UNDERSTANDING CNN

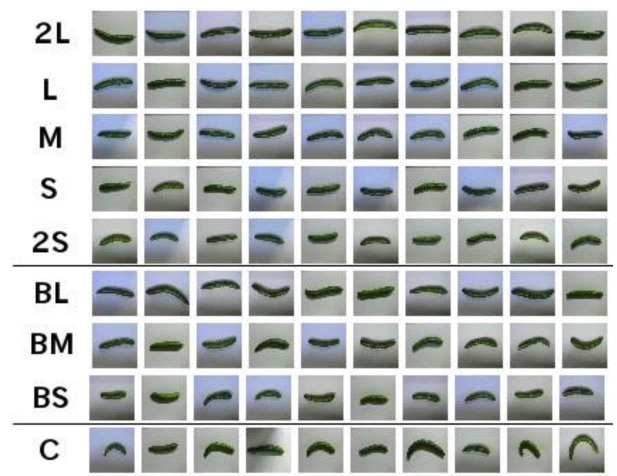
Speaker: Hyung Il Koo, Ajou University

TENSORFLOW-POWERED CUCUMBER SORTER

Cucumber sorting

- Each cucumber has different color, shape, quality and freshness.
- At Makoto's farm, they sort them into nine different classes, and his mother sorts them all herself — spending up to eight hours per day at peak harvesting times.

Makoto Koike, center, with his parents at the family cucumber farm



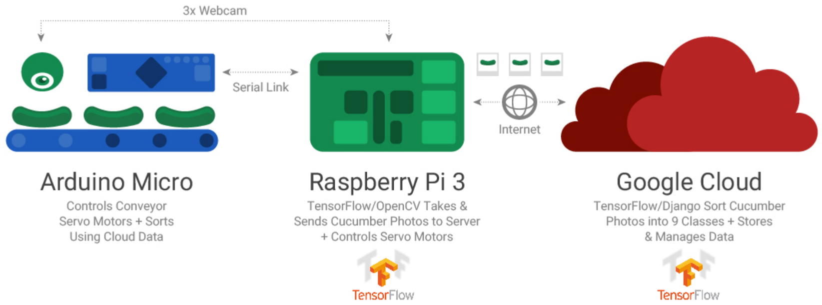
Cucumber sorting

- You have to look at not only the size and thickness, but also the color, texture, small scratches, whether or not they are crooked and whether they have prickles. It takes months to learn the system and **you can't just hire part-time workers during the busiest period**. I myself only recently learned to sort cucumbers well," Makoto said.

- Makoto doesn't think sorting is an essential task for cucumber farmers. "**Farmers want to focus and spend their time on growing delicious vegetables**. I'd like to automate the sorting tasks before taking the farm business over from my parents.

Tensorflow-powered cucumber sorter

- Makoto used the sample TensorFlow code **Deep MNIST for Experts** with minor modifications to the convolution, pooling and last layers, changing the network design to adapt to the pixel format of cucumber images and the number of cucumber classes.

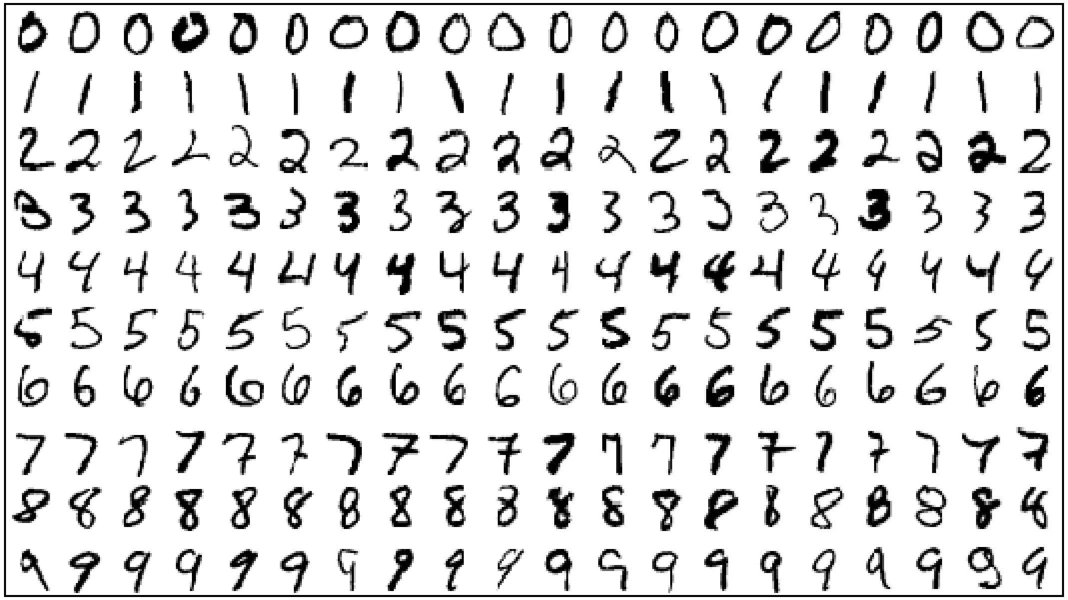


Cucumber sorter by Makoto Koike

MNIST & LENET

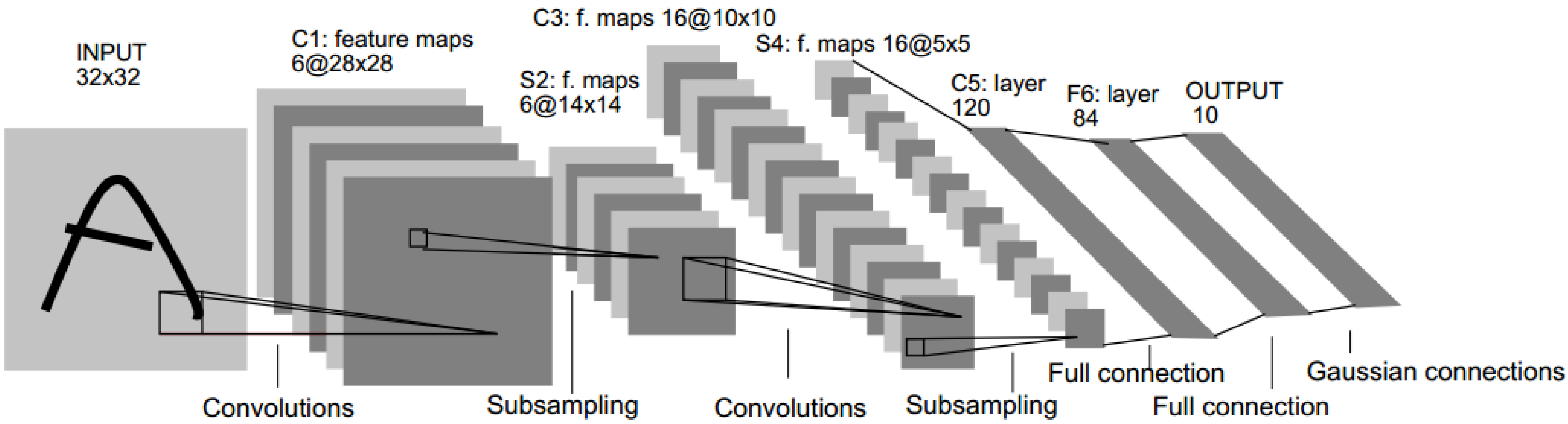
MNIST dataset

- handwritten digits
- a training set of 60,000 examples
- 24x24 images

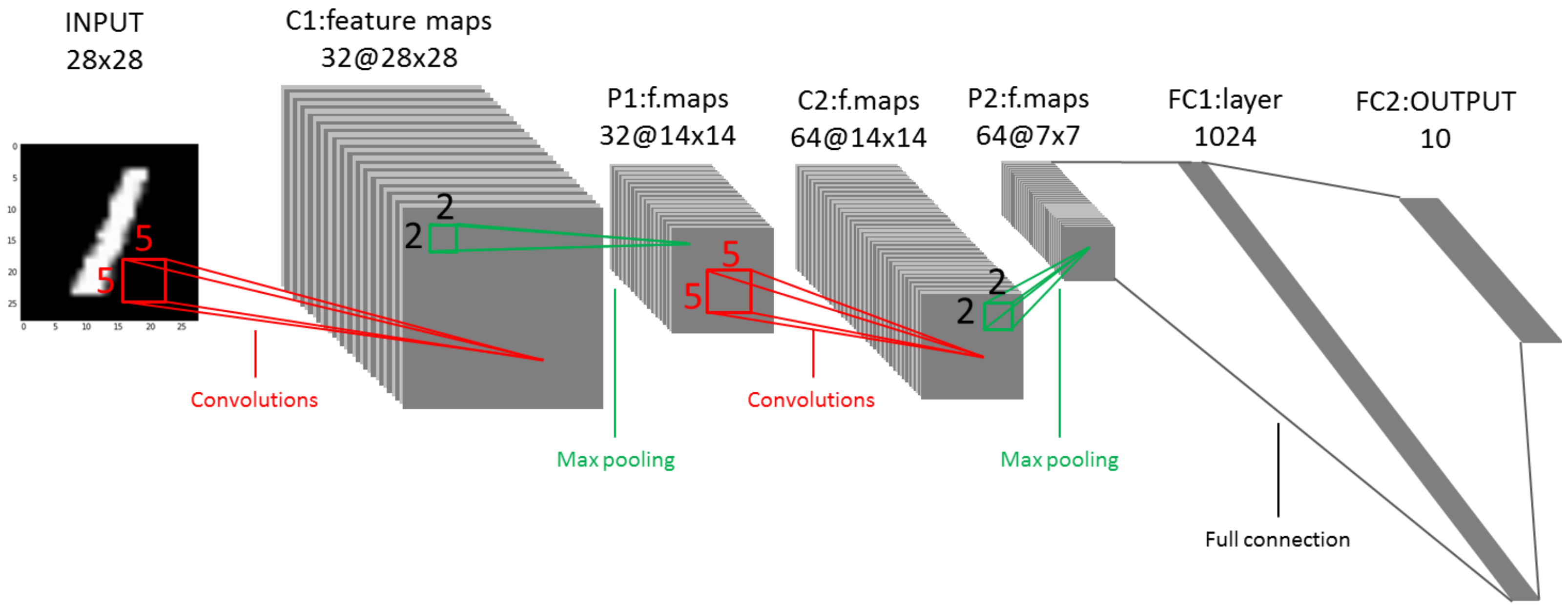


LeNet

- Yann LeCun and his collaborators developed a recognizer for handwritten digits by using back-propagation in a feed-forward net



LeNet

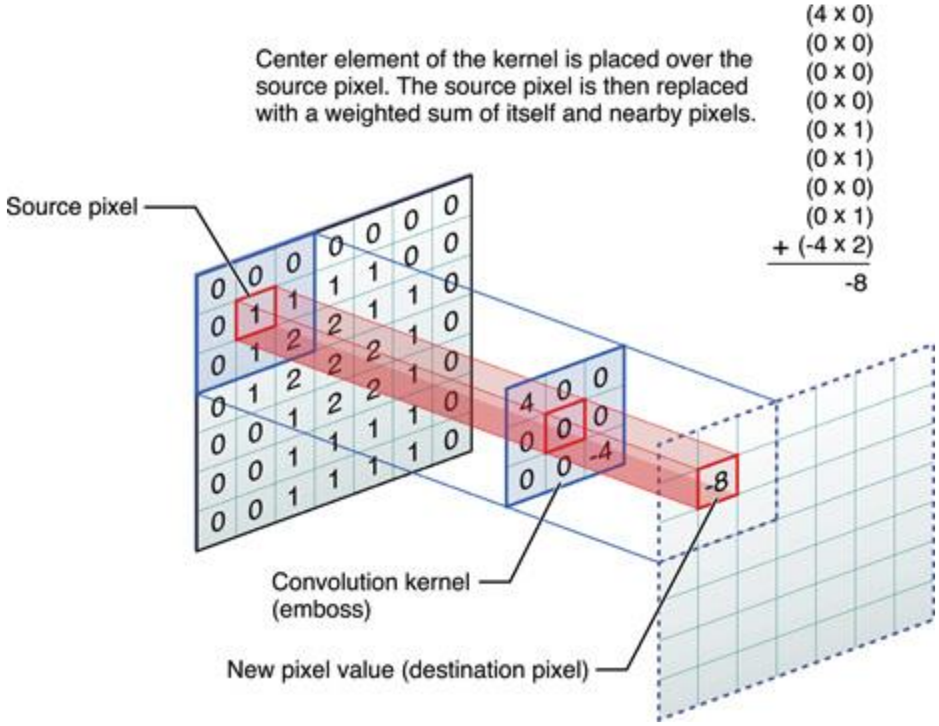


#(Parameter) = 3,274,634

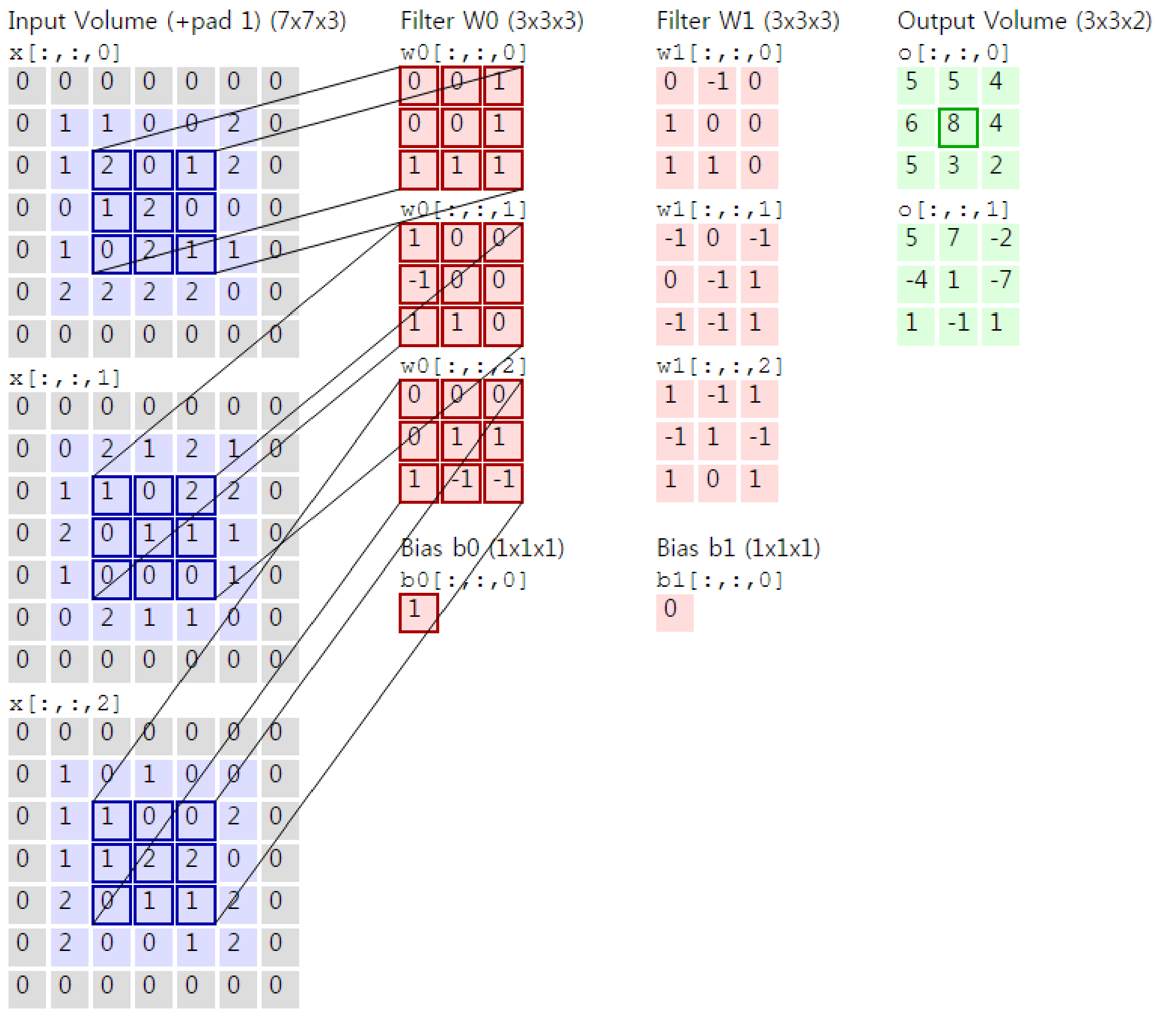
Layer	C1	C2	FC1	FC2
Weight	800	51,200	3,211,264	10,240
Bias	32	64	1,024	10

CNN BUILDING BLOCKS

Convolution

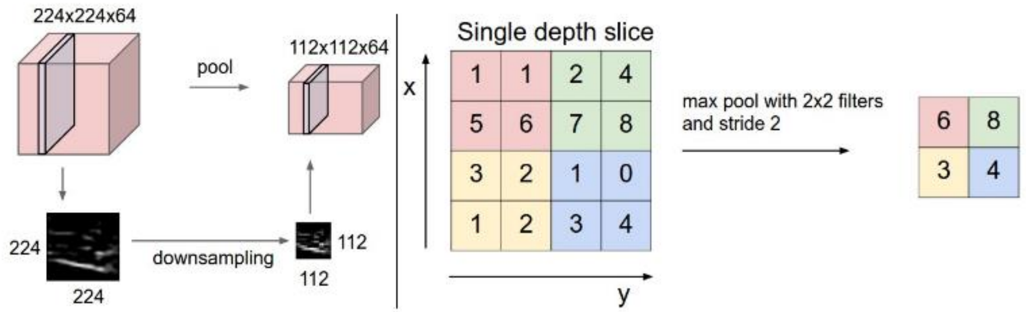


Convolutions in CNNs



Pooling

- Max vs Average pooling



Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. **Left:** In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. **Right:** The most common downsampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).

DEEP MNIST FOR EXPERTS

Deep MNIST for Experts

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets('MNIST_data', one_hot=True)

def weight_variable(shape):
    # tf.truncated_normal: Outputs random values from a truncated normal distribution.
    # values whose magnitude is more than 2 standard deviations from the mean are dropped and re-picked.
    initial = tf.truncated_normal(shape, stddev=0.1)
    return tf.Variable(initial)

def bias_variable(shape):
    initial = tf.constant(0.1, shape=shape)
    return tf.Variable(initial)

# convolution & max pooling
def conv2d(x, W):
    return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

def max_pool_2x2(x):
  return tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

x = tf.placeholder(tf.float32, [None, 784])
y_ = tf.placeholder(tf.float32, [None, 10])

# [5, 5, 1, 32]: 5x5 convolution patch, 1 input channel, 32 output channel.
W_conv1 = weight_variable([5, 5, 1, 32])
b_conv1 = bias_variable([32])

x_image = tf.reshape(x, [-1,28,28,1])

# convolution, relu, max pooling
h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)

# [5, 5, 32, 64]: 5x5 convolution patch, 32 input channel, 64 output channel.
W_conv2 = weight_variable([5, 5, 32, 64])
b_conv2 = bias_variable([64])

# convolution, relu, max pooling
h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)

# fc layer 1
W_fc1 = weight_variable([7 * 7 * 64, 1024])
b_fc1 = bias_variable([1024])
```

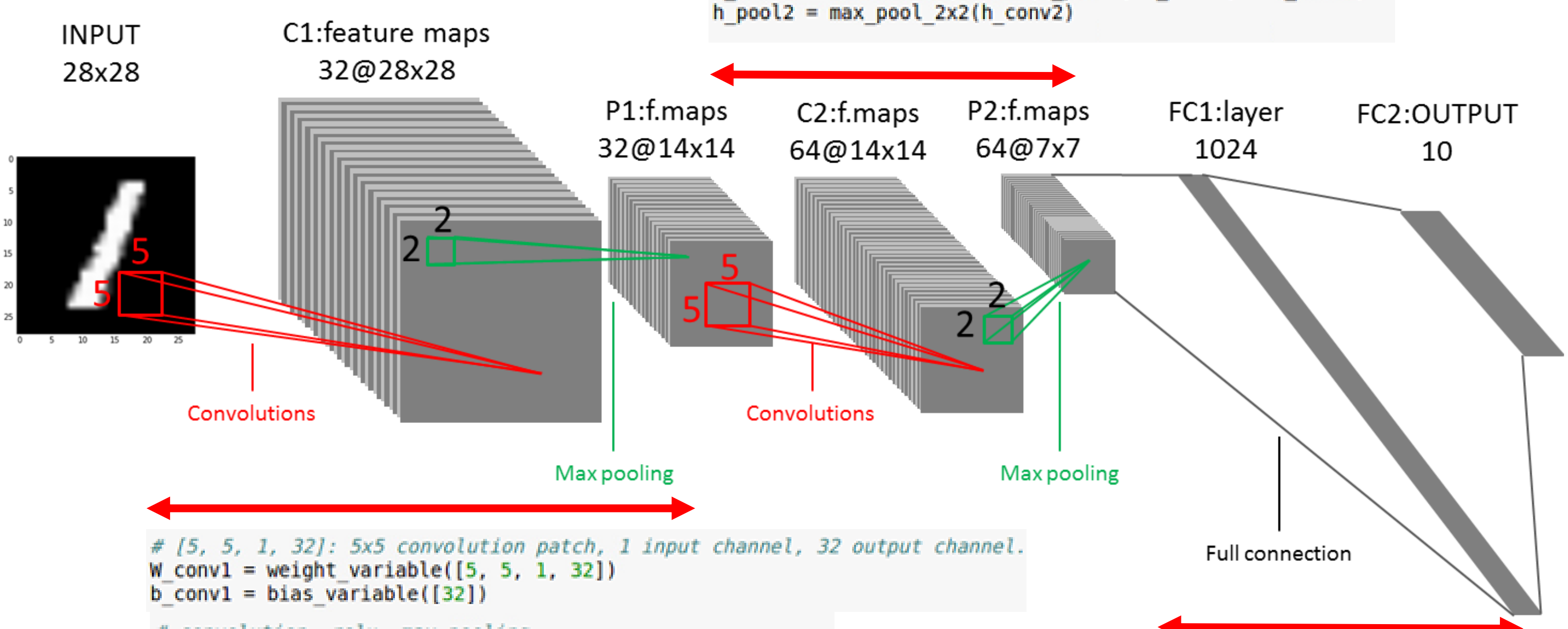
Deep MNIST for Experts

```python
W_fc1 = weight_variable([7 * 7 * 64, 1024])
b_fc1 = bias_variable([1024])

h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)

keep_prob = tf.placeholder("float")
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

# fc layer 2
W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])

y_conv=tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)



cross_entropy = -tf.reduce_sum(y_*tf.log(y_conv))
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
init = tf.initialize_all_variables()

ce_sum = tf.scalar_summary("cross entropy", cross_entropy)
acc_sum = tf.scalar_summary("accuracy", accuracy)
merged = tf.merge_summary([ce_sum, acc_sum])

sess = tf.Session()
writer = tf.train.SummaryWriter("./sumlog", sess.graph)

sess.run(init)

with sess.as_default():
    for i in range(20000):
        batch = mnist.train.next_batch(50)
        if i%400 == 0:
            train_accuracy = accuracy.eval(feed_dict={x:batch[0], y_: batch[1], keep_prob: 1.0})
            print "step %d, training accuracy %g" % (i, train_accuracy)
            print "test accuracy %g" % accuracy.eval(feed_dict={x: mnist.test.images, y_: mnist.test.labels,
                                                    keep_prob: 1.0})
            writer.add_summary(merged.eval(feed_dict={x: mnist.test.images, y_: mnist.test.labels,
                                                    keep_prob: 1.0}),i)

        train_step.run(feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})
```

LeNet

```
# [5, 5, 32, 64]: 5x5 convolution patch, 32 input channel, 64 output channel.
W_conv2 = weight_variable([5, 5, 32, 64])
b_conv2 = bias_variable([64])
```

```
# convolution, relu, max pooling
h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)
```




```
# [5, 5, 1, 32]: 5x5 convolution patch, 1 input channel, 32 output channel.
W_conv1 = weight_variable([5, 5, 1, 32])
b_conv1 = bias_variable([32])
```

```
# convolution, relu, max pooling
h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)
```

