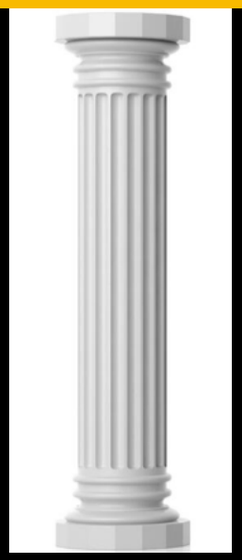
Supervised Learning:
A Deeper Dive into Algorithms and their Visualizations

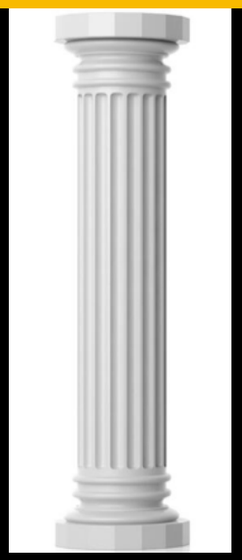
Earl Wong

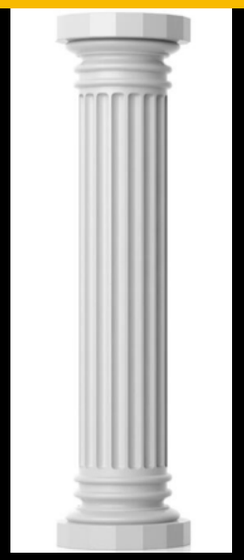
Dataset

Algorithm (Architecture)

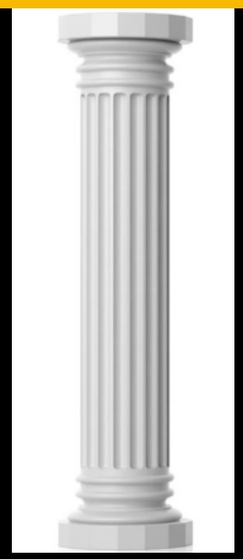
Loss Function







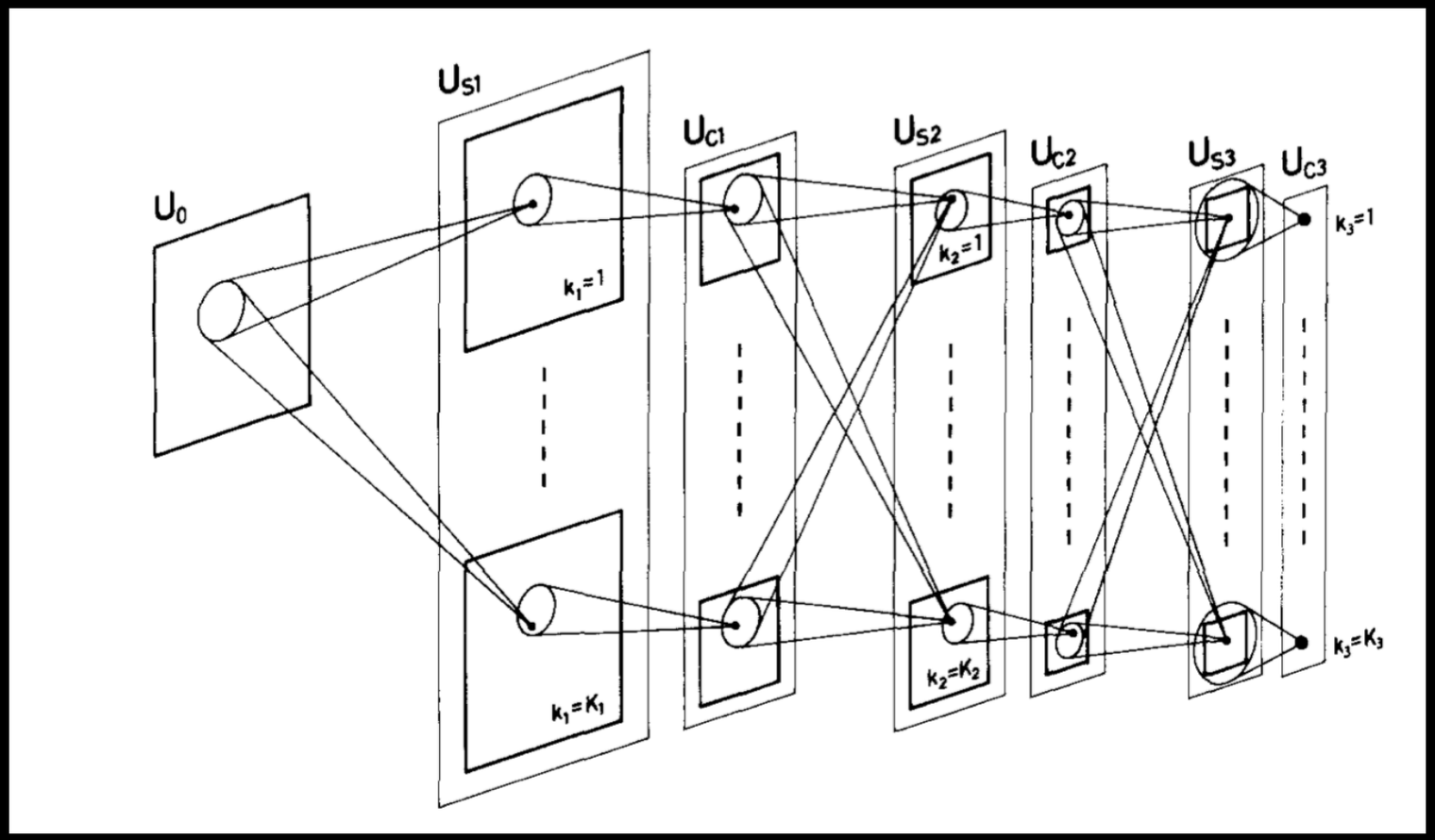
Algorithm (Architecture)



