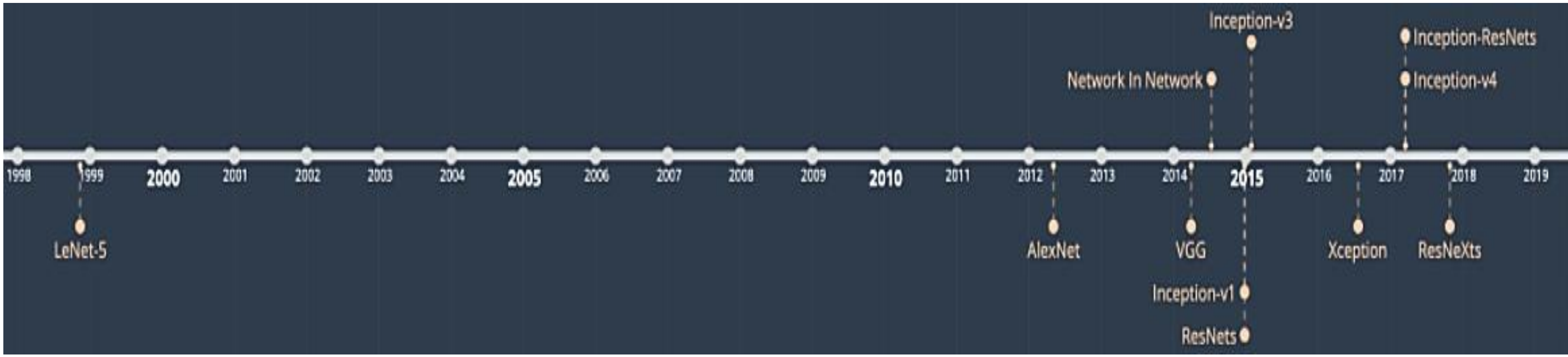
CNN Architectures

CNN models whose pre-trained weights are usually shared by deep learning libraries (such as TensorFlow, Keras and PyTorch) for users to use.

Some of these models have shown success in competitions like the [ImageNet Large Scale Visual Recognition Challenge](#) (ILSVRC).



Illustrated: 10 CNN Architectures. A compiled visualisation of the common convolutional neural networks, Raimi Karim, Jul 29, 2019, https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

… mainly due to the advances of deep learning, more concretely convolutional networks, the quality of image recognition and object detection has been progressing at a dramatic pace.

One encouraging news is that most 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.**

[Christian Szegedy et al, Going deeper with convolutions, arXiv:1409.4842v1 [cs.CV] 17 Sep 2014, https://arxiv.org/pdf/1409.4842.pdf)]

Keras Applications

Keras Applications are deep learning models that are made available alongside pre-trained weights. These models can be used for **prediction, feature extraction, and fine-tuning**.

Weights are downloaded automatically when instantiating a model. They are stored at ~/.keras/models/.

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth

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

Using a CNN, we make a prediction and obtain the predicted class multinomial distribution (p_i).

In the case of **top-1** score, you check if the top class (the one having the highest probability) is the same as the target label.

In the case of **top-5** score, you check if the target label is one of your top 5 predictions (the 5 ones with the highest probabilities).

In both cases, the top score is computed as the number of times a predicted label matched the target label, divided by the number of data-points evaluated.