```
h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)

keep_prob = tf.placeholder("float")
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)
```

#(Parameter) = 3,274,634

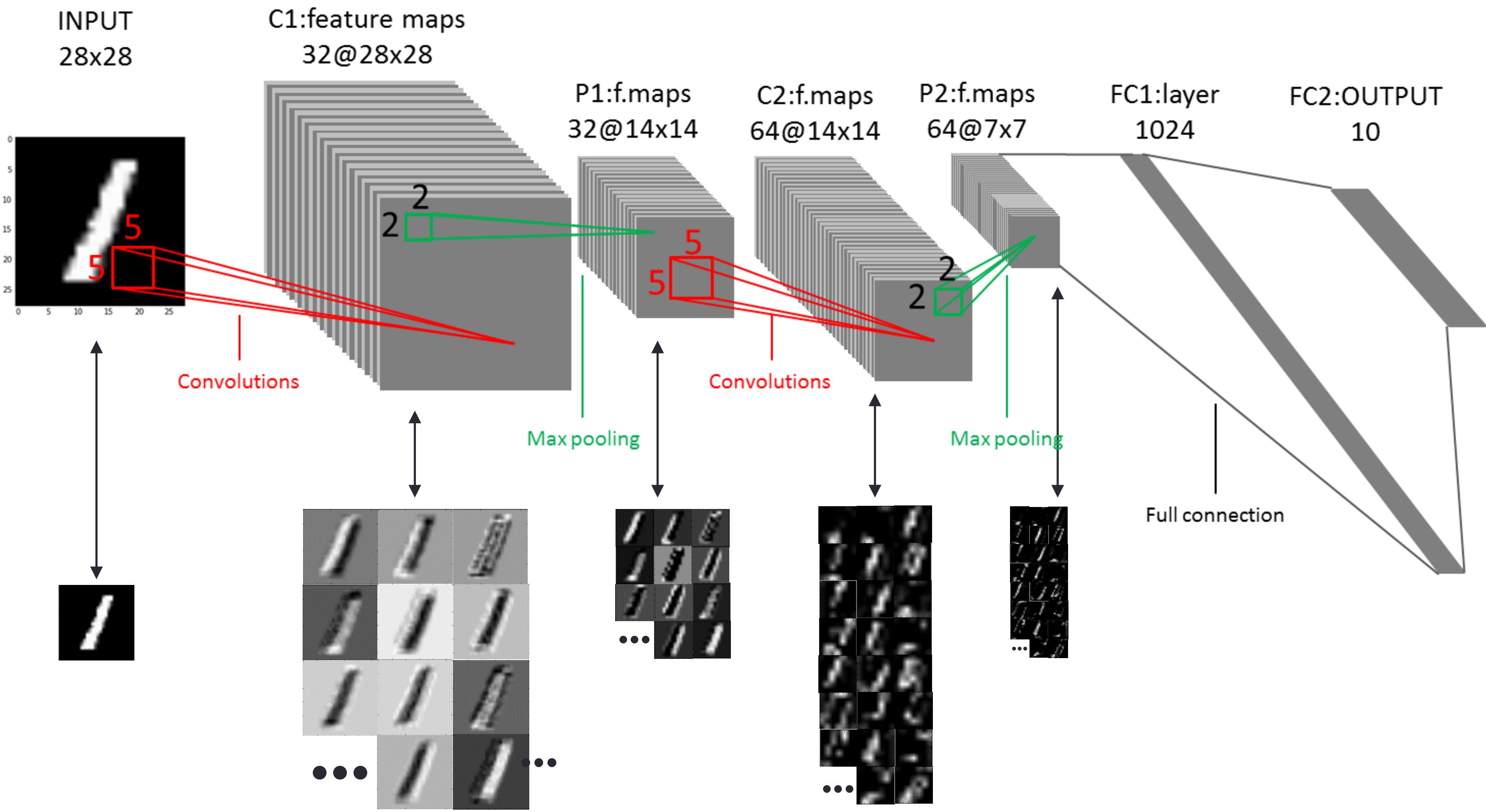
Layer	C1	C2	FC1	FC2
Weight	800	51,200	3,211,264	10,240
Bias	32	64	1,024	10

The 82 errors by LeNet5

Notice that most of the errors are cases that people find quite easy.

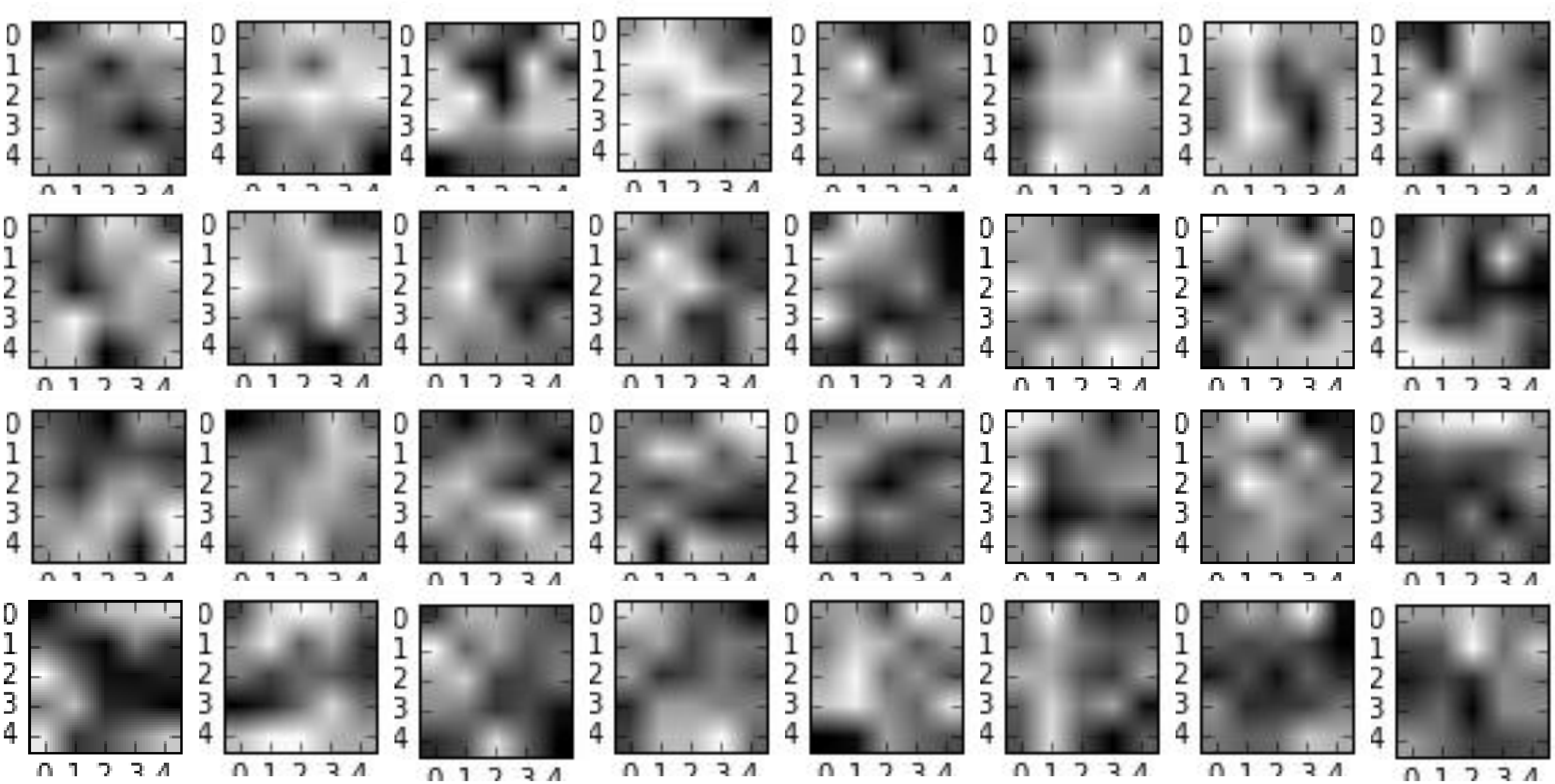
The human error rate is probably 20 to 30 errors but nobody has had the patience to measure it.

Feature map results



Learned Filters

Trained 32 filters on C1 layer



Learned Filters

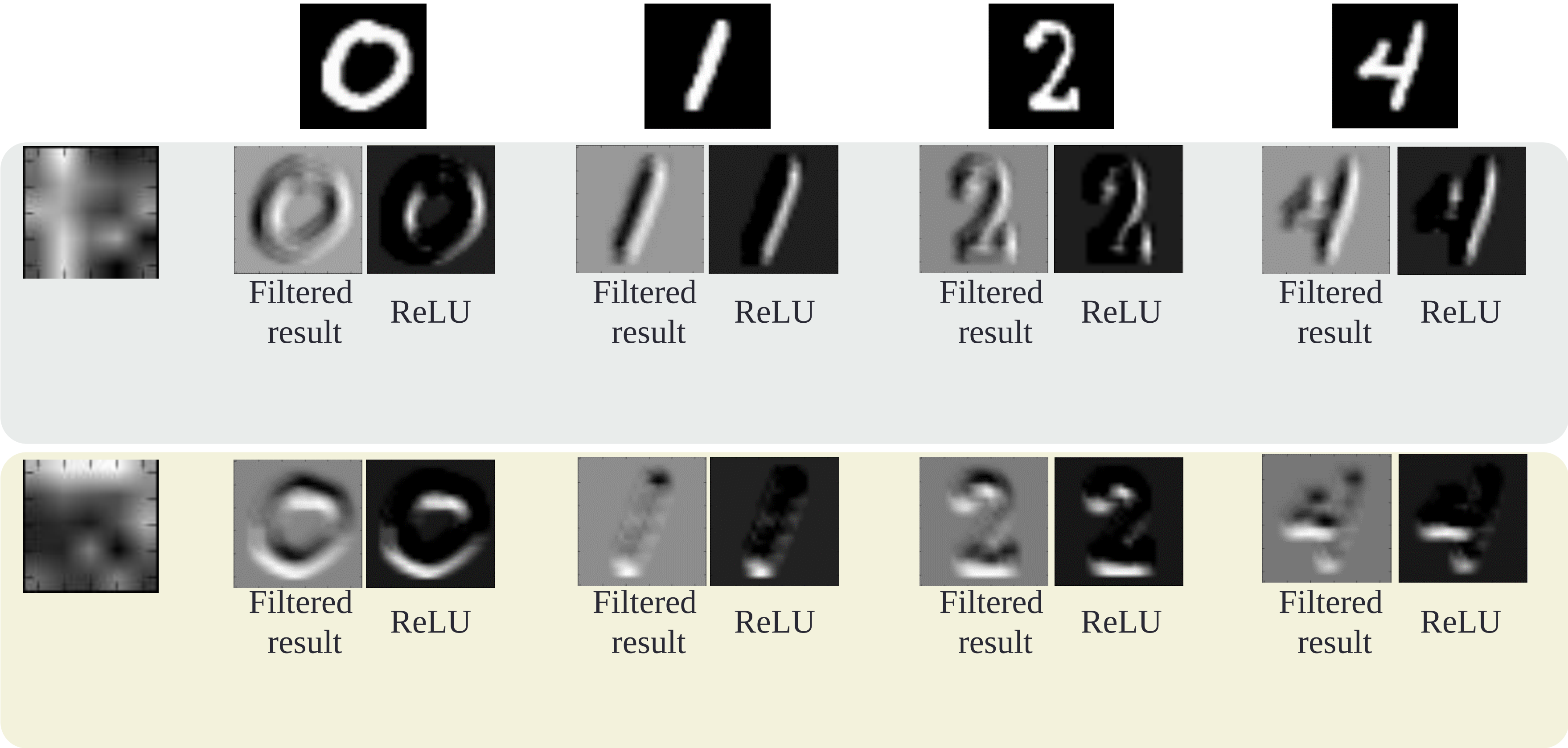
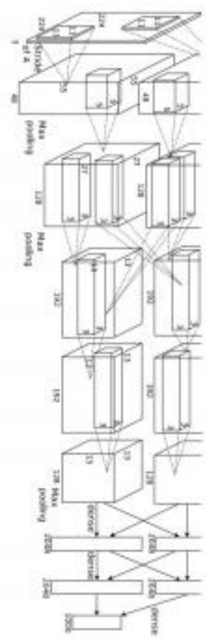


IMAGE CLASSIFICATION

Image Classification (ImageNet)

Year 2012

SuperVision



[Krizhevsky NIPS 2012]

Year 2014

GoogLeNet

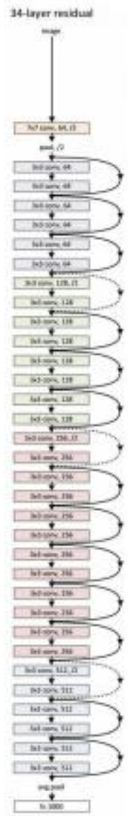
[Szegedy arxiv 2014]

VGG

[Simonyan arxiv 2014]

Year 2015

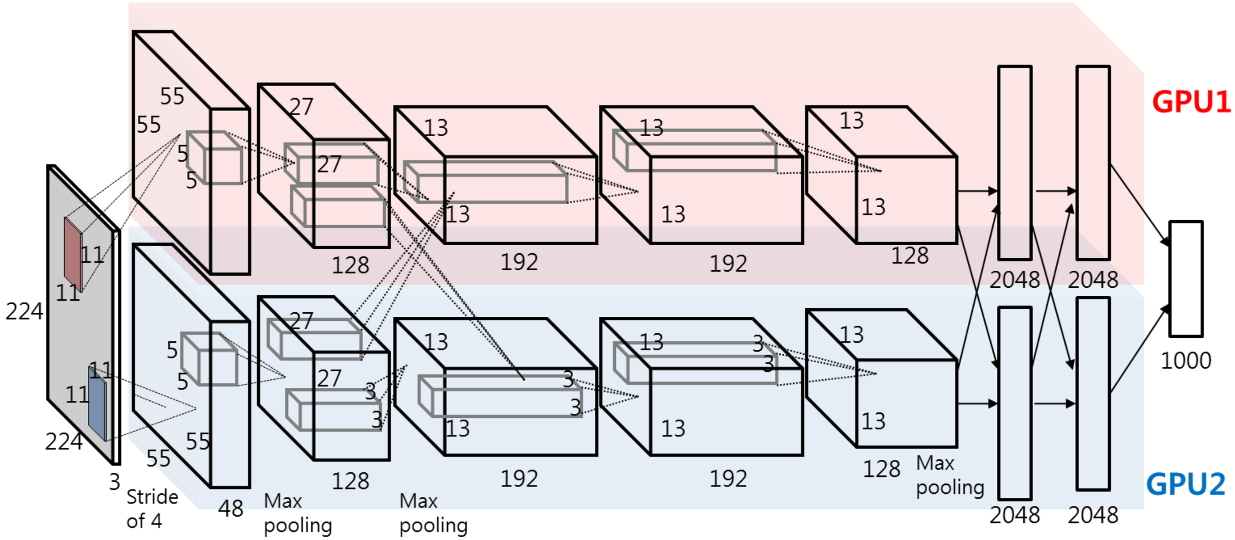
MSRA



ALEXNET

AlexNet

- AlexNet: won the 2012 ImageNet competition by making 40% l ess error than the next best competitor
  - It is composed of 5 convolutional layers
  - The input is a color RGB image
  - Computation is divided over 2 GPU architectures

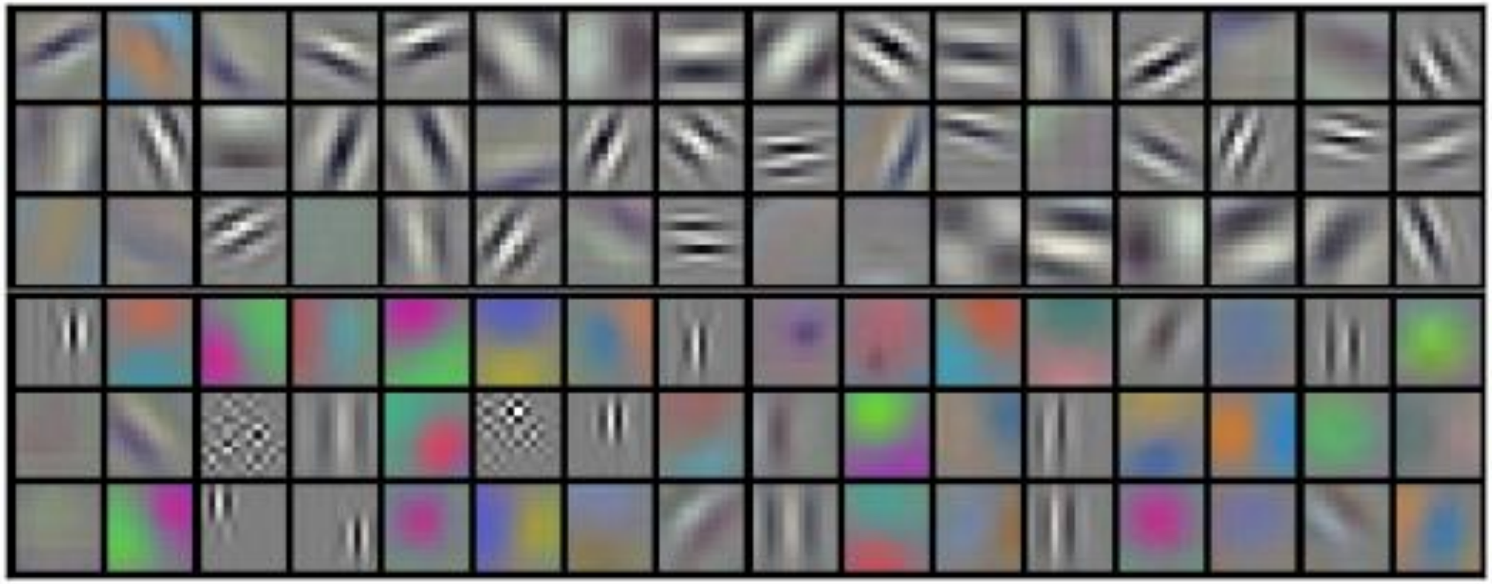


AlexNet results

- AlexNet TensorFlow codes and some results

AlexNet Visualization

- Filters learned by the first convolutional layer. The top half corresponds to the layer on one GPU, the bottom on the other. From Krizehvsky et al. (2012)
