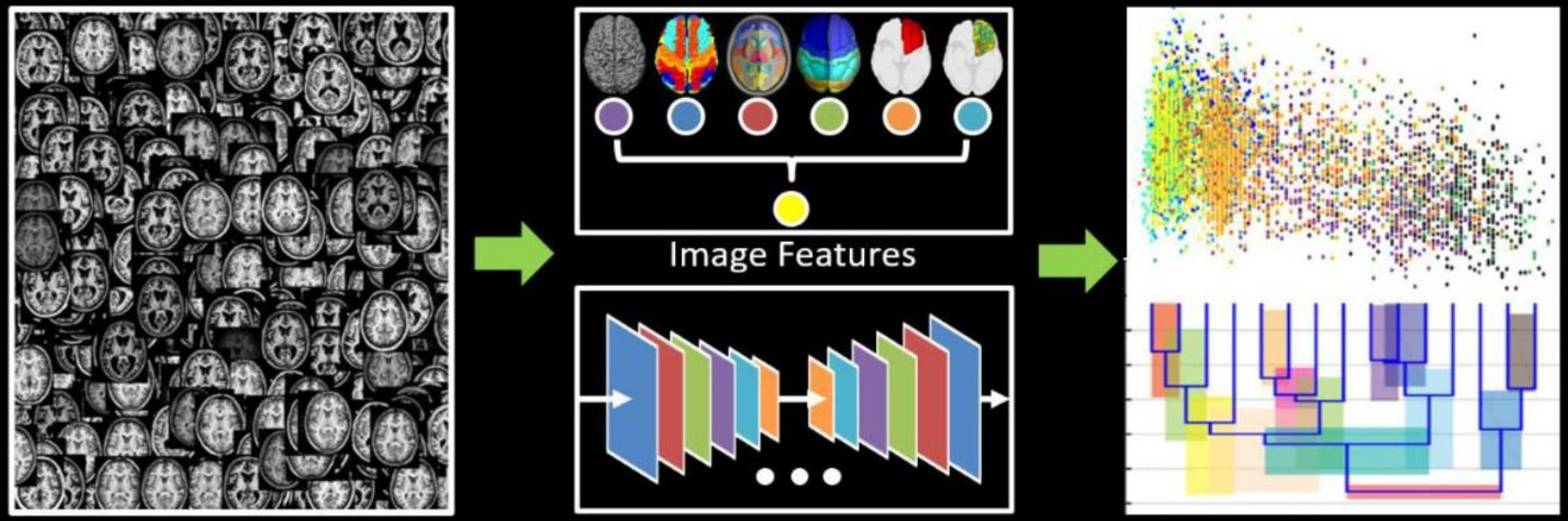


Deep Image Classification (Medical Image Diagnosis)

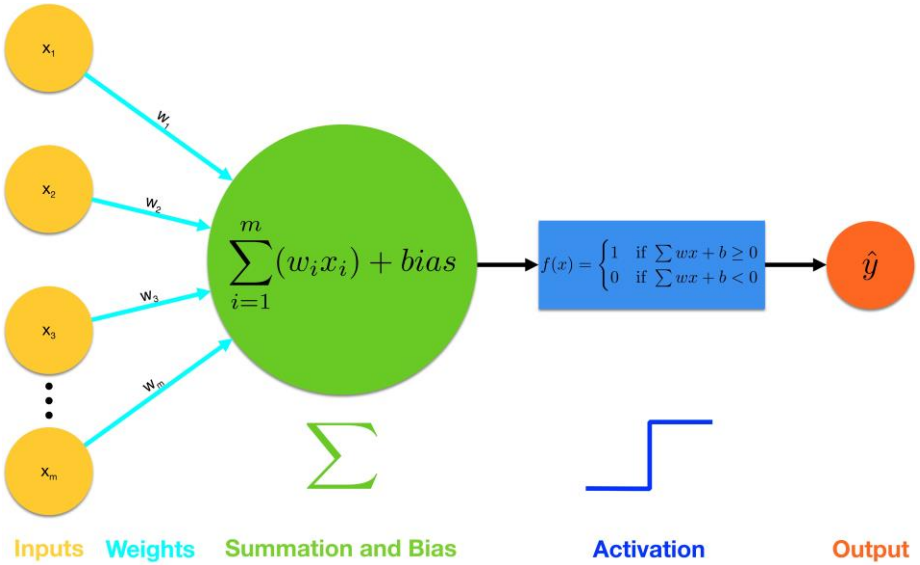
[Spring 2020 CS-8395 Deep Learning in Medical Image Computing]

Instructor: Yuankai Huo, Ph.D.
Department of Electrical Engineering and Computer Science
Vanderbilt University

Topics

- Review
- Classification
- Discuss the Paper

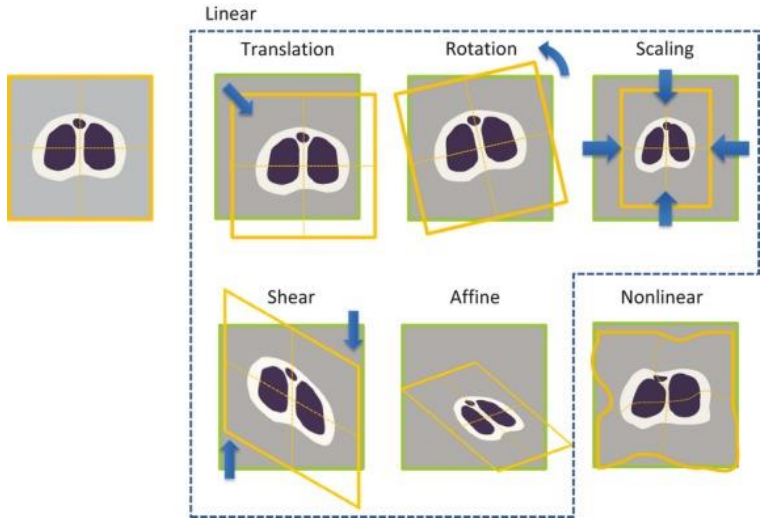
Neural Network



$$\hat{y} = \sigma\left(w^T x + b\right)$$

$$w^T x + b$$

linear transformation
(e.g. affine registration)

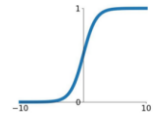


http://nahuakang.com/page3/

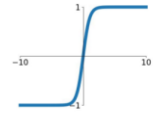
$$\sigma\left(\cdot\right)$$

Activation Functions

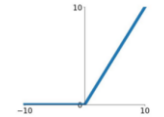
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



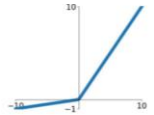
tanh
$\tanh(x)$



ReLU
$\max(0, x)$

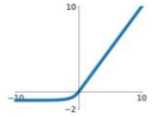


Leaky ReLU
$\max(0.1x, x)$

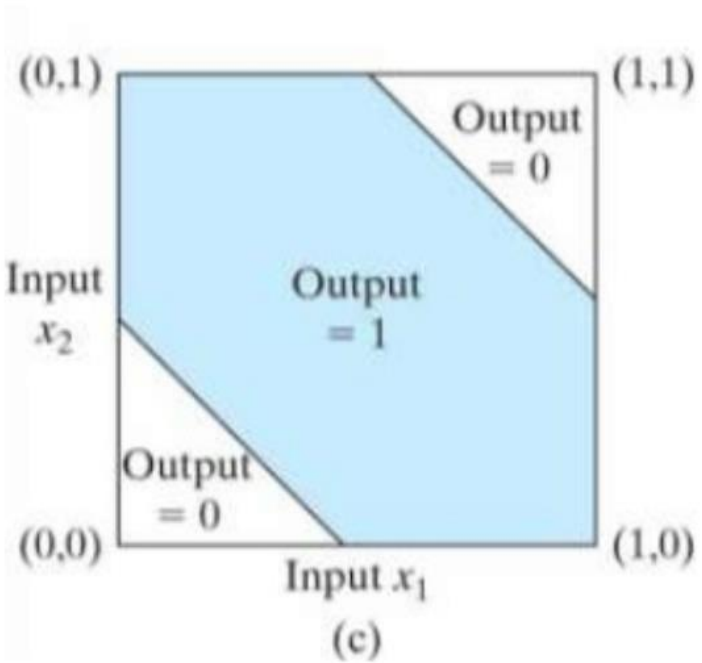


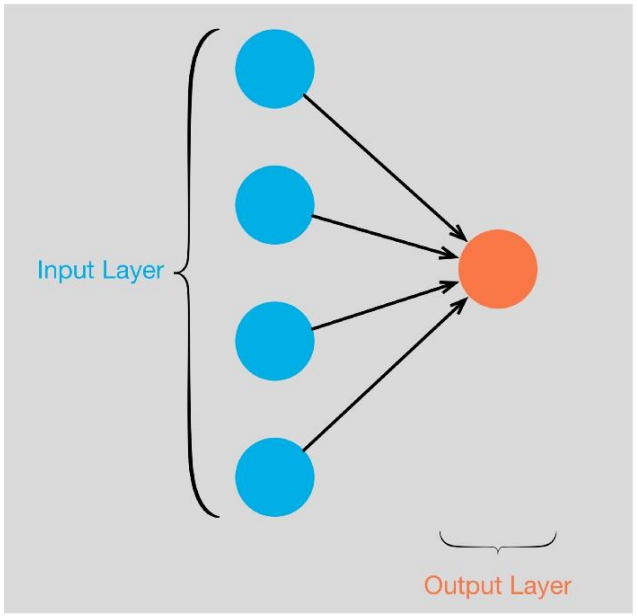
Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

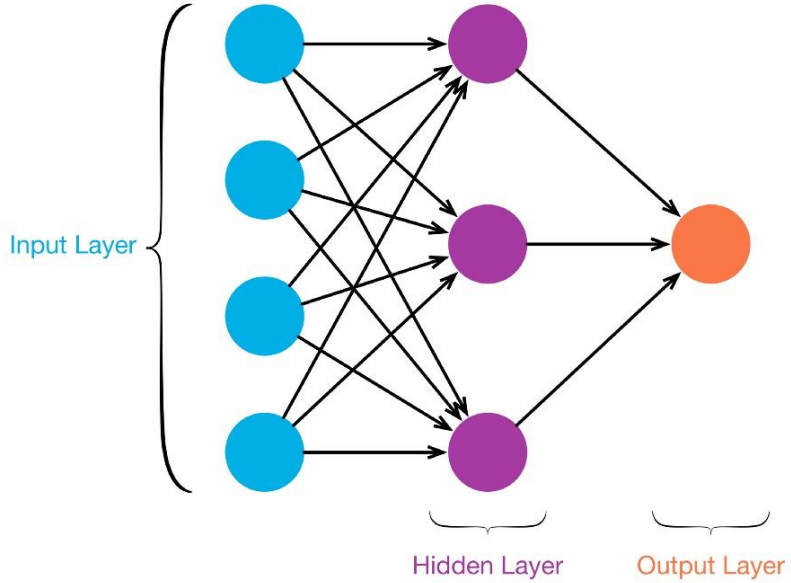
ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Multi-Layer Perceptrons



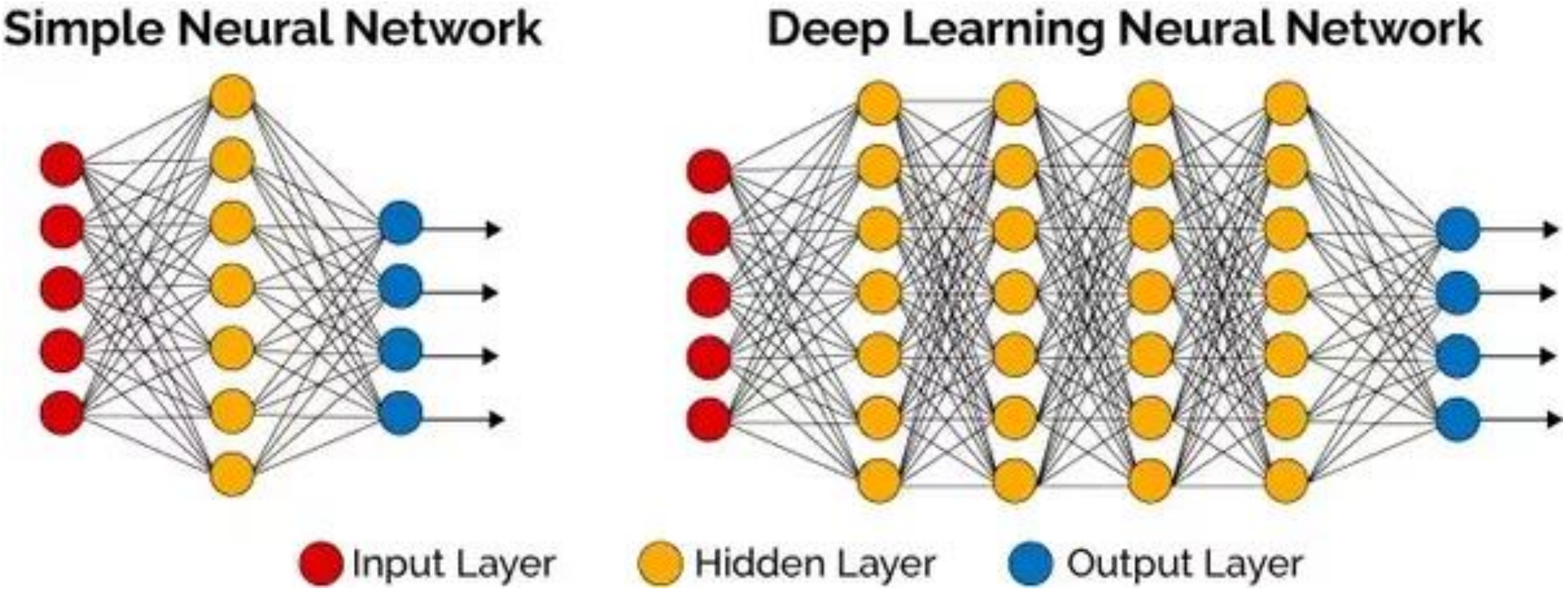




https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f

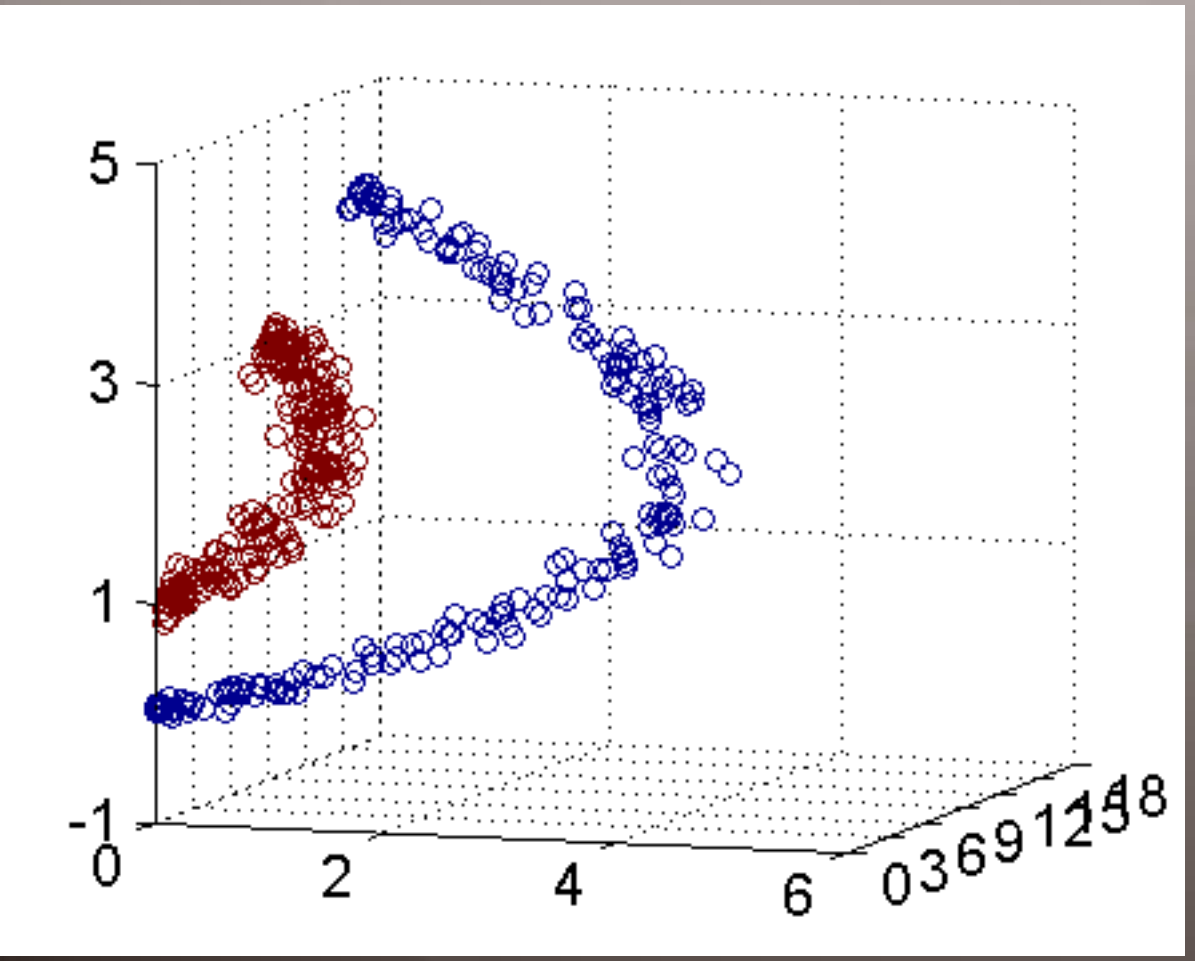
https://www.ibm.com/developerworks/library/cc-cognitive-neural-networks-deep-dive/index.html

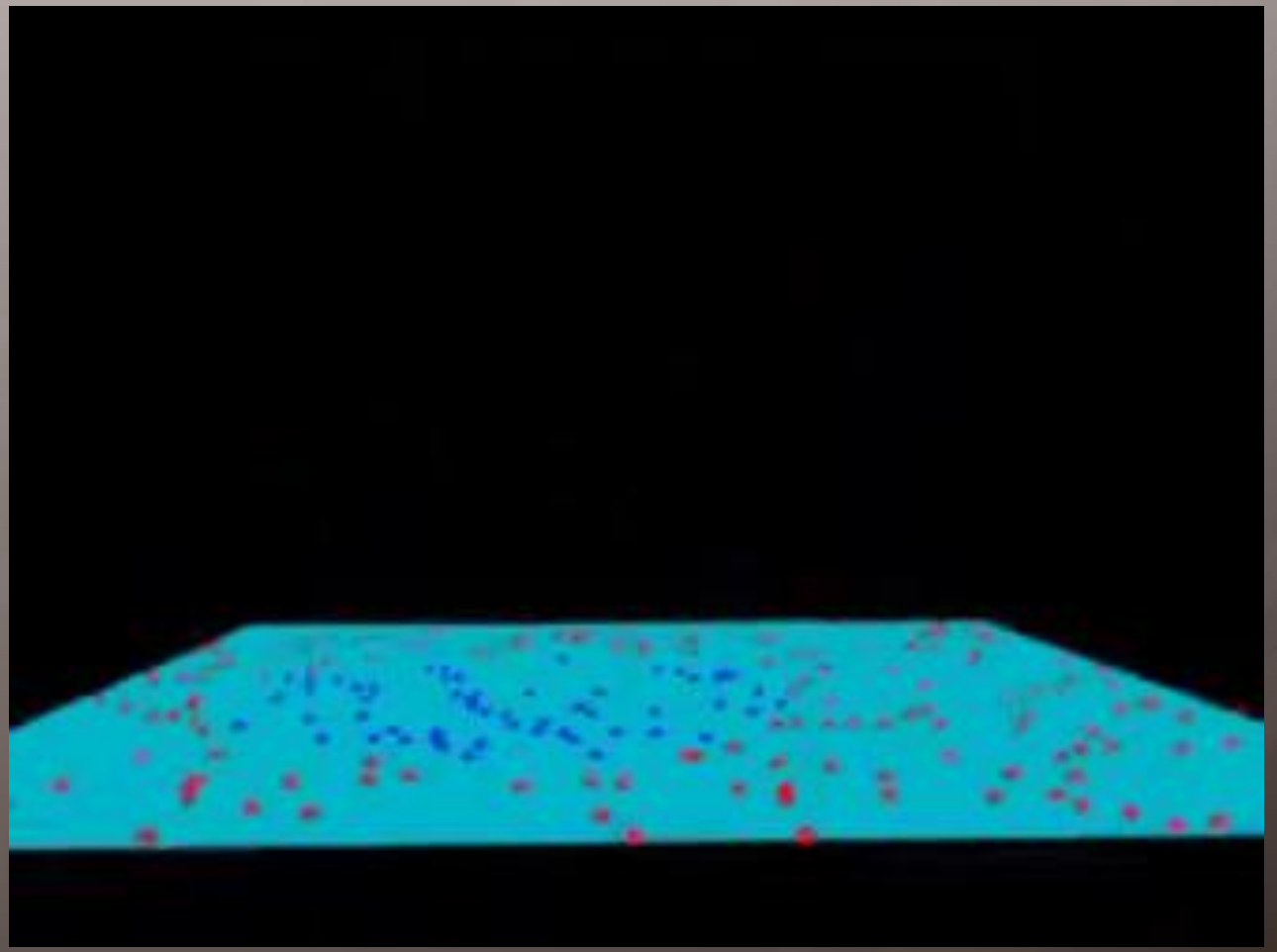
Deep Learn = Deep Neural Network



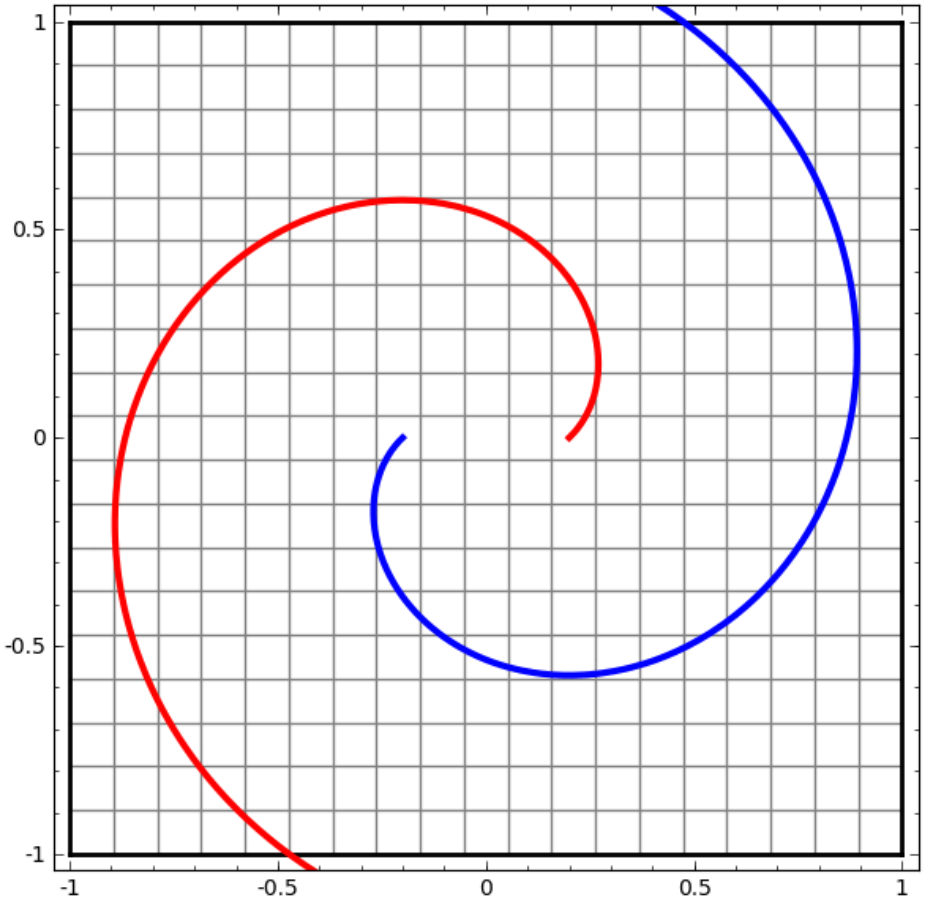
https://www.quora.com/What-is-the-difference-between-Neural-Networks-and-Deep-Learning