Convolutional Neural Networks Architectures - The Big 3

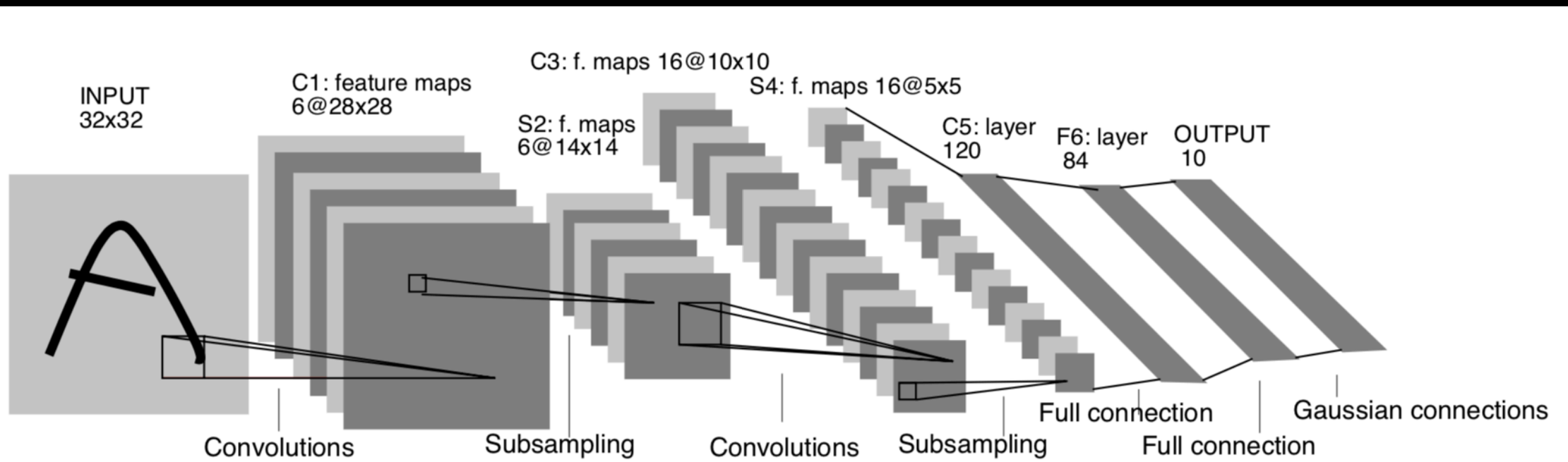
- Neocognitron
- LeNet5
- AlexNet

Neocognitron



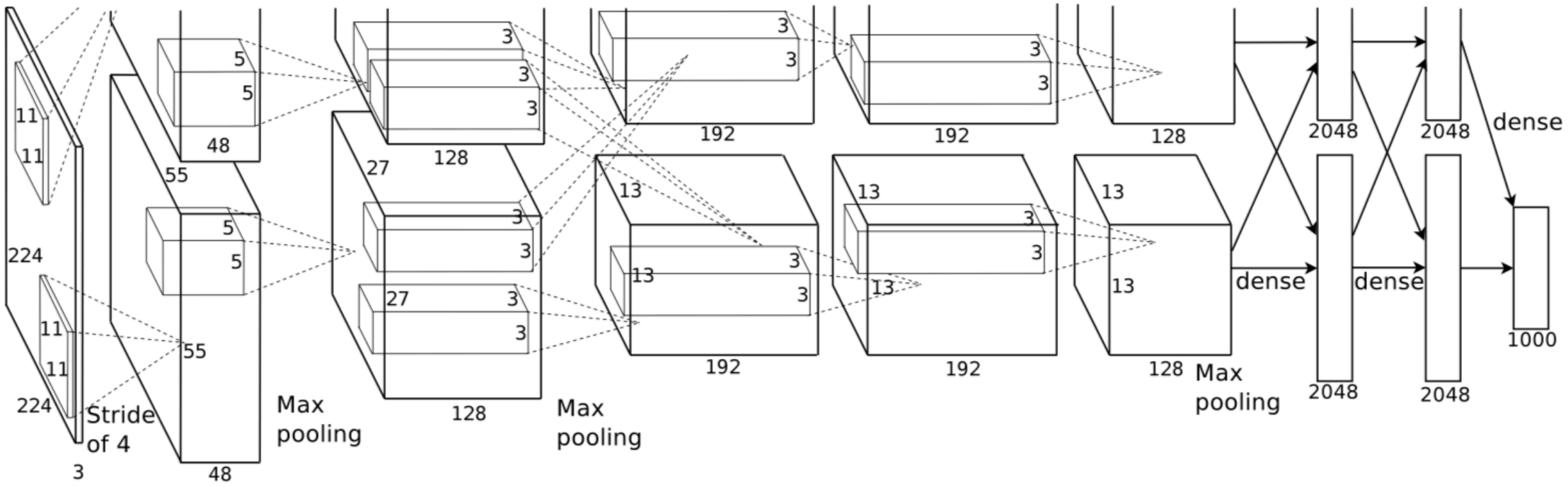
1) Fukushima modeled the Neocognitron architecture based on the hierarchical model of the human visual system (HVS) proposed by Hubel and Wiesel.
2) At the time, back propagation had not yet been invented.
3) Hence the network self organized by “learning without a teacher”.
4) Neocognitron acquired the ability to recognize stimulus patterns based on geometrical similarity, without being affected by their positions. i.e. Convolutions

LeNet5



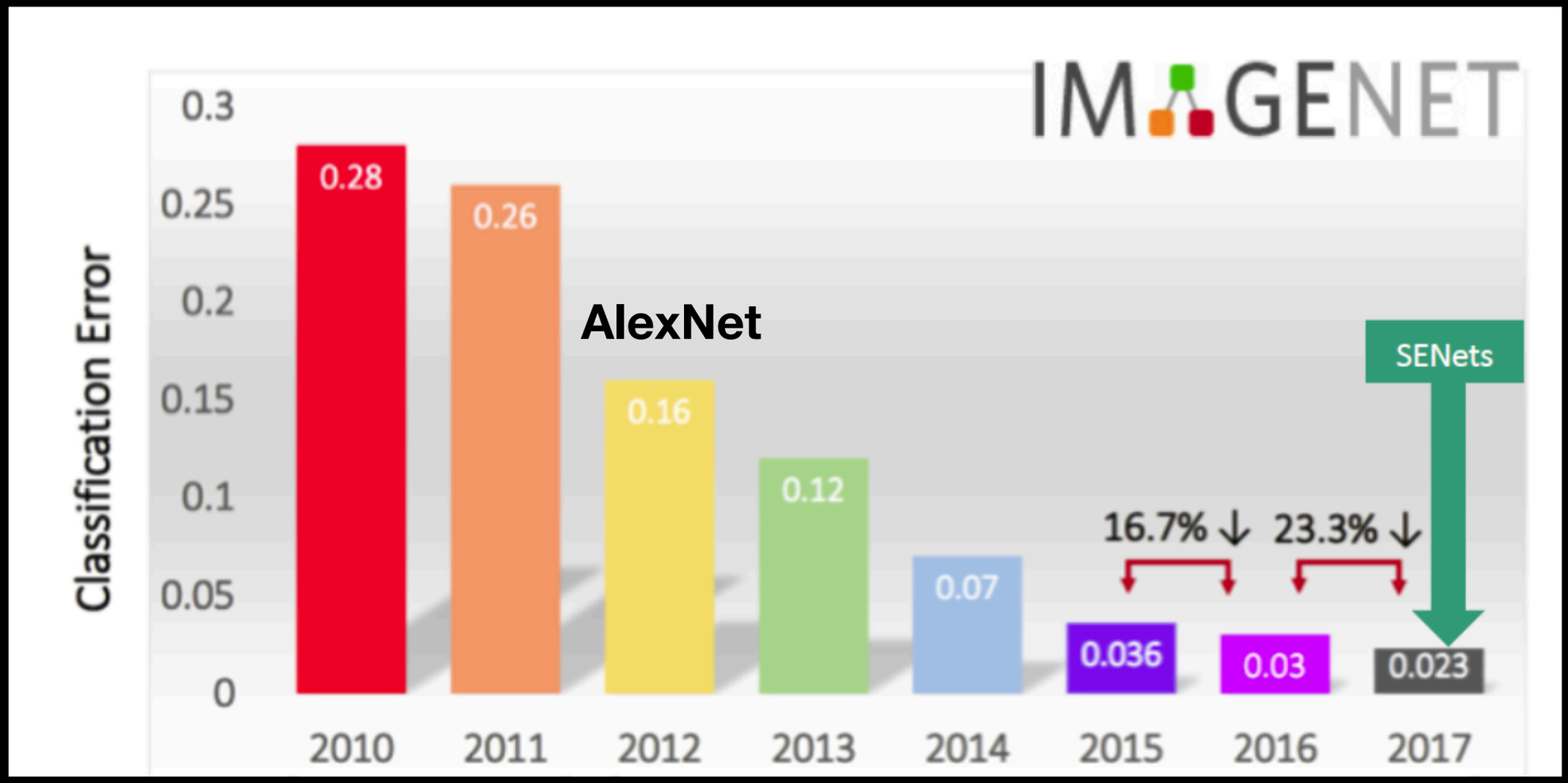
1) This was the first commercial application of a convolutional neural network (CNN).
2) The CNN was applied to handwritten character recognition by LeCun.
3) The CNN was trained using gradient descent.
4) The architecture used 60K weight parameters - very big, for that time.

AlexNet



1) This network shattered the Imagenet Image Recognition Challenge by double digits.
2) This was the first deep convolutional neural network.
3) The network used 60 million weight parameters and needed to be split into two parts, in order to be trained.