- Each of the 96 filters is of size [11x11x3]



VISUALIZATION

Motivation

- It is well known that Artificial Neural Networks show **remarkable performance** in image classification
- However, **we actually understand little** of why certain models work and others don’t

- There have been some attempts to visualize at each layer in the neural network
  - to know “how neural networks work and what each layer has learned”

https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

Why is this important?

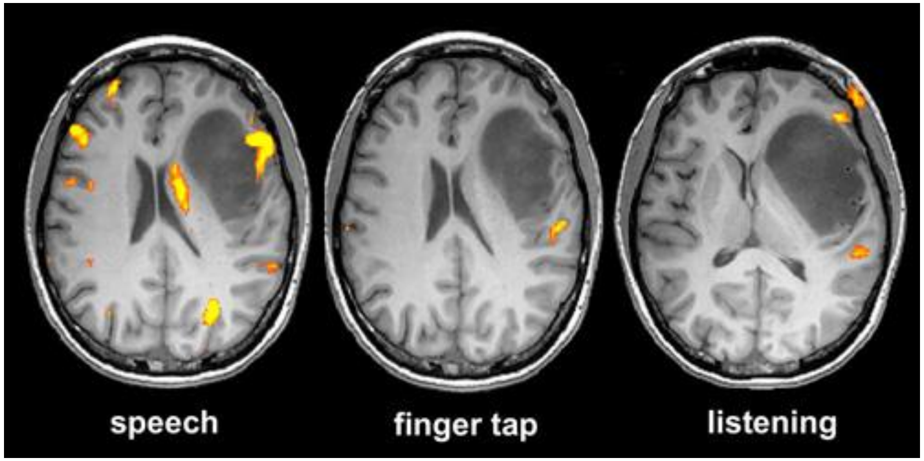
- There is a need of training networks with information we want to learn

But this program couldn't ignore what we don't care about

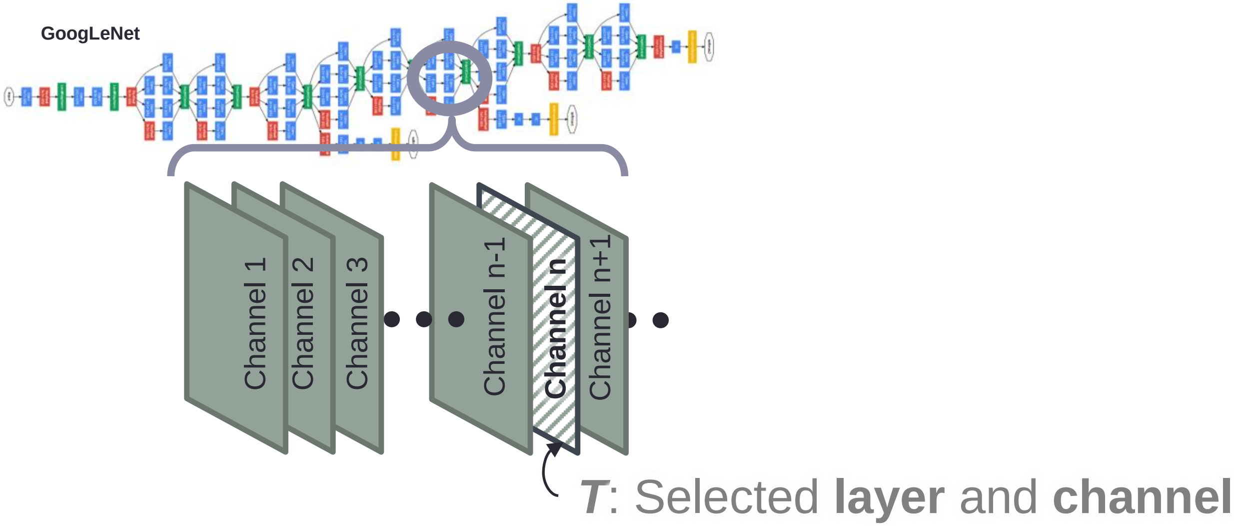
http://www.dailymail.co.uk/sciencetech/article-3145887/Google-apologises-Photos-app-tags-black-people-gorillas-Fault-image-recognition-software-mislabelled-picture.html

Visualization method

- Deconvolution
  - Matthew D. Zeiler and Rob Fergus, “Visualizing and Understanding Convolutional Networks,” ECCV 2014
- Input optimization
  - Naïve visualization
  - Low/High frequency normalization
    - With image prior
    - **With Laplacian (pyramid gradient) normalization**


speech
finger tap
listening

Naïve visualization

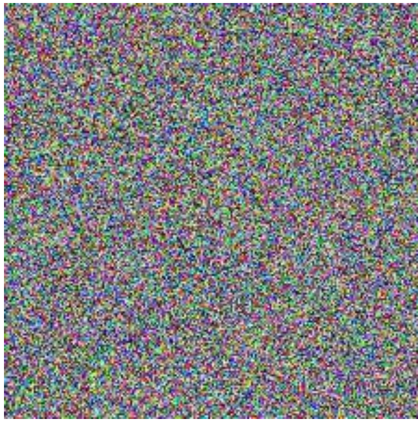


Objective function = L = mean(T)

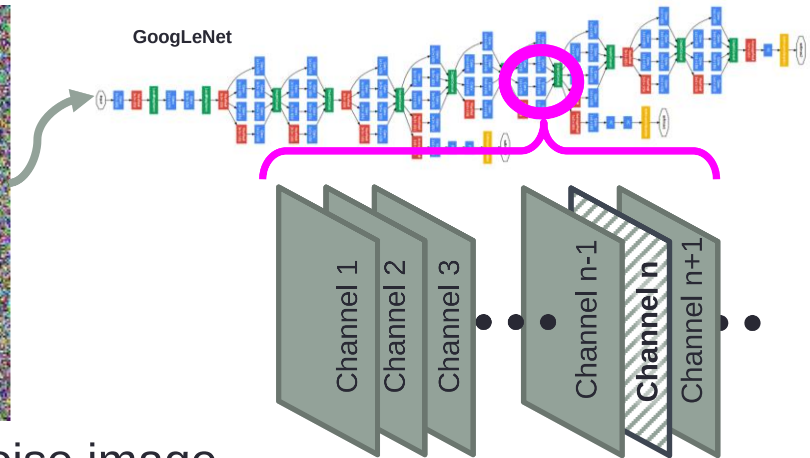
GRADIENT ASCENT:

$$img_{new} \leftarrow img_{old} + \alpha \times \frac{\partial(L)}{\partial(img)}\Big|_{img_{old}}$$

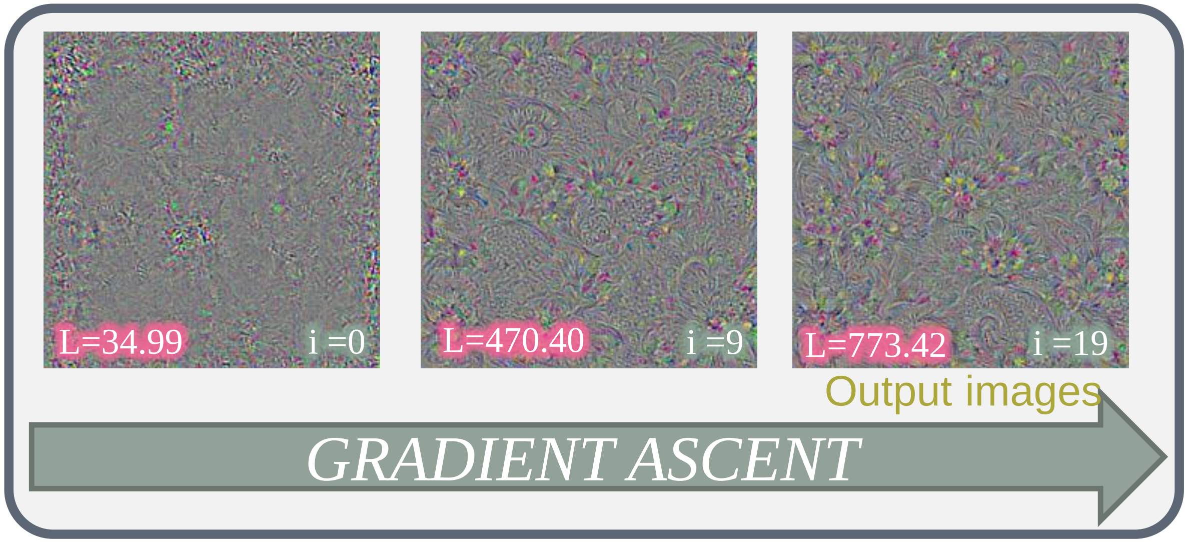
Naïve visualization

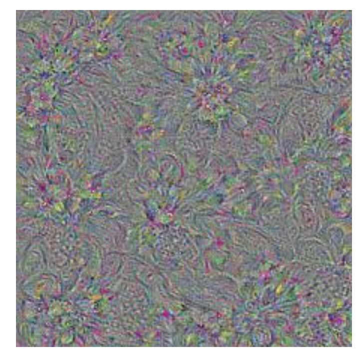


Initial input: an arbitrary noise image

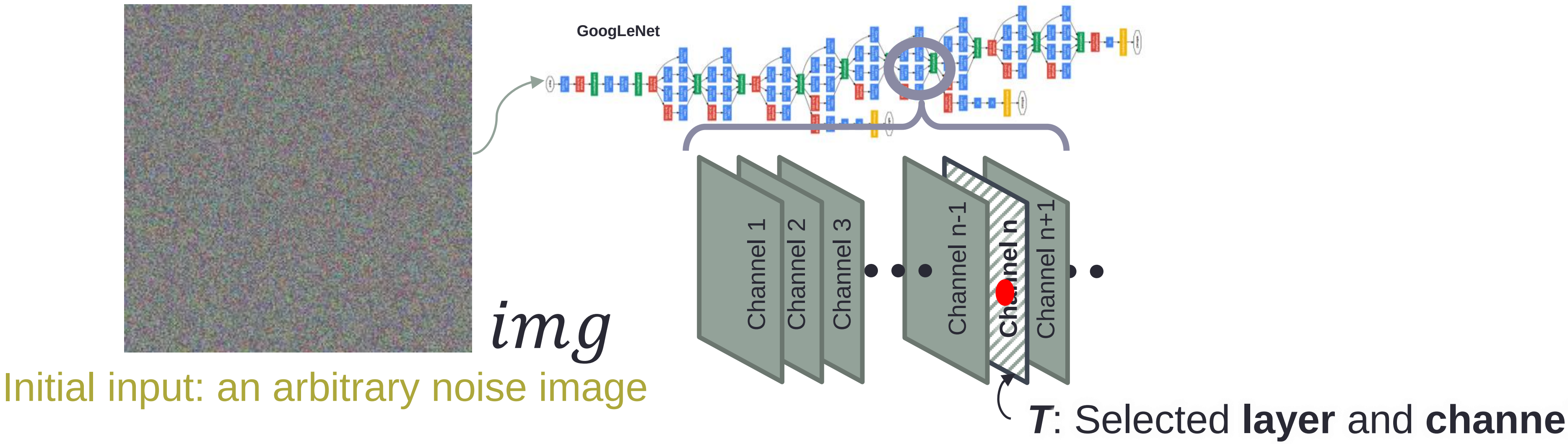


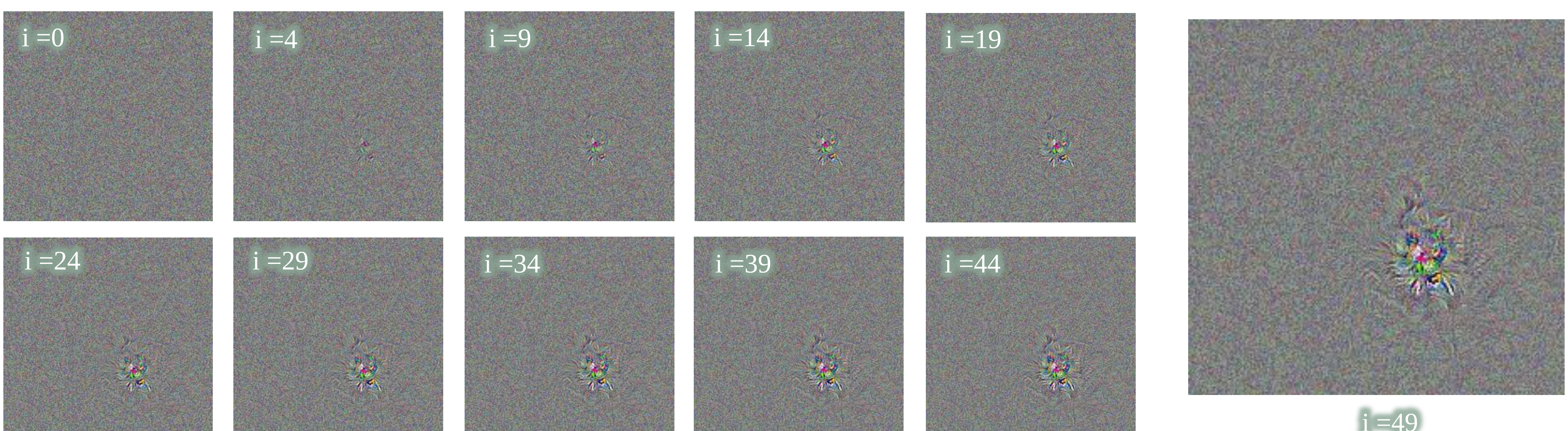
Selected **layer** and **channel**



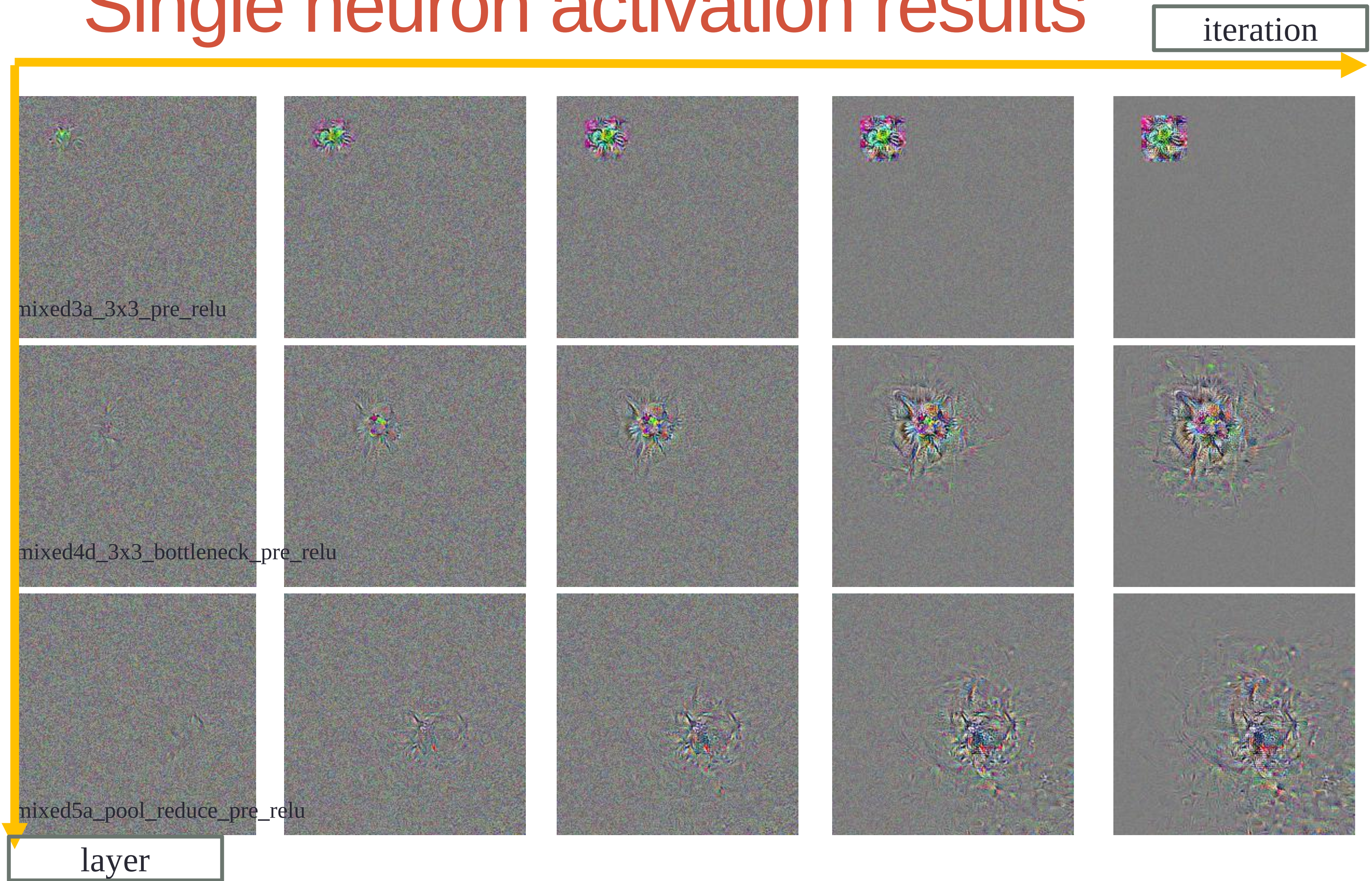


Single neuron activation

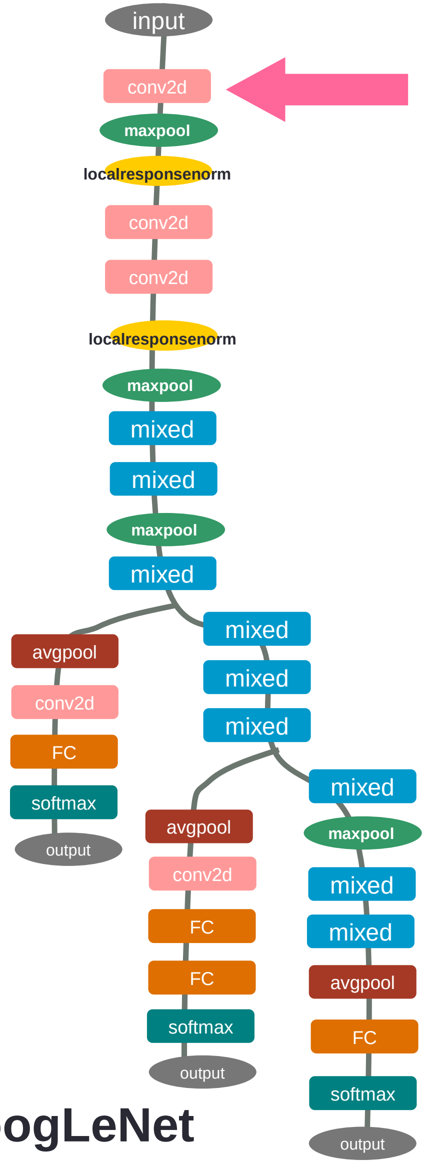




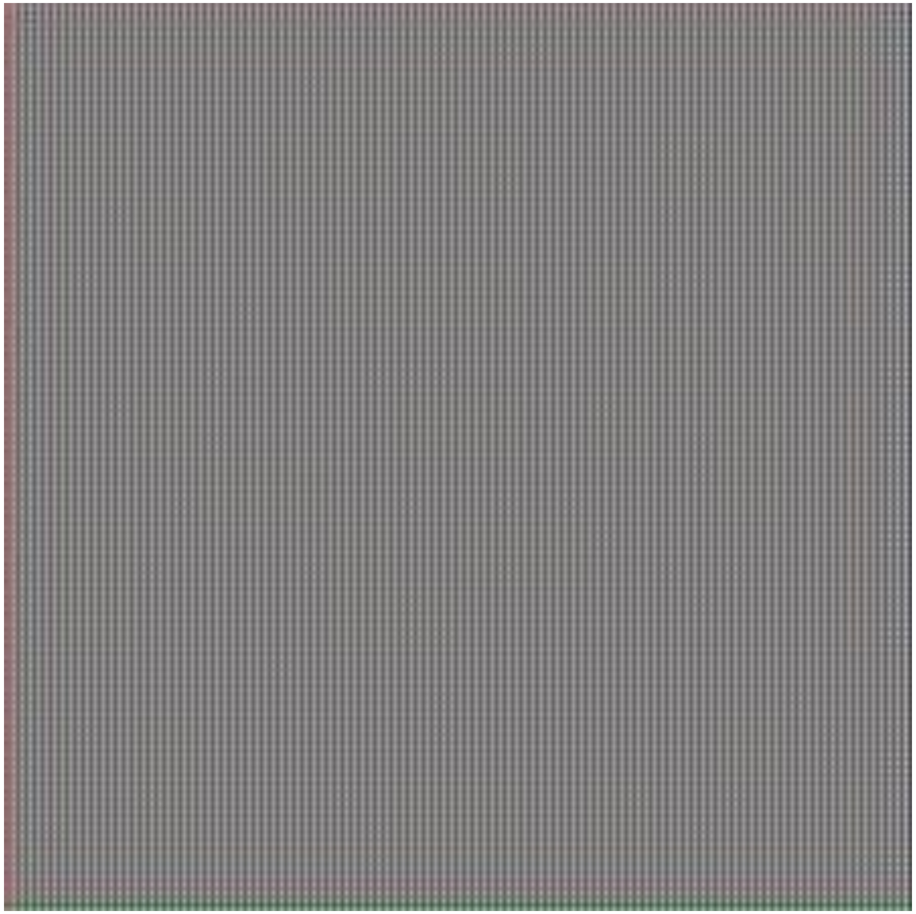
Single neuron activation results



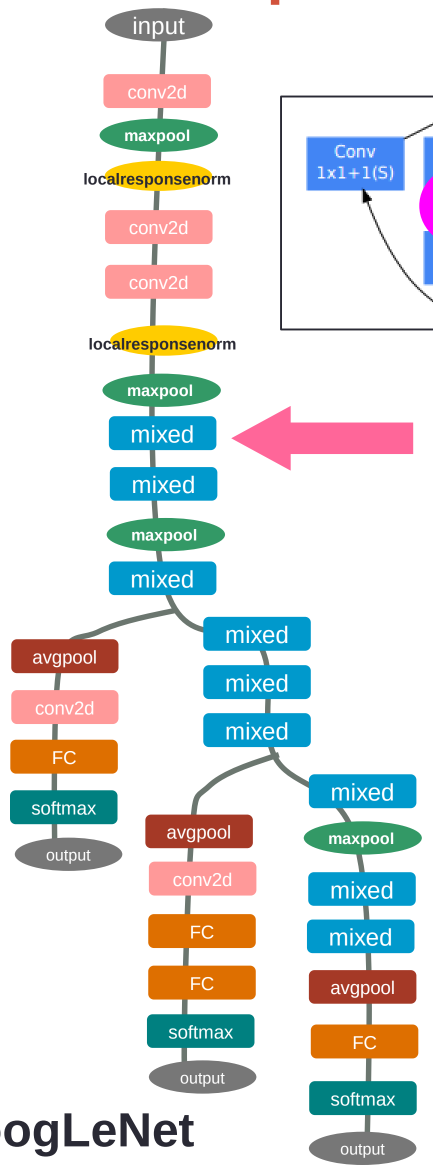
Examples of naïve feature visualization

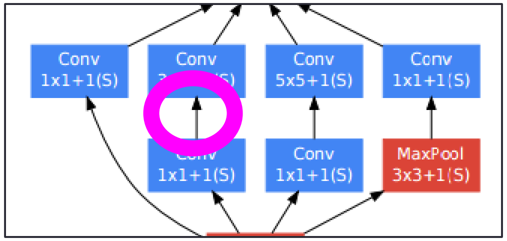


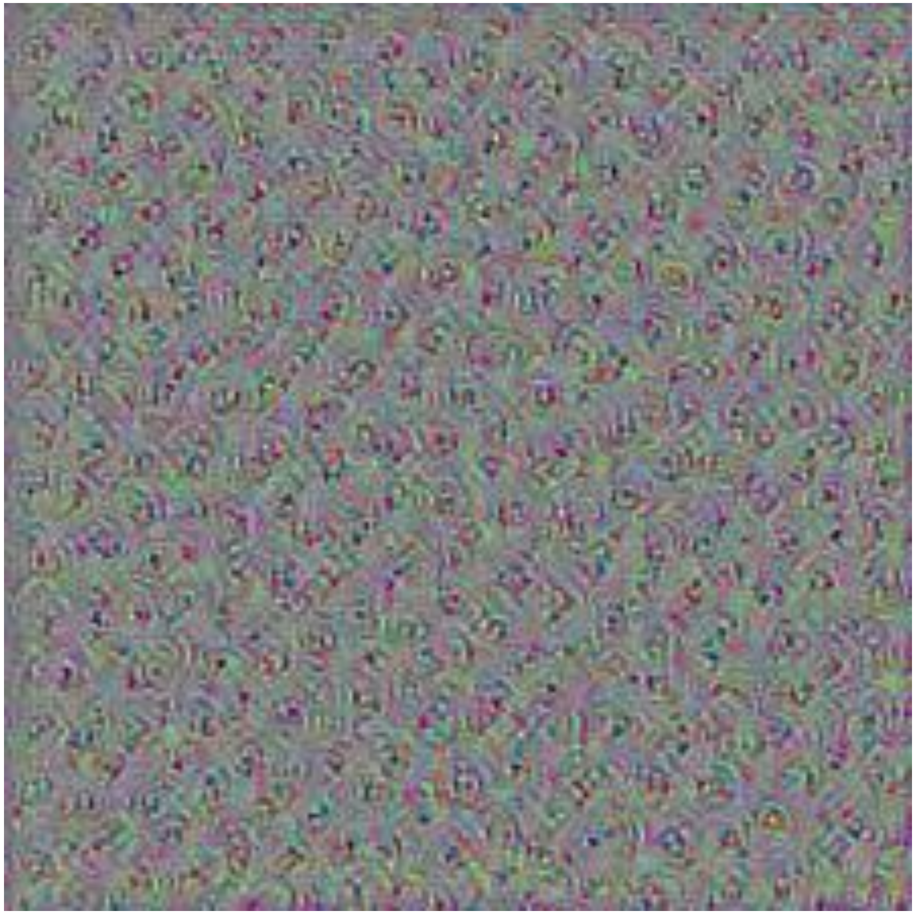
GoogLeNet



Examples of naïve feature visualization

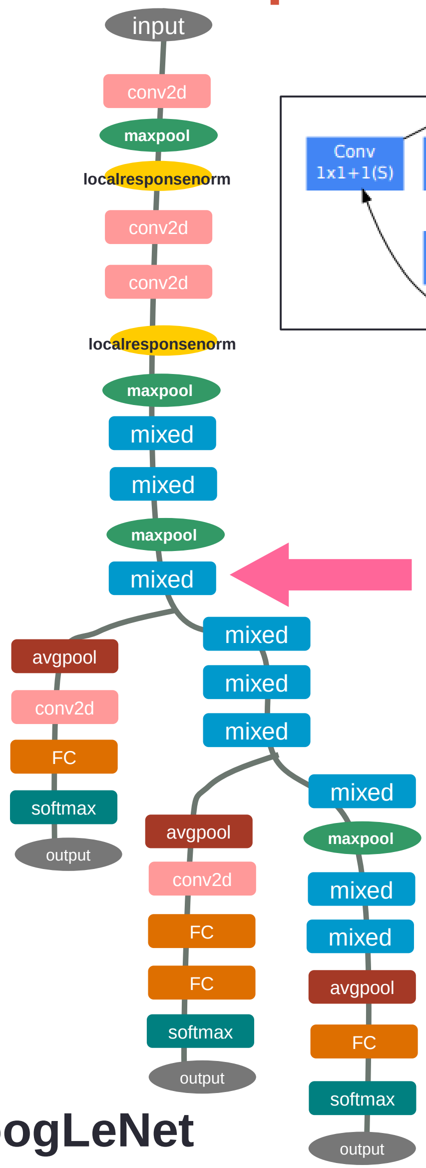


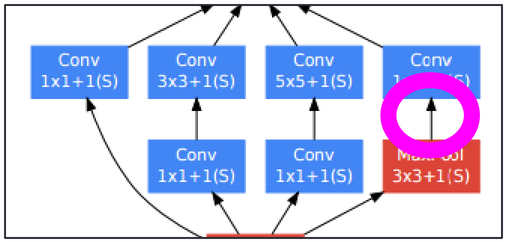




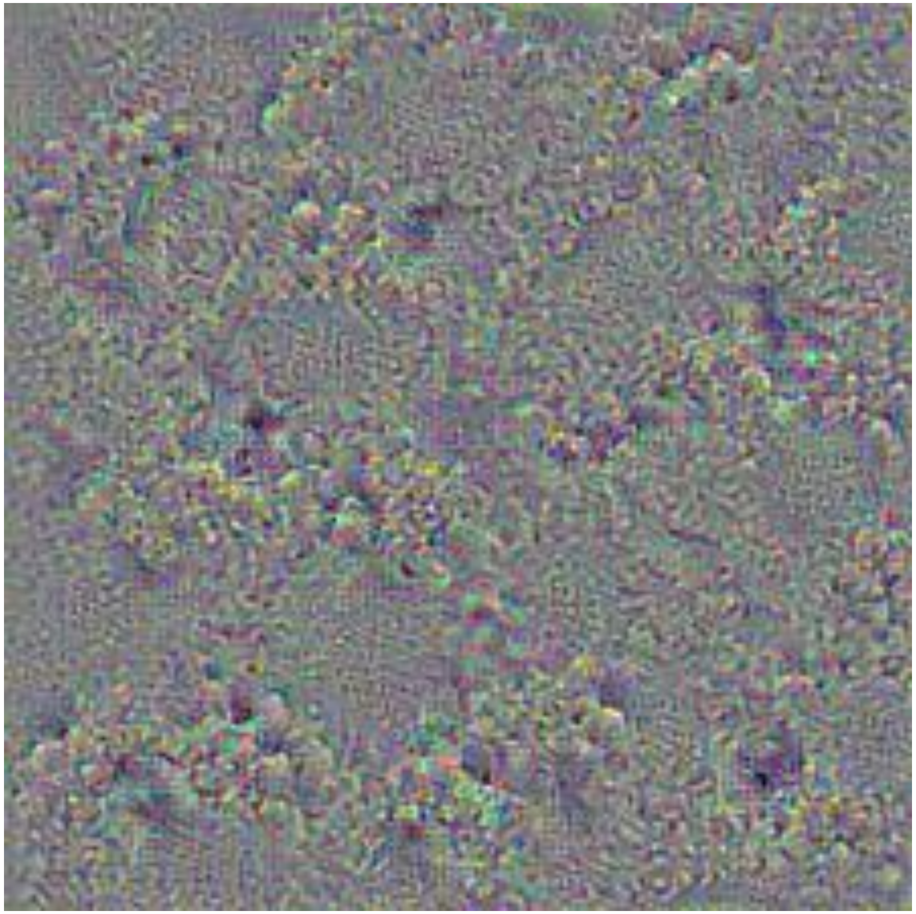
GoogLeNet

Examples of naïve feature visualization

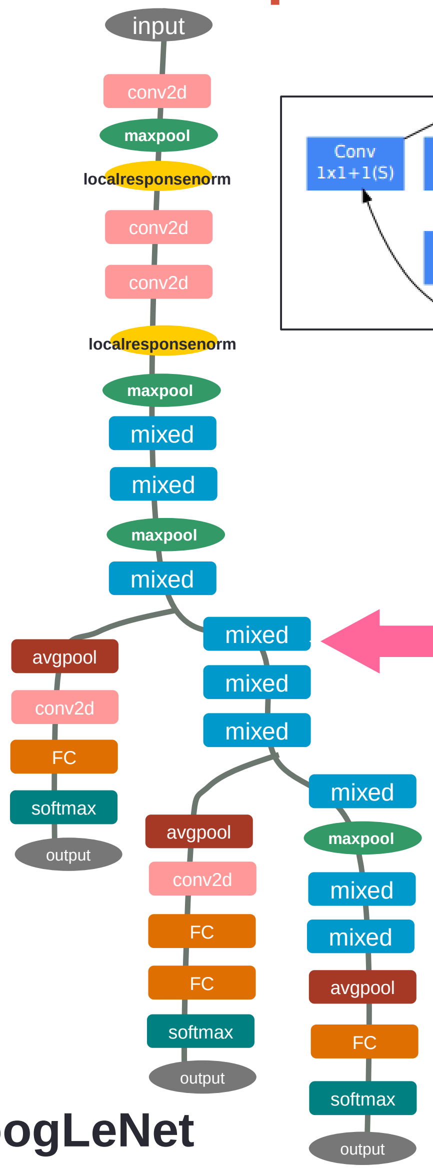


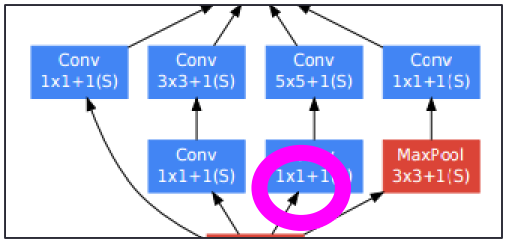


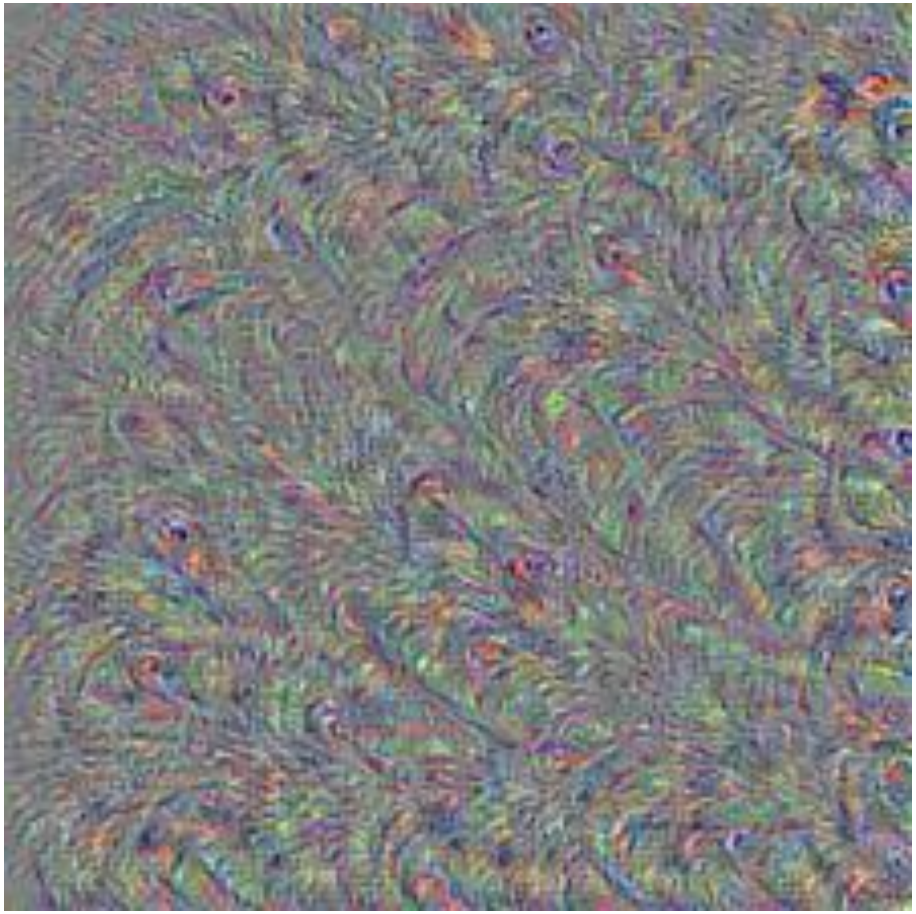
GoogLeNet



Examples of naïve feature visualization

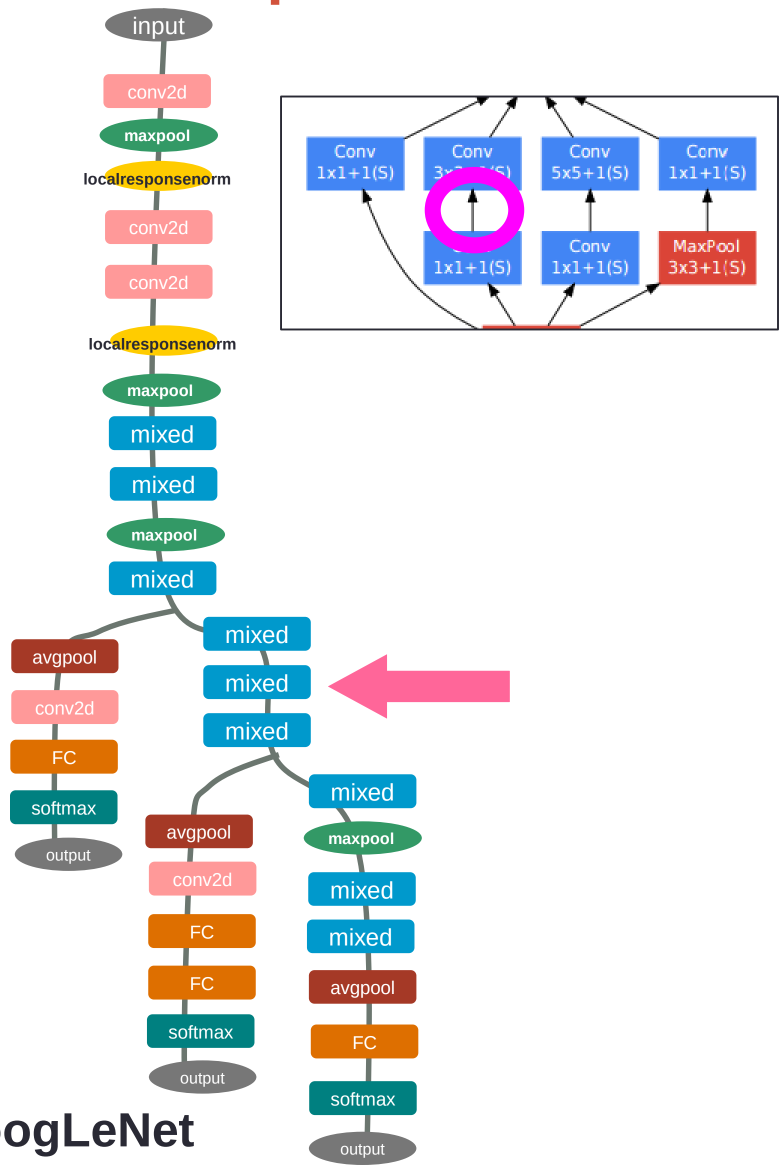


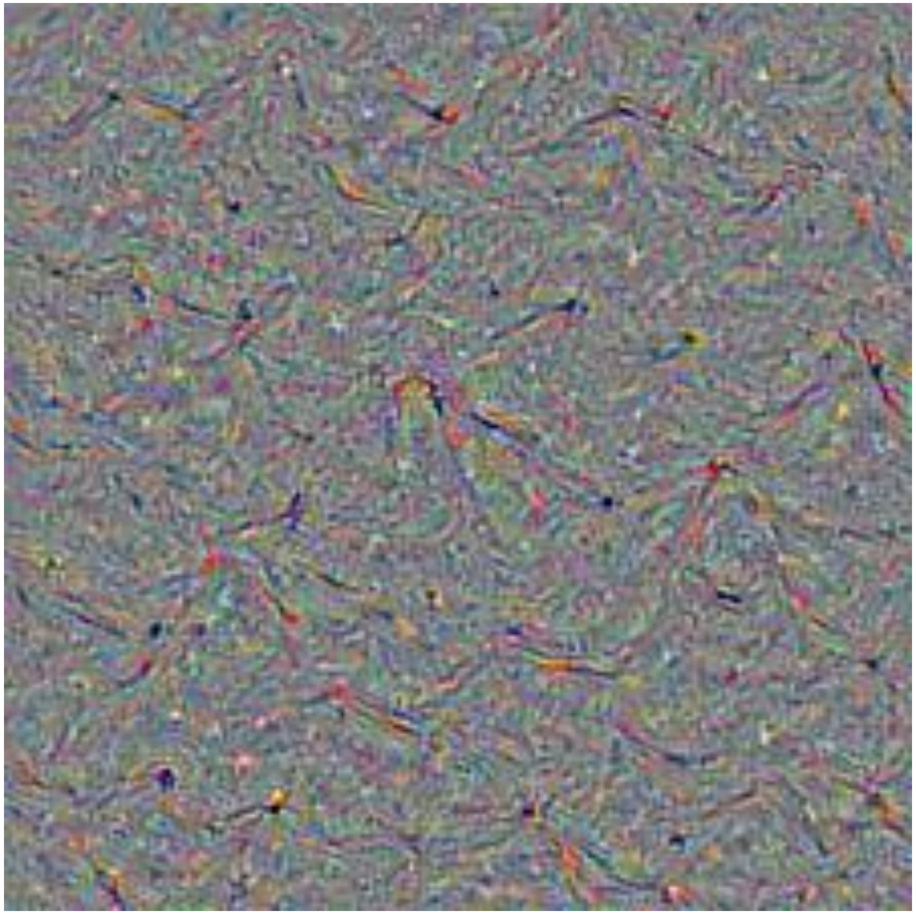




GoogLeNet

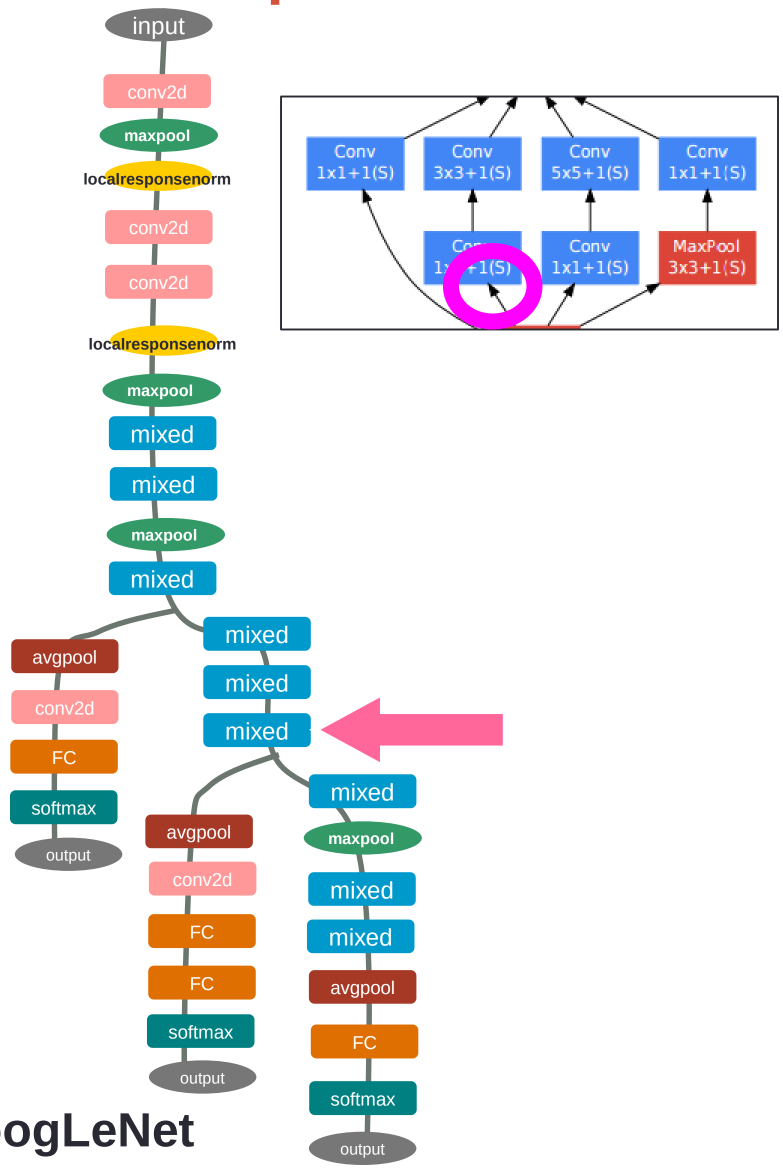
Examples of naïve feature visualization



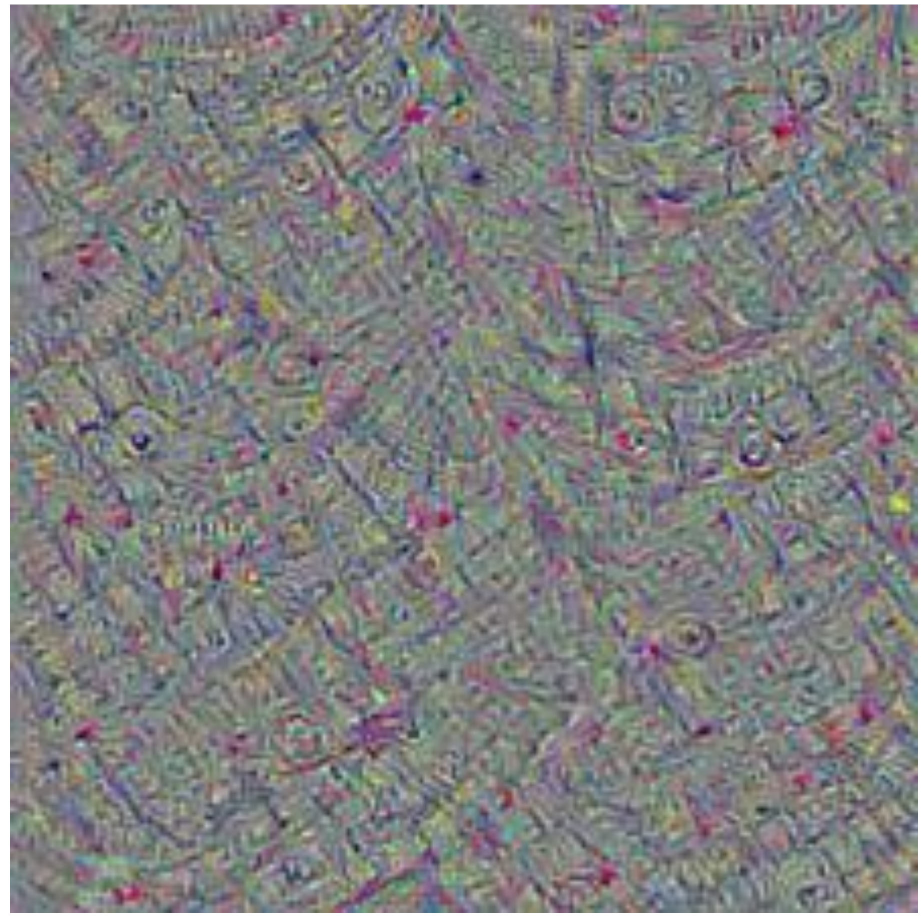


GoogLeNet

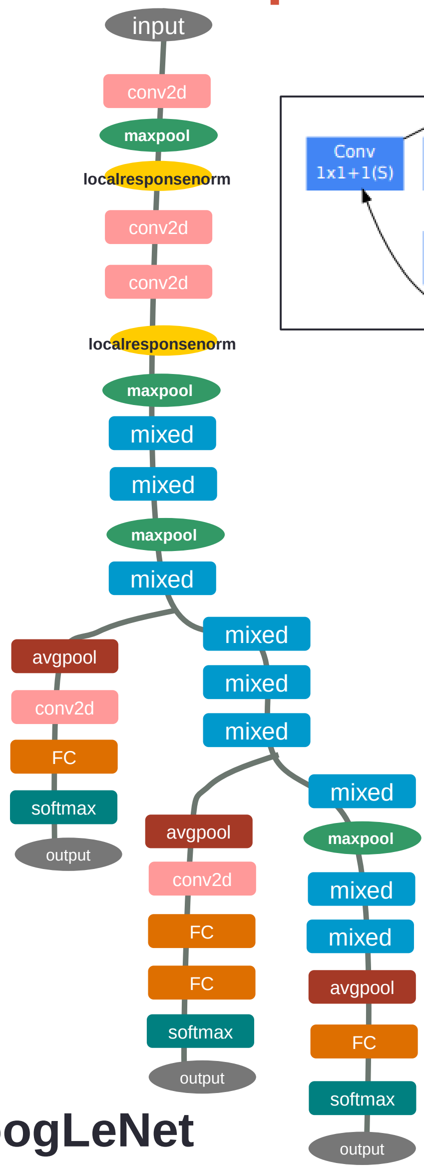
Examples of naïve feature visualization