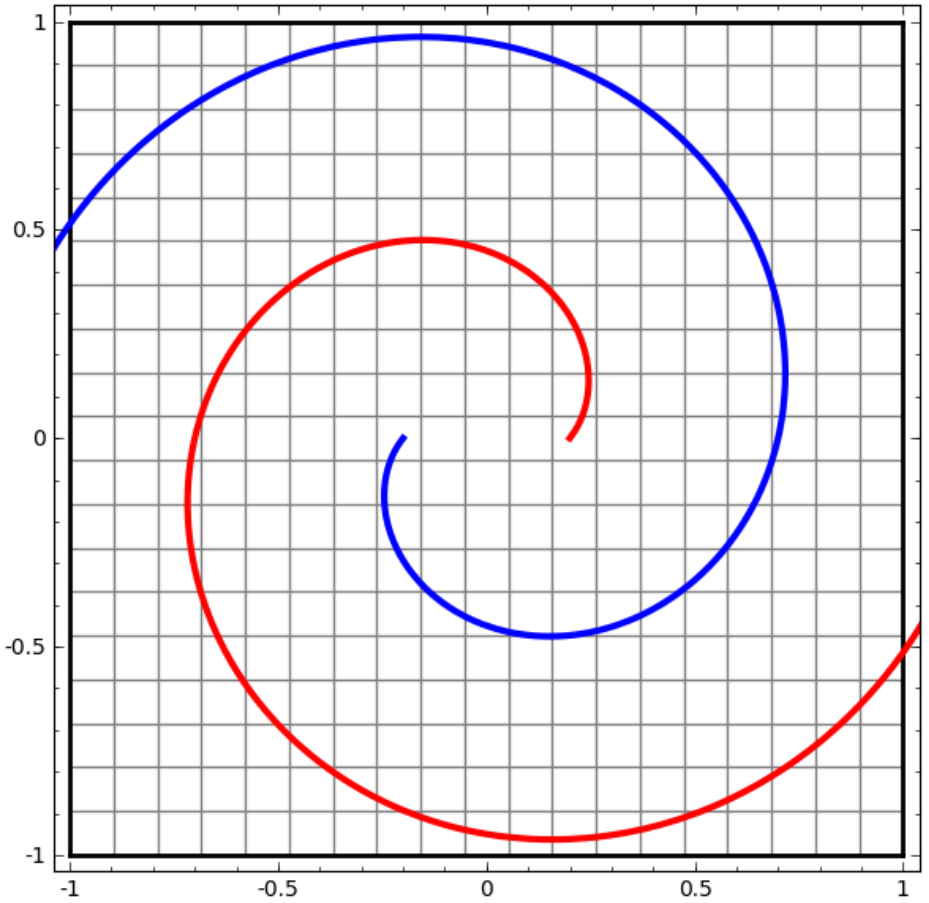
Hyper Plane





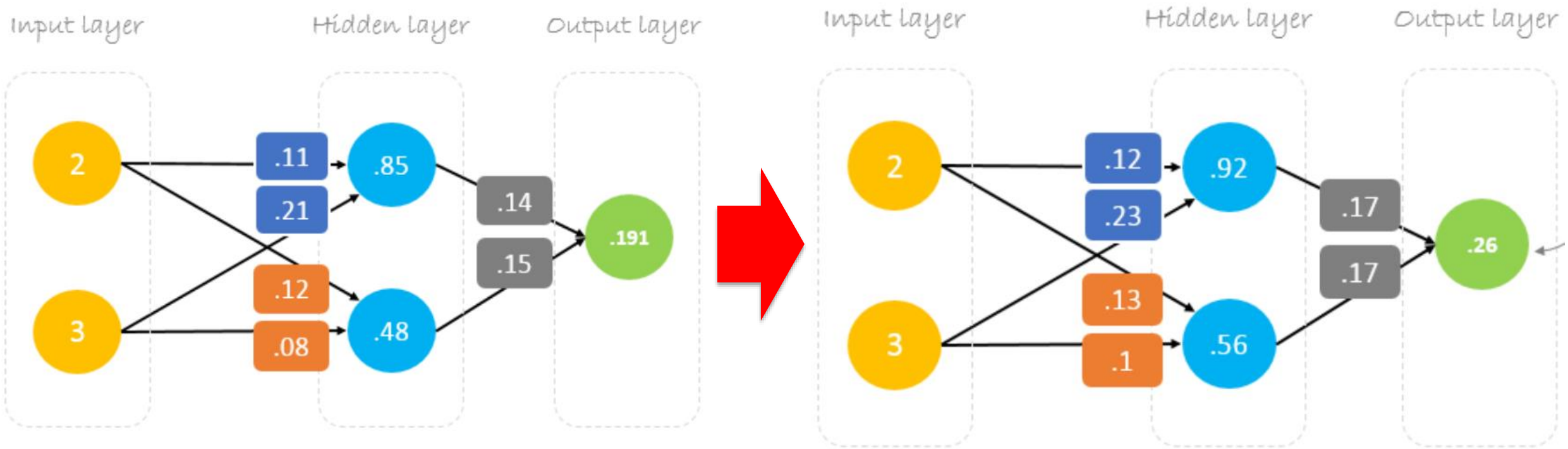
More Hidden Layers





http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Backward Path



http://hmkcode.github.io/ai/backpropagation-step-by-step/

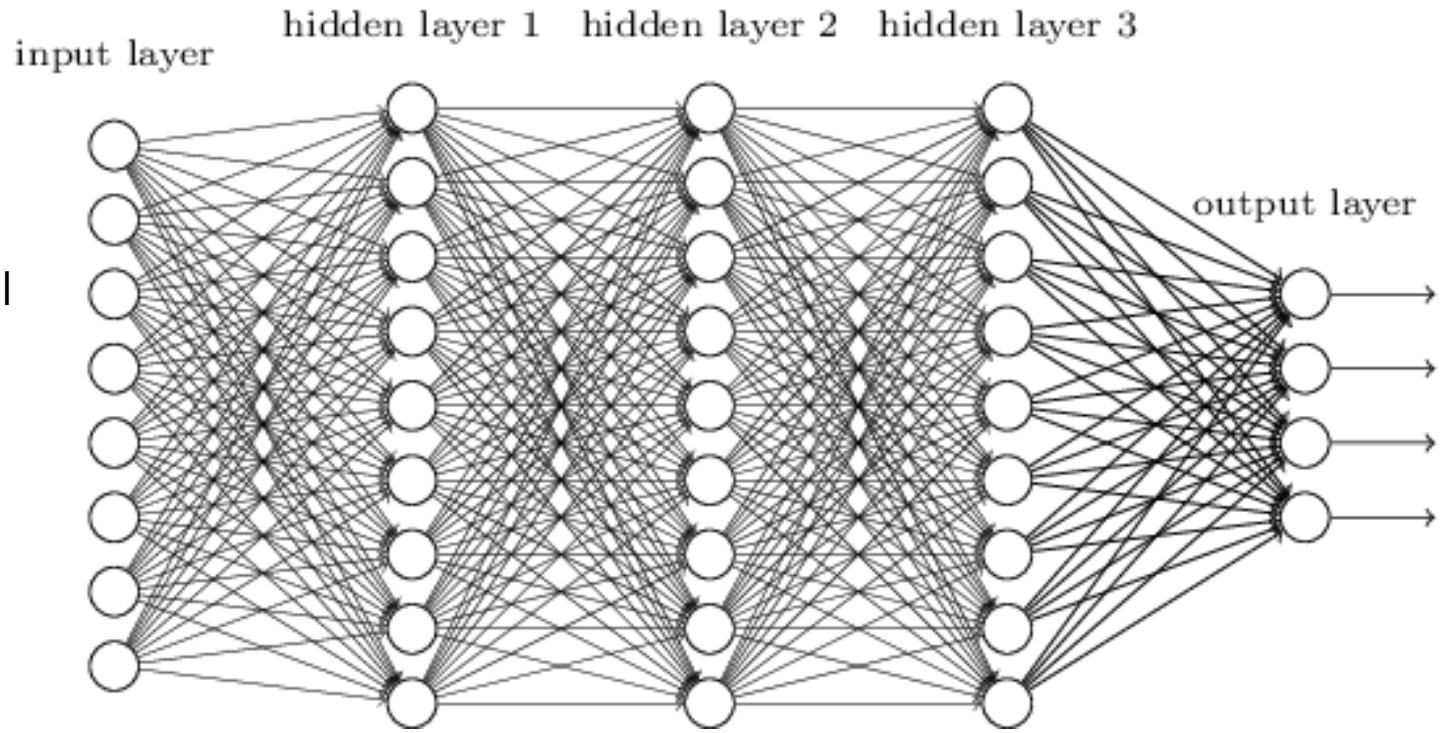
1D Data

GPA

SAT

Football

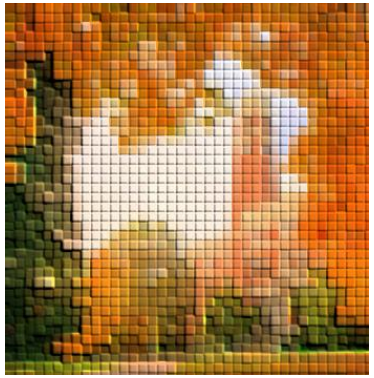
…

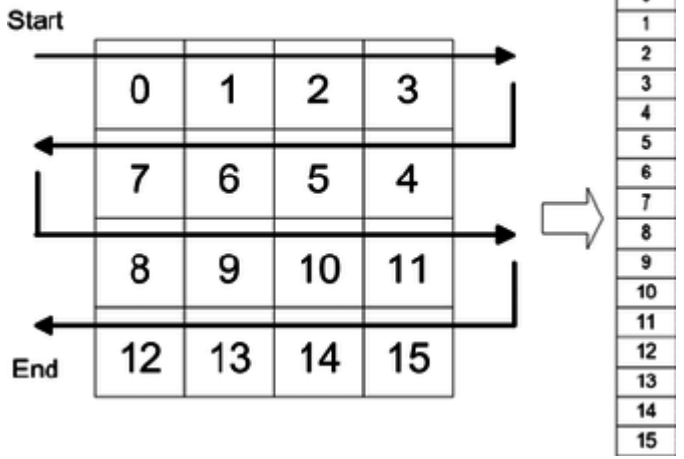


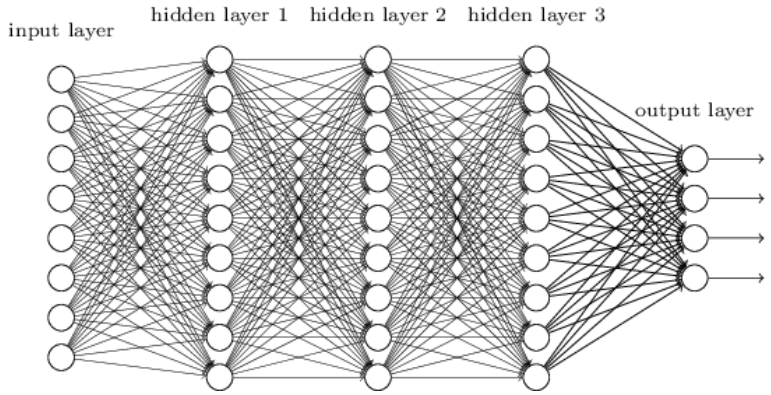
Vanderbilt

UTK

Image Representation 1: Dense



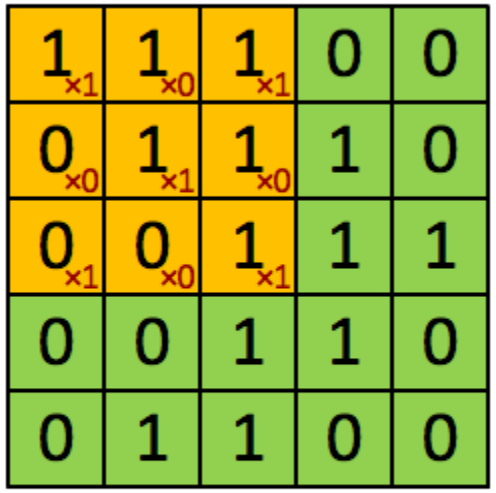


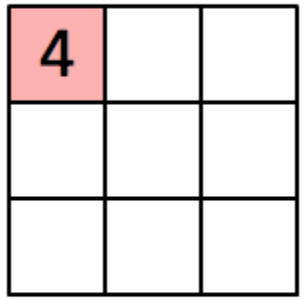


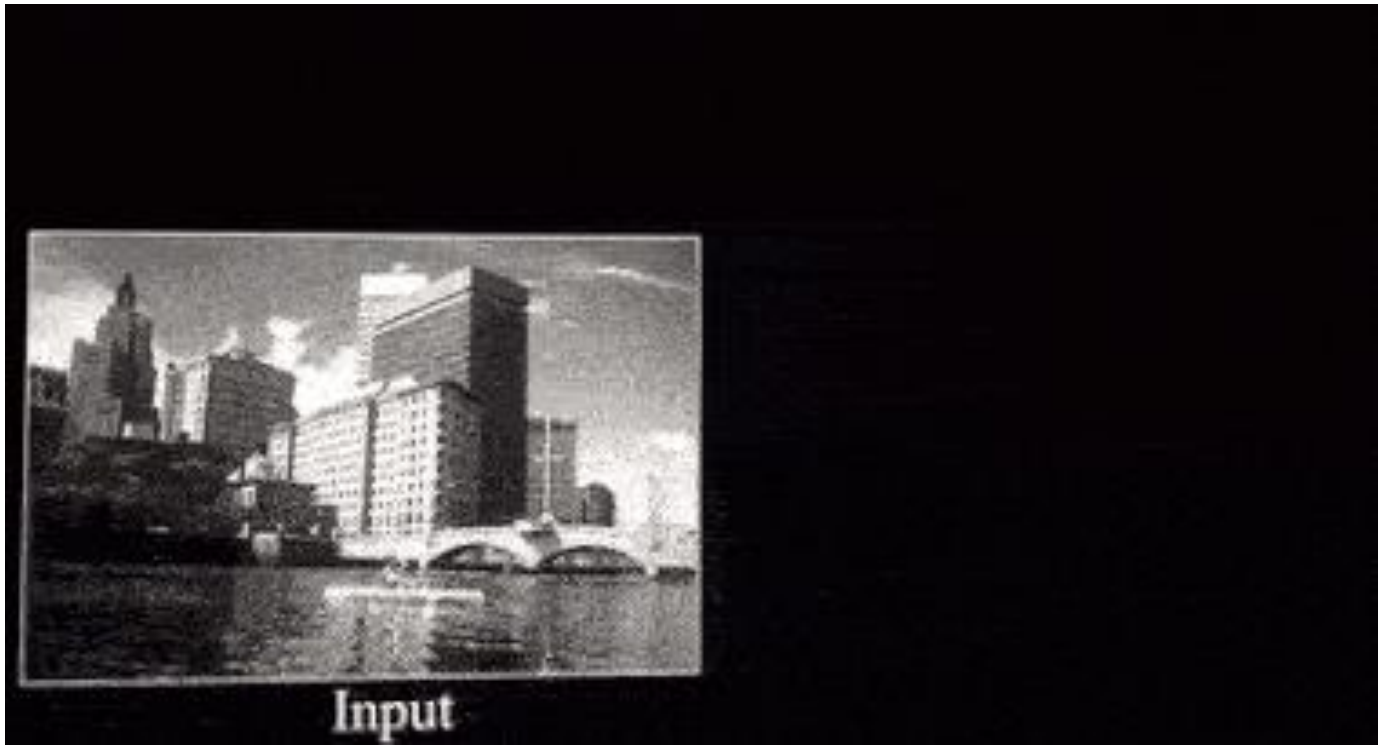
Vanderbilt

https://www.usnews.com/best-colleges/vanderbilt-3535
https://www.researchgate.net/figure/Scanning-2D-blocks-left-into-1D-vector-as-input-into-logical-transform-middle-the_fig4_228345705

Image Representation 2: Convolution





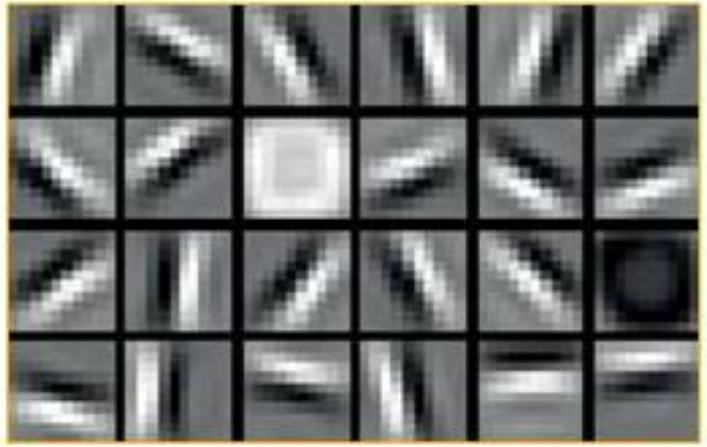


http://ufldl.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/convolutional_neural_networks.html

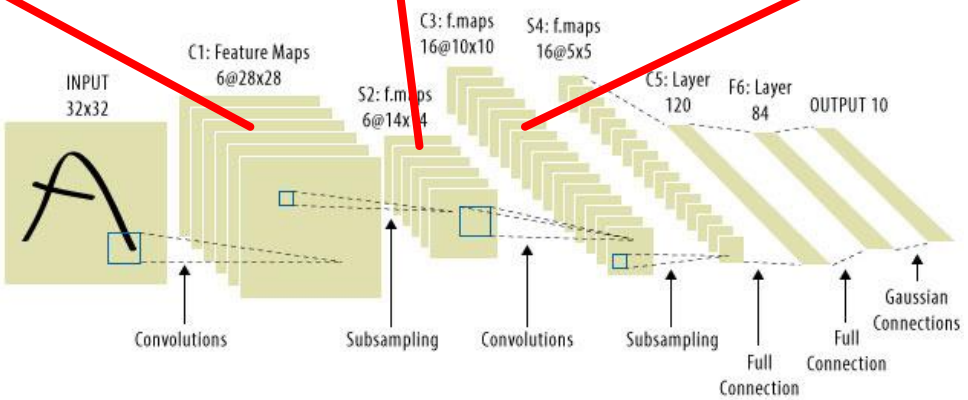
Different Layers

Low-level features



Mid-level features

High-level features



https://www.all-electronics.de/wp-content/uploads/2016/05/Bild31.jpg

"Audi A7"

Parameters

https://www.usnews.com/best-colleges/vanderbilt-3535

```
import utilities
```

```
img_rows, img_cols = 224, 224
colors = 3
input_size = img_rows * img_cols * colors
input_shape = (img_rows, img_cols, colors)

num_classes = 10
```

https://medium.com/@zhang_yang/number-of-parameters-in-dense-and-convolutional-neural-networks-34b54c2ec349

DNN vs CNN

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential([
    Dense(32, activation='relu', input_shape=(input_size,)),
    Dense(64, activation='relu'),
    Dense(128, activation='relu'),
    Dense(num_classes, activation='softmax')
])

model.summary()
```

```
Using TensorFlow backend.
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 32)                4816928
_________________________________________________________________
dense_2 (Dense)              (None, 64)                2112
_________________________________________________________________
dense_3 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_4 (Dense)              (None, 10)                1290
=================================================================
Total params: 4,828,650
Trainable params: 4,828,650
Non-trainable params: 0
```

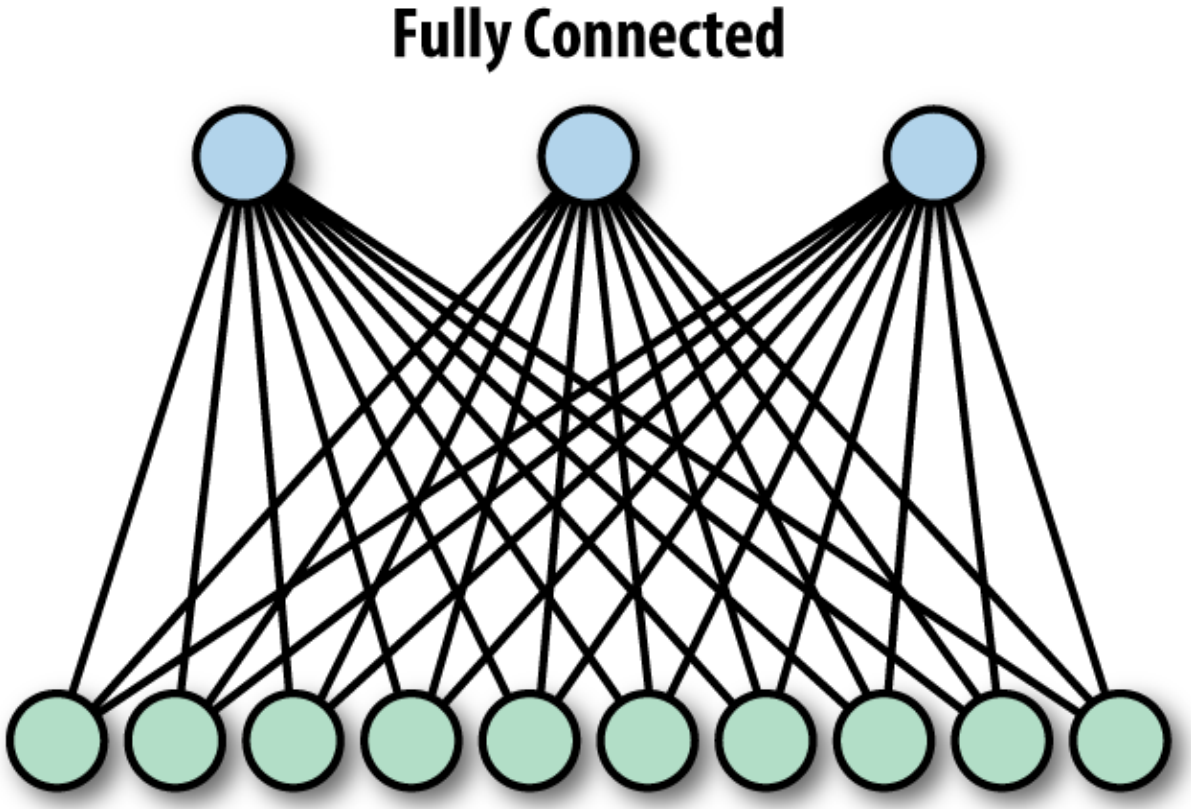
DNN

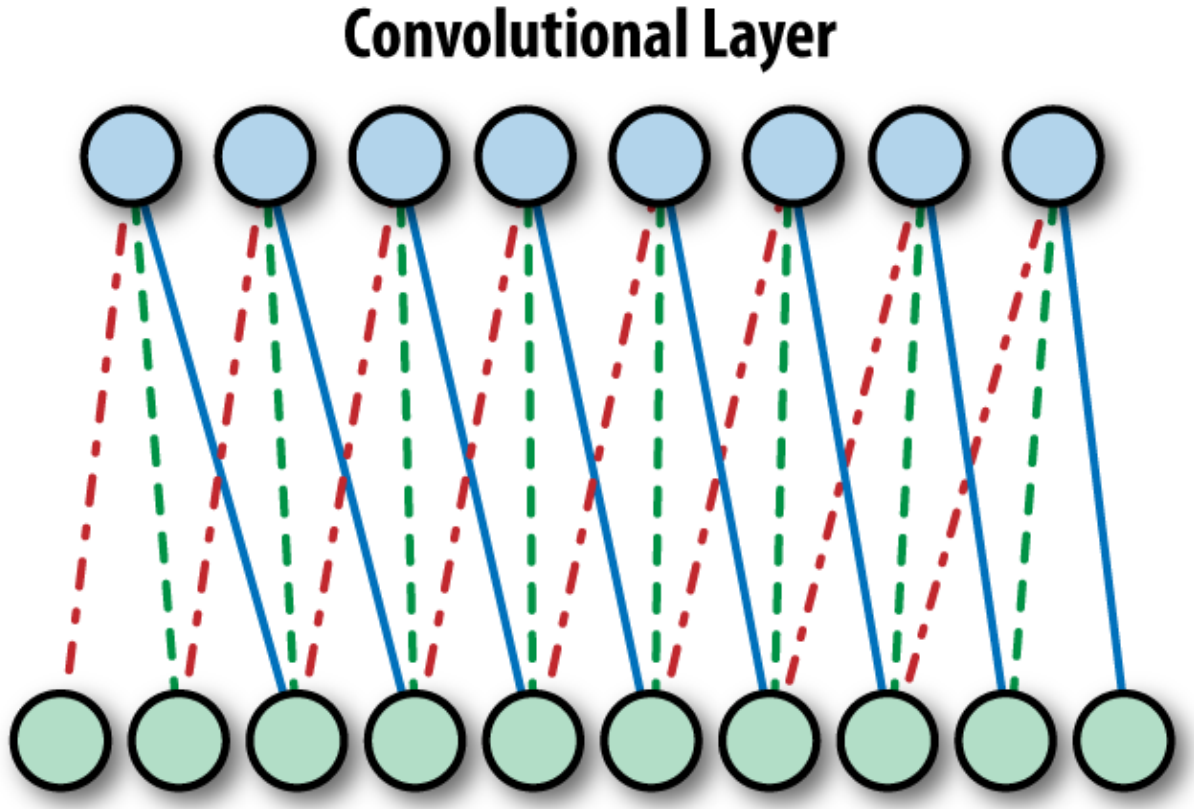
```
from keras.models import Sequential
from keras.layers import Conv2D


model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
    Conv2D(64, (3, 3), activation='relu'),
    Conv2D(128, (3, 3), activation='relu'),
    Dense(num_classes, activation='softmax')
])

