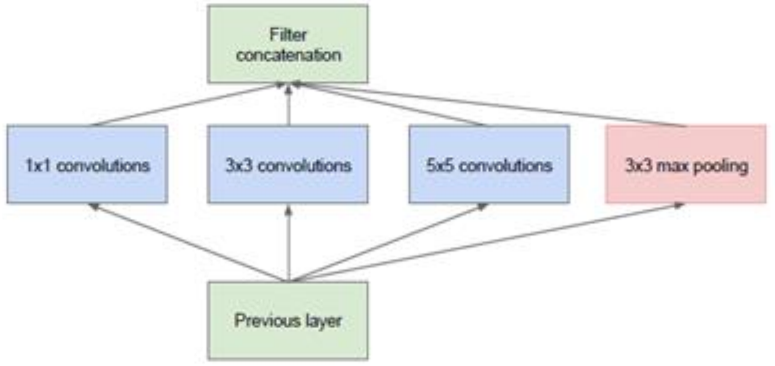

Going deeper with convolutions

Christian Szegedy, Google Inc.
Wei Liu, University of North Carolina, Chapel Hill
Yangqing Jia, Google Inc.
Pierre Sermanet, Google Inc.
Scott Reed, University of Michigan
Dragomir Anguelov, Google Inc.
Dumitru Erhan, Google Inc.
Vincent Vanhoucke, Google Inc.
Andrew Rabinovich, Google Inc.

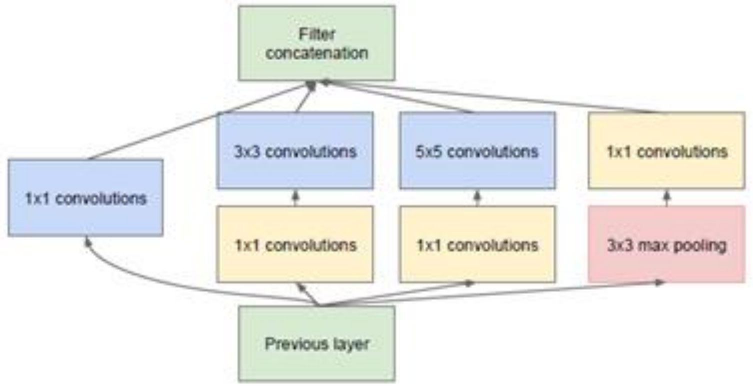
- ImageNet 2014 winner
- Contributions:
  - Inception module
    - Dramatically reduced parameters (from 60M in AlexNet to 4M)
  - Avg Pooling at the top, instead of fully-connected layer ➔ Reduced number of parameters
- Motivation:
  - Going bigger (in depth or width) means too many parameters.
  - Go bigger by maintaining sparse connections.



Inception module: “network in network” (inspired from Lin et al., 2013)

