CNN Architectures – Structure Overview

Karel Horak
Brno University of Technology / Czech Technical University in Prague
horak@feec.vutbr.cz

Slides credit: K. Horak, R. Karim

Contents

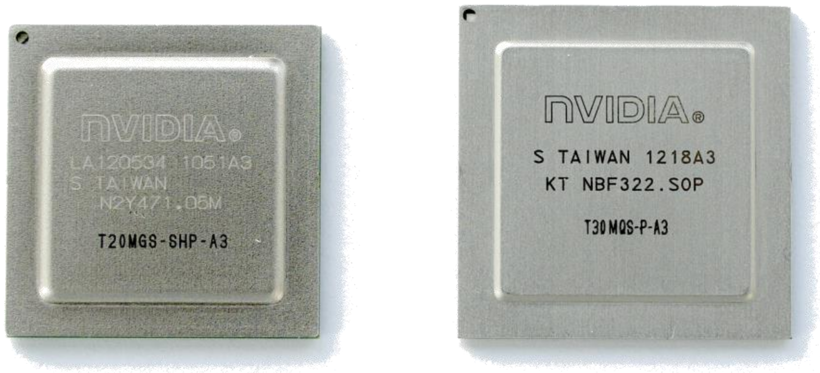
- 10 selected CNN Architectures – illustrated overview.

- The big bang of modern AI – deep learning:

  - Yann LeCun: work in convolutional neural nets (CNN)
  - Geoff Hinton: back-propagation and Stochastic Gradient Descent (SGD) approach to training
  - Andrew Ng: large-scale use of GPUs to accelerate deep neural networks (DNN)

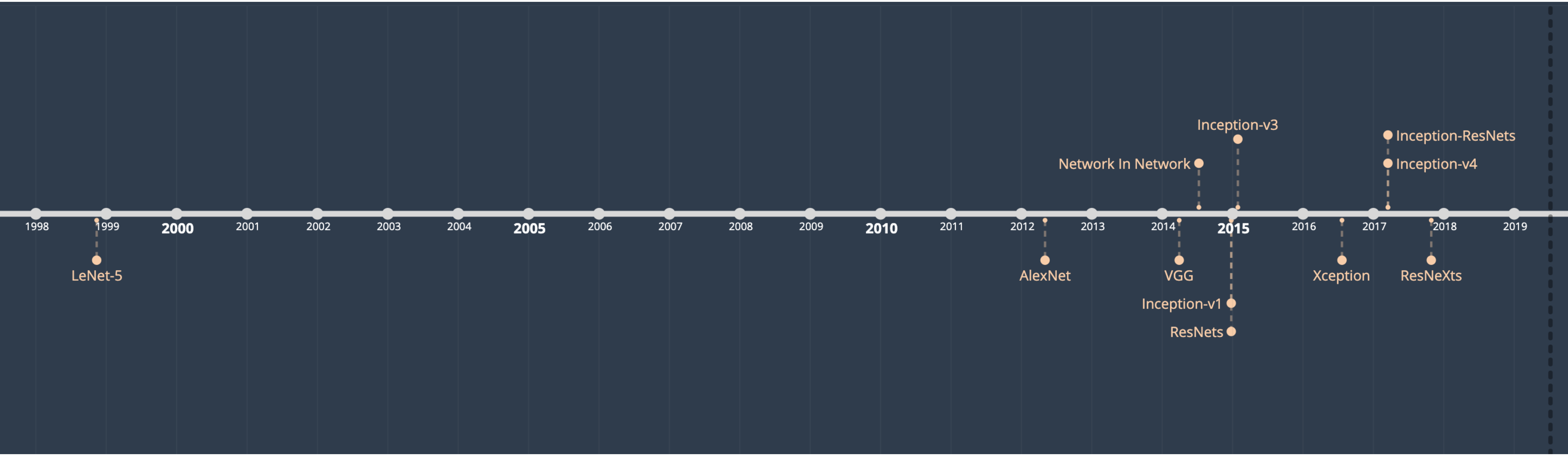
- Nvidia GPUs in AI:

  - 2009: Fermi architecture
  - 2012: Kepler architecture
  - 2015: Tegra X1 – dive into deep learning