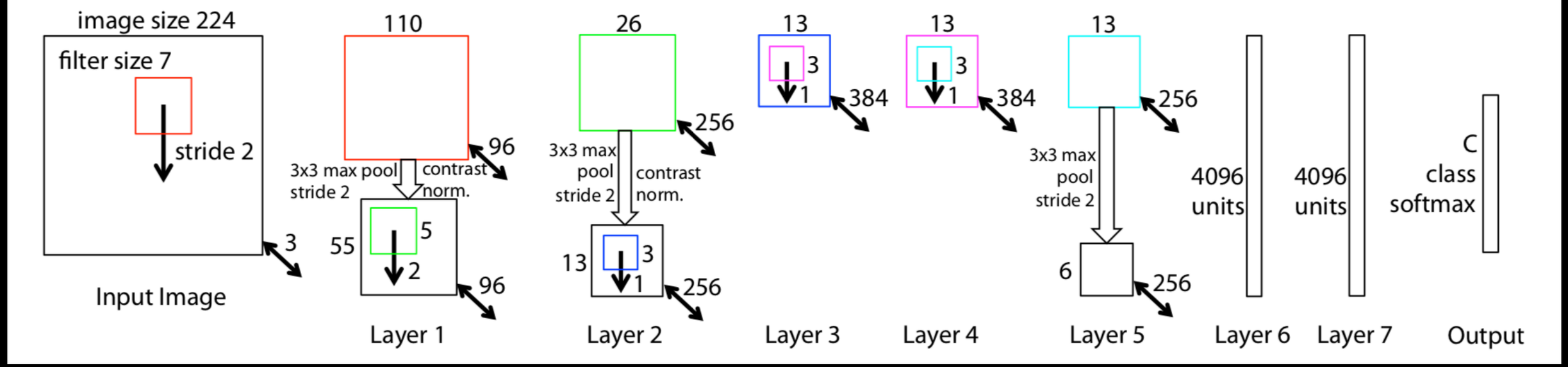
The Progression



Progression

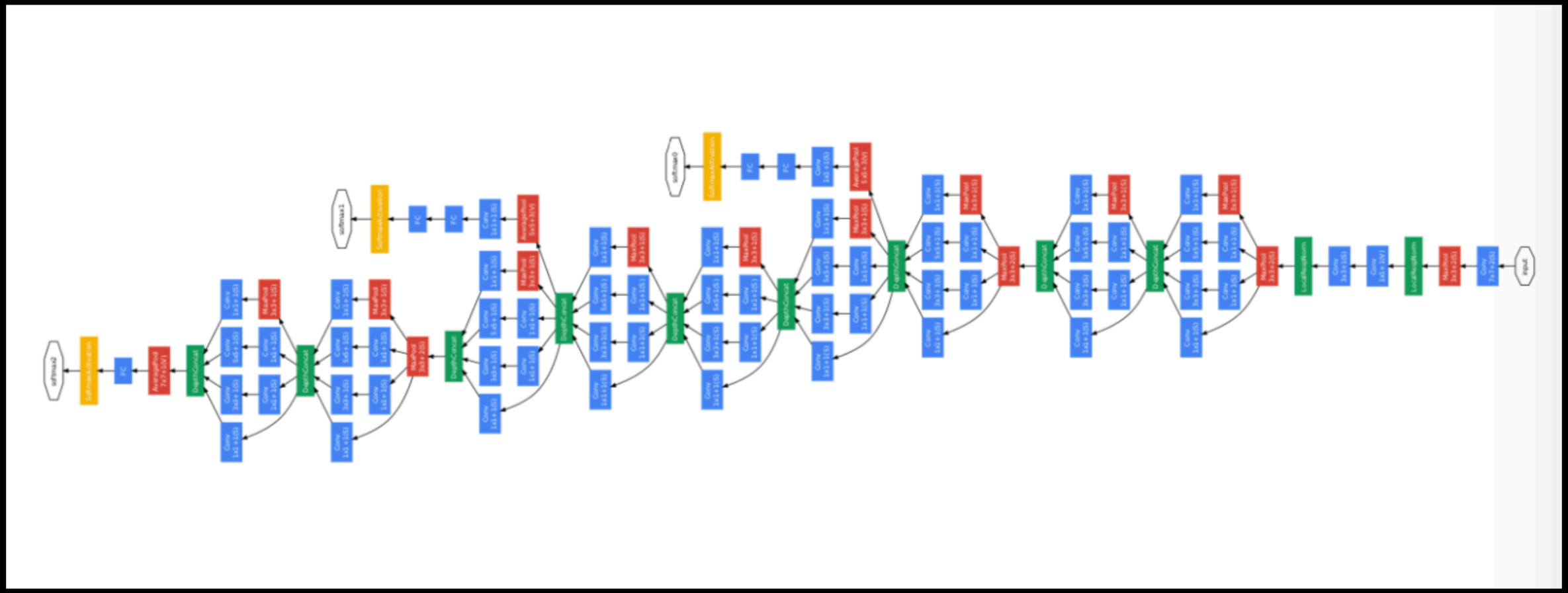
- 2013-ZFNet
- 2014-GoogleNet & VGGNet
- 2015-ResNet
- 2016-ResNext
- 2017-SqueezeAndExciteNet, MobileNet, ShuffleNet
- 2018/2019- AmoebaNet, NASNet
- 2020-EfficientNet

2013 - ZFNet



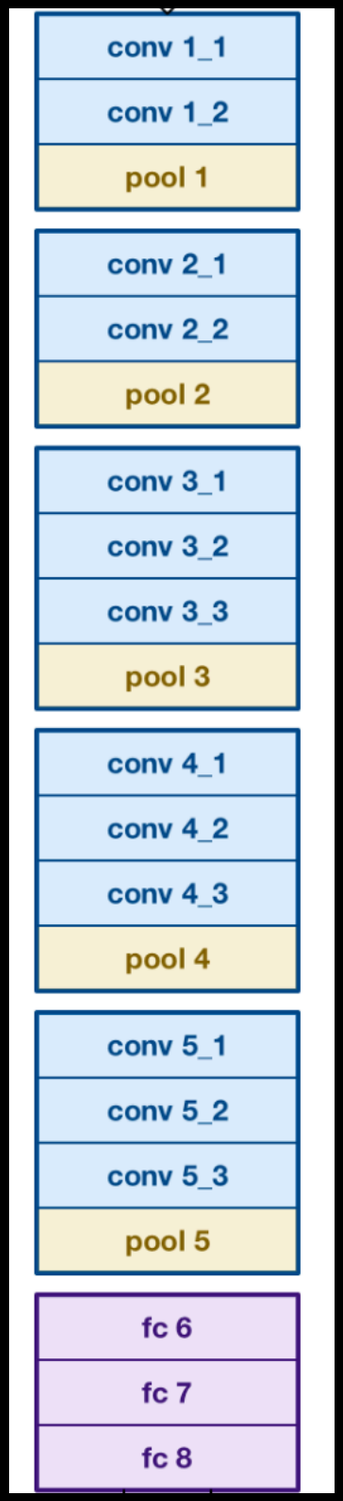
1) By employing visualization tools to help “understand” the inner working of AlexNet, shortcomings in AlexNet were discovered. (For example: Ineffective V1 filters were discovered in AlexNet.)
2) Using this understanding, the AlexNet filter sizes and strides were decreased, resulting in improved performance.

2014 - GoogLeNet/ Inception



1) Instead of using a fixed convolutional size architecture for each layer, this network permitted multiple convolutional sizes for a layer.
2) i.e. Parallel 1x1, 3x3 and 5x5 convolutions were performed in an inception module.
3) The results were then stacked, and a final output was computed.
4) This network made a very strong effort at reducing the number of weight parameters, through the use of 1x1 bottlenecks and the removal of the FC layers.

2014 - VGGNet

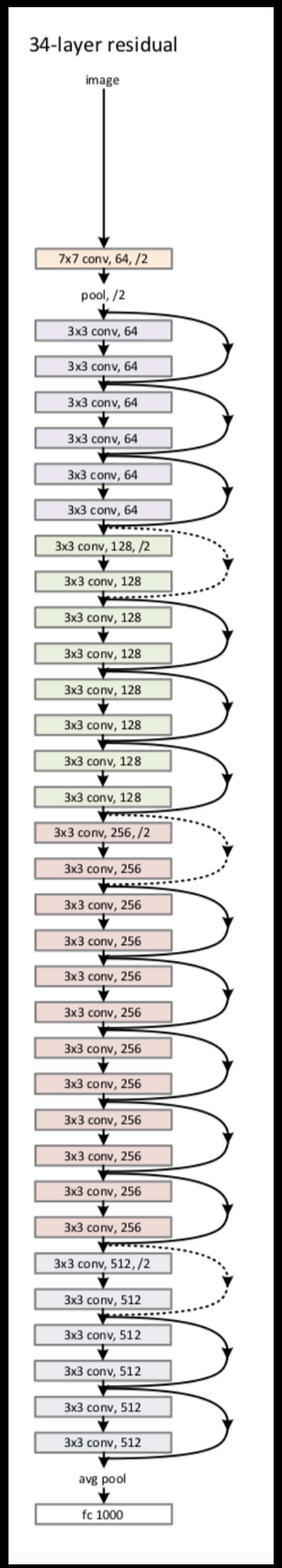


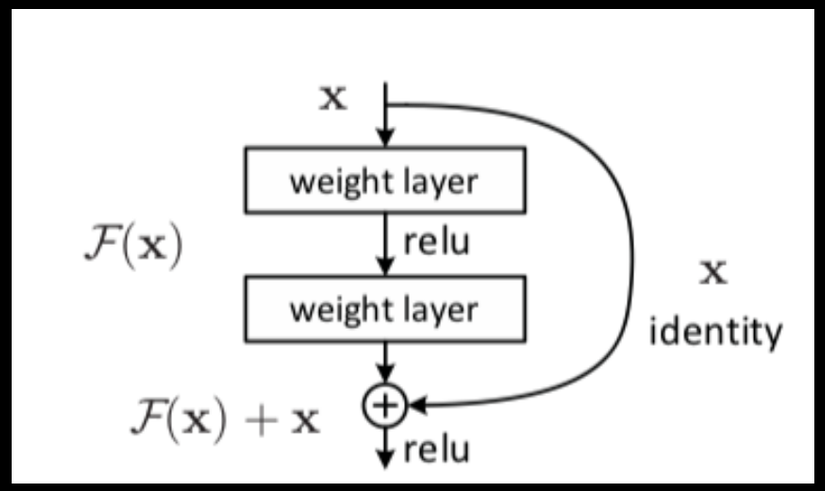
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128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

1) This network explored the notion of “going deep” by decreasing the spatial resolution (height and width) by 2x at each layer, while simultaneously increasing the depth by 2x.

2) A whopping 138 million weight parameters were used - very big at that time.

