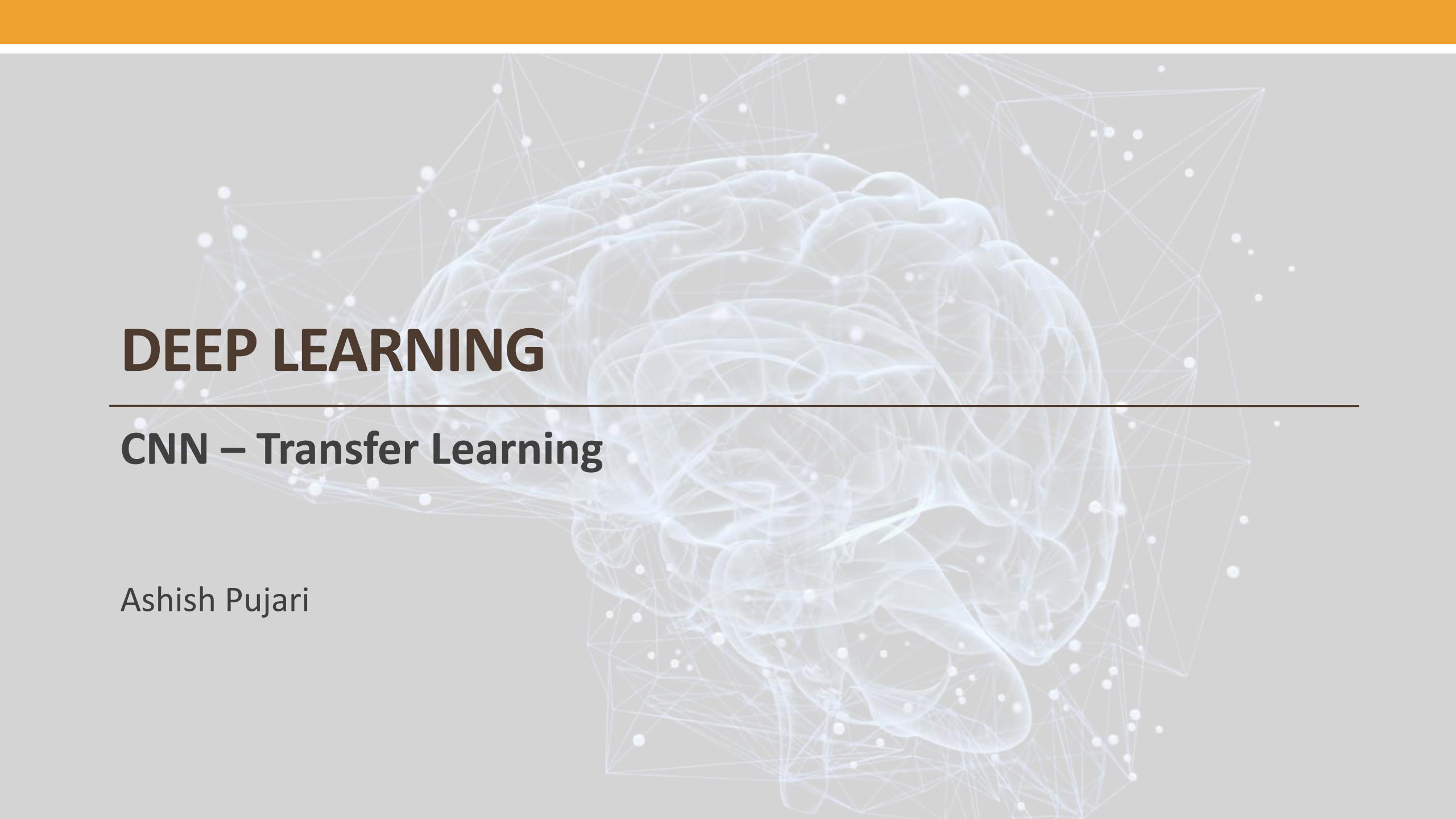


DEEP LEARNING

CNN – Transfer Learning

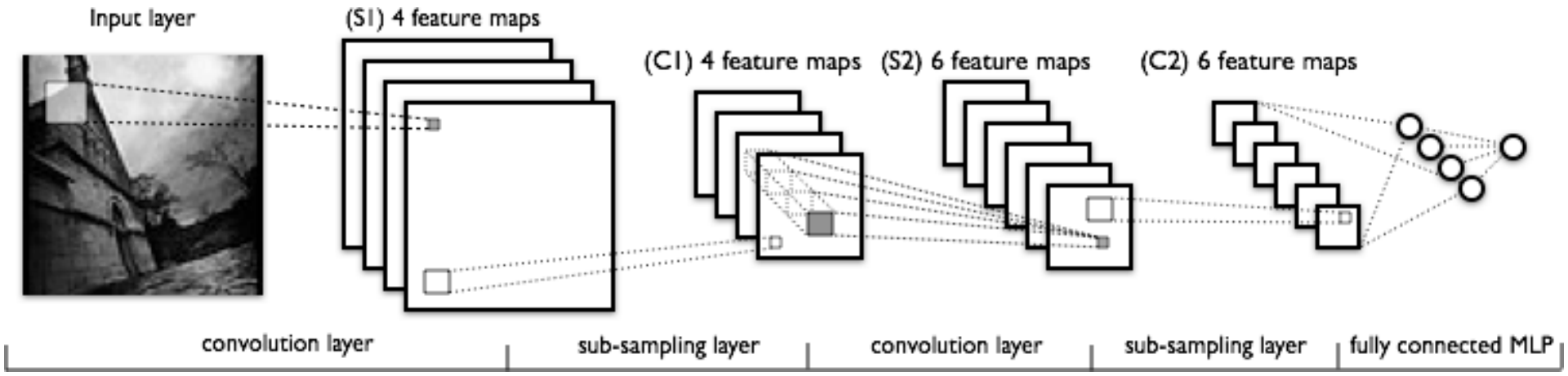
Ashish Pujari

Transfer Learning

1. CNN Architectures
   - LeNet, AlexNet, VGG
   - GoogLeNet, ResNet, DenseNet
   - EfficientNet
2. Transfer Learning

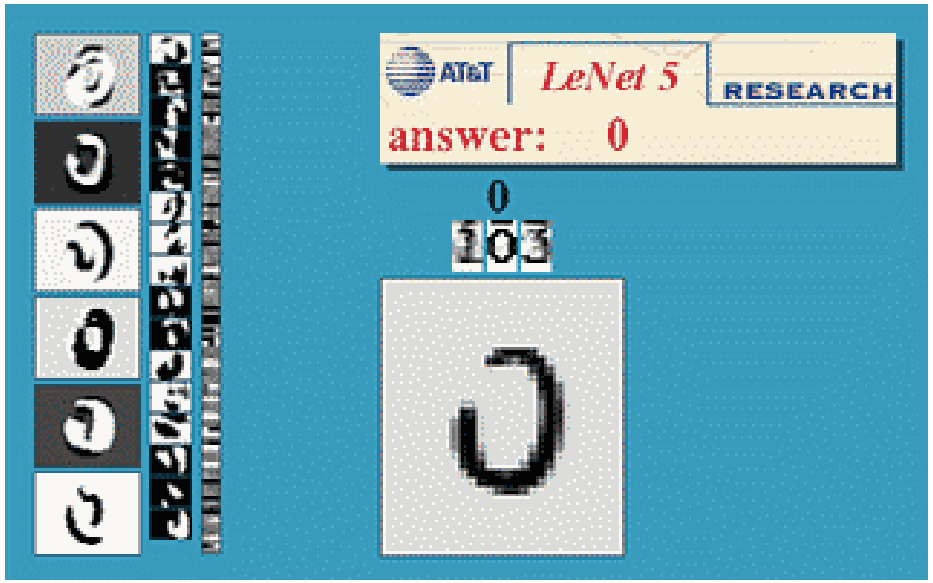
LeNet (1998)

