Technische Universität Braunschweig

28-May-24

CNN Architecture – GoogLeNet, Inception, Xception

Group 5

Amulya Harihara Narayana Rao

Vigneshwar Karuppiah Ramanathan

M.Sc. Computational Sciences in Engineering

Outline

CNN Architecture – GoogLeNet, Inception, Xception

1. Convolutional Neural Networks (CNNs) and it’s challenges
2. GoogLeNet
3. Inception modules and network
4. Inception v3
5. Xception
6. Model architecture
7. Competition results
8. Advantages and disadvantages

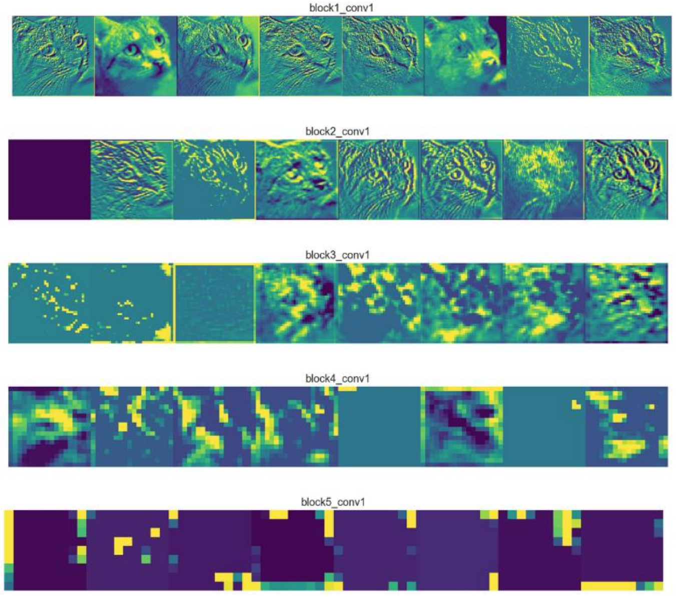
Introduction

What is Convolutional Neural Network (CNN)?

powerful class of neural networks that excel in tasks involving image and spatial data by leveraging convolutional operations to automatically learn and detect patterns

Traditional components of CNN:

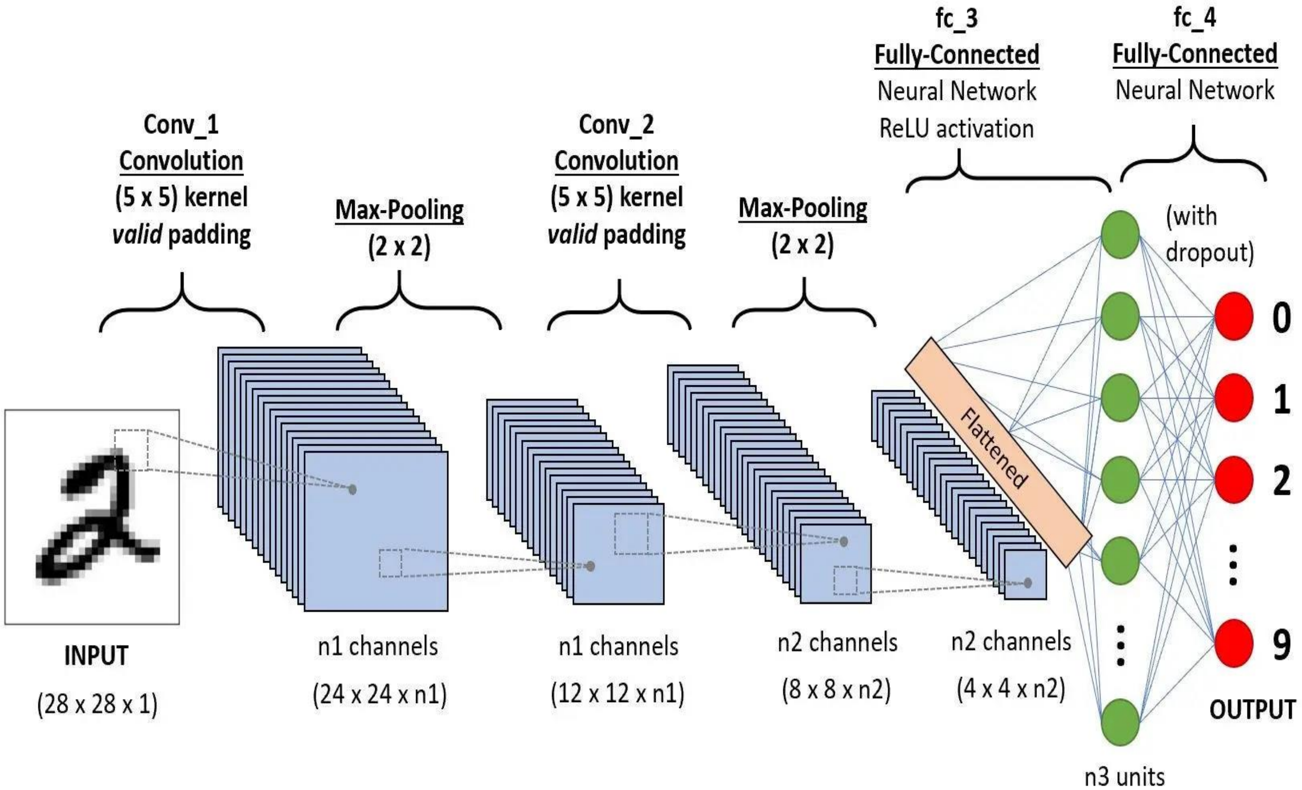
1. Convolutional layer
   - kernels / filters convolute over the images
   - produces feature maps
2. Pooling layer
   - dimensionality reduction
   - important information retained
   - controls overfitting, reduce computation
3. Fully Connected Layers
4. Softmax
5. Activation function (ReLu)
   - introduces non-linearity in CNN



Source : Medium

Building blocks of CNN

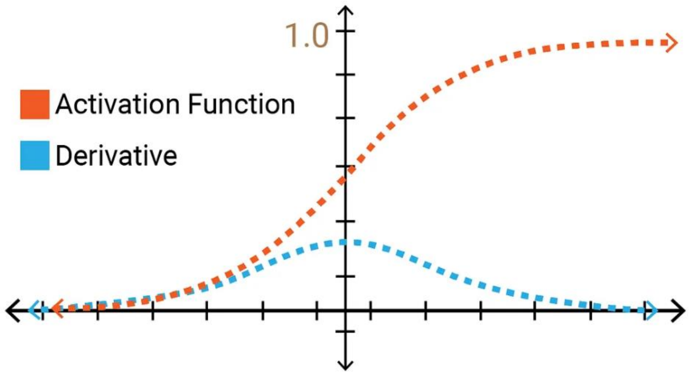
- Kernel / Filter
- Receptive field
- Feature map
- Pooling
- Stride
- Padding
- Activation function