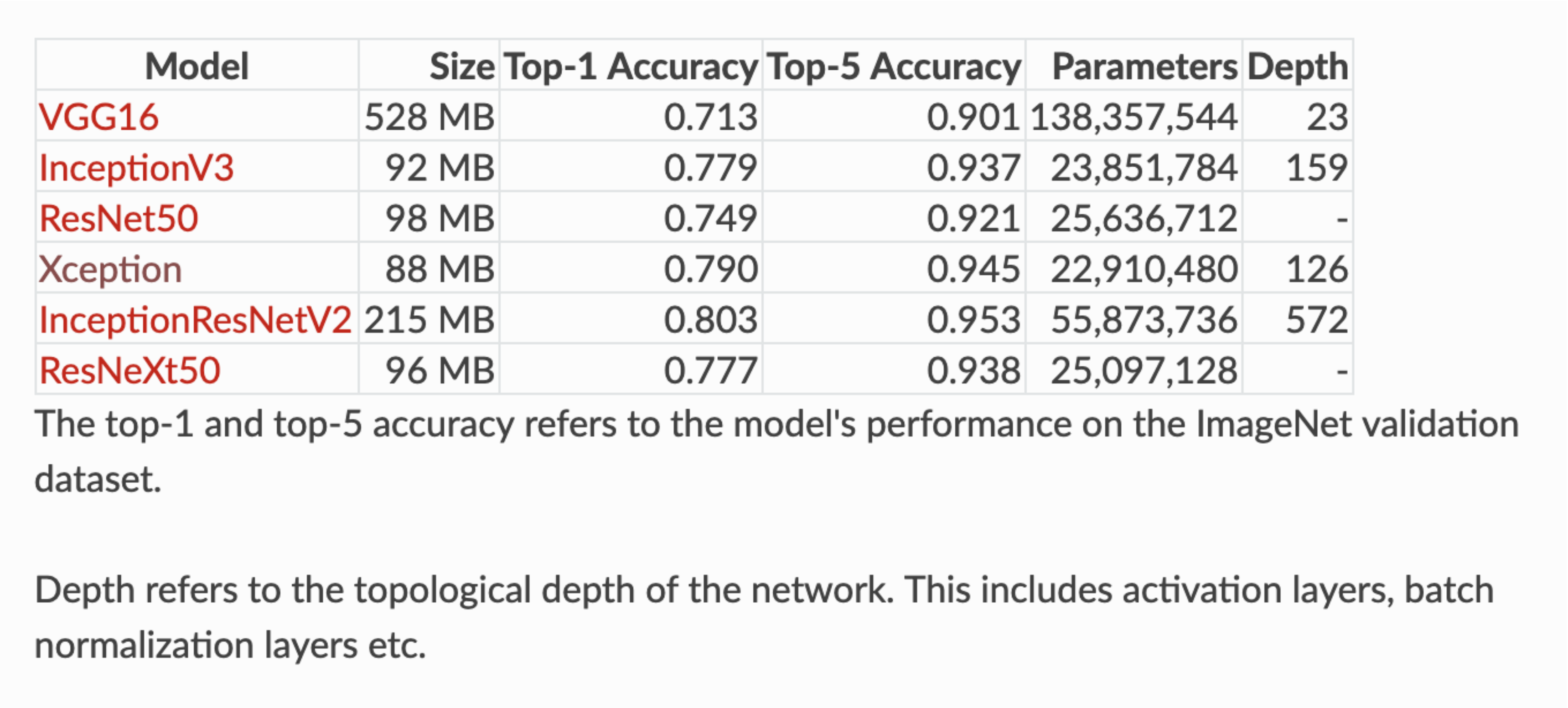
CNN Architectures – timeline

- 10 selected CNN Architectures – the year their papers were published.



CNN Architectures – Keras pre-trained

- Pre-trained weights which are available in Keras for several of the architectures:



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.

CNN Architectures – Legend

Layers

🟥 conv 3×3	Convolutional operations, in **red**
▨ avg-pool 2×2	Pooling operations, in **grey**
🟪 concat	Merge operations eg. concat, add in **purple**
🟦	Dense layer, **blue**

Activation Functions

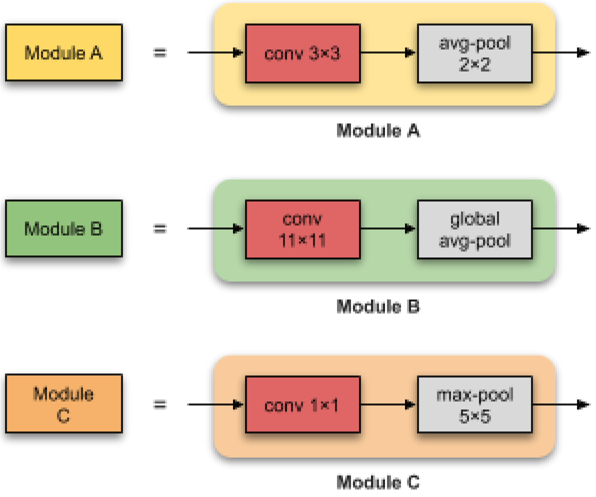
T	Tanh
R	ReLU

Other Functions

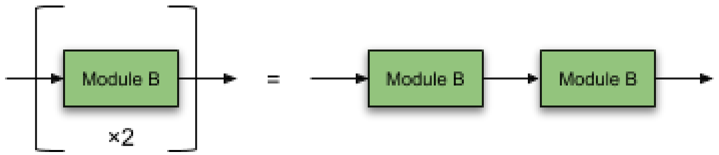
B	Batch normalisation
S	Softmax

Modules/Blocks

Modules (groups of convolutional, pooling and merge operations), in **yellow, green,** or **orange**.
The operations that make up these modules will also be shown.



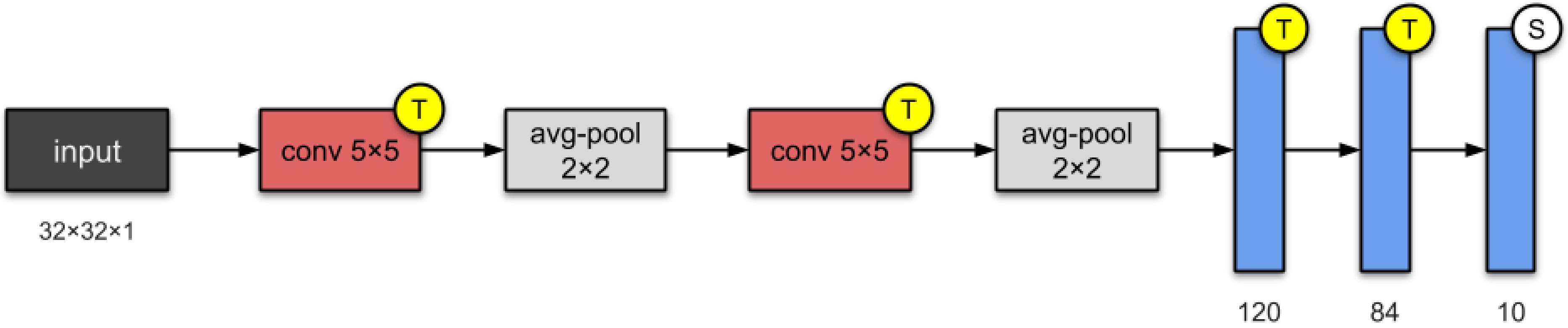
Repeated layers or modules/blocks



1. LeNet-5 (1998)

- 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.