- Handwritten and machine-printed character recognition [LeCun et al., 1998]
- Displayed Invariance to scale, translation, stroke width



http://yann.lecun.com/exdb/lenet/

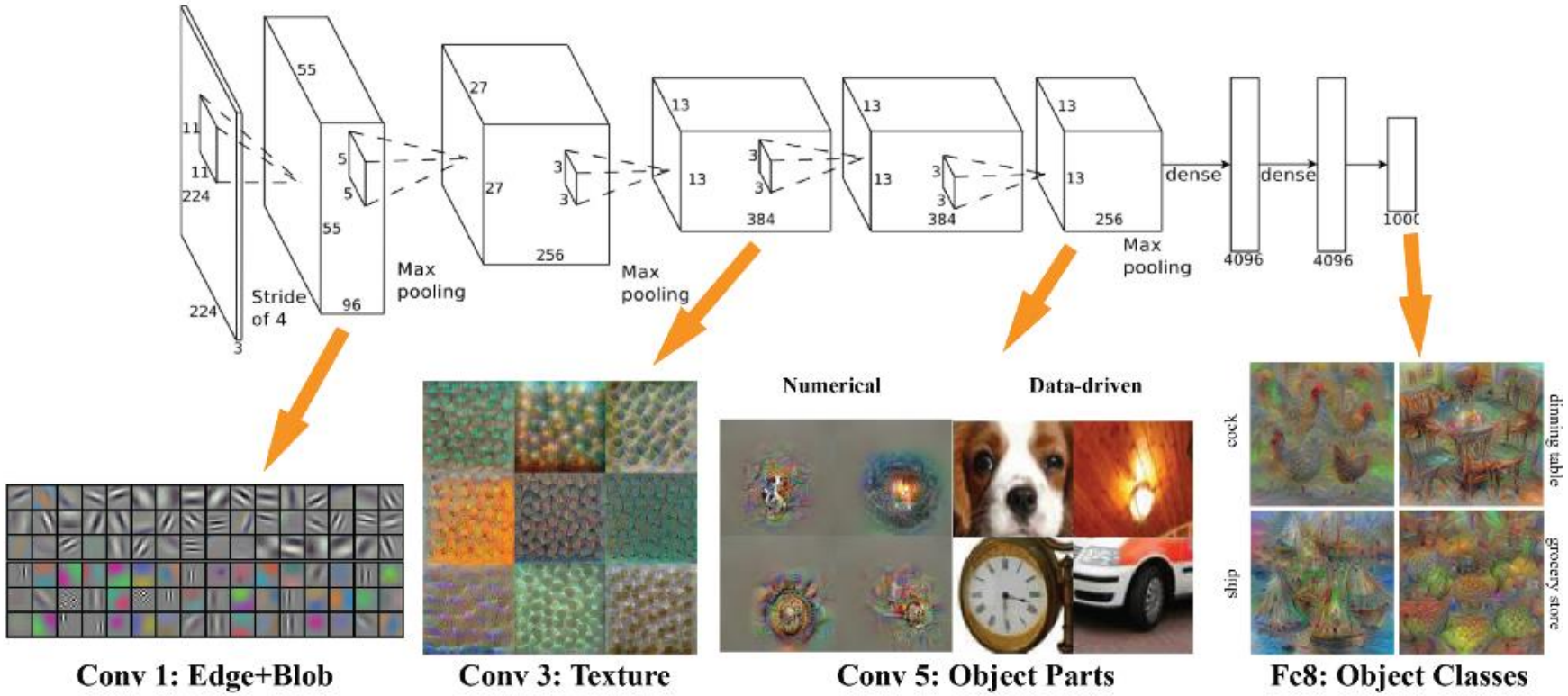
LeNet (1998)



http://yann.lecun.com/exdb/lenet/

Alexnet (2012)

- Won the IMAGENET Challenge 2012 - first time that Deep Learning was used for solving this problem
- 60 million parameters , 500,000 neurons, 5 convolution layers, 1000-way softmax

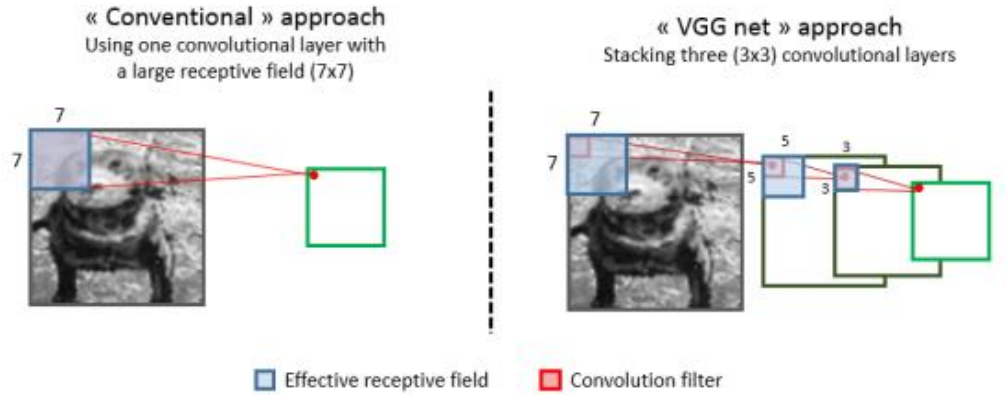


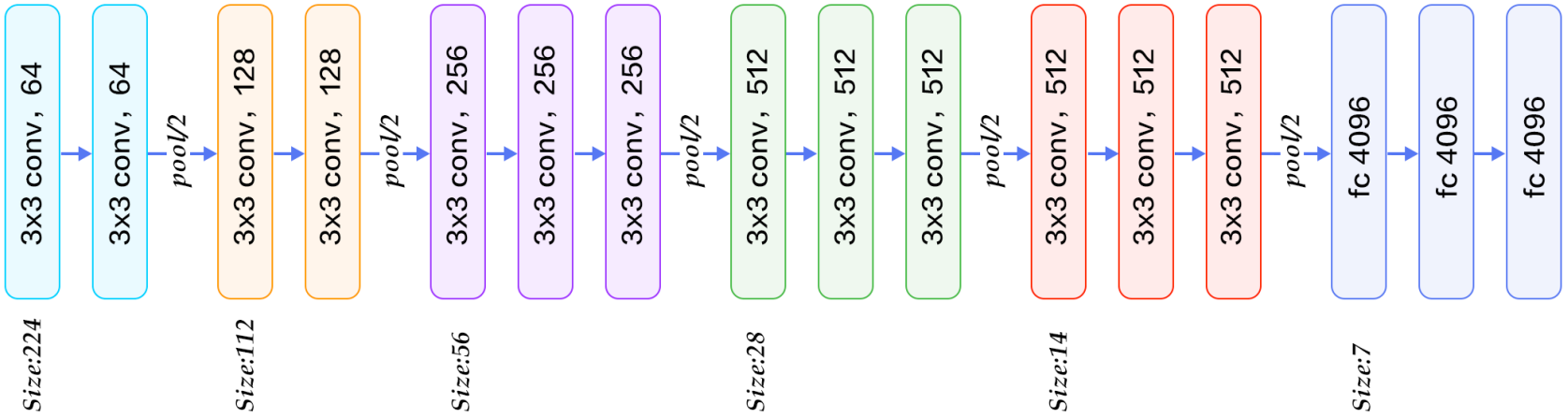
A visual representation of a convolutional neural net from the mNeuron plugin created for MIT's computer vision courses/teams.