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

Usage examples for image classification models

Keras API reference / Keras Applications;
https://keras.io/api/applications/

Available models in keras, 1/2

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88

Keras API reference / Keras Applications;
https://keras.io/api/applications/

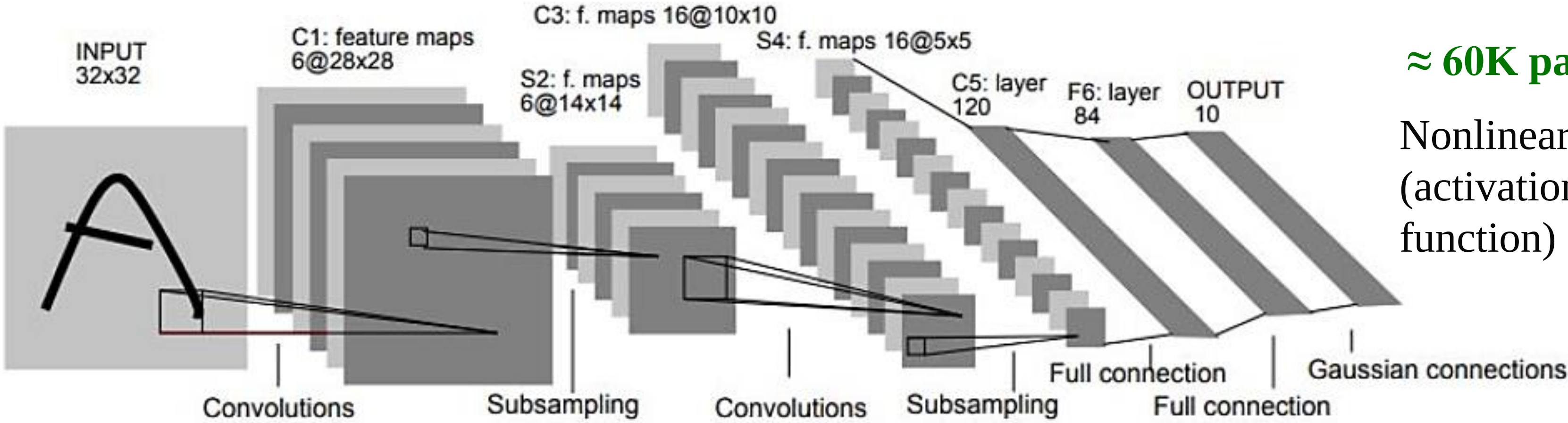
Available models in keras, 2/2

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-
EfficientNetB0	29 MB	-	-	5,330,571	-
EfficientNetB1	31 MB	-	-	7,856,239	-
EfficientNetB2	36 MB	-	-	9,177,569	-
EfficientNetB3	48 MB	-	-	12,320,535	-
EfficientNetB4	75 MB	-	-	19,466,823	-
EfficientNetB5	118 MB	-	-	30,562,527	-
EfficientNetB6	166 MB	-	-	43,265,143	-
EfficientNetB7	256 MB	-	-	66,658,687	-

Keras API reference / Keras Applications;
https://keras.io/api/applications/

LeNet-5 Architecture (1998)

Original Image published in [LeCun et al., 1998]



≈ 60K parameters

Nonlinearity (activation function) - **sigmoid**

Convolutions	Subsampling	Convolutions	Subsampling	Full connection	Full connection	Gaussian connections
5x5	average pooling	5x5	average pooling	FC	FC	softmax
s=1	2x2	s=1	2x2			
$n_c = 6$	s=2	$n_c = 16$	s=2			

The LeNet-5 architecture consists of two sets of convolutional and average pooling layers, followed by a flattening convolutional layer, then two fully-connected layers and finally a softmax classifier.

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.

