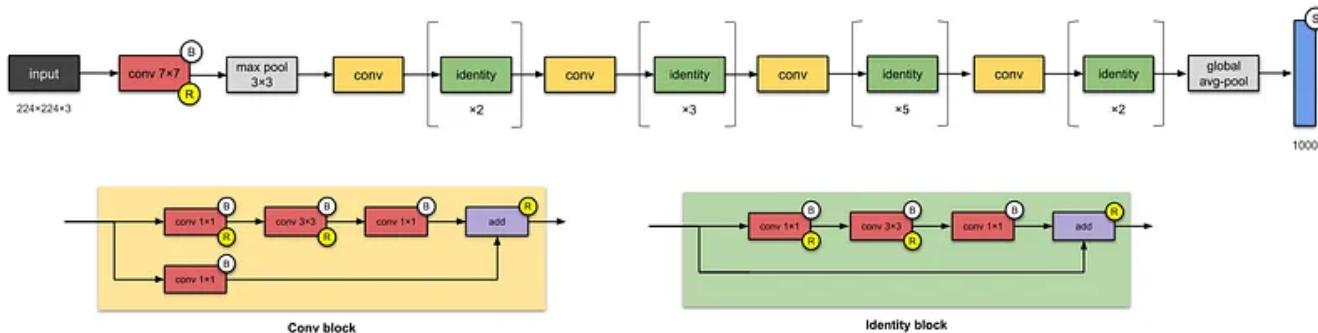


[Open in app](#)[Sign up](#)[Sign In](#)

Search Medium



What architecture is this? 🤔

You have **2 free member-only stories** left this month. [Sign up](#) for Medium and get an extra one.

★ Member-only story

INSIDE AI

Illustrated: 10 CNN Architectures

A compiled visualisation of the common convolutional neural networks



Raimi Karim · Follow

Published in Towards Data Science

12 min read · Jul 29, 2019

[Listen](#)[Share](#)

(*TL;DR – jump to the illustrations [here](#)*)

Changelog:

5 Jan 2022 – Fix typos and improve clarity

28 Nov 2020 – Updated “What’s novel” for every CNN

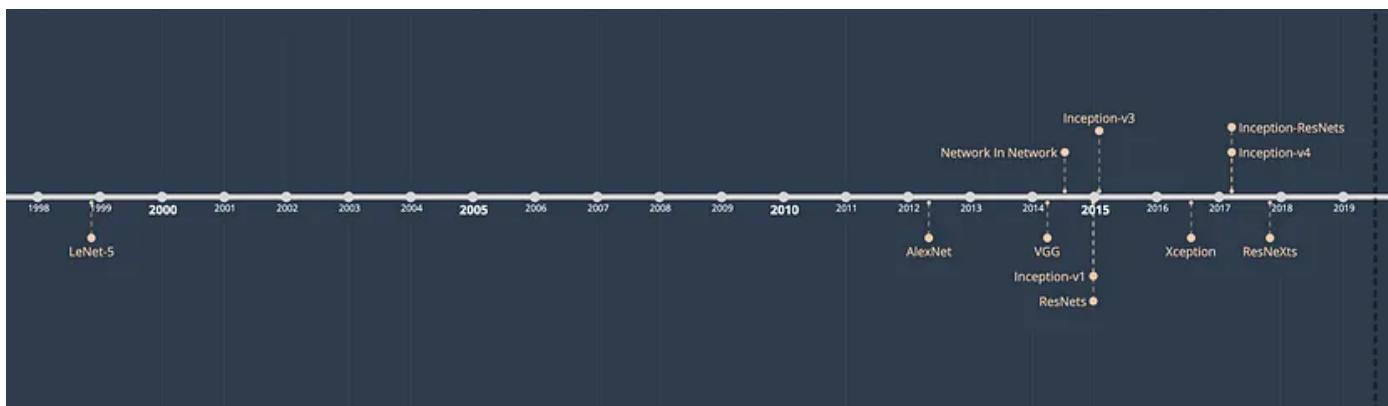
17 Nov 2020 – Edited no. of layers in last dense layer of Inceptionv1 from 4096 to 1000

24 Sep 2020 – Edited “What’s novel” section for ResNeXt-50

How have you been keeping up with the different convolutional neural networks (CNNs)? In recent years, we have witnessed the birth of numerous CNNs. These networks have gotten so deep that it has become extremely difficult to visualise the entire model. We stop keeping track of them and treat them as black-box models.

Fine, maybe you don't. But if you're guilty too then hey, you've come to the right place! This article is a visualisation of 10 *common* CNN architectures, hand-picked by yours truly. These illustrations provide a more compact view of the entire model, without having to scroll down a couple of times just to see the softmax layer. Apart from these images, I've also sprinkled some notes on how they 'evolved' over time — from 5 to 50 convolutional layers, from plain convolutional layers to modules, from 2–3 towers to 32 towers, from 7×7 to 5×5— but more on these later.

By 'common', I am referring to those models whose pre-trained weights are usually shared by deep learning libraries (such as TensorFlow, Keras and PyTorch) for users to use, and models that are usually taught in classes. Some of these models have shown success in competitions like the [ImageNet Large Scale Visual Recognition Challenge](#) (ILSVRC).



The 10 architectures that will be discussed and the year their papers were published.

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
VGG16	528 MB	0.713	0.901	138,357,544	23
InceptionV3	92 MB	0.779	0.937	23,851,784	159
ResNet50	98 MB	0.749	0.921	25,636,712	-
Xception	88 MB	0.790	0.945	22,910,480	126
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
ResNeXt50	96 MB	0.777	0.938	25,097,128	-

The top-1 and top-5 accuracy refers to the model's performance on the ImageNet validation dataset.

Depth refers to the topological depth of the network. This includes activation layers, batch normalization layers etc.

Pre-trained weights are available in Keras for 6 of the architectures that we will talk about. Adapted from a table in the [Keras documentation](#).

The motivation for writing this is that there aren't many blogs and articles out there with these compact visualisations (if you do know of any, please share them with me). So I decided to write one for our reference. For this purpose, I have read the papers and the code (mostly from TensorFlow and Keras) to come up with these vizzes.

Here I'd like to add that a plethora of CNN architectures we see in the wild are a result of many things — improved computer hardware, ImageNet competition, solving specific tasks, new ideas and so on. Christian Szegedy, a researcher at Google once mentioned that

“[m]ost of this progress is not just the result of more powerful hardware, larger datasets and bigger models, but mainly a consequence of new ideas, algorithms and improved network architectures.” (Szegedy et al, 2014)

Now let's get on with these beasts and observe how network architectures improve over time!

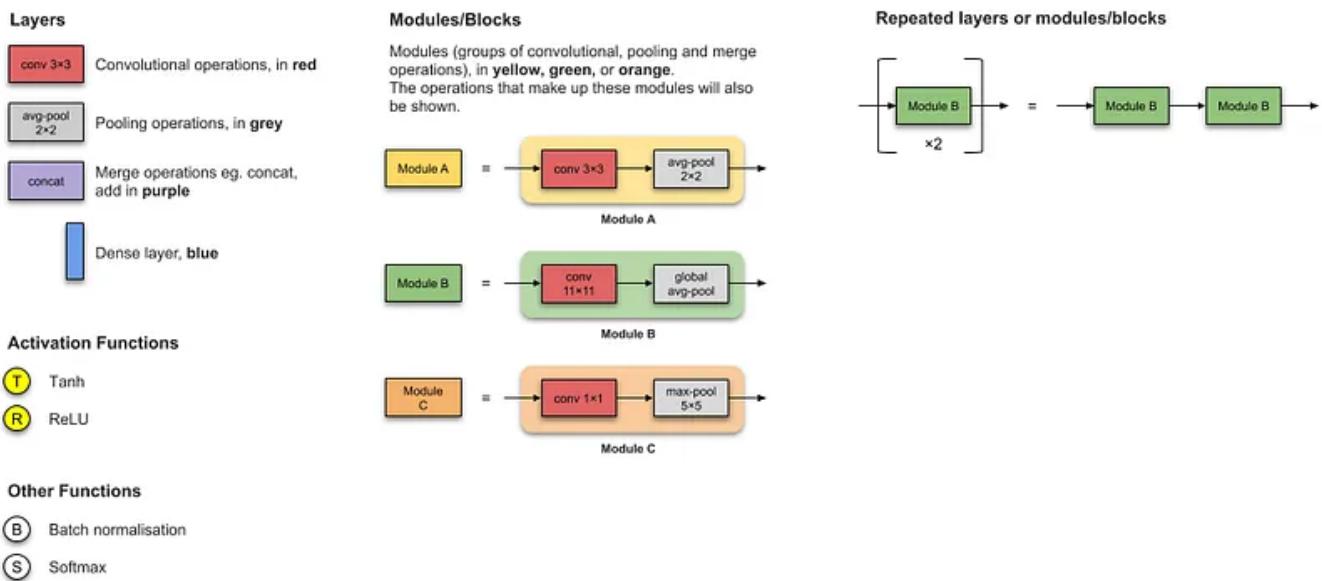
A note on the visualisations

Note that I have excluded information like the number of convolutional filters, padding, stride, dropouts, and the flatten operation in the illustrations.

Contents (ordered by year of publication)

1. [LeNet-5](#)
2. [AlexNet](#)
3. [VGG-16](#)
4. [Inception-v1](#)
5. [Inception-v3](#)
6. [ResNet-50](#)
7. [Xception](#)
8. [Inception-v4](#)
9. [Inception-ResNets](#)
10. [ResNeXt-50](#)

Legend



1. LeNet-5 (1998)

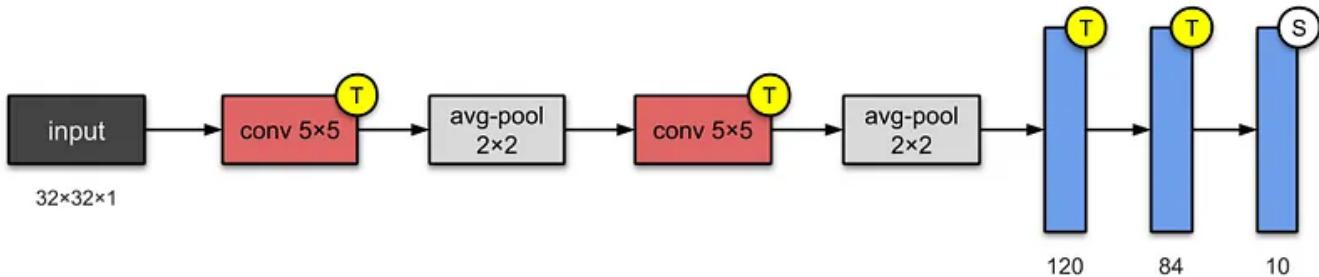


Fig. 1: LeNet-5 architecture, based on their [paper](#)

LeNet-5 is one of the simplest architectures. It has 2 convolutional and 3 fully-connected layers (hence “5” — it is very common for the names of neural networks to be derived from the number of *convolutional* and *fully connected* layers that they have). The average-pooling layer as we know it now was called a *sub-sampling layer* and it had trainable weights (which isn’t the current practice of designing CNNs nowadays). This architecture has about **60,000 parameters**.