http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

VGG (2014)

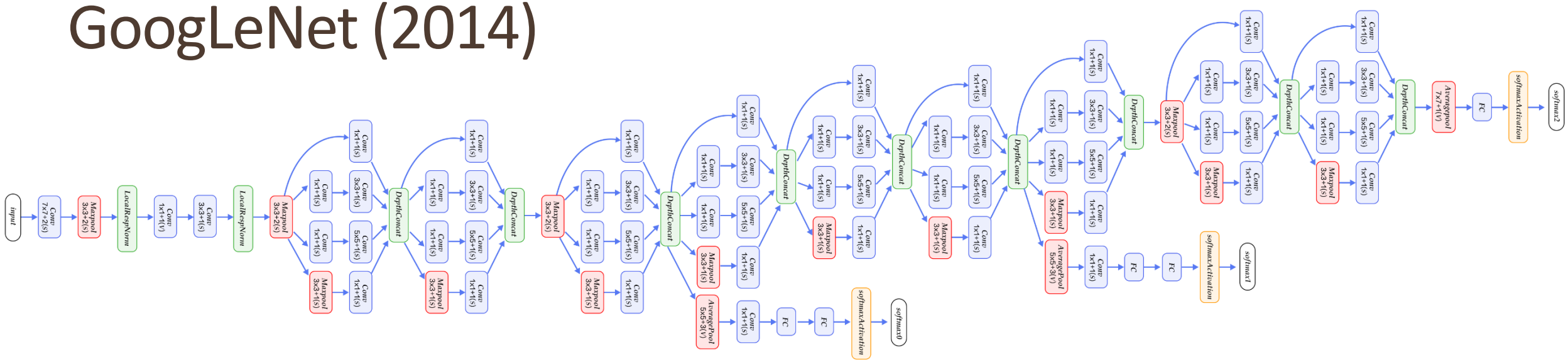
- Image classification network - Visual Geometry Group - University of Oxford
- Improved AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layers, respectively) with multiple stacked 3X3 kernel-sized filters one after another
- 16-layer network used by the VGG team in the ILSVRC-2014 competition





http://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

GoogLeNet (2014)



- GoogLeNet devised the *inception module* that approximates a sparse CNN with a normal dense construction (improvement over VGG)
- GoogleNet applies incredible number of convolutions in between and sometimes in parallel with maxpooling layers
- Won the IMAGENET Challenge 2014

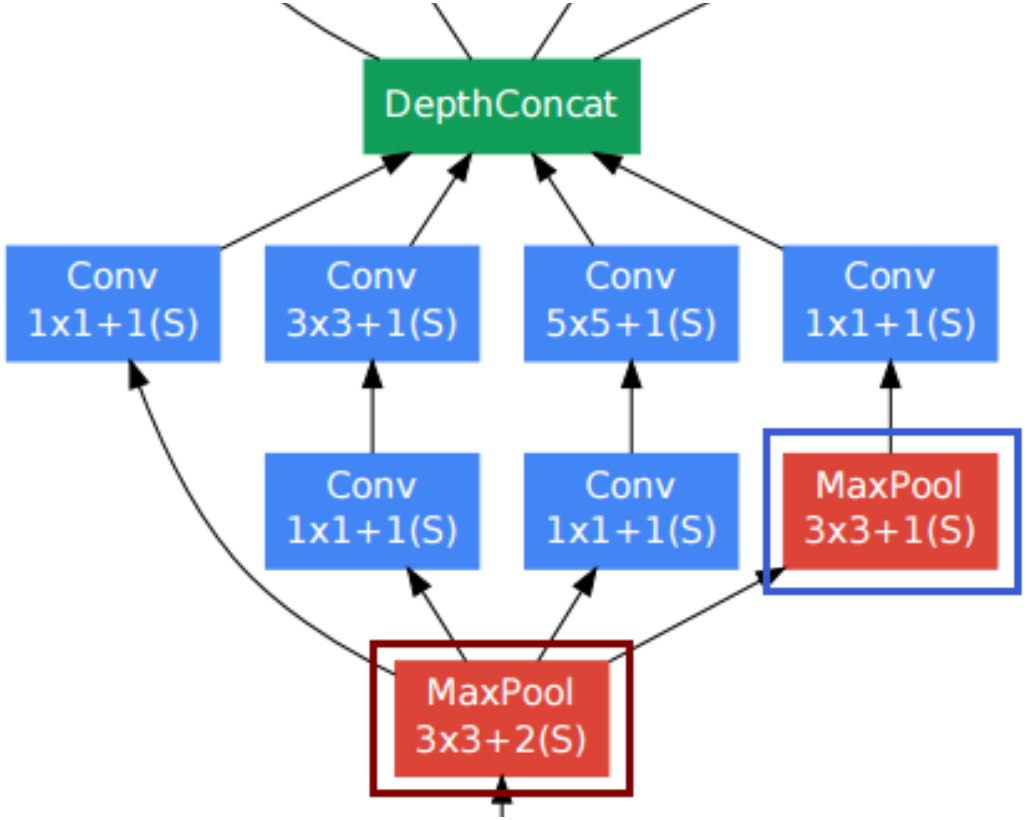
type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Table 1: GoogLeNet incarnation of the Inception architecture

http://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/
http://googleresearch.blogspot.co.uk/2015/06/inceptionism-going-deeper-into-neural.html

GoogLeNet - Inception Module

- Multiple convolution filters are processed on the same input
- This allows the CNN to take advantage of multi-level feature extraction from each input. For instance, it extracts general (5x5) and local (1x1) features at the same time
- Shown to improve network performance



https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf

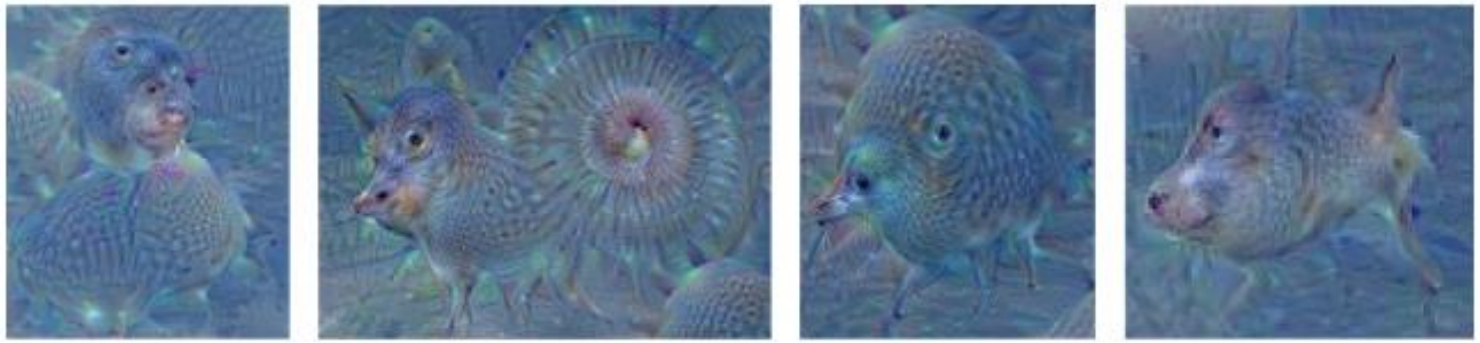
Feature Visualization

Feature visualization allows us to see how GoogLeNet, trained on the ImageNet dataset, builds up its understanding of images over many layers. CNNs learn to detect edges, then shapes, and finally actual objects.