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 222, 222, 32)      896
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 220, 220, 64)      18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 218, 218, 128)     73856
_________________________________________________________________
dense_5 (Dense)              (None, 218, 218, 10)      1290
=================================================================
Total params: 94,538
Trainable params: 94,538
Non-trainable params: 0
```

CNN

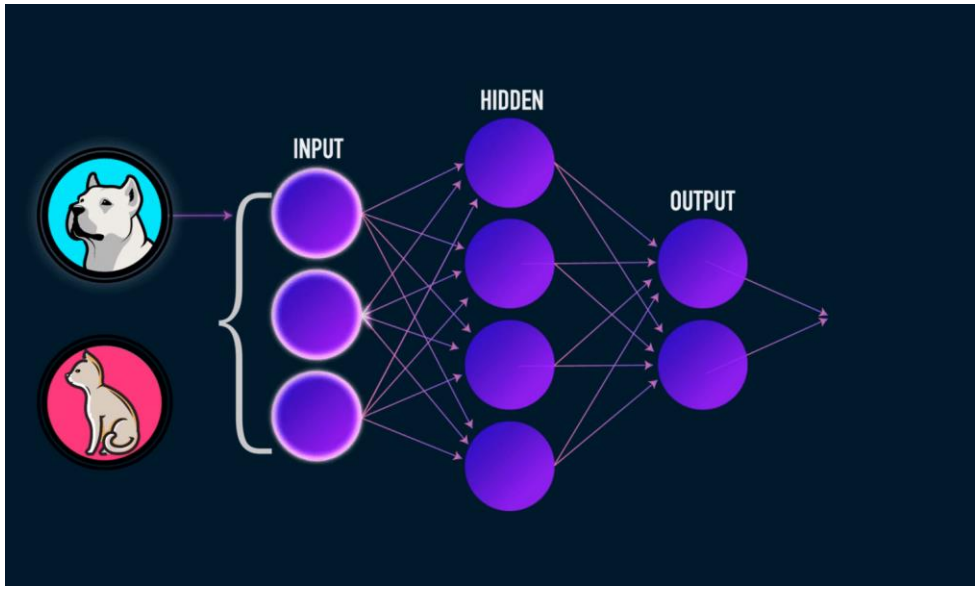
Reduce Parameters

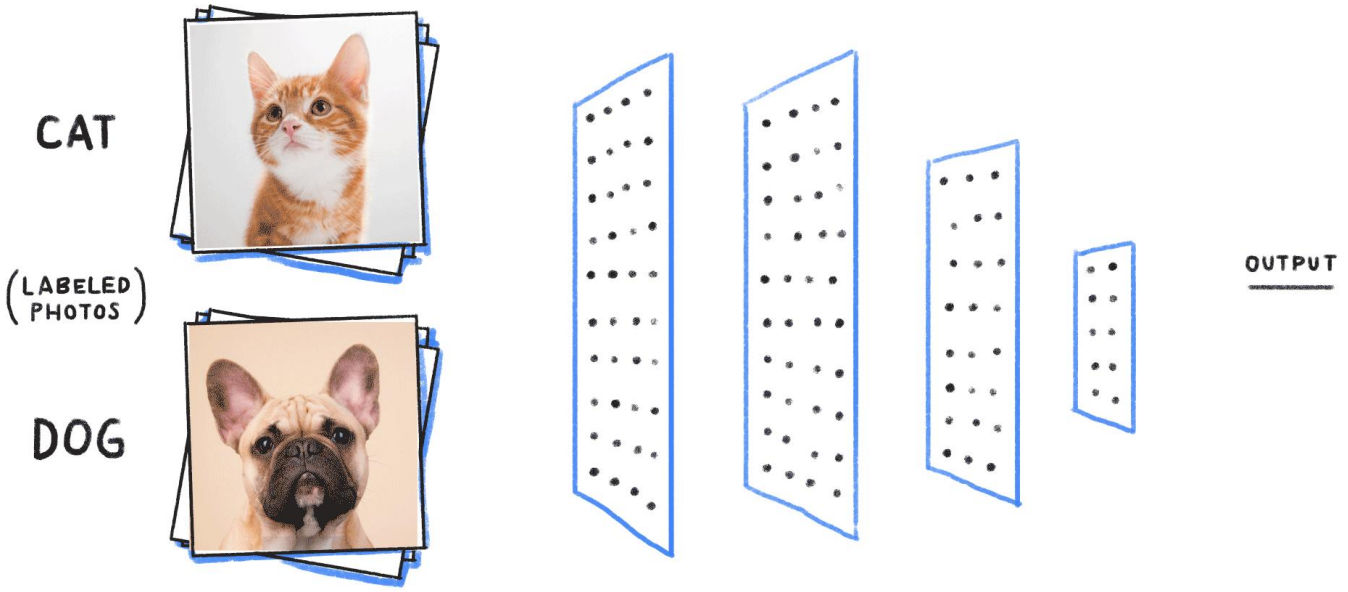




https://www.safaribooksonline.com/library/view/learning-tensorflow/9781491978504/ch04.html

Go Deep!



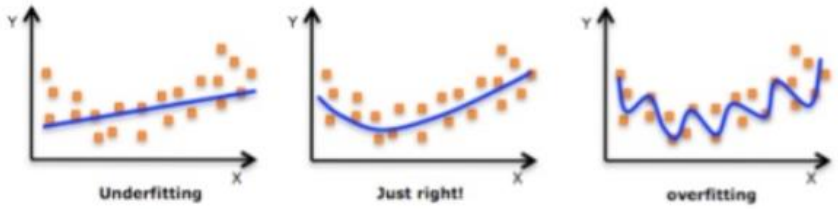


https://www.analyticsindiamag.com/most-common-activation-functions-in-neural-networks-and-rationale-behind-it/

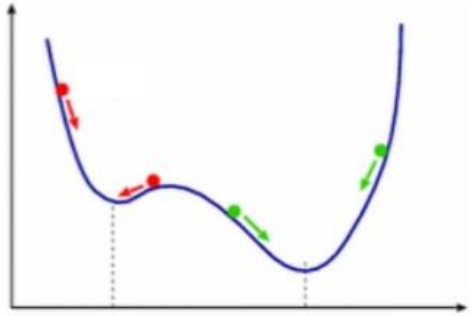
https://becominghuman.ai/building-an-image-classifier-using-deep-learning-in-python-totally-from-a-beginners-perspective-be8dbaf22dd8

Challenges

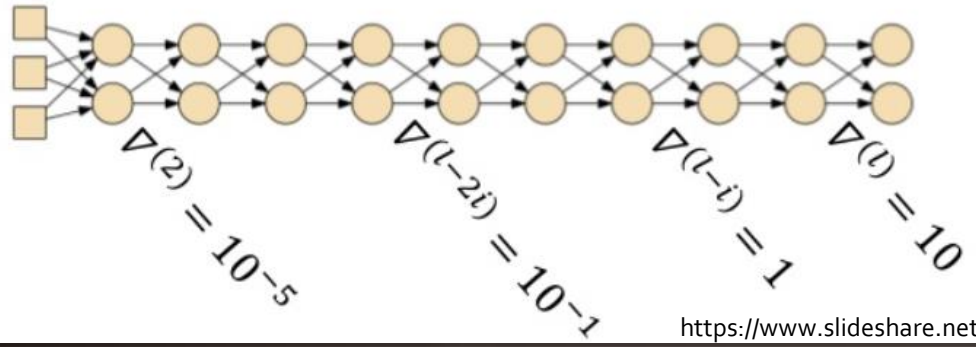
- Big networks => Too huge separating ability => **Overfitting**



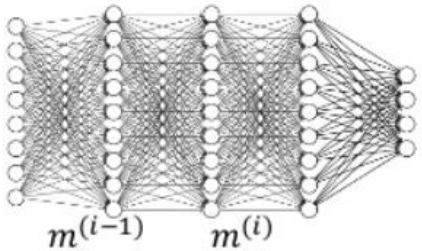
- Complex error's surface => **Local minimum**



- **Vanishing gradient problem** during training



- Curse of dimensionality => **memory & computations**

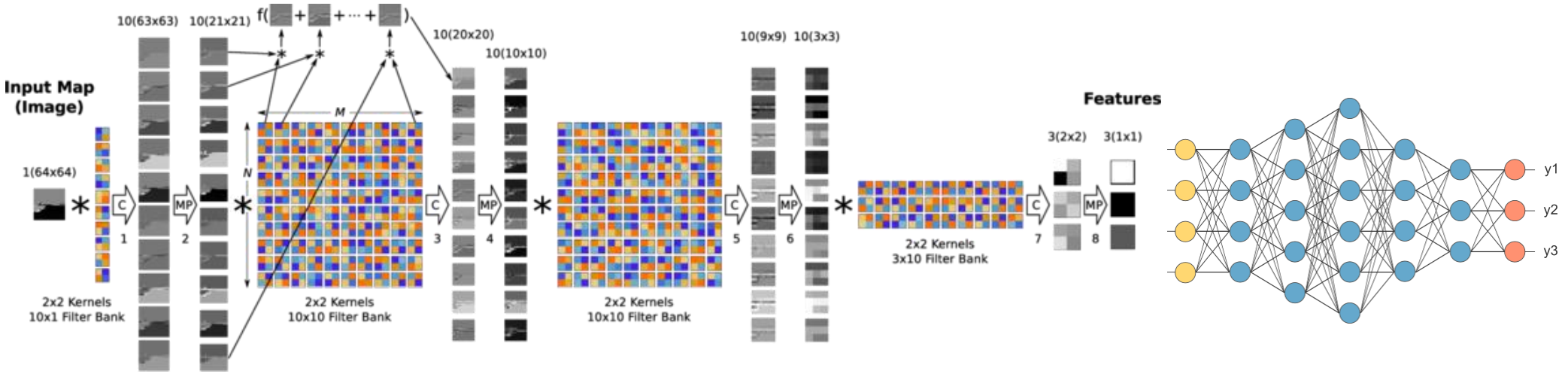


$$\dim(W^{(i)}) = m^{(i-1)} * m^{(i)}$$

Overfitting
Local minimum
Gradient Vanishing
Computation & Memory

https://www.slideshare.net/baiev1/neural-networks-overview

Deep Convolutional Network Classification



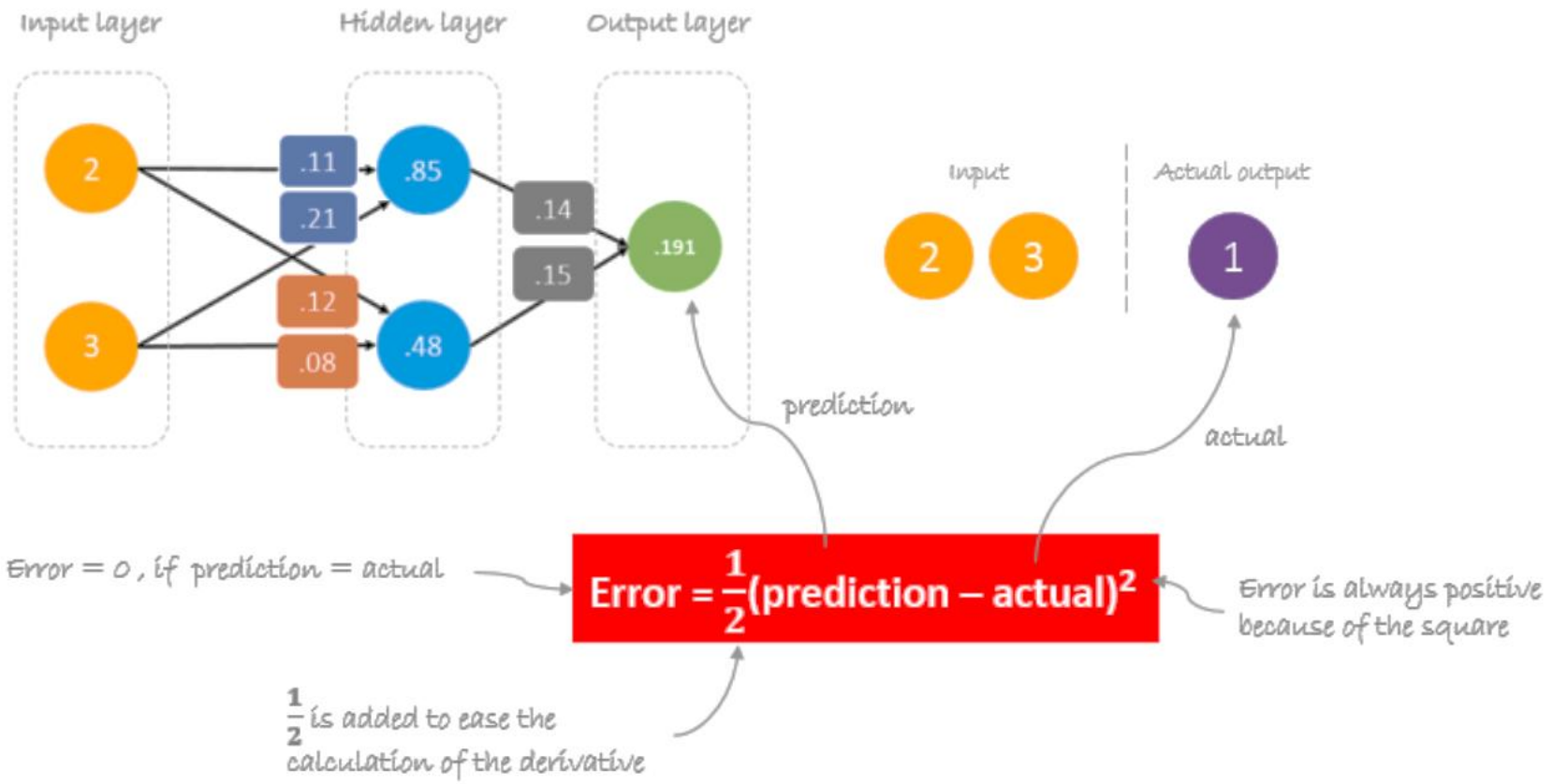
Loss Function

Convolutional Layer

Fully Connected Layer (Dense Layer)

https://www.researchgate.net/figure/Max-Pooling-Convolutional-Neural-Network-MPCNN-with-8-layers-alternating-between_fig3_266656356

How To Get Loss



Three Examples

computed			targets			correct?
0.3	0.3	0.4	0	0	1	yes
0.3	0.4	0.3	0	1	0	yes
0.1	0.2	0.7	1	0	0	no

computed			targets			correct?
0.1	0.2	0.7	0	0	1	yes
0.1	0.7	0.2	0	1	0	yes
0.3	0.4	0.3	1	0	0	no

Classification Accuracy: CA = (#correct / #all)

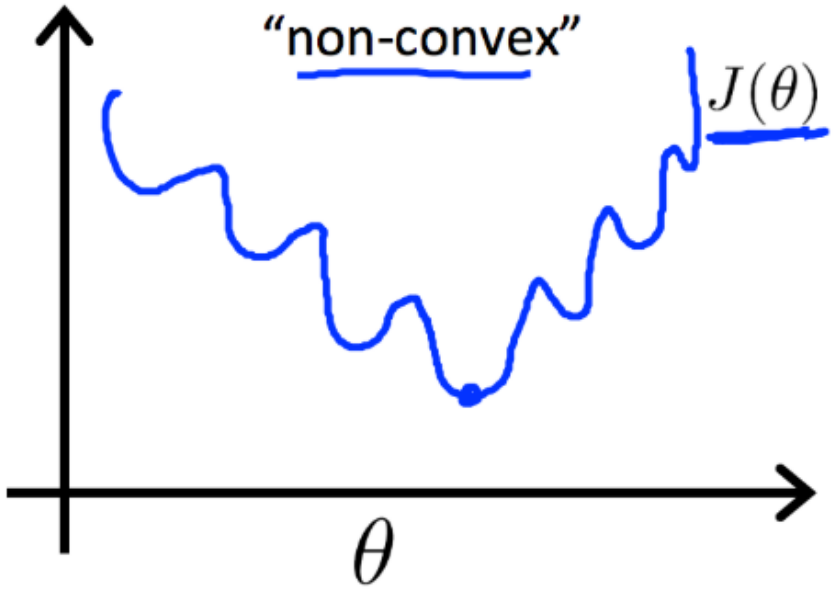
Mean square Error (MSE)

MSE $\quad \mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$

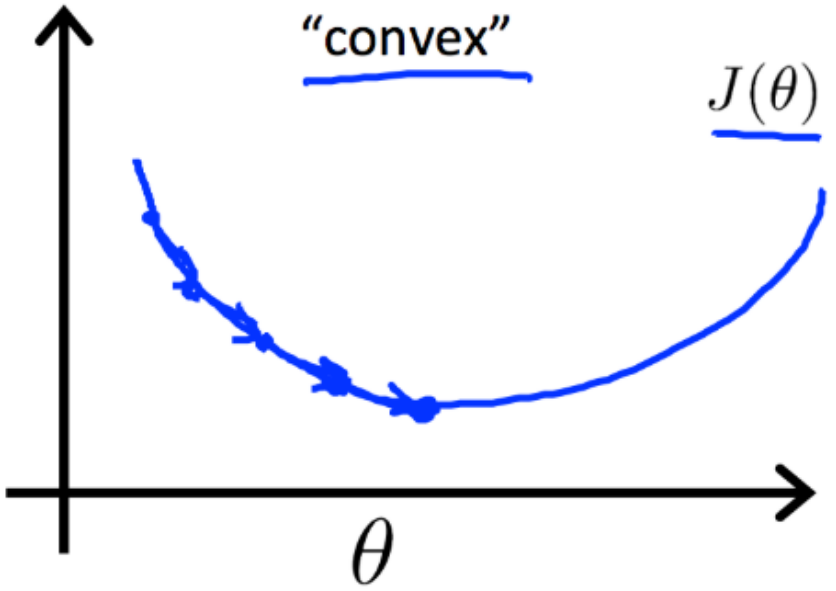
Cross-entropy $\quad H(p,q) = -\sum_x p(x)\,\log q(x)$

MSE vs Cross-entropy

MSE



Cross-entropy



Andrew Ng

Cross Entropy

$$H(y, \hat{y}) = \sum_i y_i \log \frac{1}{\hat{y}_i} = -\sum_i y_i \log \hat{y}_i$$

Entropy

Claude Shannon

https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/

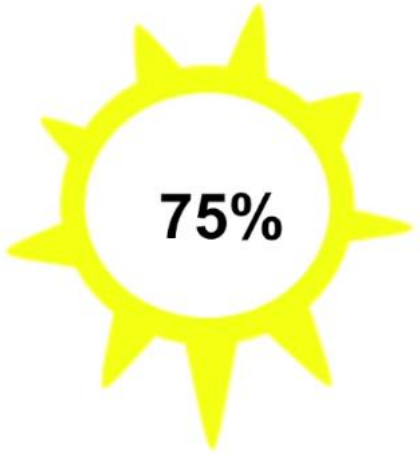
Entropy

Entropy:

$$H(\mathbf{p}) = -\Sigma_i\, p_i \log_2(p_i)$$

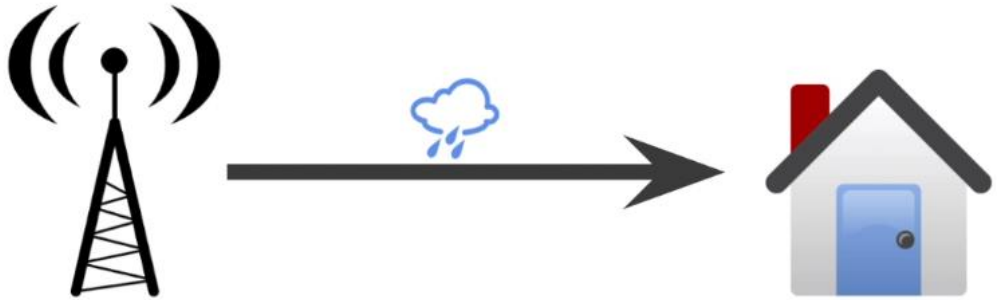
https://www.youtube.com/watch?v=ErfnhcEV1O8

Entropy



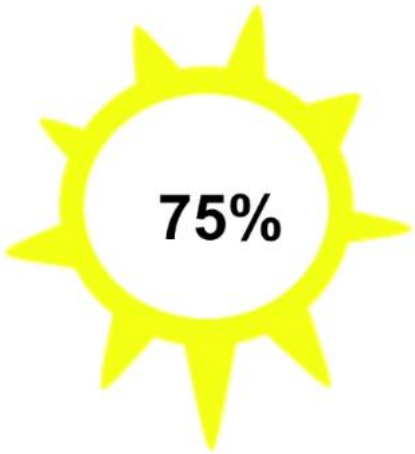
Entropy:

$$H(\mathbf{p}) = -\Sigma_i\, p_i \log_2(p_i)$$



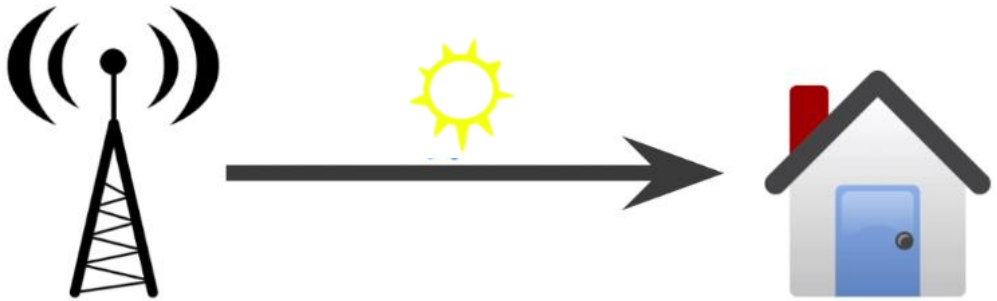
https://www.youtube.com/watch?v=ErfnhcEV1O8

Entropy



Entropy:

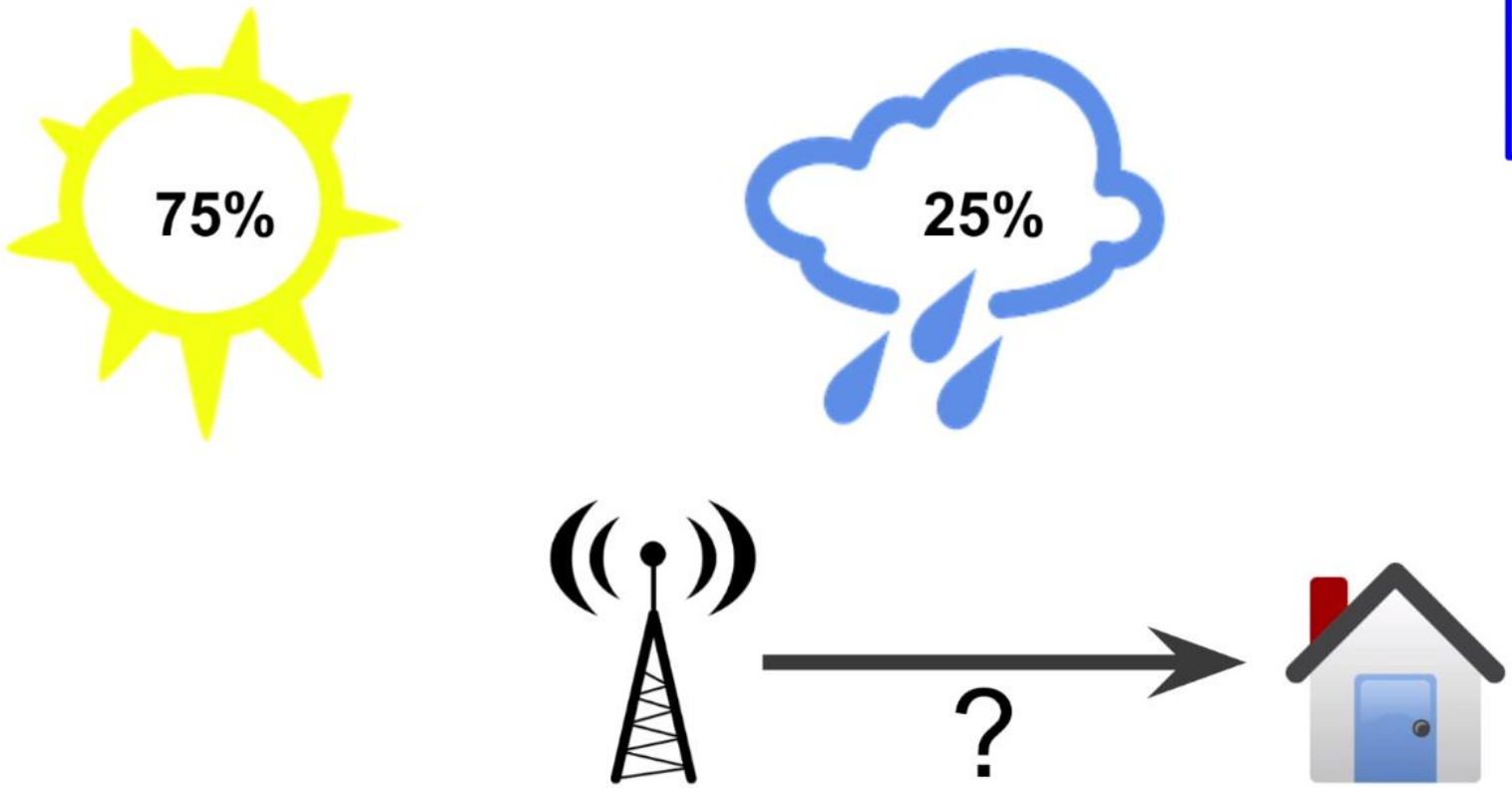
$$H(\mathbf{p}) = -\Sigma_i\, p_i \log_2(p_i)$$



https://www.youtube.com/watch?v=ErfnhcEV1O8

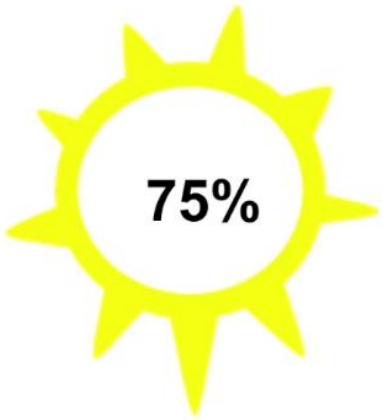
Entropy

Entropy:
$H(\mathbf{p}) = -\Sigma_i p_i \log_2(p_i)$



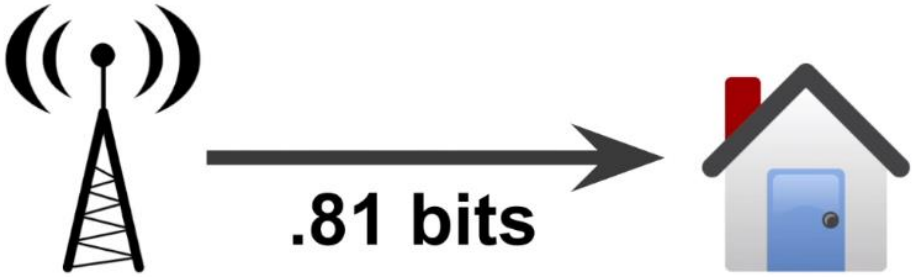
https://www.youtube.com/watch?v=ErfnhcEV1O8

Entropy



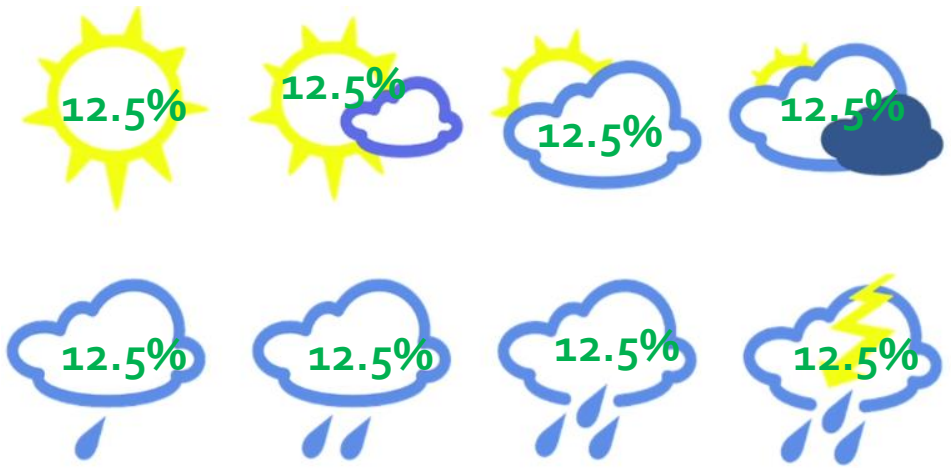
Entropy:
$H(\mathbf{p}) = -\Sigma_i\, p_i \log_2(p_i)$

75% × 0.41
+ 25% × 2
= 0.81 bits

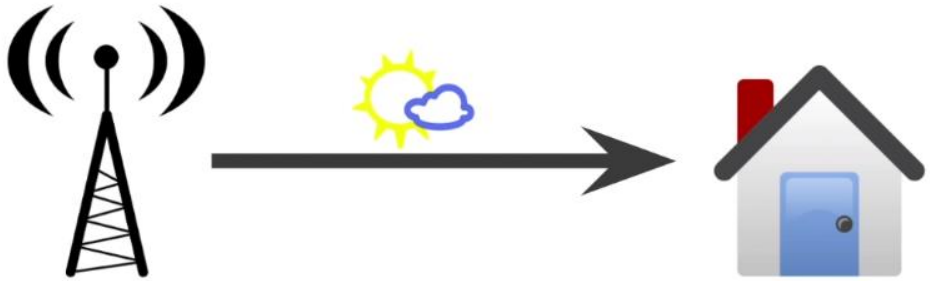


https://www.youtube.com/watch?v=ErfnhcEV1O8

Entropy



Entropy:
$H(\mathbf{p}) = -\Sigma_i\, p_i \log_2(p_i)$



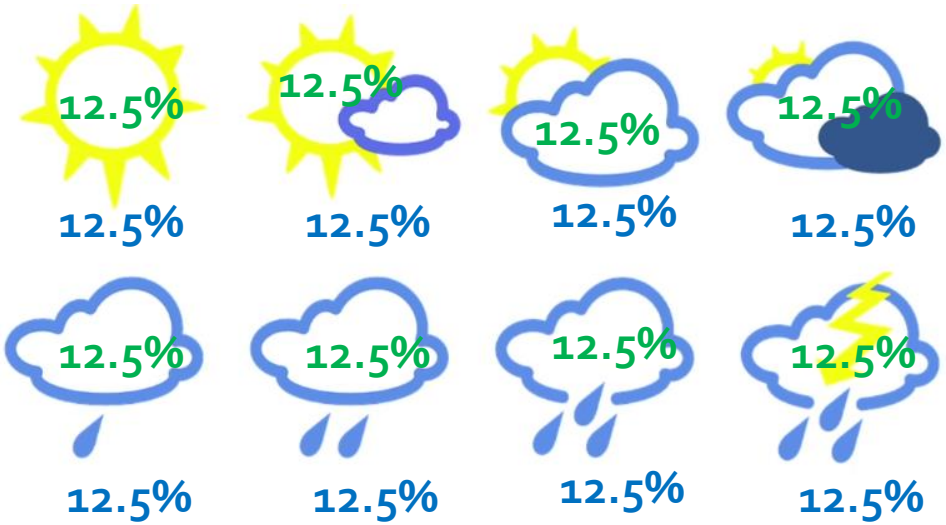
https://www.youtube.com/watch?v=ErfnhcEV1O8

Cross Entropy

Cross Entropy: Average Message Length

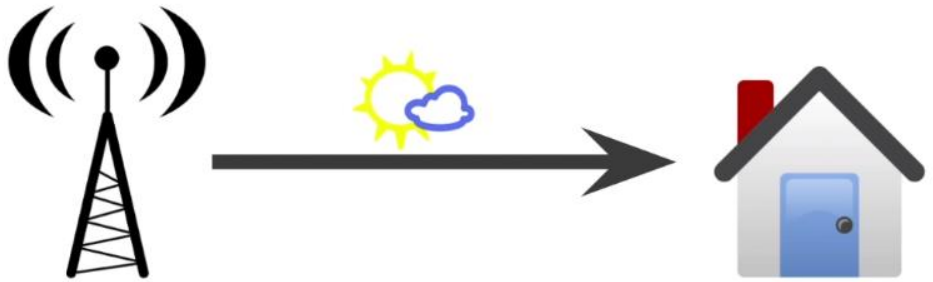
p = true distribution

q = predicted distribution



Entropy:
$H(\mathbf{p}) = -\Sigma_i\, p_i \log_2(p_i)$

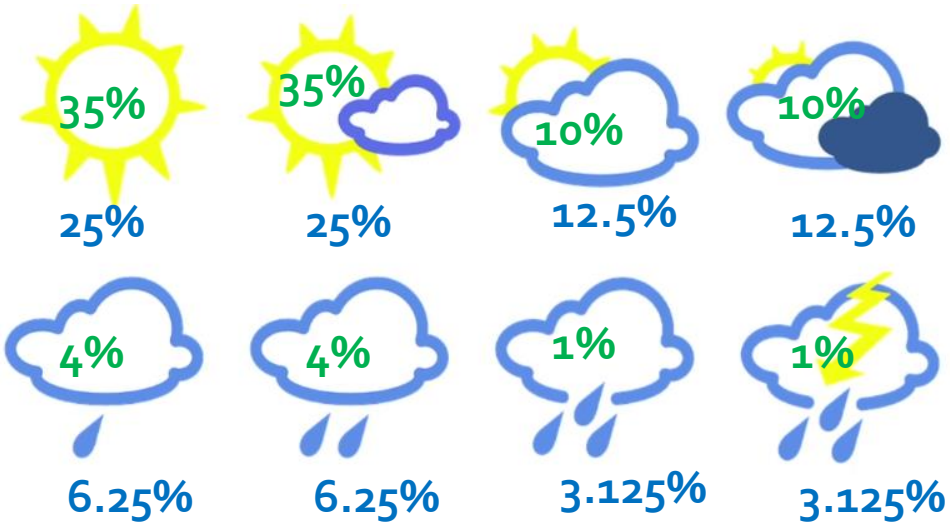
Cross-Entropy:
$H(\mathbf{p}, \mathbf{q}) = -\Sigma_i\, p_i \log_2(q_i)$



https://www.youtube.com/watch?v=ErfnhcEV1O8

Cross Entropy

Decent Weather Report

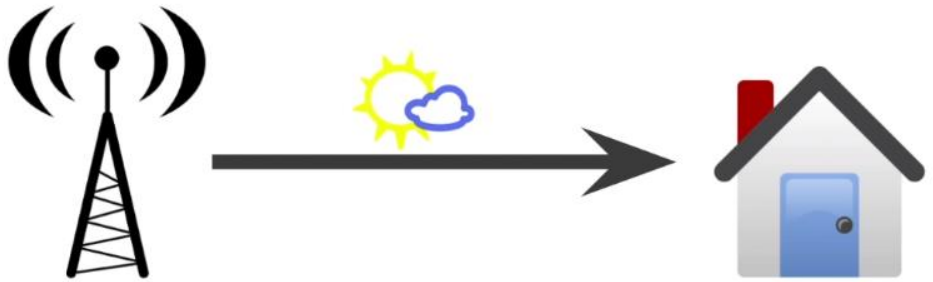


p = true distribution

q = predict distribution

Entropy:
$H(\mathbf{p}) = -\Sigma_i\, p_i \log_2(p_i)$

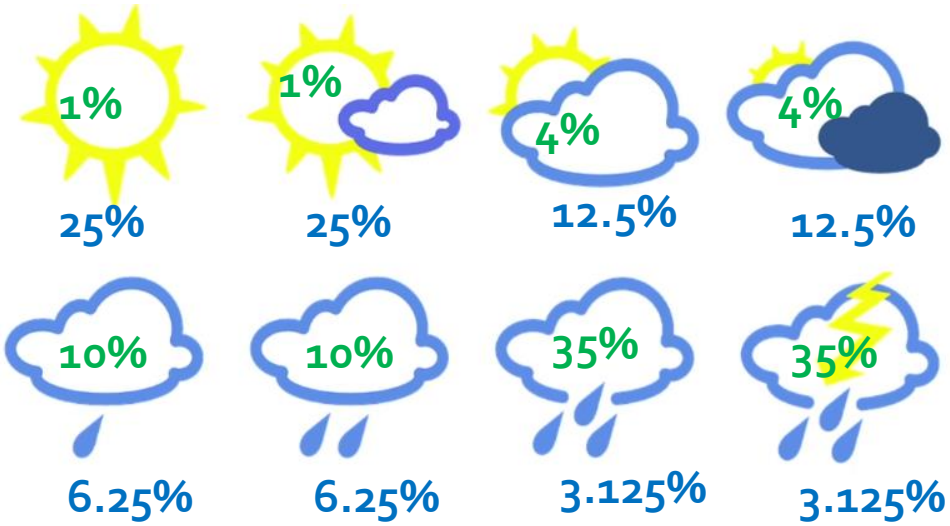
Cross-Entropy:
$H(\mathbf{p}, \mathbf{q}) = -\Sigma_i\, p_i \log_2(q_i)$



https://www.youtube.com/watch?v=ErfnhcEV1O8

Cross Entropy

Bad Weather Report

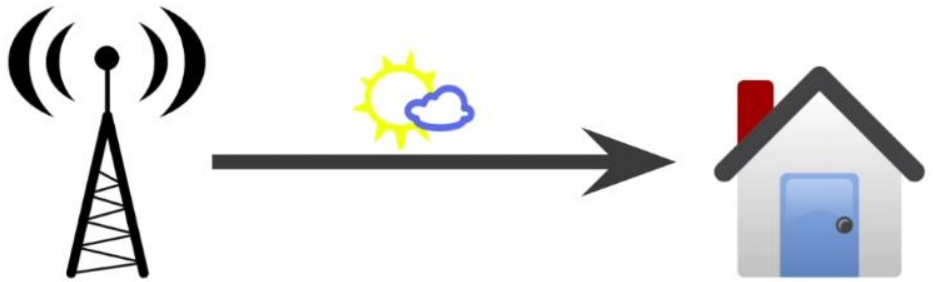


p = true distribution

q = predict distribution

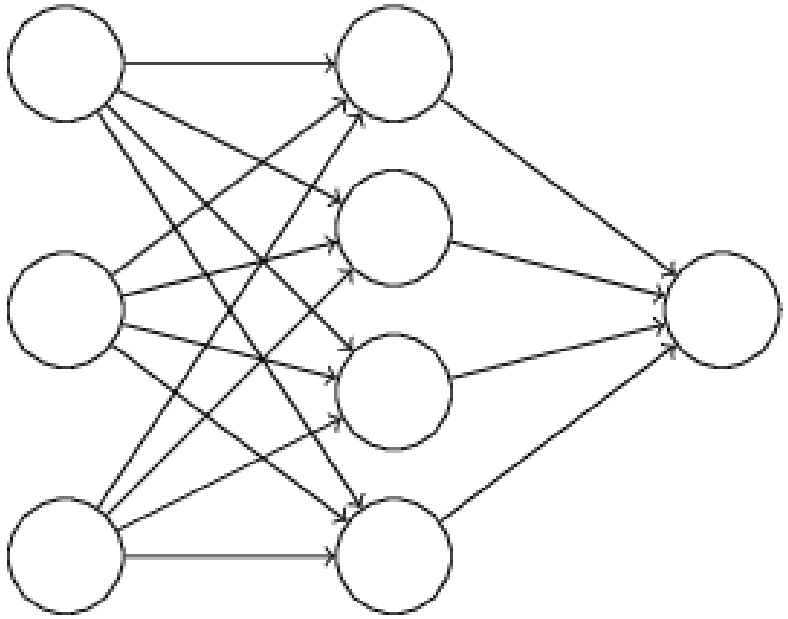
Entropy:
$H(\mathbf{p}) = -\Sigma_i\, p_i \log_2(p_i)$

Cross-Entropy:
$H(\mathbf{p}, \mathbf{q}) = -\Sigma_i\, p_i \log_2(q_i)$



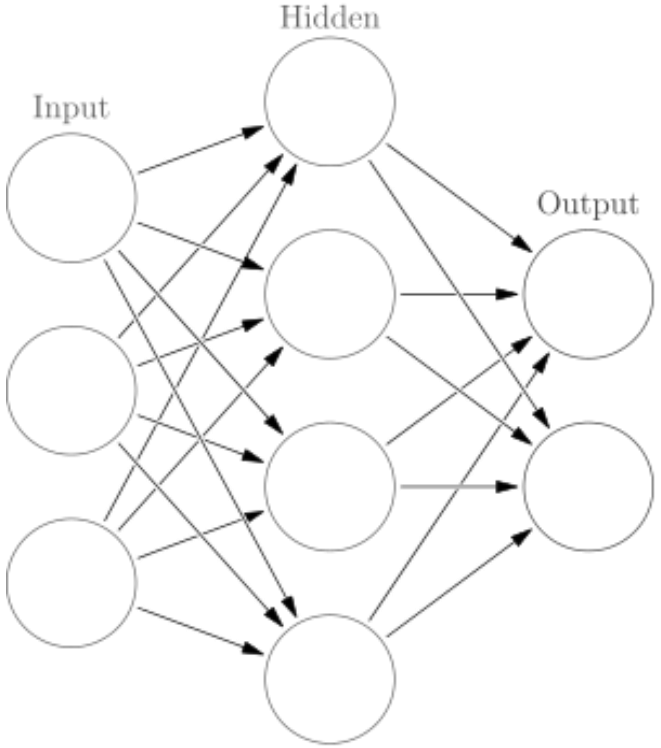
https://www.youtube.com/watch?v=ErfnhcEV1O8

How to Code the Output



http://neuralnetworksanddeeplearning.com/chap1.html