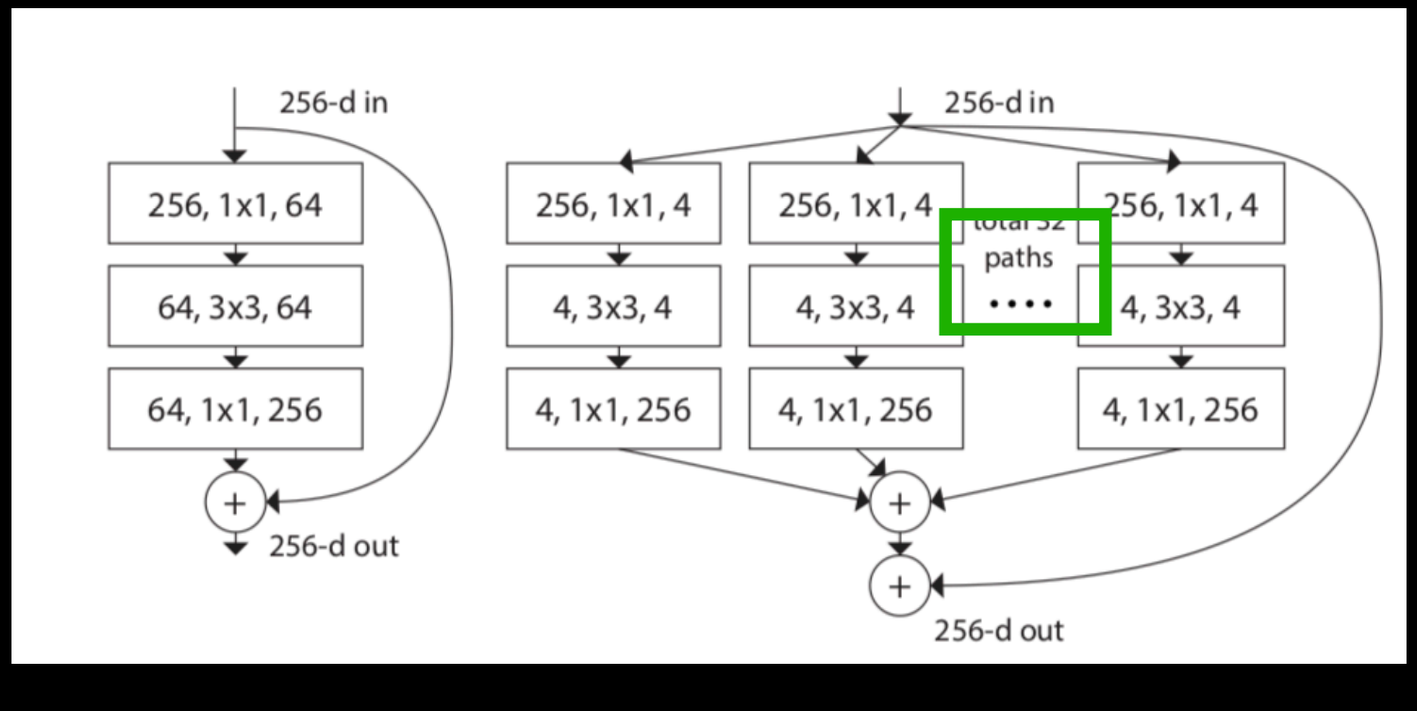
2015 - ResNet





1) This network went deeper than any previous architecture.
2) Initially, as the network was made deeper and deeper, performance degraded.
3) Ideally, if the added layers were short circuited though, the deeper networks should perform comparable to their original counterparts.
4) This led to the notion of skip connections.
5) Skip connections allowed successful end to end backpropagation, by addressing the “vanishing gradient” problem.

2016 - ResNext



Left: ResNet (Cardinality = 1), Right ResNext (Cardinality = 32)

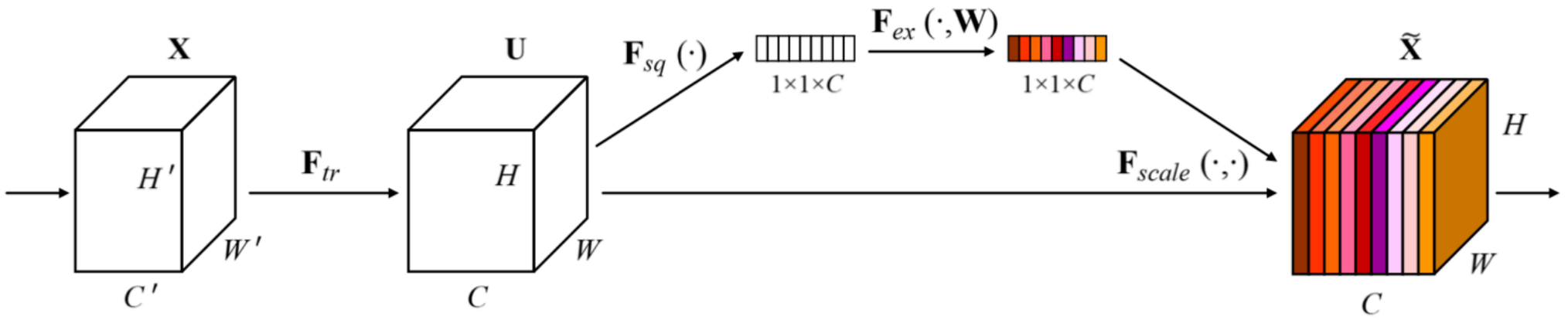
1) Like VGGNet and ResNet, ResNext stacked modules of the same topology.

2) However, ResNext modules had significantly larger “cardinality” (= paths) …

3) BUT possessed the same number of parameters as ResNet modules.

4) When the higher cardinality modules were stacked, improved classification accuracy was obtained.

2017 - SqueezeAndExiteNet

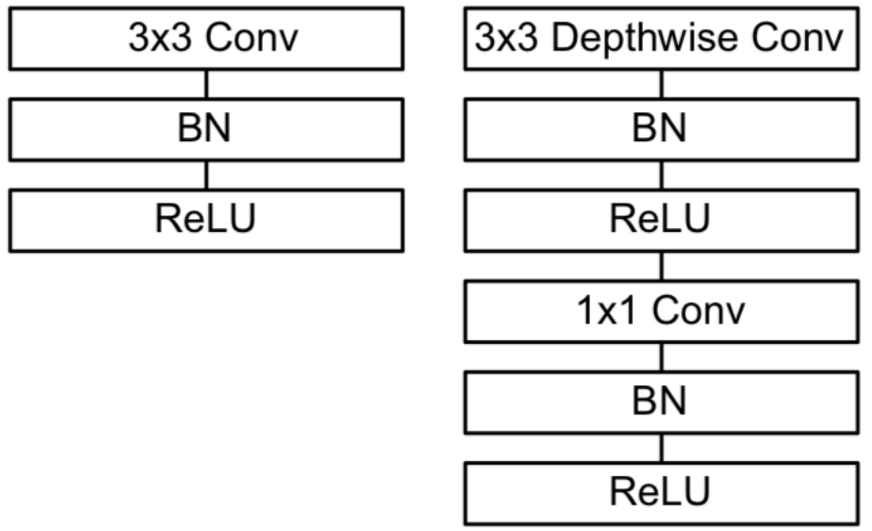


1) A new architectural unit known as the SqueezeAndExcite (SE) block was introduced.
2) This block examined the features in each channel of a CNN layer, and reweighed the different channels accordingly.
3) The SE block could be dropped into any of the previous CNN architectures and produce an immediate improvement in classification accuracy.

