Session 4 & Assignment

Due Dec 9, 2019 by 11:59pm **Points** 4,000 **Submitting** a website url **Available** until Dec 11, 2019 at 11:59pm

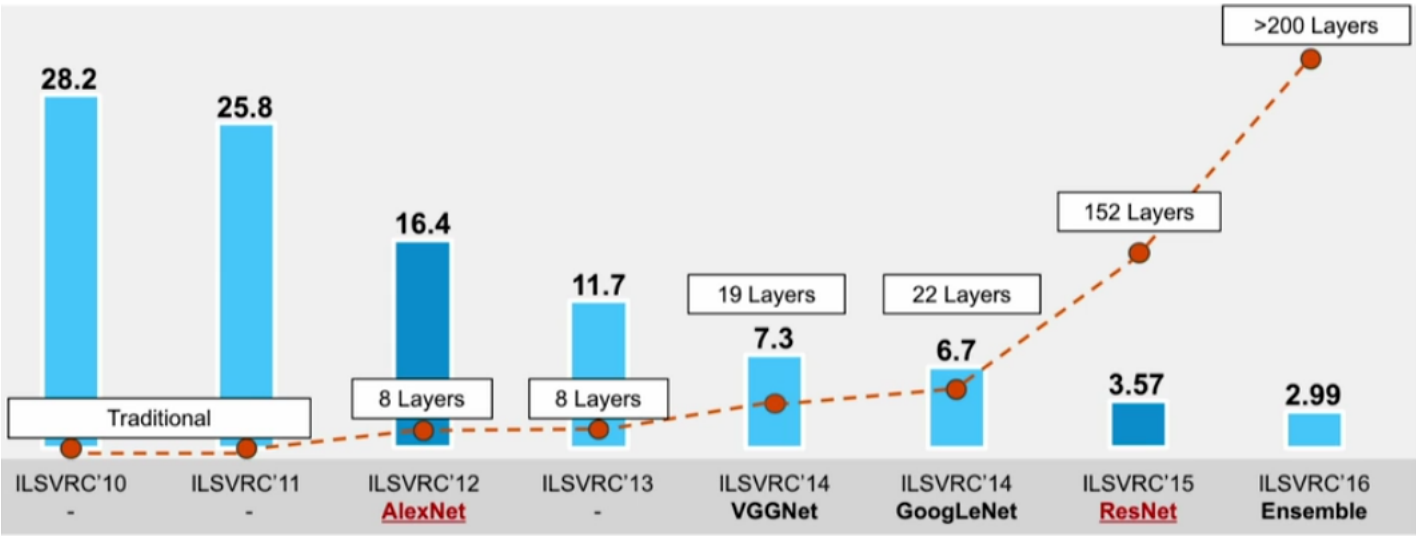
This assignment was locked Dec 11, 2019 at 11:59pm.

Modern DNN Architectures and Image Augmentation

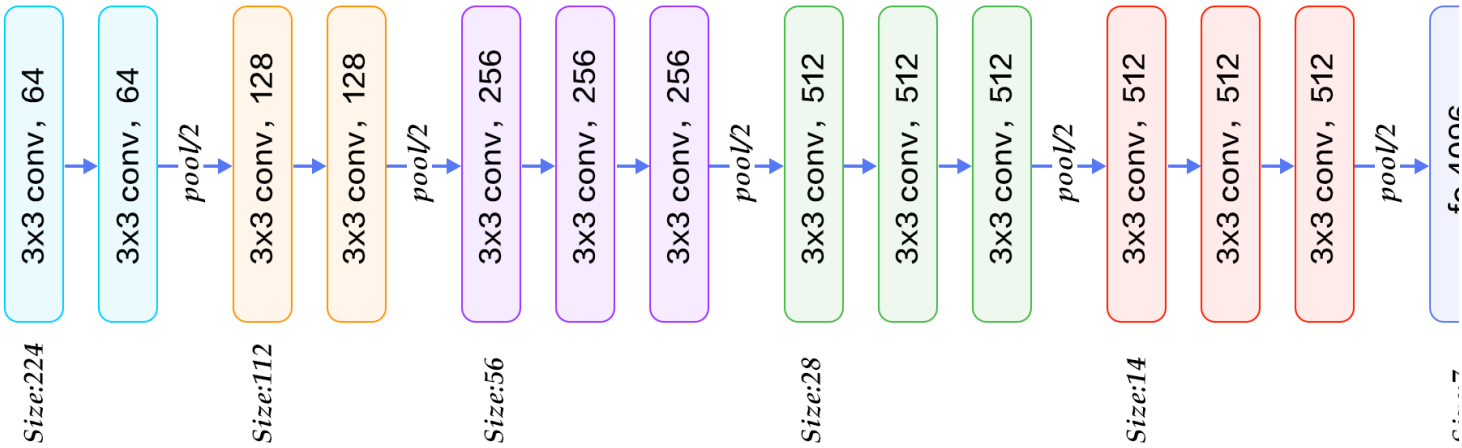
Session 4 Video

Saturday Video

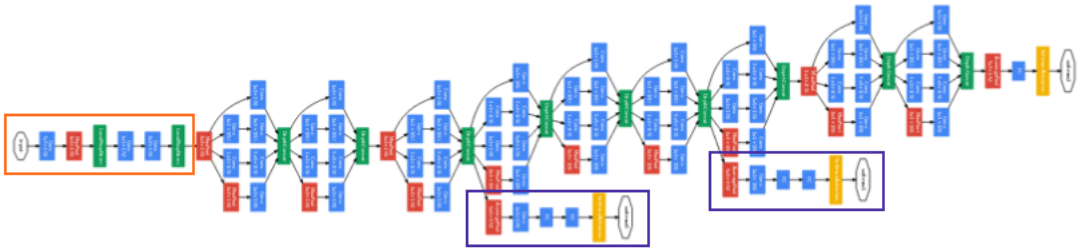
ILSVRC

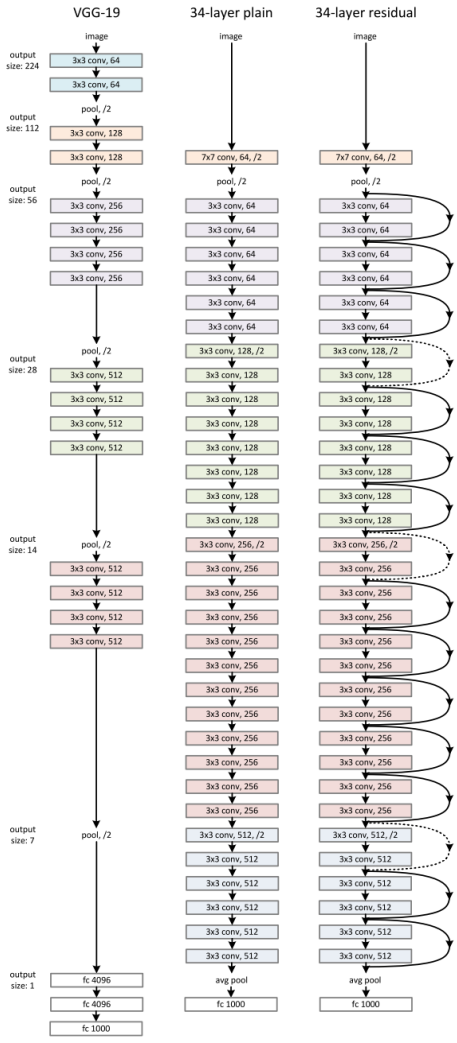


Sequential Architectures:



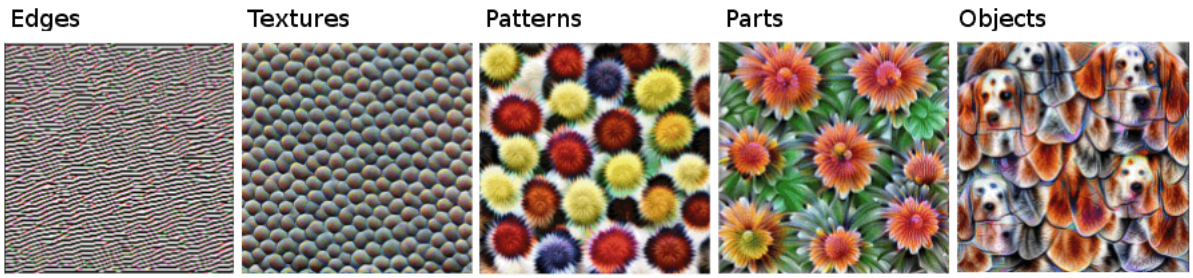
Parallel Architectures:




VGG-19
34-layer plain
34-layer residual
image
output size: 224
3x3 conv, 64
pool, /2
output size: 112
3x3 conv, 128
output size: 56
3x3 conv, 256
output size: 28
3x3 conv, 512
output size: 14
output size: 7
output size: 1
fc 4096
fc 1000
7x7 conv, 64, /2
3x3 conv, 128, /2
3x3 conv, 256, /2
3x3 conv, 512, /2
avg pool

But WHY?

Let's divide our image's receptive field of 400x400 and look at edges, gradients, textures, patterns, parts and objects.



If we remember, we had discussed that the **size of the object needs to be equal to the size of the image we are looking at.** Let's discuss why that was the case.

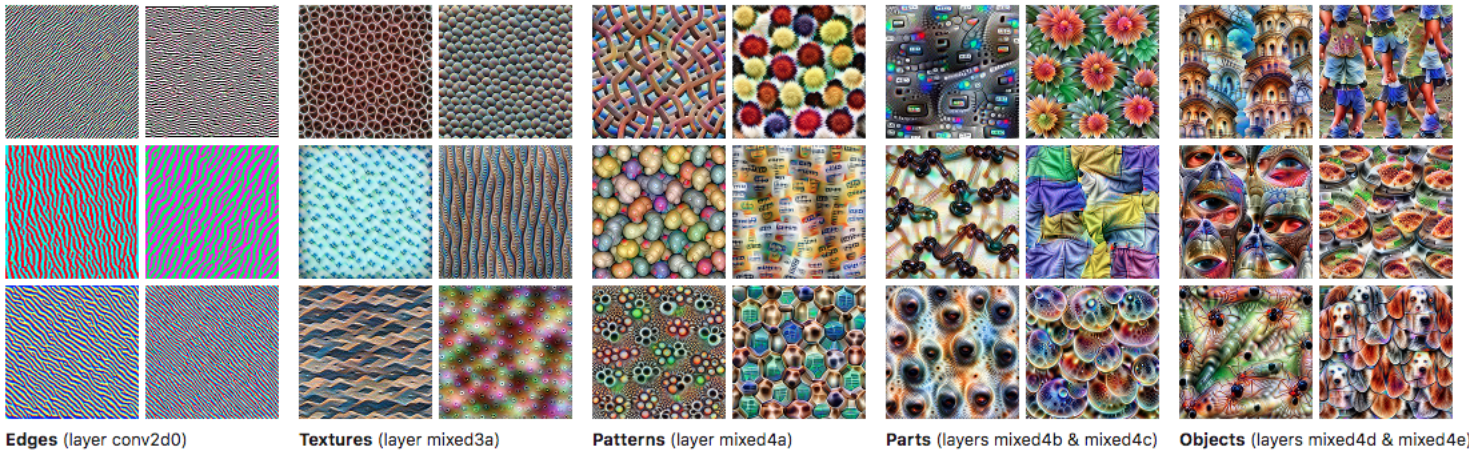
What if size of the object is not equal to the size of the object which is nearly always the case?

That's where the "parallelism magic happens"!

Parallelism helps take care of requirements of different receptive fields required for different objects!

What is receptive field of the network is much more than that of the data it is trained on?

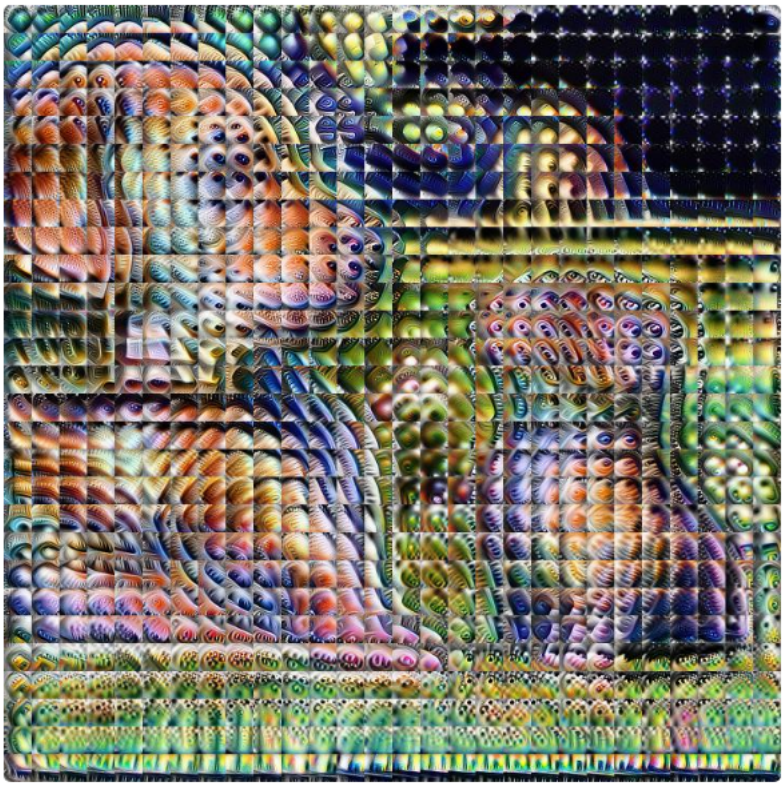
And as we have seen, as RF increases we make:



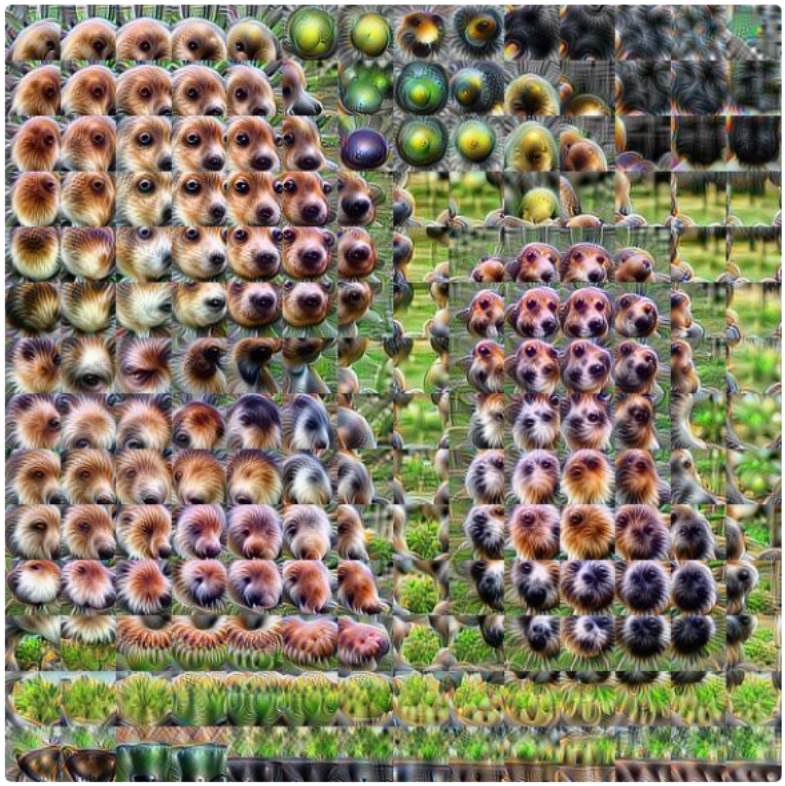
Let's dive in! (https://distill.pub/2018/building-blocks/)

Ths starting image:

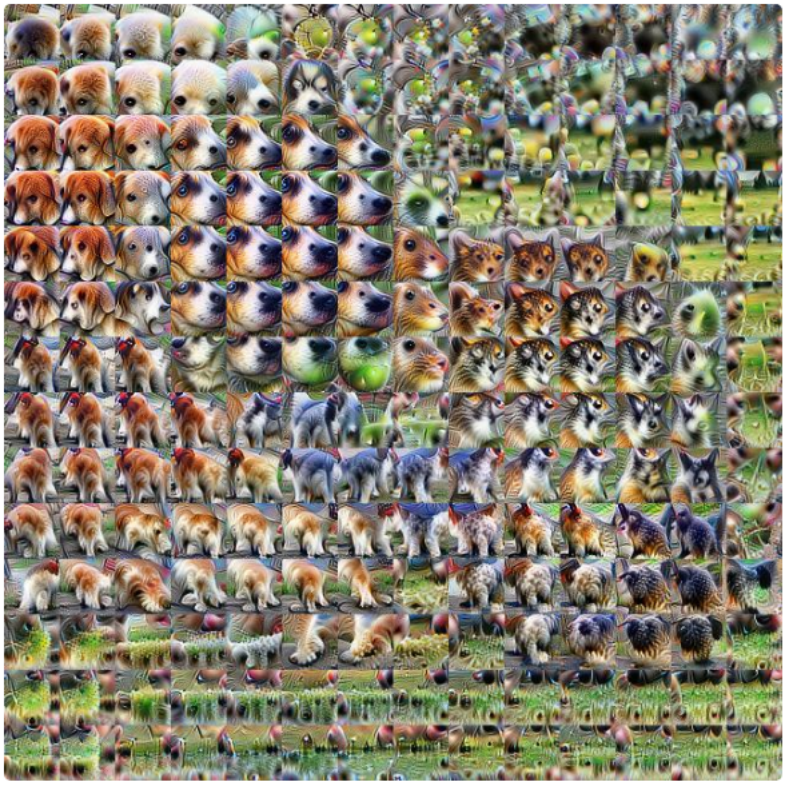
Image after few convolutions:



Few more convolution layers later:

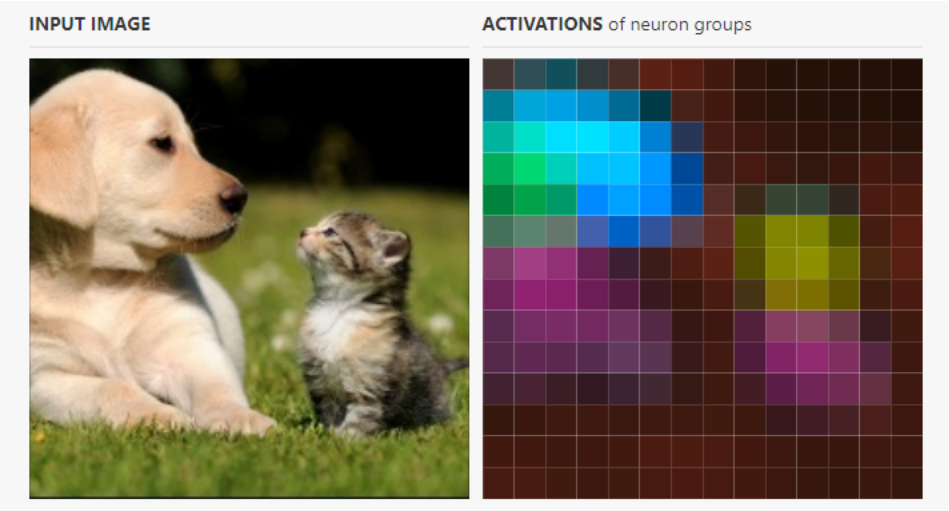


Few move convolution layers later:



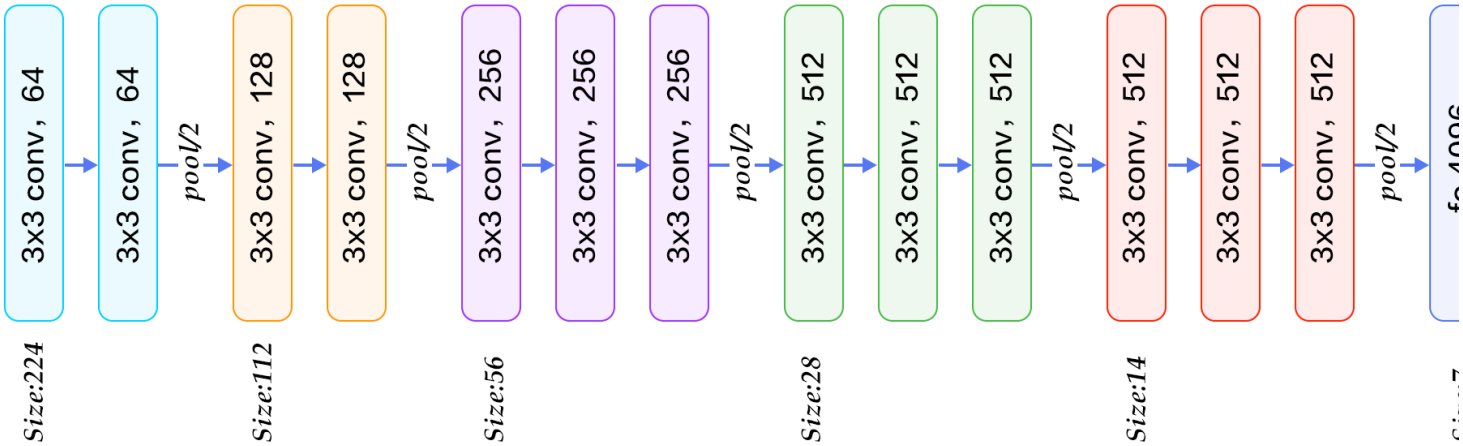
Few more convolution layers later (closer to softmax now)

And close to softmax:

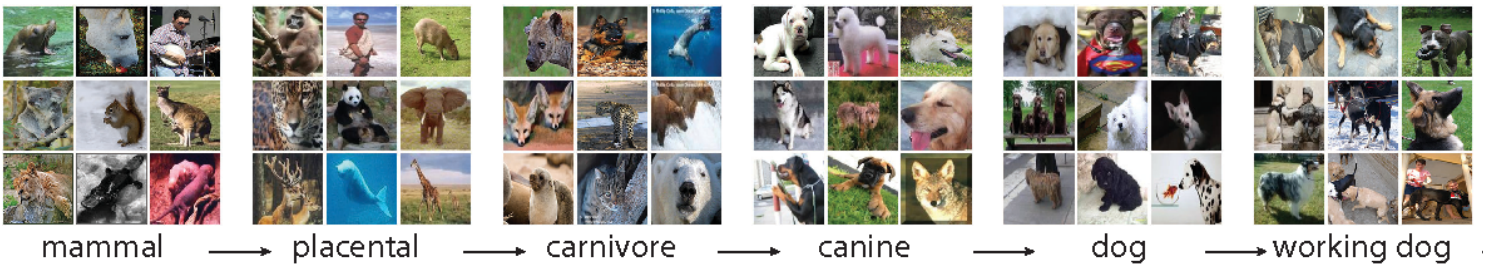


Different Networks

VGG



What problem do we face in actual DNN (older point of view)?



Scale.

INCEPTION V1

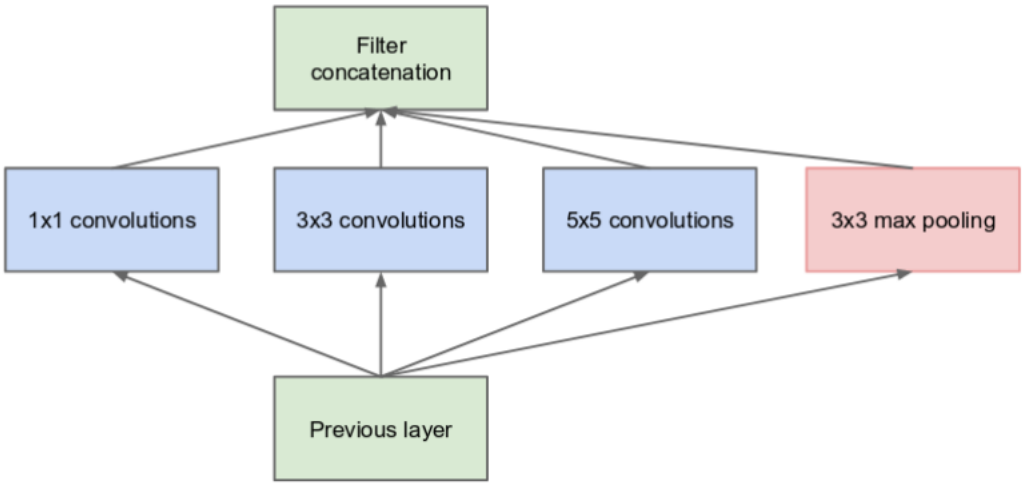
But initially we were thinking like this:

1. huge variation in the location of information, choosing the right kernel size is tough (wrong line of thought, why?)
2. very deep networks are prone to overfitting. It is also hard to pass gradient updates
3. naively (VGG) stacked large conv ops is computationally expensive.