1, 2

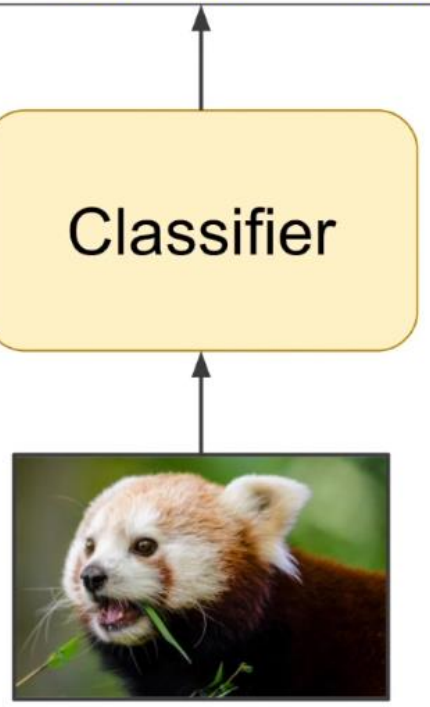


https://en.wikipedia.org/wiki/Artificial_neural_network

[1 0],[0 1]

Classification Example

	Cat	Dog	Fox	Cow	Red Panda	Bear	Dolphin
True distribution:	0%	0%	0%	0%	100%	0%	0%
Predicted distribution:	2%	30%	45%	0%	25%	5%	0%



Cross-Entropy:
$H(\mathbf{p}, \mathbf{q}) = -\Sigma_i \, p_i \log_2(q_i)$

https://www.youtube.com/watch?v=ErfnhcEV1O8

Classification Example

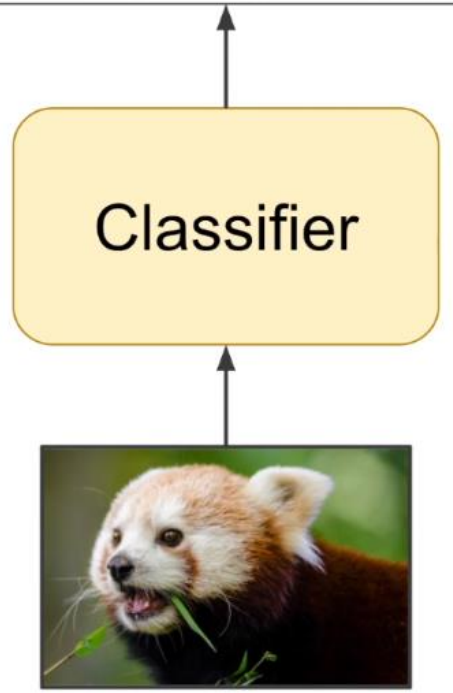
	Cat	Dog	Fox	Cow	Red Panda	Bear	Dolphin
True distribution:	0%	0%	0%	0%	100%	0%	0%
Predicted distribution:	2%	30%	45%	0%	25%	5%	0%

Cross-Entropy Loss:

$H(\mathbf{p}, \mathbf{q}) = -\Sigma_i\, p_i \log(q_i)$

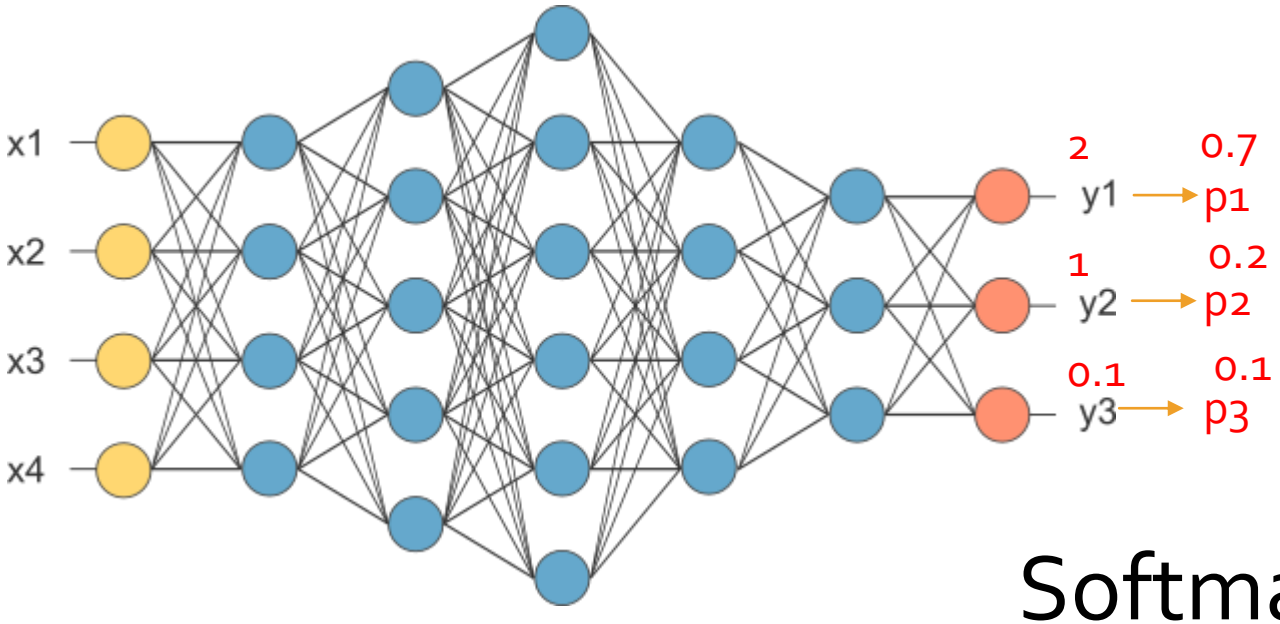
$= -\log(0.25) = 1.386$

$\log_2(x) = \log(x) / \log(2)$



https://www.youtube.com/watch?v=ErfnhcEV1O8

One Step Need

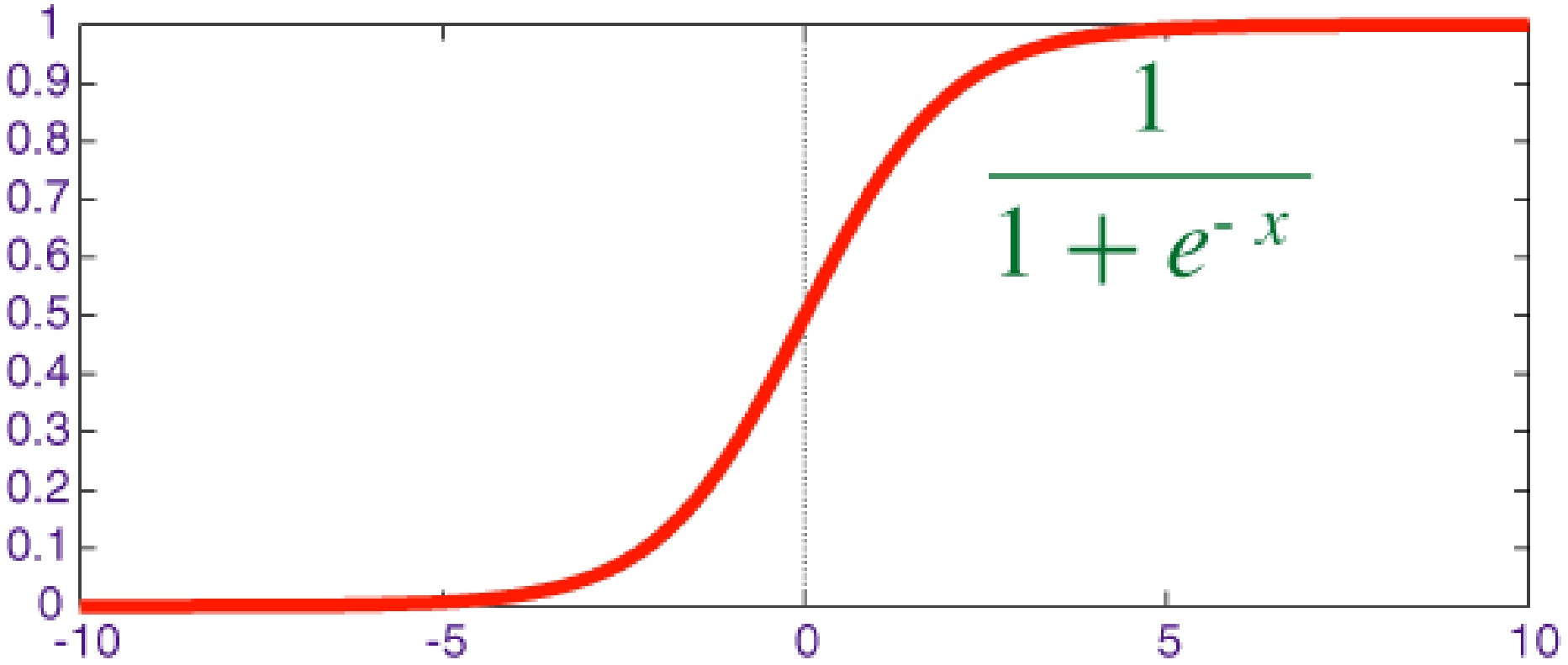


computed p1	computed p2	computed p3	targets	targets	targets	correct?
0.3	0.3	0.4	0	0	1	yes
0.3	0.4	0.3	0	1	0	yes
0.1	0.2	0.7	1	0	0	no

Softmax

https://www.quora.com/How-does-deep-learning-work-and-how-is-it-different-from-normal-neural-networks-applied-with-SVM-How-does-one-go-about-starting-to-understand-them-papers-blogs-articles

sigmoid

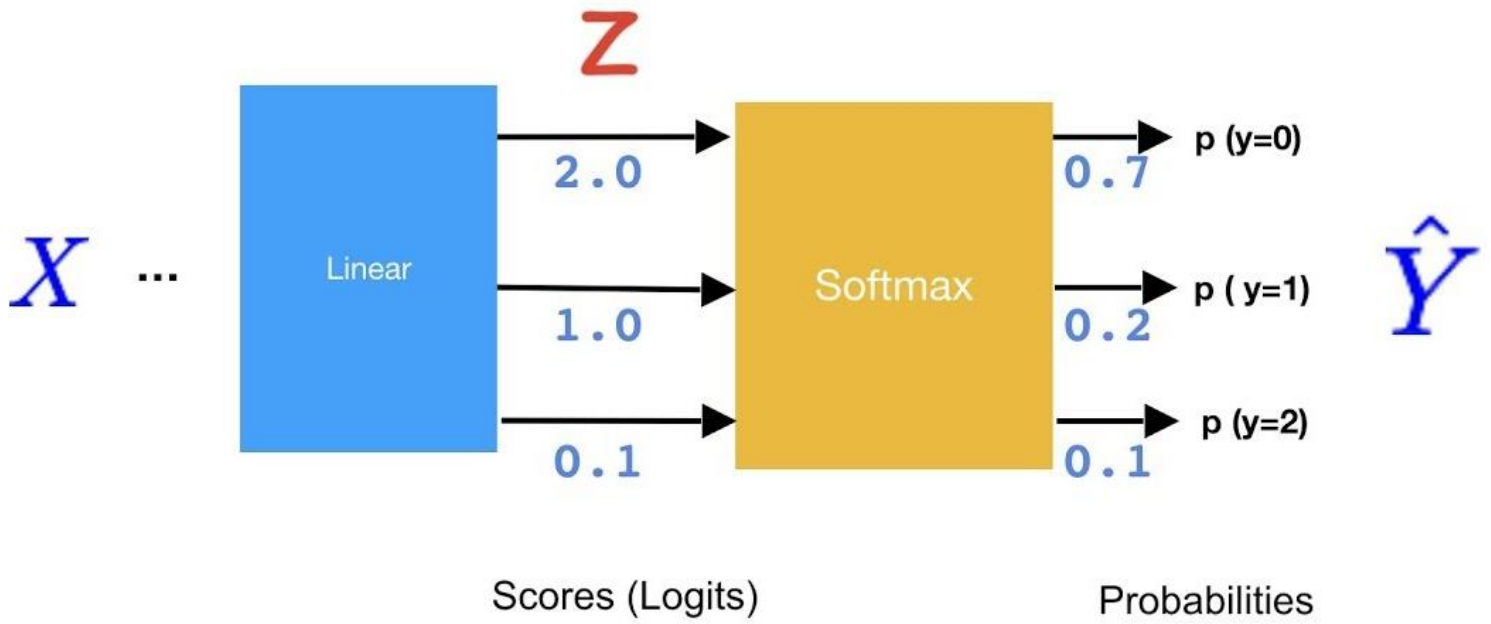


https://www.kaggle.com/vinay6666/sigmoid-function-for-logisticregression-neuralnets

Softmax

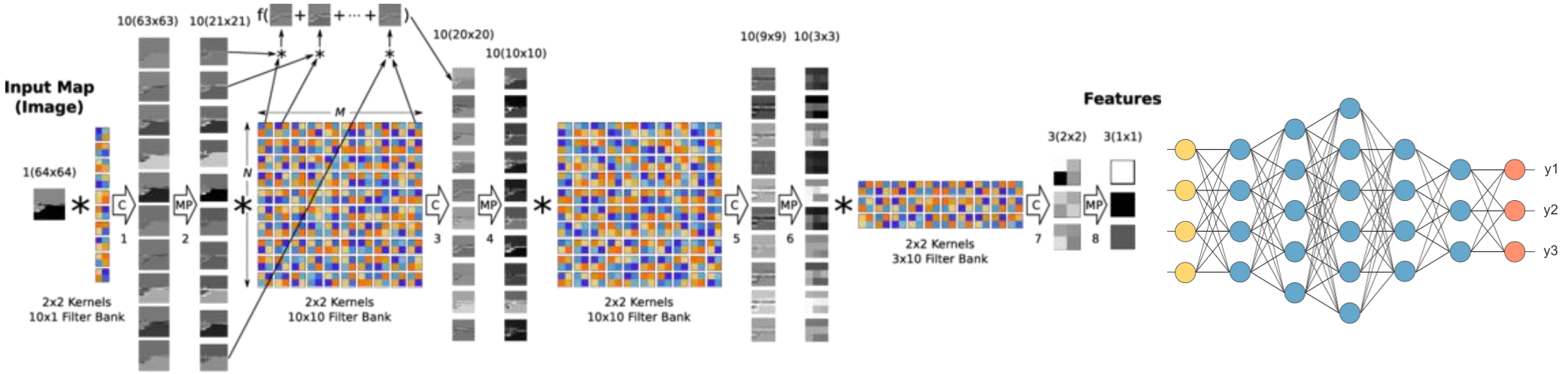
Meet Softmax

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$



https://www.youtube.com/watch?v=lvNdl7yg4Pg

Deep Convolutional Network Classification



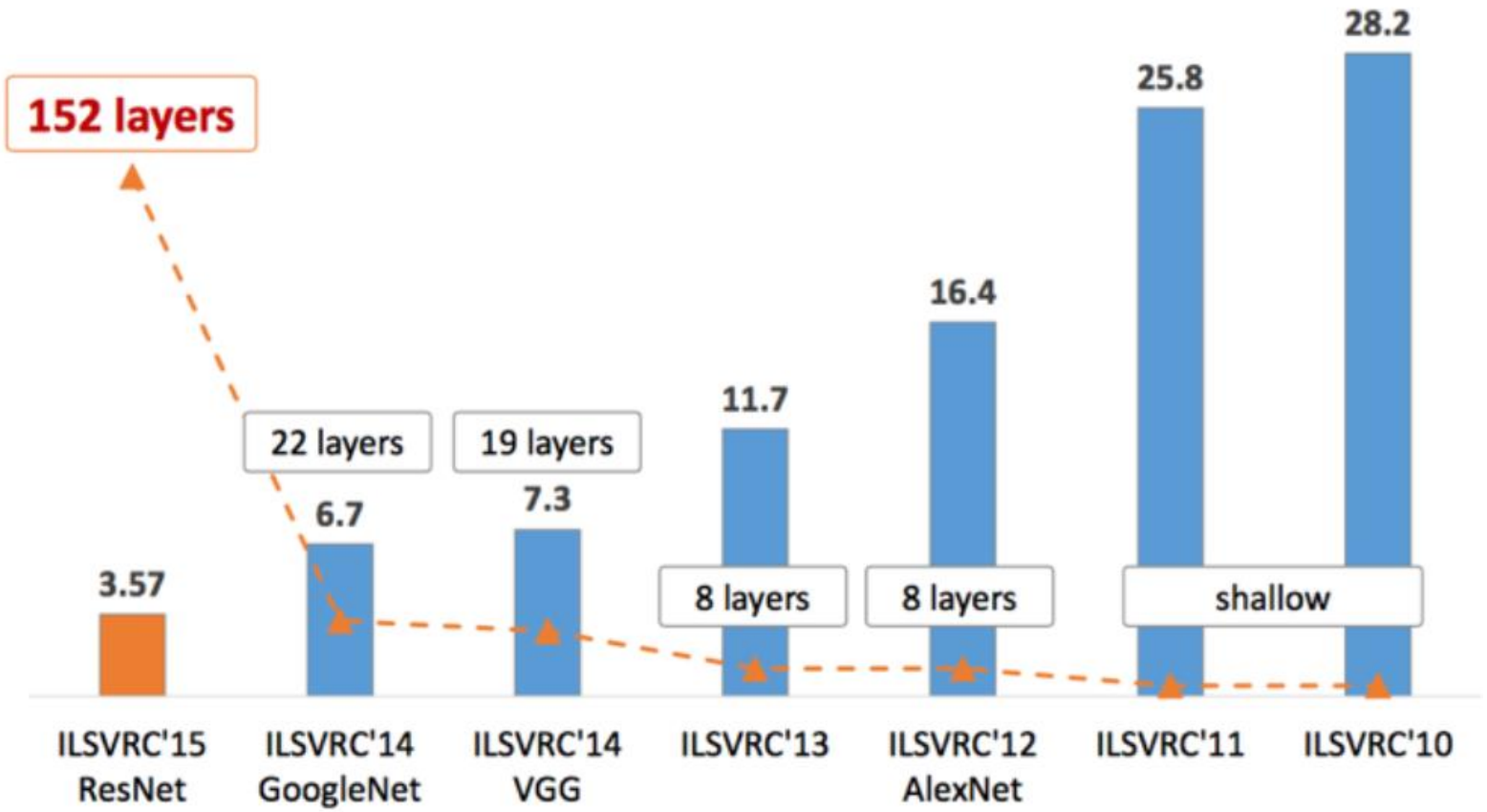
Cross Entropy

Convolutional Layer

Fully Connected Layer (Dense Layer)

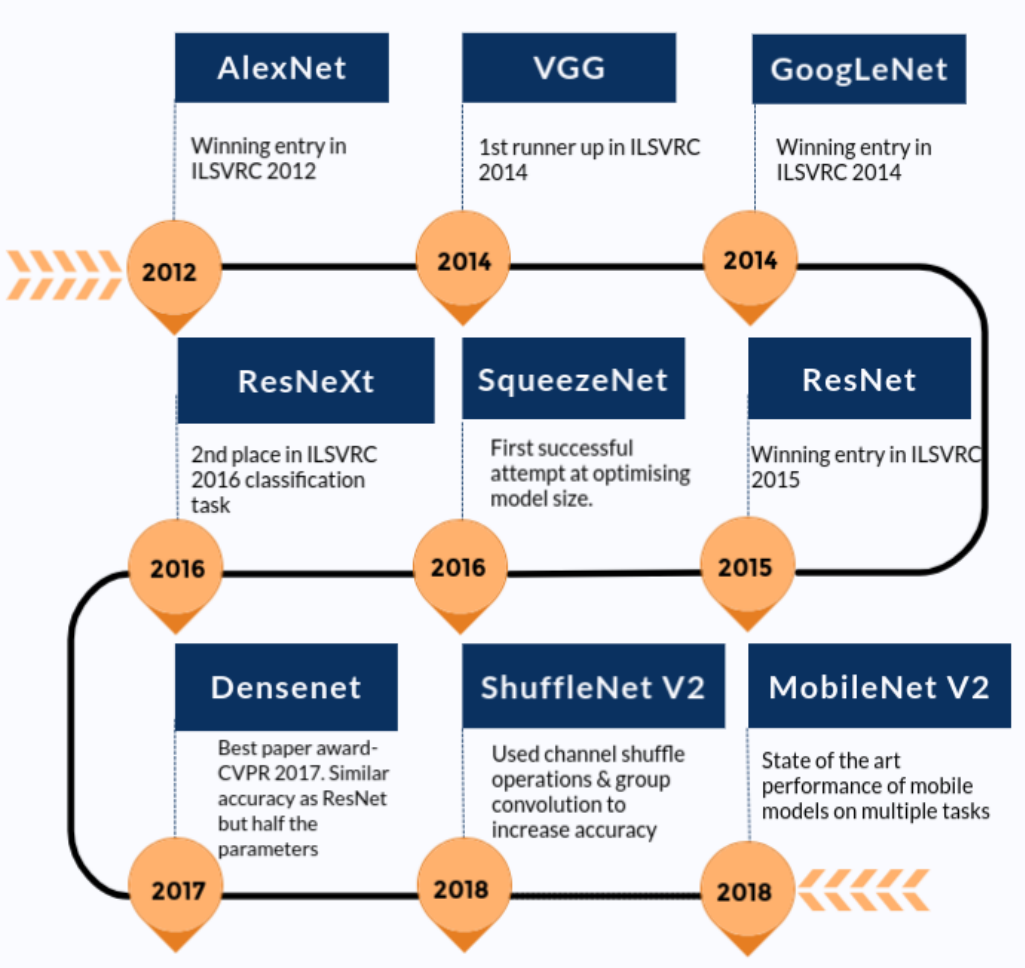
https://www.researchgate.net/figure/Max-Pooling-Convolutional-Neural-Network-MPCNN-with-8-layers-alternating-between_fig3_266656356

Winners



https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

State-of-the-art



AlexNet

VGG

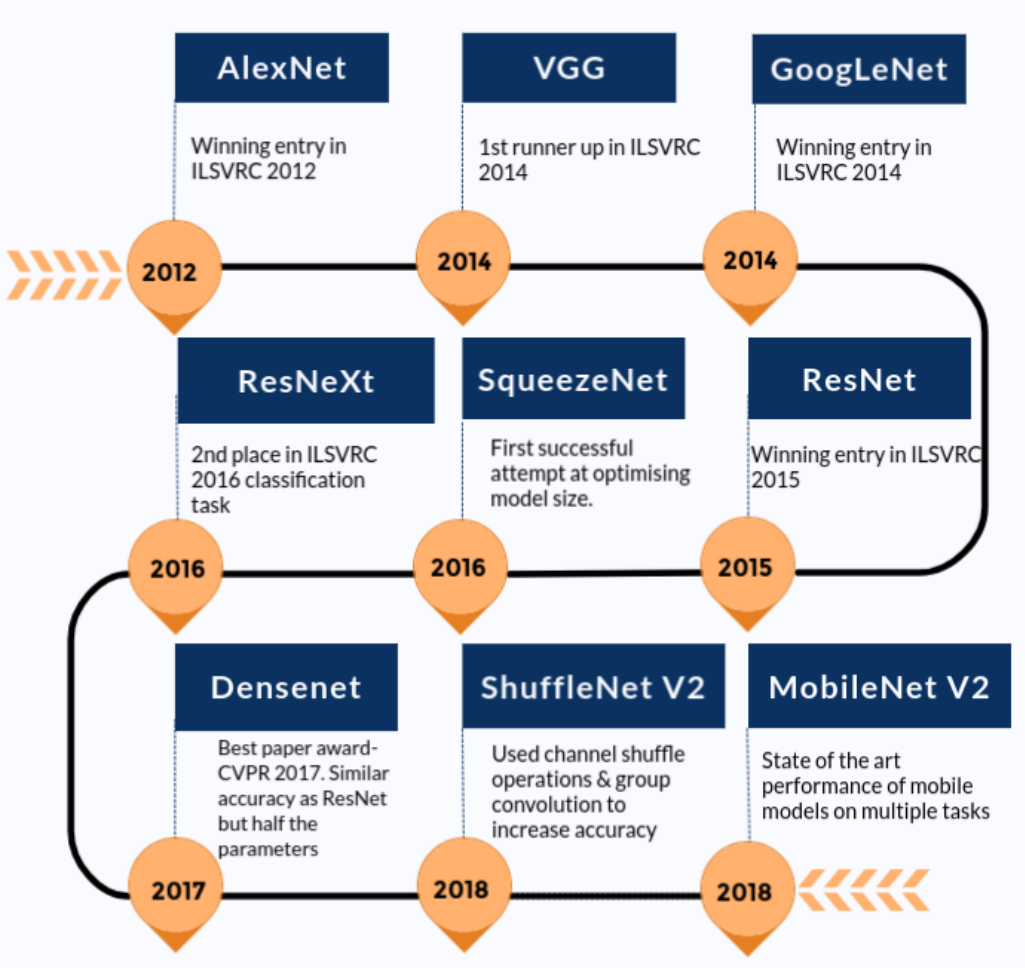
GoogleNet

ResNet

DenseNet

https://www.learnopencv.com/pytorch-for-beginners-image-classification-using-pre-trained-models/

State-of-the-art



AlexNet

VGG

GoogleNet

ResNet

DenseNet

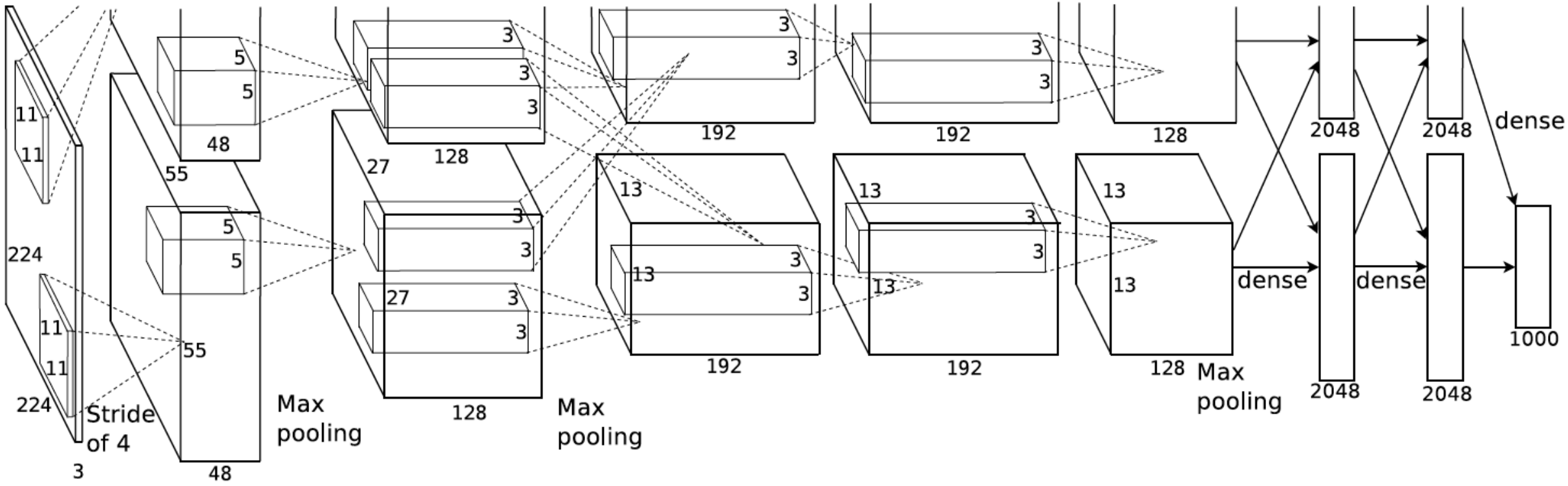
https://www.learnopencv.com/pytorch-for-beginners-image-classification-using-pre-trained-models/

AlexNet

[PDF] ImageNet Classification with Deep Convolutional ... - NIPS Proceedings

https://papers.nips.cc/.../4824-imagenet-classification-with-deep-convolutional-neural-...

by A Krizhevsky - 2012 - Cited by 27351 - Related articles



AlexNet

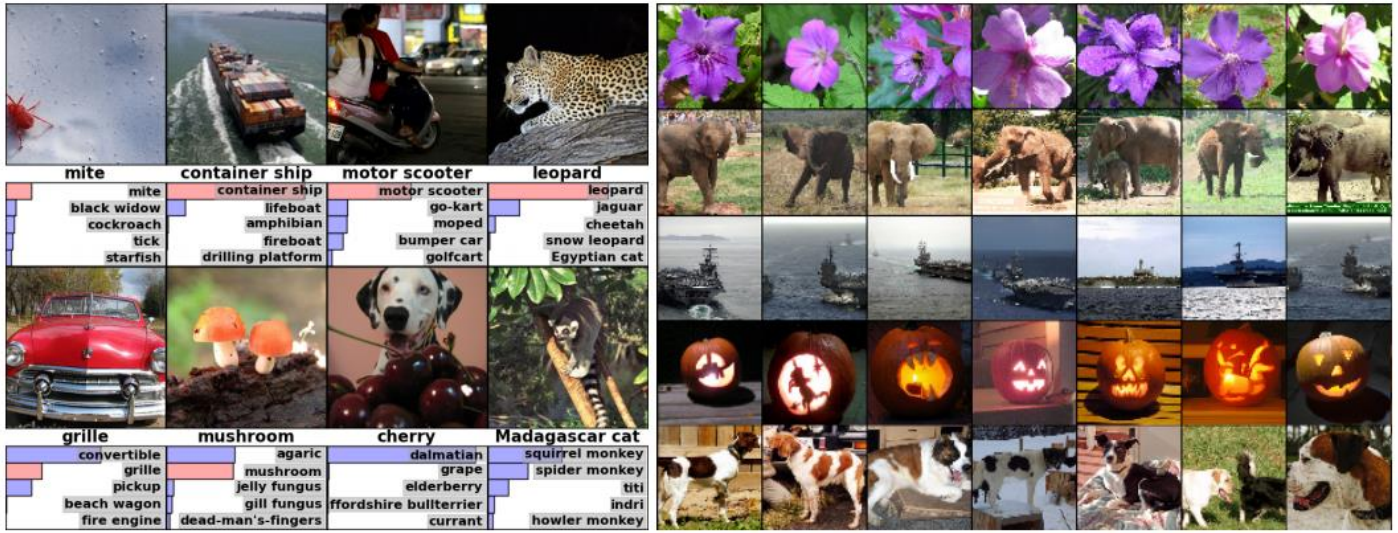


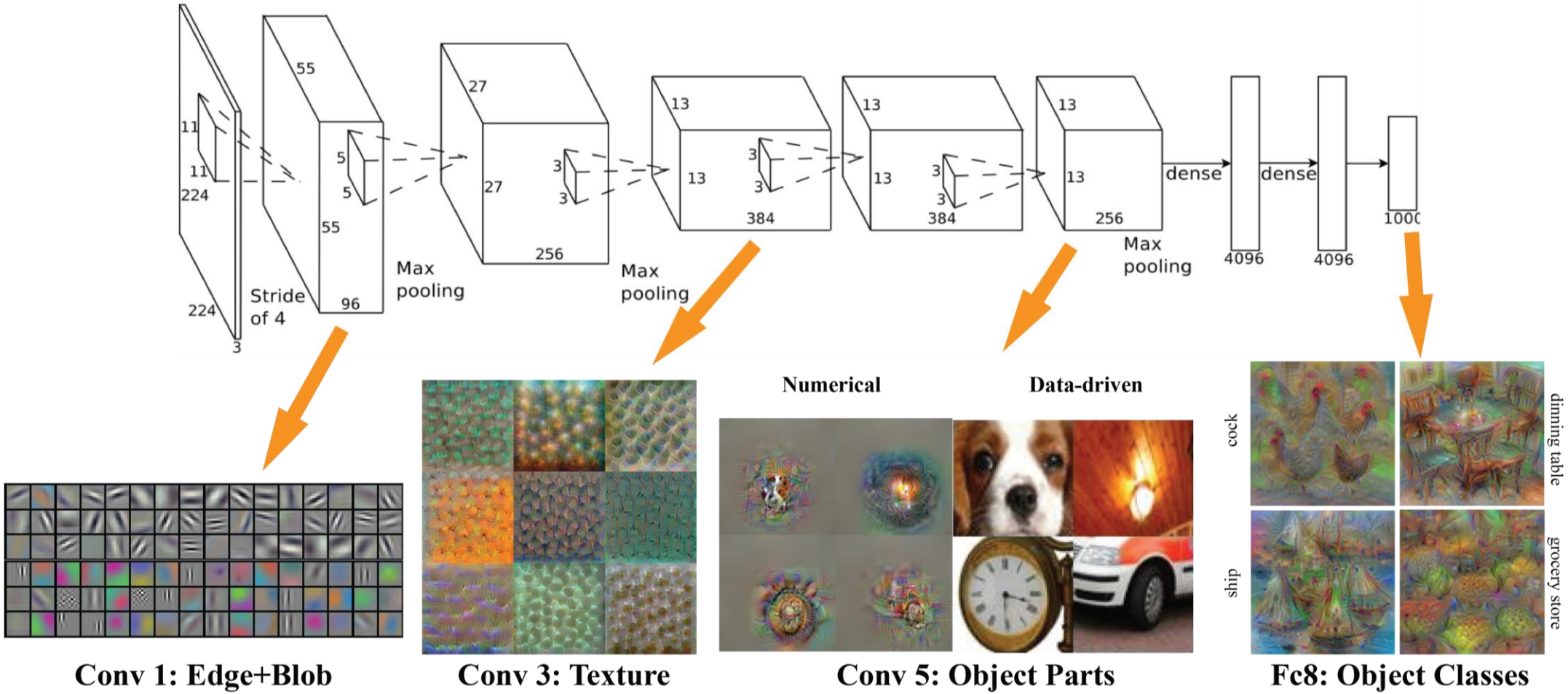
Figure 4: **(Left)** Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). **(Right)** Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.

Krizhevsky et al. 2012

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In *italics* are best results achieved by others. Models with an asterisk* were "pre-trained" to classify the entire ImageNet 2011 Fall release. See Section 6 for details.

Full AlexNet



https://www.cc.gatech.edu/~hays/compvision/proj6/

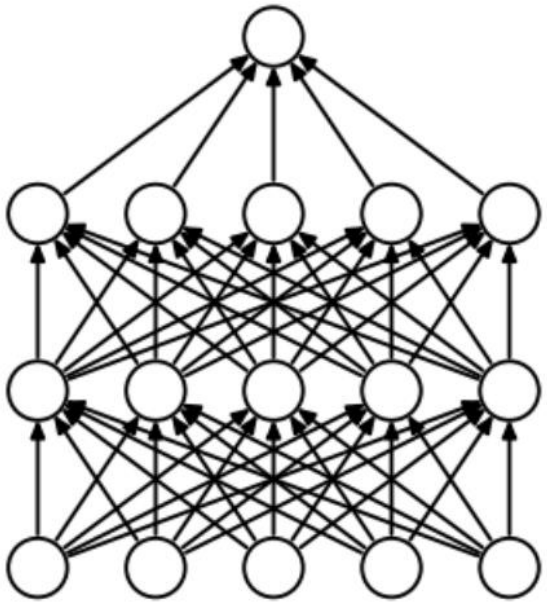
Dropout

Dropout: A Simple Way to Prevent Neural Networks from Overfitting

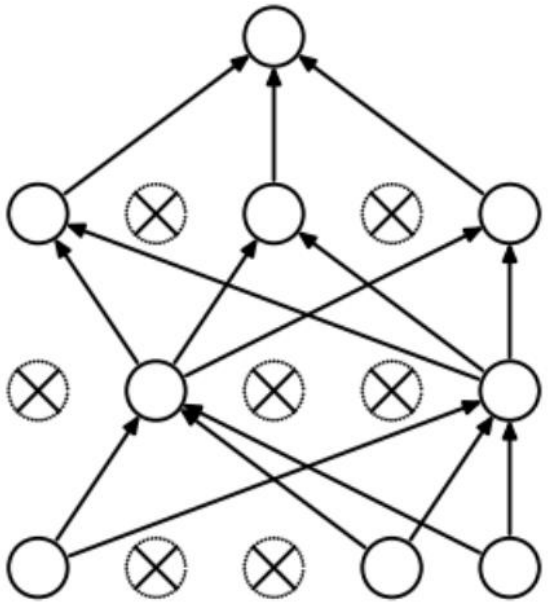
jmlr.org/papers/v15/srivastava14a.html ▾

by N Srivastava - 2014 - Cited by 7597 - Related articles

Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training.

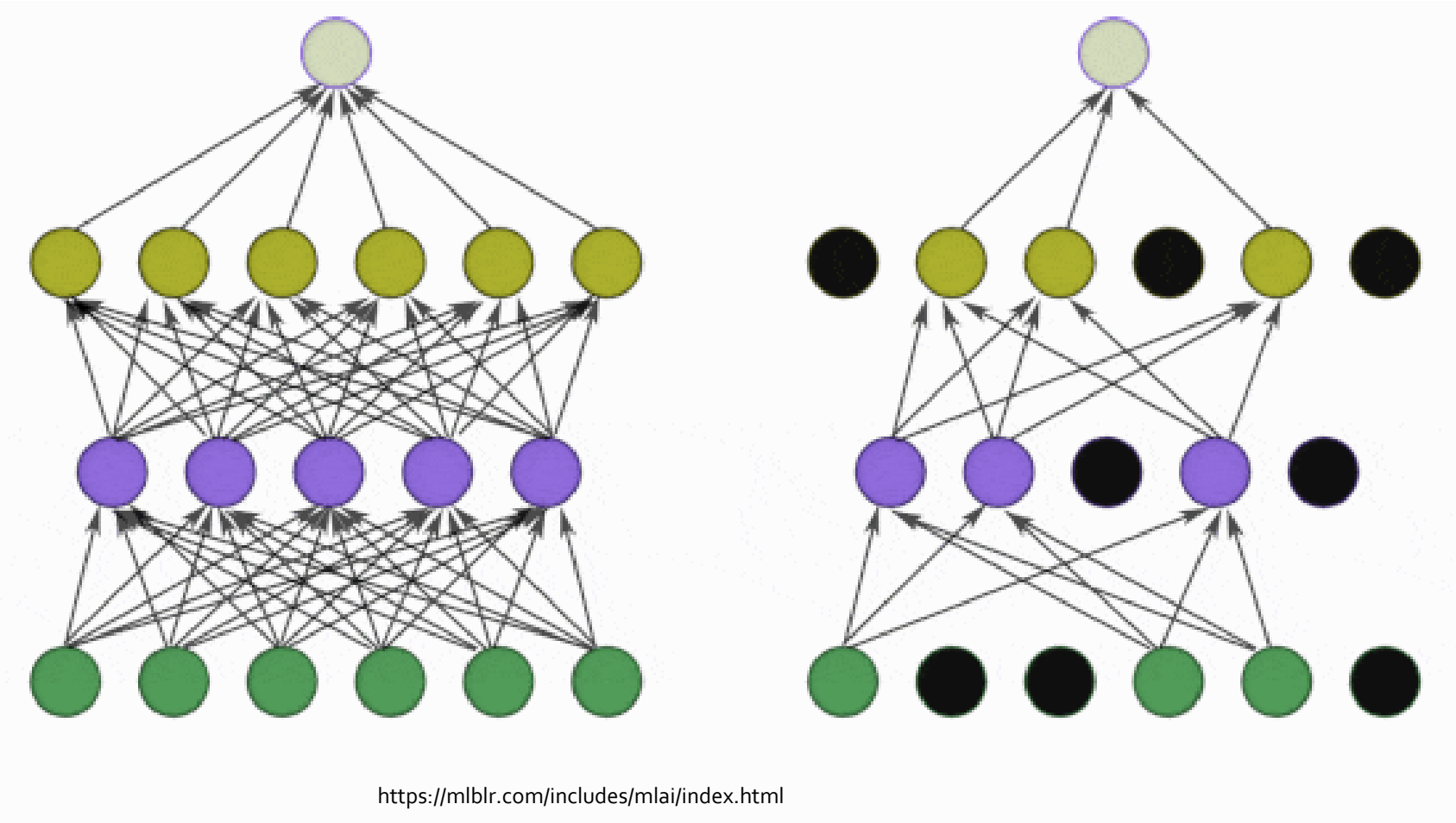


(a) Standard Neural Net



(b) After applying dropout.

Dropout



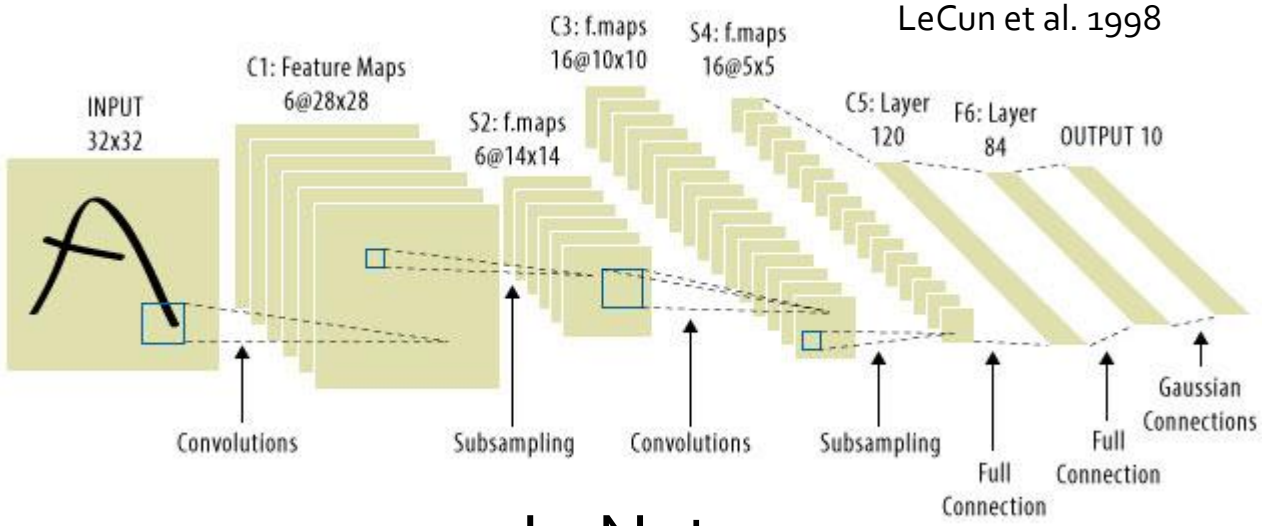
https://mlblr.com/includes/mlai/index.html

Data Augmentation

https://www.learnopencv.com/understanding-alexnet/

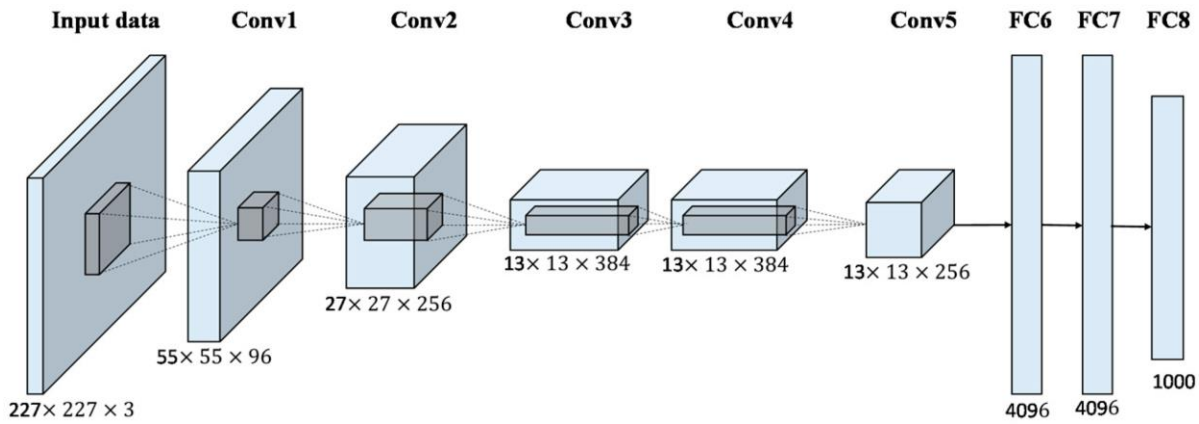
AlexNet vs LeNet-5

LeCun et al. 1998



LeNet-5

Krizhevsky et al. 2012

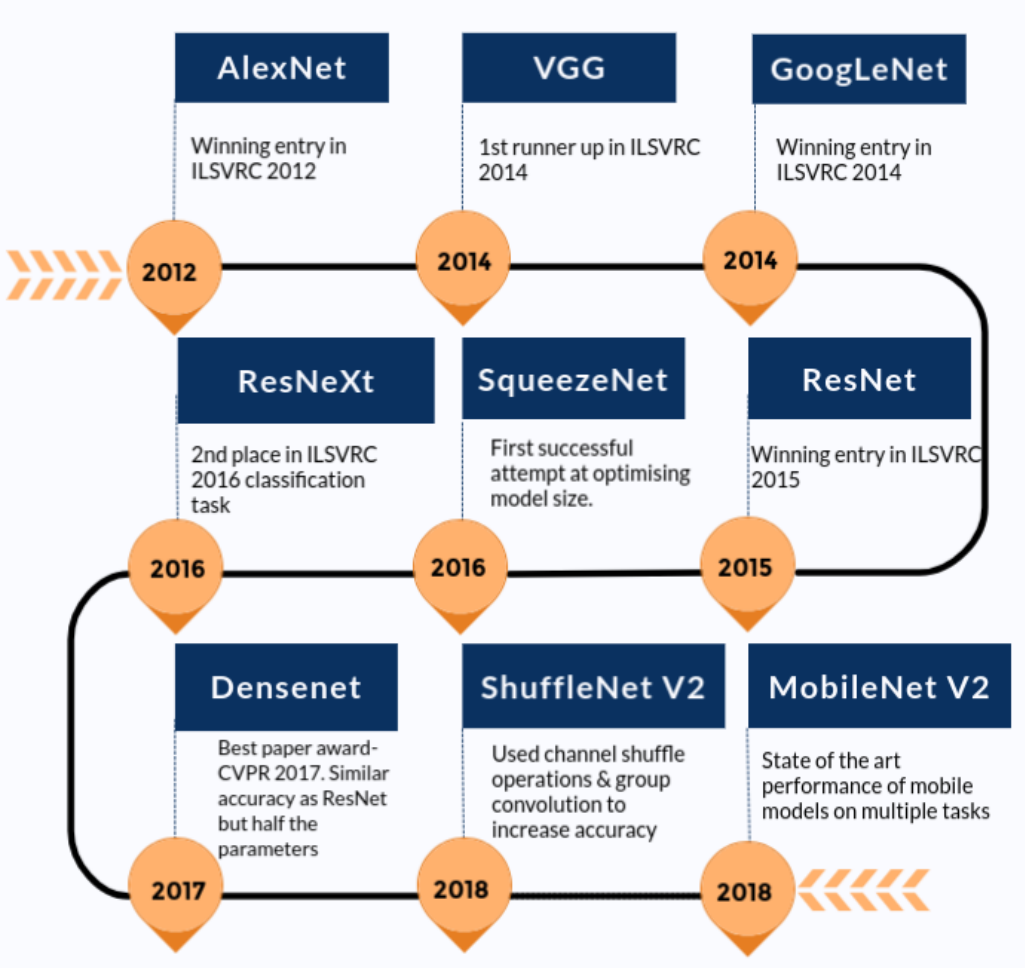


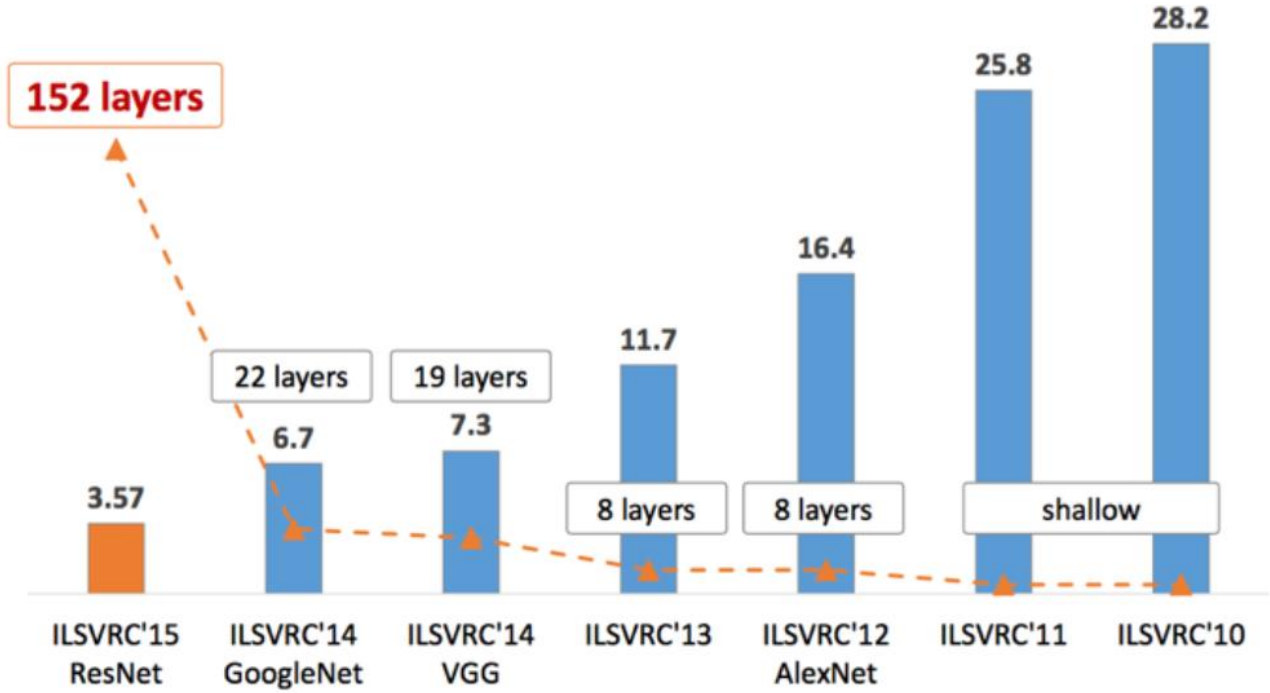
AlexNet

The network had a very similar architecture as LeNet by Yann LeCun et al but was deeper, with more filters per layer, and with stacked convolutional layers. It consisted 11x11, 5x5,3x3, convolutions, max pooling, dropout, data augmentation, ReLU activations, SGD with momentum. It attached ReLU activations after every convolutional and fully-connected layer. AlexNet was trained for 6 days simultaneously on two Nvidia Geforce GTX 580 GPUs which is the reason for why their network is split into two pipelines.

https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

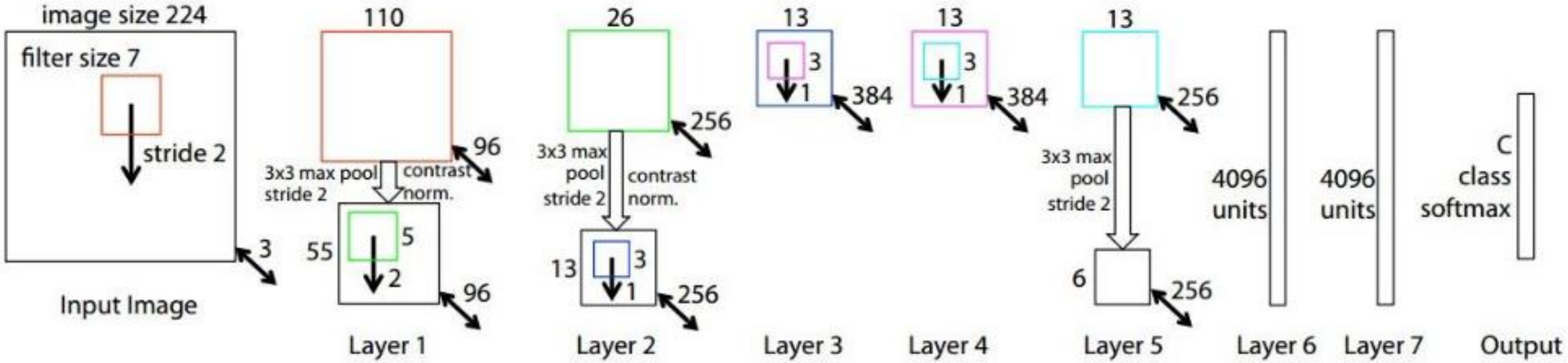
State-of-the-art





https://www.learnopencv.com/pytorch-for-beginners-image-classification-using-pre-trained-models/

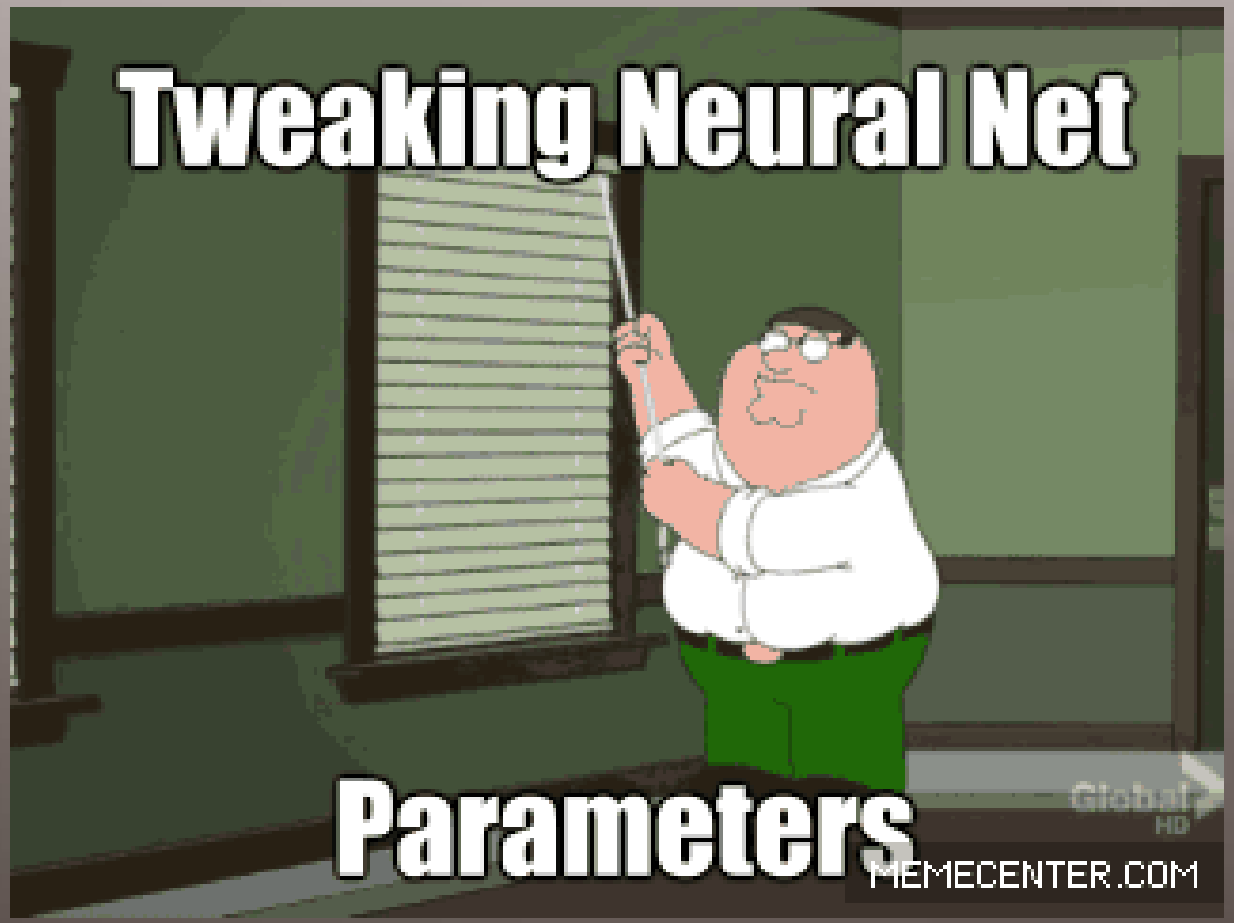
ZFNet (2013)