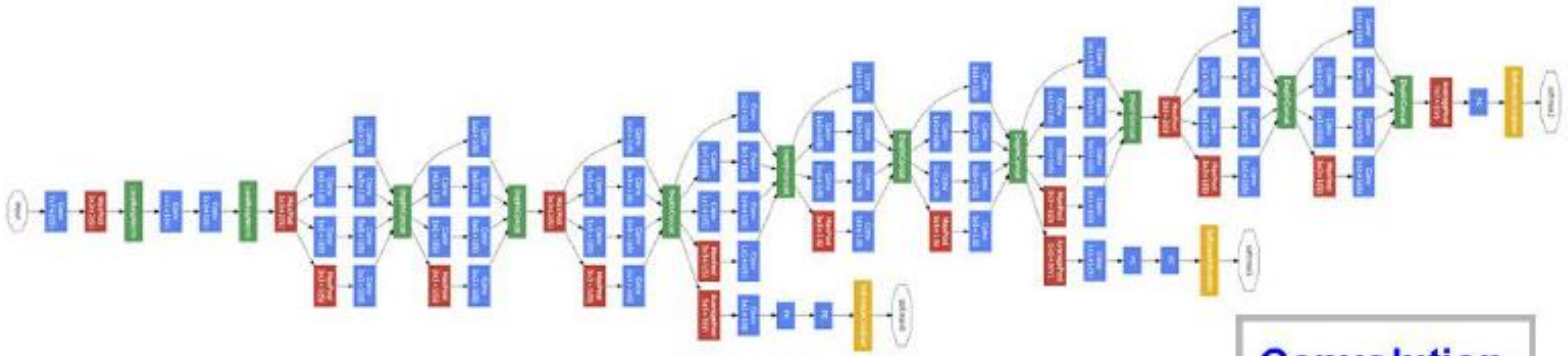


(a) Inception module, naïve version

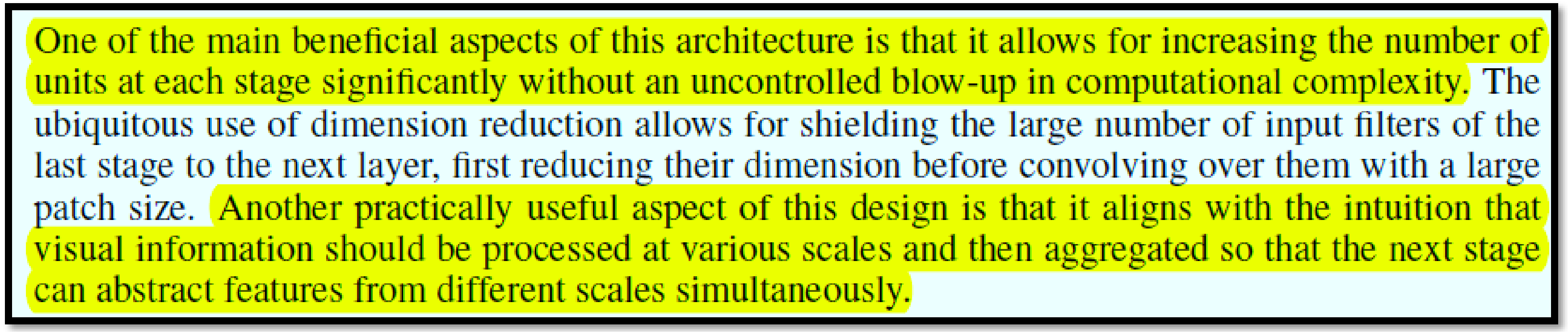


(b) Inception module with dimension reductions

- Concatenation is performed along the “channels” (depth).
  - The output of inception layers must have the same size.
- The naïve version has a tendency to blow up in number of channels.
  - Why? Max-pooling does not change the number of channels. When concatenated with other filter responses, number of channels increase with every layer.
  - Solution: Do 1x1 convolution to decrease the number of channels.
    - Also called “bottleneck”.
- In order to decrease the computational complexity of 3x3 and 5x5 pooling, they are also preceded by 1x1 convolution (i.e., the number of channels are reduced).