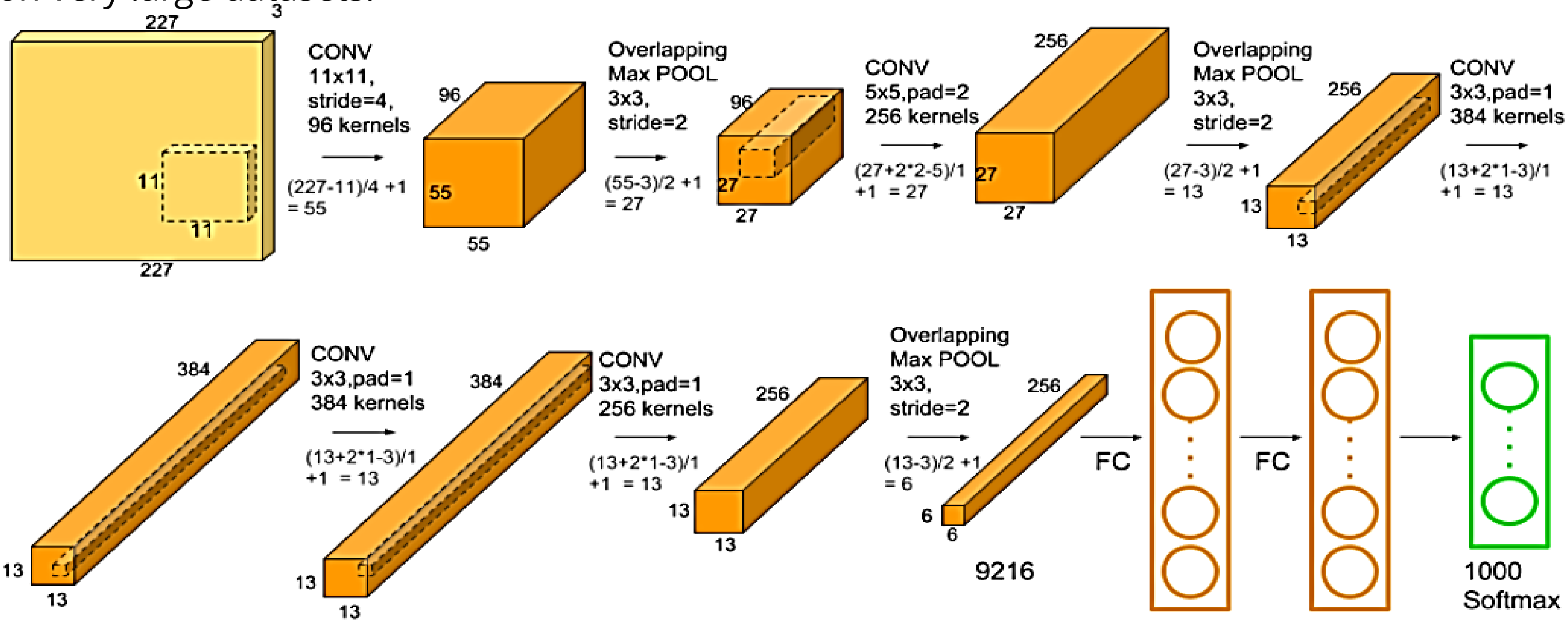
Yann LeCun, Leon Bottou, Bengio Patrick Haffner proposed a neural network architecture for handwritten and machine-printed character recognition working on 32x32 grayscale images

[https://engmrk.com/lenet-5-a-classic-cnn-architecture/]

[Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791]

AlexNet (2012)

Much larger than previous CNNs used for computer vision tasks (LeNet-5).
It has **60M parameters** and 650,000 neurons and took five to six days to train on two GTX 580 3GB GPUs. (2012)
Today there are much more complex CNNs that can run on faster GPUs very efficiently even on very large datasets.



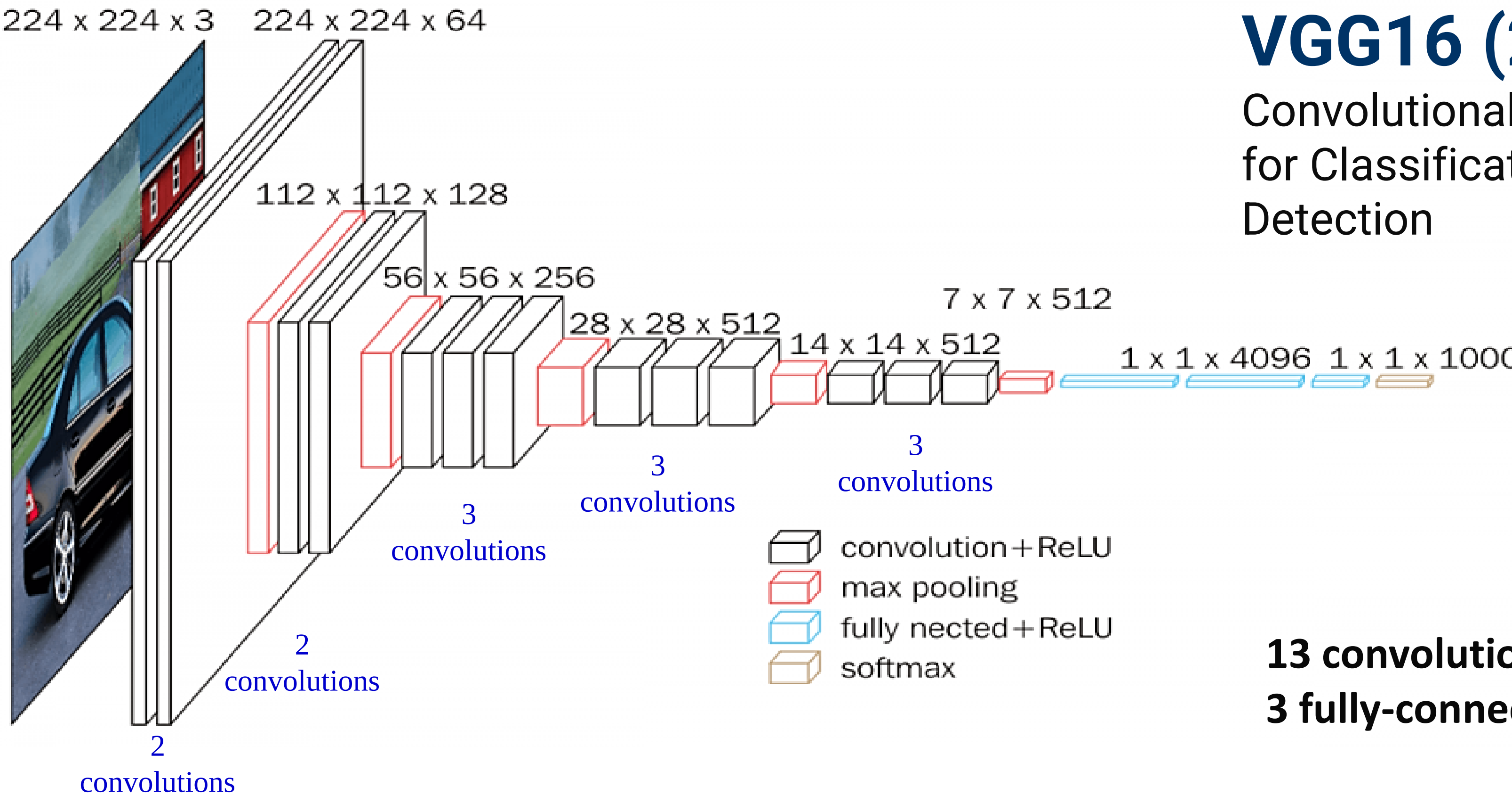
- **5 Convolutional Layers** and **3 Fully Connected Layers**
- **ReLU nonlinearity** is applied after all the convolution and fully connected layers
- A lot of hyperparameters