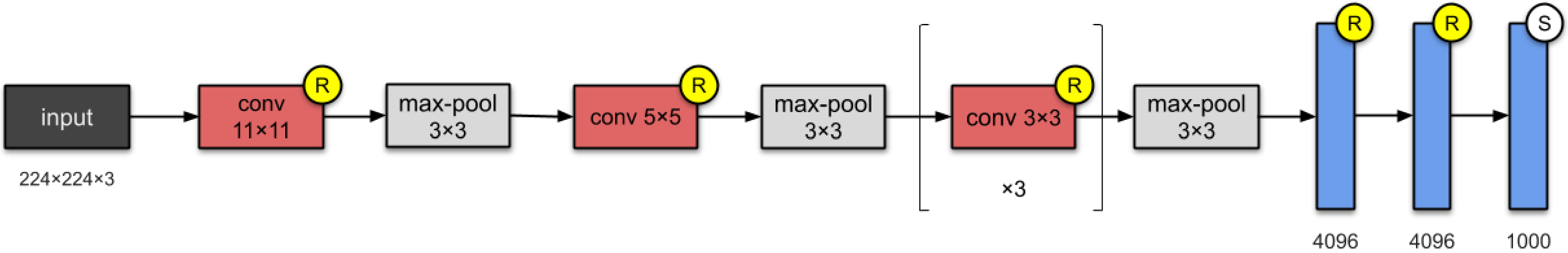
- Novel: this architecture has become the standard ‘template’: stacking convolutions with activation function, and pooling layers, and ending the network with one or more fully-connected layers.



2. AlexNet (2012)

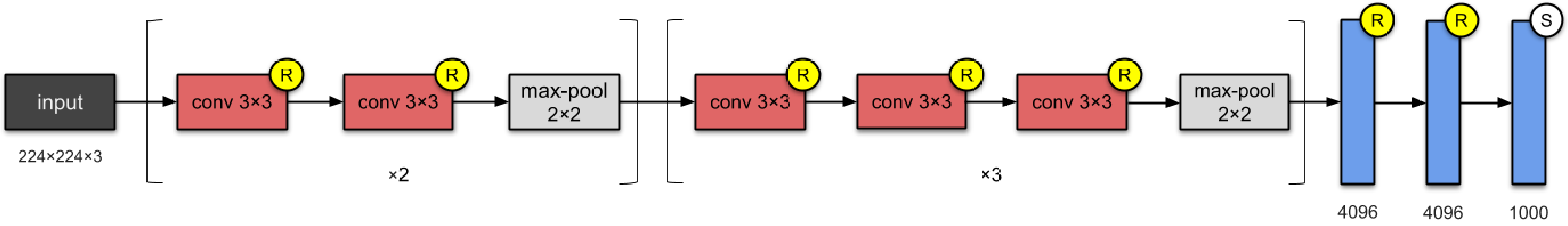
- With 60M parameters, AlexNet has 8 layers — 5 convolutional and 3 fully-connected. AlexNet just stacked a few more layers onto LeNet-5. At the point of publication, the authors pointed out that their architecture was "one of the largest convolutional neural networks to date on the subsets of ImageNet.".

- Novel:
  - They were the first to implement Rectified Linear Units (ReLUs) as activation functions.
  - Dropout.