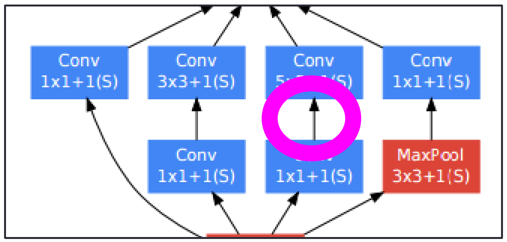


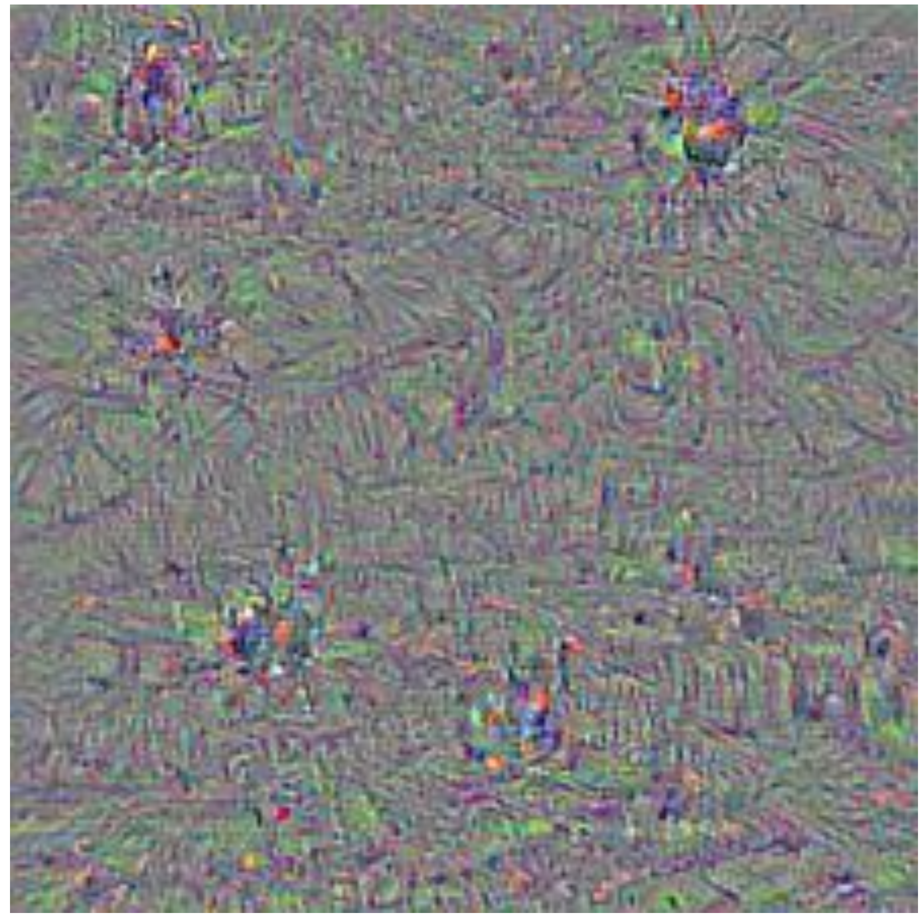
GoogLeNet



Examples of naïve feature visualization

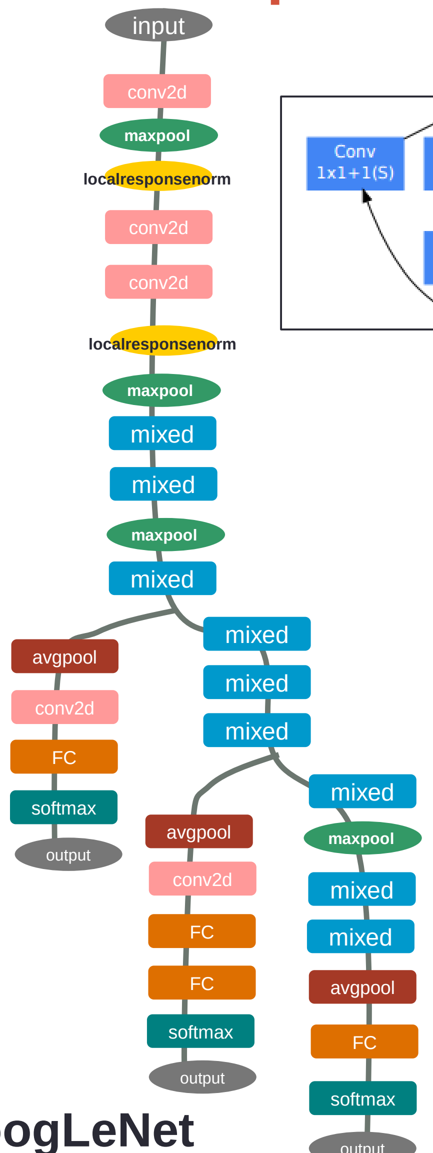


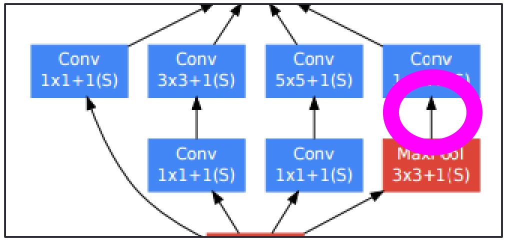


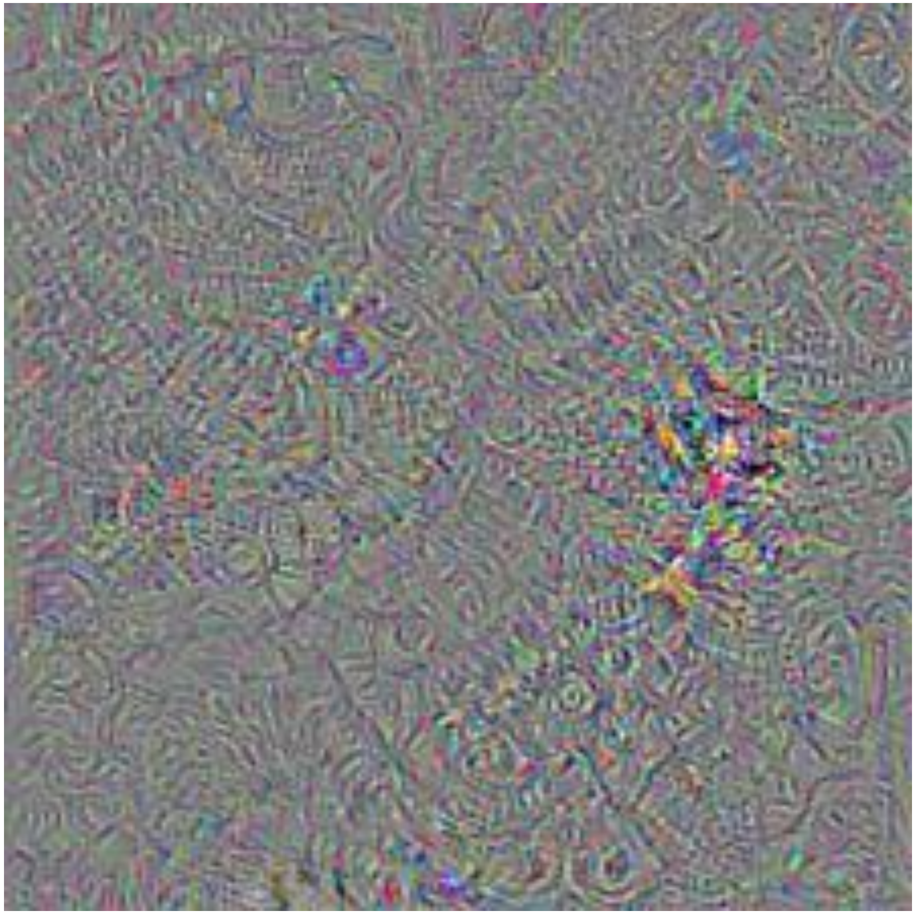


GoogLeNet

Examples of naïve feature visualization

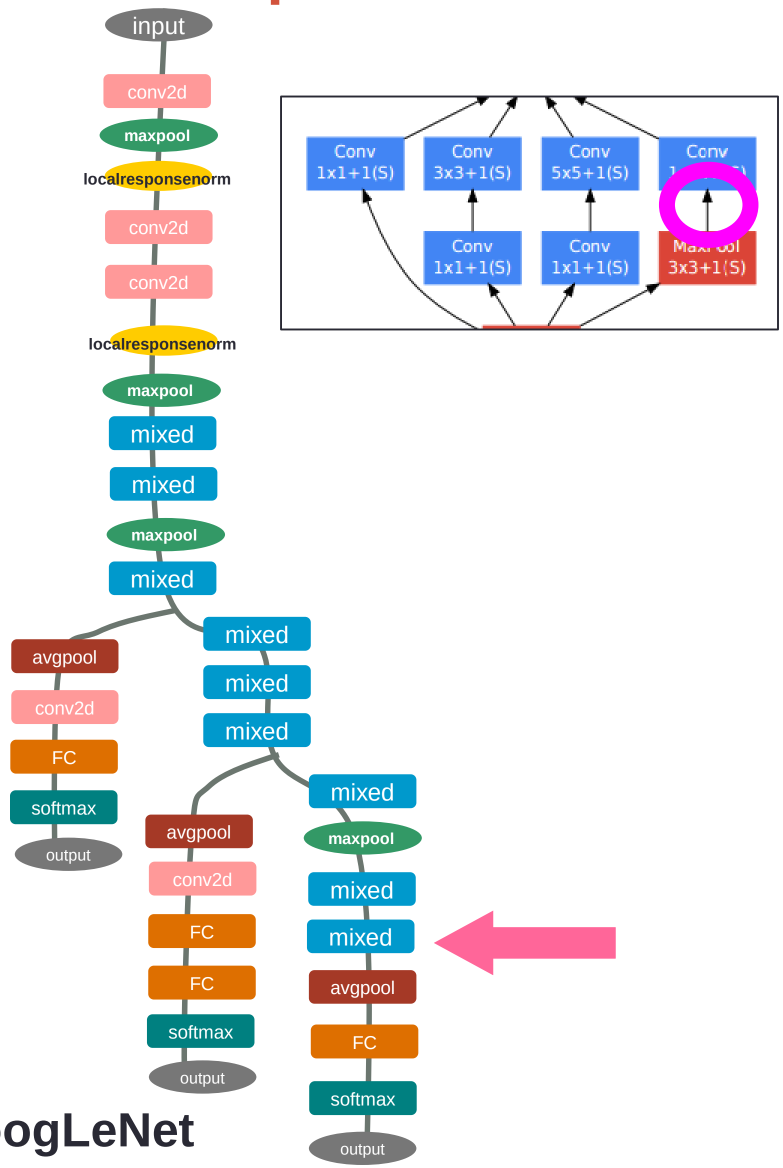


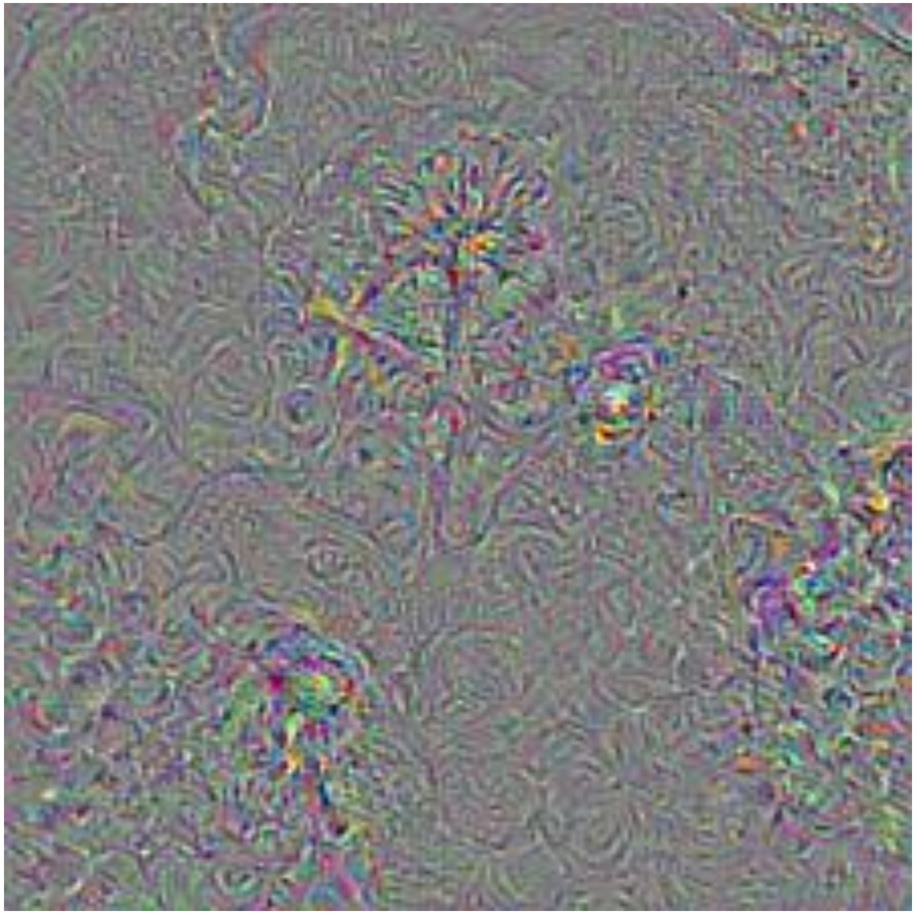




GoogLeNet

Examples of naïve feature visualization.

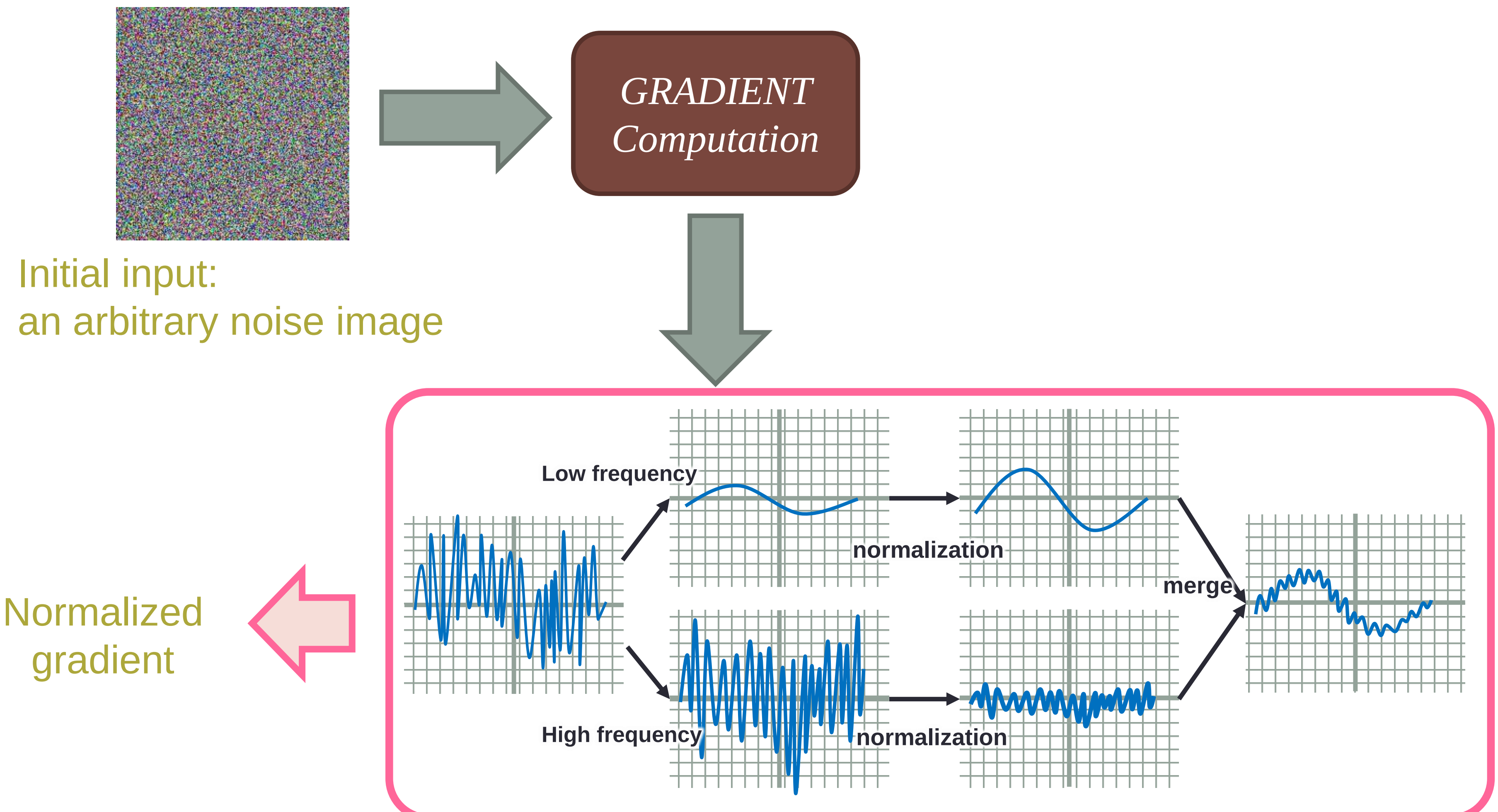




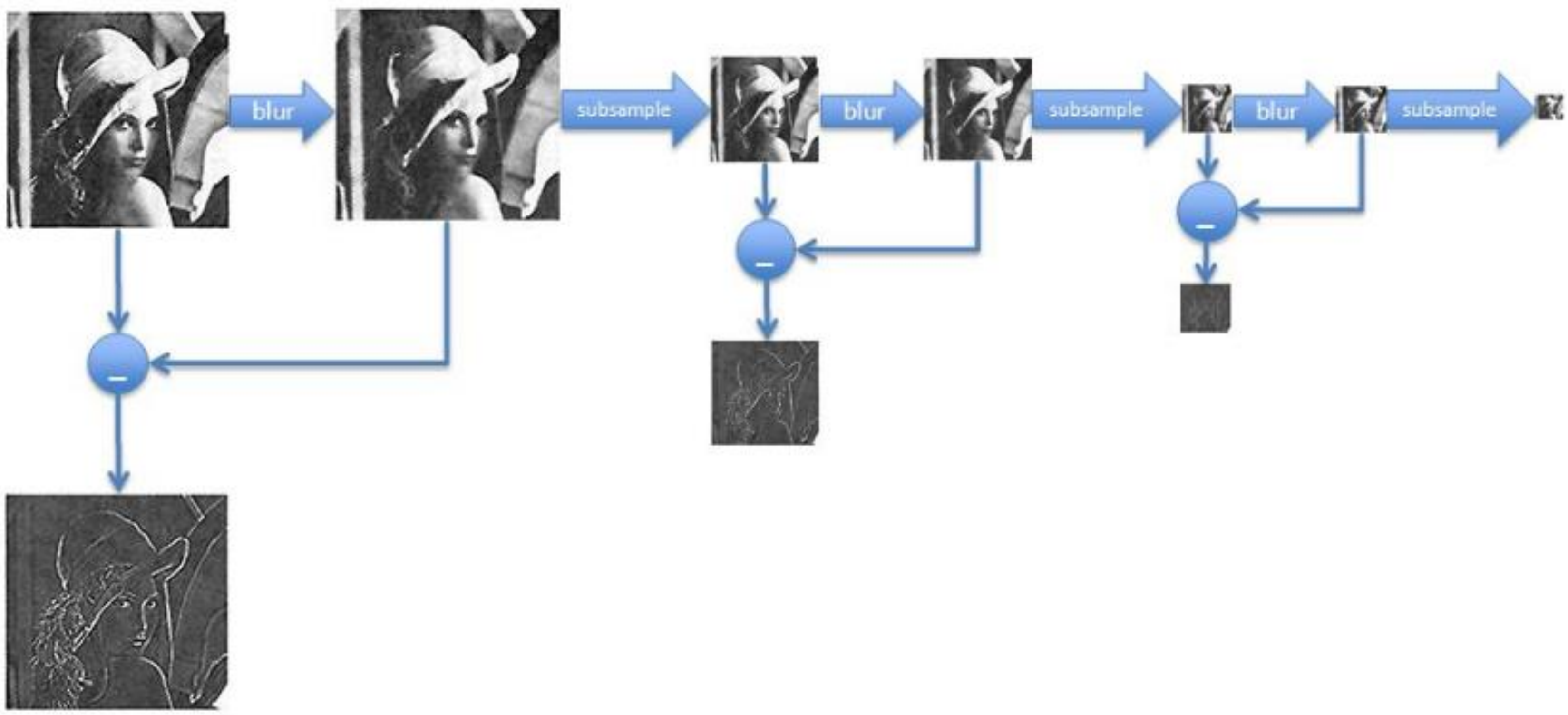
GoogLeNet

LOW/HIGH FREQUENCY NORMALIZATION

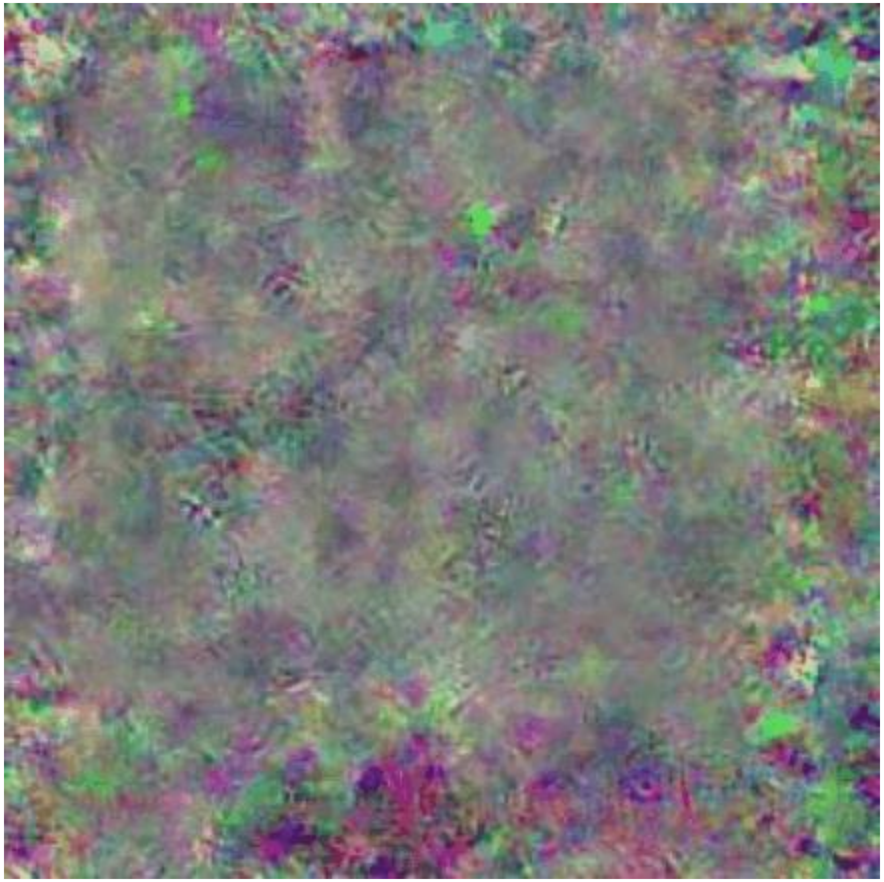
Gradient normalization



Laplacian pyramid



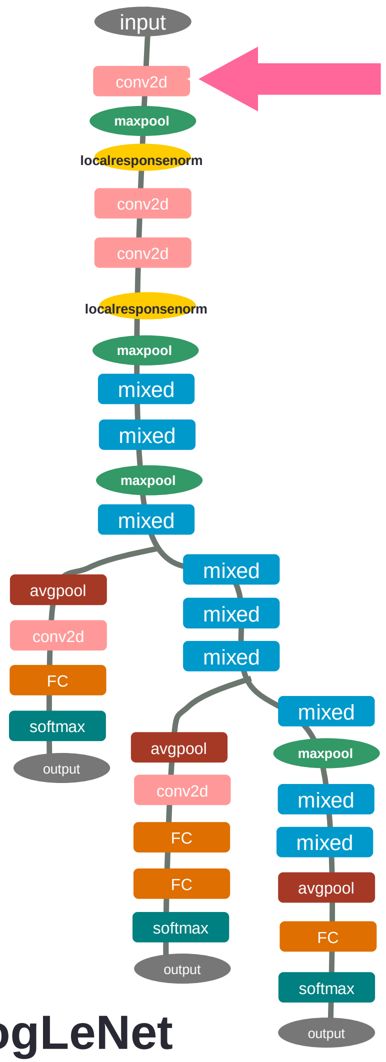
Convergence example



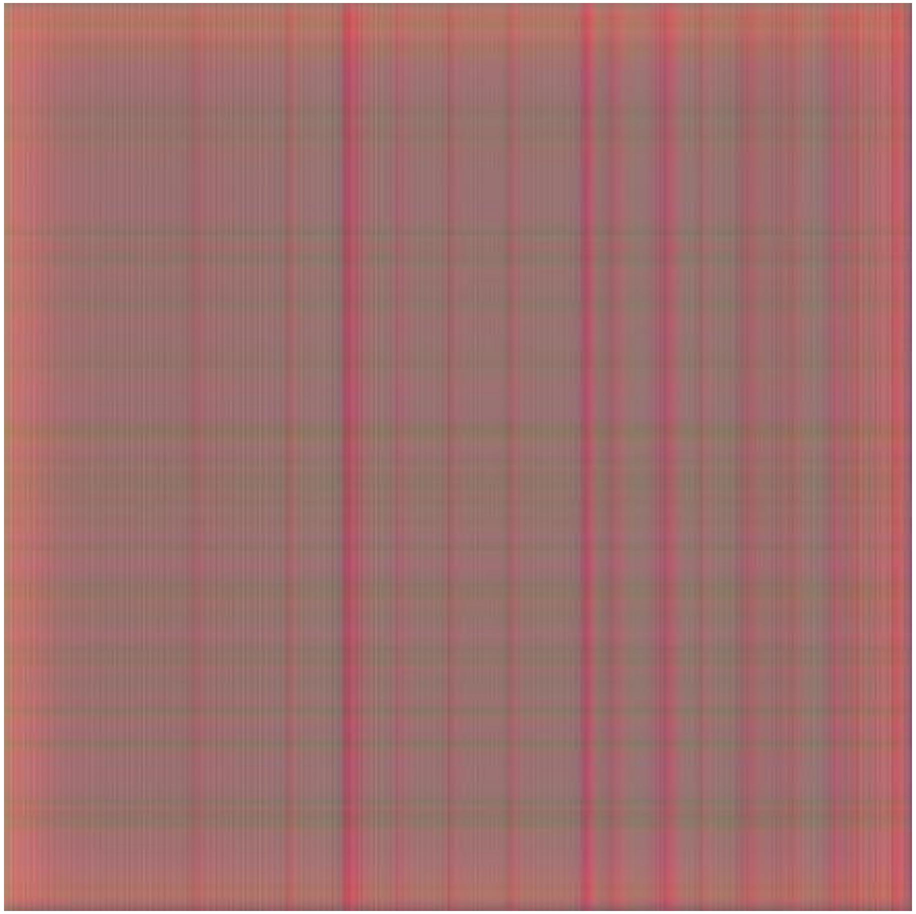
L=102.42

http://storage.googleapis.com/deepdream/visualz/tensorflow_inception/index.html

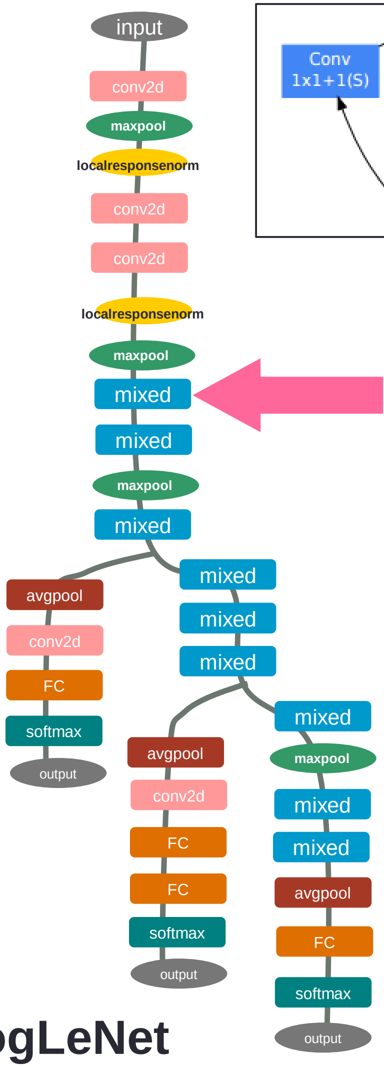
Result of Laplacian pyramid method



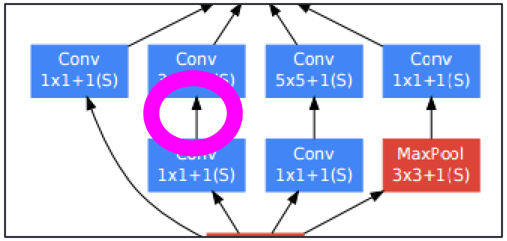
GoogLeNet

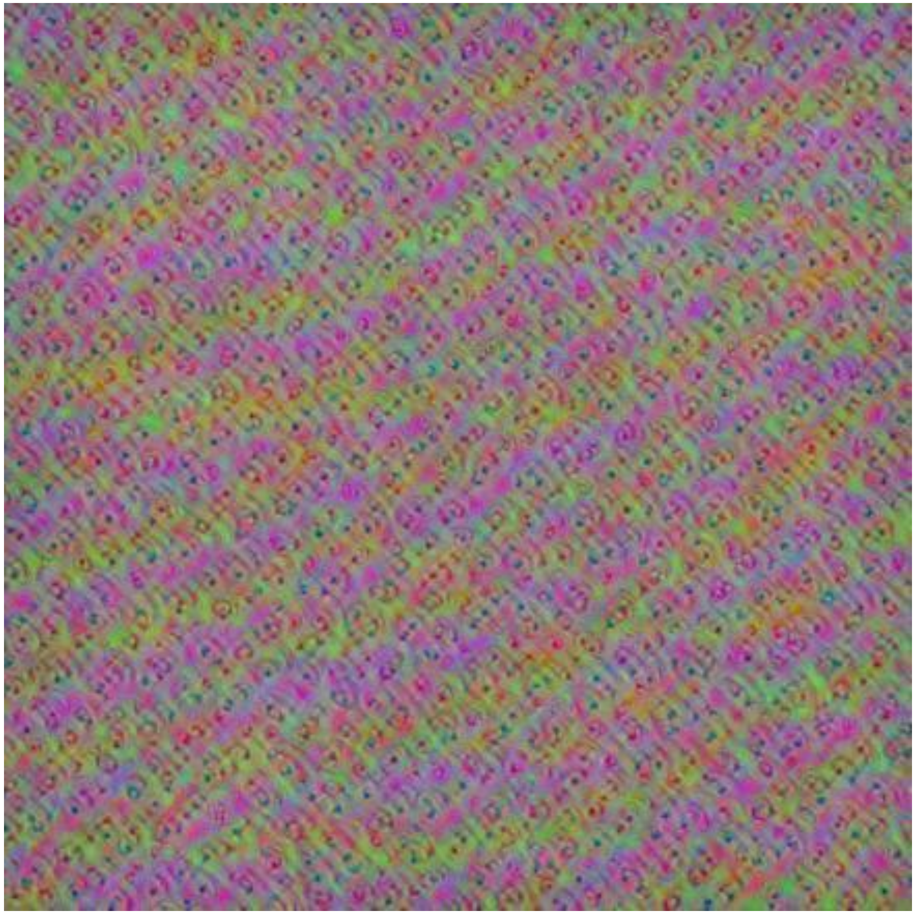


Result of Laplacian pyramid method

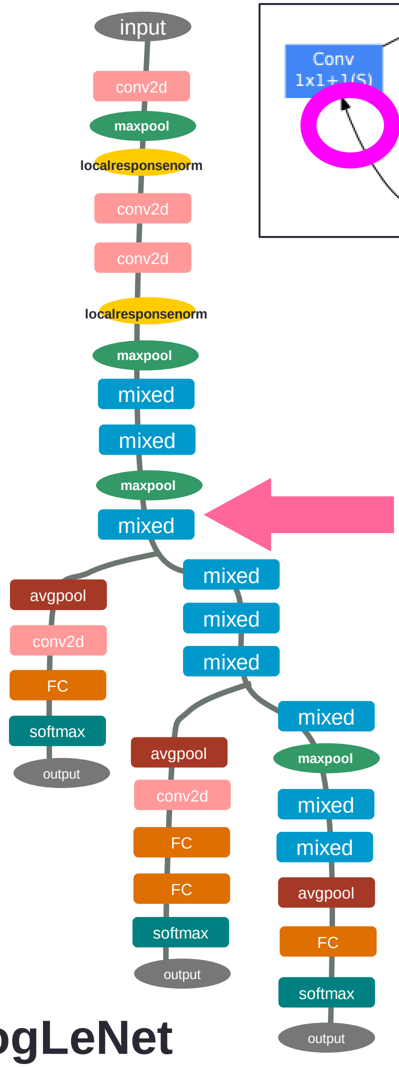


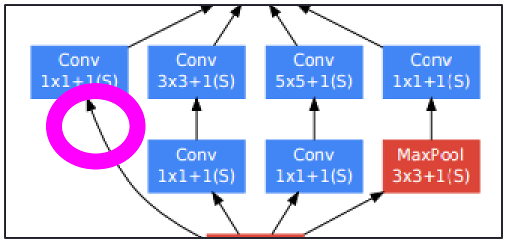
GoogLeNet

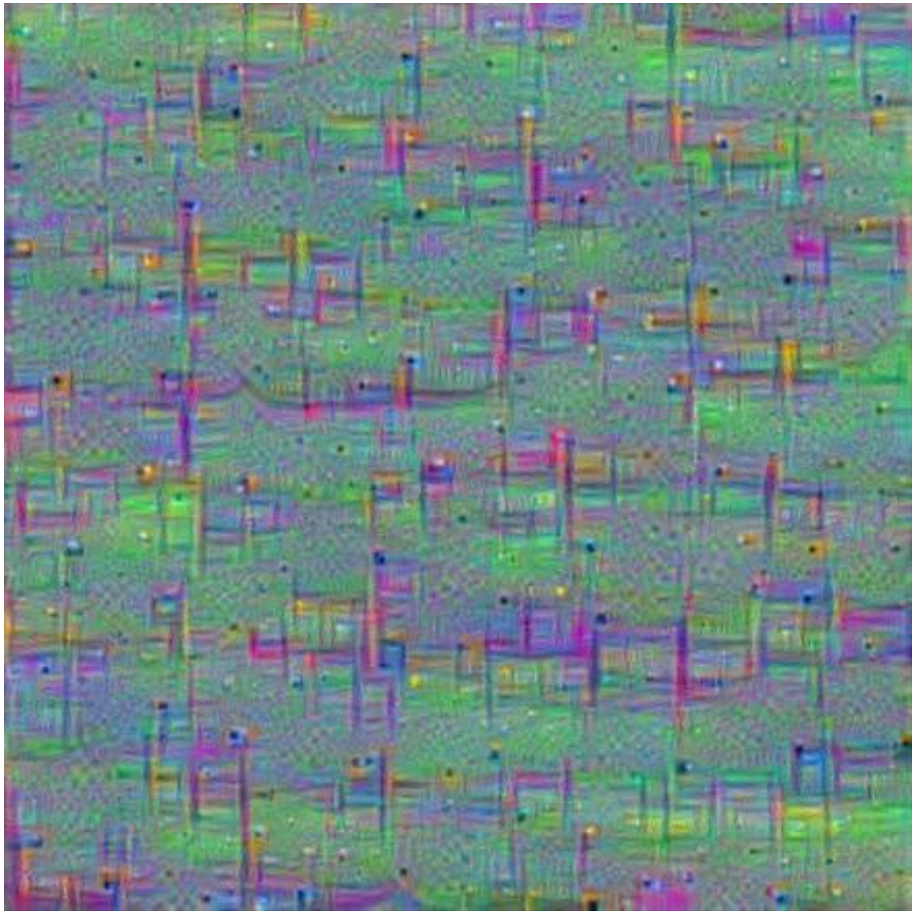




Result of Laplacian pyramid method

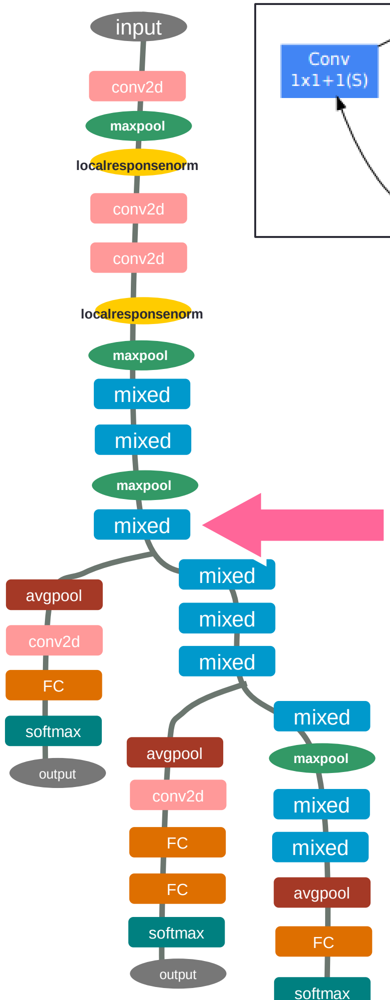




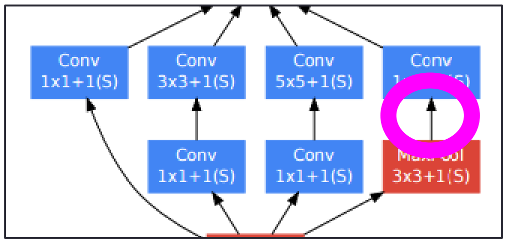


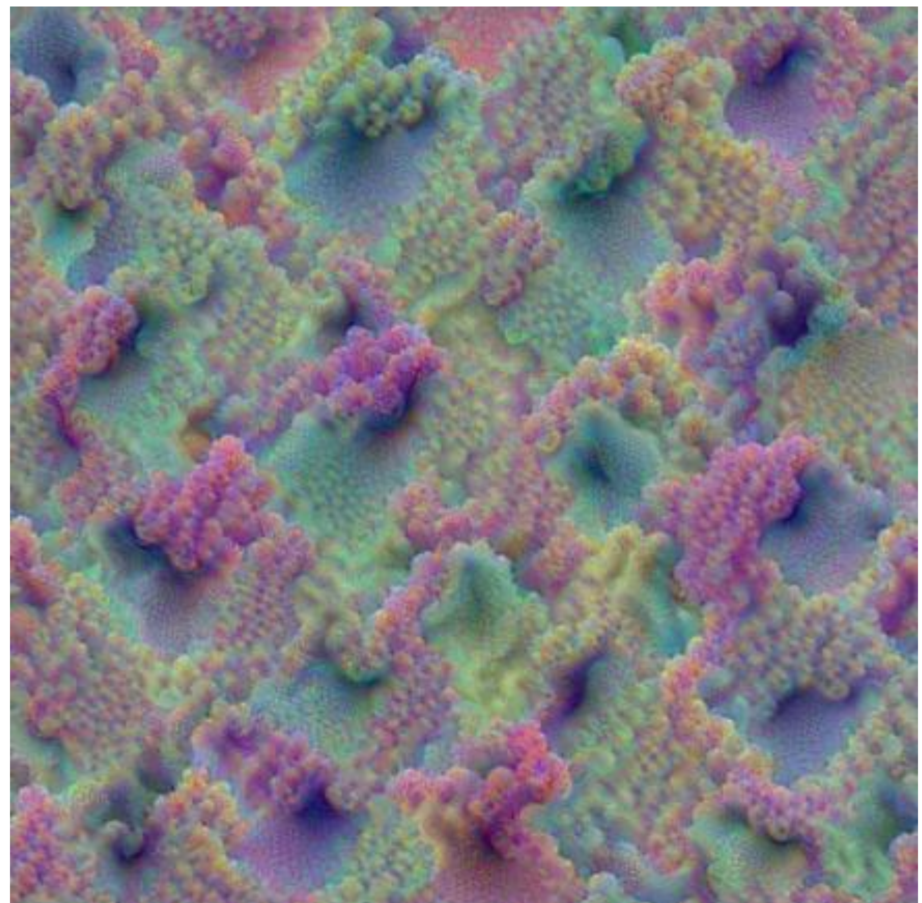
GoogLeNet

Result of Laplacian pyramid method

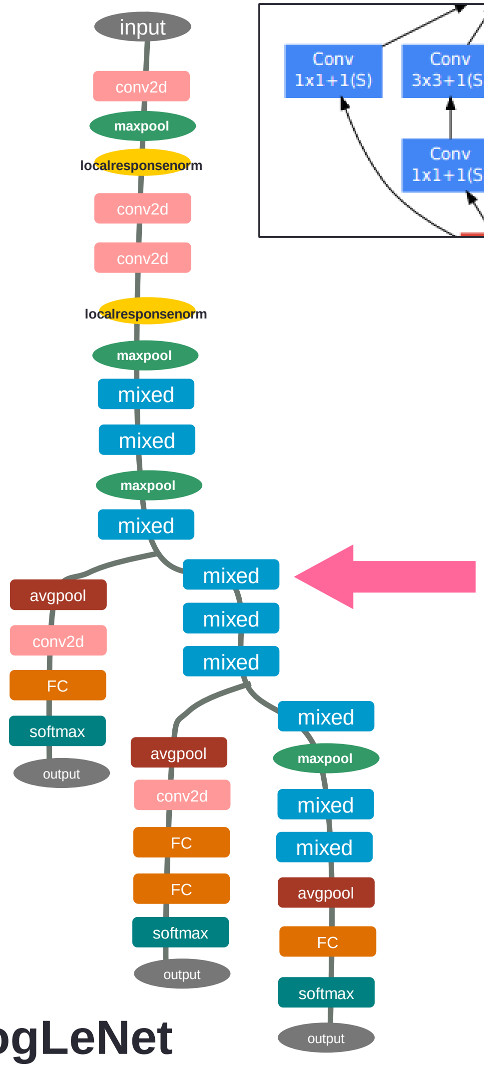