[ImageNet Classification with Deep Convolutional Neural Networks
by Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, 2012]

[https://www.learnopencv.com/understanding-alexnet/]

VGG16 (2014)

Convolutional Network for Classification and Detection



13 convolutional and 3 fully-connected layers

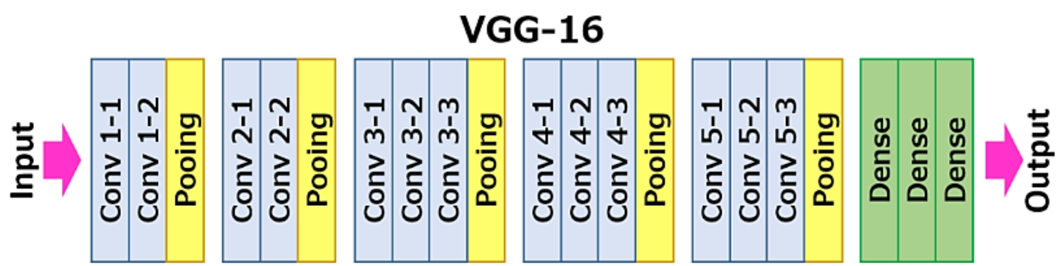
All convolutions: filter: 3x3, s=1, **same**

All max_pool: filter: 2x2, s=2

16 layers

~ 138M parameters

Deeper network with uniform convolution



[https://neurohive.io/en/popular-networks/vgg16/]

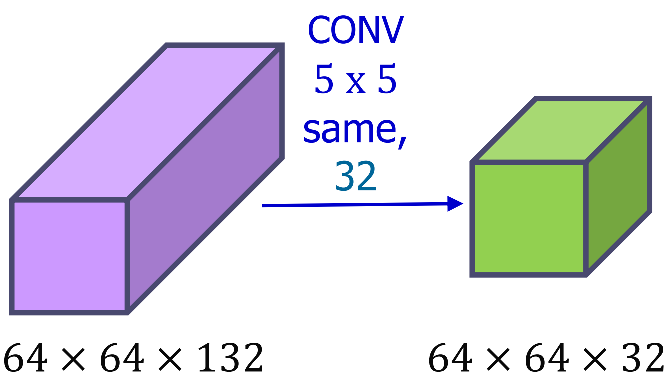
[Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." (2015).]