2017 - MobileNet

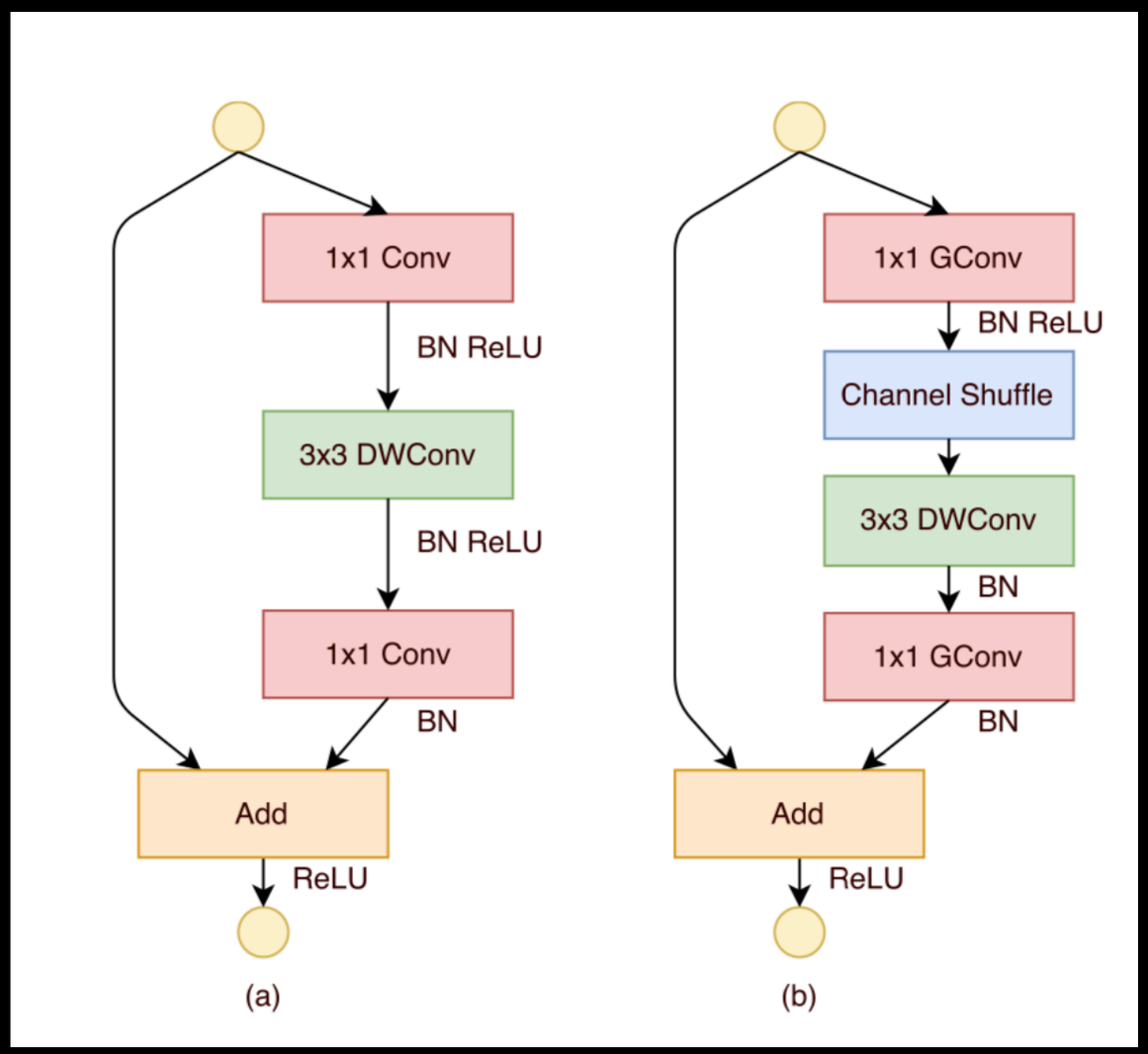
Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$



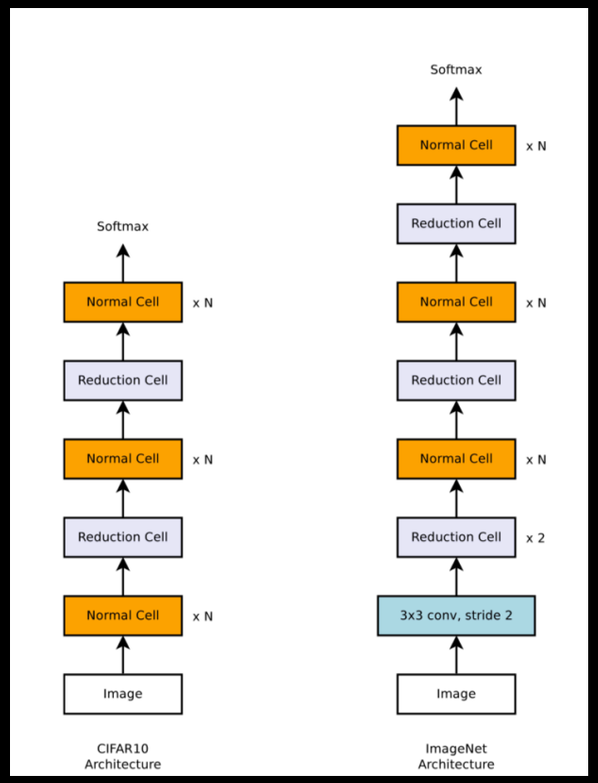
1) This network was designed primarily for mobile platforms.
2) It achieved significant reductions in computational cost by introducing the depth wise separable convolution.
3) i.e. Convolutions were performed in 2 steps.
4) First, a spatial convolution was performed.
5) Then, the channels were aggregated using 1x1xC channel convolutions.

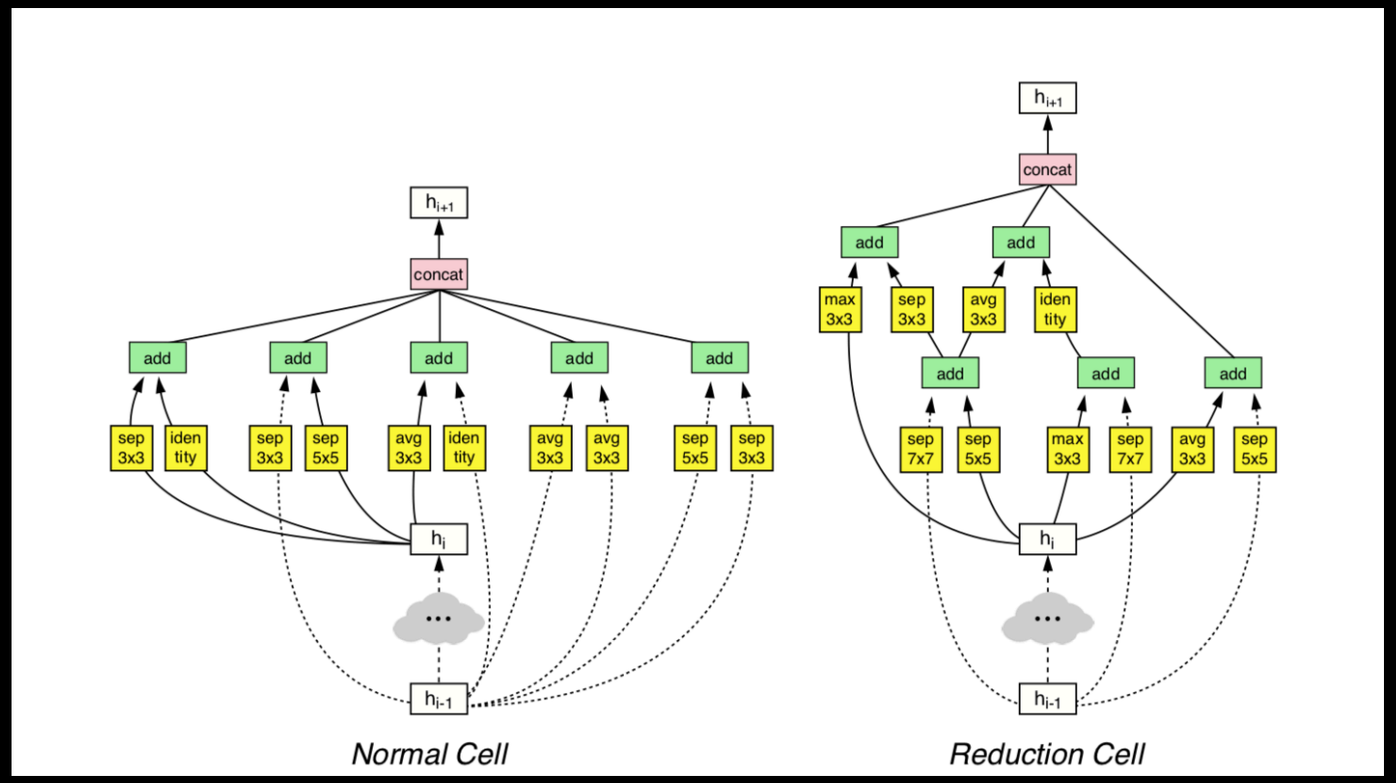
2017 - ShuffleNet



1) This network was also designed for mobile platforms.
2) It achieved significant reductions in computational cost by introducing the notion of channeling shuffling and group convolutions.

2018 - NASNet





1) The idea of using search (network architecture search = NAS) to find an optimal architecture was explored.
2) In order to build scalable architectures, it was deemed that two type of cells were needed - normal cells and reduction cells.
3) Normal cells return feature maps of the same size. Reduction cells returned feature maps half their original size.
4) Using this constraint, NAS was then performed using reinforcement learning.