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

Inceptionism/Deepdream

- Neural networks trained to classify images have quite a bit of the information needed to generate images
- Amplify learning in a layer and feed it back to the network
- Understanding learning in hidden layers, debugging art – remix visual concepts.



http://googleresearch.blogspot.co.uk/2015/06/inceptionism-going-deeper-into-neural.html

Increasing CNN layers

- Initially, CNNs showed better performance with increase in number of layers
- However deep networks are hard to train because of the vanishing gradient problem

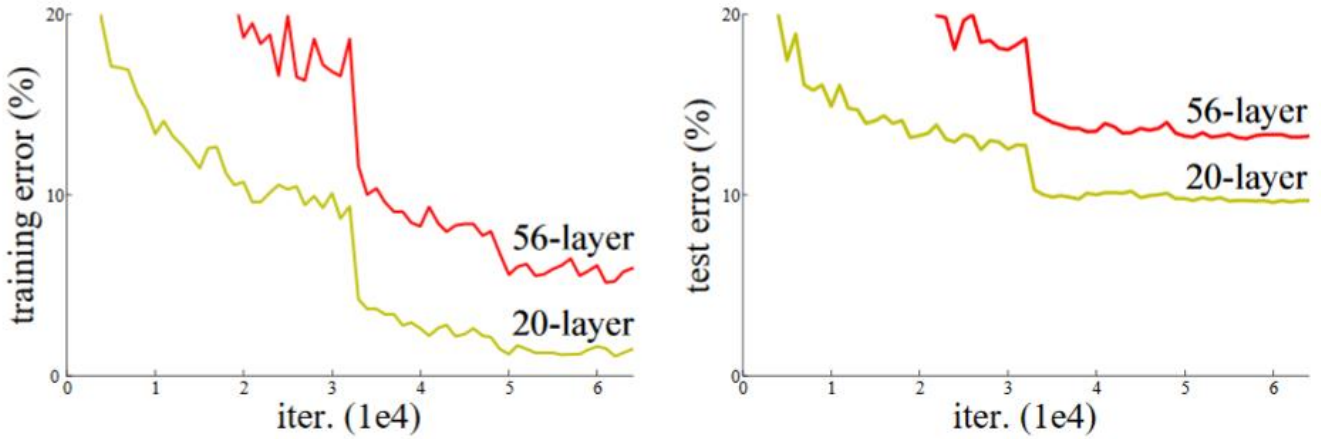
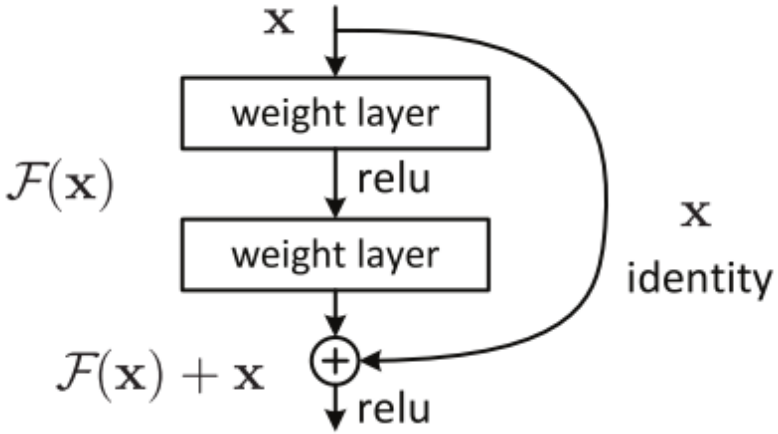


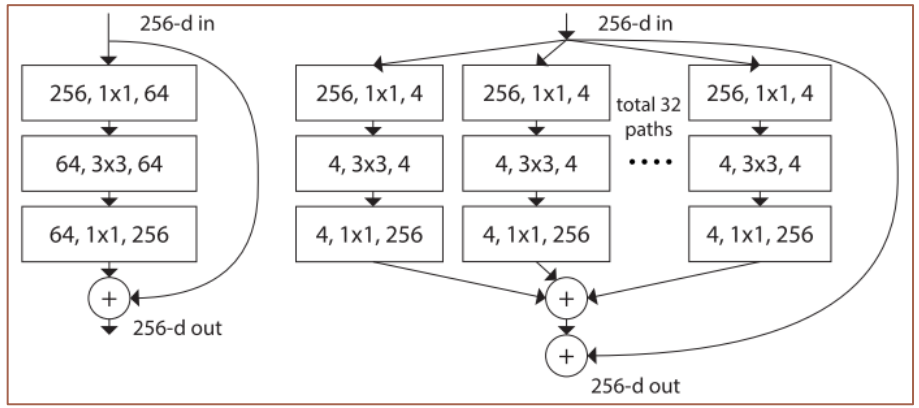
Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error.