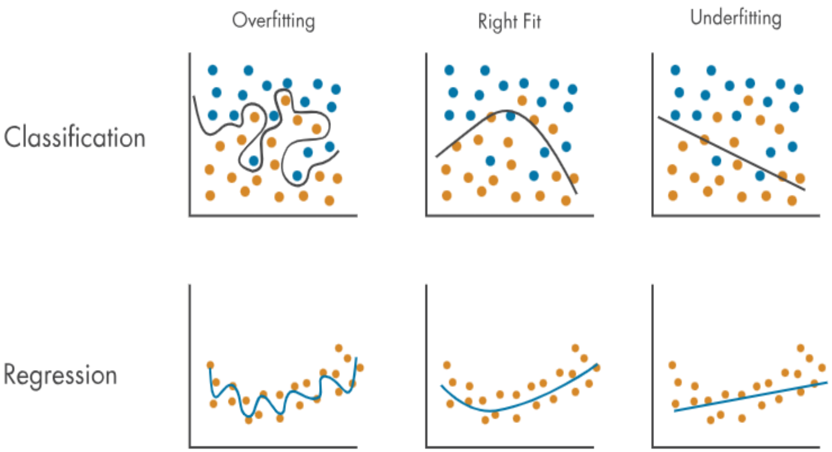
Source : SaturnCloud

Challenges in Deep learning

1. **Overfitting**
   - when a model learns to perform well on the training data but does not generalize well to unseen data
2. **Vanishing and Exploding Gradients**
   - gradients during backpropagation is either too small or too large
   - difficult for model to learn or unstable learning
3. **Computationally expensive**
4. **Model architecture selection**
5. **Insufficient data**
6. **Kernel size determination**



Source : codeodysseys

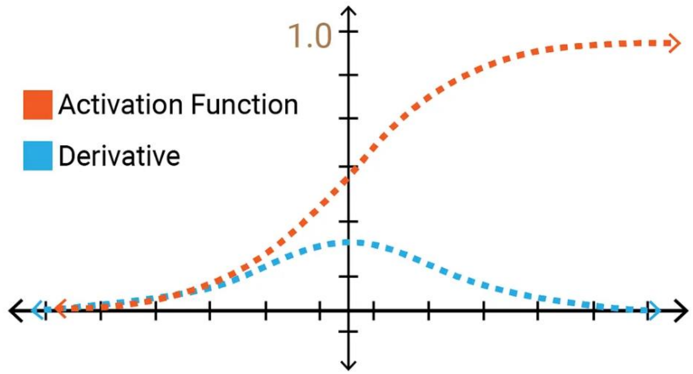


Source :MathWorks

Activation Functions

Vanishing and Exploding Gradients

- gradients during backpropagation is either too small or too large
- difficult for model to learn or unstable learning



ReLU

- Introduce non-linearity into the network
- Useful as most real time data would be non-linearity
- Negative pixels replaced by zero
- If input is greater than zero, output is the value itself
- f(x) = max (0,x)

Input

-249	-91	-37
250	-134	101
27	61	-153

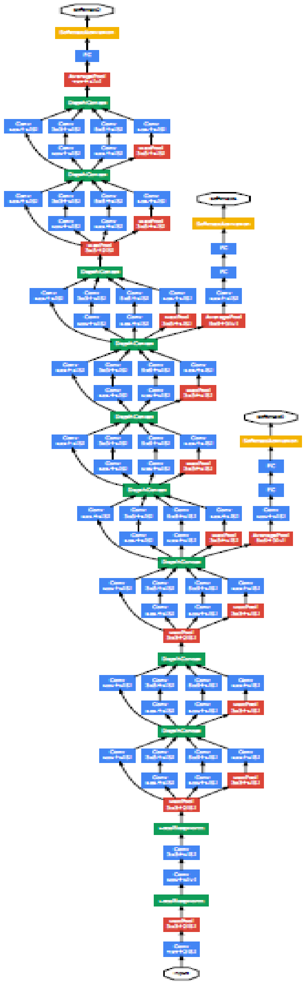
ReLU

0	0	0
250	0	101
27	61	0

Source :Researchgate

GoogLeNet

- Developed by researchers at **Google** in 2014
- research paper: **Going Deeper with Convolutions** (Paper)
- **Inception V1** is the architecture
- **GoogLeNet** is the specific implementation of Inception V1
- Team name in ILSVRC14 competition
- winner at the ILSVRC14 for task1b (object detection with additional training data)
- Runner at the ILSVRC14 for Task2a (Classification + localization)
- significant **decrease in error rate** compared to previous models like AlexNet, ZF-Net, and VGG



Source : (Paper)

Motivation

Traditional approach to improve performance

- Increase the size (Depth, Width)

Problems faced:

- Large number of parameters
- Prone to overfitting
- Dataset creation is tricky
- Vanishing gradient problem
- Increase in computational resources

Designing CNNs in a nutshell. Fun fact, this meme was referenced in the first inception net paper.

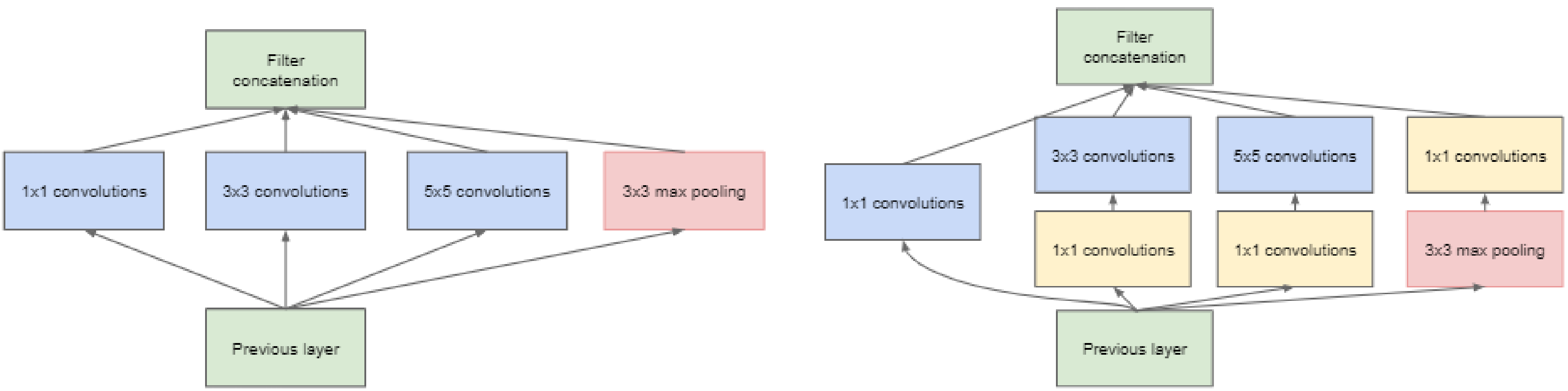
Motivation to develop GoogLeNet

- Improve utilization of **computing resources**
- To check feasibility of **sparsely connected architecture**