2019 - AmoebaNet

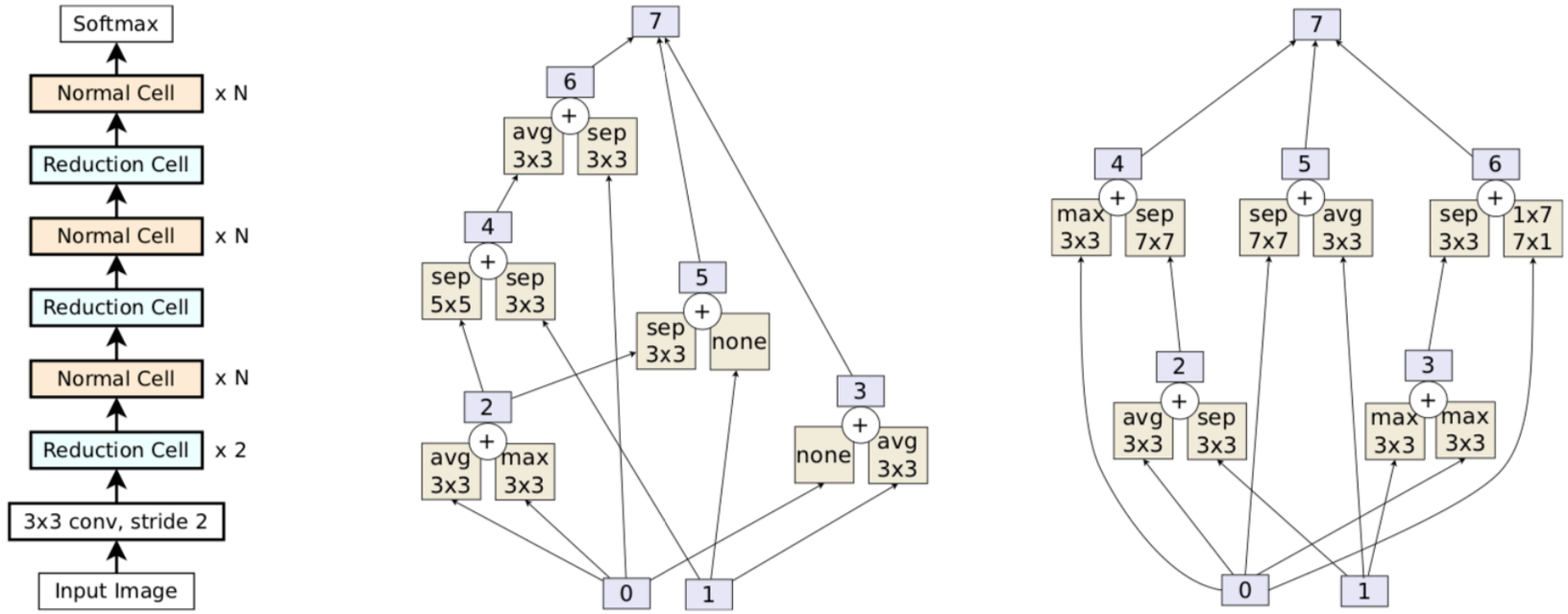


Figure 5: AmoebaNet-A architecture. The overall model [54] (LEFT) and the AmoebaNet-A normal cell (MIDDLE) and reduction cell (RIGHT).

1) The same ideas for NAS were employed.
2) Only now, search was performed using an evolutionary algorithm, instead of reinforcement learning.

2020 - EfficientNet

Table 1. **EfficientNet-B0 baseline network** – Each row describes a stage i with $\hat{L}_i$ layers, with input resolution $\langle \hat{H}_i, \hat{W}_i \rangle$ and output channels $\hat{C}_i$. Notations are adopted from equation 2.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

$$\text{depth: } d = \alpha^\phi$$
$$\text{width: } w = \beta^\phi$$
$$\text{resolution: } r = \gamma^\phi$$
$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$
$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

1) This paper describes an efficient way to scale up a CNN architecture.
2) A compound scaling model is introduced.
3) The model efficiently scaled the length (depth = layers), width (channels) and resolution (HxW) using an empirically derived formula.

Comments

- Tremendous progress has been made since 2012. As of 2020, Top 1% accuracy on ImageNet is now ~85%.

- To get there, existing networks have been 1) made deeper (= more layers), 2) have employed skip connections to enable deeper network training, 3) have exploited the weightings of different channels in the architecture, etc.

- However, one cannot discount the brittleness of existing networks.

- When applied to scenes containing objects with different viewpoints, or in the presence of background clutter, recognition performance using current state of the art methods degrades by “multiple double digits”.

- In addition, the plethora of work being done in the area of adversarial attacks, also suggest that something may be amiss.

Comments

- i.e. Do we have the right labeled dataset for training?
- ImageNet was a significant improvement over CIFAR. What will constitute a significant improvement over ImageNet?
- i.e. Is there a better architecture?
- Deeper and “more data” proved to be beneficial in transitioning from networks trained on CIFAR versus network trained on ImageNet.
- However, is this the right path to future results that are robust to viewpoint, background clutter and adversarial attacks?

Comments

- Visualizations (to be discussed next), could provide some interesting insights into these question.

- However, visualizations also introduce questions of their own.

References

- Neocognitron, Fukushima, Transactions IECE 1979
- Gradient Based Learning Applied to Document Recognition, LeCun, et. al., Proceedings IEEE 1998
- ImageNet Classification with Deep Convolutional Neural Networks, Krizhevsky, et. al., NIPS 2012
- Visualizing and Understanding Convolutional Networks, Zeiler, et. al., arXiv November 2013
- Going Deeper With Convolutions, Szegedy, et. al., arXiv September 2014
- Very Deep Convolutional Networks for Large Scale Image Recognition, Simonyan and Zisserman, arXiv September 2014
- Deep Residual Learning for Image Recognition, He, et. al., arXiv December 2015

References

- Aggregated Residual Transformations for Deep Neural Networks, Xie, et. al., arXiv November 2016

- Squeeze and Excitation Networks, Hu, et. al., arXiv September 2017 MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Howard, et. al., arXiv April 2017

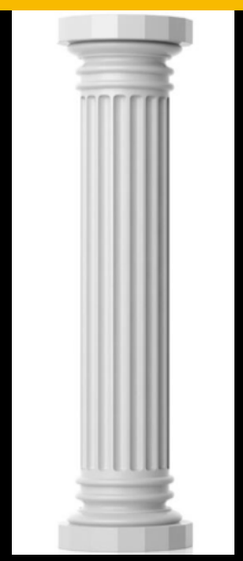
- ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices, Zhang, et. al., arXiv July 2017

- Learning Transferable Architectures for Scalable Image Recognition, Zoph, et. al., arXiv July 2017

- Regularized Evolution for Image Classifier Architecture Search, Real, et. al., arXiv February 2018

- EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, Tan, et. al., arXiv May 2019

Algorithm (Architecture Visualizations)



Algorithm (Architecture Visualizations)

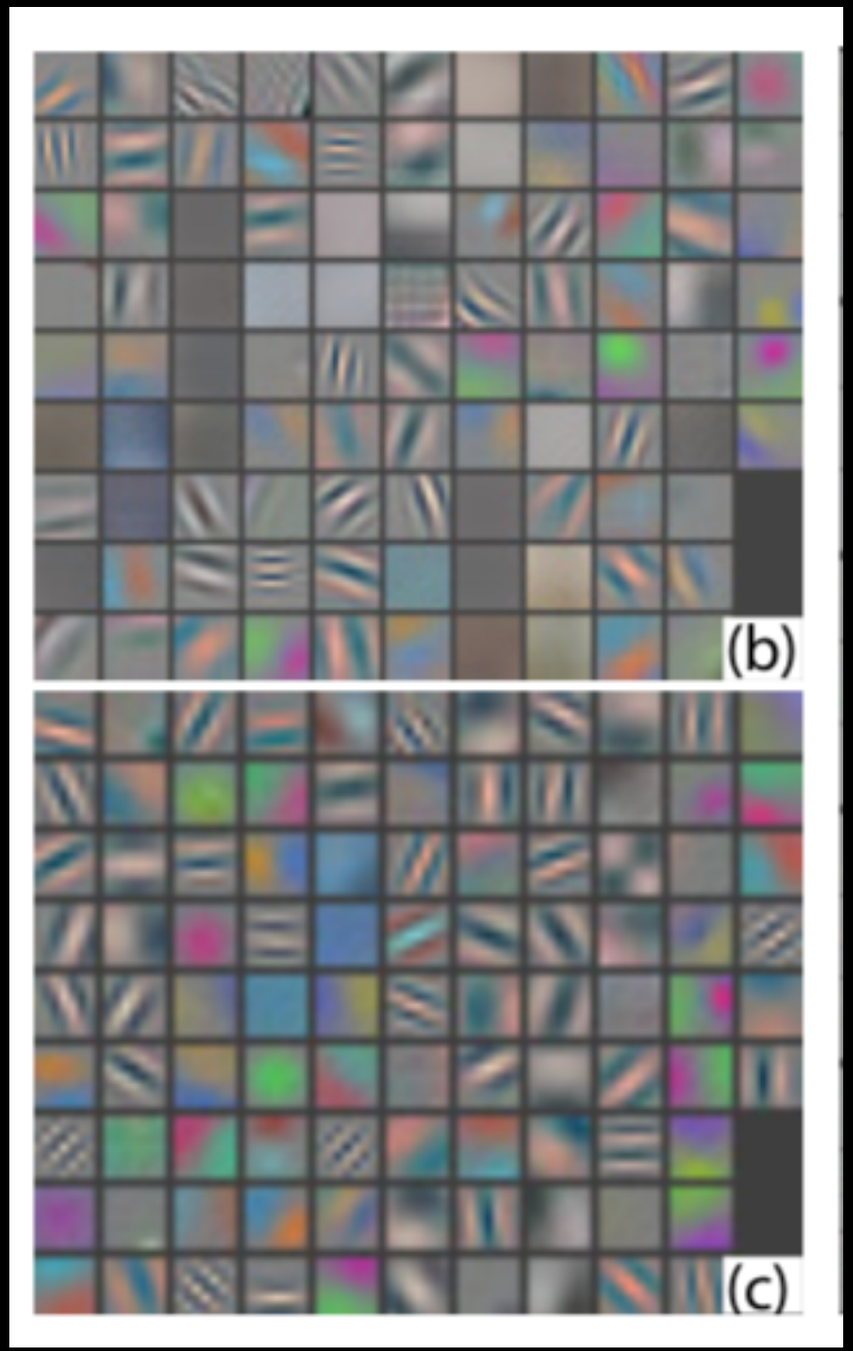
What did the different architectures learn in the various layers?