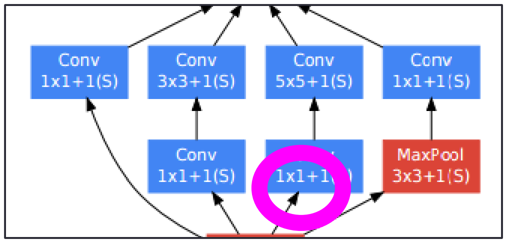
GoogLeNet

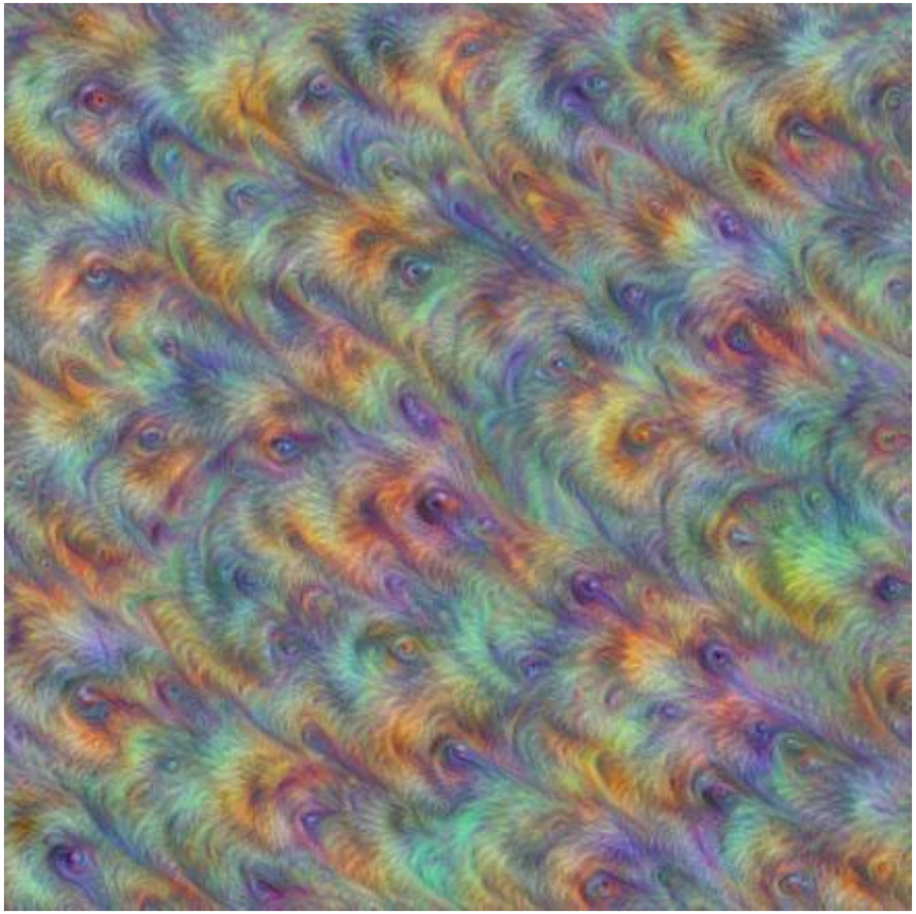




Result of Laplacian pyramid method

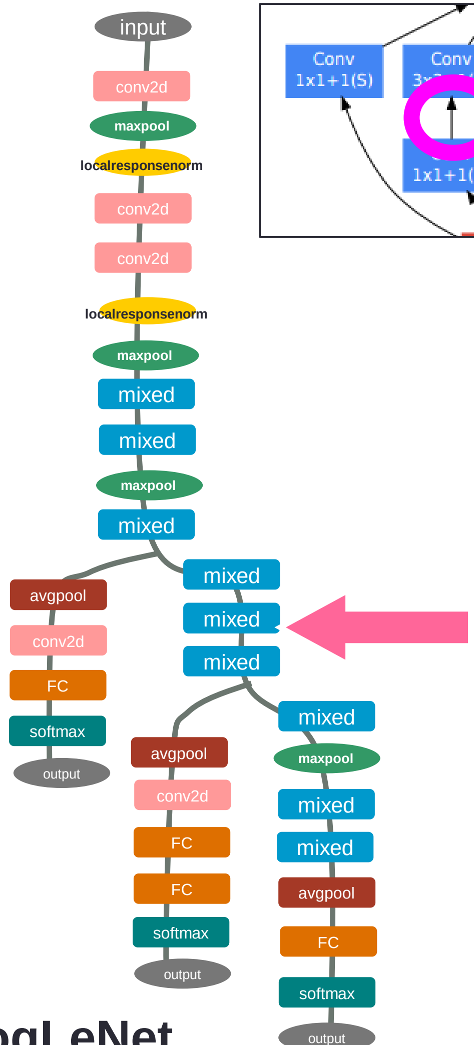




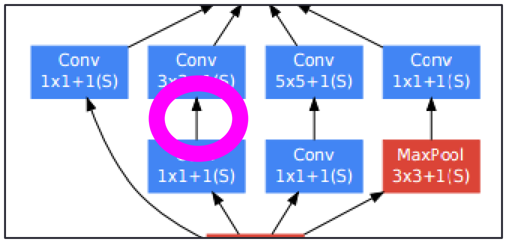


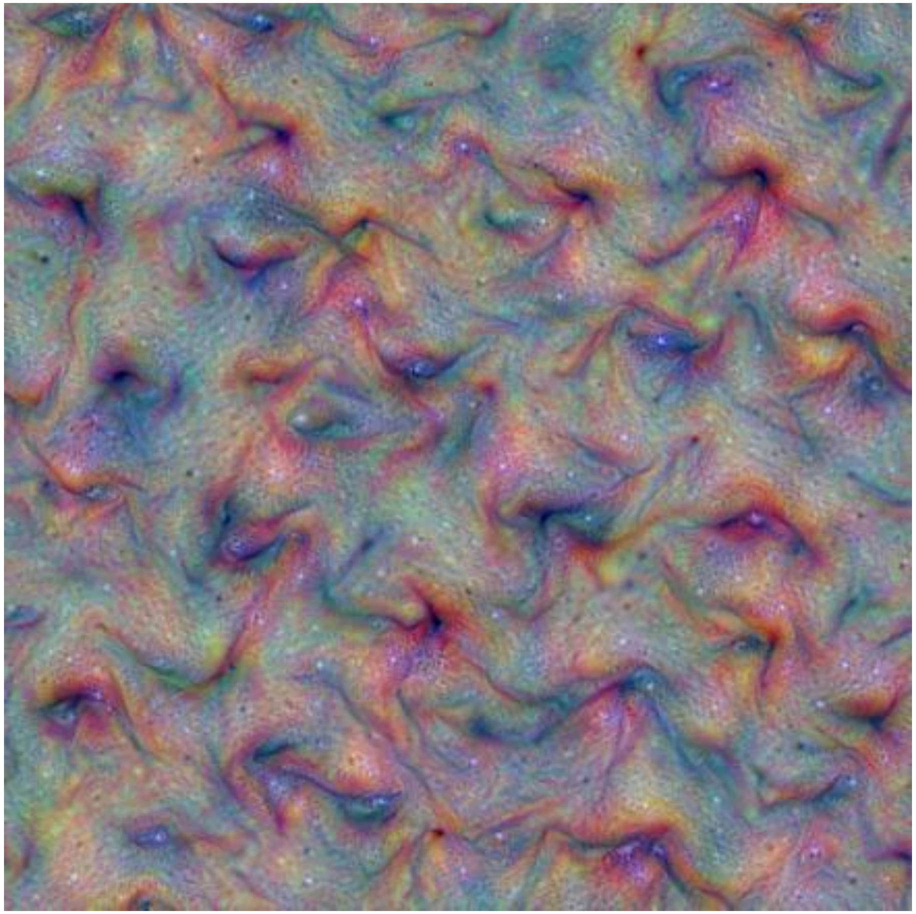
GoogLeNet

Result of Laplacian pyramid method

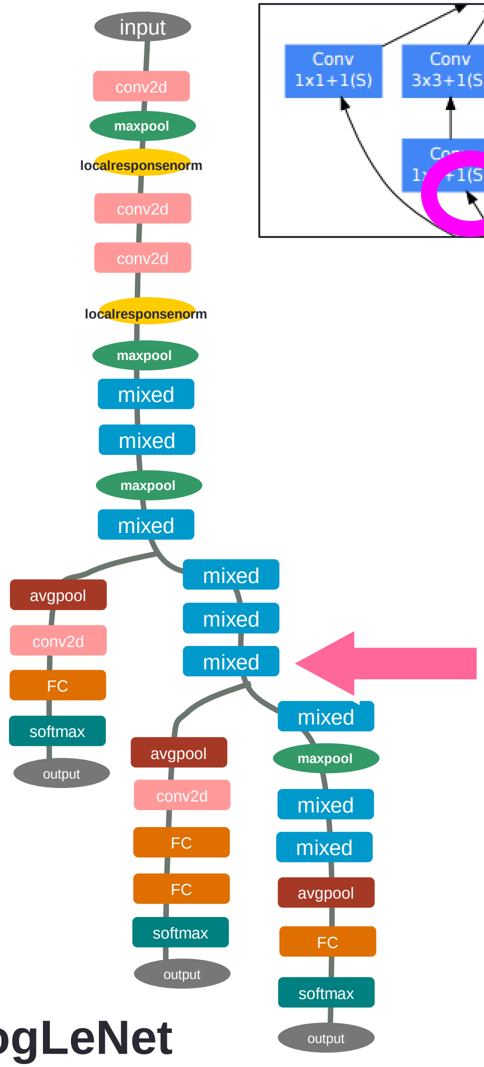


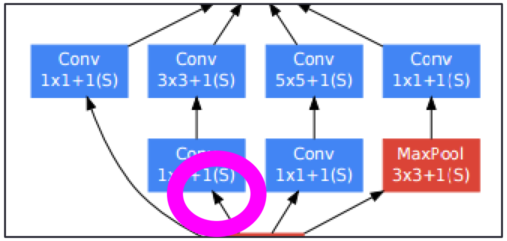
GoogLeNet

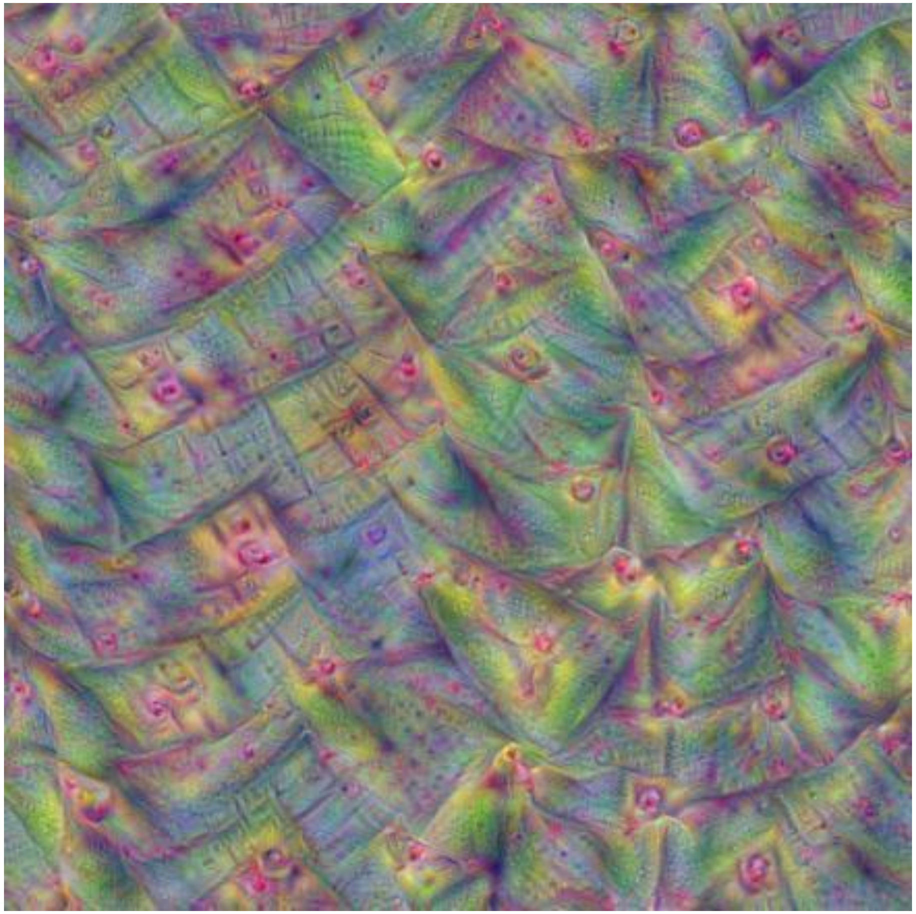




Result of Laplacian pyramid method

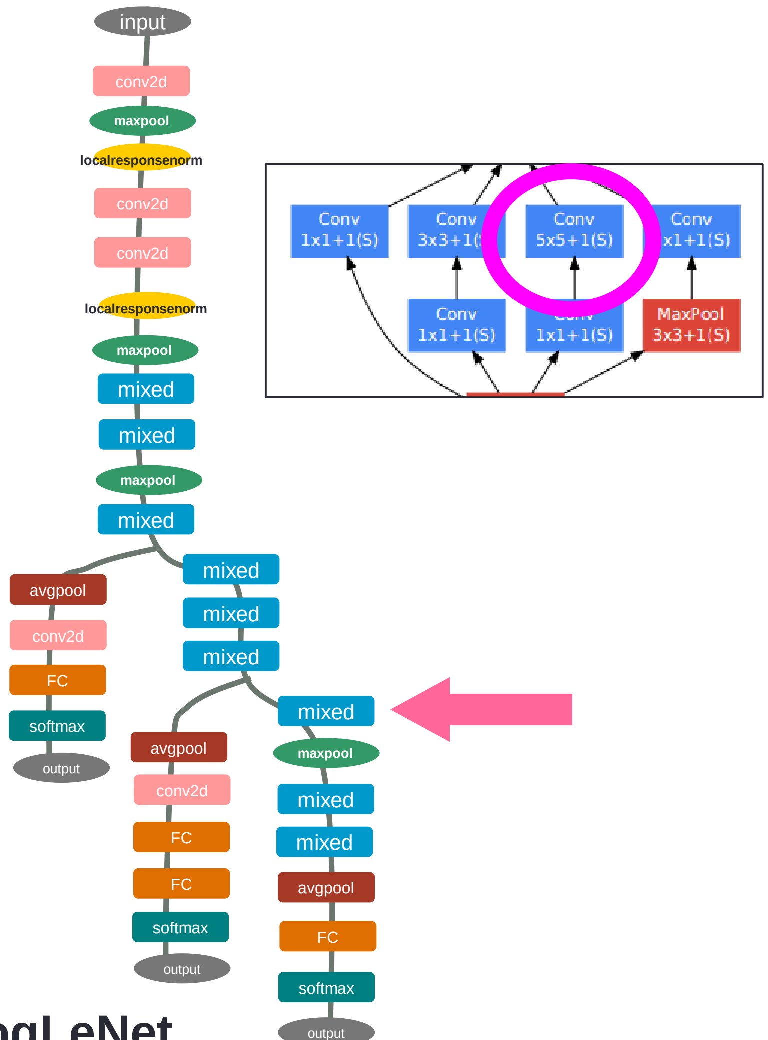




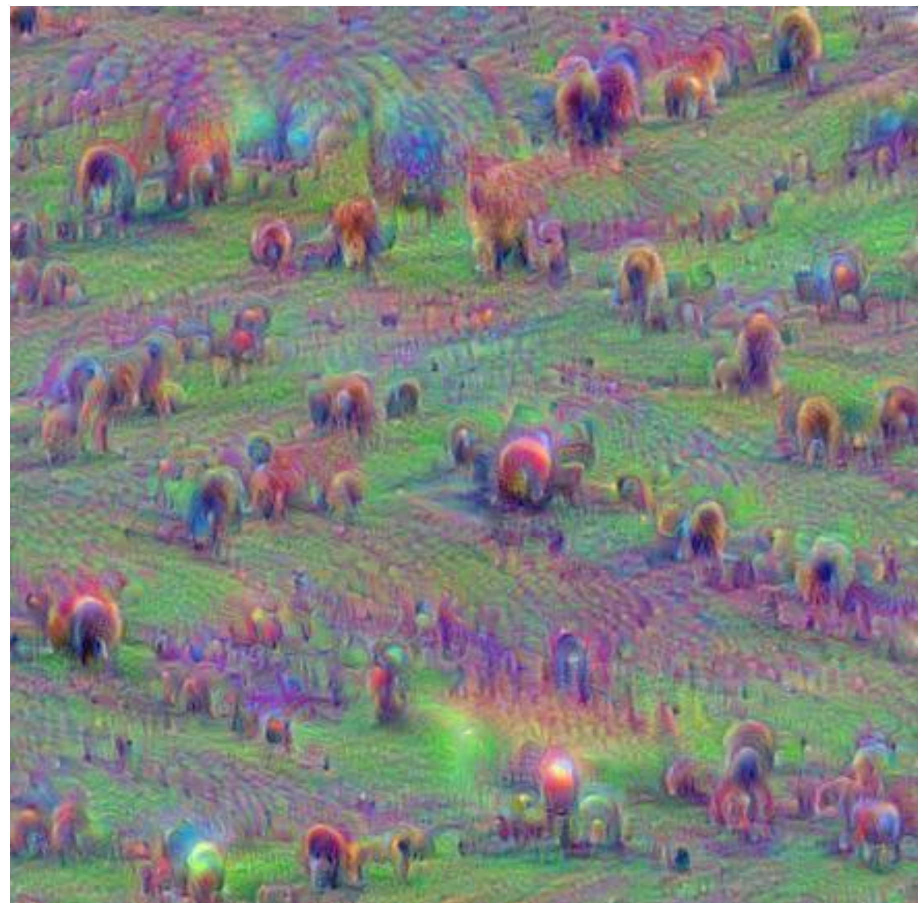


GoogLeNet

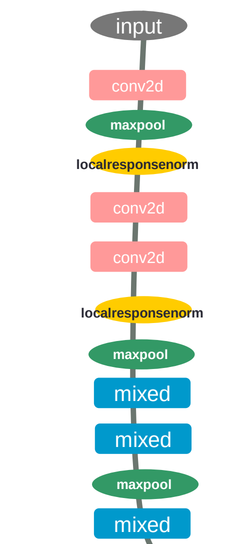
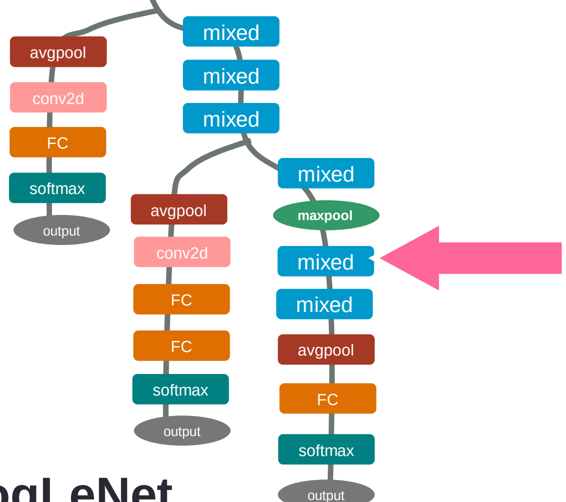
Result of Laplacian pyramid method

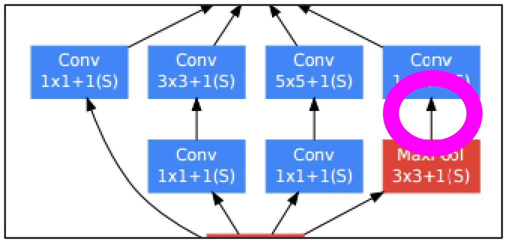


GoogLeNet



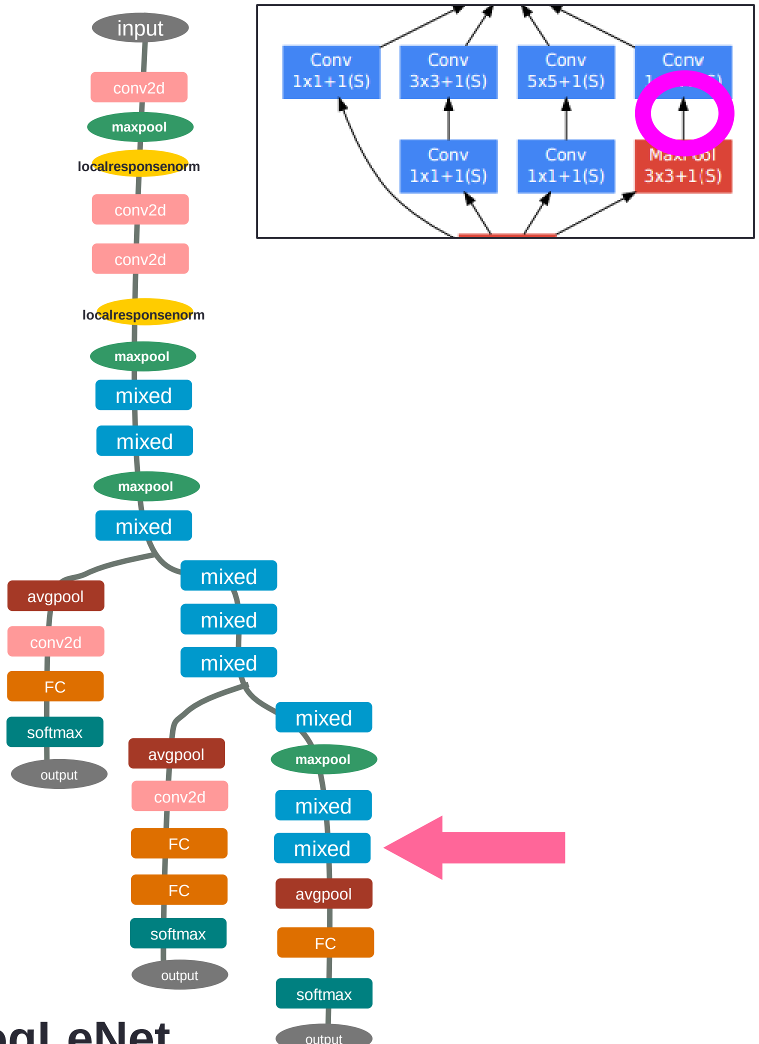
Result of Laplacian pyramid method

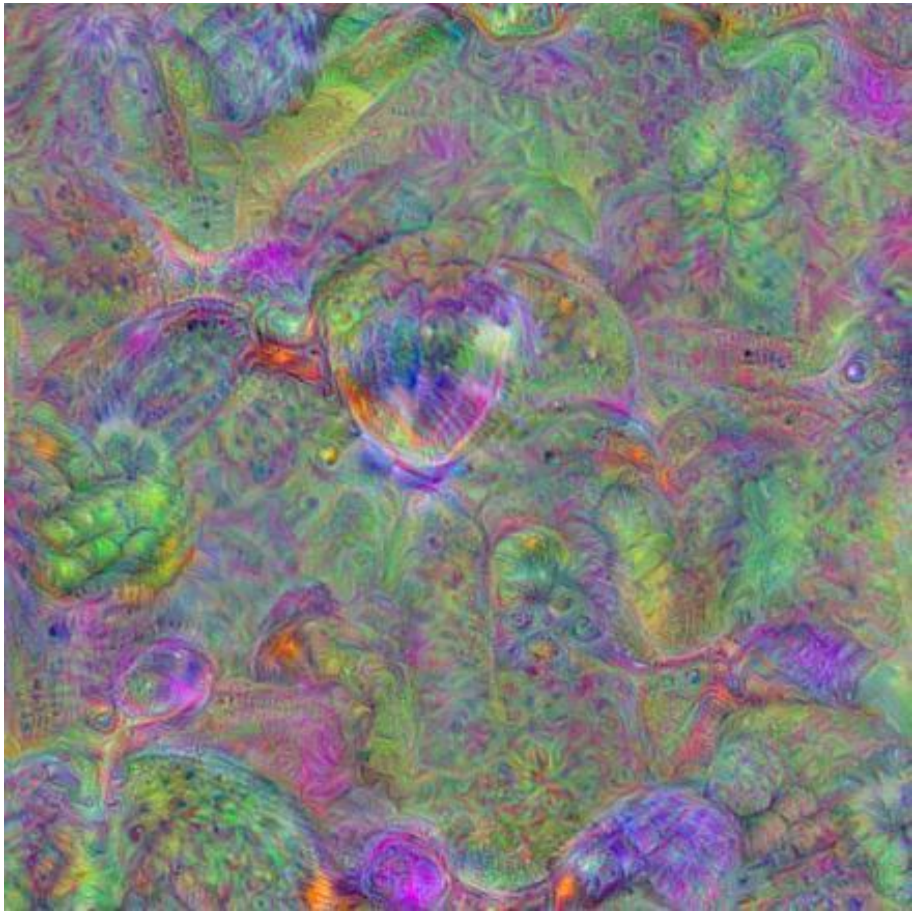





GoogLeNet

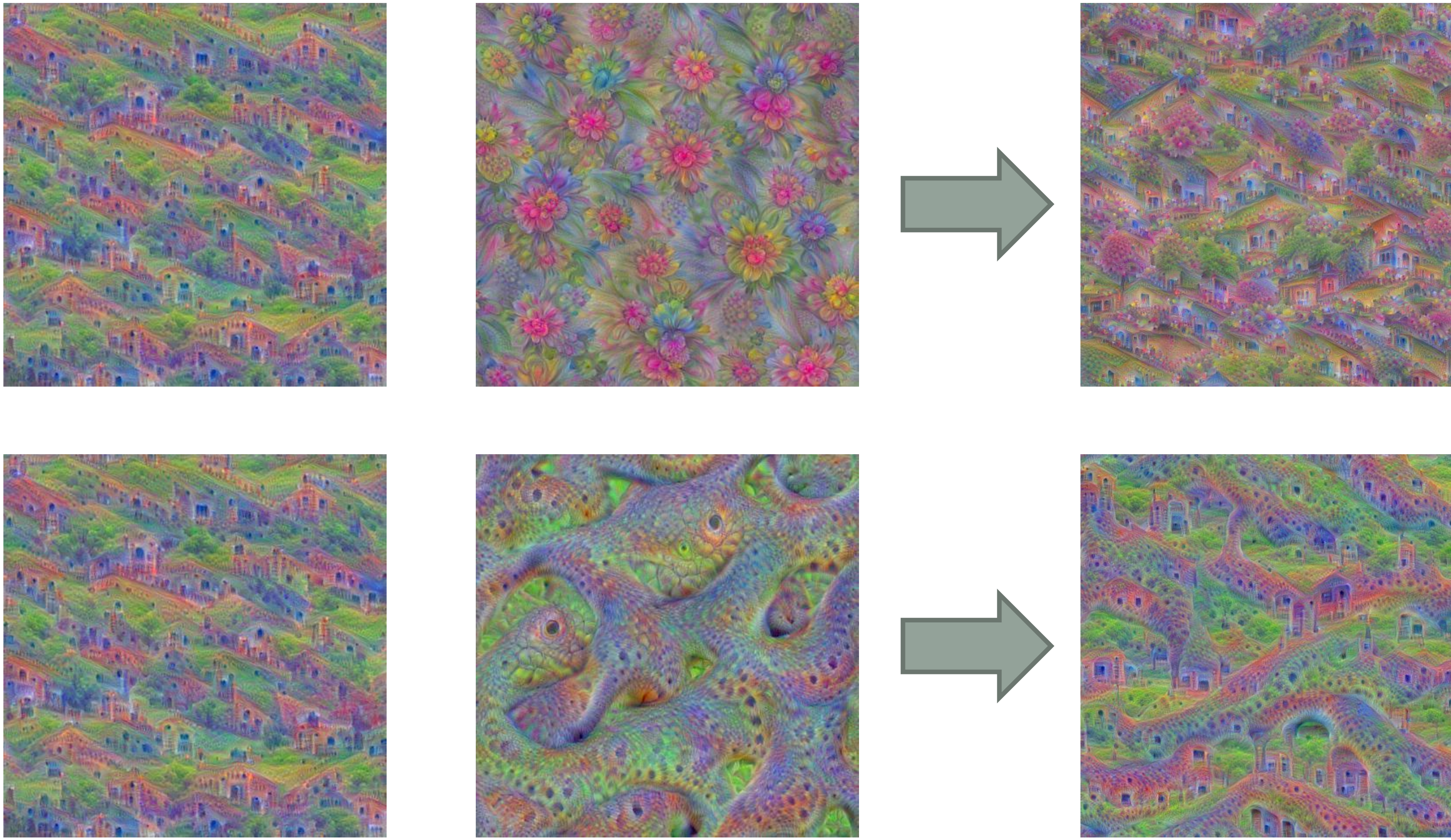
Result of Laplacian pyramid method.



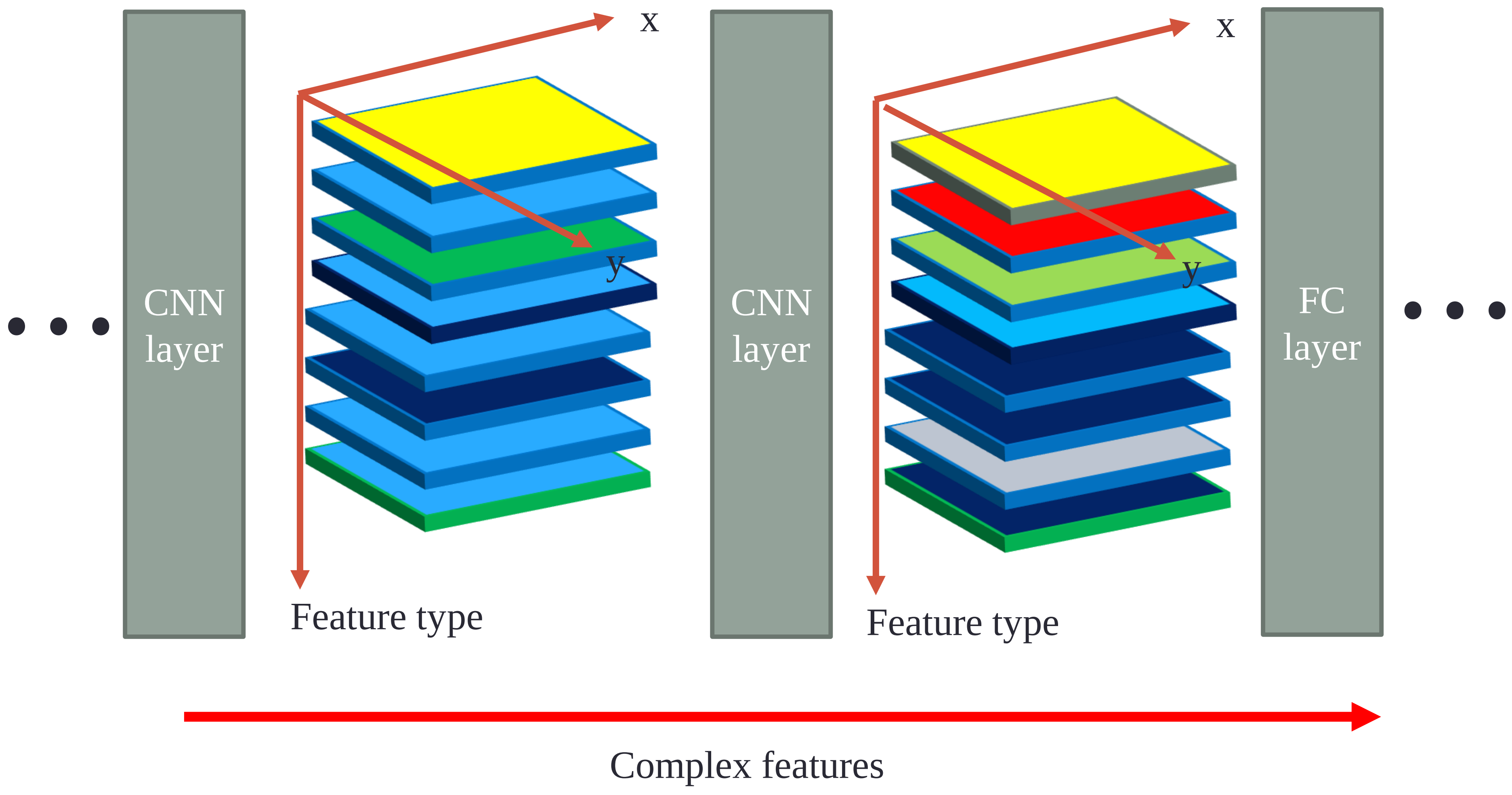


GoogLeNet

Results with two channels



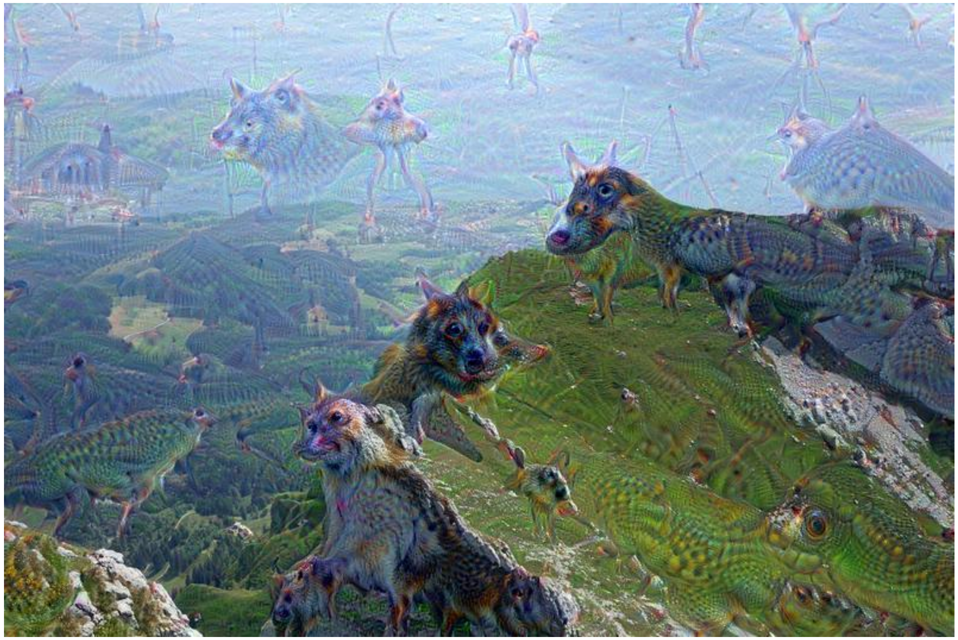
Summary



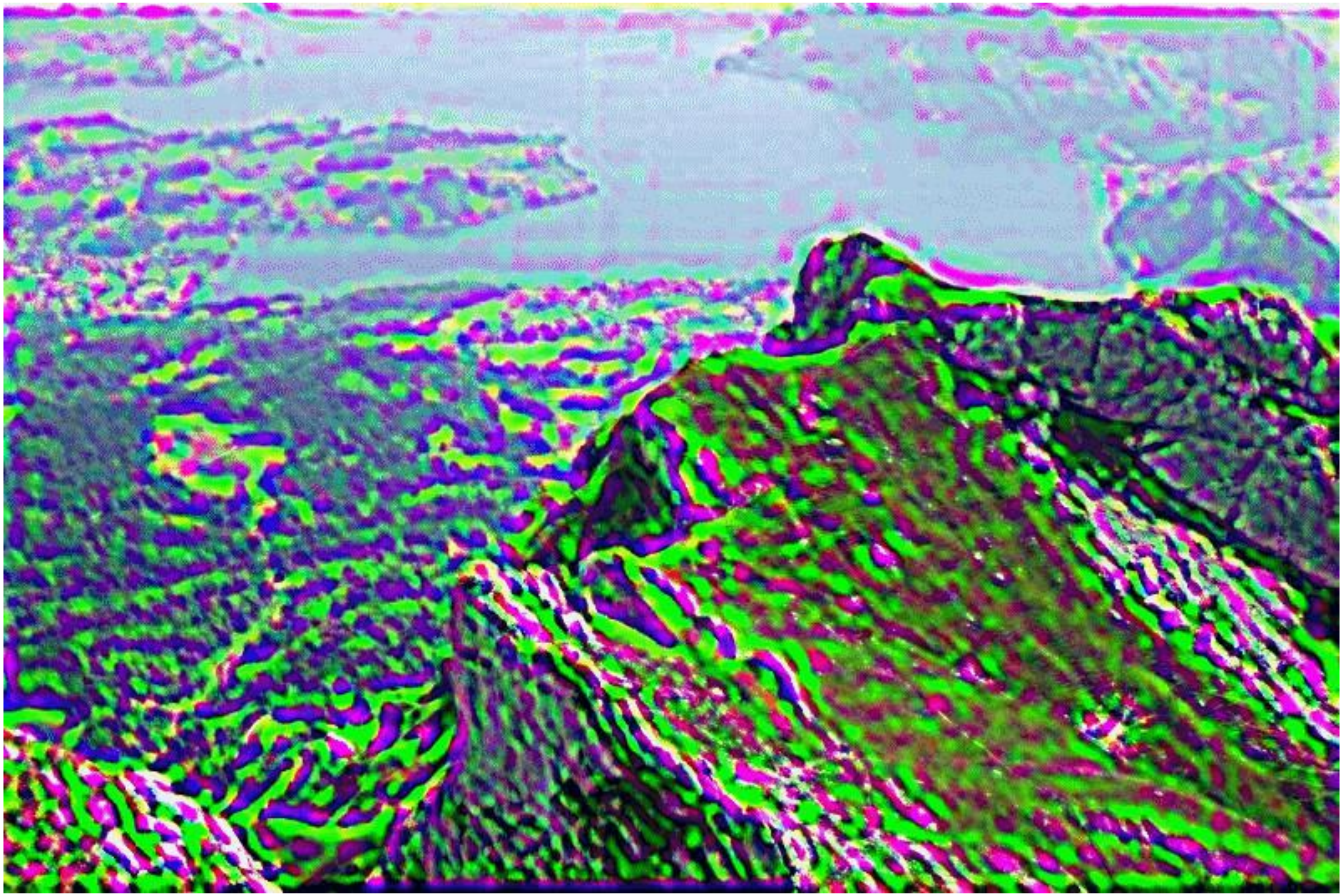
CNN layer
x
y
Feature type
CNN layer
x
y
Feature type
FC layer
Complex features

DEEPDREAM

DeepDream



Examples (all feature maps in a layer)



DeepDream example

http://www.pyimagesearch.com/2015/08/03/deep-dream-visualizing-every-layer-of-googlenet/

DeepDream example