GoogleNet handles this in with introduction of **Inception modules**

Inception Module

- handles **objects at multiple scale** better
- use **filters** with **multiple size** on the **same level**
- Network gets **wider** rather than deeper
- *1×1, 3×3, 5×5* convolution and *3×3* max pooling performed in a **parallel way**
- output of these are **stacked together** to generated final output
- **9 modules** stacked linearly in GoogLeNet



(a) Inception module, naïve version (b) Inception module with dimension reductions

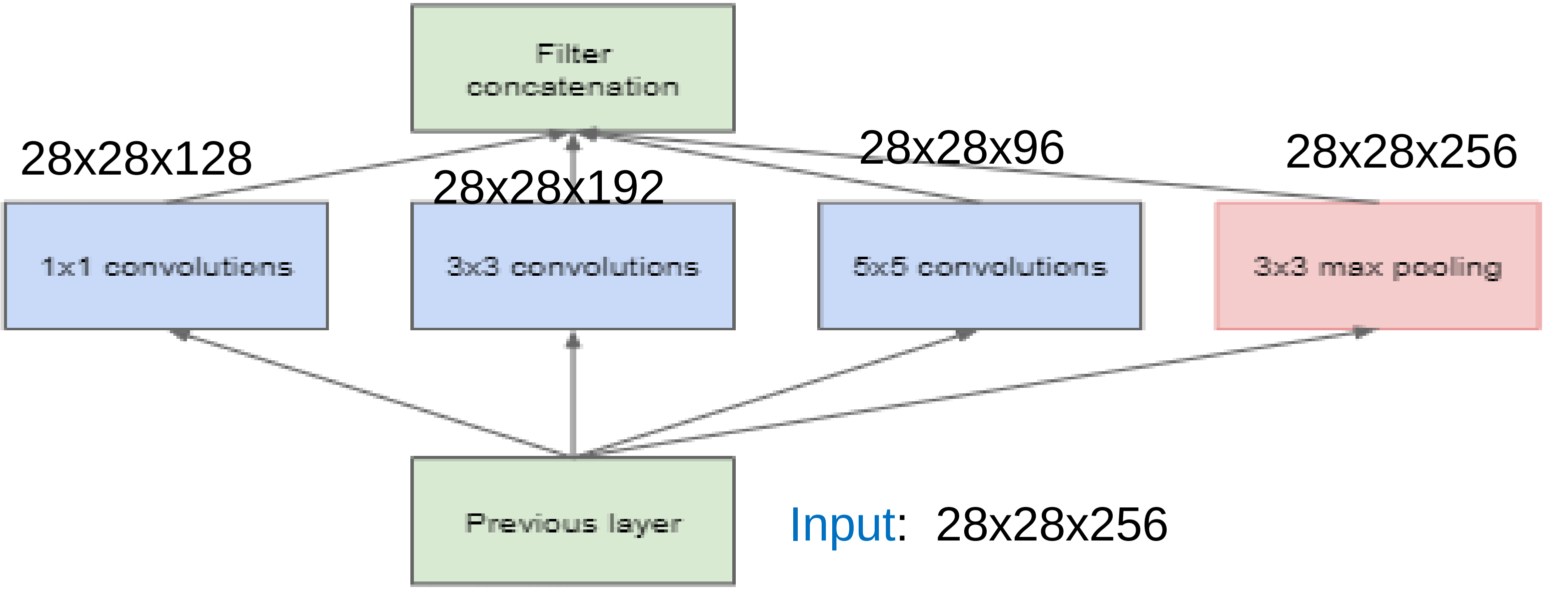
Source : (Paper)

Inception Module (Naïve version)

- Very expensive computation

Total Params: 28x28x672

Total # of convolutional operation: 854 M



(a) Inception module, naïve version

Source : (Paper)

Inception Module with dimension reductions

- 1x1 convolutions are **cheaper**
- compute **reductions** before the expensive 3x3 and 5x5 convolutions
- Also include ReLu
- decrease in **number of parameters** (weights and biases)