- 1) First Layer: Visualization via Low Level V1 filters
- 2) Last Layer: Visualization via Semantic reductions
- 3) Intermediate Layers: Visualization via Activations (Patches)
- 4) Intermediate Layers: Visualization via Gradient Ascent
- 5) Intermediate Layers: Visualization via Inverting Representations

First Layer - Low Level V1 Filters

- Not all first layers are created equal.
- Zwieler and Fergus demonstrated that improved recognition results could be obtained by improving the AlexNet V1 filters.
- More recently, in “Simulating a Primary Visual Cortex at the Front of CNN’s Improves Robustness to Image Perturbations”, (12/20) it was shown that robustness to adversarial attack could be improved, by employing more accurate V1 filters.

Learned V1 Filters

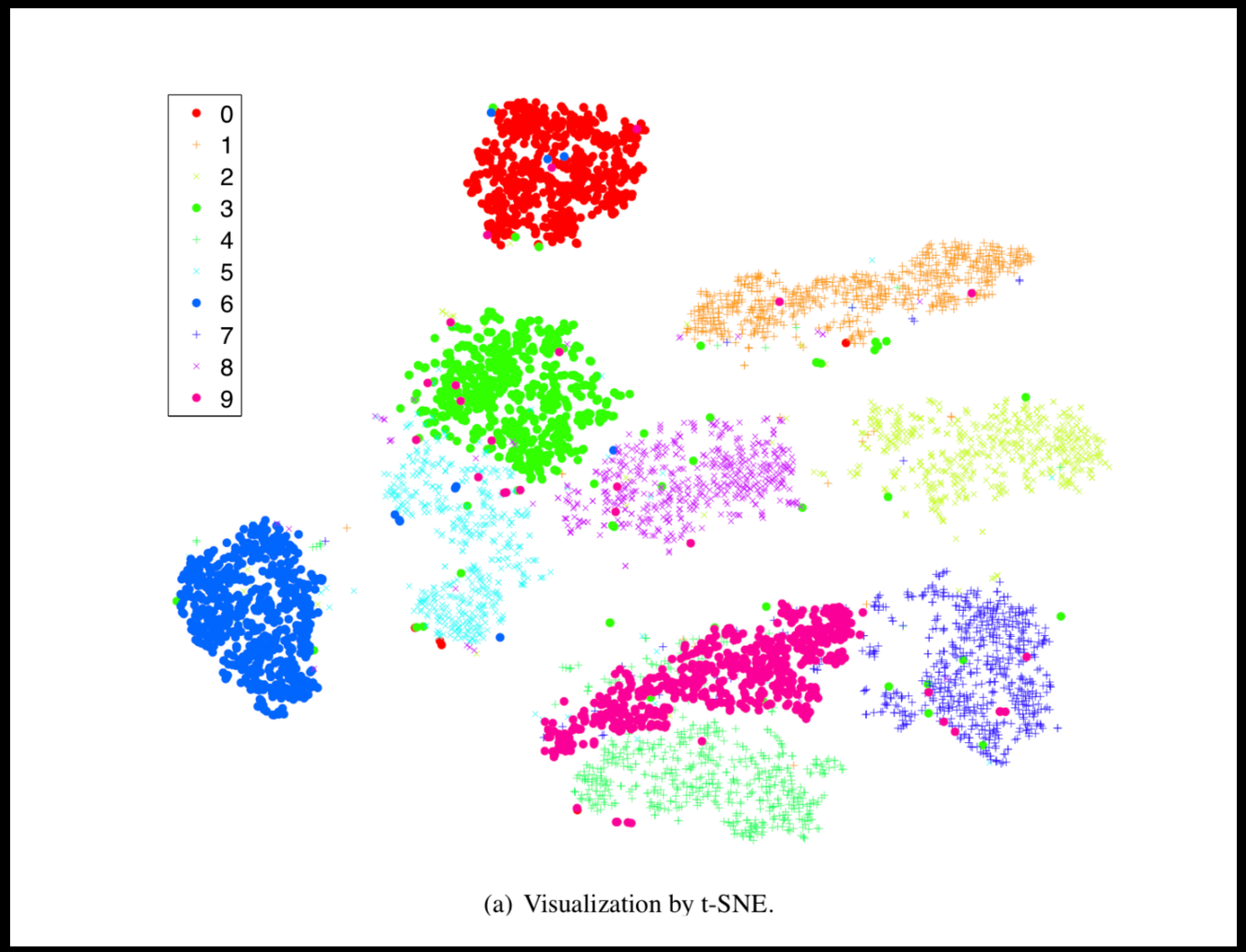


Top: AlexNet

Bottom: ZFNet

The filters learned by ZFNet were significantly better than those learned by AlexNet.

Last Layer - Semantic Reductions



(a) Visualization by t-SNE.

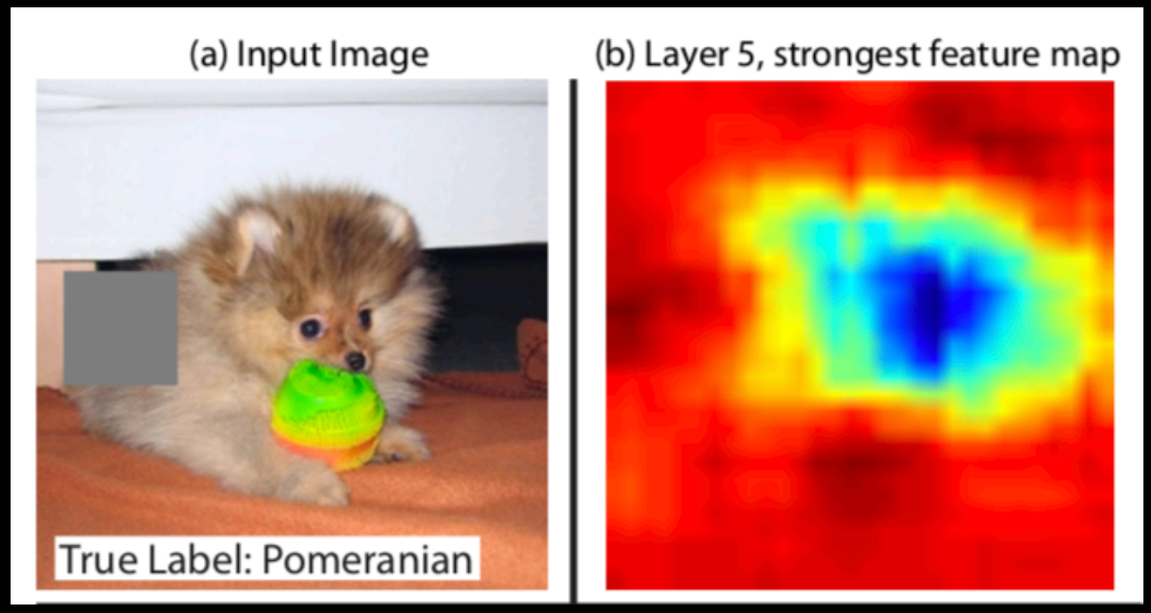
MNIST Digits

784 dimensional vector reduced to 30 dimensions using PCA.

30 dimensional vector is then reduced to 2 dimensions using t-SNE.

- By employing dimensionality reduction techniques like t-SNE to the output vector of a network, the network efficacy can be potentially probed.
- If the semantic reductions are meaningful, useful information can then be observed and extracted.

Intermediate Layers - Activations (Patches)



- Show the network various input image and find the activation maps in the network with the largest activations corresponding to the input.

- Then, ablate patches in the input image and observe the changes in the activation maps.