★ What's novel?

This architecture has become the standard ‘template’: stacking convolutions with activation function, and pooling layers, and ending the network with one or more fully-connected layers.

Publication

- Paper: [Gradient-Based Learning Applied to Document Recognition](#)
- Authors: Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner
- Published in: Proceedings of the IEEE (1998)

2. AlexNet (2012)

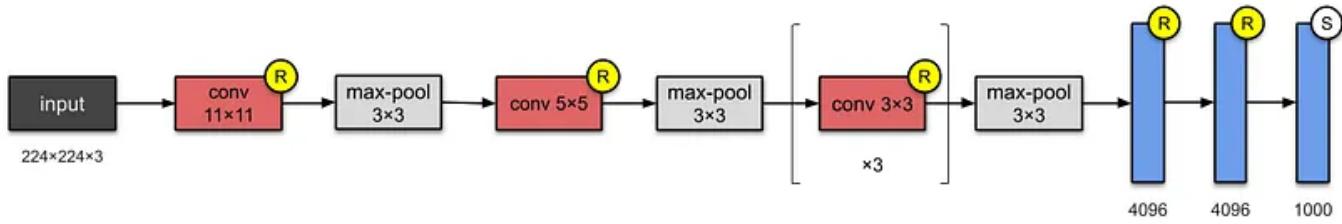


Fig. 2: AlexNet architecture, based on their [paper](#).

With **60M parameters**, AlexNet has 8 layers — 5 convolutional and 3 fully connected. AlexNet just stacked a few more layers onto LeNet-5. At the point of publication, the authors pointed out that their architecture was “one of the largest convolutional neural networks to date on the subsets of ImageNet.”

What's novel?

1. They were the first to implement Rectified Linear Units (ReLUs) as activation functions.
2. Dropout.

Publication

- Paper: [ImageNet Classification with Deep Convolutional Neural Networks](#)
- Authors: Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton. University of Toronto, Canada.

- Published in: NeurIPS 2012

3. VGG-16 (2014)

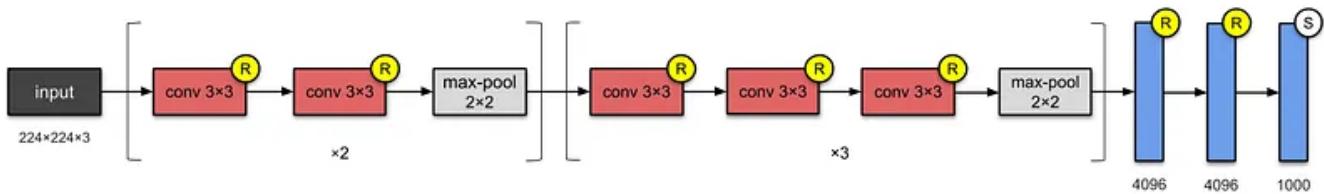


Fig. 3: VGG-16 architecture, based on their [paper](#).

By now you would've already noticed that CNNs were starting to get deeper and deeper. This is because the most straightforward way of improving the performance of deep neural networks is by increasing their size (Szegedy et. al). The folks at Visual Geometry Group (VGG) invented the VGG-16 which has 13 convolutional and 3 fully-connected layers, carrying with them the ReLU tradition from AlexNet. This network stacks more layers onto AlexNet, and use smaller size filters (2×2 and 3×3). It consists of **138M parameters** and takes up about 500MB of storage space 🤯. They also designed a deeper variant, VGG-19.

★ What's novel?

- As mentioned in their abstract, the contribution from this paper is the designing of *deeper* networks (roughly twice as deep as AlexNet). This was done by stacking uniform convolutions.

Publication

- Paper: [Very Deep Convolutional Networks for Large-Scale Image Recognition](#)
- Authors: Karen Simonyan, Andrew Zisserman. University of Oxford, UK.
- arXiv preprint, 2014

4. Inception-v1 (2014)

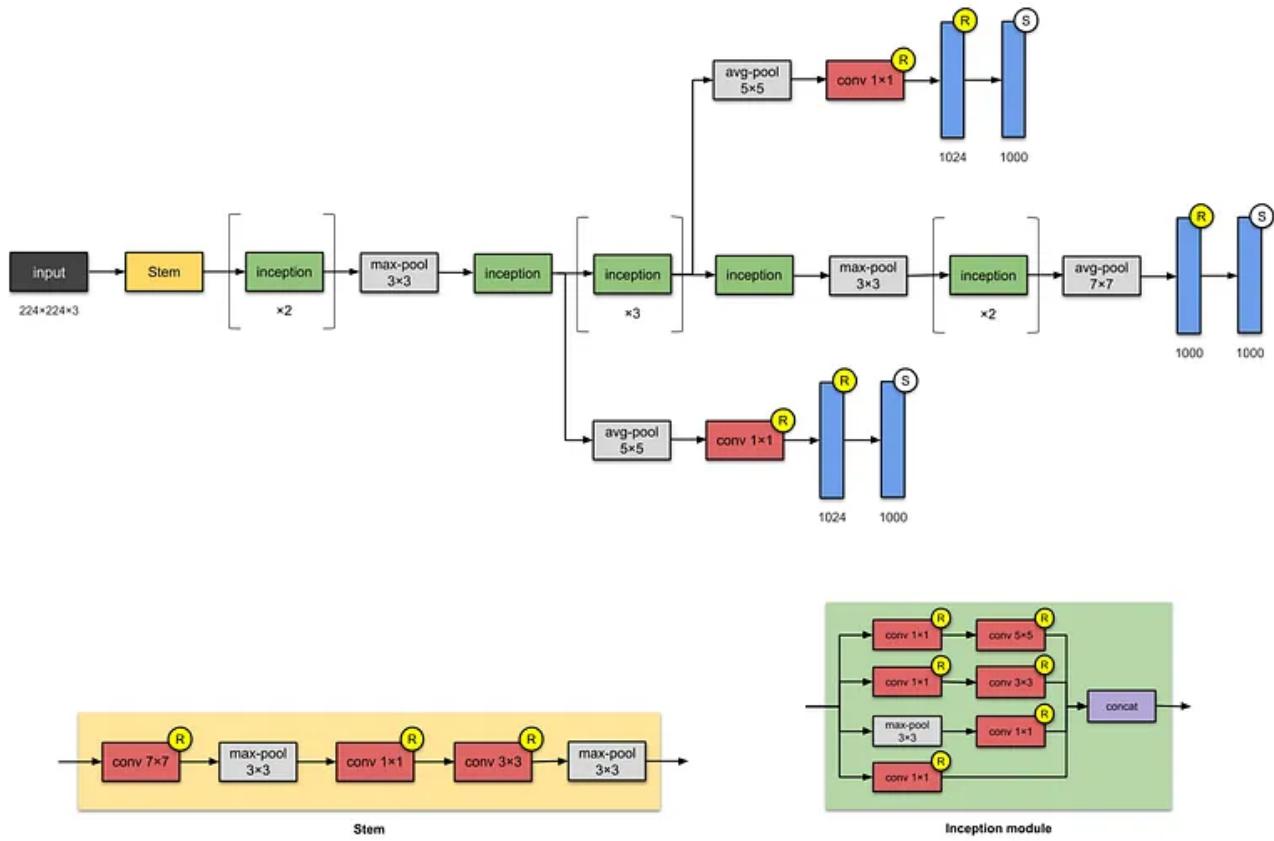


Fig. 4: Inception-v1 architecture. This CNN has two auxiliary networks (which are discarded at inference time). Architecture is based on Figure 3 in the [paper](#).

This 22-layer architecture with 5M parameters is called the Inception-v1. Here, the Network In Network (see [Appendix](#)) approach is heavily used, as mentioned in the paper. The Network In Network is implemented via *Inception modules*. The design of the architecture of an Inception module is a product of research on approximating sparse structures (read the paper for more!). Each module presents 3 ideas:

1. Having **parallel towers** of convolutions with different filters, followed by concatenation, captures different features at 1x1, 3x3 and 5x5, thereby ‘clustering’ them. This idea is motivated by Arora et al. in the paper *Provable bounds for learning some deep representations*, suggesting a layer-by-layer construction in which one should analyse the correlation statistics of the last layer and cluster them into groups of units with high correlation.
2. 1x1 convolutions are used for dimensionality reduction to remove computational bottlenecks.

3. Due to the activation function from 1×1 convolution, its addition also adds nonlinearity. This idea is based on the Network In Network paper. See appendix [here](#).
4. The authors also introduced two **auxiliary classifiers** to encourage discrimination in the lower stages of the classifier, to increase the gradient signal that gets propagated back, and to provide additional regularisation. The **auxiliary networks** (the branches that are connected to the auxiliary classifier) are discarded at inference time.

It is worth noting that “[t]he main hallmark of this architecture is the improved utilisation of the computing resources inside the network.”

Note:

The names of the modules (Stem and Inception) were not used for this version of Inception until its later versions i.e. Inception-v4 and Inception-ResNets. I have added them here for easy comparison.

★ What's novel?

1. Building networks using modules/blocks. Instead of stacking convolutional layers, we stack modules or blocks, within which are convolutional layers. Hence the name Inception (with reference to the 2010 sci-fi movie *Inception* starring Leonardo DiCaprio).