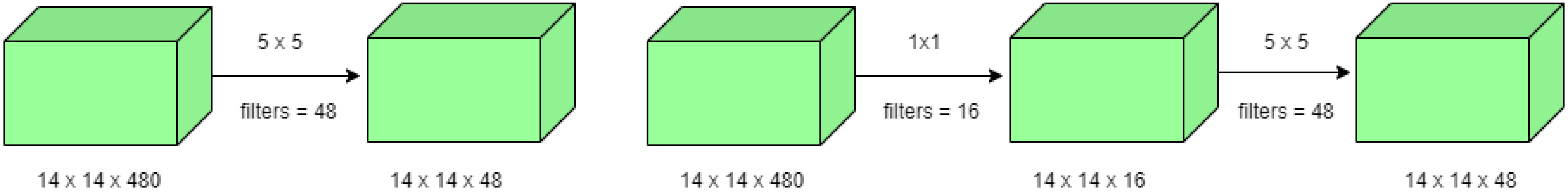
5×5 convolution

No intermediate 1x1 convolution Total Number of operations : *112.9 M*

5×5 convolution with 1x1 convolution

Total Number of operations : *5.3 M*

Much smaller than 112.9 M



Preserves spatial dimensions, only reduces depth

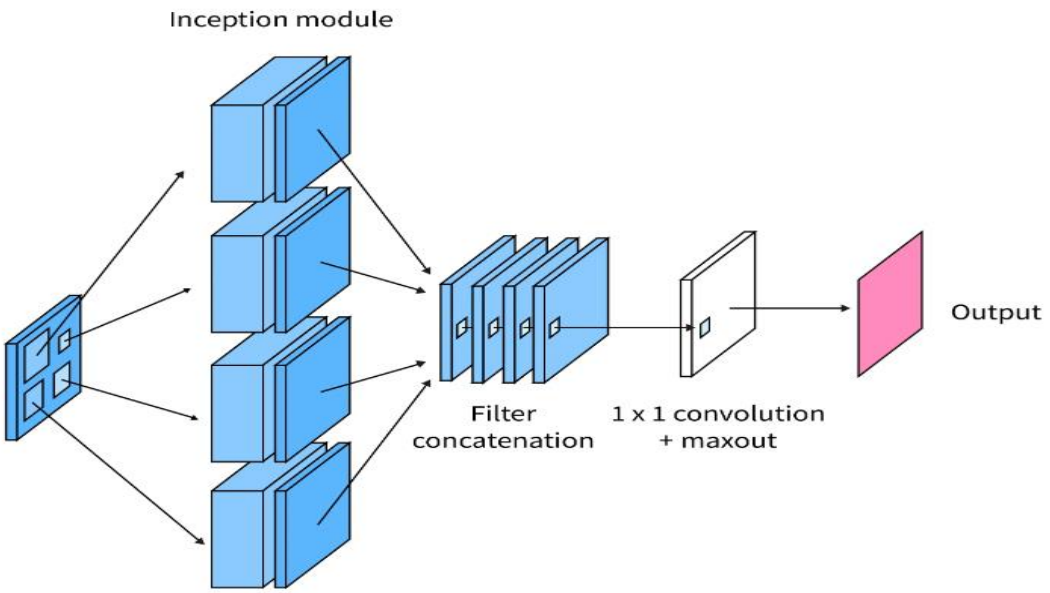
Source : Geeksforgeeks

Inception Network

- Modules **stacked** upon each other
- Occasional **max pooling layer** with stride 2
- significantly increase in # of units with **low computational complexity**

Recommendation:

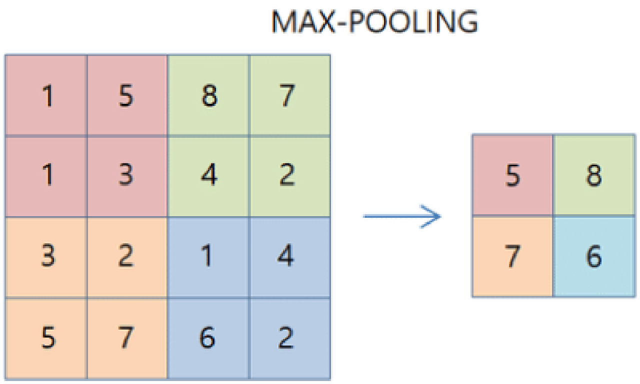
- Use inception modules in higher layers
- Lower layers should be traditional convolution fashion



Source : Scaler

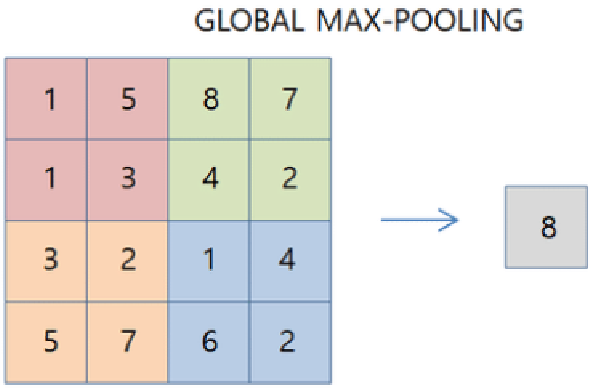
GoogLeNet architecture

- Input: 224 x 224 x 3 (RGB image)
- 9 inception modules stacked linearly
- 22 layers deep (27, including the pooling layers)
- uses global average pooling at the end



Global Average Pooling

- Previous models used fully connected layers
- Increase in computation
- High number of parameters
- Therefore, Global average pooling used at the end
- Improve in accuracy by 0.6%



Refer Googlenet architecture.pdf

Source : ResearchGate

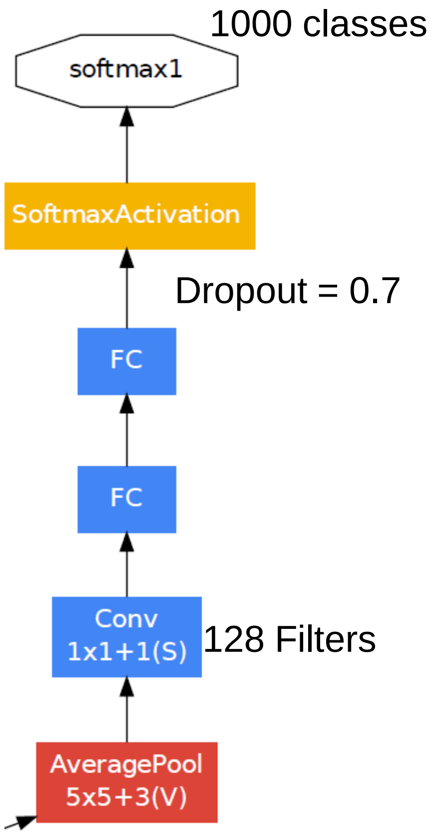
Auxiliary classifier

Motivation:

- As depth of network increases, we might face vanishing gradient problem

Auxiliary classifiers:

- Present in **middle of architecture**
- Used during **training only**, removed during inference
- Provides **additional regularization**
- Helps with **vanishing gradient problem**
- Their loss gets **added to the total loss** of the network with weight of 0.3



Source : (Paper)

Model Architecture

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

Source : (Paper) Table 1: GoogLeNet incarnation of the Inception architecture

ILSVRC 2014 Classification Challenge

- 1000 leaf node categories
- 1.2 M images for training
- 50,000 images for validation
- 100,000 images for testing
- Top 5 error of 6.67%
- 56.5% reduction when compared to supervision
- 40% reduction when compared to Clarifai

Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no

Table 2: Classification performance

ILSVRC 2014 Detection Challenge

- 200 output classes
- 50% threshold
- Reported using mAP
- "ensemble" indicates the number of different models combined to improve the detection performance of the system. ‘?’ indicates the number of models used is unclear

Team	Year	Place	mAP	external data	ensemble	approach
UvA-Euvision	2013	1st	22.6%	none	?	Fisher vectors
Deep Insight	2014	3rd	40.5%	ImageNet 1k	3	CNN
CUHK DeepID-Net	2014	2nd	40.7%	ImageNet 1k	?	CNN
GoogLeNet	2014	1st	43.9%	ImageNet 1k	6	CNN

Table 4: Detection performance

Conclusion

- Shift to a sparsely connected network is feasible and useful idea too
- Handles vanishing gradient problem well
- Computationally efficient
  - Parallel way
  - Dimensionality reduction
- Improved feature extraction
- Reduced overfitting