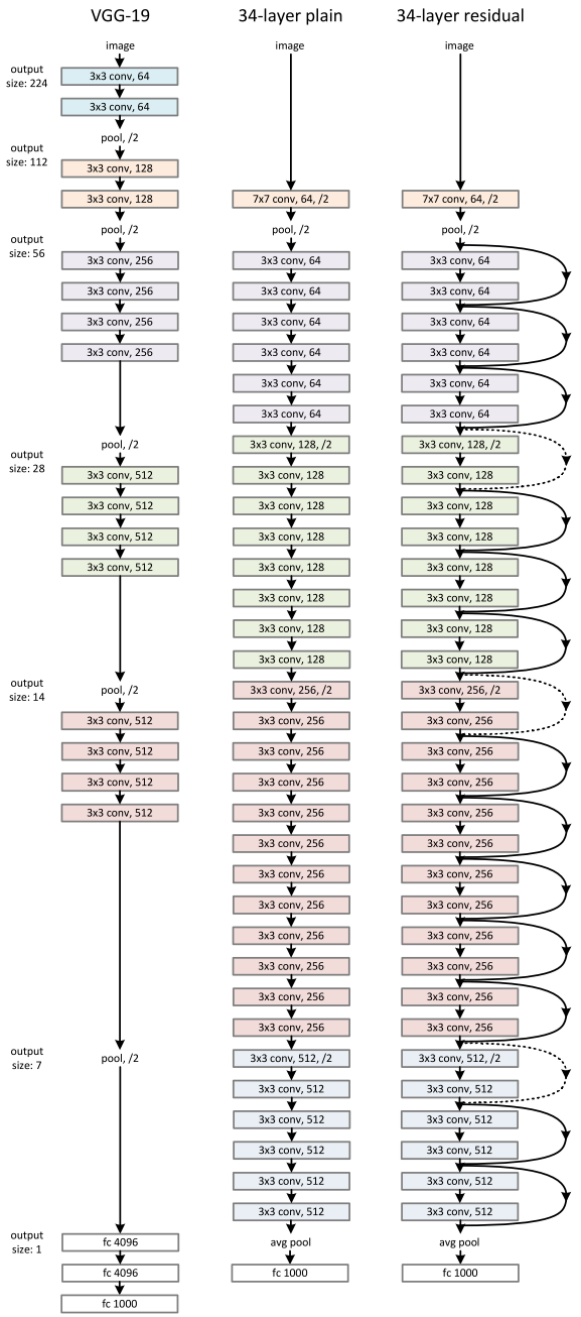
https://arxiv.org/pdf/1512.03385.pdf

ResNet (2015)

- ResNet makes it possible to train up to hundreds or even thousands of layers with high performance
- Main idea : “identity shortcut connection” that skips one or more layers; like Inception follows split-transform-merge paradigm
- Hypothesis: it is easy to optimize the residual mapping function $F(x)$ than to optimize the original, unreferenced mapping $H(x)$

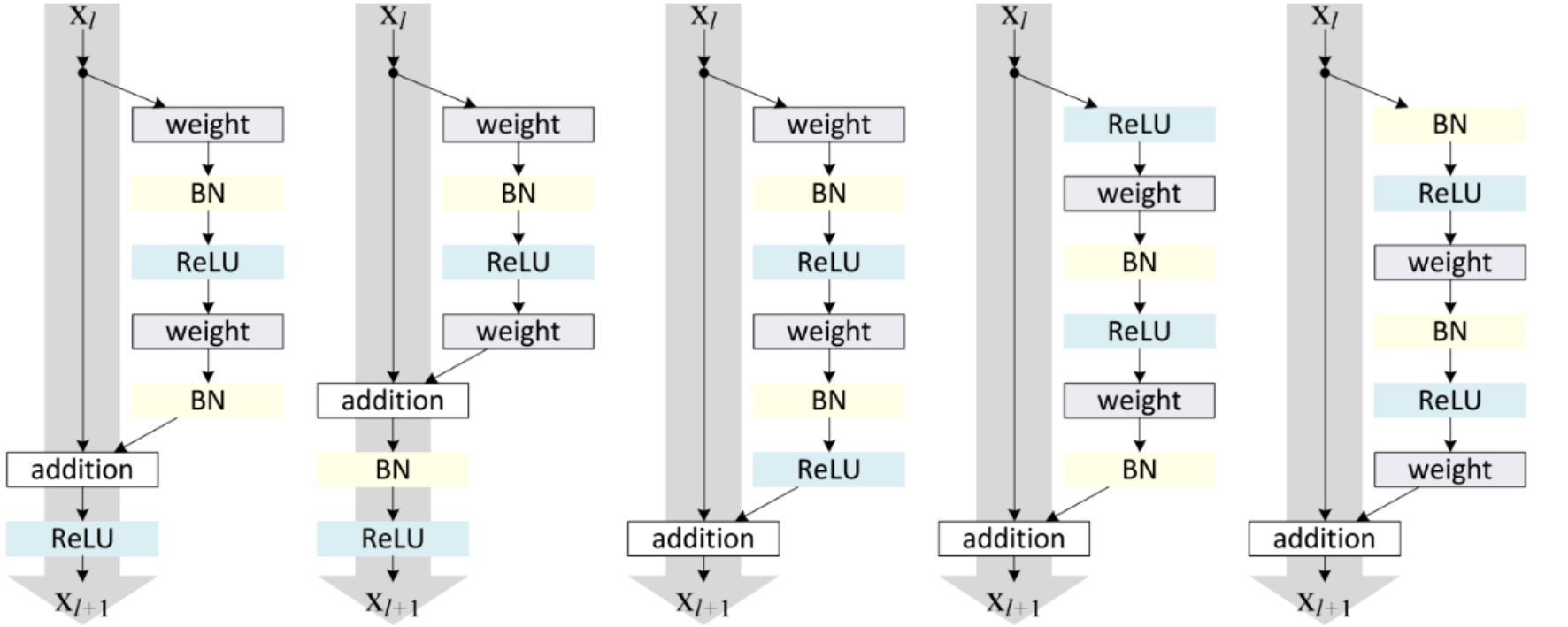






https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035

Residual Block Variants

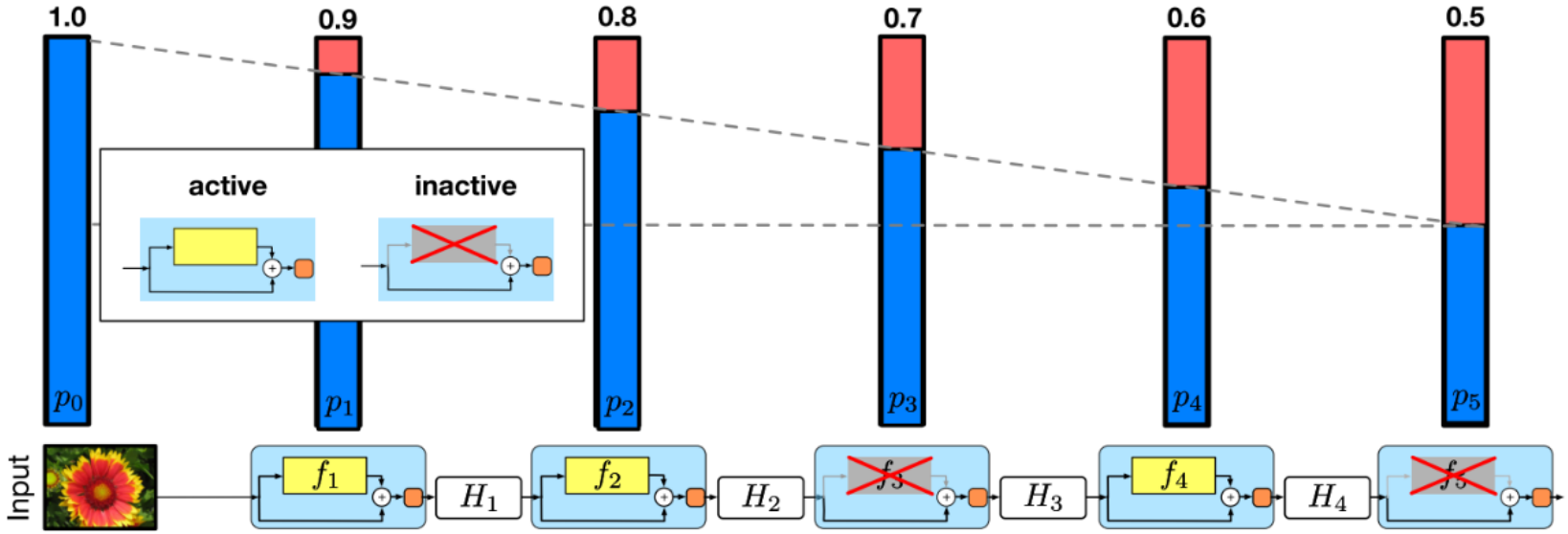


variants of residual blocks

https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035

ResNet with Stochastic Depth

- Similar to Dropout, a deep network with stochastic depth is like training an ensemble of many smaller ResNets.
- This method randomly drops an entire layer while Dropout only drops part of the hidden units in one layer during training.