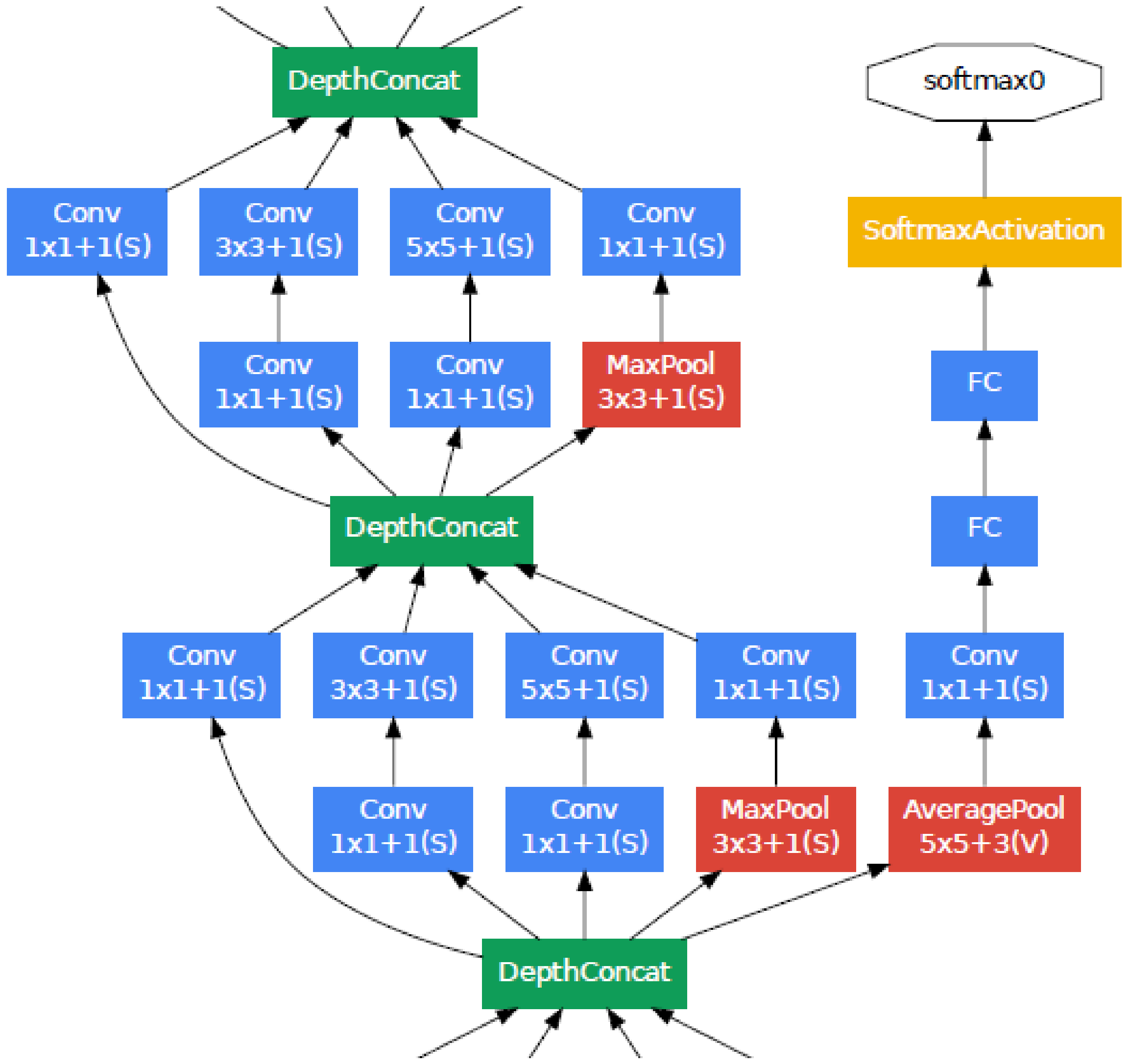


Convolution
Pooling
Softmax
Other



One of the main beneficial aspects of this architecture is that it allows for increasing the number of units at each stage significantly without an uncontrolled blow-up in computational complexity. The ubiquitous use of dimension reduction allows for shielding the large number of input filters of the last stage to the next layer, first reducing their dimension before convolving over them with a large patch size. Another practically useful aspect of this design is that it aligns with the intuition that visual information should be processed at various scales and then aggregated so that the next stage can abstract features from different scales simultaneously.

- Since the intermediate layers learn to discriminate features specific to a class, we can directly link them to the loss term.
    - Encourages these layers to become more discriminative
    - Increases propagation of gradient signal to earlier stages



GoogleNet: More Details

- ReLU after all layers
- Max pooling in inception modules as well as a whole layer occasionally
- Avg pooling instead of fully-connected layers
  - Only a minor change in the accuracy (0.6%)
  - However, less number of parameters
- Other usual tricks (e.g., dropout, augmentation etc.) are used.
- Trained on CPUs using a distributed machine learning system.
- SGD with momentum (0.9).
- Fixed learning rate scheme with 4% decrease every 8 epochs
- They trained many different models with different initializations and parameters. They combined these models using different methods and tricks. There is no single training method that yields the results they achieved.