Disadvantages

- Complexity in design
- Choice of hyperparameter

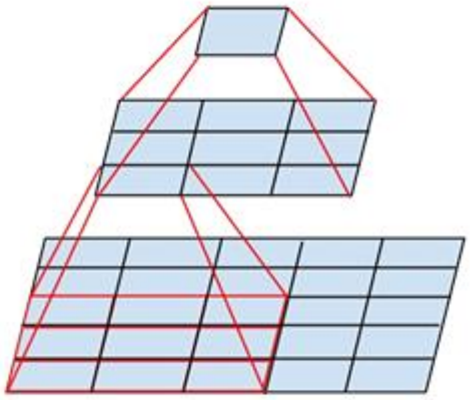
Inception - History

Year	Inception Version	Highlights	Source
2014	Inception V1	Basic Inception Network (GoogLeNet)	2014 Paper
2015	Inception V2	Batch Normalisation	2015 Paper
	Inception V3	Redesign of Network (Factorising Convolutional Kernel)	
2016	Inception V4	- Streamlined version of V3 - More uniform Architecture - Better Recognition Performance	2016 Paper
	Inception-ResNet	- Fusion of Residual Block and the Inception Block - Addresses the vanishing gradient problem	

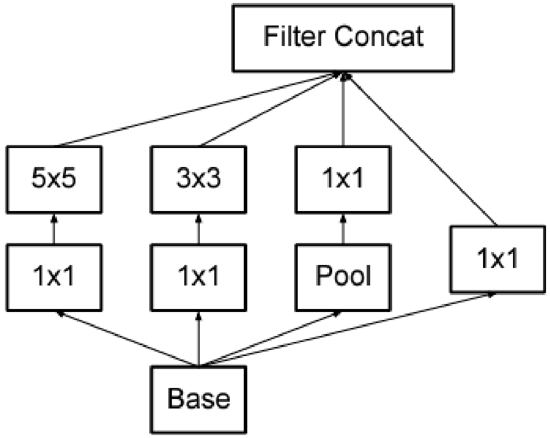
Inception V2/V3

Idea

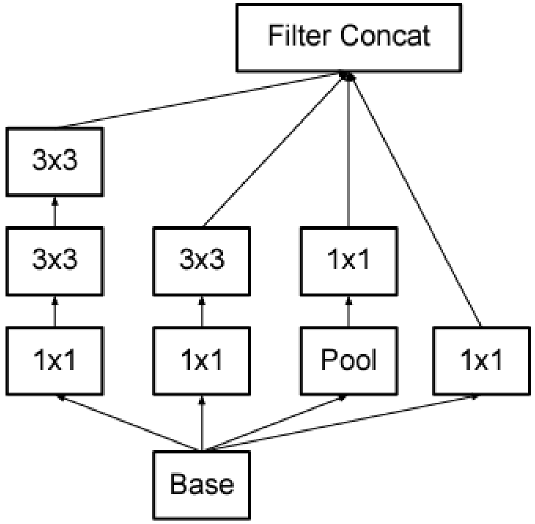
- Convolutions with large spatial filters (5x5 or 7x7) are computationally expensive
- Factorising larger convolutions into smaller ones with lesser parameters with same input size and output depth can reduce the computational load



(a)Mini Network replacing the 5x5 convolutions



(b) Original Inception V1 Module



(c) Inception Module proposed

- Batch Normalisation at the input of each Layer to improve learning rate and accuracy

Inception V2/V3

Architecture

type	patch size/stride or remarks	input size
conv	$3\times3/2$	$299\times299\times3$
conv	$3\times3/1$	$149\times149\times32$
conv padded	$3\times3/1$	$147\times147\times32$
pool	$3\times3/2$	$147\times147\times64$
conv	$3\times3/1$	$73\times73\times64$
conv	$3\times3/2$	$71\times71\times80$
conv	$3\times3/1$	$35\times35\times192$
$3\times$Inception	As in figure 5	$35\times35\times288$
$5\times$Inception	As in figure 6	$17\times17\times768$
$2\times$Inception	As in figure 7	$8\times8\times1280$
pool	8×8	$8\times8\times2048$
linear	logits	$1\times1\times2048$
softmax	classifier	$1\times1\times1000$

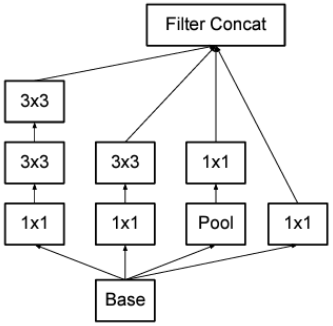


Fig 5

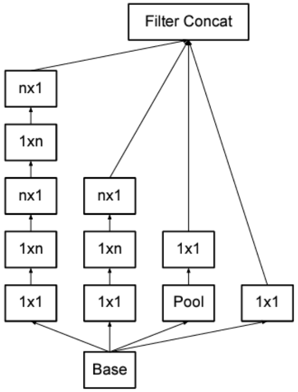


Fig 6

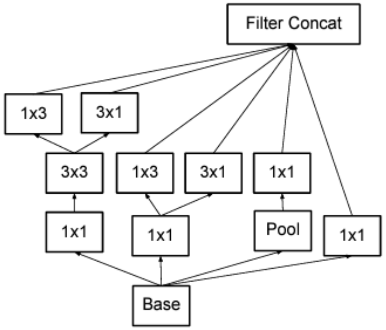


Fig 7

Inception V2/V3

Experimental Results

Network	Top-1 Error	Top-5 Error	Cost Bn Ops
GoogLeNet [20]	29%	9.2%	1.5
BN-GoogLeNet	26.8%	-	**1.5**
BN-Inception [7]	25.2%	7.8	2.0
Inception-v2	23.4%	-	3.8
Inception-v2 RMSProp	23.1%	6.3	3.8
Inception-v2 Label Smoothing	22.8%	6.1	3.8
Inception-v2 Factorized 7×7	21.6%	5.8	4.8
Inception-v2 BN-auxiliary	**21.2%**	**5.6%**	4.8