Not surprisingly, the ILSVRC 2013 winner was also a CNN which became known as ZFNet. It achieved a top-5 error rate of 14.8% which is now already half of the prior mentioned non-neural error rate. It was mostly an achievement by tweaking the hyper-parameters of AlexNet while maintaining the same structure with additional Deep Learning elements as discussed earlier in this essay.

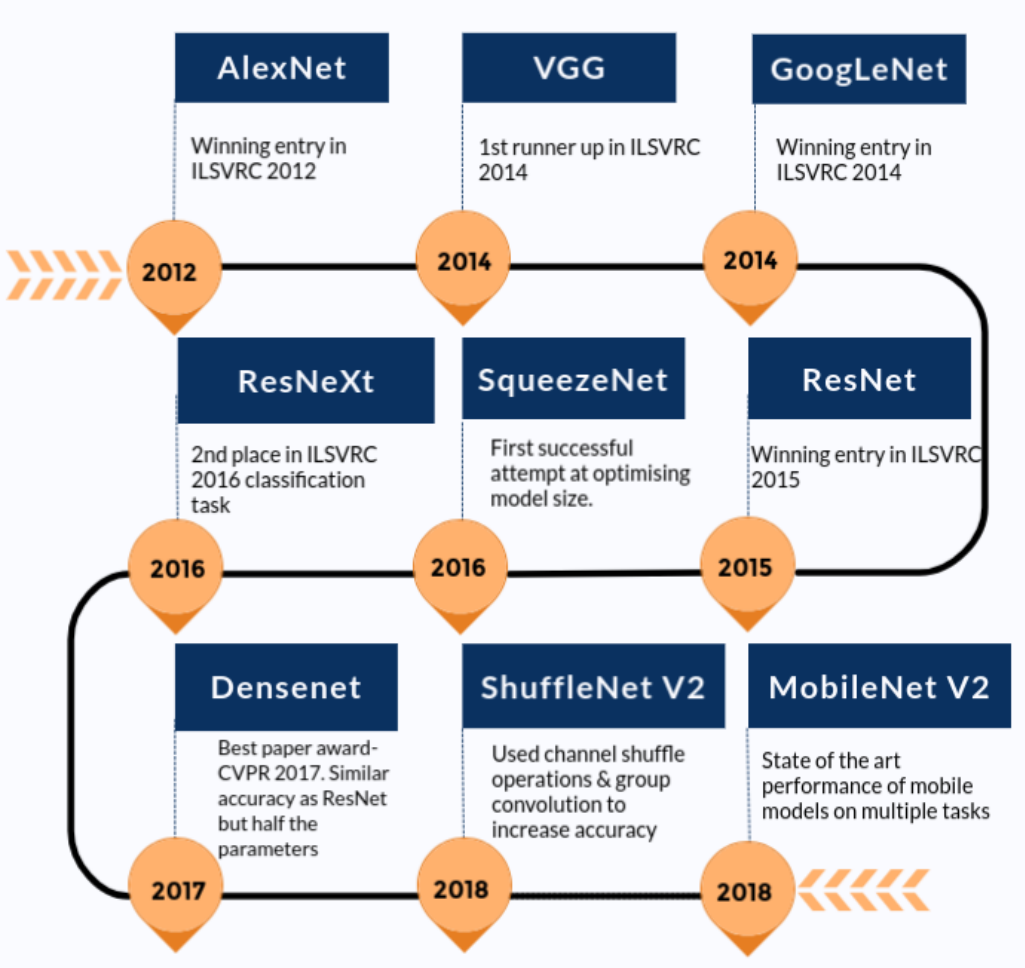
https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

ZFNet : That's it



http://bbabenko.tumblr.com/post/83319141207/convolutional-learnings-things-i-learned-by

State-of-the-art



AlexNet

VGG

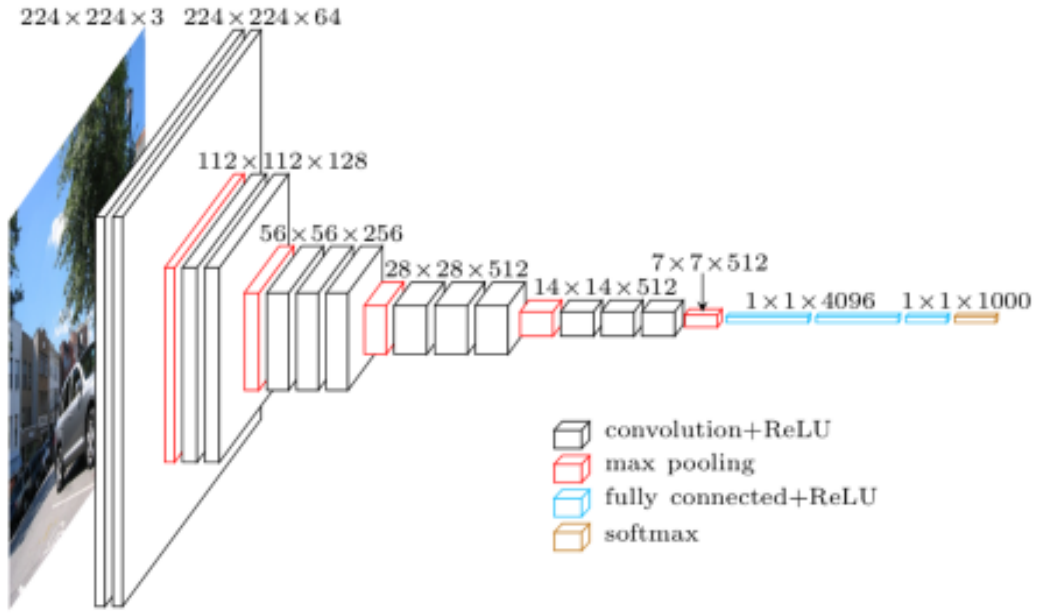
GoogleNet

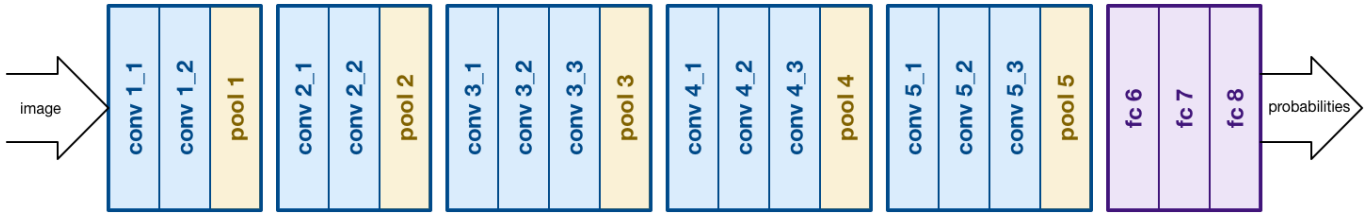
ResNet

DenseNet

https://www.learnopencv.com/pytorch-for-beginners-image-classification-using-pre-trained-models/

VGGNet (2014)

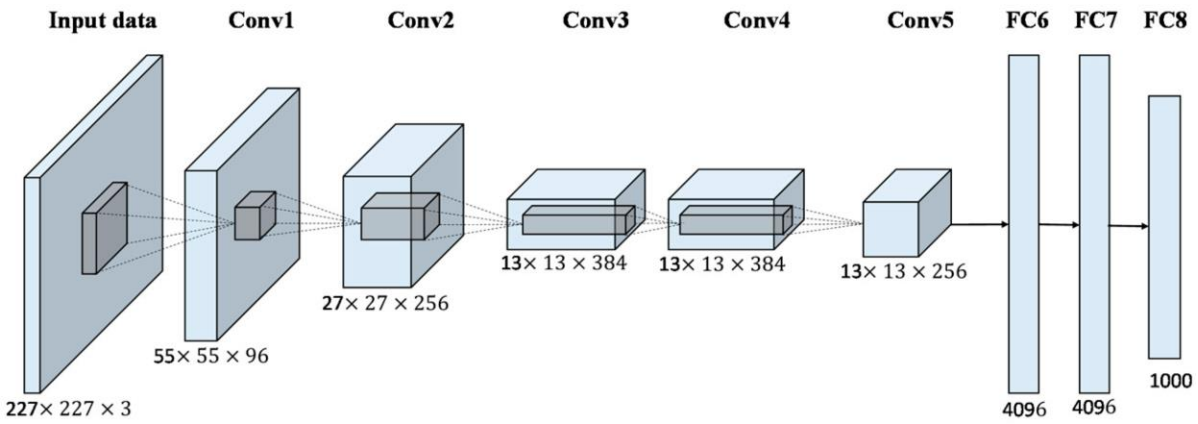




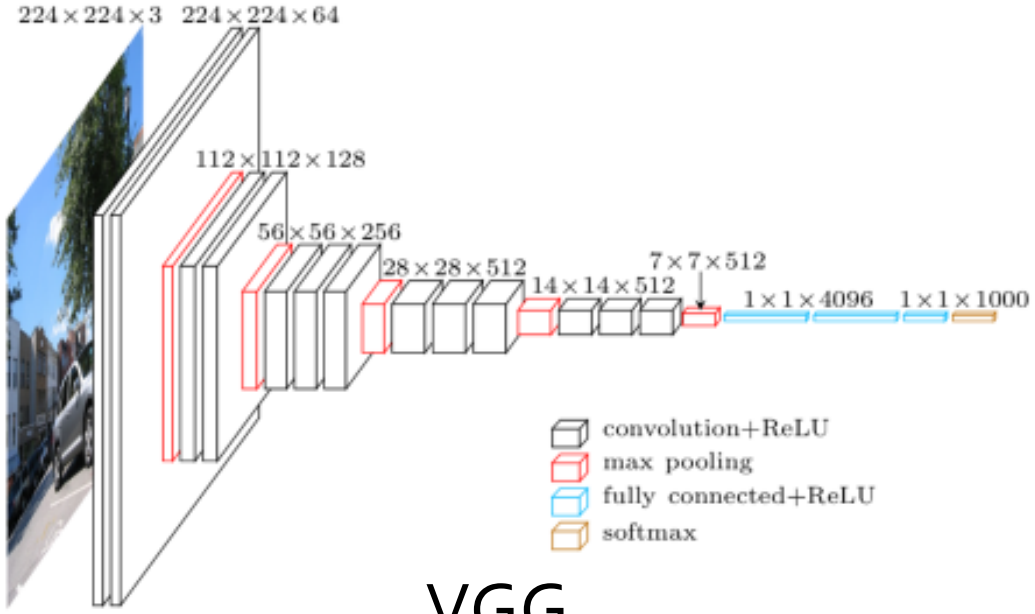
https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/
https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

AlexNet vs VGG

Krizhevsky et al. 2012



AlexNet

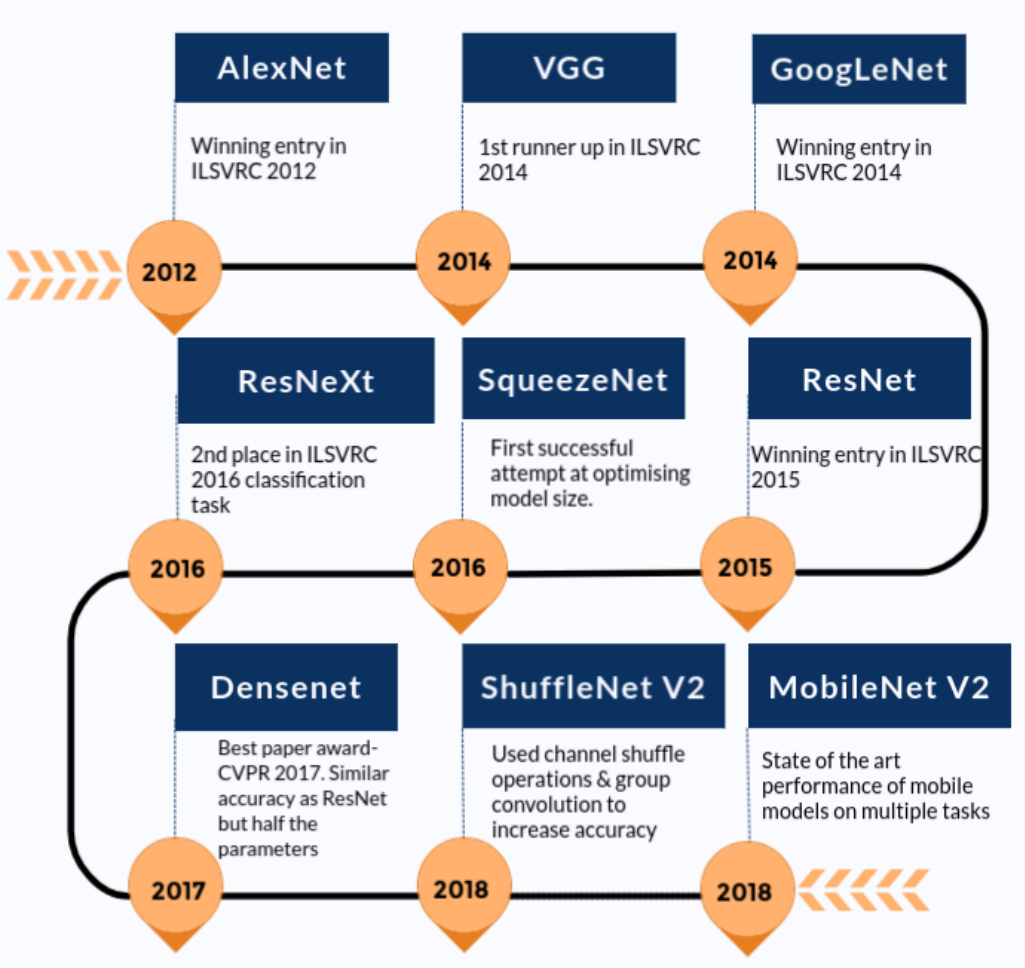


VGG

The idea behind having fixed size kernels is that all the variable size convolutional kernels used in Alexnet (11x11, 5x5, 3x3) can be replicated by making use of multiple 3x3 kernels as building blocks. The replication is in terms of the receptive field covered by the kernels. This reduces the number of trainable variables by 44.9% (62.8%). Reduced number of trainable variables means faster learning and more robust to over-fitting. max poolings should be placed after each 2 convolutions and the number of filters should be doubled after each max-pooling.

https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

State-of-the-art



AlexNet

VGG

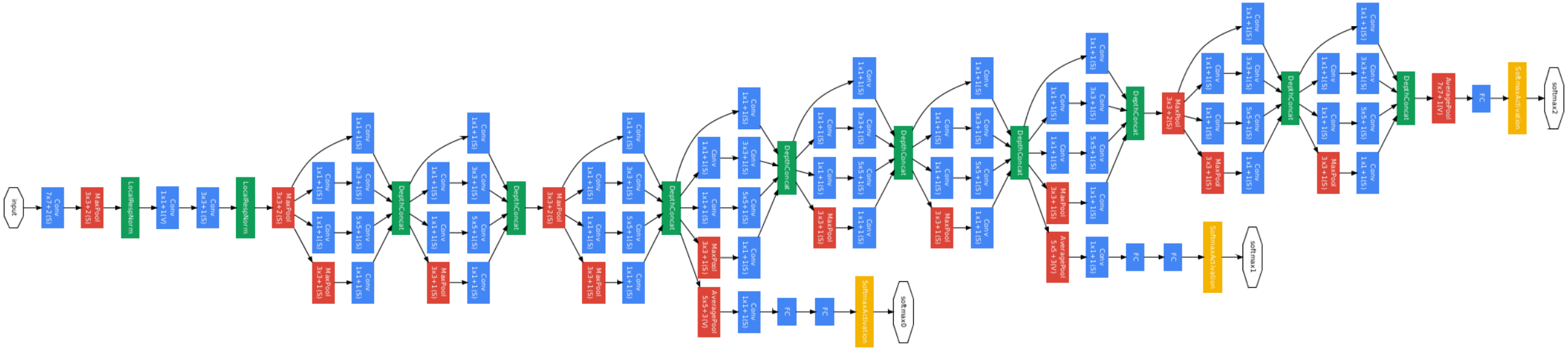
GoogleNet

ResNet

DenseNet

https://www.learnopencv.com/pytorch-for-beginners-image-classification-using-pre-trained-models/

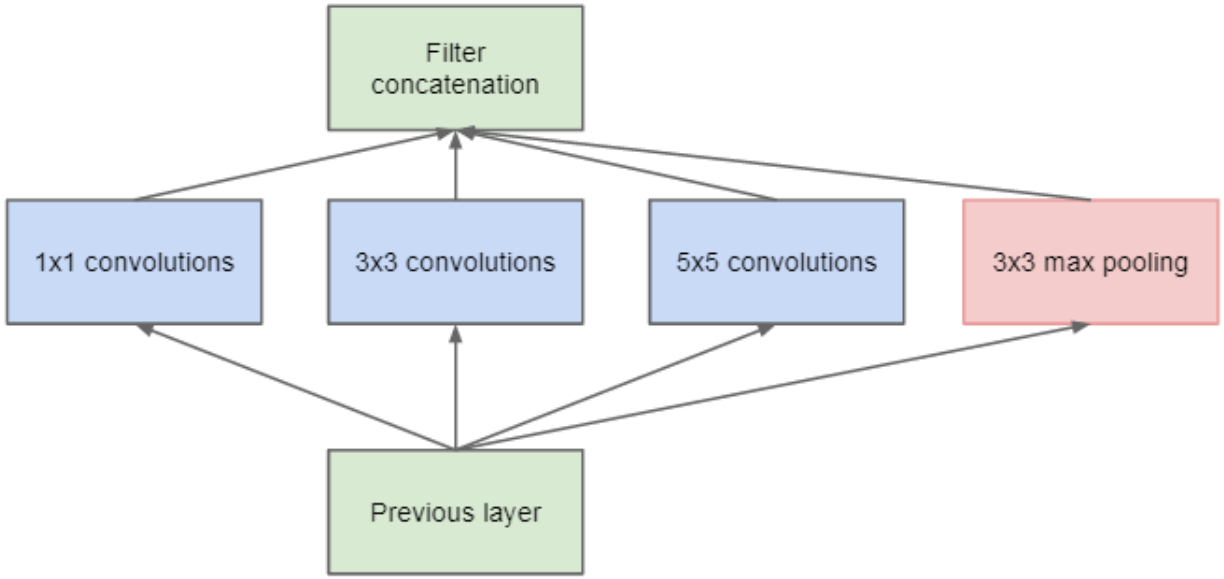
GoogleNet/Inception(2014)



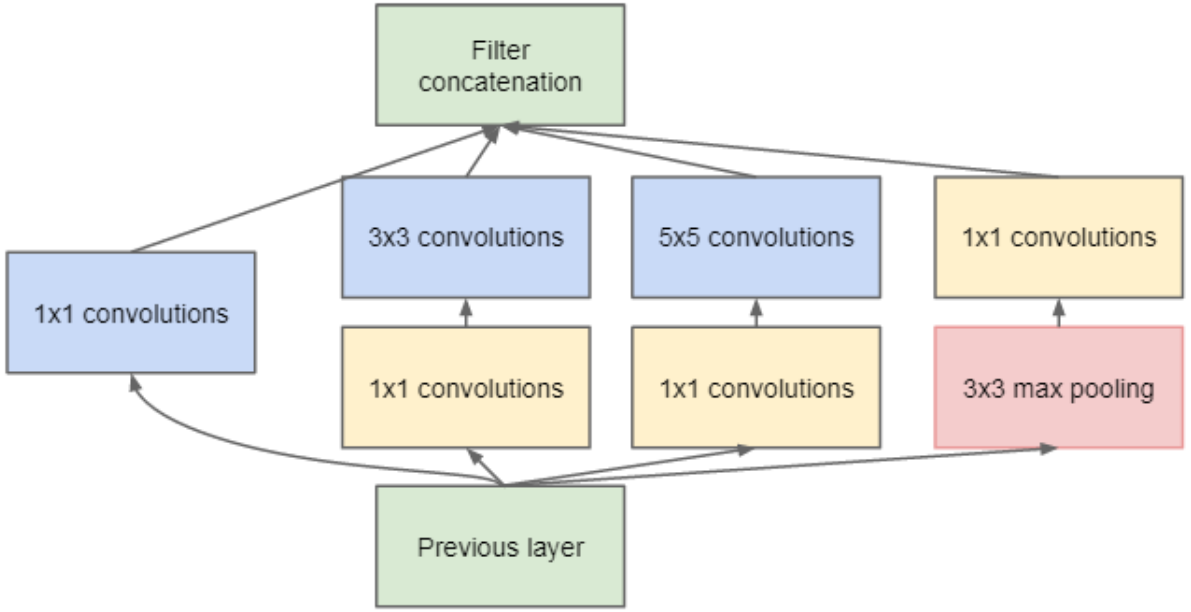
The winner of the ILSVRC 2014 competition was GoogleNet(a.k.a. Inception V1) from Google. It achieved a top-5 error rate of 6.67%! This was very close to human level performance which the organisers of the challenge were now forced to evaluate. As it turns out, this was actually rather hard to do and required some human training in order to beat GoogLeNets accuracy. After a few days of training, the human expert (Andrej Karpathy) was able to achieve a top-5 error rate of 5.1%(single model) and 3.6%(ensemble). The network used a CNN inspired by LeNet but implemented a novel element which is dubbed an inception module. It used batch normalization, image distortions and RMSprop. This module is based on several very small convolutions in order to drastically reduce the number of parameters. Their architecture consisted of a 22 layer deep CNN but reduced the number of parameters from 60 million (AlexNet) to 4 million.

https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

Inception



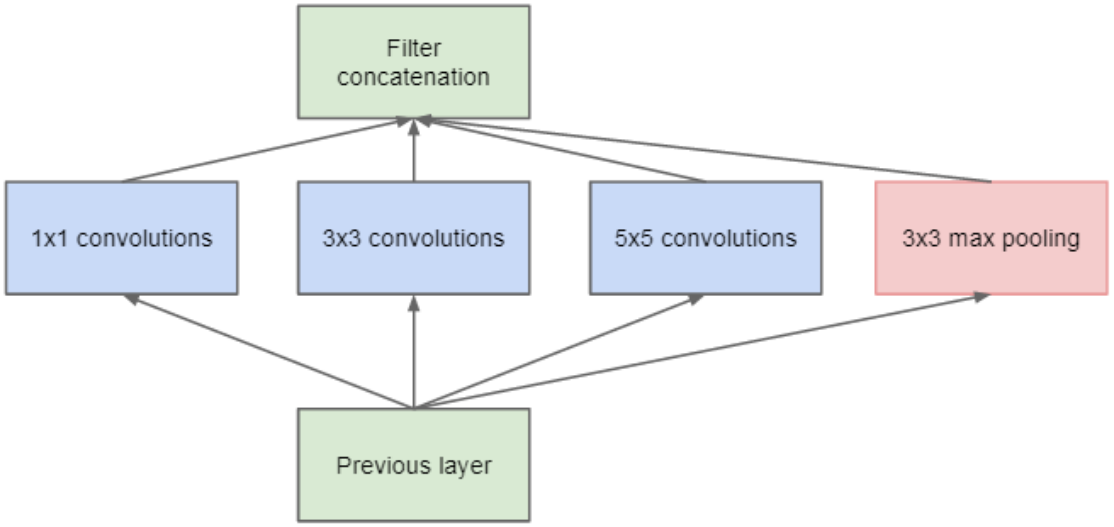
(a) Inception module, naïve version



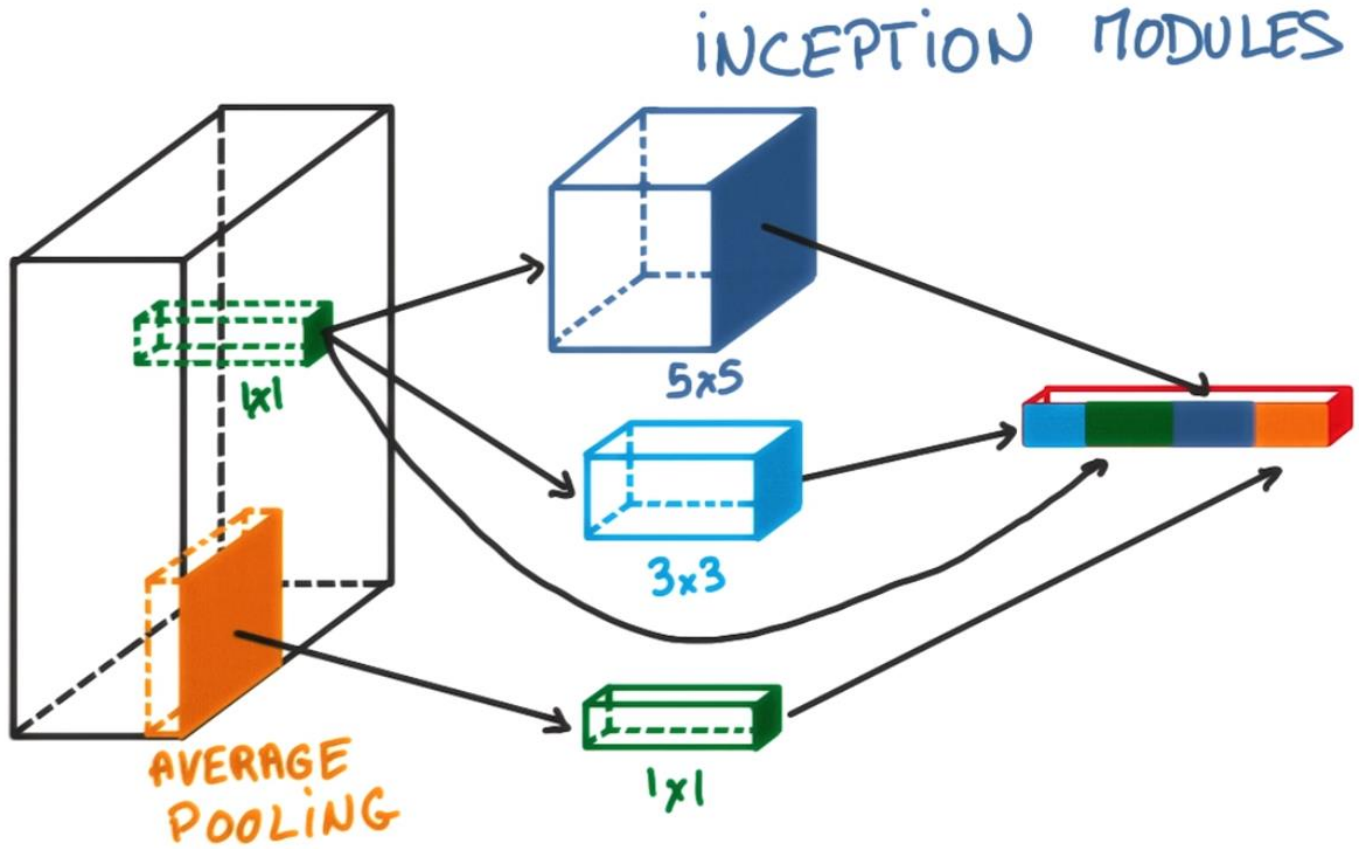
(b) Inception module with dimension reductions

https://becominghuman.ai/understanding-and-coding-inception-module-in-keras-eb56e9056b4b

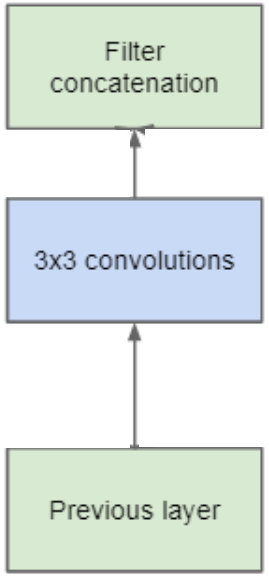
Inception



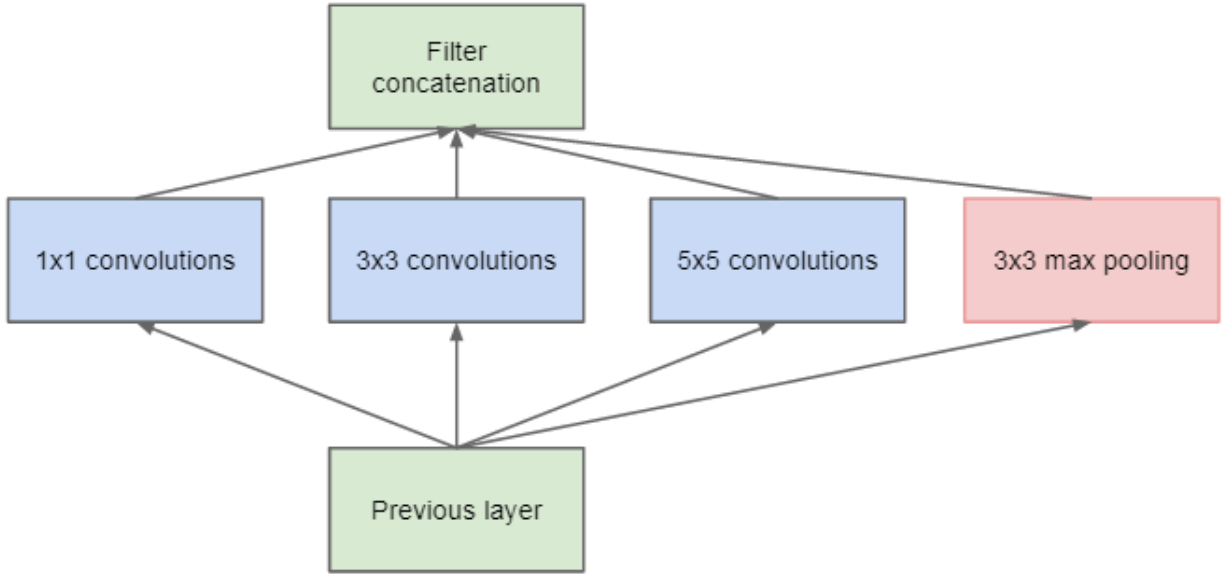
(a) Inception module, naïve version



Difference from VGG



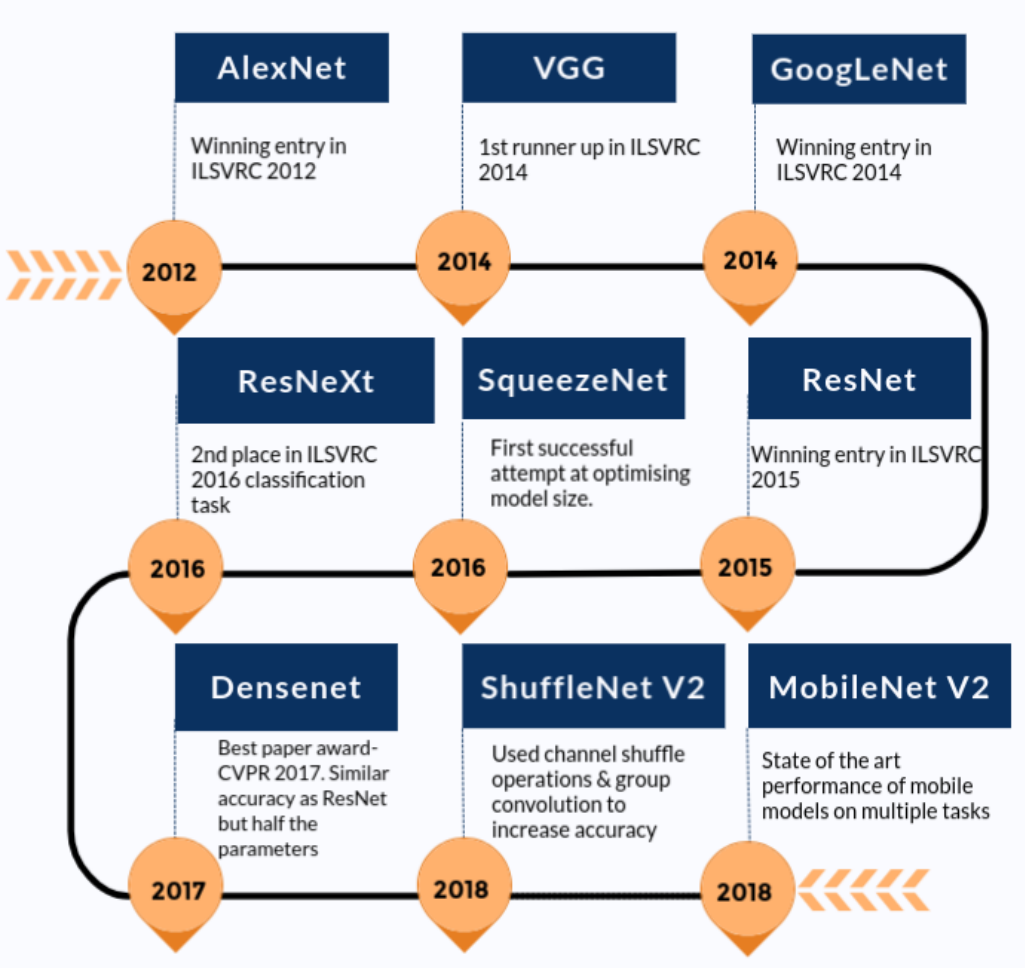
VGG



Google

https://becominghuman.ai/understanding-and-coding-inception-module-in-keras-eb56e9056b4b

State-of-the-art



AlexNet

VGG

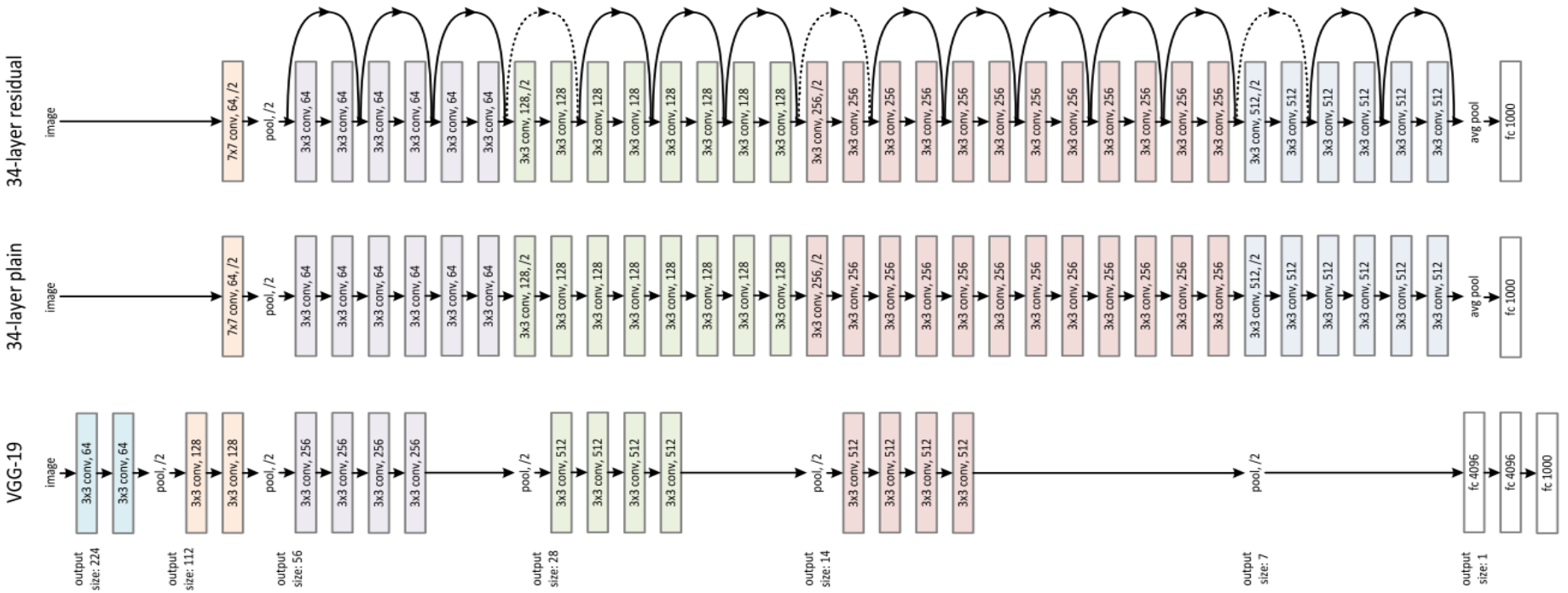
GoogleNet

ResNet

DenseNet

https://www.learnopencv.com/pytorch-for-beginners-image-classification-using-pre-trained-models/

ResNet(2015) vs VGG



Ay the ILSVRC 2015, the so-called Residual Neural Network (ResNet) by Kaiming He et al introduced anovel architecture with “skip connections” and features heavy batch normalization. Such skip connections are also known as gated units or gated recurrent units and have a strong similarity to recent successful elements applied in RNNs. Thanks to this technique they were able to train a NN with 152 layers while still having lower complexity than VGGNet. It achieves a top-5 error rate of 3.57% which beats human-level performance on this dataset.

https://medium.com/@pierre_guillou/understand-how-works-resnet-without-talking-about-residual-64698f157e0c

https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

ResNet

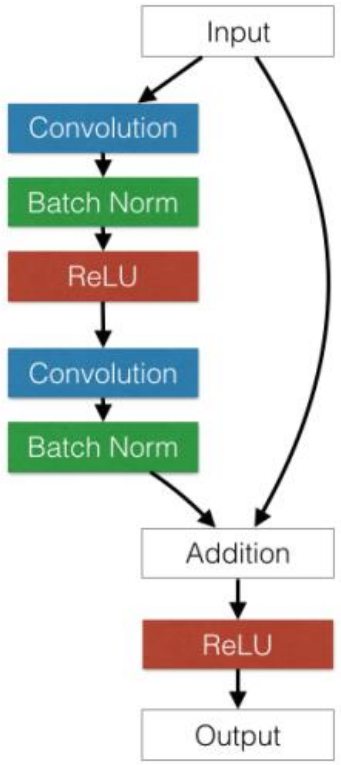
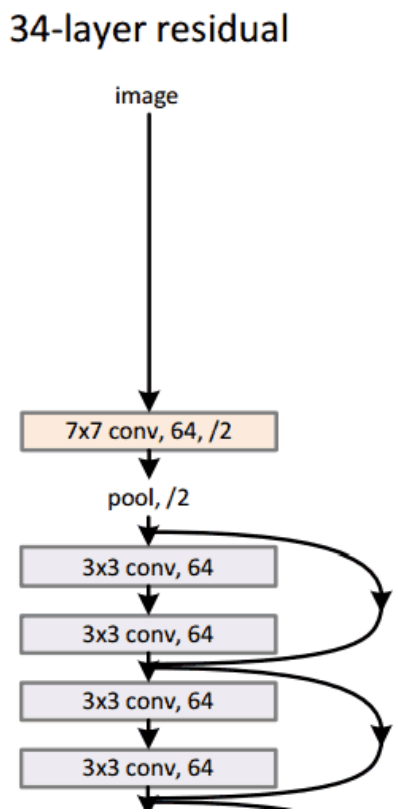


Figure 1. A RestNet basic block



Netscope CNN Analyzer

A web-based tool for visualizing and analyzing convolutional neural network architectures (or technically, any directed acyclic graph). Currently supports Caffe's prototxt format.

Basis by ethereon. Extended for CNN Analysis by dgschwend.

Gist Support

If your `.prototxt` file is part of a GitHub Gist, you can visualize it by visiting this URL:

```
http://dgschwend.github.io/netscope/#/gist/your-gist-id
```

https://dgschwend.github.io/netscope/#/preset/resnet-50

https://www.kdnuggets.com/2016/09/9-key-deep-learning-papers-explained.html/2

http://dgschwend.github.io/netscope/quickstart.html

After ResNet

IMAGENET **Large Scale Visual Recognition Challenge 2016 (ILSVRC2016)**

DET LOC VID Scene **Team information**

Legend:
Yellow background = winner in this task according to this metric; authors are willing to reveal the method
White background = authors are willing to reveal the method
Grey background = authors chose not to reveal the method
Italics = authors requested entry not participate in competition

Object detection (DET)[top]

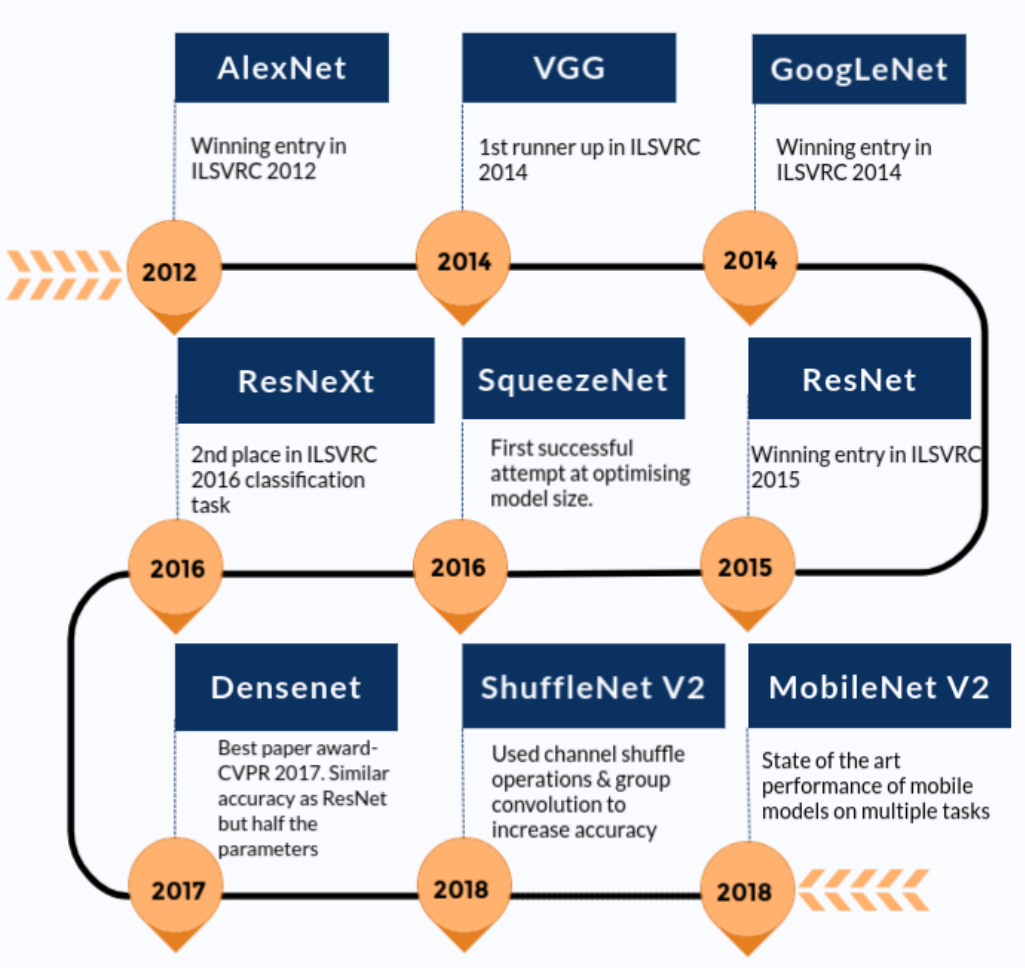
Task 1a: Object detection with provided training data

Ordered by number of categories won

Team name	Entry description	**Number of object categories won**	mean AP
CUImage	Ensemble of 6 models using provided data	109	0.662751
Hikvision	Ensemble A of 3 RPN and 6 FRCN models, mAP is 67 on val2	30	0.652704
Hikvision	Ensemble B of 3 RPN and 5 FRCN models, mean AP is 66.9, median AP is 69.3 on val2	18	0.652003

ImageNet 2017

State-of-the-art



AlexNet

VGG

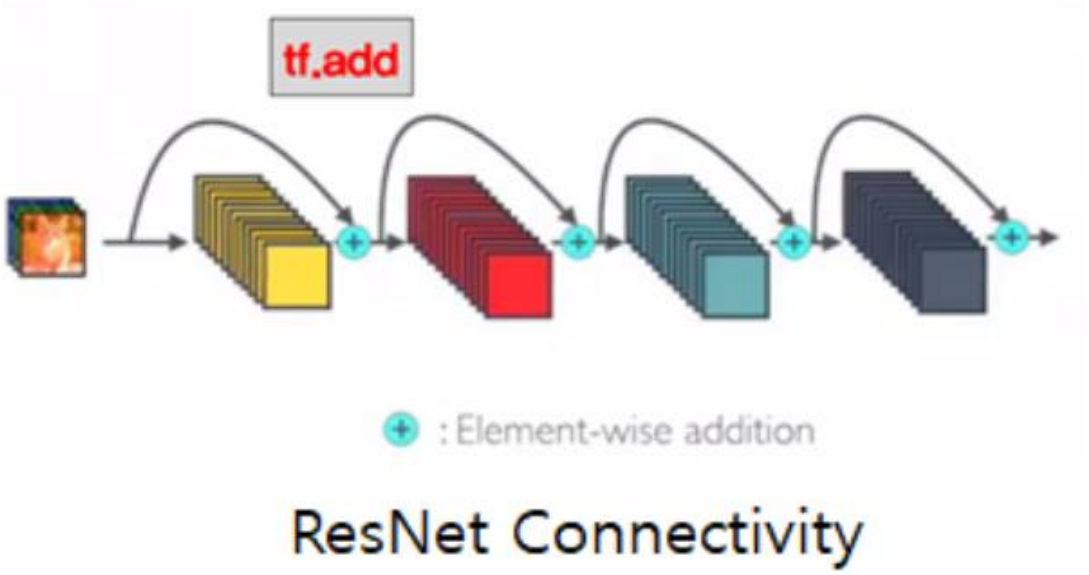
GoogleNet

ResNet

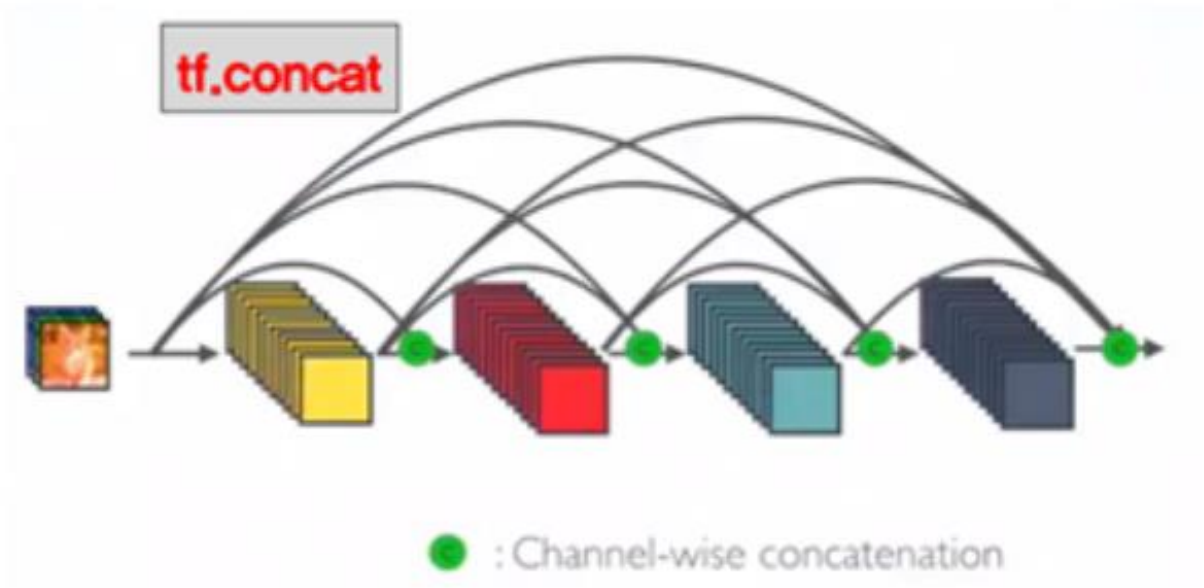
DenseNet

https://www.learnopencv.com/pytorch-for-beginners-image-classification-using-pre-trained-models/

More Connections



ResNet Connectivity



Dense Connectivity

https://jayhey.github.io/deep%20learning/2017/10/13/DenseNet_1/

Densenet

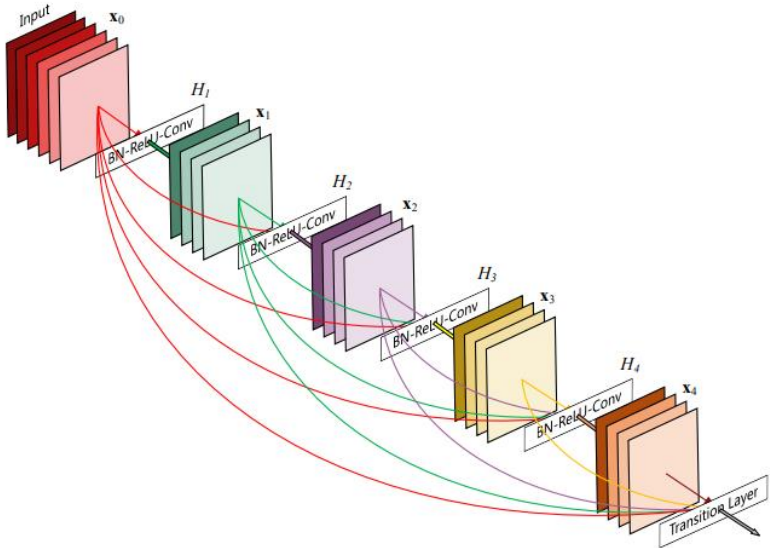
Densely Connected Convolutional Networks

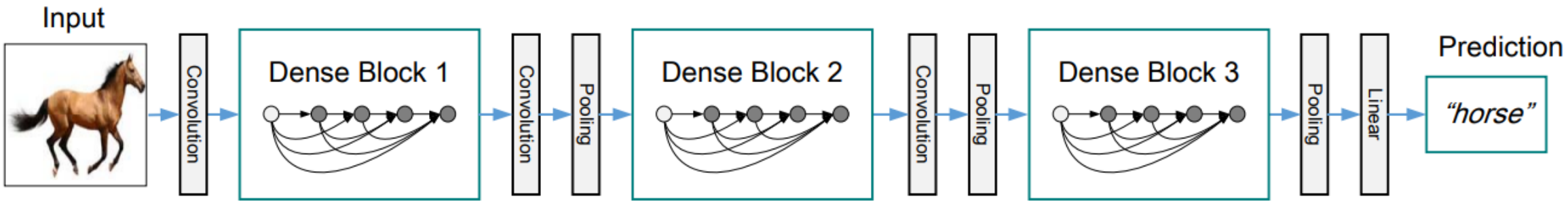
Gao Huang*
Cornell University
gh349@cornell.edu

Zhuang Liu*
Tsinghua University
liuzhuang13@mails.tsinghua.edu.cn

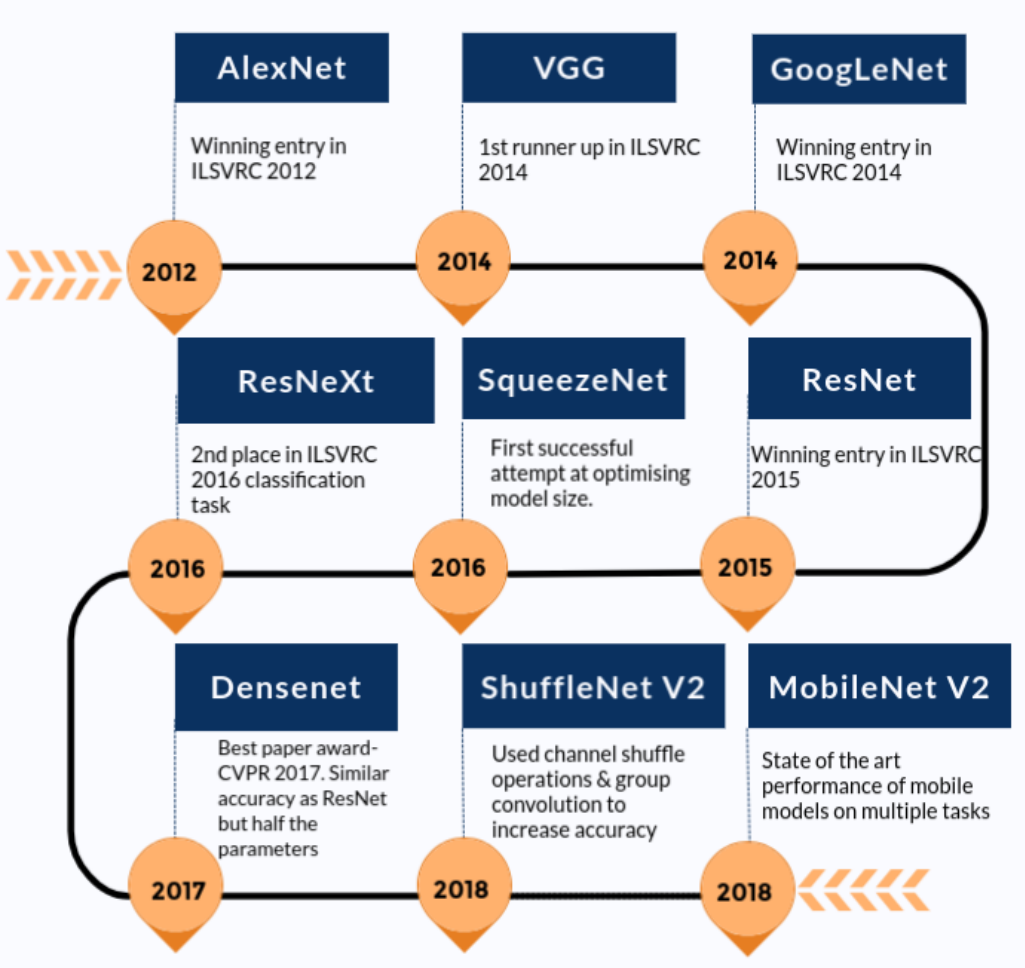
Laurens van der Maaten
Facebook AI Research
lvdmaaten@fb.com

Kilian Q. Weinberger
Cornell University
kqw4@cornell.edu





State-of-the-art



AlexNet

VGG

GoogleNet

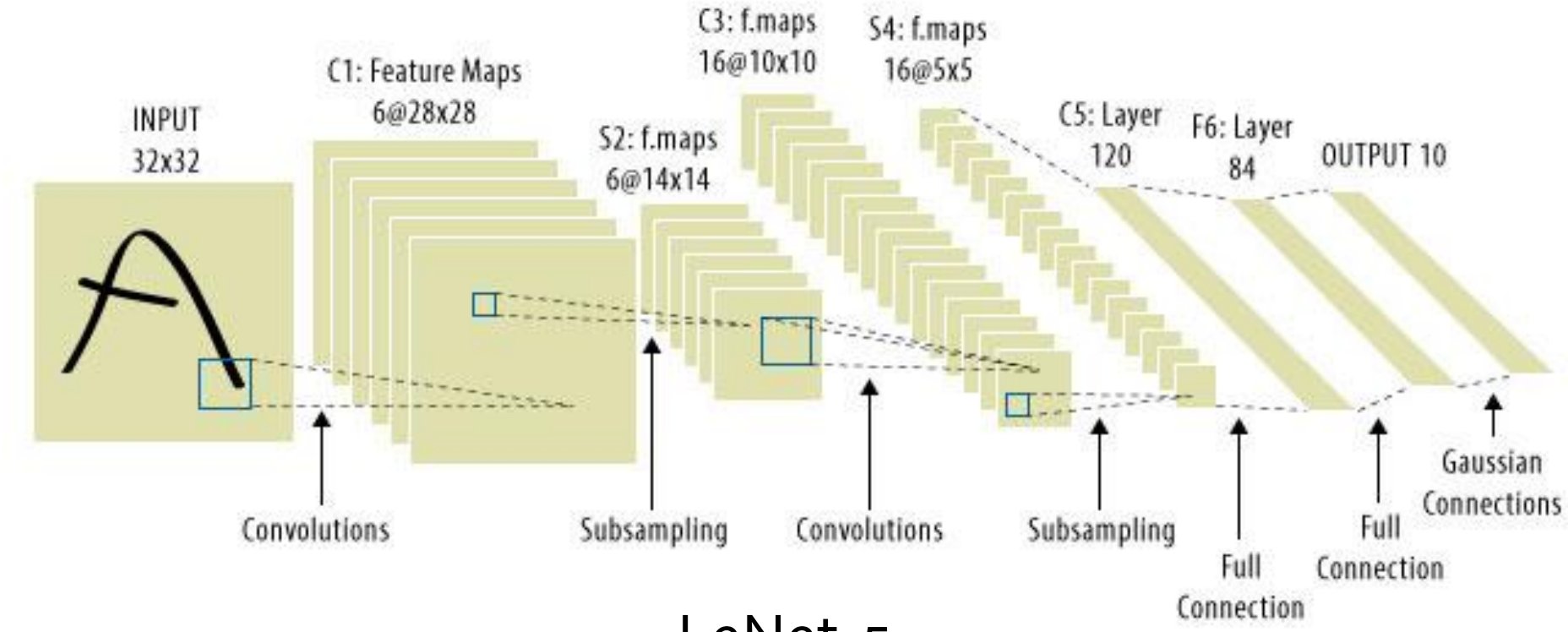
ResNet

DenseNet

https://www.learnopencv.com/pytorch-for-beginners-image-classification-using-pre-trained-models/

Summary of Classification

"CONV-POOL-FC"



LeNet-5

Assignment 0