Solution:

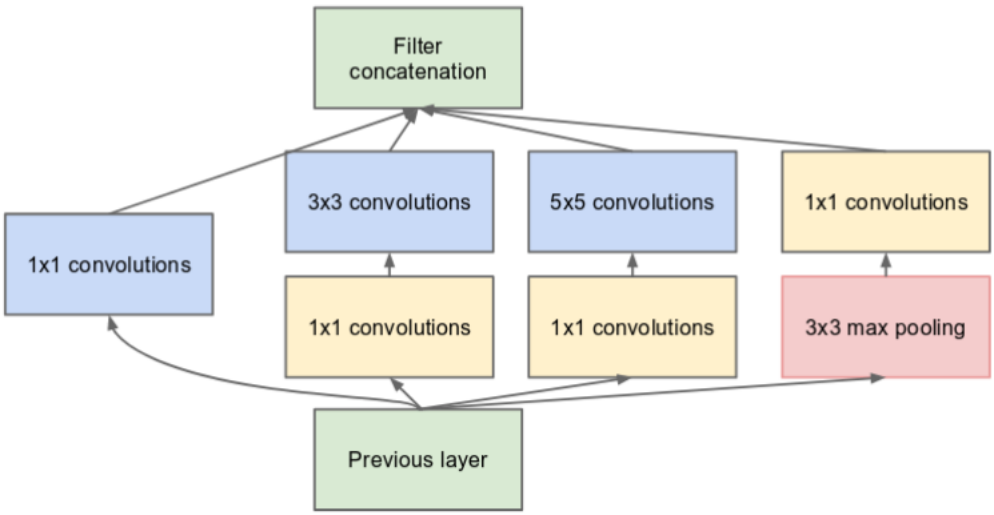
1. why not use filters with multiple sizes operating on the same input.
2. this way network would get wider, and not deeper, so we solve for 1 and 2.

Naive Inception Module



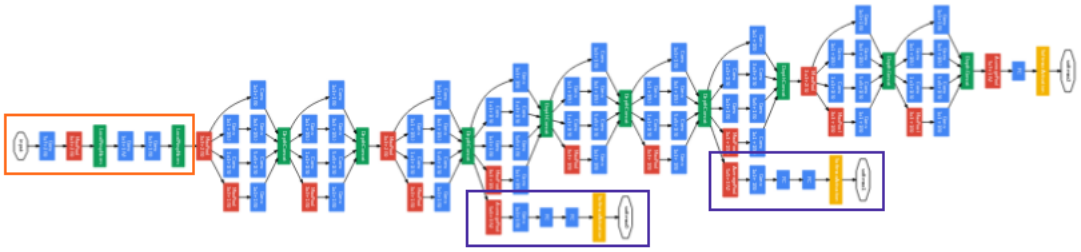
(a) Inception module, naïve version

But this is computationally expensive. To save computation we request 1x1 to save us:



(b) Inception module with dimension reductions

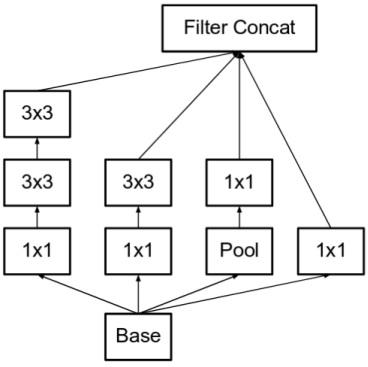
Stacking this we got GoogLeNet (Inception V1)

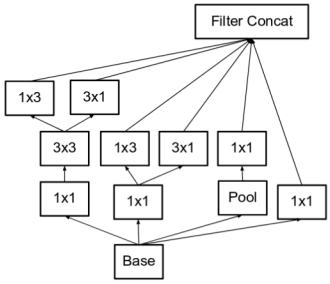


the winner against VGG.

Inception V2 & V3:

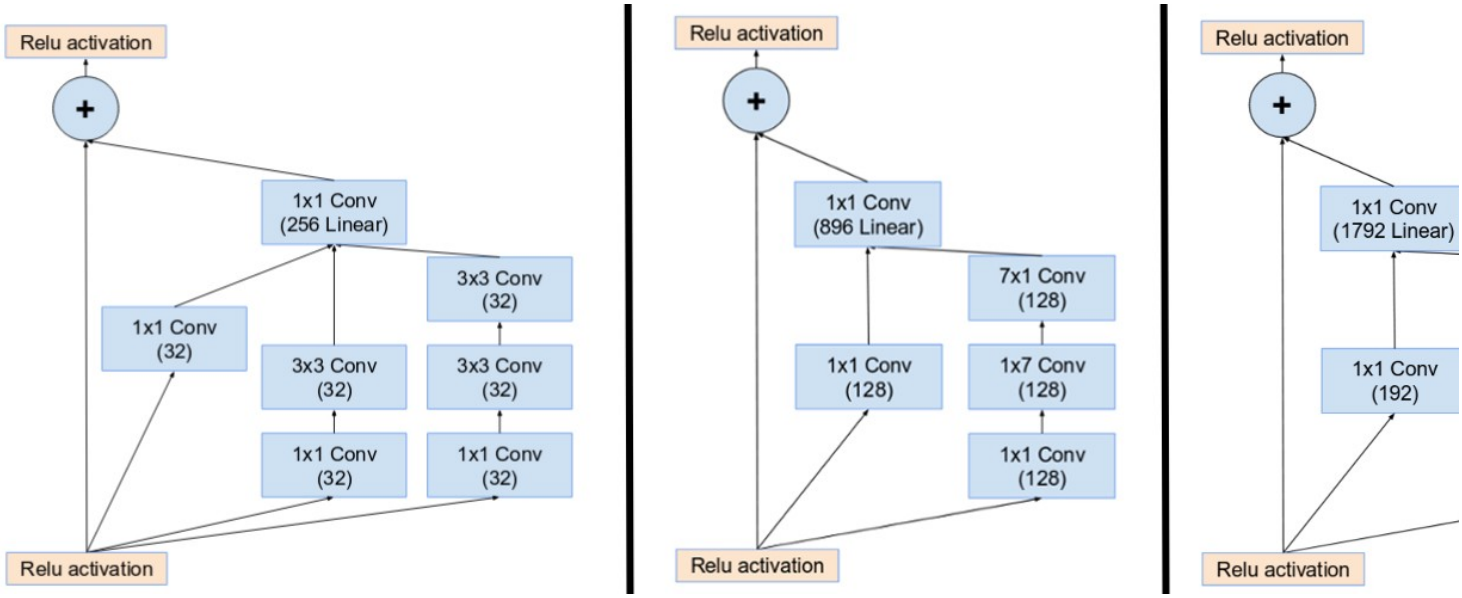
1. Computation Speed: increase by factorizing 5x5 convolution to two 3x3 convolution operation (a 5x5 is 2.78 times more expensive than 3x3 convolution)
2. More-over factorize nxn convolution to 1xn and nx1.