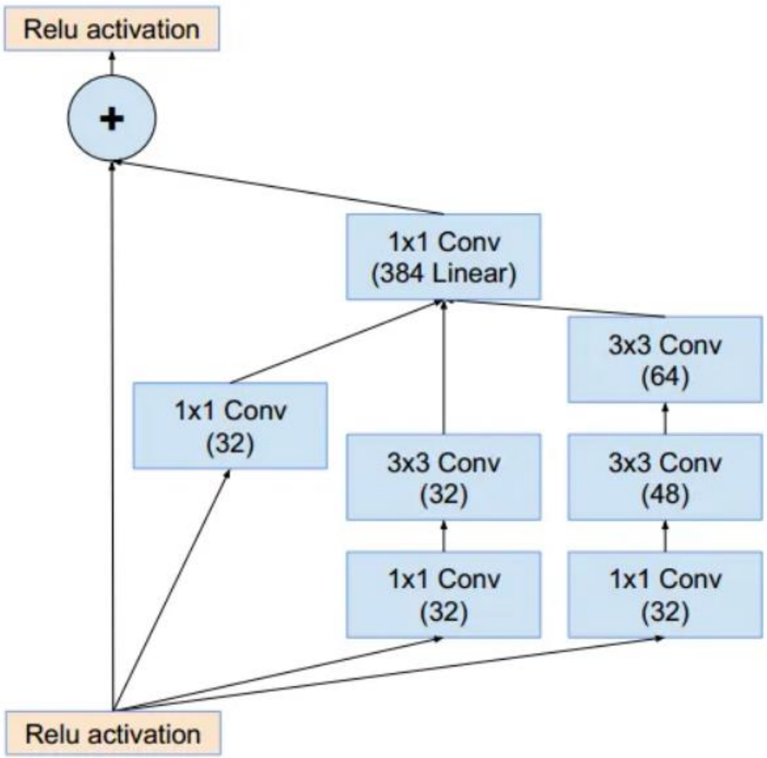
(a) Experimental Results (Single Crop)

Network	Crops Evaluated	Top-5 Error	Top-1 Error
GoogLeNet [20]	10	-	9.15%
GoogLeNet [20]	144	-	7.89%
VGG [18]	-	24.4%	6.8%
BN-Inception [7]	144	22%	5.82%
PReLU [6]	10	24.27%	7.38%
PReLU [6]	-	21.59%	5.71%
Inception-v3	12	19.47%	4.48%
Inception-v3	144	**18.77%**	**4.2%**

(a) Experimental Results (Multi-crop) (ILSVRC 2012)

Inception ResNet V2

Combination of residual connections and Inception Architecture

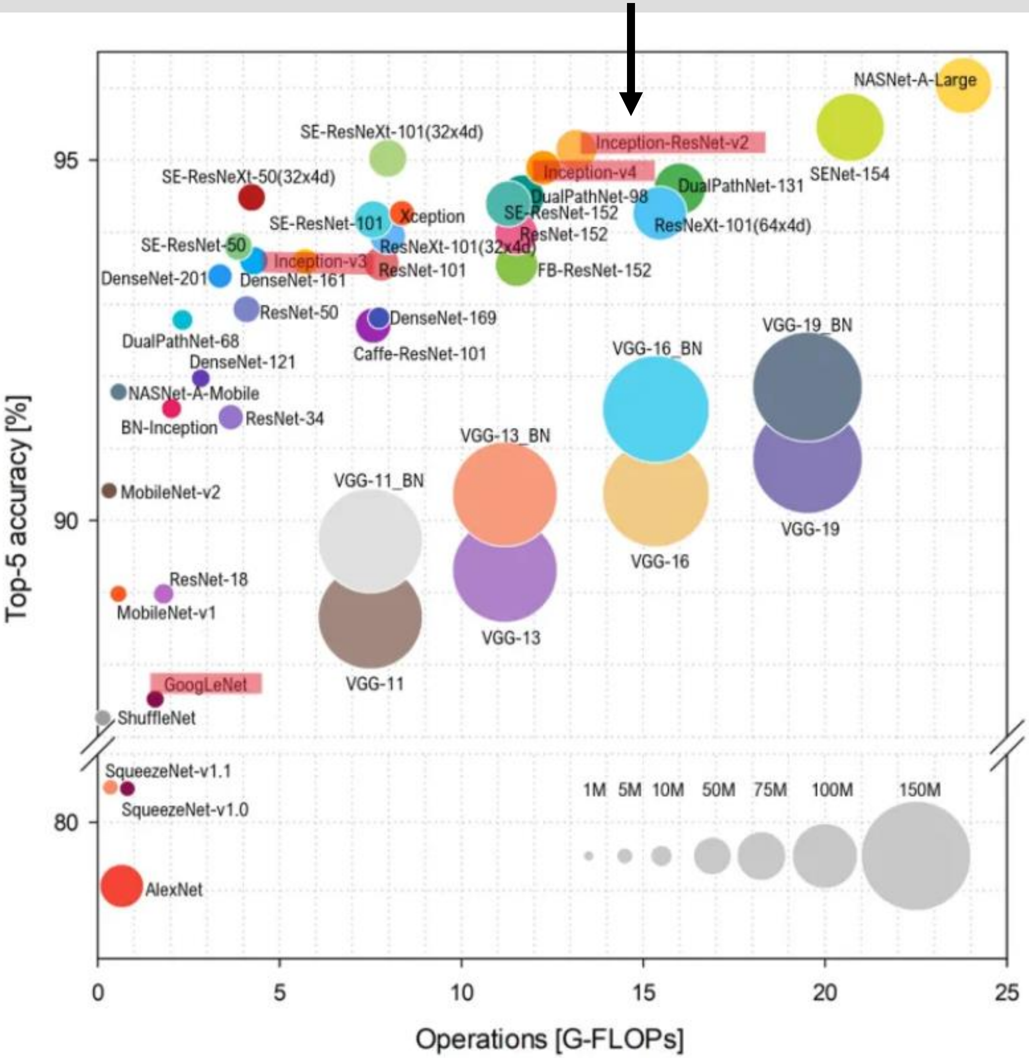


Network	Crops	Top-1 Error	Top-5 Error
ResNet-151 [5]	10	21.4%	5.7%
Inception-v3 [15]	12	19.8%	4.6%
Inception-ResNet-v1	12	19.8%	4.6%
Inception-v4	12	18.7%	4.2%
Inception-ResNet-v2	12	18.7%	4.1%

Xception

- Developed by researchers at Google in 2017
- Xception: Deep Learning with Depthwise Separable Convolutions (Paper)
- even better than **Inception-v3**
- Inspired from Inception-v3
- Involves Depthwise Separable Convolutions
- Lightweight
- overperformed VGG-16, ResNet and Inception V3 in most challenges

Inception – Performance



How does Xception work?