VGGNet (2014)

- ImageNet runner up in 2014
- Contribution:
  - Use small RFs & increase depth as much as possible
  - 16 CONV/FC layers.
  - 3x3 CONVs and 2x2 pooling from beginning to the end
- Although performs slightly worse than GoogleNet in image classification, VGGNet may perform better at other tasks (such as transfer learning problems).
- Downside: Needs a lot of memory & parameters (140M)

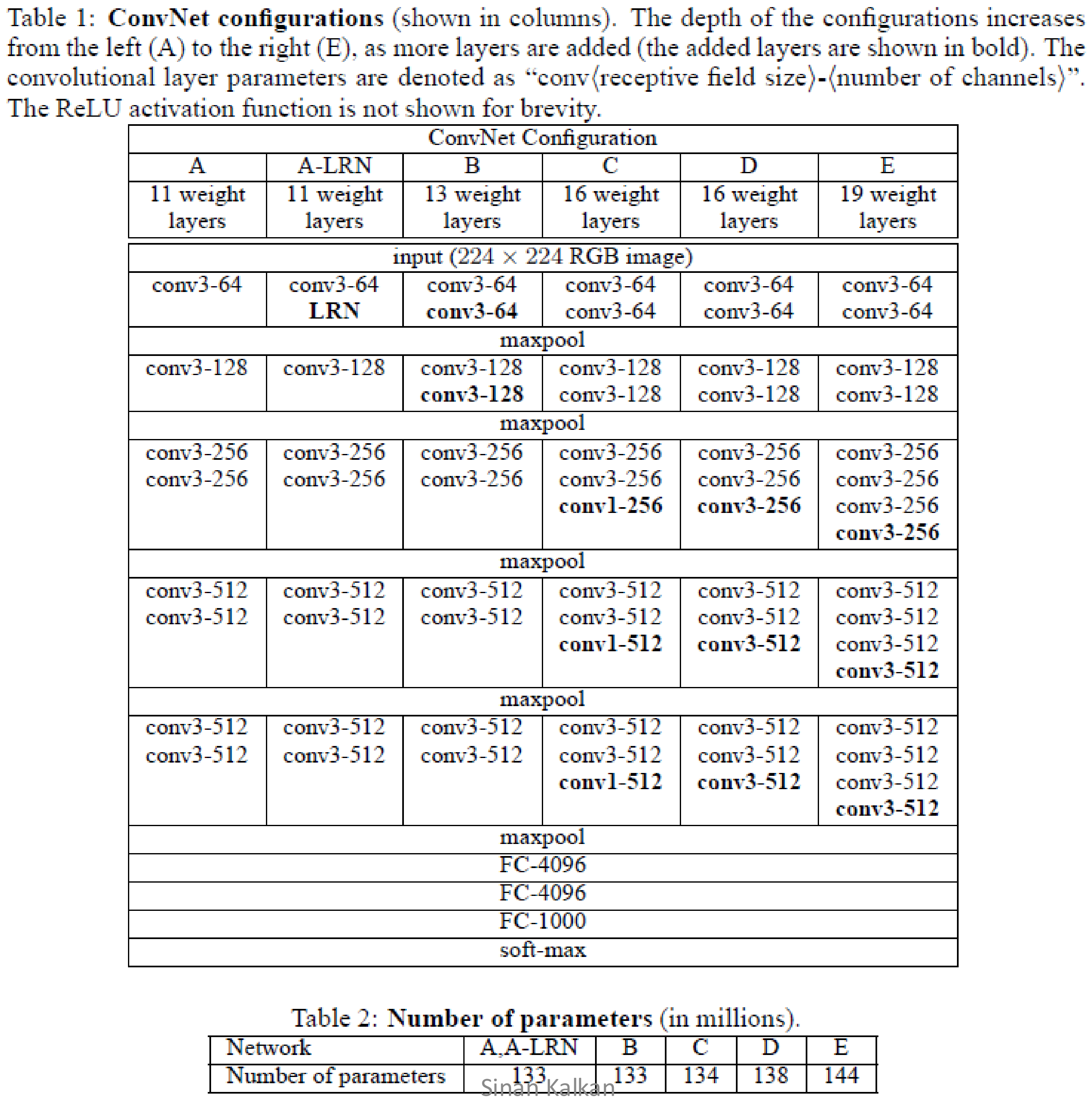


Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

ResNet (2015)