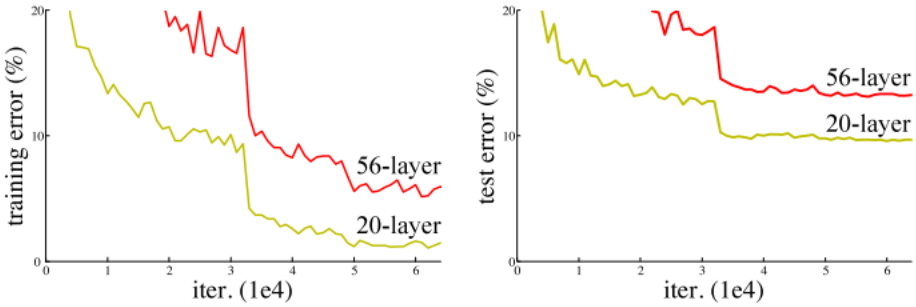
Inception V4 & Inception-ResNet:

1. Add 7x1, 1x7
2. More variations of the block of varying sizes
3. Add Residual Connections as well (Learn from Microsoft)

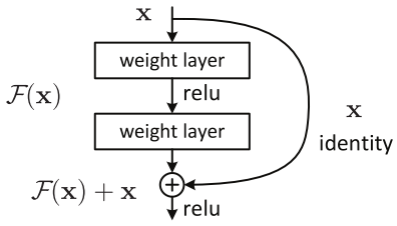


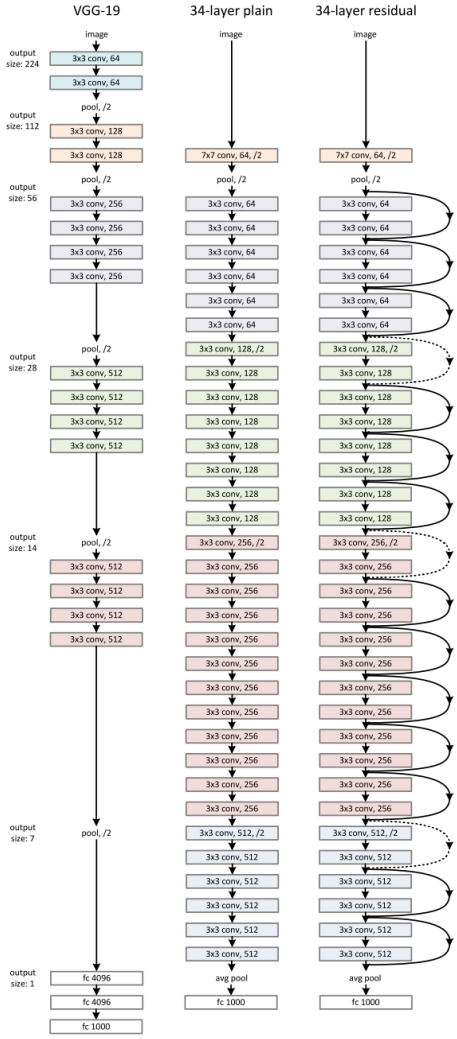
RESNET

1. Networks need to go deeper
2. How to solve over-fitting?
3. And currently

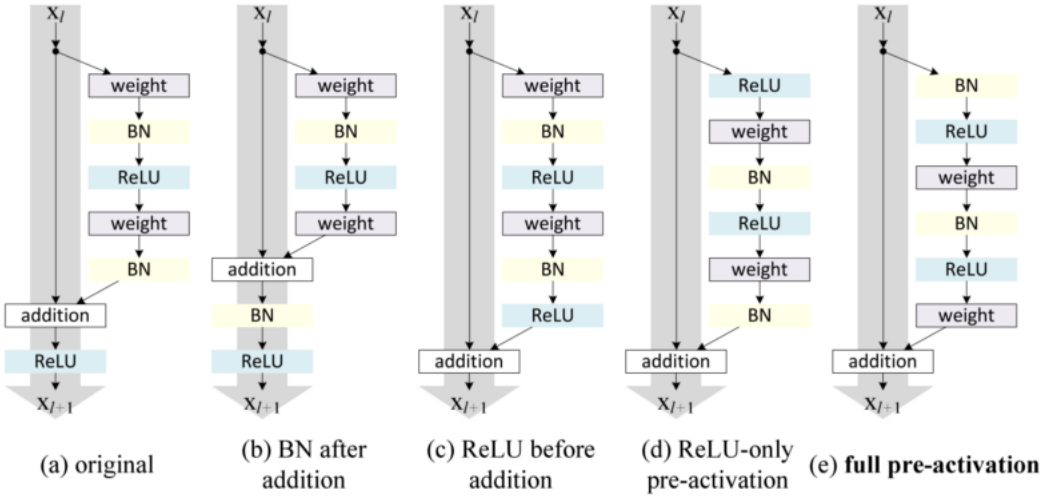


Solution:





And it's variants:



Exception, Inception and DenseNets are better than ResNet, then why do we love ResNet?