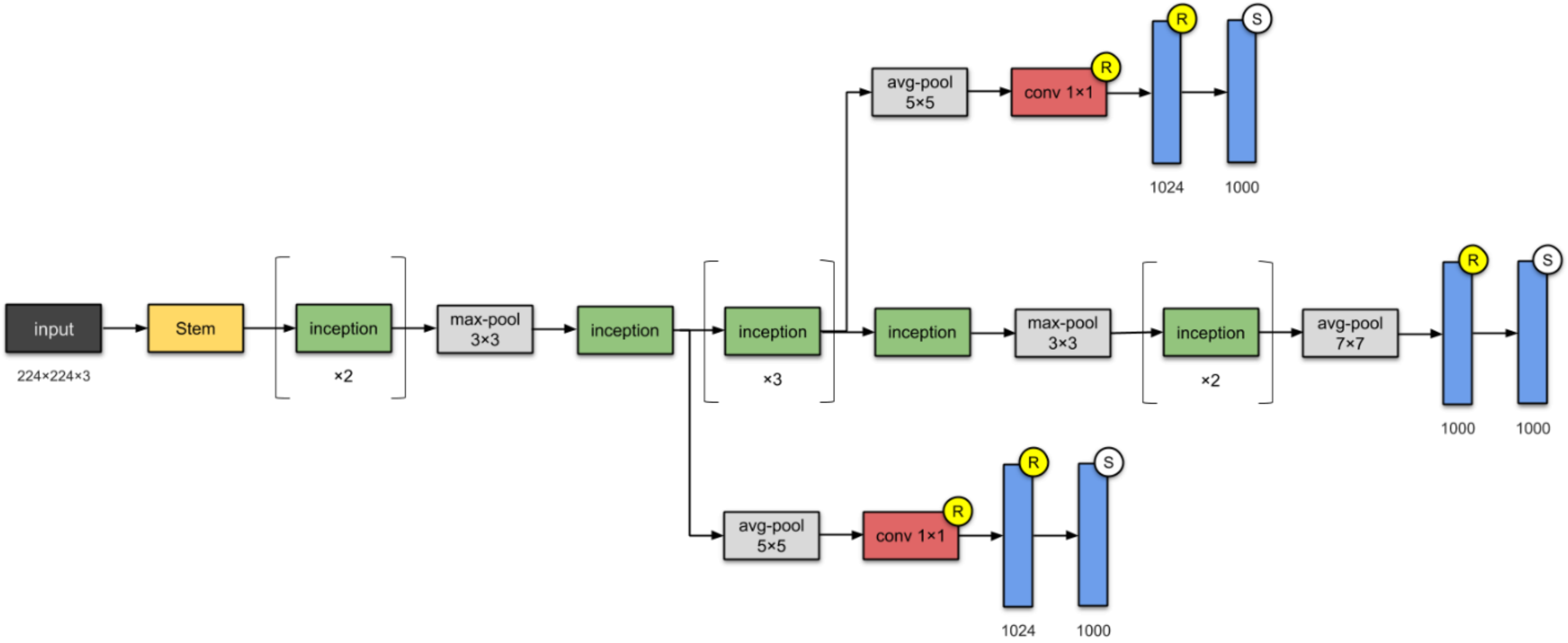


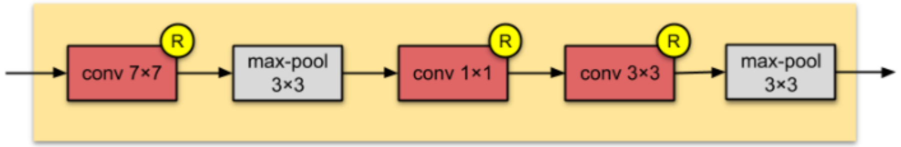
3. VGG-16 (2014)

- By now CNNs were starting to get deeper and deeper. This is because the most straightforward way of improving 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.

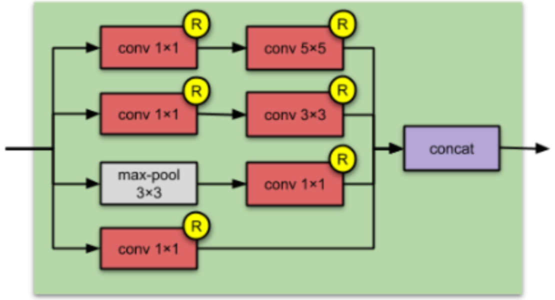


4. Inception-v1 (2014)





Stem

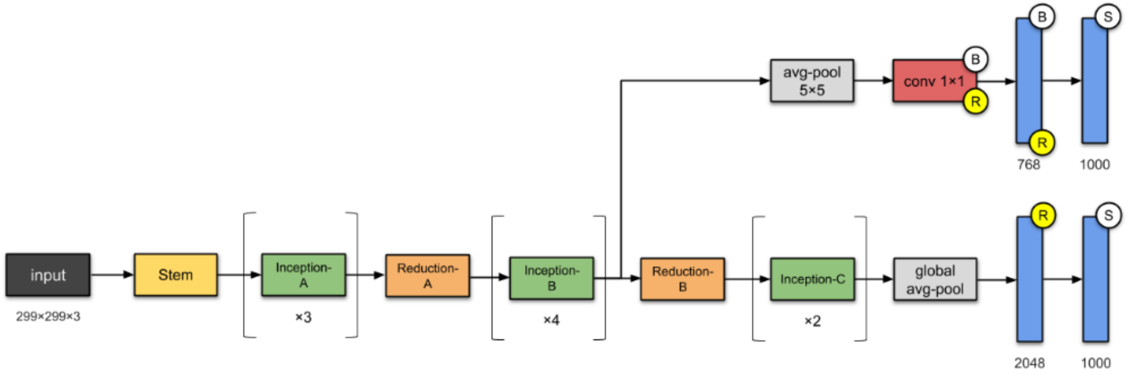


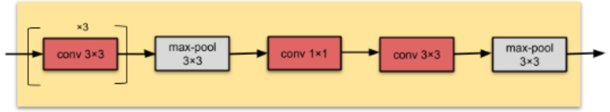
Inception module

4. Inception-v1 (2014)

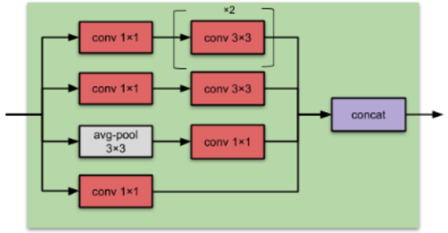
- 22-layer architecture with 5M parameters. Here, the Network In Network approach is heavily used, as mentioned in the paper. This is done by means of ‘Inception modules’. The design of the architecture of an Inception module is a product of research on approximating sparse structures. Each module presents 3 ideas:
  - parallel towers of convolutions with different filters, followed by concatenation, captures different features at 1×1, 3×3 and 5×5, 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.
  - 1×1 convolutions are used for dimensionality reduction to remove computational bottlenecks. Due to the activation function from 1×1 convolution, its addition also adds nonlinearity. This idea is based on the Network In Network paper.
  - 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.