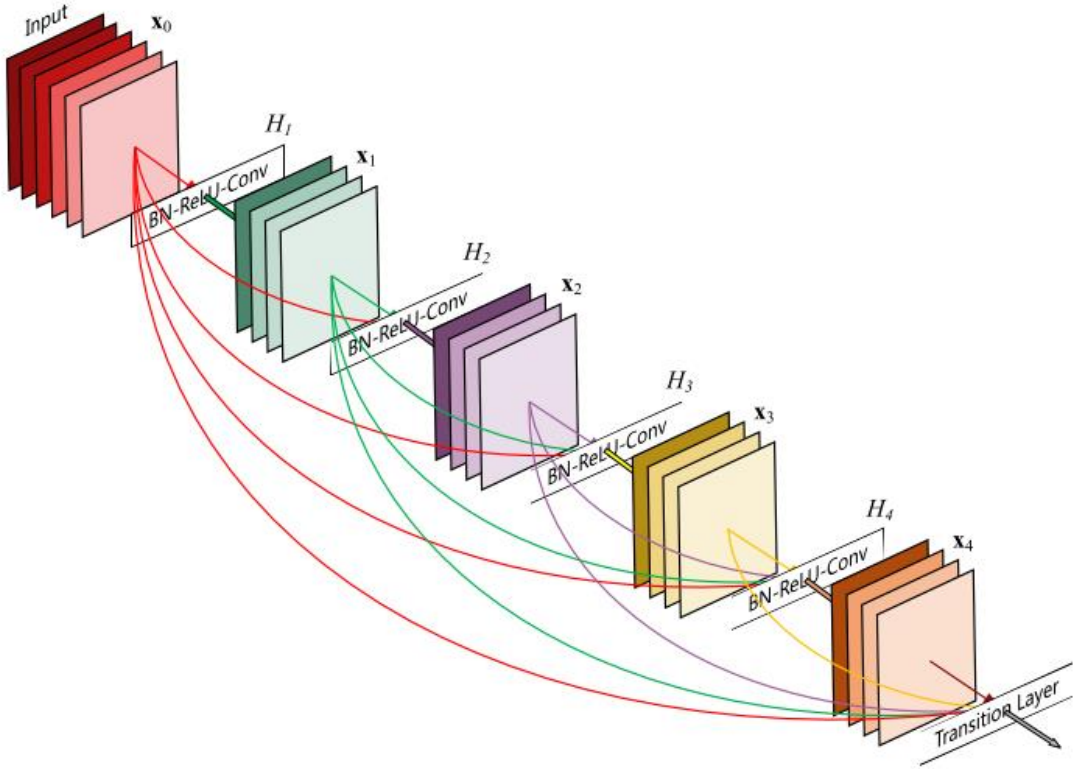


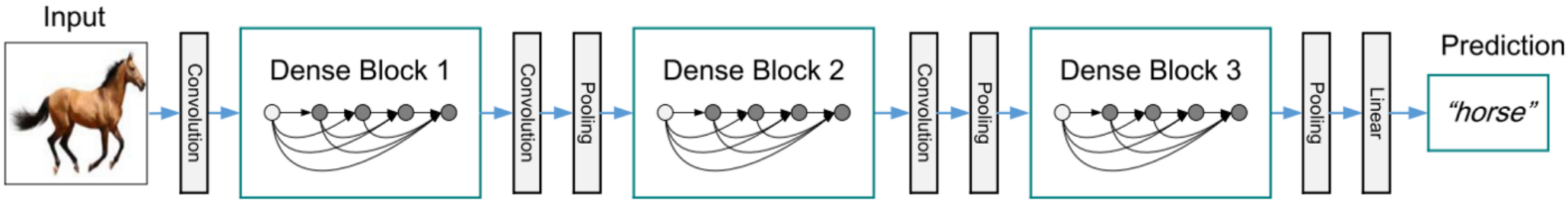
during training, each layer has a probability of being disabled

https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035

DenseNet (2016)

- DenseNet further exploits the effects of shortcut connections — it connects all layers directly with each other.
- The input of each layer consists of the feature maps of all earlier layers, and its output is passed to each subsequent layer.
- "*Other than tackling the vanishing gradients problem, this architecture also encourages feature reuse, making the network highly parameter-efficient*"



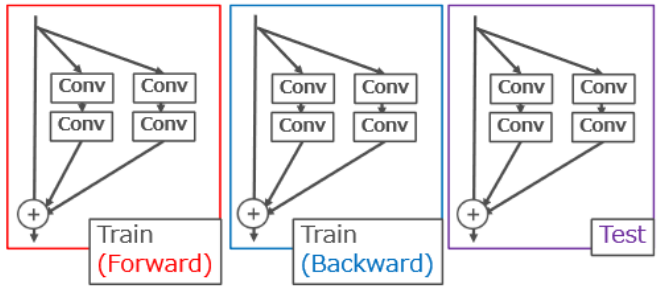


https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035

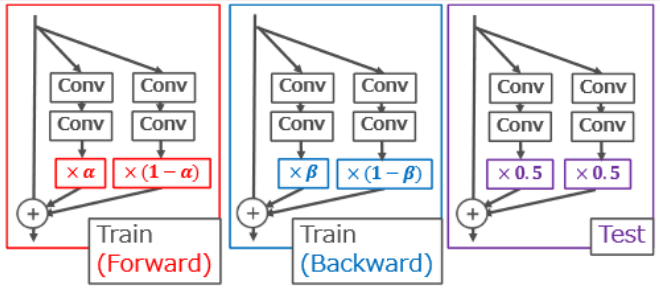
https://arxiv.org/abs/1608.06993

ResNet Regularization

Method	Description	Function
Stochastic Depth	Overcame problems of ResNet such as vanishing gradients. It makes the network apparently shallow in learning by dropping residual blocks stochastically selected X and G(x) are the input and output of the residual block, respectively, and F(x) is the output of the residual branch on the residual block	On the l^{th} residual block from input layer the Stochastic Depth process is given as $G(x) = x + b_l F(x),$ Where $b_l = \epsilon \{0,1\}$ is a Bernoulli random variable with the probability of p_l. The linear decay rule which defines p_l as $p_l = 1 - \frac{l}{L}(1 - p_L),$
Shake-Shake	Powerful regularization method for improving ResNeXt architectures	$G(x) = x + \alpha F_1(x) + (1 - \alpha) F_2(x),$



(a) ResNeXt (Xie et al., 2017), in which some processing layers omitted for conciseness.