Publication

- Paper: [Going Deeper with Convolutions](#)
- Authors: Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. Google, University of Michigan, University of North Carolina
- Published in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

5. Inception-v3 (2015)

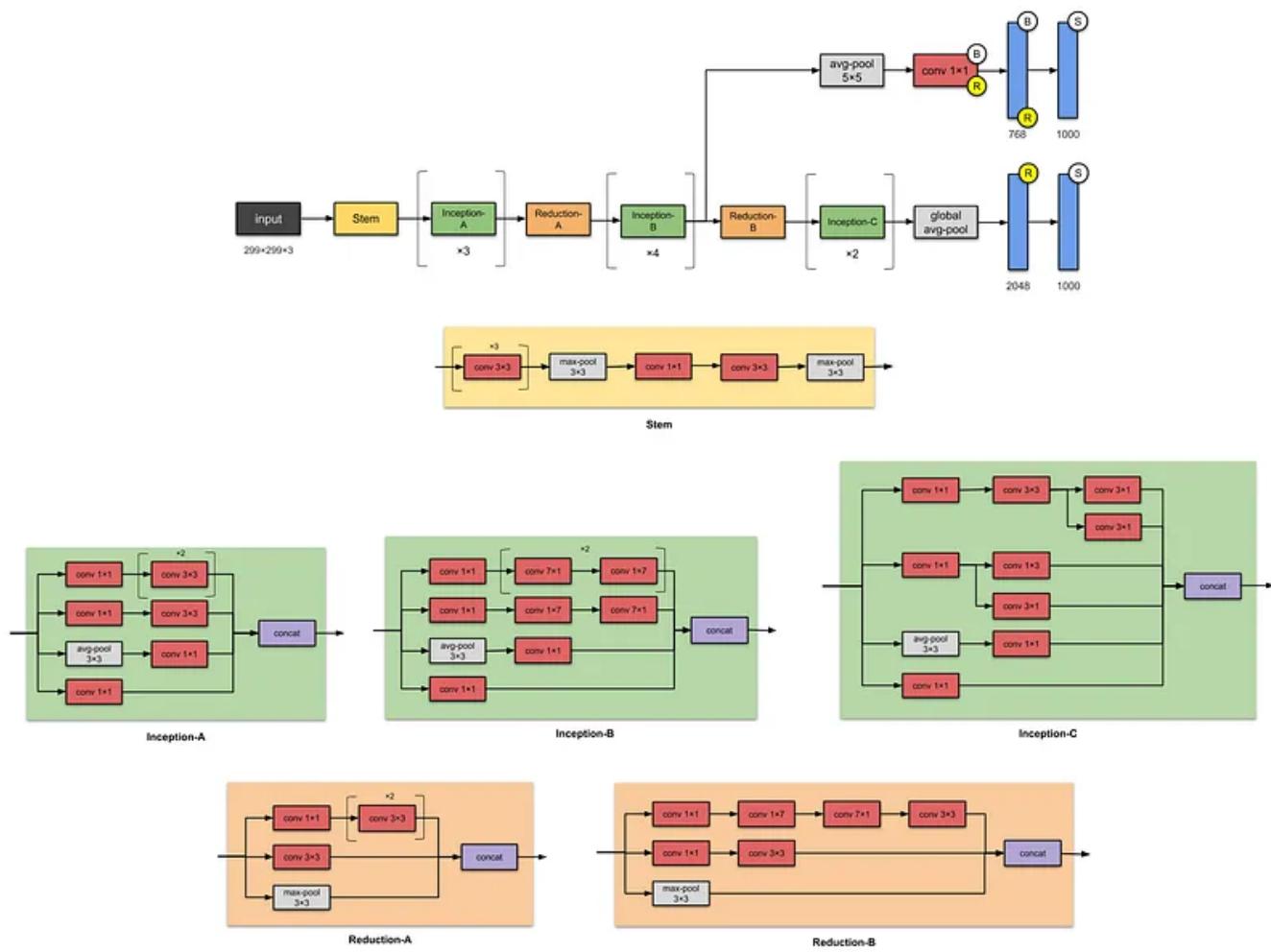


Fig. 5: Inception-v3 architecture. This CNN has an auxiliary network (which is discarded at inference time).
 *Note: All convolutional layers are followed by batch norm and ReLU activation. Architecture is based on their [GitHub code](#).

Inception-v3 is a successor to Inception-v1, with 24M parameters. Wait where's Inception-v2? Don't worry about it — it's an earlier prototype of v3 hence it's very similar to v3 but not commonly used. When the authors came out with Inception-v2, they ran many experiments on it and recorded some successful tweaks. Inception-v3 is the network that incorporates these tweaks (tweaks to the optimiser, loss function and adding batch normalisation to the auxiliary layers in the auxiliary network).

The motivation for Inception-v2 and Inception-v3 is to avoid *representational bottlenecks* (this means drastically reducing the input dimensions of the next layer) and have more efficient computations by using factorisation methods.

Note:

The names of the modules (Stem, Inception-A, Inception-B etc.) were not used for this

version of Inception until its later versions i.e. Inception-v4 and Inception-ResNets. I have added them here for easy comparison.

⭐ What's novel?

1. Among the first designers to use batch normalisation (not reflected in the above diagram for simplicity).

◆ What's improved from the previous version, Inception-v1?

1. Factorising $n \times n$ convolutions into asymmetric convolutions: $1 \times n$ and $n \times 1$ convolutions
2. Factorise 5×5 convolution to two 3×3 convolution operations
3. Replace 7×7 with a series of 3×3 convolutions

✍ Publication

- Paper: Rethinking the Inception Architecture for Computer Vision
- Authors: Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna. Google, University College London
- Published in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

6. ResNet-50 (2015)

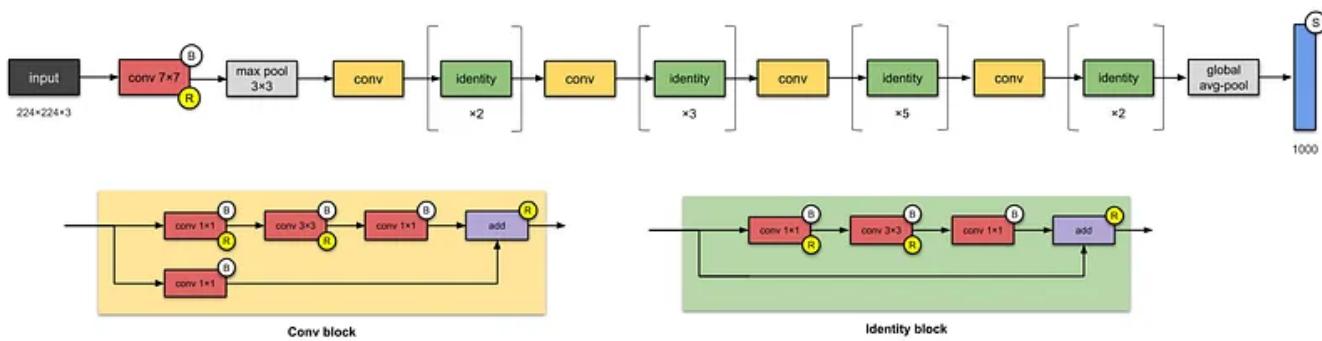


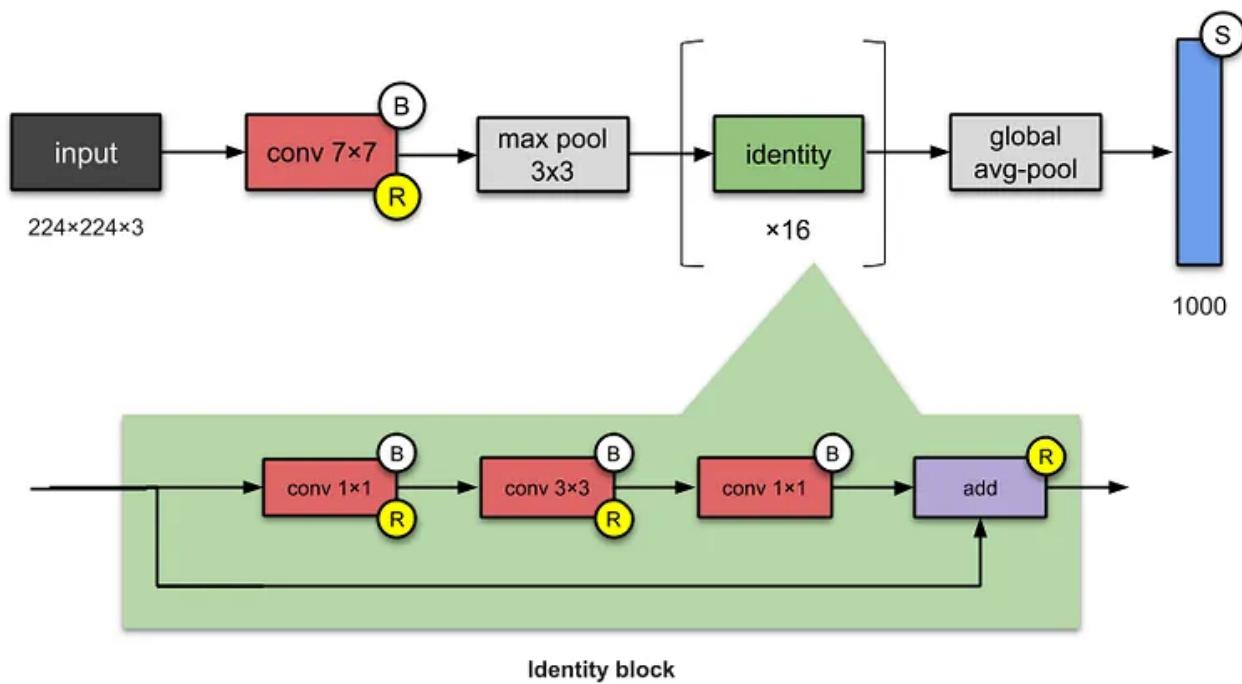
Fig. 6: ResNet-50 architecture, based on the GitHub [code](#) from keras-team.

Yes, it's the answer to the question you see on the top of the article [here](#) ("what architecture is this?").

From the past few CNNs, we have seen nothing but an increasing number of layers in the design and achieving better performance. But "with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly." The folks from Microsoft Research addressed this problem with ResNet — using skip connections (a.k.a. shortcut connections, residuals) while building deeper models.

ResNet is one of the early adopters of batch normalisation (the batch norm paper authored by Ioffe and Szegedy was submitted to ICML in 2015). Shown above is ResNet-50, with 26M parameters.

The basic building block for ResNets is the conv and identity blocks. Because they look alike, you might simplify ResNet-50 like this (don't quote me for this!):



★ What's novel?

1. *Popularised* skip connections (they weren't the first to use skip connections).

2. Designing even deeper CNNs (up to 152 layers) without compromising the model's generalisation power
3. Among the first to use batch normalisation.