http://www.pyimagesearch.com/2015/08/03/deep-dream-visualizing-every-layer-of-googlenet/

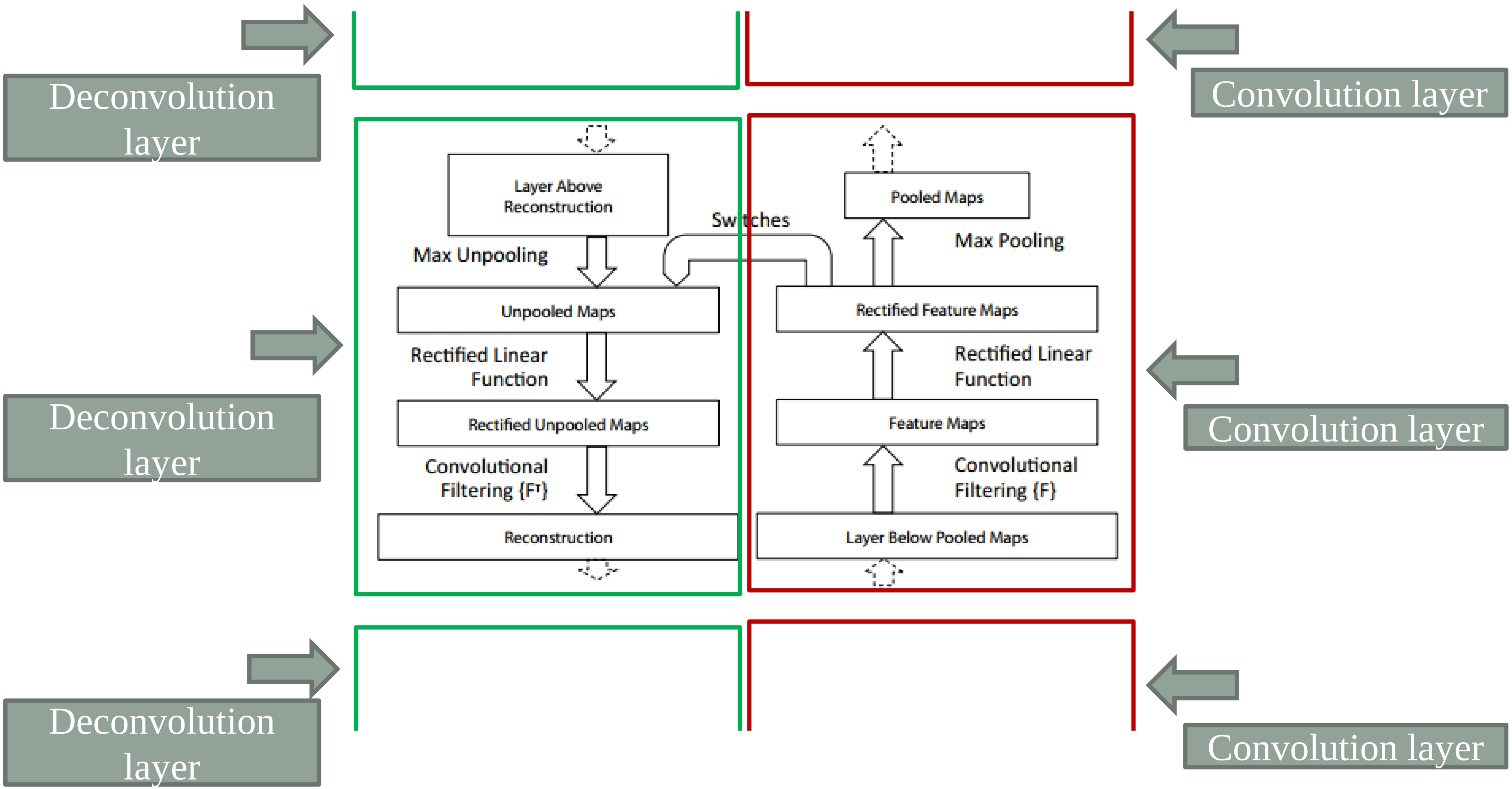
http://www.pyimagesearch.com/2015/08/03/deep-dream-visualizing-every-layer-of-googlenet/

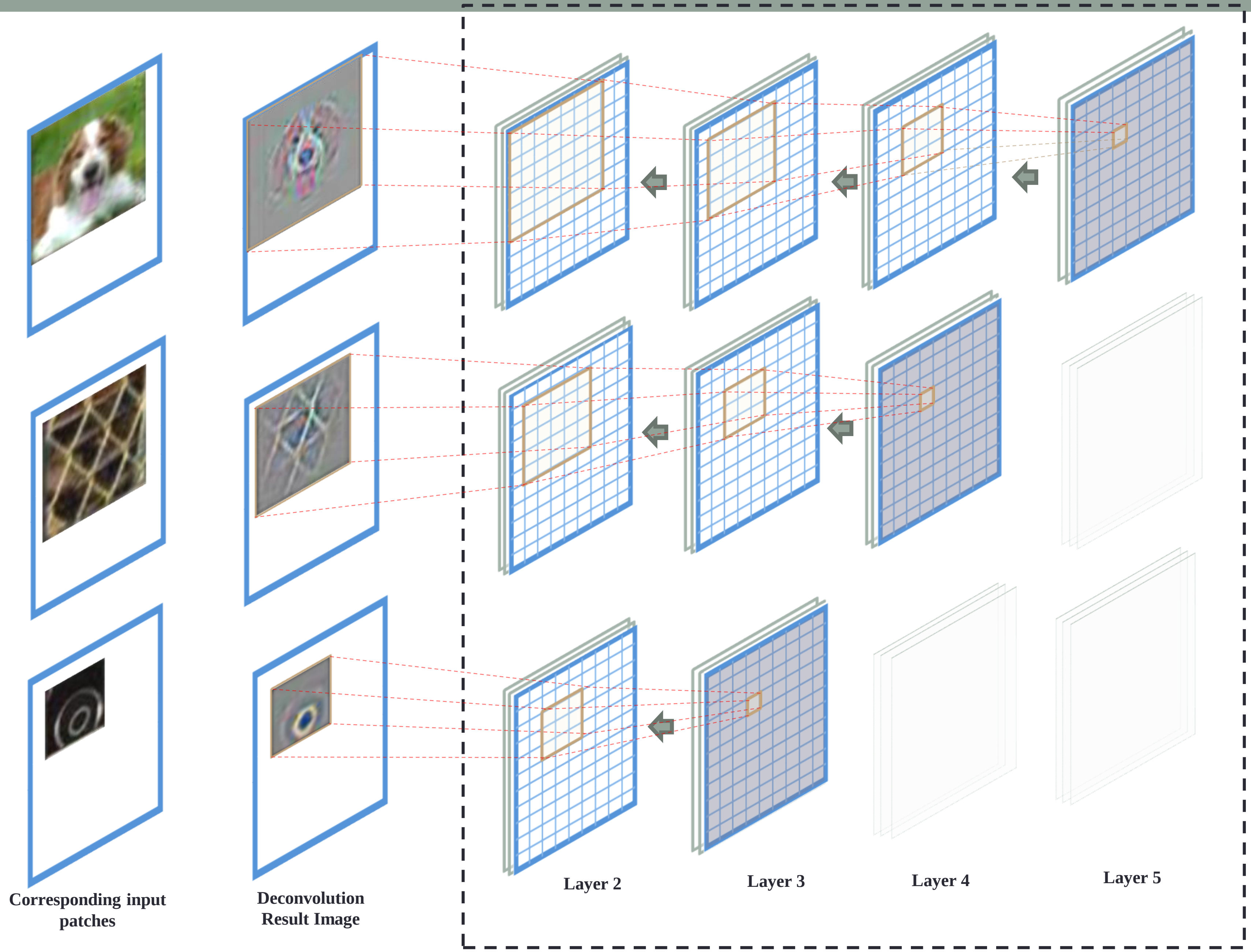
DISCUSSIONS

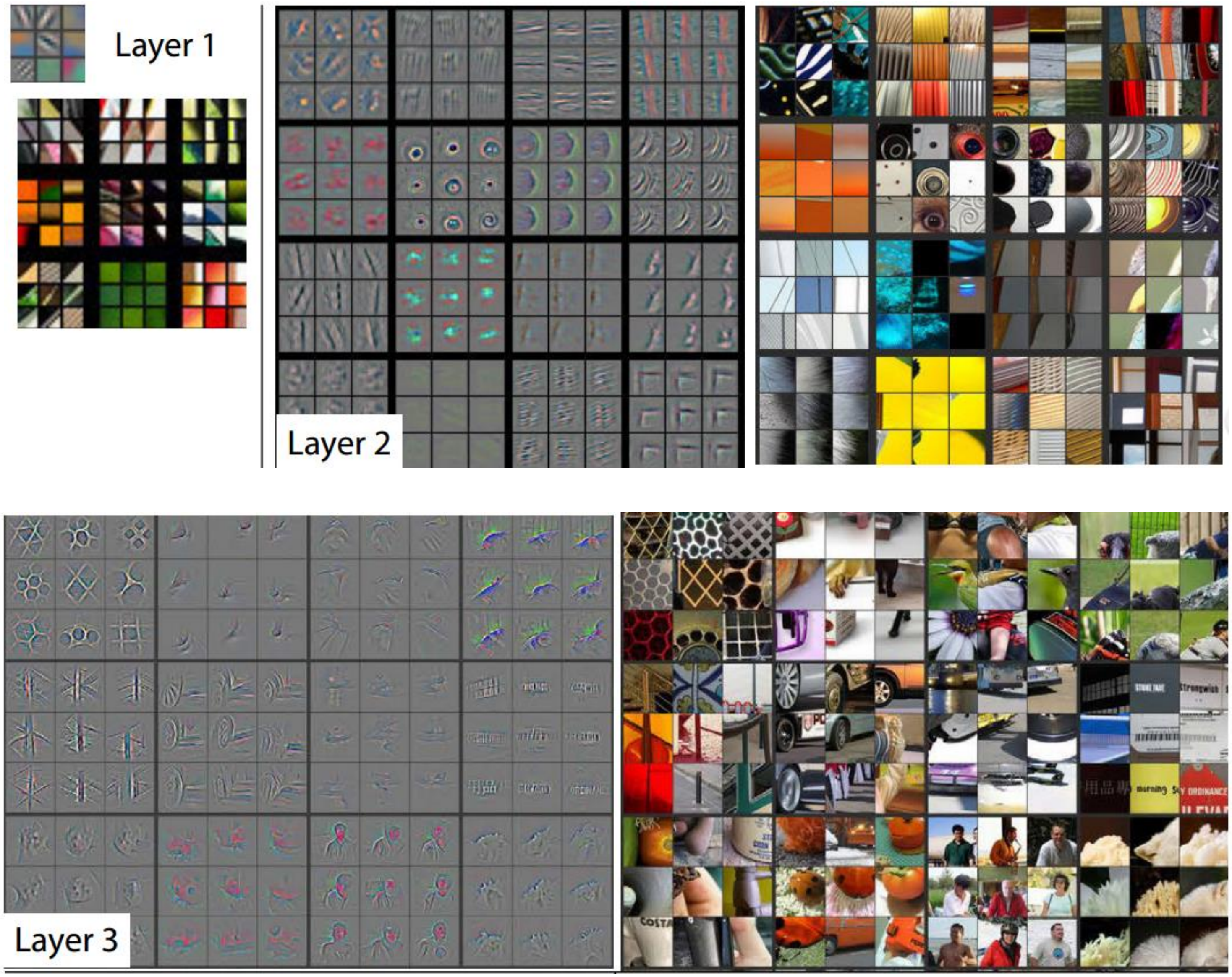
Some keywords in CNN

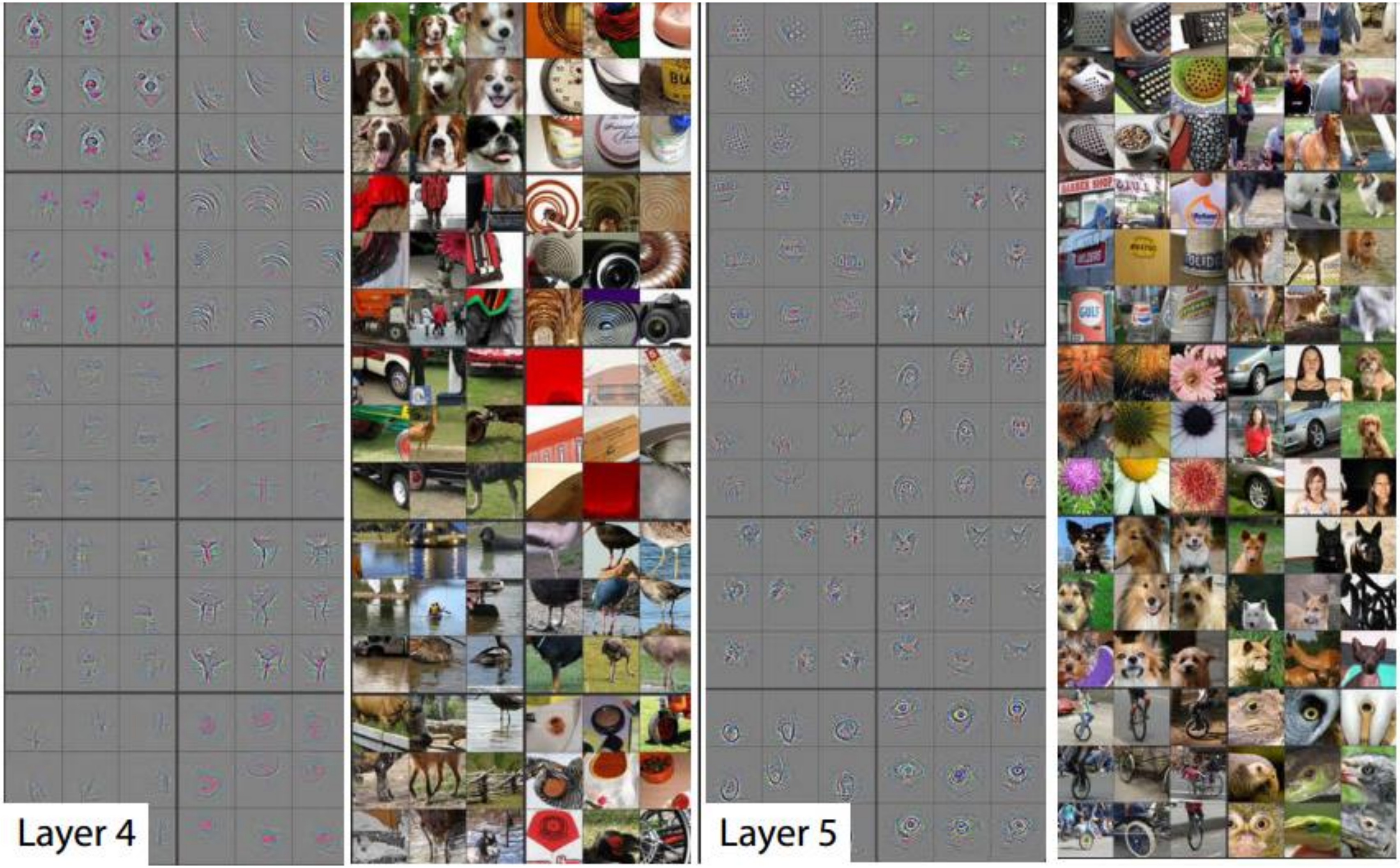
- Deconvolution
  - Matthew D. Zeiler and Rob Fergus, "Visualizing and Understanding Convolutional Networks," ECCV 2014
- 1x1 convolution: Network in network
- Inception module: GoogLeNet

Deconvolution

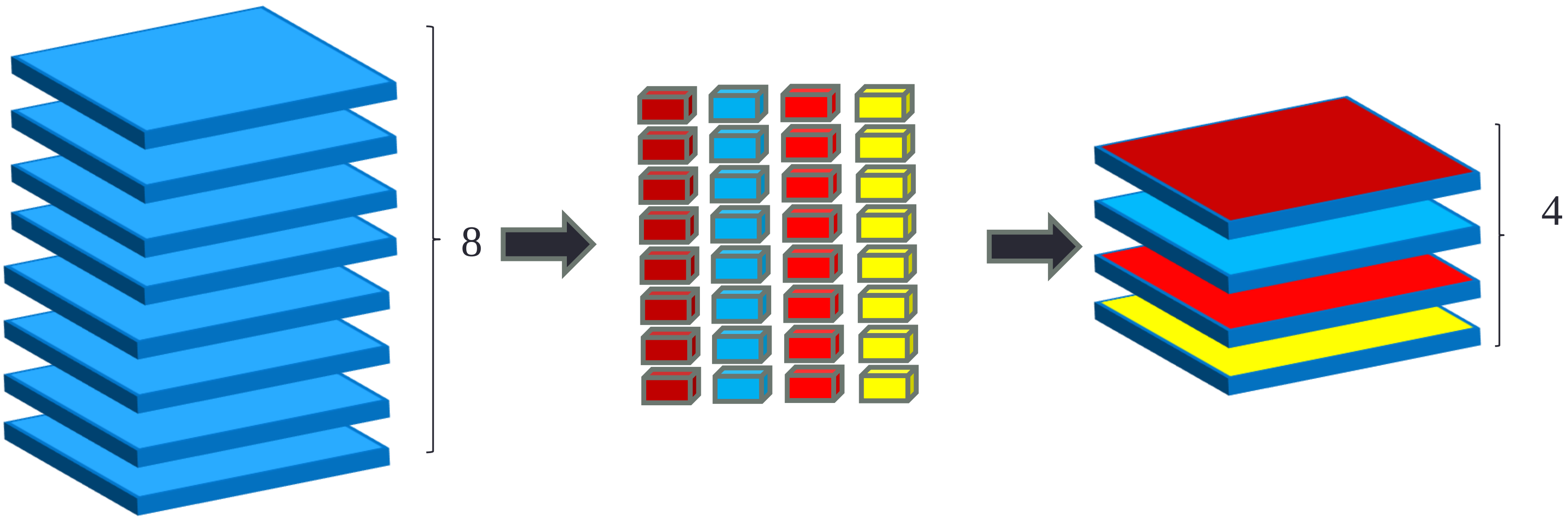



Corresponding input patches
Deconvolution Result Image
Layer 2
Layer 3
Layer 4
Layer 5


Layer 1
Layer 2
Layer 3

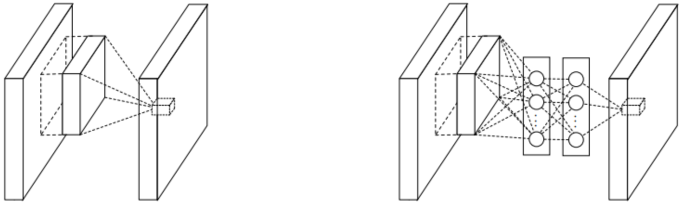

Layer 4
Layer 5

1 x 1 convolution?



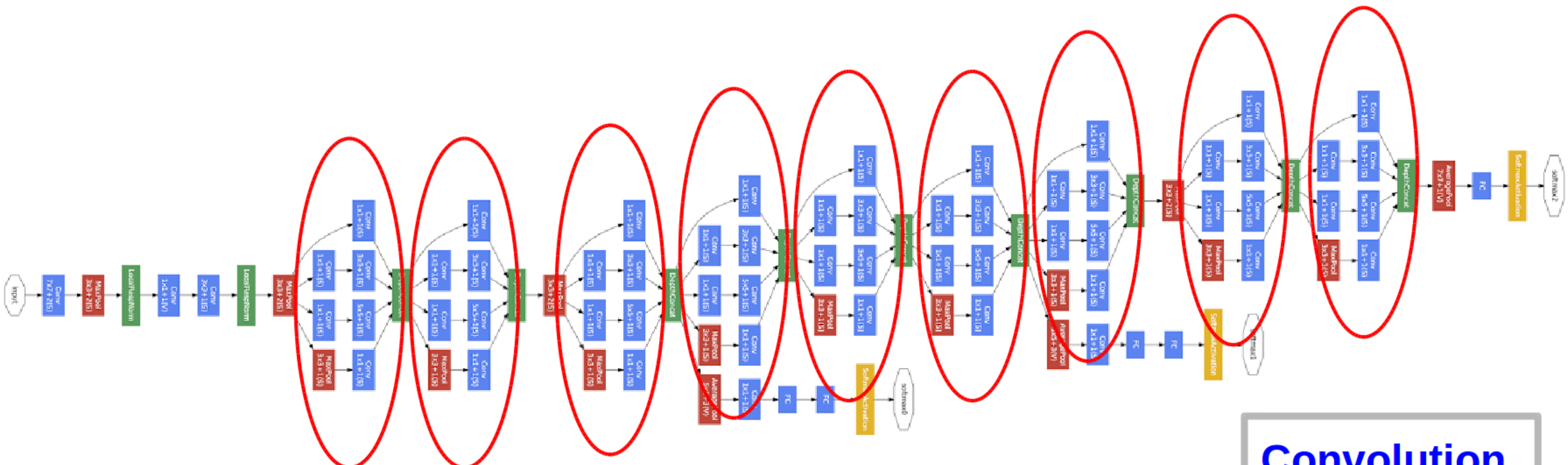
1 x 1 convolutions + nonlinear activation

- 1x1 convolution. first investigated by Network in Network.
- Dimension reduction using 1x1 convolution



(a) Linear convolution layer (b) Mlpconv layer

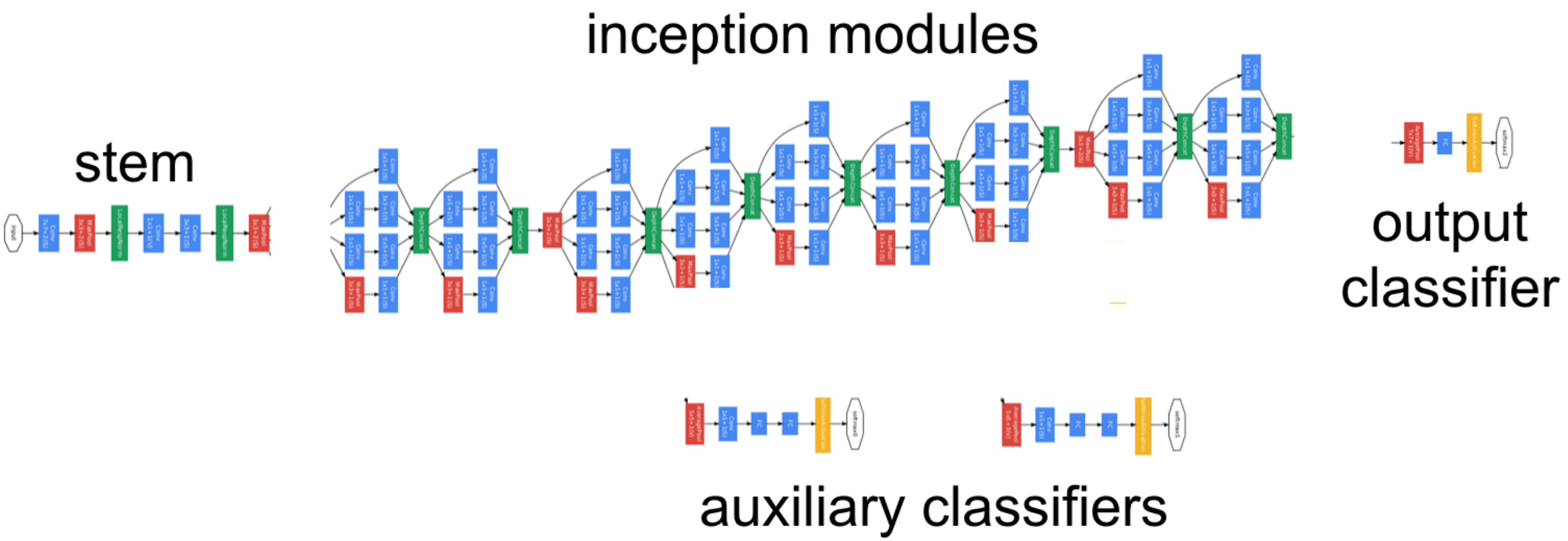
GoogLeNet



Convolution
Pooling
Softmax
Concat/Normalize

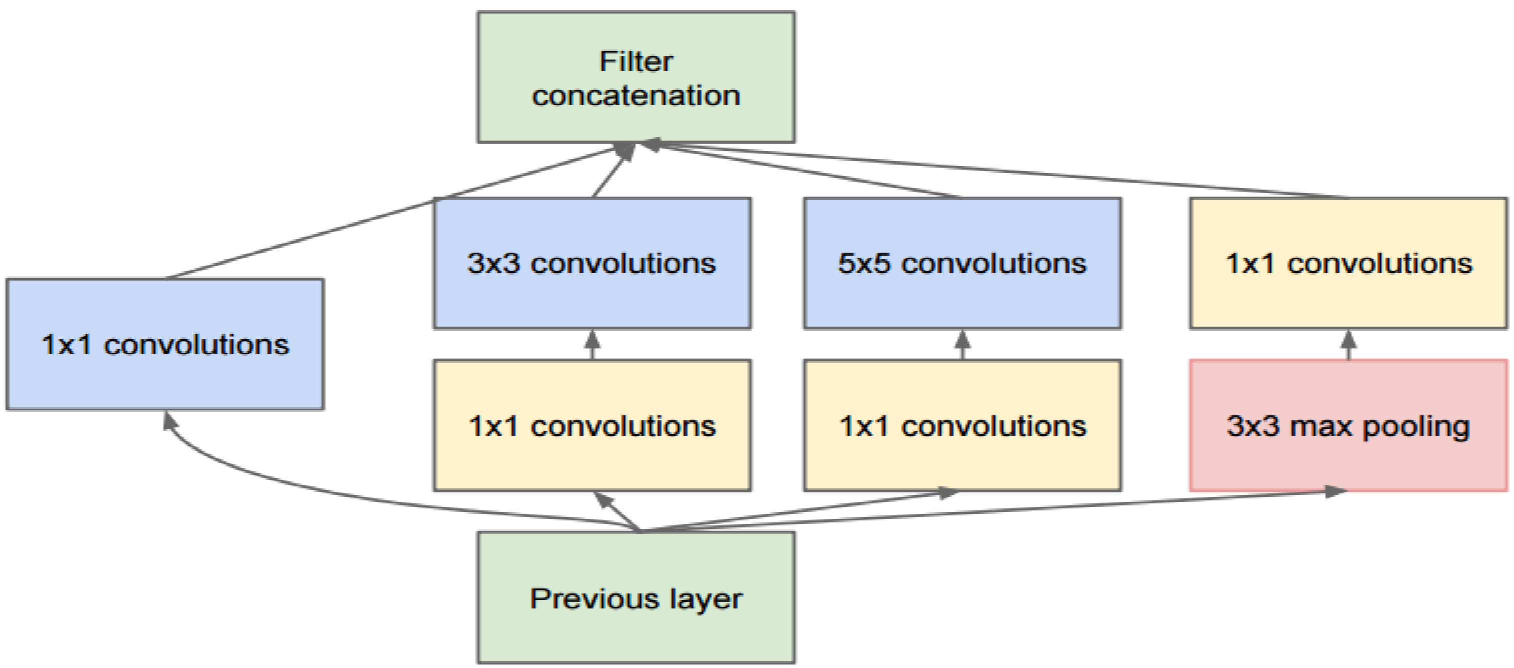
GoogLeNet

- Inception module
- One FC layer



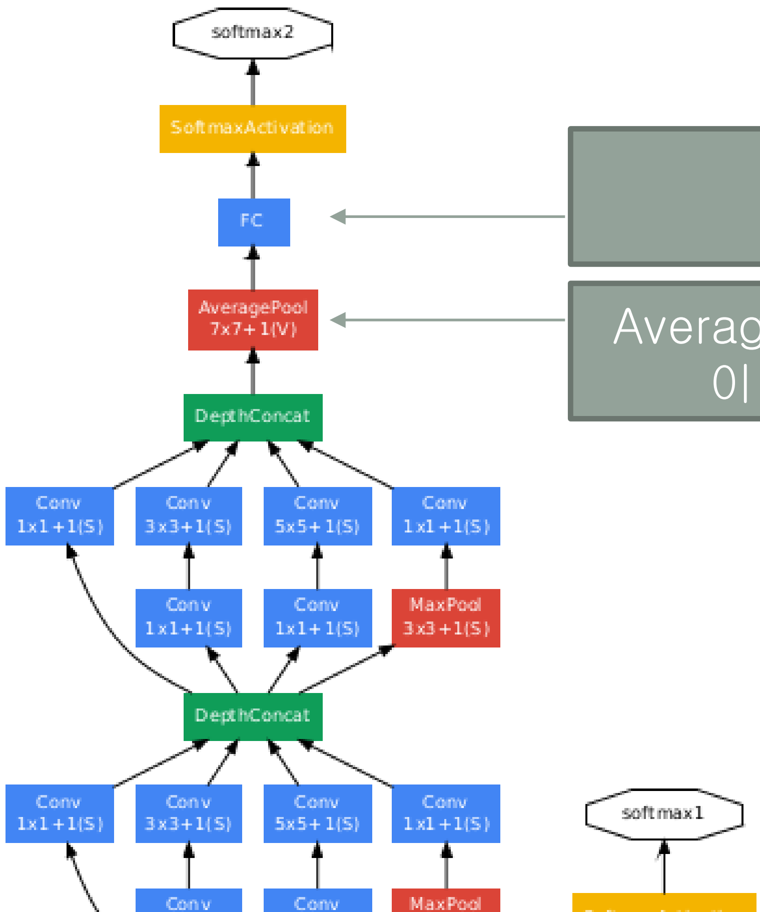
Inception module in GoogLeNet

- **In GoogLeNet,** 1x1 convolutions are used to compute reductions before the expensive 3x3 and 5x5 convolutions



FC layer in GoogLeNet

- 1 FC layer