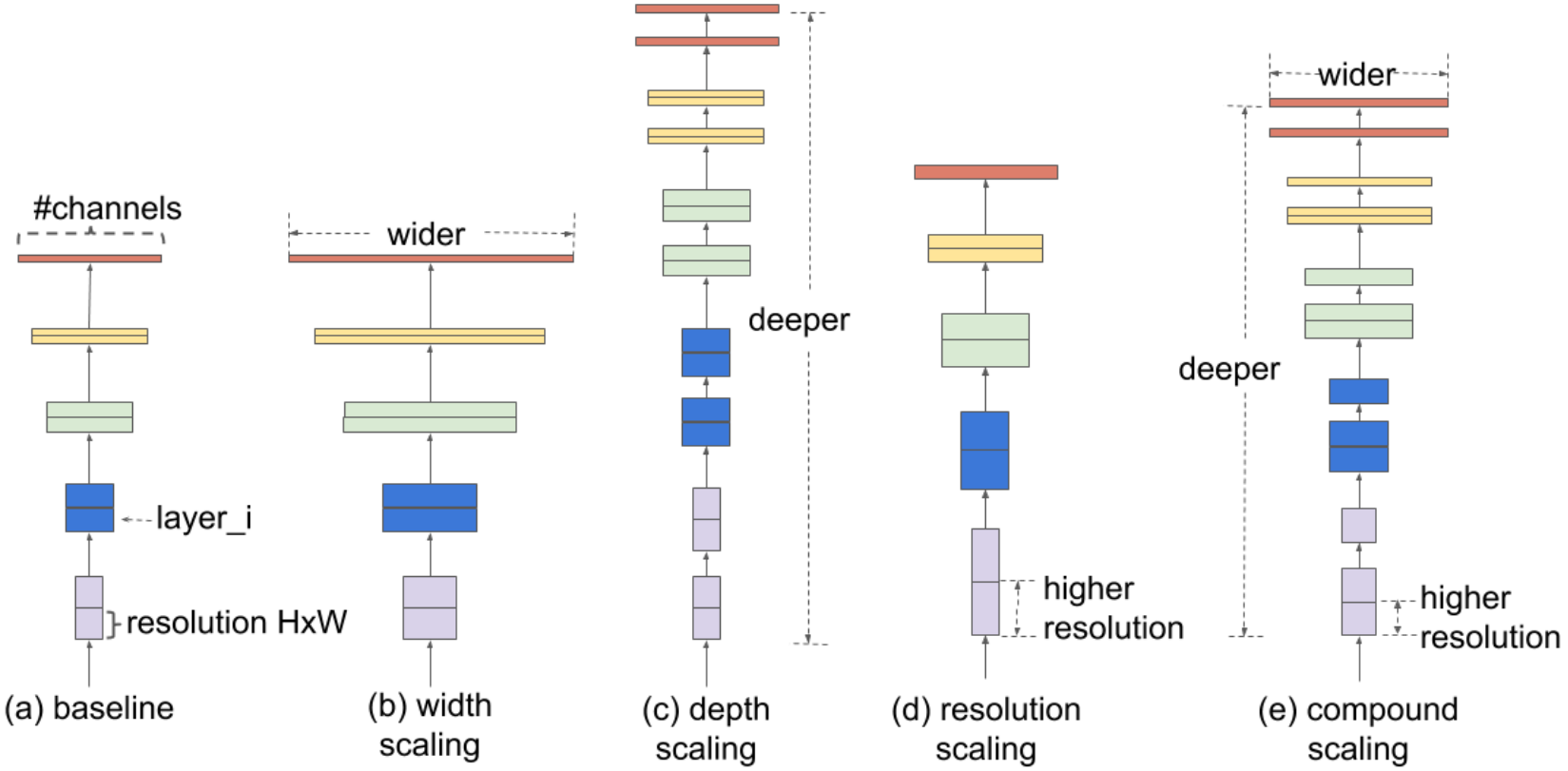


(b) Shake-Shake (ResNeXt + Shake-Shake) (Gastaldi, 2017), in which some processing layers omitted for conciseness.

https://openreview.net/pdf?id=S1NHaMW0b

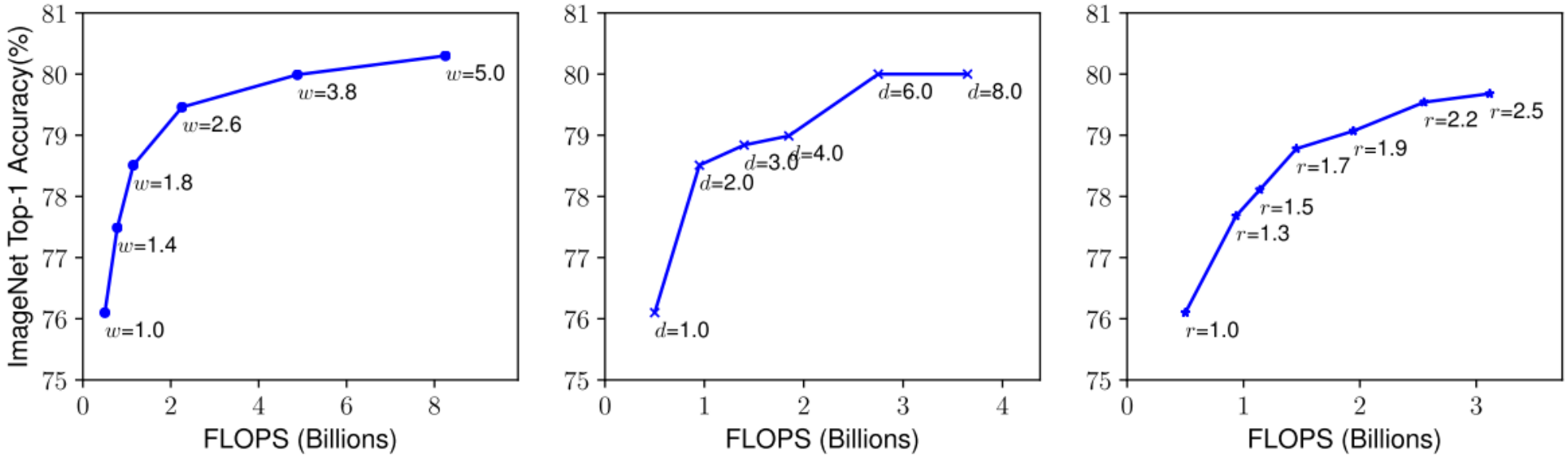
Efficient Net (2019)

- Uses a novel scaling method that uses a simple yet highly effective compound coefficient to scale up CNNs in a more structured manner.



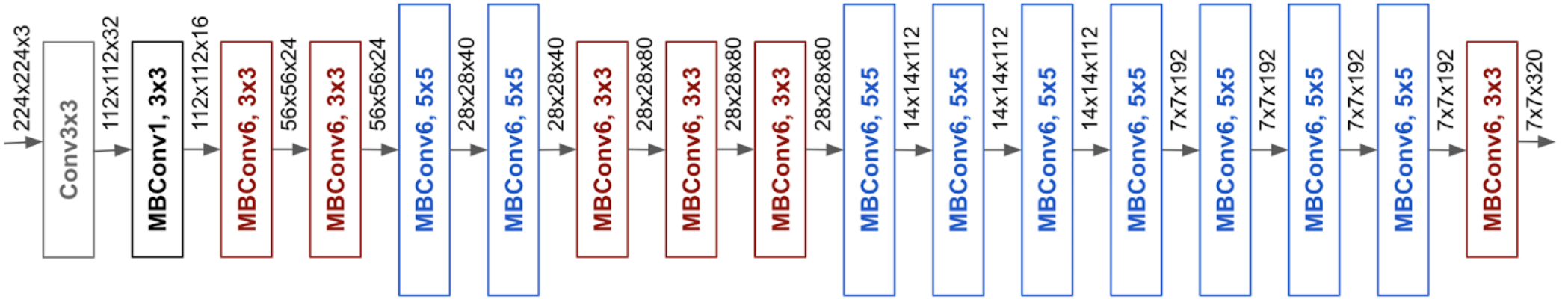
Efficient Net (2019)

- Observations
  - Scaling up any dimension of network width, depth, or resolution improves accuracy, but the accuracy gain diminishes for bigger models.
  - In order to pursue better accuracy and efficiency, it is critical to balance all dimensions of network width, depth, and resolution during ConvNet scaling.