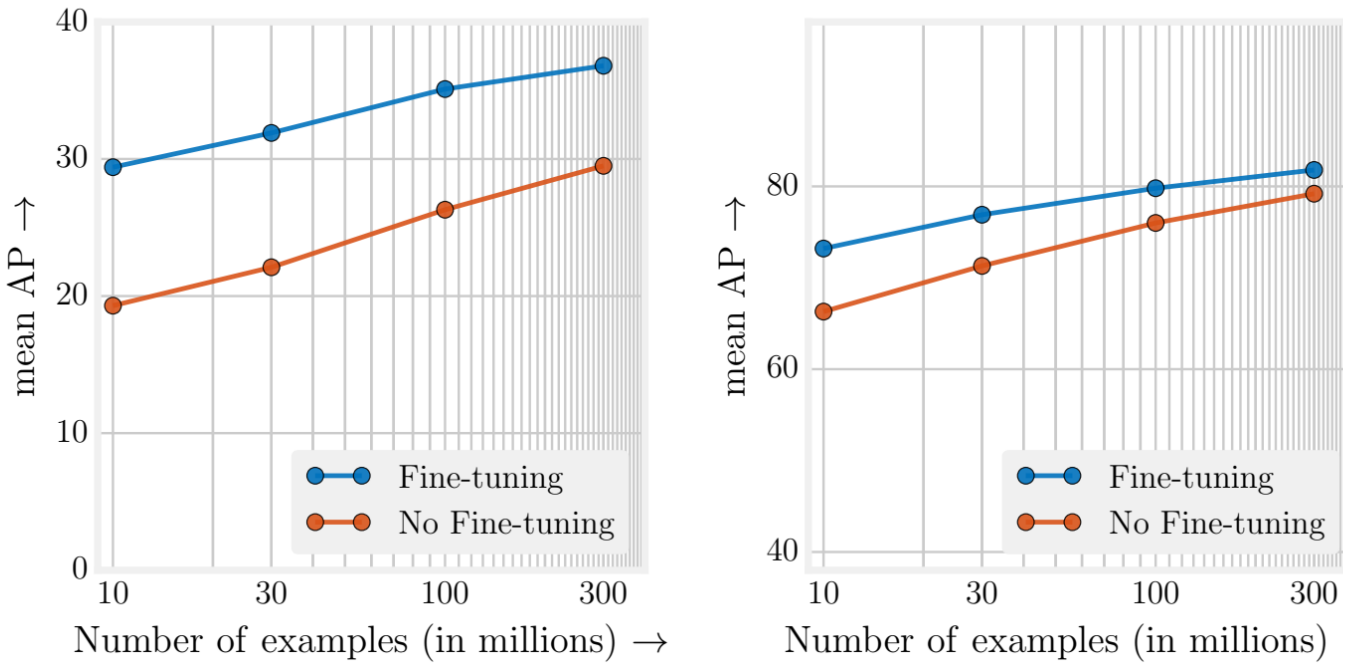
Data Augmentation

Data augmentation (DA) is an effective alternative to obtaining valid data, and can generate new labeled data based on existing data using "label-preserving transformations".

Designing appropriate DA policies requires a lot of expert experience and is time-consuming, and the evaluation of searching the optimal policies is costly. Moreover, the policies generated using DA policies are usually not reusable.

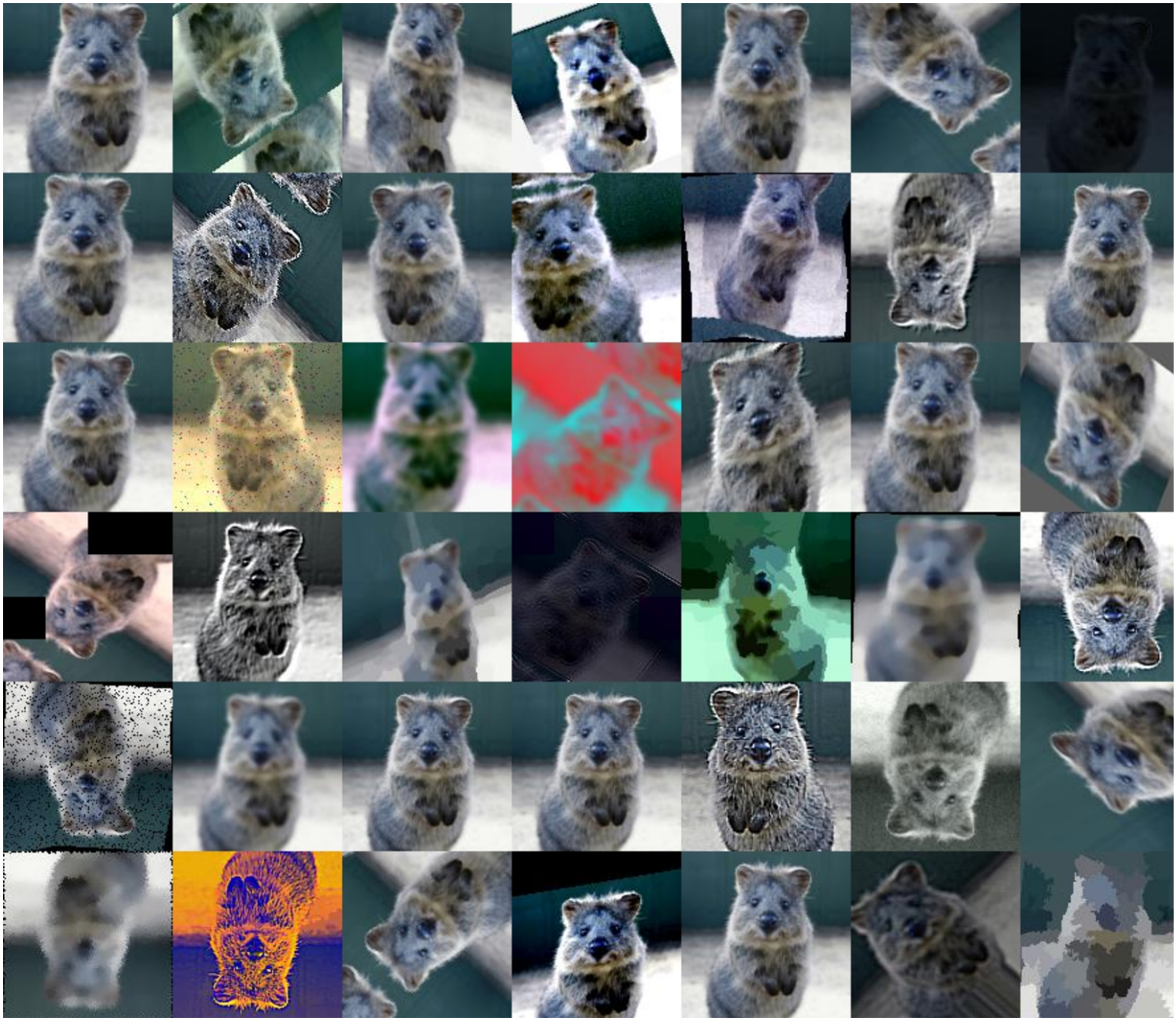
Generally, the more data, the better deep neural networks can do. Insufficient data can lead to model overfitting, which will reduce the generalization performance of the model on the test set.

Source *(http://openaccess.thecvf.com/content_ICCV_2017/papers/Sun_Revisiting_Unreasonable_Effectiveness_ICCV_2017_paper.pdf) : we find that the performance on vision tasks increases logarithmically based on the volume of training data size*



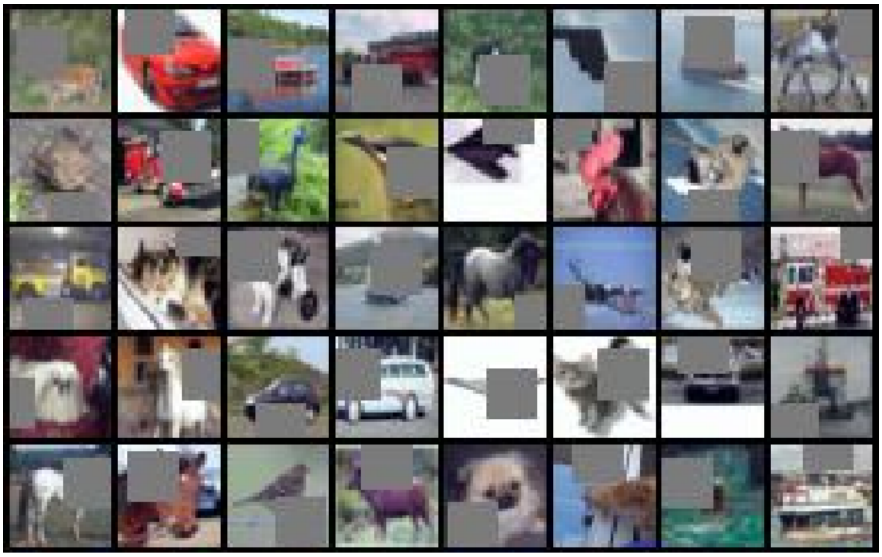
PMDAs

Scale; Translation; Rotation; Blurring; Image Mirroring; Color Shifting / Whitening