- Increasing the depth naively may not give you better performance after a number of depths
- Why?
  - This is shown to be not due to overfitting (since training error also gets worse) or vanishing gradients (suitable non-linearities used)
  - Accuracy is somehow saturated. Though reported in several studies.
- Solution: Make shortcut connections

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

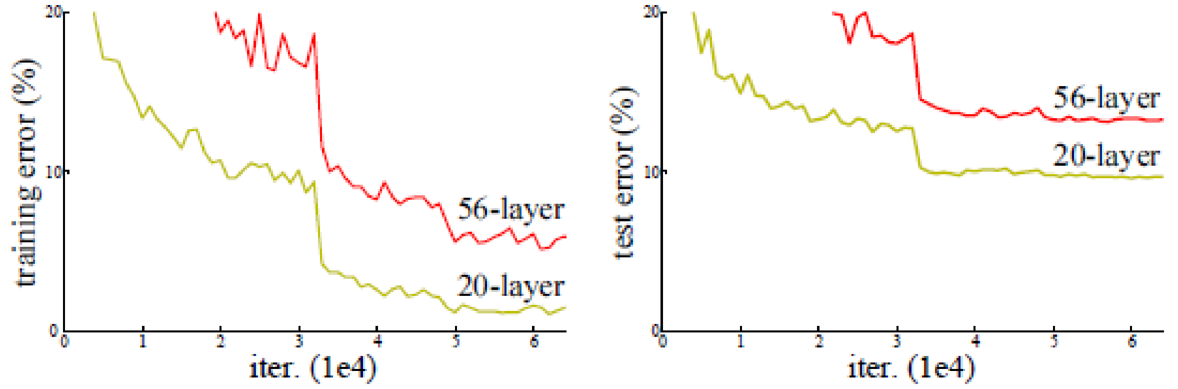
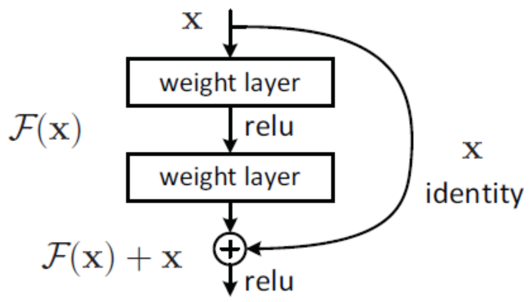


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

ResNet (2015)

- Residual (shortcut) connections



x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2. Residual learning: a building block.