5. Inception-v3 (2015)

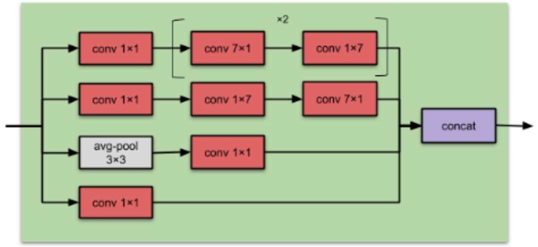




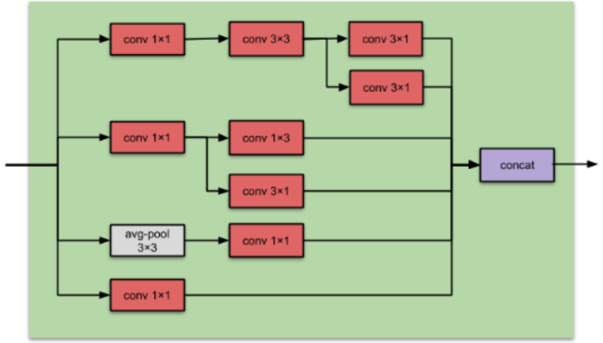
Stem



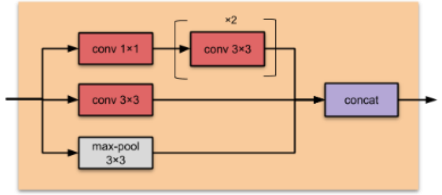
Inception-A



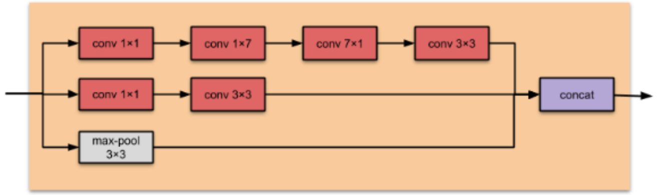
Inception-B



Inception-C



Reduction-A



Reduction-B

5. Inception-v3 (2015)

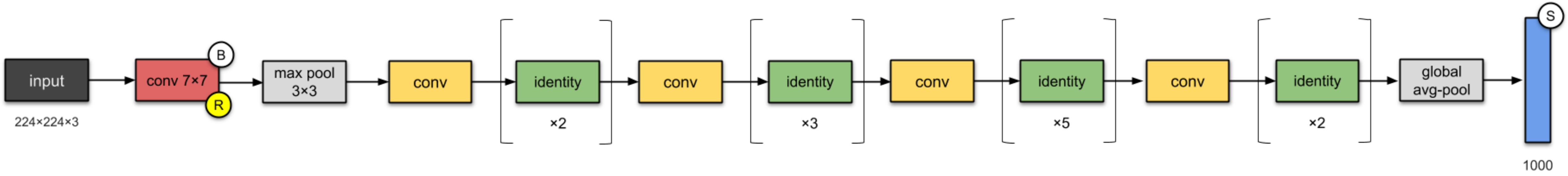
- Inception-v3 is a successor to Inception-v1, with 24M parameters. Note that Inception-v2 is an earlier prototype of v3 hence it’s very similar to v3 but not commonly used. When the authors came out with Inception-v2, they ran many experiments on it, and recorded some successful tweaks. Inception-v3 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.

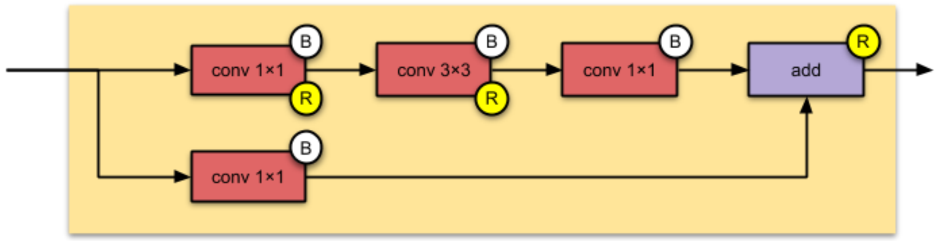
- Novel:
  - using batch normalisation

6. ResNet-50 (2015)

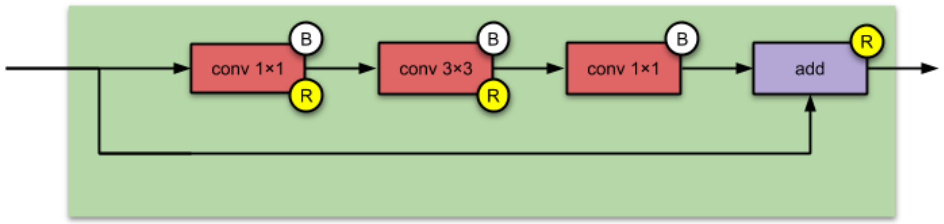
- From the past few CNNs, there can be seen nothing but an increasing number of layers in the design, and achieving better performance. But “with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly.” The folks from Microsoft Research addressed 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 below is ResNet-50, with 26M parameters.