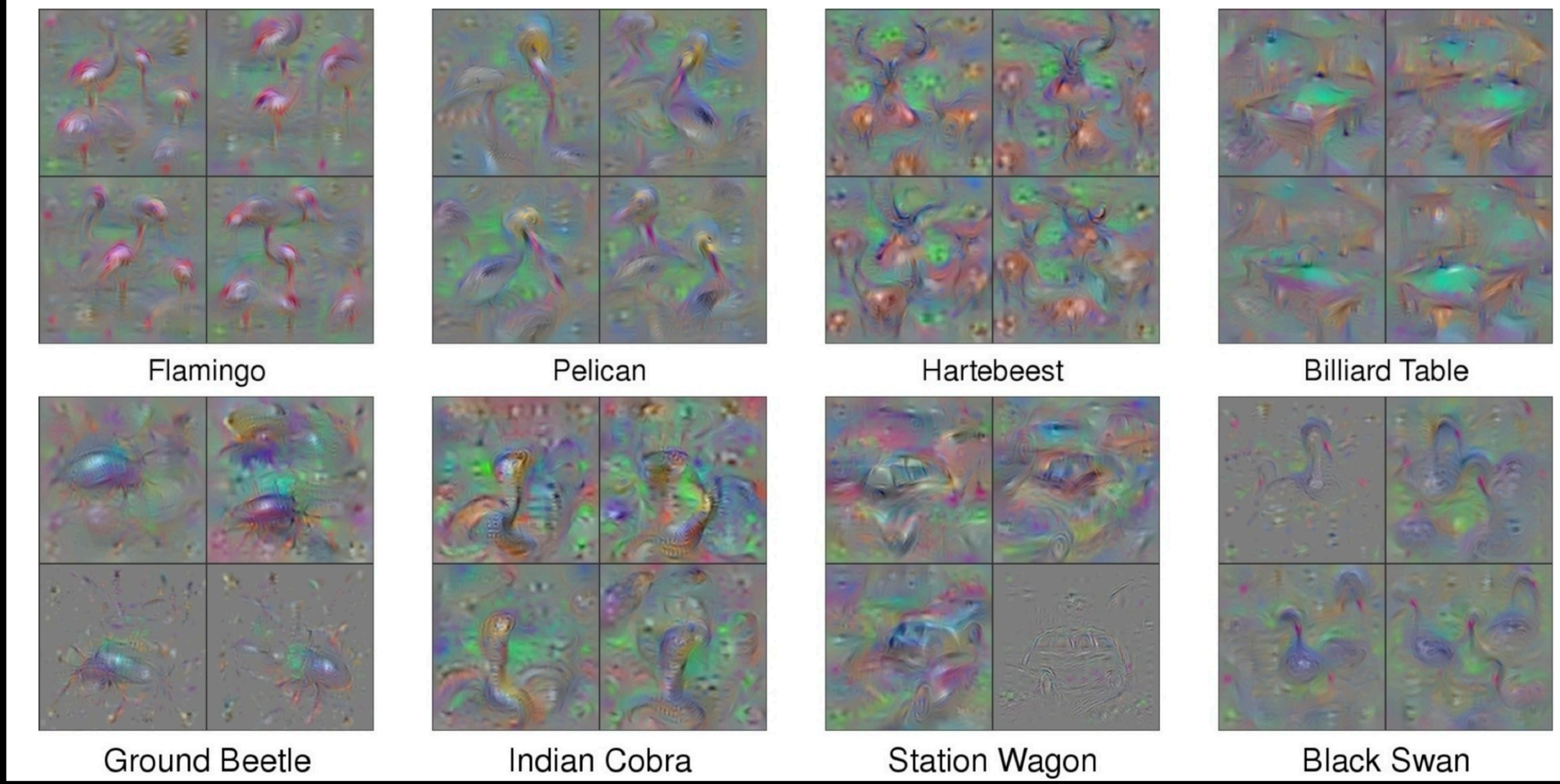
Intermediate Layer - Gradient Ascent

- Objective: Synthesize images that created high activations for specific neurons.
- i.e. Take the high activation maps, perform gradient ascent, and examine the newly created “input image” that maximized the activation map.
- Often, the optimization process created images that did not resemble natural images. Hence, not a lot of information could be gleaned / not a lot of conclusions could be reached.
- Better results were possible by using L2 regularization, or, natural image priors (the inverting representations technique).
- Better results were also possible if a two step process was taken, where updates were alternated between argmax(a_i(x)) and r_theta( ) for various r_theta.

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} (a_i(\mathbf{x}) - R_\theta(\mathbf{x}))$$

$$\mathbf{x} \leftarrow r_\theta \left(\mathbf{x} + \eta \frac{\partial a_i}{\partial \mathbf{x}} \right)$$

Intermediate Layers - Gradient Ascent



Results for 4 different r_theta terms corresponding to different regularizations.

Intermediate Layer - Inverting Representations

- This approach was another attempt to answer the question: What information was preserved by the network output?

- Since the inversion was a one to many operation (non-unique), many re-created input images x* were possible.

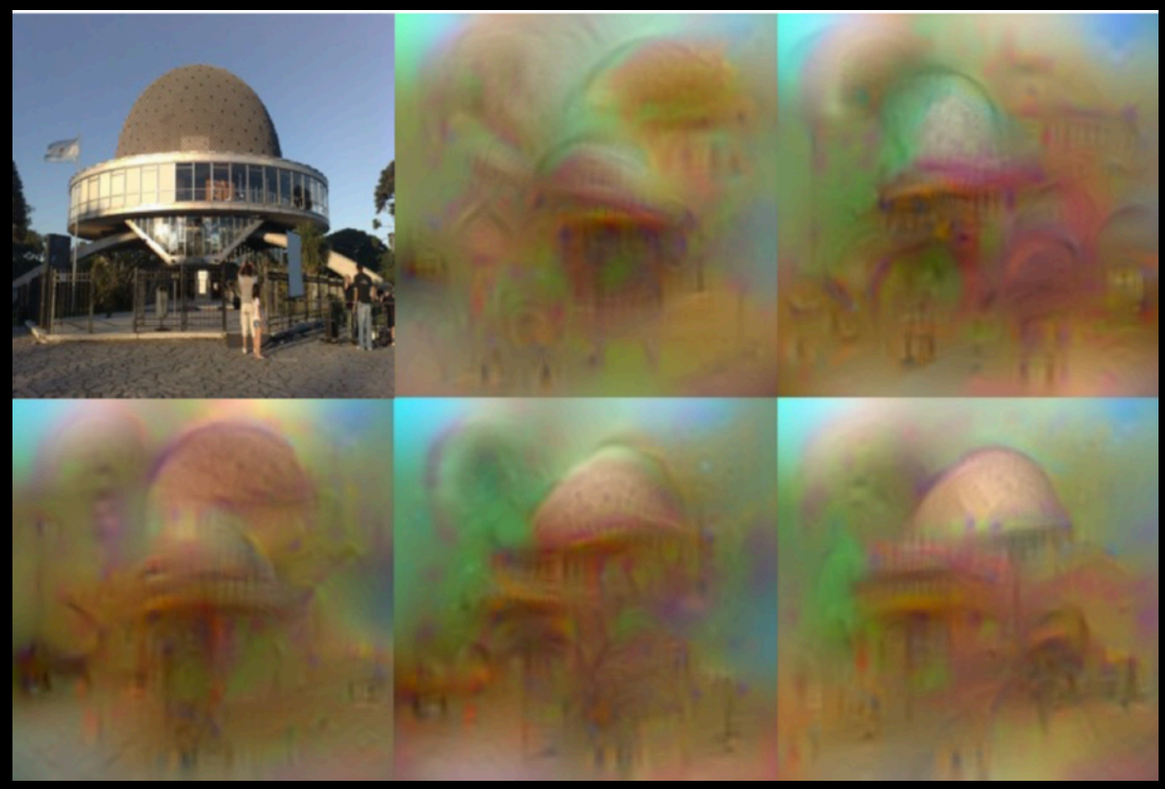
- For the specific implementation details, please refer to the source code.

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

$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$

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

Intermediate Layer - Inverting Representations



1) The upper left image generates a representation for a specific layer of a CNN.
2) The inversion formula is then applied, using the upper left image as a prior.
3) Five possible reconstructed images are shown.

Comments

- Visualization is a rich and valuable area for research.
- However, how effective are these visualizations?
  - When the visualizations result from disentangled information, the information is strong.
  - However, the understanding of how to “correctly disentangle” is still in its infancy.
  - In the meantime, you are trying to visualize and extract meaning from potentially “highly corrupted” features.

References

- Simulating a Primary Visual Cortex at the Front of CNN's Improves Robustness to Image Perturbations, Dapello, et. al., bioarXiv June 2020

- Receptive Fields, Binocular Interaction and Functional Architecture in a Cat's Visual Cortex, Hubel and Wiesel, The Journal of Physiology 1962

- Feature Visualization, Olah, et. al., Distill

- Visualizing and Understanding Convolutional Networks, Zeiler and Fergus, arXiv November 2013

- Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, Simonyan, et. al., arXiv December 2013

- Understanding Neural Networks Through Deep Visualization, Yosinski, et. al., arXiv June 2015

- Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned by each Neuron in Deep Neural Networks, Nguyen, et. al., arXiv February 2016

- Understanding Deep Image Representations by Inverting Them, Mahendran and Vedaldi, arXiv November 2014