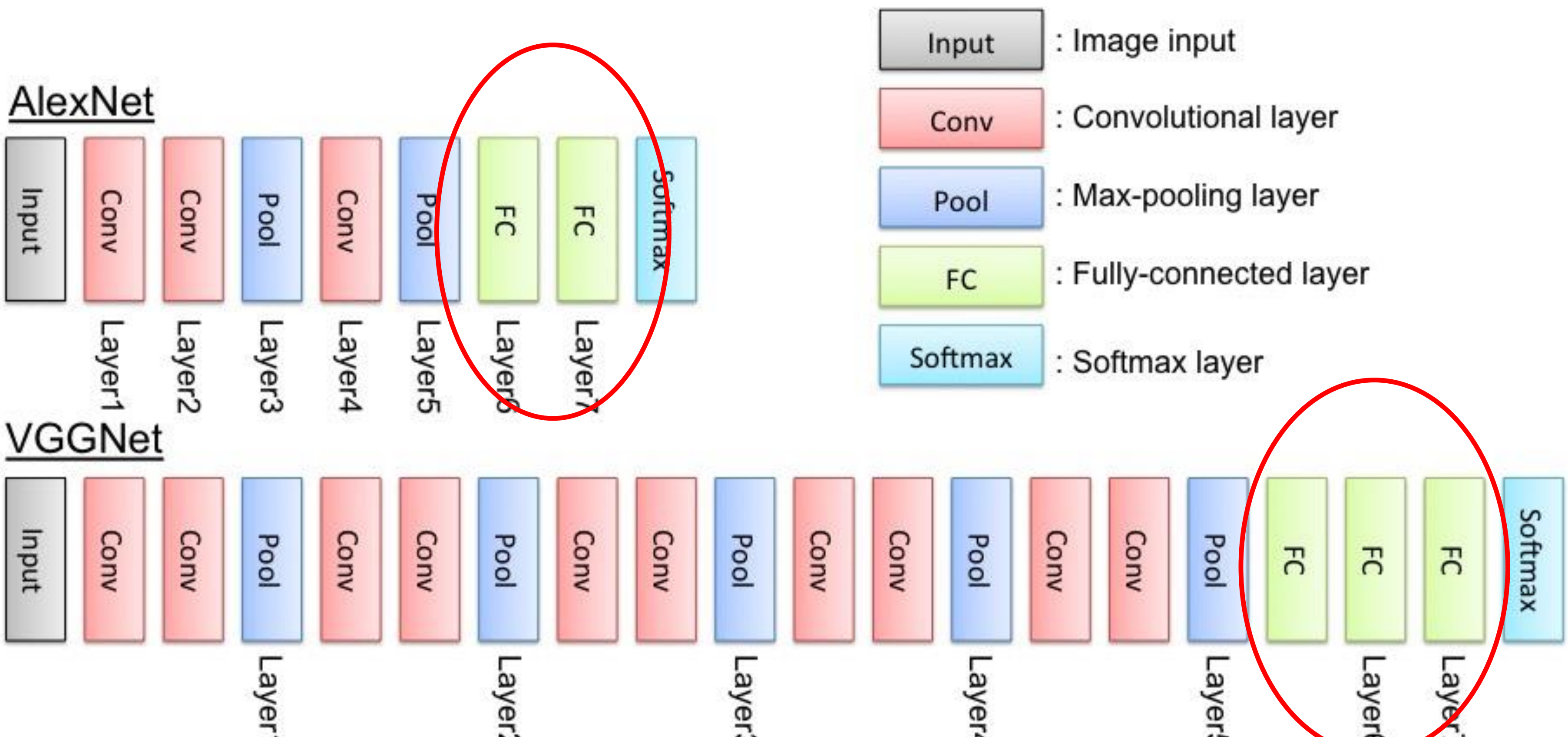


충분히 high level feature이고
Linearly seperable 함

Average Pooling: 충분히 high level feature
이고 더 이상 위치는 중요하지 않음

FC layers in AlexNet and VGGNet

- AlexNet: 2 FC layers
- VGGNet: 3 FC layer



AlexNet VS VGG-19 VS GoogLeNet

	Parameters	Operations (MACs)	*Top-1 accuracy (%)	*Top-5 accuracy %
AlexNet	60 M	832 M	56.9	80.1
VGG-19	144 M	19,632 M	68.5	88.5
GoogLeNet	6.8 M	1,502 M	68.7	89.0

*Evaluated with ImageNet2012

*https://github.com/BVLC/caffe/wiki/Models-accuracy-on-ImageNet-2012-val

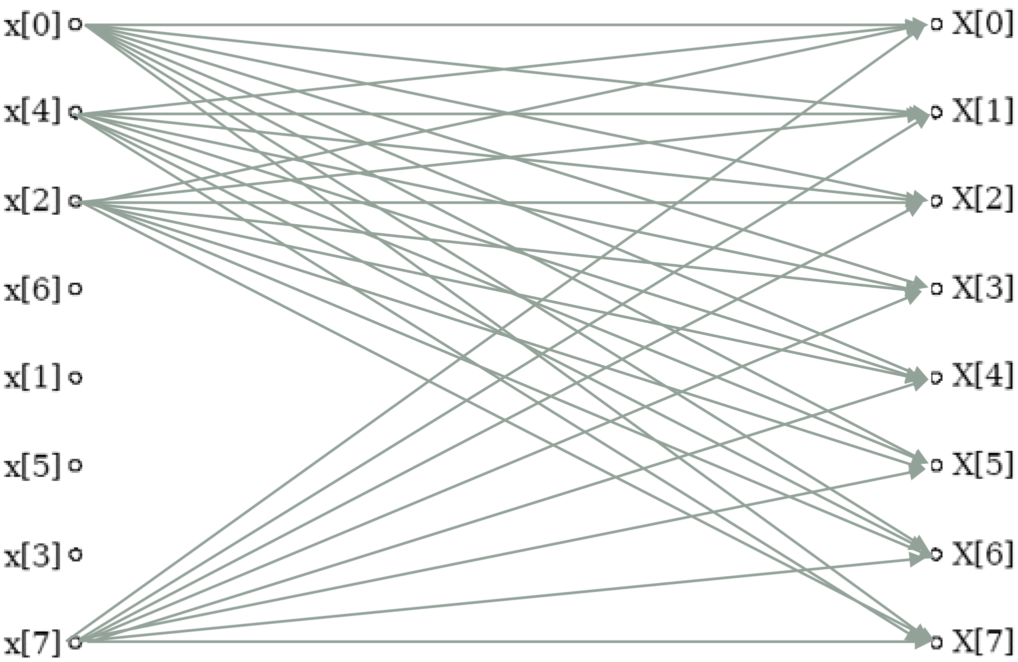
CONCLUSIONS

Conclusions

- The output of each convolution layer can be considered a 3D feature map consisting of (feature type, horizontal position, vertical position)
- Features in each feature map can be visualized with several techniques (deconvolution, activation maximization, deep dream…)
- How to find efficient/effective/sparse connections in each layer is an open problem, although the convolution seems to be the best choice in image processing

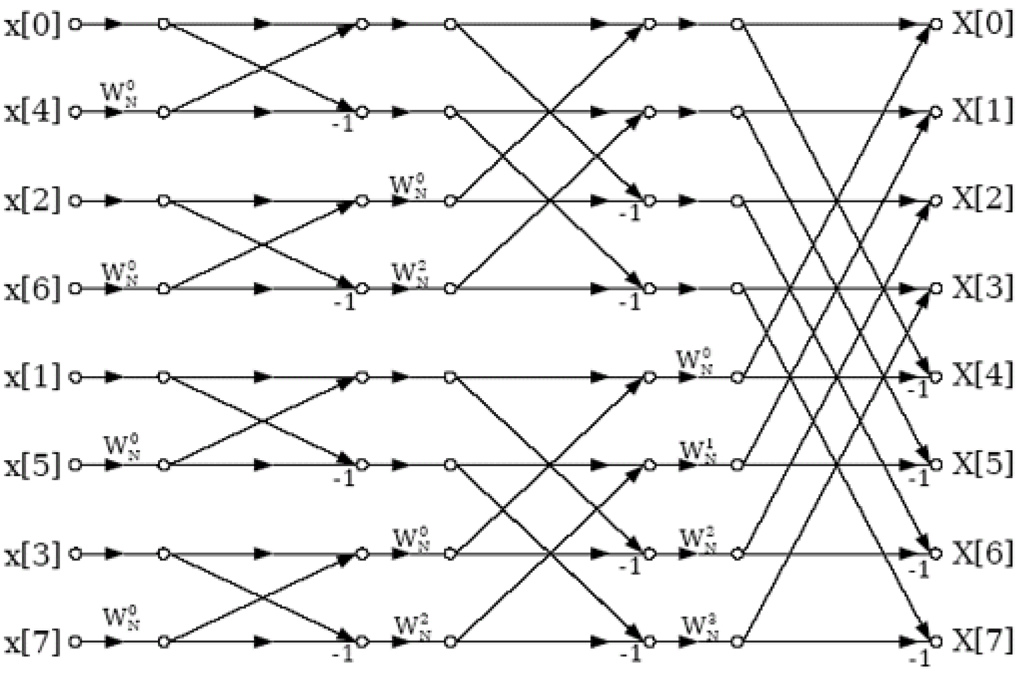
FFT Analogy

- Discrete Fourier Transform

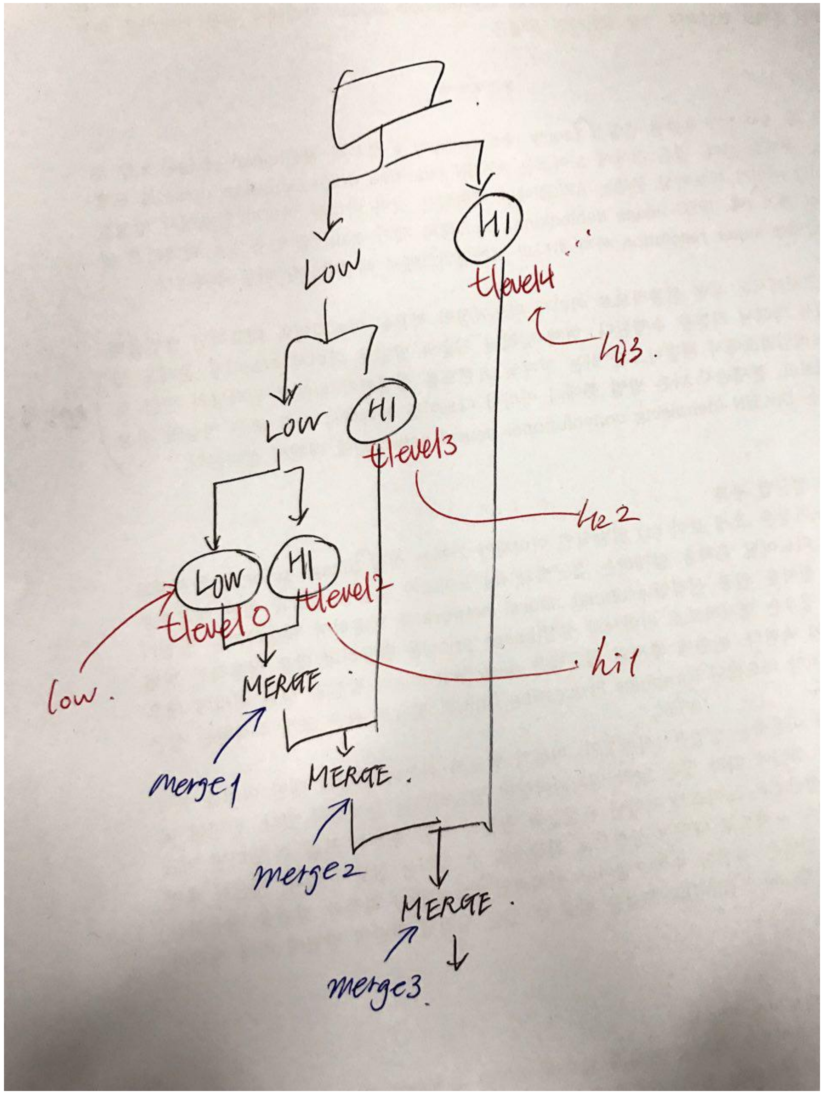


FFT Analogy

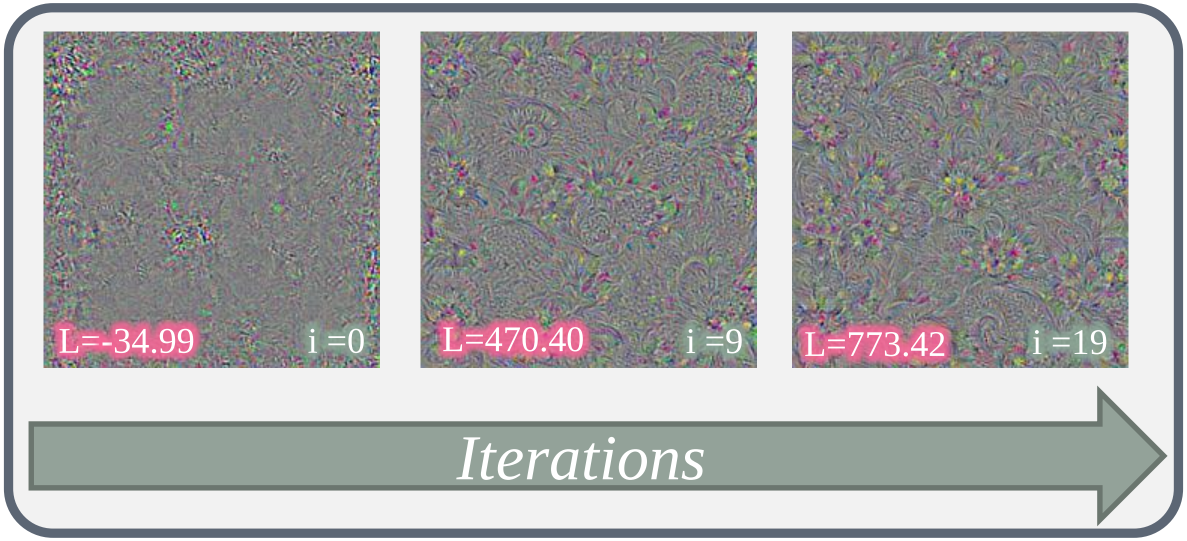
- Fast Fourier Transform



BACKUPS


HI
LOW
tlevel4
hi3.
LOW
HI
tlevel3
hi2
LOW
HI
tlevel0
tlevel2
hi1
low.
MERGE
merge1
MERGE
merge2
MERGE
merge3

Objective function values



L=102.42

http://storage.googleapis.com/deepdream/visualz/tensorflow_inception/index.html