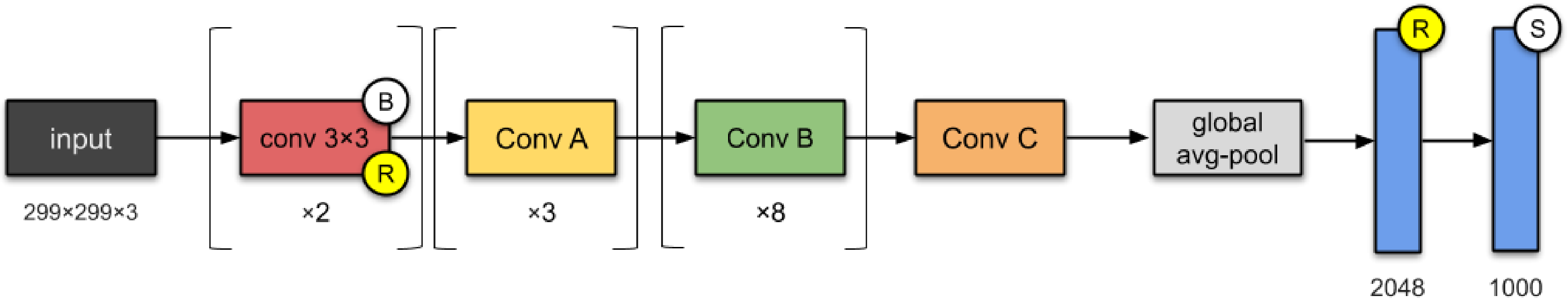


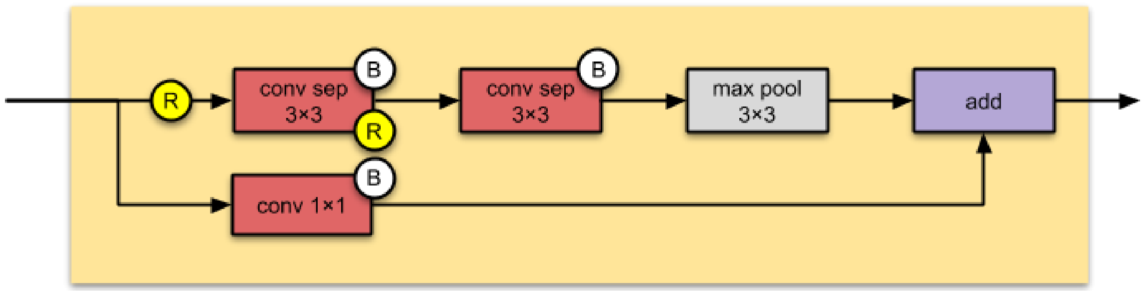
Conv block



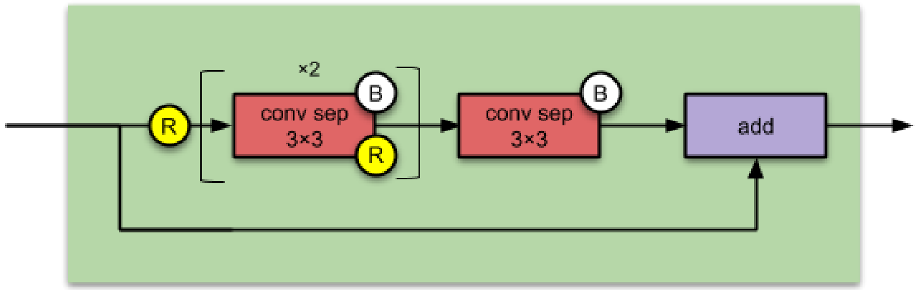
Identity block

7. Xception (2016)

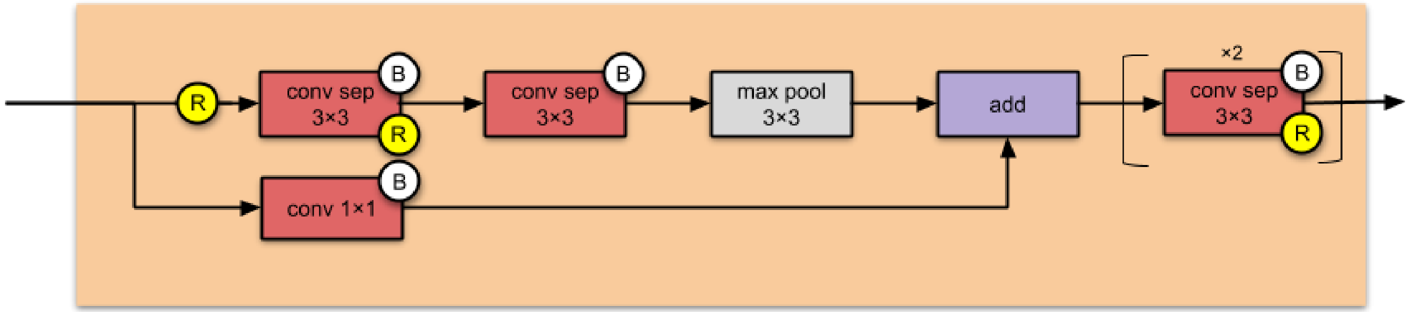




Conv A



Conv B



Conv C

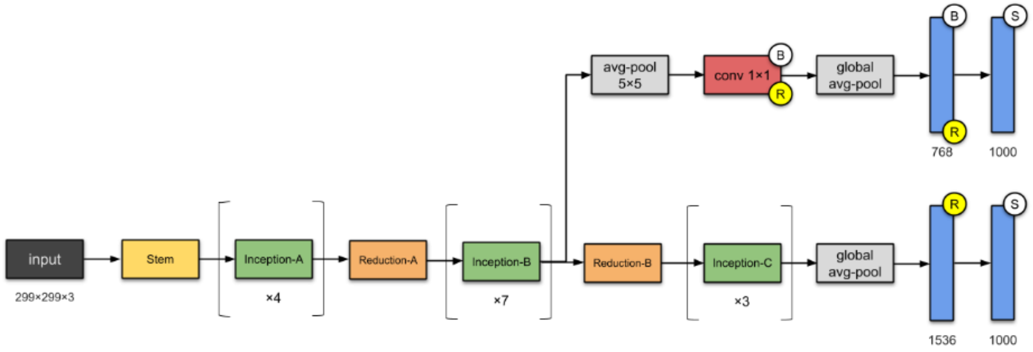
7. Xception (2016)

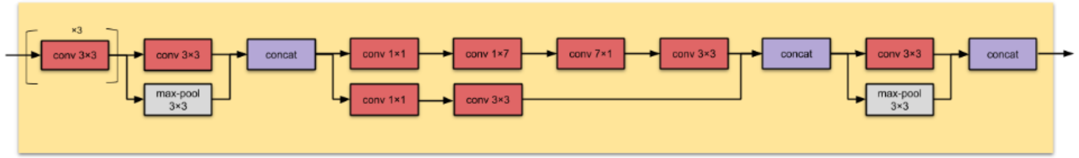
- Xception is an adaptation from Inception, where the Inception modules have been replaced with depthwise separable convolutions. It has also roughly the same number of parameters as Inception-v1 (23M).

- Novel:
    - cross-channel (or cross-feature map) correlations are captured by 1×1 convolutions
    - spatial correlations within each channel are captured via the regular 3×3 or 5×5 convolutions

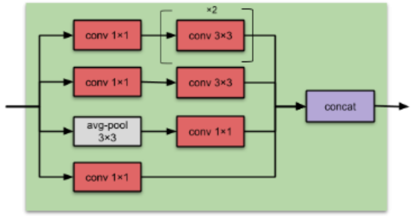
- Taking this idea to an extreme means performing 1×1 to every channel, then performing a 3×3 to each output. This is identical to replacing the Inception module with depthwise separable convolutions.