Depends on two concepts:

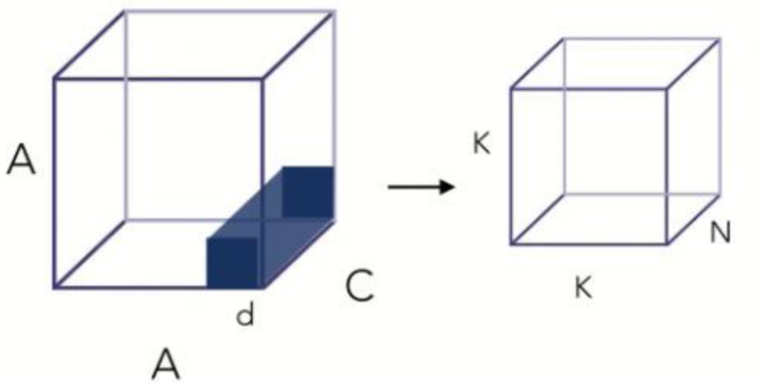
- Depthwise Separable convolution
- Shortcuts between convolution blocks

Limitations of convolutions:

Expensive operation

For 1 Kernel, we have $\boldsymbol{K^2} \times \boldsymbol{d^2} \times$ C

For N Kernel, we have $\boldsymbol{K^2} \times \boldsymbol{d^2} \times$ C $\times$ N



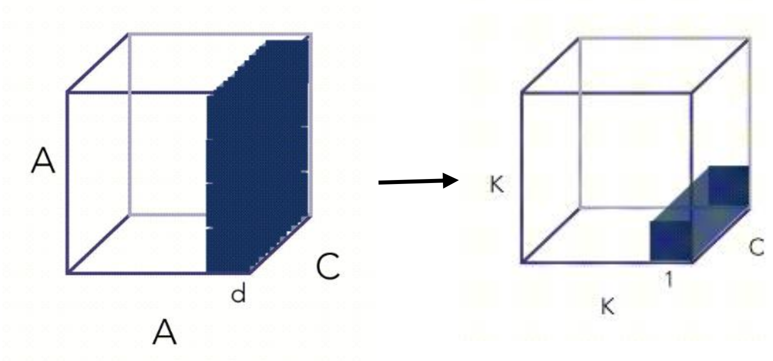
K = A = 100 x 100, d = 3 x 3,

C = 3

Source : biped.ai

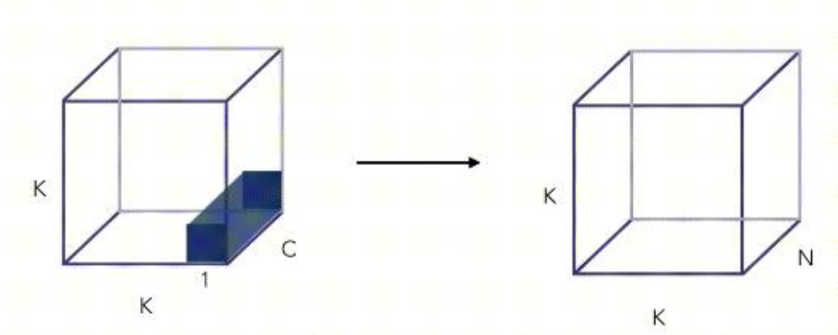
Depthwise convolution

- Convolution **not performed** over all channels (d x d x C)
- Instead we do it for **1 channel** (d x d x 1)
- Convoluted for only 1 filter and **not N filter**.



Pointwise convolution

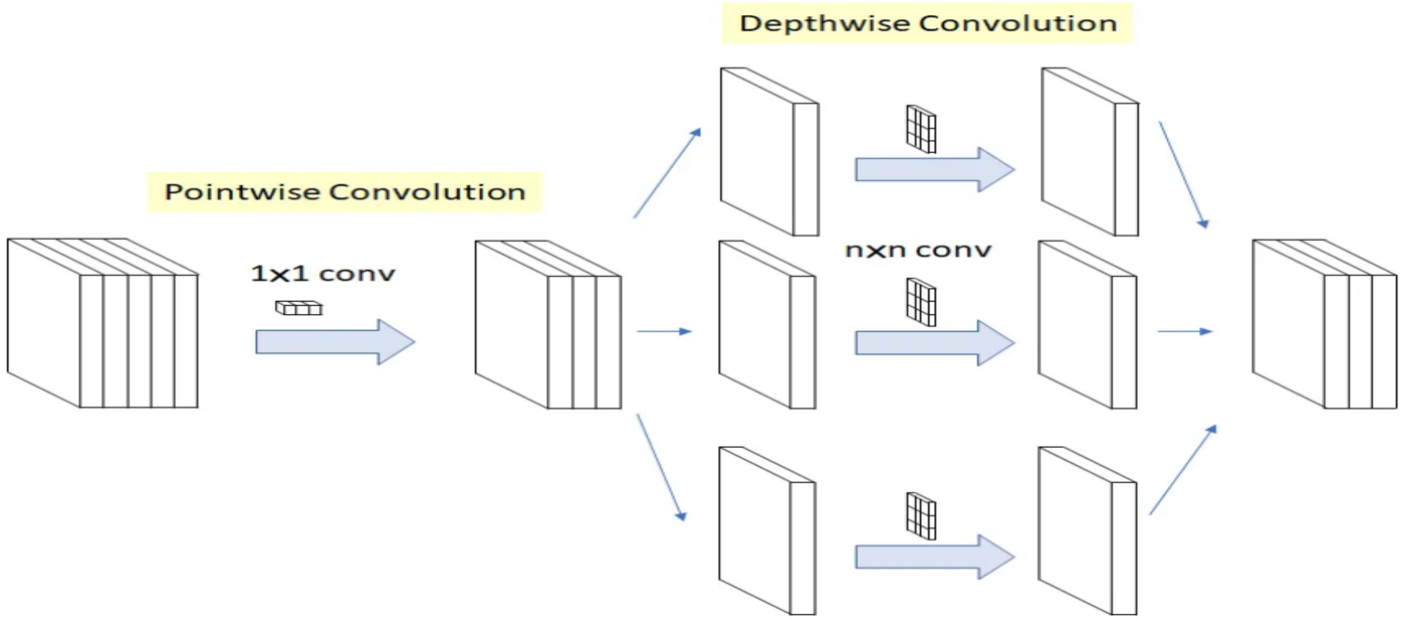
- Classical convolution
- Size : 1 x 1 x N
- After convolution : K x K x N
- # of operations reduced by factor of $\frac{1}{N}$



Source : biped.ai

Xception architecture

- Modified depthwise separable convolution
- Inspired from Inception v3
- No intermediate ReLU