```
from __future__ import print_function
import argparse
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision import datasets, transforms
```

Assignment o

```
def main():
    # Training settings
    parser = argparse.ArgumentParser(description='PyTorch MNIST Example')
    parser.add_argument('--batch-size', type=int, default=64, metavar='N',
                        help='input batch size for training (default: 64)')
    parser.add_argument('--test-batch-size', type=int, default=1000, metavar='N',
                        help='input batch size for testing (default: 1000)')
    parser.add_argument('--epochs', type=int, default=10, metavar='N',
                        help='number of epochs to train (default: 10)')
    parser.add_argument('--lr', type=float, default=0.01, metavar='LR',
                        help='learning rate (default: 0.01)')
    parser.add_argument('--momentum', type=float, default=0.5, metavar='M',
                        help='SGD momentum (default: 0.5)')
    parser.add_argument('--no-cuda', action='store_true', default=False,
                        help='disables CUDA training')
    parser.add_argument('--seed', type=int, default=1, metavar='S',
                        help='random seed (default: 1)')
    parser.add_argument('--log-interval', type=int, default=10, metavar='N',
                        help='how many batches to wait before logging training status')
    args = parser.parse_args()
    use_cuda = not args.no_cuda and torch.cuda.is_available()
```

Assignment o

```
torch.manual_seed(args.seed)