8. Inception-v4 (2016)

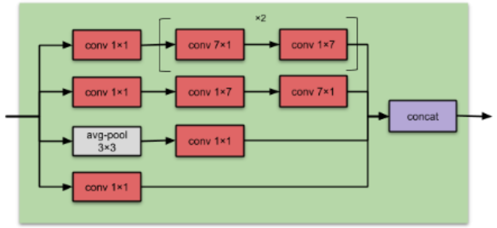




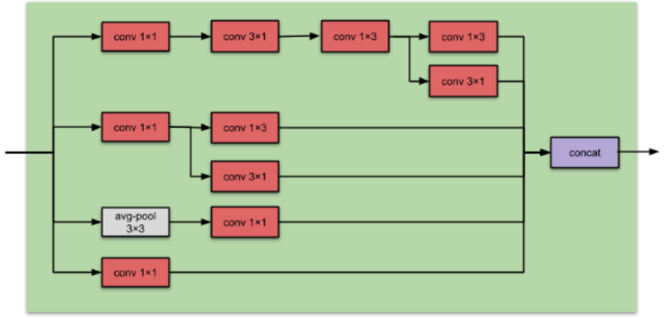
Stem



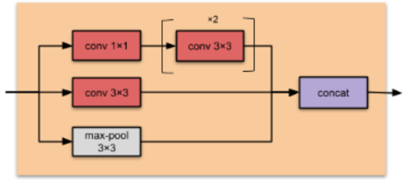
Inception-A



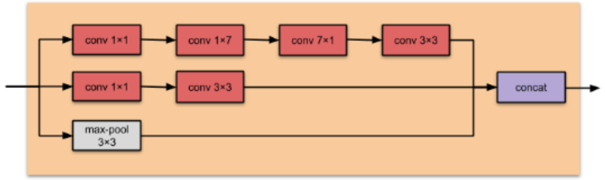
Inception-B



Inception-C



Reduction-A



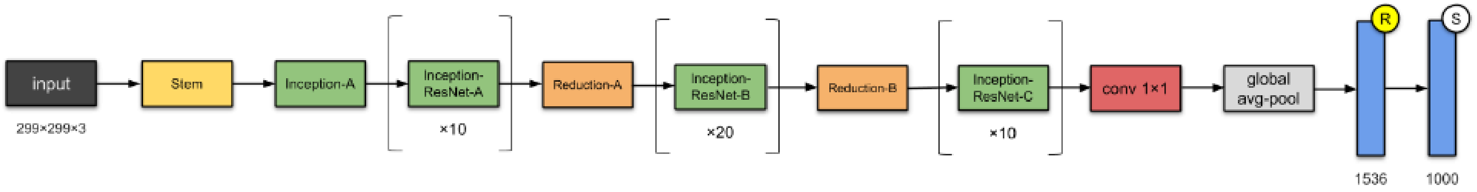
Reduction-B

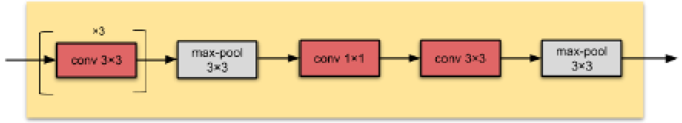
8. Inception-v4 (2016)

- Google “strike again” with Inception-v4, 43M parameters.

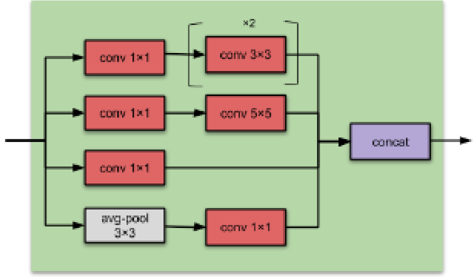
- It is an improvement from Inception-v3:
  - change in Stem module
  - adding more Inception modules
  - uniform choices of Inception-v3 modules, meaning using the same number of filters for every module