Efficient Net (2019)

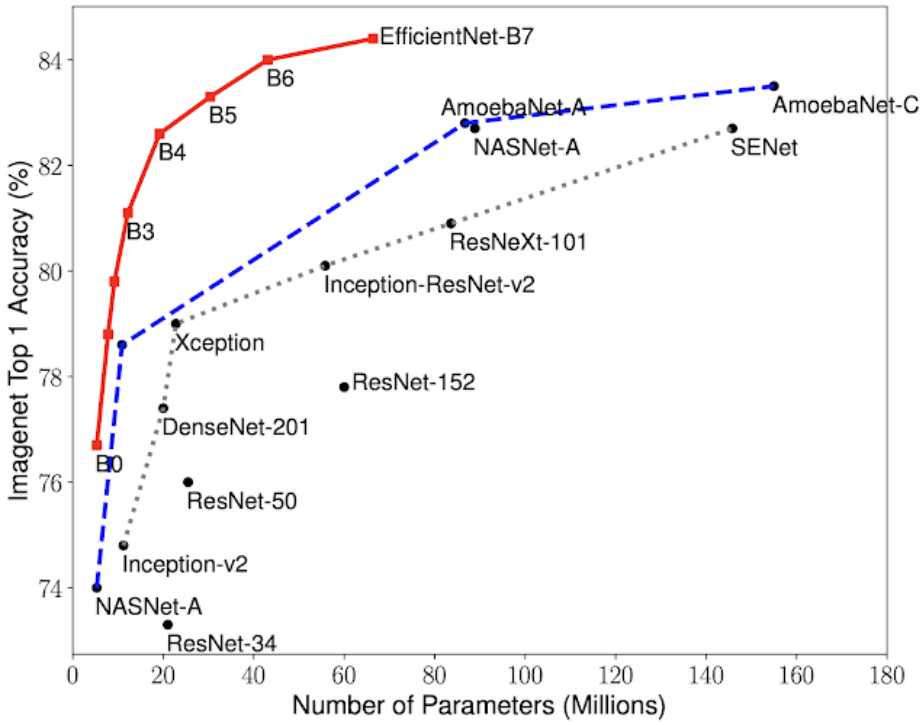
- Balances all dimensions of the network- width, depth, and image resolution - against the available resources would best improve overall performance
- Achieves both higher accuracy and better efficiency (up to 10x) in most cases



Efficient Net (2019)

- Uses a compound coefficient to uniformly scale network width, depth, and resolution in a principled way through a set of fixed scaling coefficients.
- E.g. if we want to use 2^N times more computational resources, then we can simply increase network depth by α^N , width by β^N , and image size by γ^N , where are constant coefficients determined by a small grid search on the original small model.

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



Keras Applications

Step 1 - Model

Step 2 - Preprocess

Step 3 - Predict

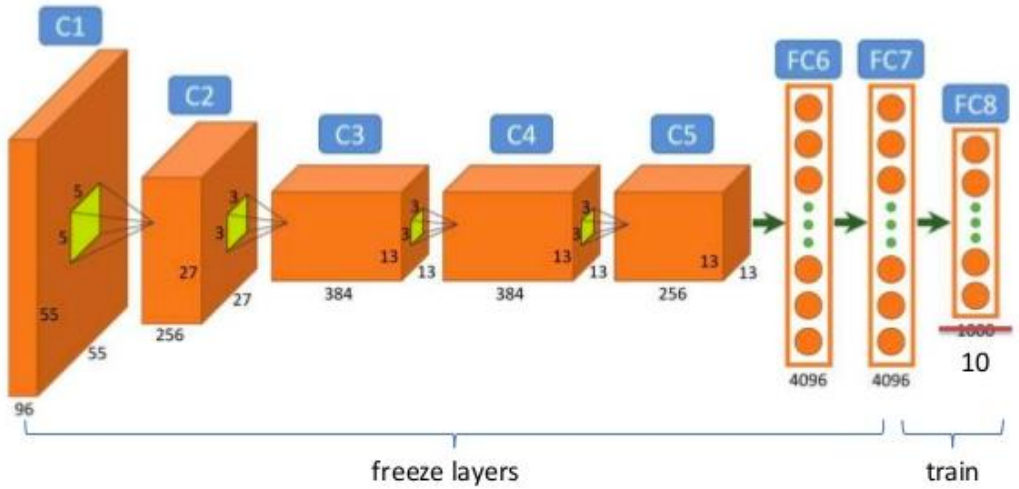
Step 4 - Decode Predictions

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5
ResNet152V2	232	78.0%	94.2%	60.4M	307	107.5	6.6
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
InceptionResNetV2	215	80.3%	95.3%	55.9M	449	130.2	10.0
MobileNet	16	70.4%	89.5%	4.3M	55	22.6	3.4
MobileNetV2	14	71.3%	90.1%	3.5M	105	25.9	3.8
DenseNet121	33	75.0%	92.3%	8.1M	242	77.1	5.4
DenseNet169	57	76.2%	93.2%	14.3M	338	96.4	6.3
DenseNet201	80	77.3%	93.6%	20.2M	402	127.2	6.7
NASNetMobile	23	74.4%	91.9%	5.3M	389	27.0	6.7
NASNetLarge	343	82.5%	96.0%	88.9M	533	344.5	20.0
EfficientNetB0	29	77.1%	93.3%	5.3M	132	46.0	4.9
EfficientNetB1	31	79.1%	94.4%	7.9M	186	60.2	5.6
EfficientNetB2	36	80.1%	94.9%	9.2M	186	80.8	6.5

https://keras.io/api/applications/

Transfer Learning

Fine-tune a neural network architecture that has done well on a more general challenge for another specific but related problem



Strategies

1. CNN as fixed feature extractor: Take a CNN pre-trained on ImageNet, remove the last fully-connected layer (this layer's outputs are the 1000 class scores for a different task like ImageNet), then treat the rest of the CNN as a fixed feature extractor for the new dataset

2. Fine-tuning the CNN: The second strategy is to not only replace and retrain the classifier on top of the CNN on the new dataset but to also fine-tune the weights of the pre-trained network by continuing the backpropagation

3. Pretrained models: Since modern CNNs take 2-3 weeks to train across multiple GPUs on ImageNet, it is common to see people release their final CNN checkpoints for the benefit of others who can use the networks for fine-tuning

http://cs231n.github.io/transfer-learning/ https://github.com/BVLC/caffe/wiki/Model-Zoo