Publication

- Paper: [Deep Residual Learning for Image Recognition](#)
- Authors: Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Microsoft
- Published in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

7. Xception (2016)

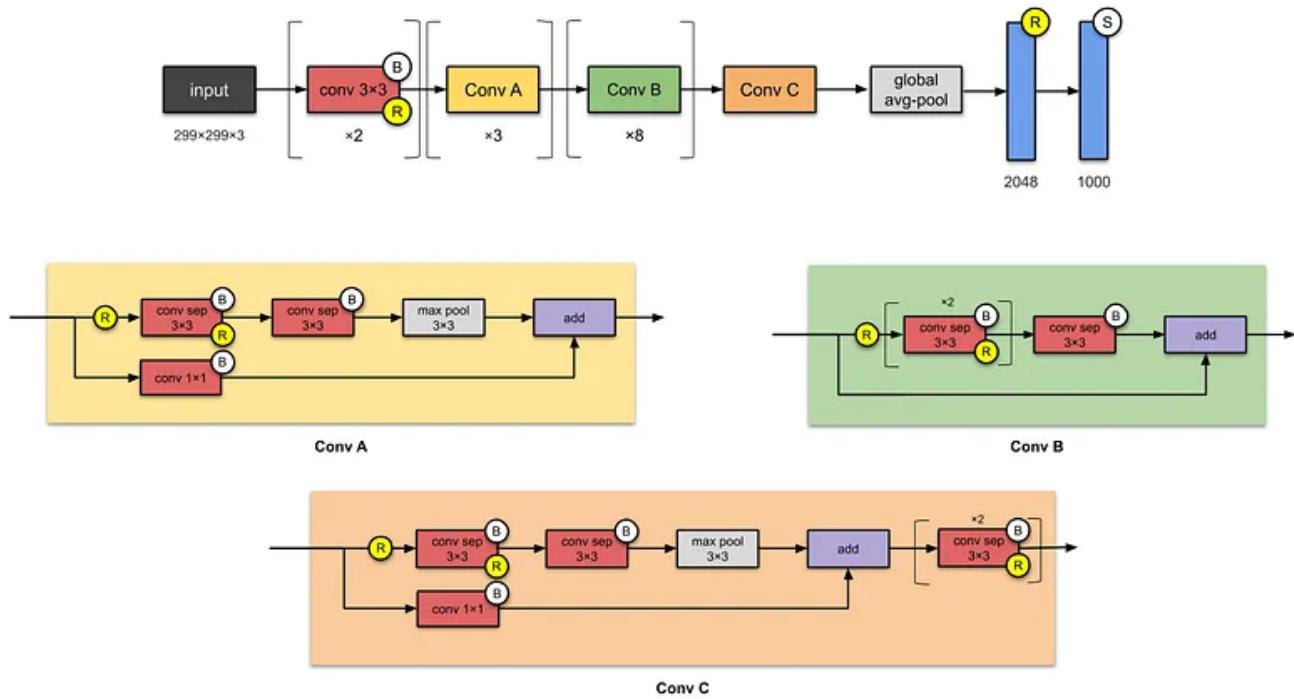


Fig. 7: Xception architecture, based on the GitHub [code](#) from keras-team. Depthwise separable convolutions are denoted by 'conv sep.'

Xception is an adaptation from Inception, where the Inception modules have been replaced with depthwise separable convolutions. It has also roughly the same number of parameters as Inception-v1 (23M).

Xception takes the Inception hypothesis to an *eXtreme* (hence the name). What's the Inception hypothesis again? Thank goodness this was explicitly and concisely mentioned in this paper (thanks François!).

- Firstly, cross-channel (or cross-feature map) correlations are captured by 1×1 convolutions.
- Consequently, spatial correlations within each channel are captured via the regular 3×3 or 5×5 convolutions.

Taking this idea to an extreme means performing 1×1 to *every* channel, then performing a 3×3 to *each* output. This is identical to replacing the Inception module with depthwise separable convolutions.

★ What's novel?

1. Introduced CNN based entirely on depthwise separable convolution layers.

Publication

- Paper: [Xception: Deep Learning with Depthwise Separable Convolutions](#)
- Authors: François Chollet. Google.
- Published in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

8. Inception-v4 (2016)

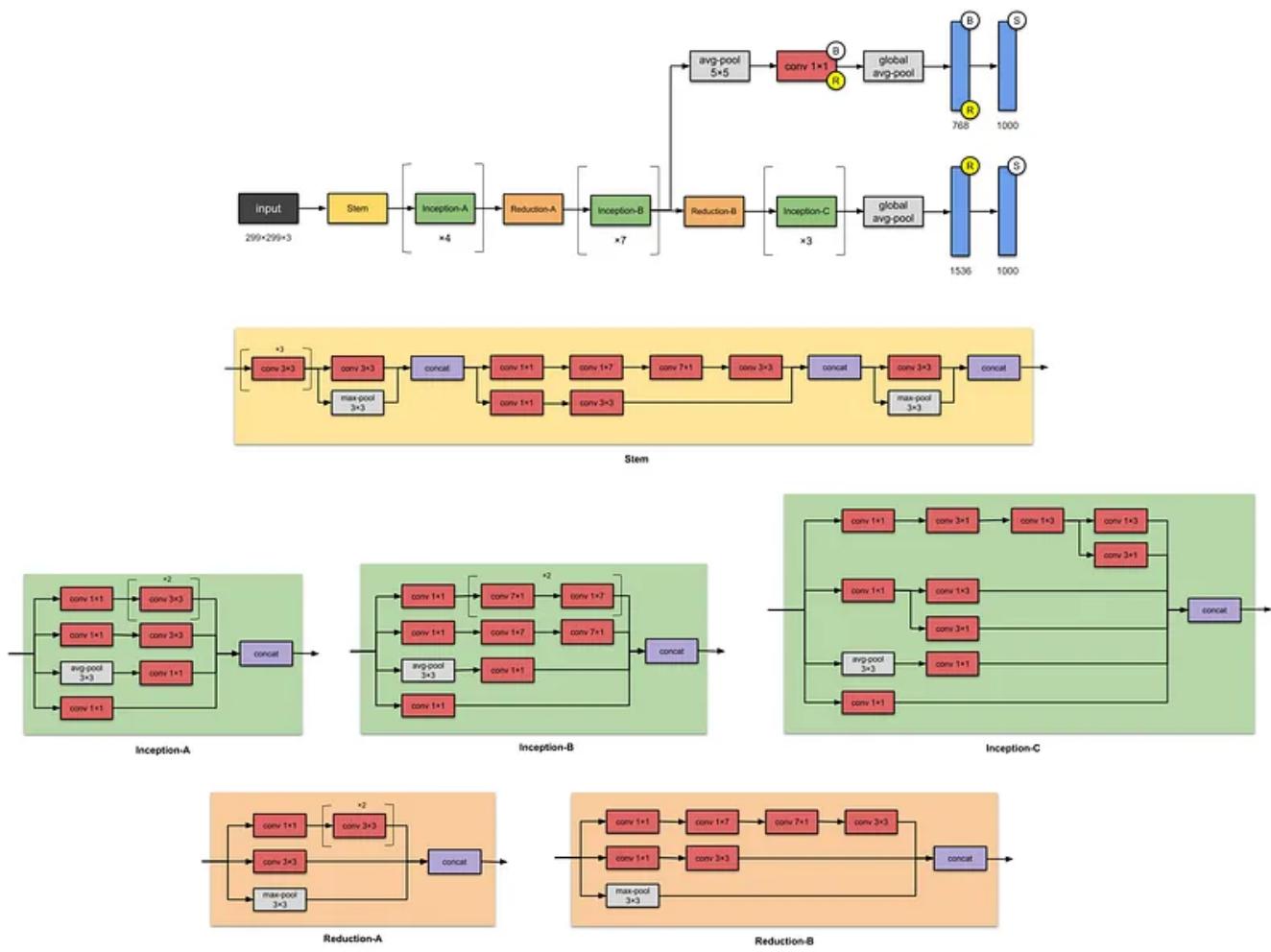


Fig. 8: Inception-v4 architecture. This CNN has an auxiliary network (which is discarded at inference time).
 *Note: All convolutional layers are followed by batch norm and ReLU activation. Architecture is based on their [GitHub code](#).

The folks from Google strike again with Inception-v4, 43M parameters. Again, this is an improvement from Inception-v3. The main difference is the Stem group and some minor changes in the Inception-C module. The authors also “made uniform choices for the Inception blocks for each grid size.” They also mentioned that having “residual connections leads to dramatically improved training speed.”

All in all, note that it was mentioned that Inception-v4 works better because of the increased model size.

💡 What's improved from the previous version, Inception-v3?

1. Change in Stem module.
2. Adding more Inception modules.

3. Uniform choices of Inception-v3 modules, meaning using the same number of filters for every module.

Publication

- Paper: [Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning](#)
- Authors: Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi. Google.
- Published in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence

9. Inception-ResNet-V2 (2016)

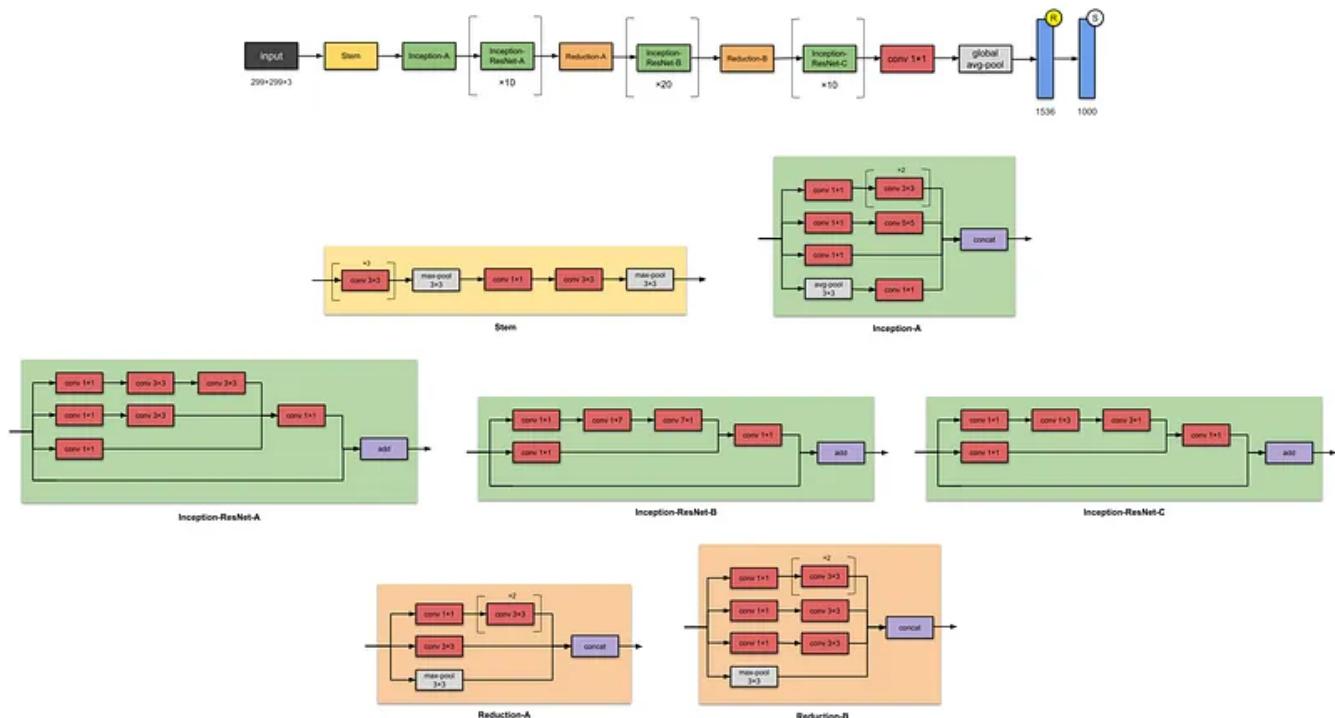


Fig. 9: Inception-ResNet-V2 architecture. *Note: All convolutional layers are followed by batch norm and ReLU activation. Architecture is based on their GitHub [code](#).

In the same paper as Inception-v4, the same authors also introduced Inception-ResNets — a family of Inception-ResNet-v1 and Inception-ResNet-v2. The latter member of the family has 56M parameters.