device = torch.device("cuda" if use_cuda else "cpu")

kwargs = {'num_workers': 1, 'pin_memory': True} if use_cuda else {}
train_loader = torch.utils.data.DataLoader(
    datasets.MNIST('../data', train=True, download=True,
                   transform=transforms.Compose([
                       transforms.ToTensor(),
                       transforms.Normalize((0.1307,), (0.3081,))
                   ])),
    batch_size=args.batch_size, shuffle=True, **kwargs)
test_loader = torch.utils.data.DataLoader(
    datasets.MNIST('../data', train=False, transform=transforms.Compose([
                       transforms.ToTensor(),
                       transforms.Normalize((0.1307,), (0.3081,))
                   ])),
    batch_size=args.test_batch_size, shuffle=True, **kwargs)
```

Assignment o

```
model = Net().to(device)
optimizer = optim.SGD(model.parameters(), lr=args.lr, momentum=args.momentum)

for epoch in range(1, args.epochs + 1):
    train(args, model, device, train_loader, optimizer, epoch)
    test(args, model, device, test_loader)
```

Assignment 0

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 10, kernel_size=5)
        self.conv2 = nn.Conv2d(10, 20, kernel_size=5)
        self.conv2_drop = nn.Dropout2d()
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
        x = x.view(-1, 320)
        x = F.relu(self.fc1(x))
        x = F.dropout(x, training=self.training)
        x = self.fc2(x)
        return F.log_softmax(x, dim=1)
```

Assignment o

```
def train(args, model, device, train_loader, optimizer, epoch):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        loss.backward()
        optimizer.step()
        if batch_idx % args.log_interval == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(train_loader.dataset),
                100. * batch_idx / len(train_loader), loss.item()))
```

Assignment o

```
def test(args, model, device, test_loader):
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            test_loss += F.nll_loss(output, target, reduction='sum').item() # sum
            pred = output.max(1, keepdim=True)[1] # get the index of the max log-p
            correct += pred.eq(target.view_as(pred)).sum().item()

    test_loss /= len(test_loader.dataset)
    print('\nTest set: Average loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)\n'.format(
        test_loss, correct, len(test_loader.dataset),
        100. * correct / len(test_loader.dataset)))
```