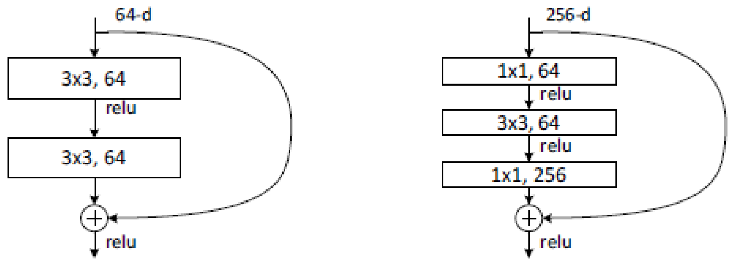
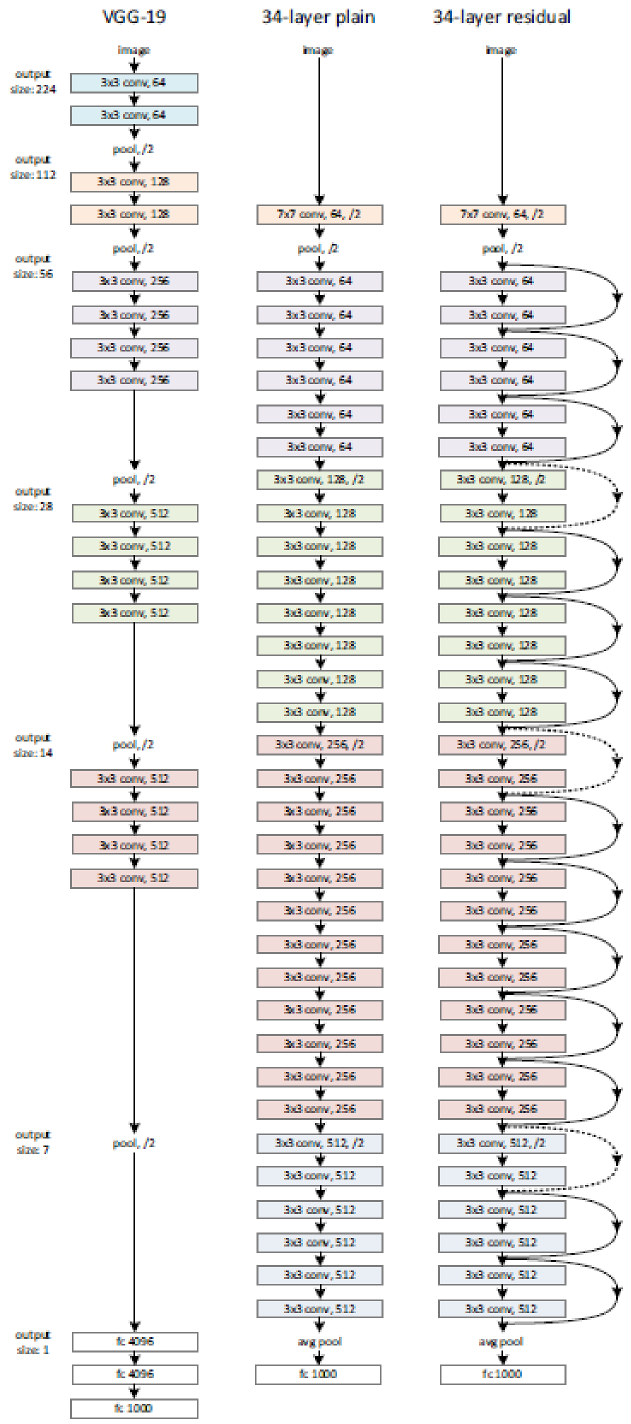


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.



ResNet (2015)

- Residual (shortcut) connections

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43[†]
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except [†] reported on the test set).