💡 What's improved from the previous version, Inception-v3?

1. Converting Inception modules to *Residual Inception blocks*.
2. Adding more Inception modules.
3. Adding a new type of Inception module (Inception-A) after the Stem module.

Publication

- Paper: [Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning](#)
- Authors: Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi. Google
- Published in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence

10. ResNeXt-50 (2017)

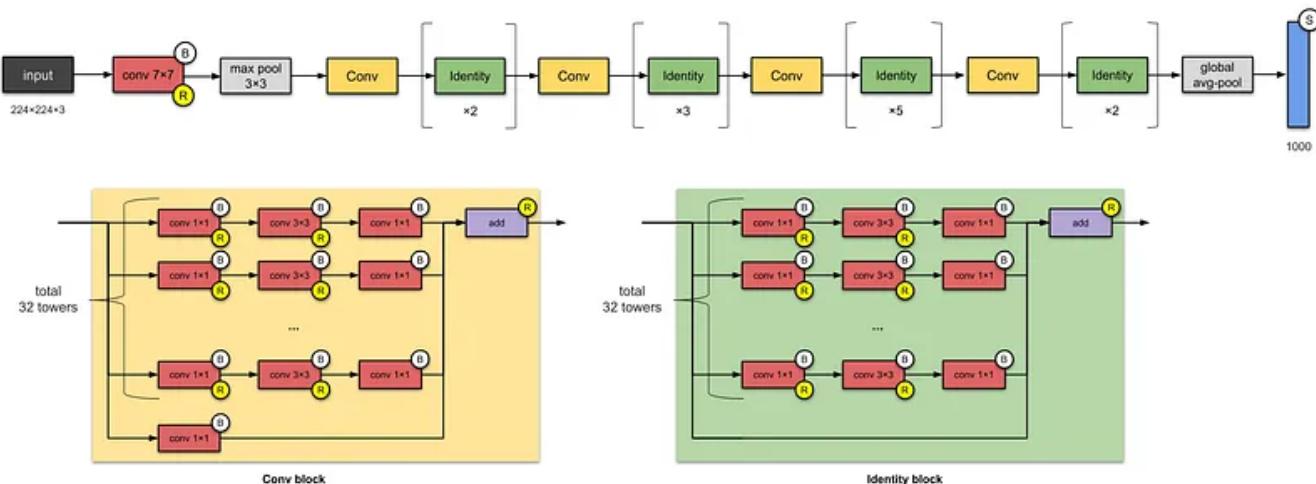


Fig. 10: ResNeXt architecture, based on their paper.

If you're thinking about ResNets, yes, they are related. ResNeXt-50 has 25M parameters (ResNet-50 has 25.5M). What's different about ResNeXts is the adding of parallel towers/branches/paths within each module, as seen above indicated by 'total 32 towers.'

 **What's novel?**

1. Scaling up the number of parallel towers (“cardinality”) within a module (well I mean this has already been explored by the Inception network, except that these towers are added here)

Publication

- Paper: [Aggregated Residual Transformations for Deep Neural Networks](#)
- Authors: Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He.
University of California San Diego, Facebook Research
- Published in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Appendix: Network In Network (2014)

Recall that in a convolution, the value of a pixel is a linear combination of the weights in a filter and the current sliding window. The authors proposed that instead of this linear combination, let's have a mini neural network with 1 hidden layer. This is what they coined as Mlpconv. So what we're dealing with here is a (simple 1 hidden layer) network in a (convolutional neural) network.

This idea of Mlpconv is likened to 1×1 convolutions and later became the main feature for Inception architectures.

What's novel?

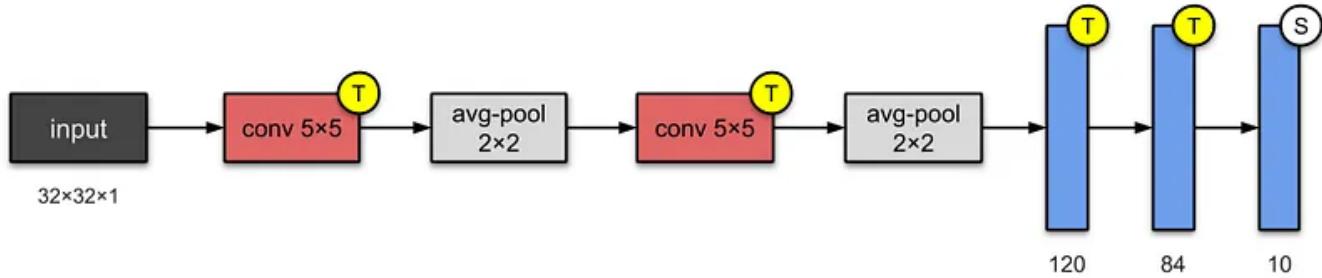
1. MLP convolutional layers, 1×1 convolutions
2. Global average pooling (taking the average of each feature map, and feeding the resulting vector into the softmax layer)

Publication

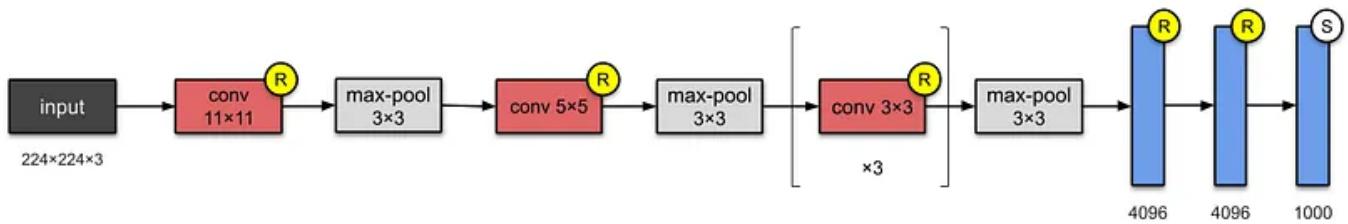
- Paper: [Network In Network](#)
- Authors: Min Lin, Qiang Chen, Shuicheng Yan. National University of Singapore
- arXiv preprint, 2013

Let's show them here again for easy reference:

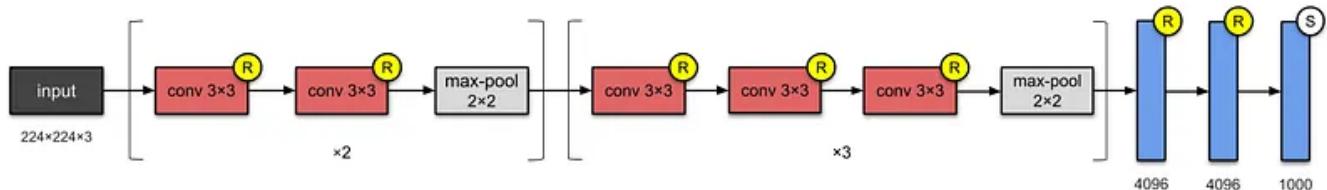
LeNet-5



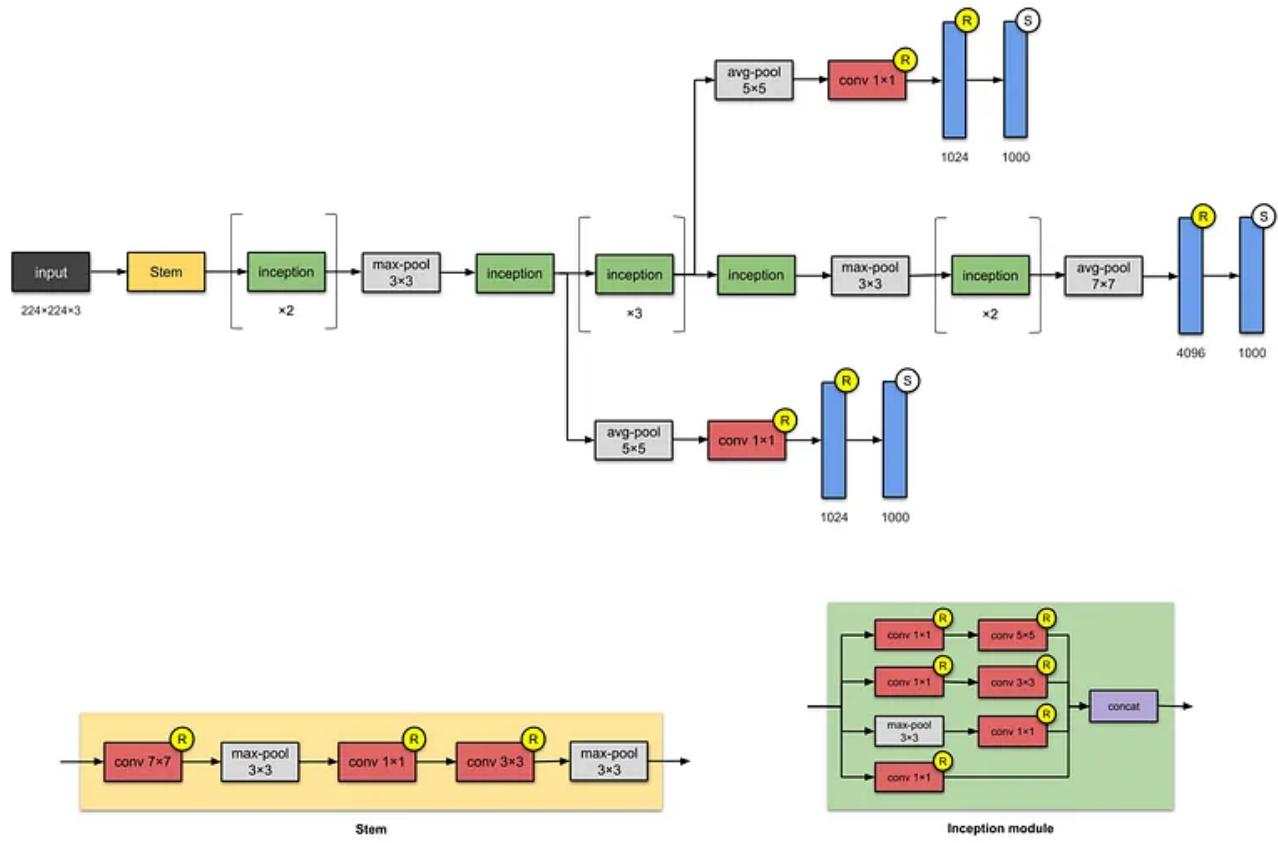
AlexNet



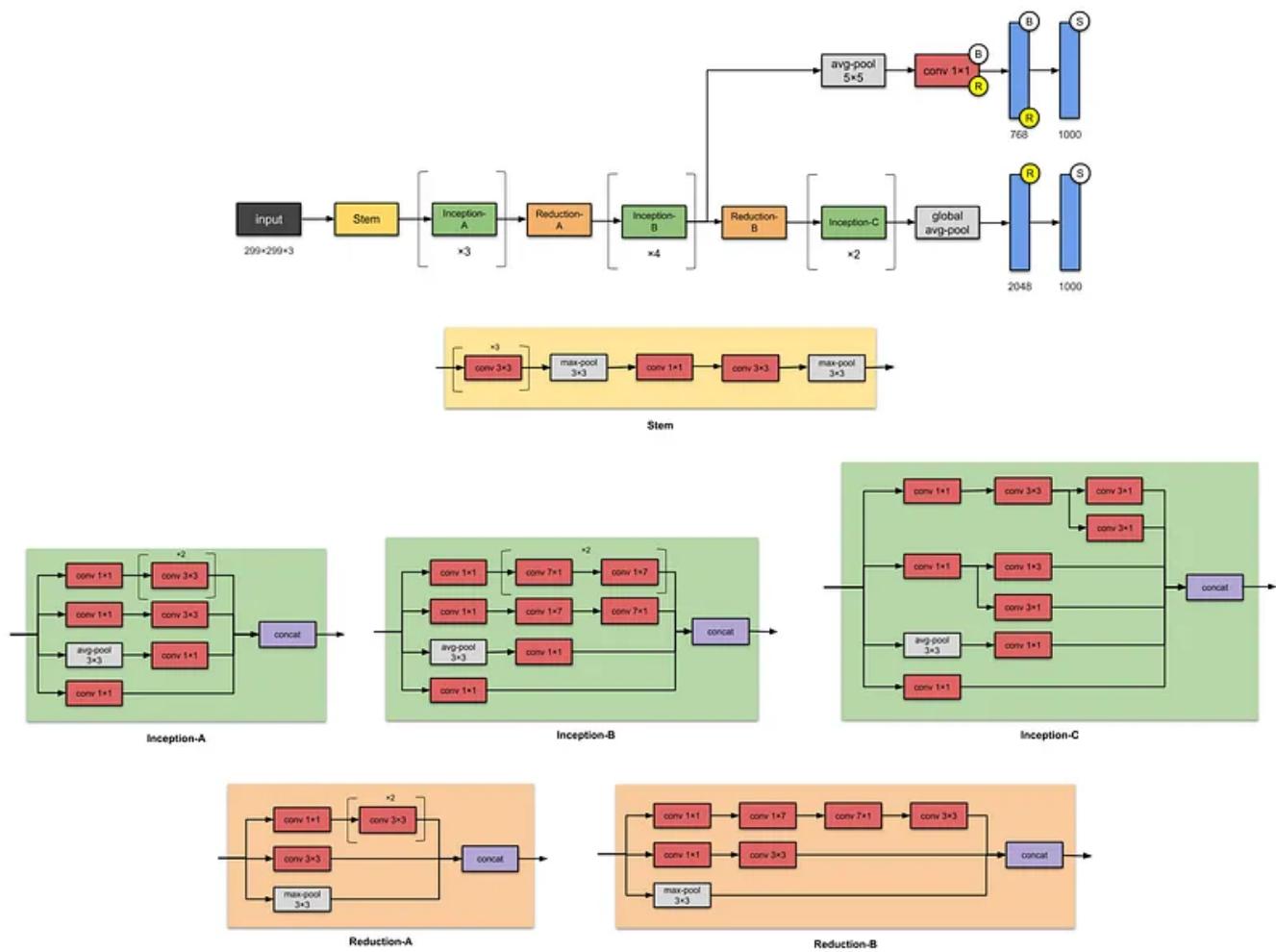
VGG-16



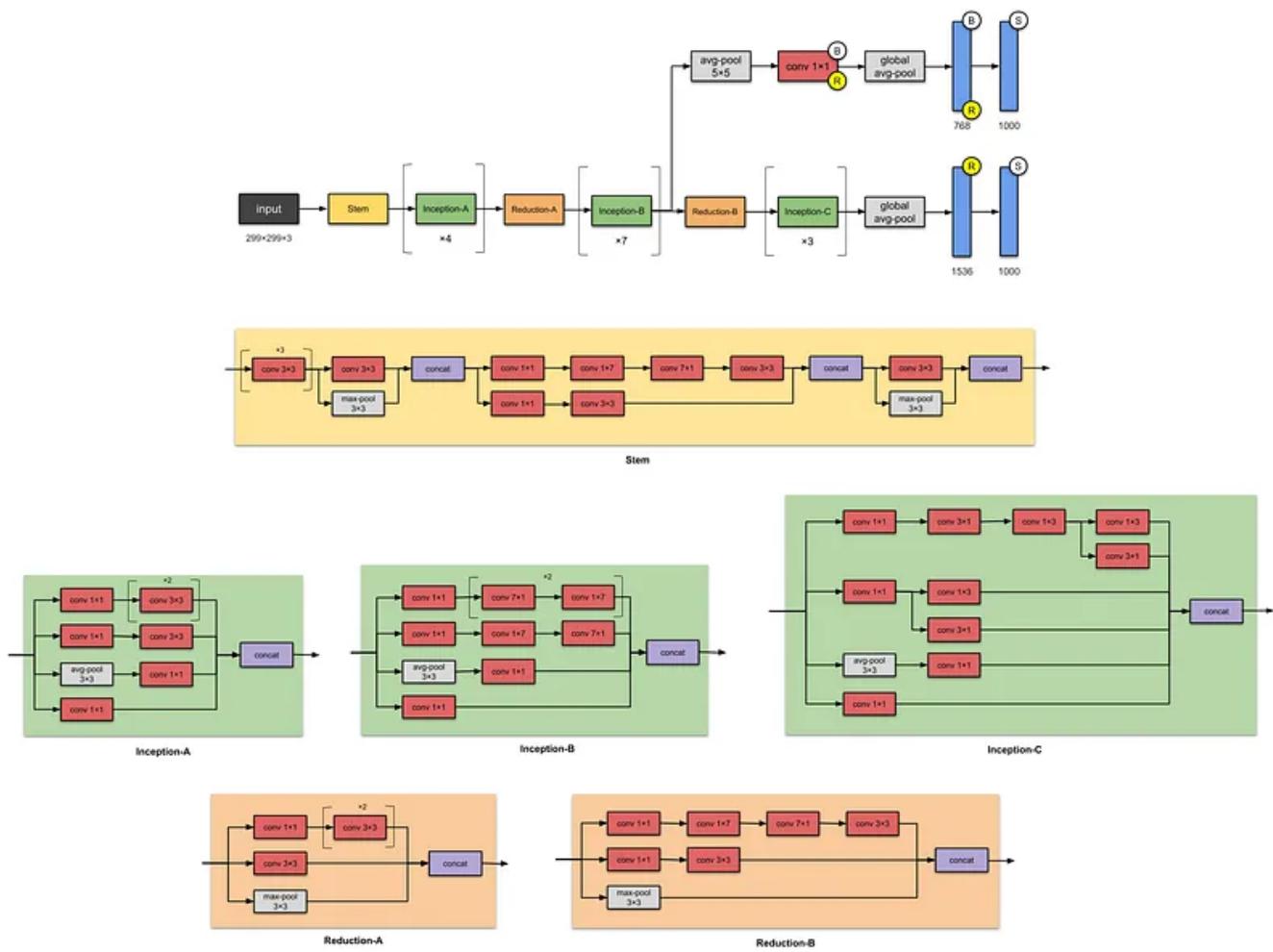
Inception-v1



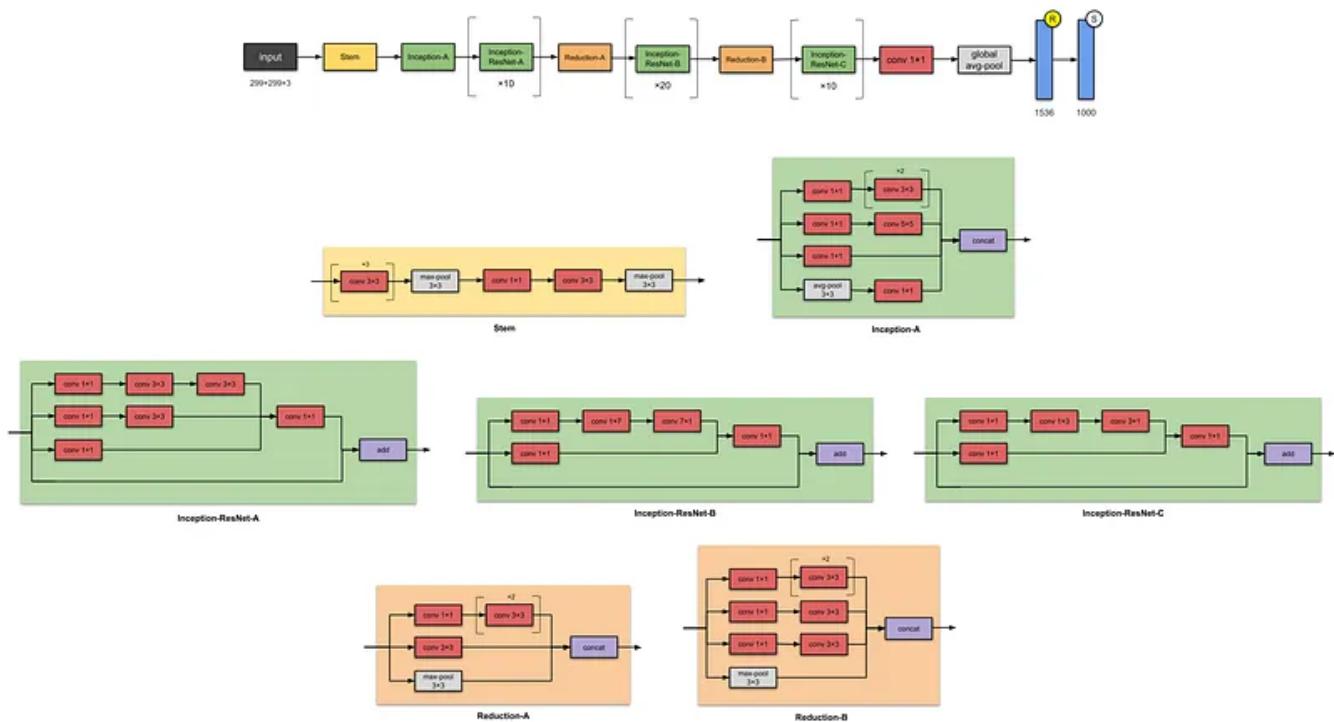
Inception-v3



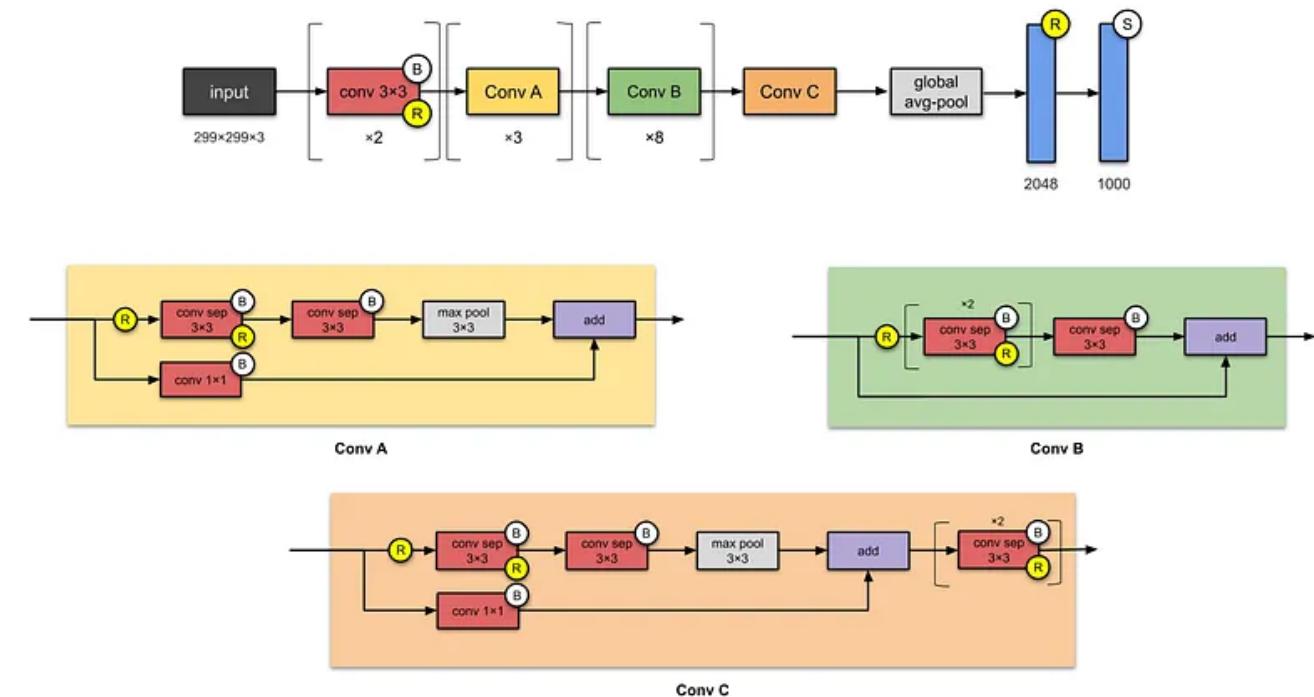
Inception-v4



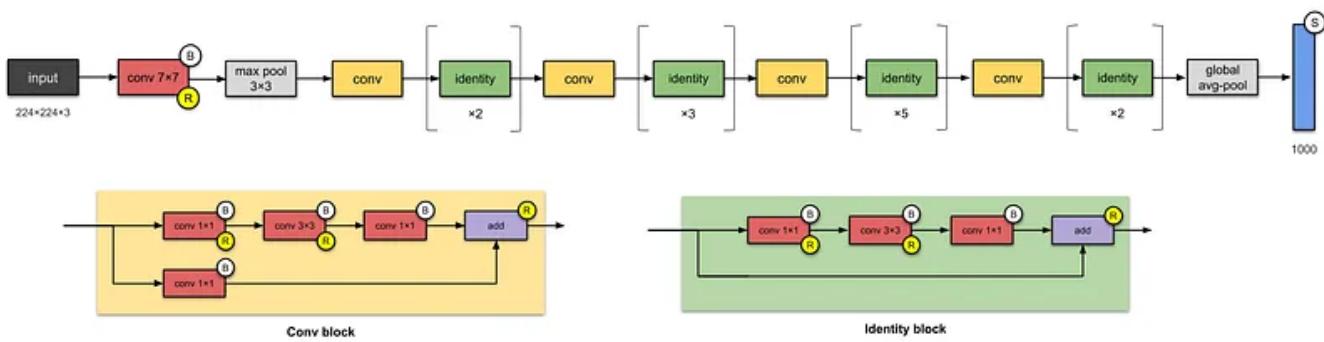
Inception-ResNet-V2



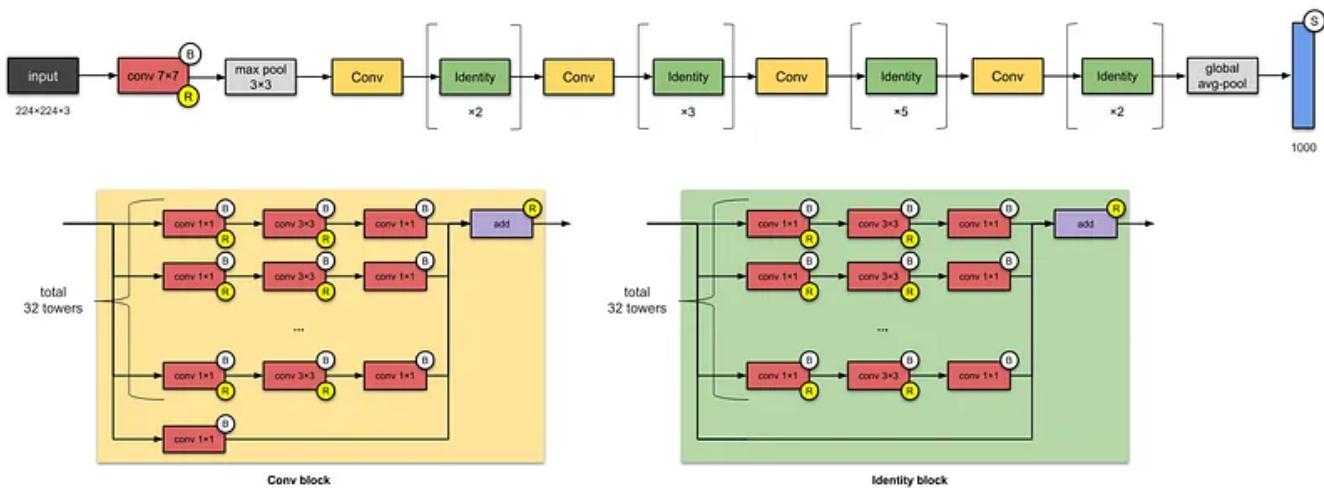
Xception



ResNet-50



ResNeXt-50



Resources for neural network visualisation

Here are some resources for you to visualise your neural network:

- [Netron](#) 😊
- [TensorBoard API](#) by TensorFlow
- [plot_model API](#) by Keras
- [pytorchviz package](#)

Similar articles

[CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more](#)

A Simple Guide to the Versions of the Inception Network

References

I have used the above-mentioned papers that produced the architectures for reference. On top of these, here are some others I used for this article:

<https://github.com/tensorflow/models/tree/master/research/slim/nets>
(github.com/tensorflow)

[Implementation of deep learning models from the Keras team](https://github.com/keras-team/keras) (github.com/keras-team)