Cutout (https://arxiv.org/pdf/1708.04552.pdf) **29 Nov 2017:**

CutOut: Image speaks better on what it is:



Method	C10	C10+	C100	C100+	SVHN
ResNet18 [5]	10.63 ± 0.26	4.72 ± 0.21	36.68 ± 0.57	22.46 ± 0.31	-
ResNet18 + cutout	9.31 ± 0.18	3.99 ± 0.13	34.98 ± 0.29	21.96 ± 0.24	-
WideResNet [22]	6.97 ± 0.22	3.87 ± 0.08	26.06 ± 0.22	18.8 ± 0.08	1.60 ± 0.05
WideResNet + cutout	$\mathbf{5.54 \pm 0.08}$	3.08 ± 0.16	$\mathbf{23.94 \pm 0.15}$	18.41 ± 0.27	$\mathbf{1.30 \pm 0.03}$
Shake-shake regularization [4]	-	2.86	-	15.85	-
Shake-shake regularization + cutout	-	$\mathbf{2.56 \pm 0.07}$	-	$\mathbf{15.20 \pm 0.21}$	-

Table 1: Test error rates (%) on CIFAR (C10, C100) and SVHN datasets. "+" indicates standard data augmentation (mirror + crop). Results averaged over five runs, with the exception of shake-shake regularization which only had three runs each. Baseline shake-shake regularization results taken from [4].

Example Code (https://github.com/yu4u/cutout-random-erasing)

Let's cover a critical concept for our next step.

Global Average Pooling (https://arxiv.org/pdf/1312.4400.pdf)

the fully connected layers are prone to overfitting, thus hampering the generalization ability of the overall network. Dropout is proposed by Hinton et al. [5] as a regularizer which randomly sets half of the activations to the fully connected layers to zero during training. It has improved generalization ability and largely prevents overfitting.

Instead of adding fully connected layers on top of the feature maps, we take the average of each feature map, and the resulting vector is fed directly into the softmax layer.

1. it is more native to the convolution structure by enforcing correspondences between feature maps and categories, thus, the feature maps can be easily interpreted as categories-confidence maps.
2. No parameter to optimize in the global average pooling, thus overfitting is avoided at this layer
3. GAP sums out the spatial information, this is more robust to the spatial translation of the input.
4. GAP has ... parameters.

Class activation Maximization/Mapping

We can learn deep features for discriminative localization

CAM is applicable only to GAP Layers.

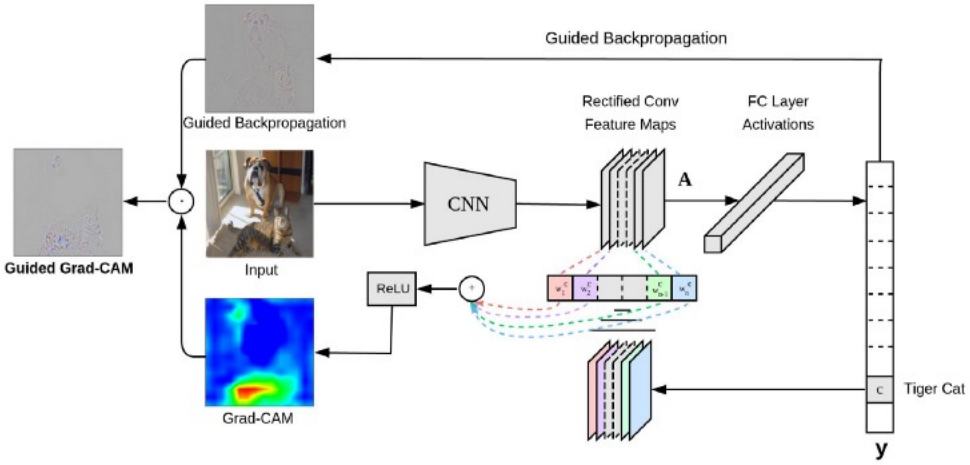
GradCAM

1. applicable to **any** CNN based network without requiring any changes (like GAP requirement)
2. can be applied to classification, captioning or Visual QA problems

Let's look at some **demos** **(http://gradcam.cloudcv.org/)**

Gradient-weighted Class Activation Mapping (GradCAM) uses the gradients of any target concept (say logits for 'dog' or even a caption), flowing into the final convolutional layer to produce a coarse localization map highlighting the important regions in the image for predicting the concept.

We take the final convolutional feature map, and then we **weight** every channel in that feature with the gradient of the class with respect to the channel. It tells us how intensely the input image activates different channels by how important each channel is with regard to the class. It does not require any re-training or change in the existing architecture.



Steps:

1. Load a pre-trained model
2. Load an image which can be processed by this model (224x224 for VGG16 why?)
3. Infer the image and get the topmost class index
4. Take the output of the final convolutional layer
5. Compute the gradient of the class output value w.r.t to L feature maps
6. Pool the gradients over all the axes leaving out the channel dimension
7. Weigh the output feature map with the computed gradients (+ve)
8. Average the weighted feature maps along channels
9. Normalize the heat map to make the values between 0 and 1

GradCAM Manual Implementation **(https://colab.research.google.com/drive/10GugXUNI7ztK2joRZUnYyqRrQbYnOQE0)**

Assignment 4A:

1. Online service for annotating the files is shared with you.
2. Please mention in the readme file that you have finished online annotation.
3. If you had already annotated before getting the online tool, then your github Repo MUST have the annotation files including images and json.
4. This part of the assignment is 2000 pts

Assignment 4B:

1. Use this code to create **ResNet18** (https://keras.io/examples/cifar10_resnet/) Network
2. Train on CIFAR10 data.
3. Run for 50 Epochs and cross 88% Validation Accuracy
4. Show GradCam results for 10 images.
5. Submit on GitHub as Assignment 4B Folder inside Assignment 4 Folder
6. This part of the assignment is 2000 pts