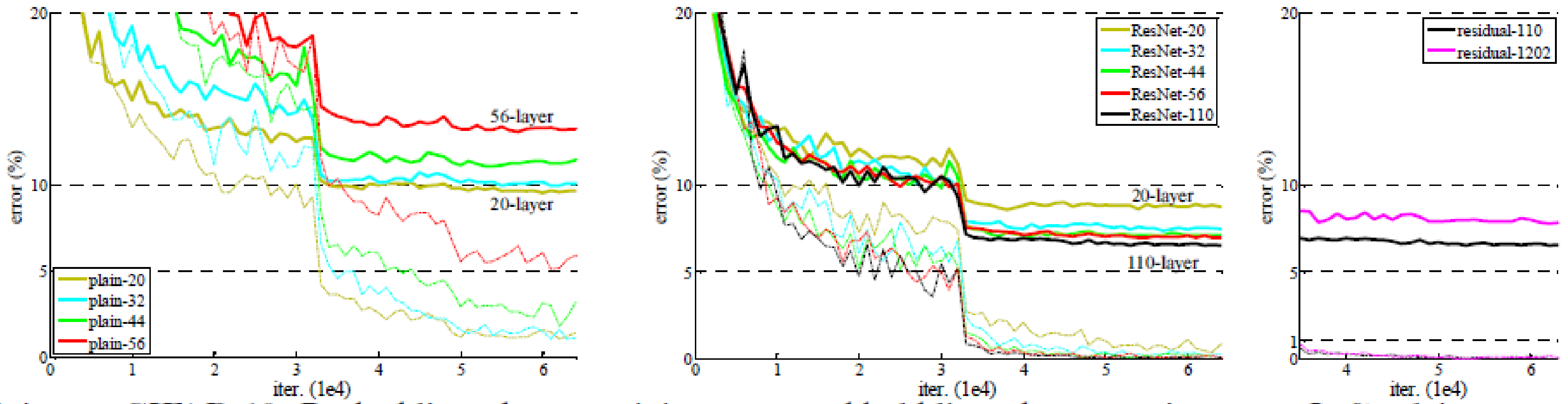
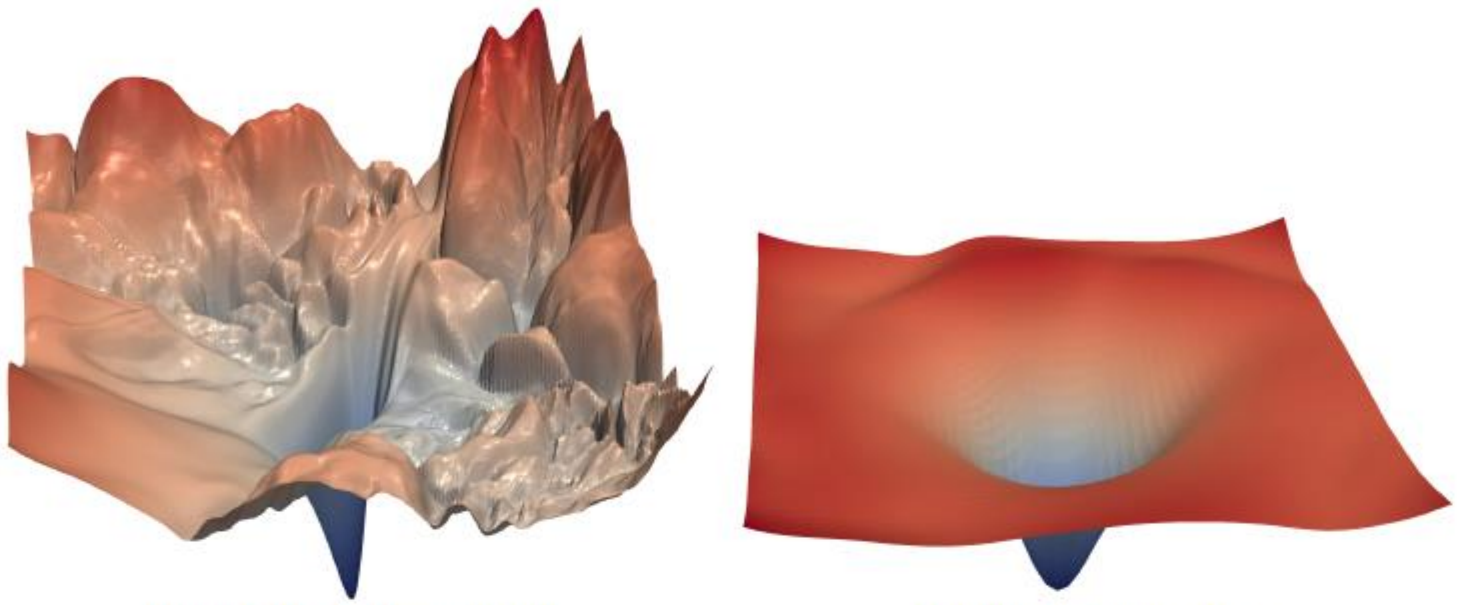


Figure 6. Training on **CIFAR-10**. Dashed lines denote training error, and bold lines denote testing error. **Left**: plain networks. The error of plain-110 is higher than 60% and not displayed. **Middle**: ResNets. **Right**: ResNets with 110 and 1202 layers.