[Lecture Notes on Convolutional Neural Network Architectures: from LeNet to ResNet](http://slazebni.cs.illinois.edu) (slazebni.cs.illinois.edu)

[Review: NIN — Network In Network \(Image Classification\)](#) (towardsdatascience.com)

Is there any error that you noticed in the visualisation? Is there anything else that you think I should've included? Drop me a comment below!

If you like my content and haven't already subscribed to Medium, subscribe via my referral link [here!](#) NOTE: A portion of your membership fees will be apportioned to me as referral fees.

Special thanks to Di Qiang, Wei Qi, Ren Jie, Fu Nan, Shirlene and Derek for reviewing this article.

Follow me on [Twitter](#) @remykarim or [LinkedIn](#). You may also reach out to me via raimi.bkarim@gmail.com. Feel free to visit my website at [remykarim.github.io](#).

Towards Data Science

Deep Learning

Machine Learning

Convolutional Network

Visualization

[Follow](#)

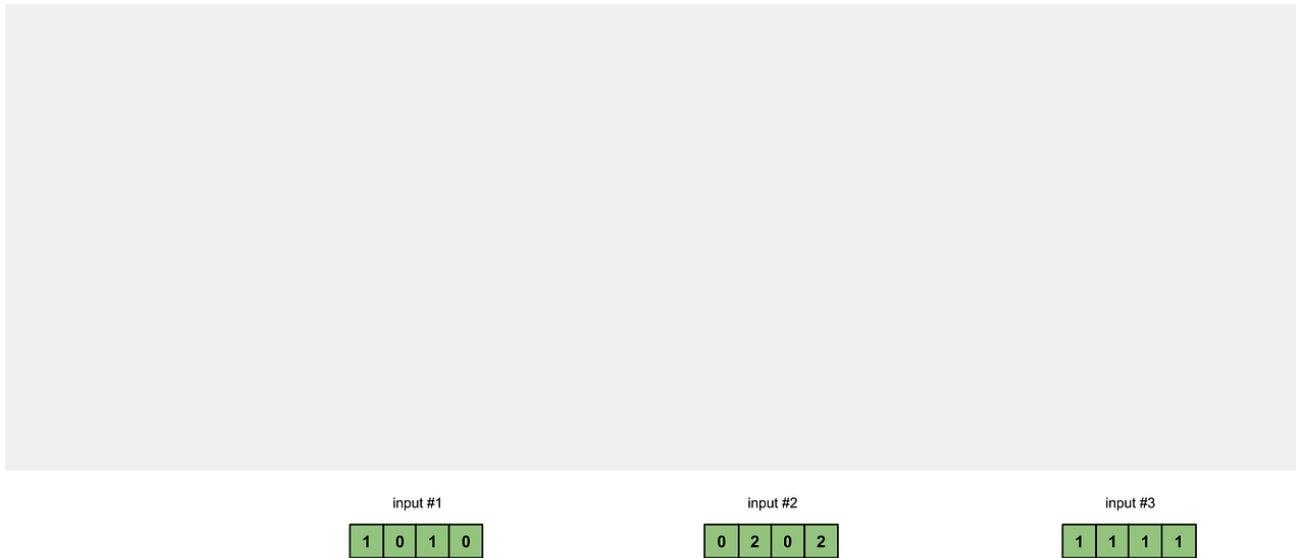
Written by Raimi Karim

2.2K Followers · Writer for Towards Data Science

sg Software Engineer at GovTech • Master of Computing AI at NUS • Subscribe at
remykarem.medium.com/membership