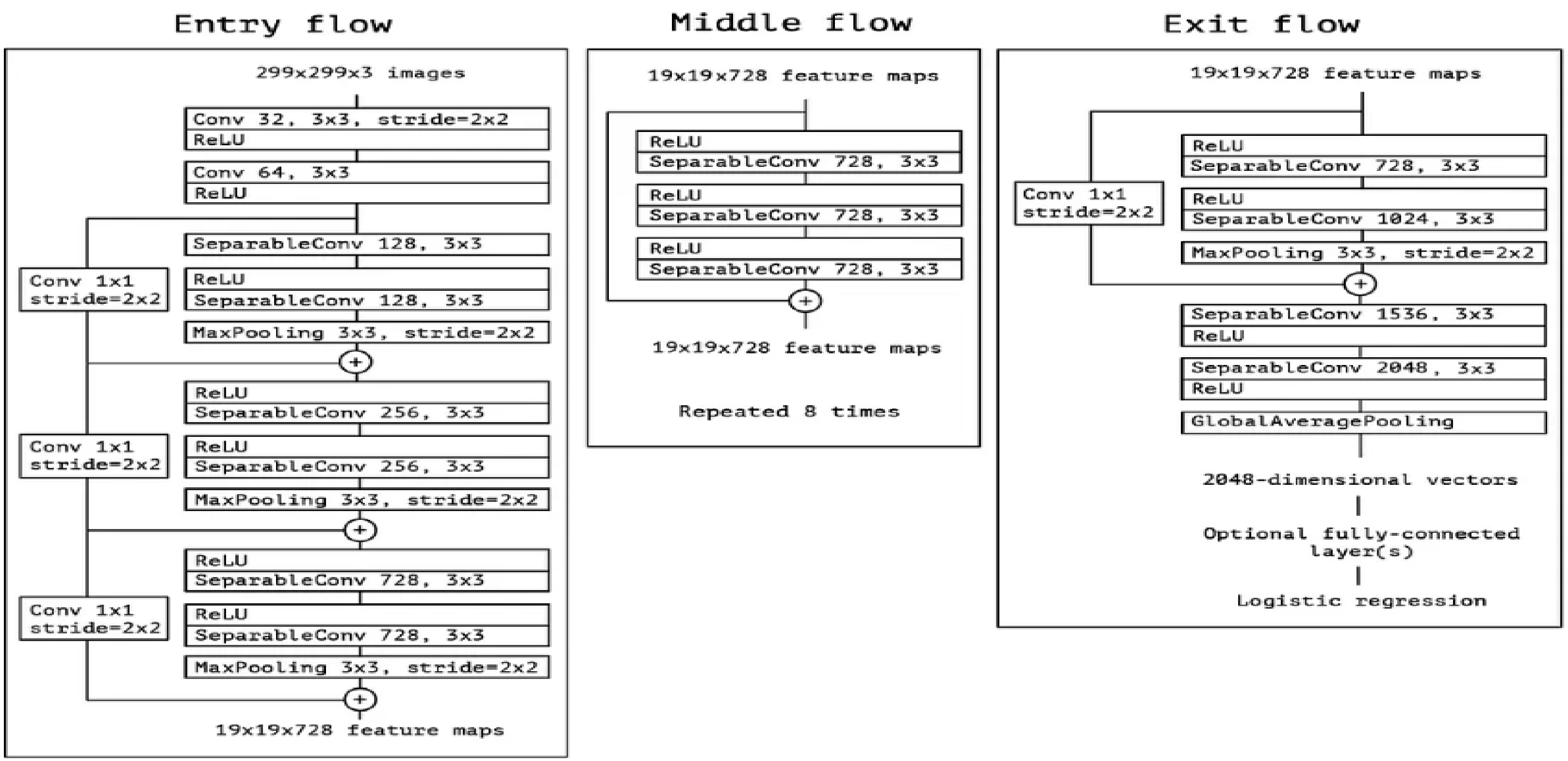


The Modified Depthwise Separable Convolution used as an Inception Module in Xception, so called "extreme" version of Inception module (n=3 here)

Source : Towardsdatascience

Model Architecture

3. Overall Architecture



Overall Architecture of Xception (Entry Flow > Middle Flow > Exit Flow)

Source : (Paper)

ILSVRC Challenge

- 1000 categories
- 1.3 M training images
- 50,000 validation images
- 100,000 testing images
- Outperforms VGGNet, ResNet, Inception v3

		Top-1 accuracy	Top-5 accuracy
VGGNet – 1st Runner Up in ILSVRC 2014	VGG-16	0.715	0.901
ResNet – Winner in ILSVRC 2015	ResNet-152	0.770	0.933
Inception-v3 – 1st Runner Up in ILSVRC 2015	Inception V3	0.782	0.941
	Xception	**0.790**	**0.945**

ImageNet: Xception has the highest accuracy

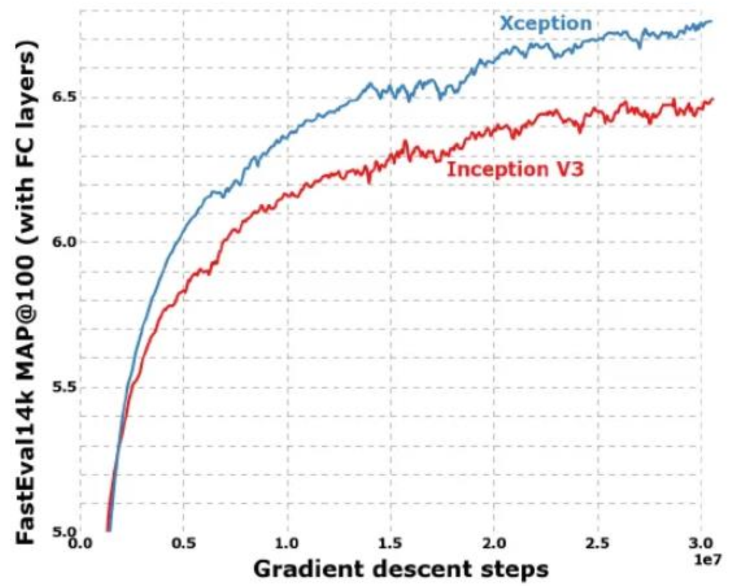
Source : (Paper)

JFT – FastEval14k

- **JFT** is Google’s internal dataset
  - 350M images with 17000 classes
- **FastEval14k** is an auxiliary dataset of google
  - 14000 images with 6000 classes
- mAP used as evaluation metric

	FastEval14k MAP@100
Inception V3 - no FC layers	6.36
Xception - no FC layers	6.70
Inception V3 with FC layers	6.50
Xception with FC layers	**6.78**

FastEval14k: Xception has highest mAP@100



FastEval14k: Validation Accuracy Against Gradient Descent Steps

Source : (Paper)

Conclusion

- Good results than it’s previous models
- Improved performance
- Enhanced feature extraction
- Easy to use in transfer learning case
- Good with classification tasks

Disadvantages

- Expensive to train
- Complexity in design
- Not good with other tasks

Reference

- **Going Deeper with Convolutions** (Paper)

  Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich

- **Xception: Deep Learning with Depthwise Separable Convolutions** (Paper)

  François Chollet

Questions ?

Thank you