Effect of residual connections



(a) without skip connections (b) with skip connections

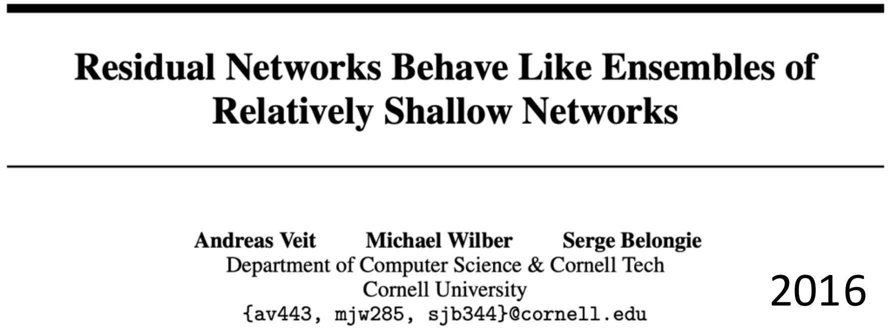
Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The vertical axis is logarithmic to show dynamic range. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

VISUALIZING THE LOSS LANDSCAPE OF NEURAL NETS

2018

Hao Li[1], Zheng Xu[1], Gavin Taylor[2], Christoph Studer[3], Tom Goldstein[1]
[1]University of Maryland, College Park, [2]United States Naval Academy, [3]Cornell University
{haoli,xuzh,tomg}@cs.umd.edu, taylor@usna.edu, studer@cornell.edu

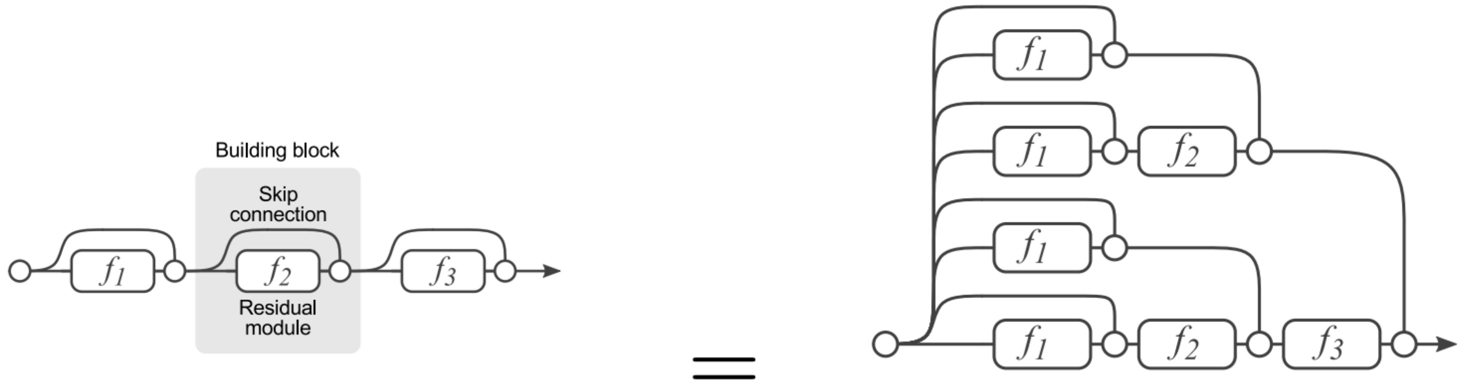
ResNet: Ensemble of Shallow Networks



Residual Networks Behave Like Ensembles of Relatively Shallow Networks

Andreas Veit **Michael Wilber** **Serge Belongie**
Department of Computer Science & Cornell Tech
Cornell University
{av443, mjw285, sjb344}@cornell.edu

2016



(a) Conventional 3-block residual network (b) Unraveled view of (a)

Figure 1: Residual Networks are conventionally shown as (a), which is a natural representation of Equation (1). When we expand this formulation to Equation (6), we obtain an *unraveled view* of a 3-block residual network (b). Circular nodes represent additions. From this view, it is apparent that residual networks have $O(2^n)$ implicit paths connecting input and output and that adding a block doubles the number of paths.