More from Raimi Karim and Towards Data Science

Self-attention



Raimi Karim in Towards Data Science

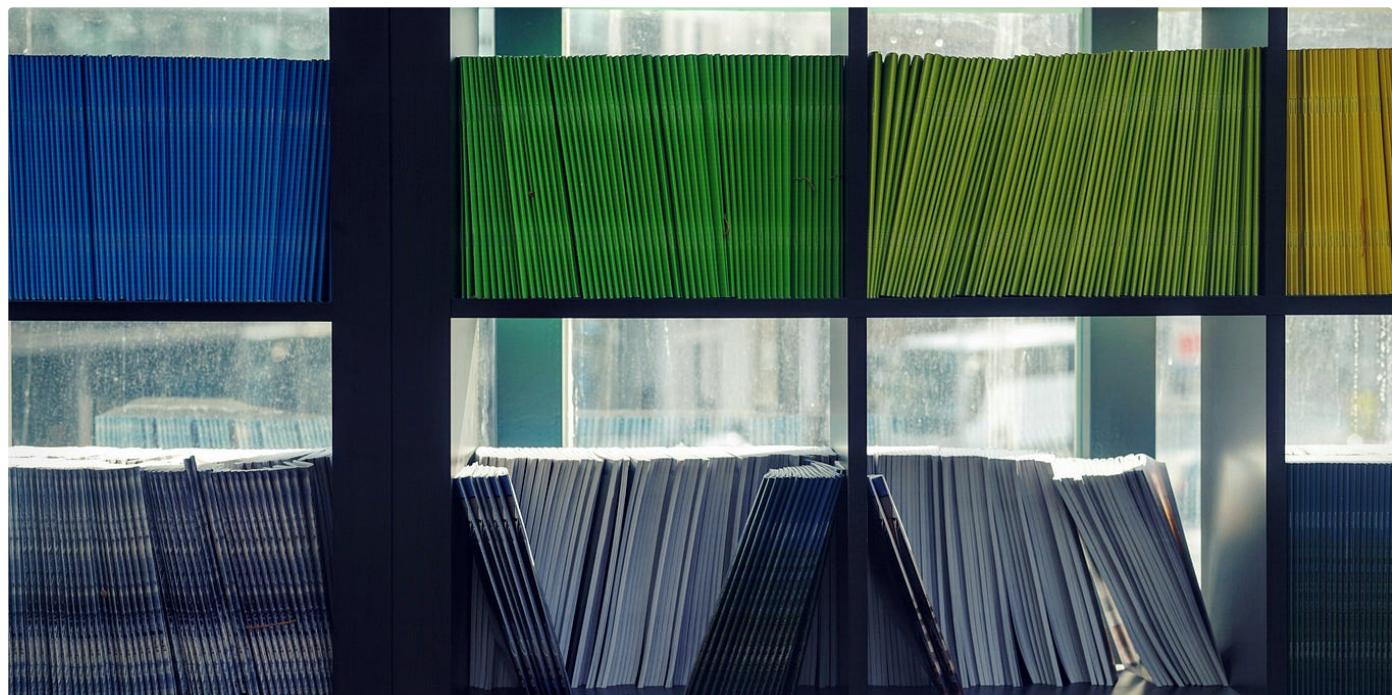
Illustrated: Self-Attention

A step-by-step guide to self-attention with illustrations and code

◆ · 9 min read · Nov 18, 2019

4.4K

28





Jacob Marks, Ph.D. in Towards Data Science

How I Turned My Company's Docs into a Searchable Database with OpenAI

And how you can do the same with your docs

15 min read · Apr 25

2.3K

30



Leonie Monigatti in Towards Data Science

Getting Started with LangChain: A Beginner's Guide to Building LLM-Powered Applications

A LangChain tutorial to build anything with large language models in Python

· 12 min read · Apr 25

1.5K

13





Raimi Karim in Towards Data Science

Intuitions on L1 and L2 Regularisation

How can they prevent overfitting?

8 min read · Dec 26, 2018



3.1K



27



See all from Raimi Karim

See all from Towards Data Science

Recommended from Medium



Alexander Nguyen in Level Up Coding

Why I Keep Failing Candidates During Google Interviews...

They don't meet the bar.

◆ · 4 min read · Apr 13

3.8K

120





Rukshan Pramoditha in Data Science 365

Determining the Right Batch Size for a Neural Network to Get Better and Faster Results

Guidelines for choosing the right batch size to maintain optimal training speed and accuracy while saving computer resources

◆ 4 min read · Sep 26, 2022

32



Lists



What is ChatGPT?

9 stories · 38 saves



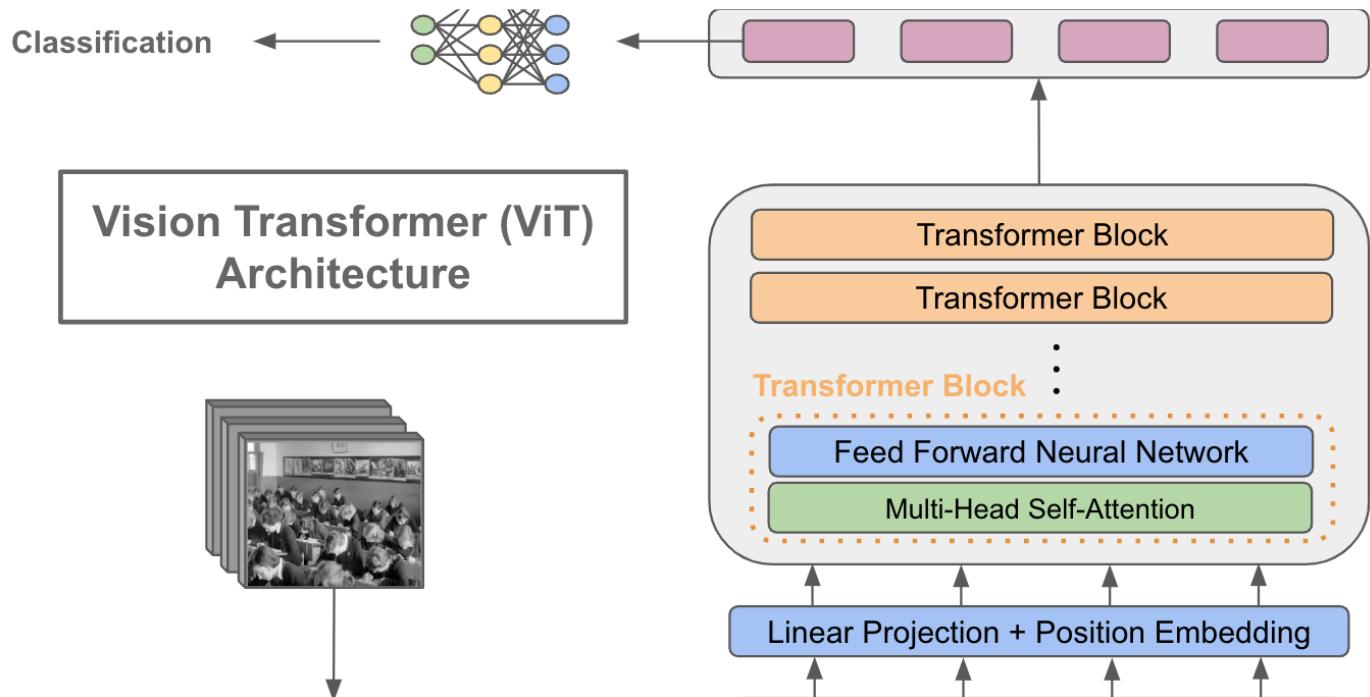
Work 101

26 stories · 4 saves



Staff Picks

307 stories · 72 saves





Cameron R. Wolfe in Towards Data Science

Using Transformers for Computer Vision

Are Vision Transformers actually useful?

★ · 13 min read · Oct 5, 2022



151



4



Matt Chapman in Towards Data Science

How I Stay Up to Date With the Latest AI Trends as a Full-Time Data Scientist

No, I don't just ask ChatGPT to tell me

★ · 8 min read · May 1

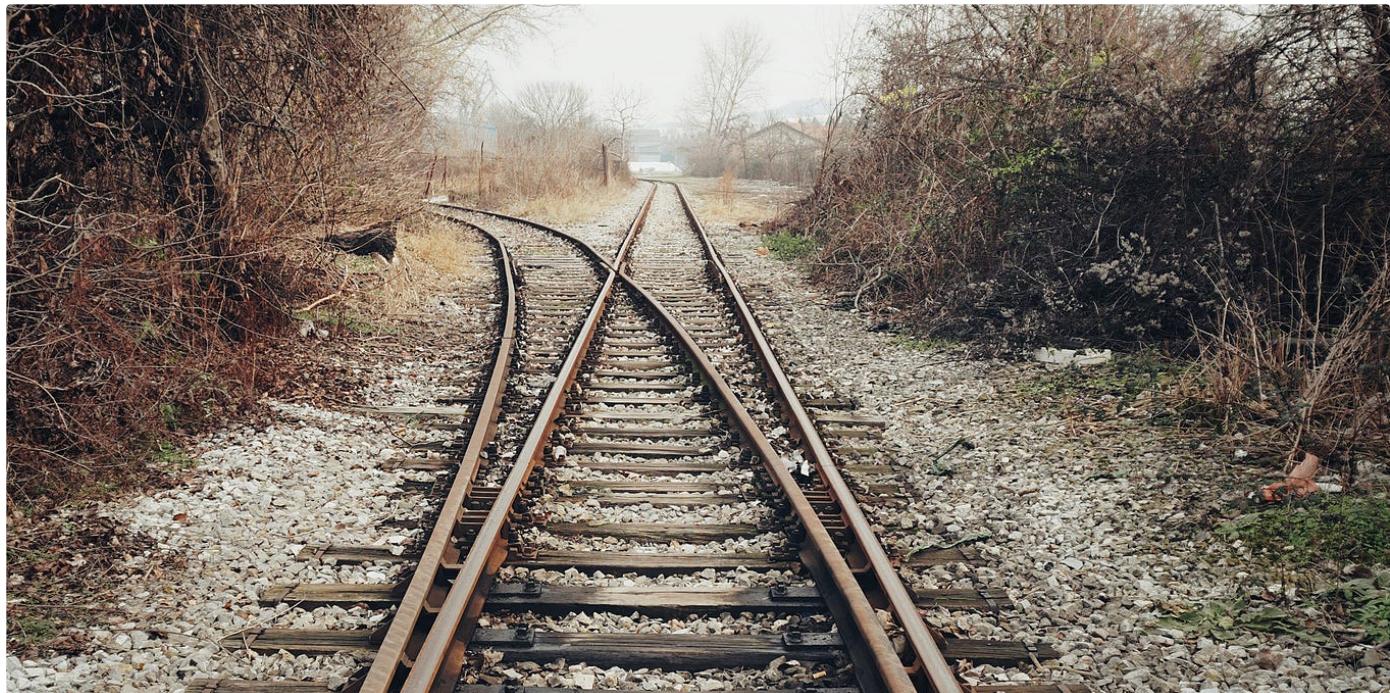


1K



21





Albers Uzila in Level Up Coding

Wanna Break into Data Science in 2023? Think Twice!

It won't be smooth sailing for you

★ · 11 min read · Dec 23, 2022

846

14





Martin Thissen in MLearning.ai

Understanding and Coding the Attention Mechanism—The Magic Behind Transformers

In this article, I'll give you an introduction to the attention mechanism and show you how to code the attention mechanism yourself.

◆ · 12 min read · Dec 6, 2022

270

2

[See more recommendations](#)