9. Inception-ResNet-V2 (2016)

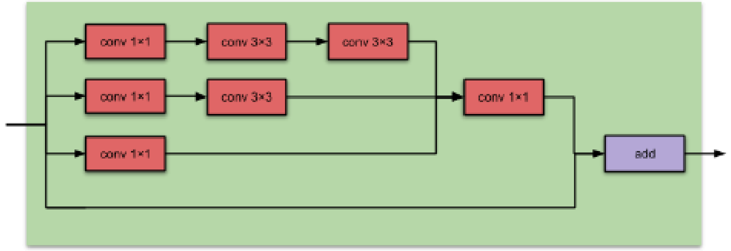




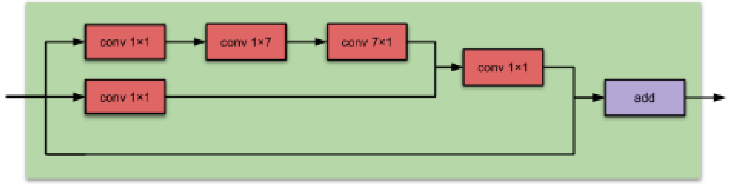
Stem



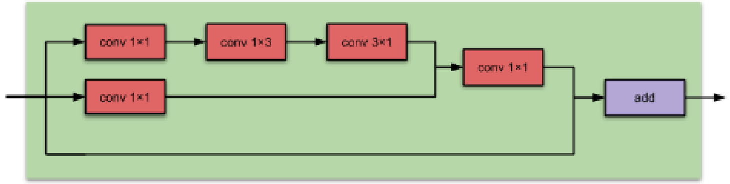
Inception-A



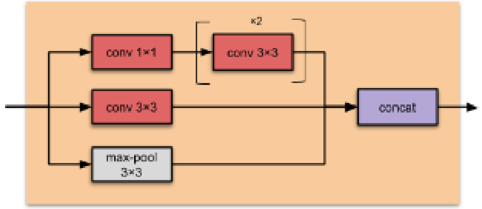
Inception-ResNet-A



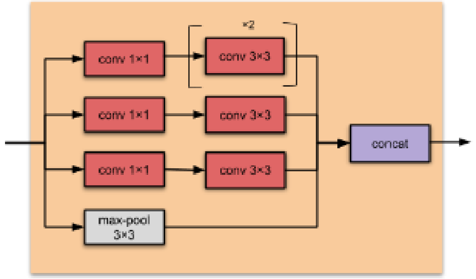
Inception-ResNet-B



Inception-ResNet-C



Reduction-A



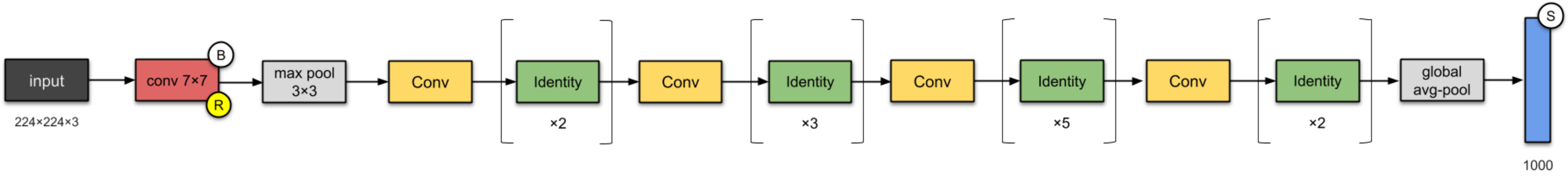
Reduction-B

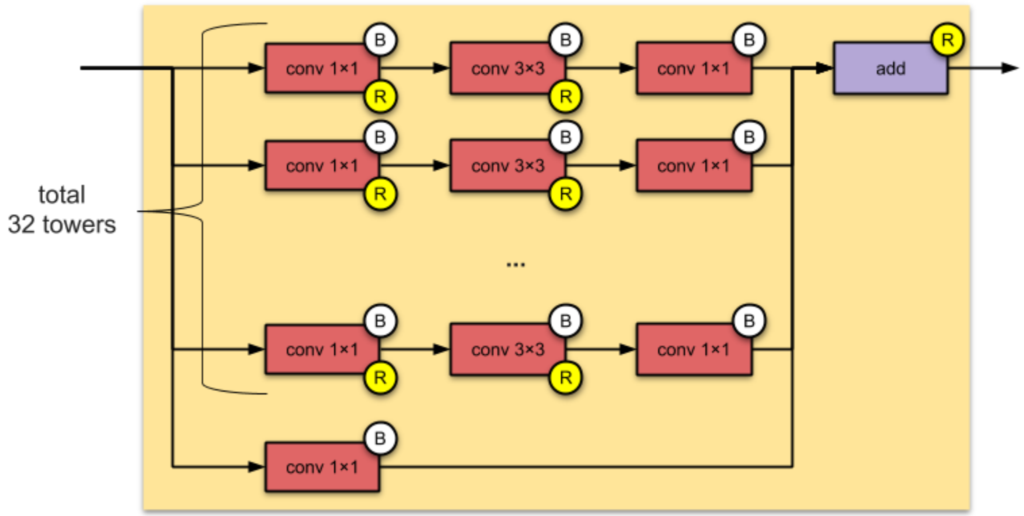
9. Inception-ResNet-V2 (2016)

- Family of Inception-ResNet-v1 and Inception-ResNet-v2. The latter member of the family has 56M parameters.

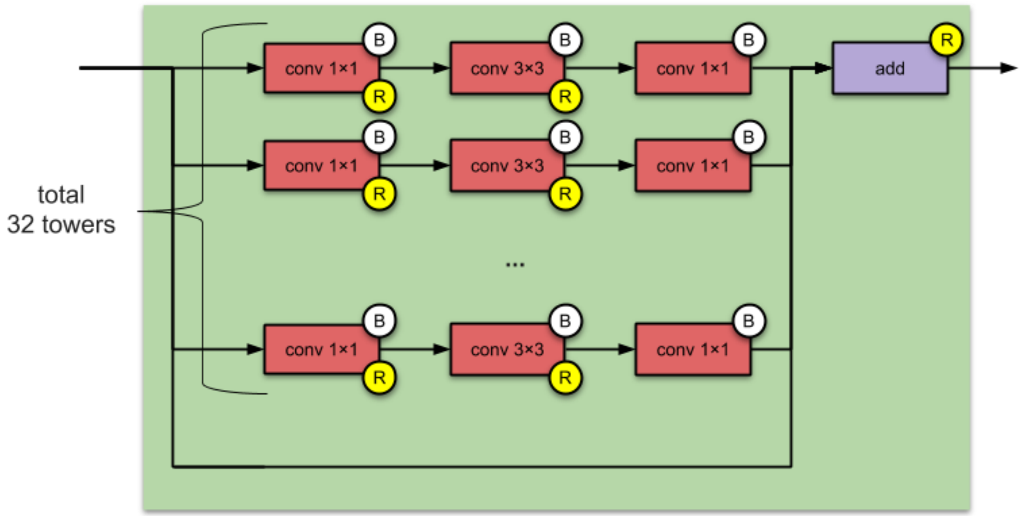
- What’s improved from the previous version, Inception-v3:
    - converting Inception modules to Residual Inception blocks
    - adding more Inception modules
    - adding a new type of Inception module (Inception-A) after the Stem module

10. ResNeXt-50 (2017)





Conv block



Identity block

10. ResNeXt-50 (2017)

- 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.'

- Novel:
  - 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)

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 became the main feature for Inception architectures.

- Novel:
    - MLP convolutional layers, 1×1 convolutions
    - global average pooling (taking average of each feature map, and feeding the resulting vector into the softmax layer)