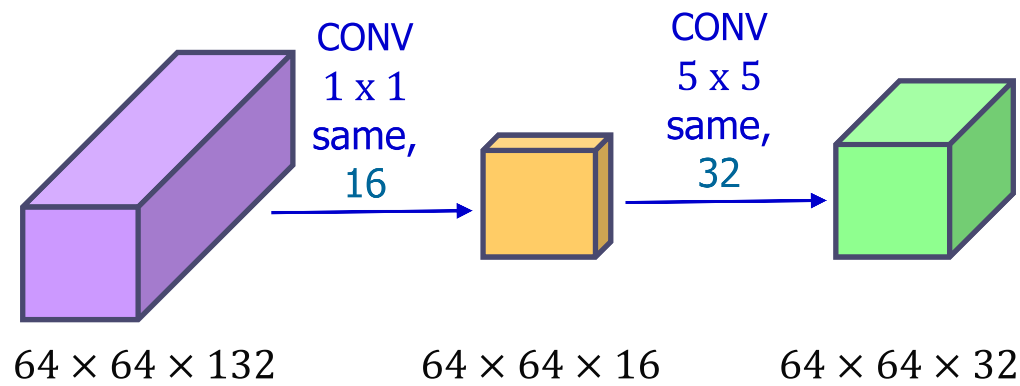
1 x 1 convolution – used in Inception network

$$(64 \times 64 \times 132) \rightarrow (64 \times 64 \times 32) \ \ using\ convolution$$

❖ A 1x1 convolution just maps an input pixel (with all its channels) to an output pixel, not looking at anything around itself.

a) **reduces the number of depth channels (and computation effort),** since it is (often very) slow to multiply volumes with extremely large depths





$$(5 \times 5 \times 132) \times (64 \times 64 \times 32) = \mathbf{432.5M}$$

Computation effort

$$(1 \times 1 \times 132) \times (64 \times 64 \times 16) = 8.65\text{M}$$

$$(5 \times 5 \times 16) \times (64 \times 64 \times 32) = 3.28\text{M}$$

$$8.65\text{M} + 3.28\ \text{M} = \mathbf{11.93M}$$

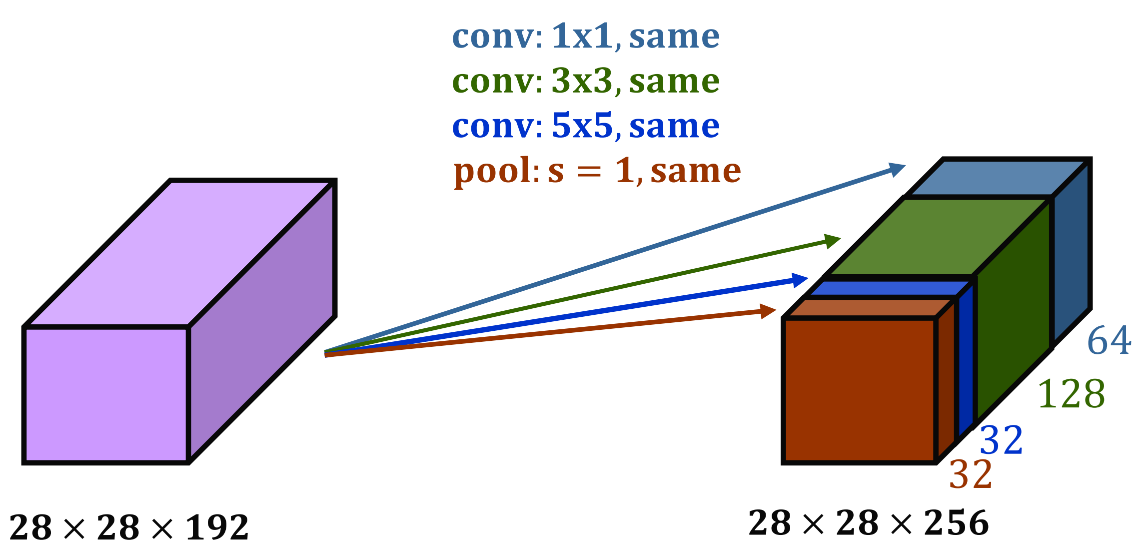
b) introduces nonlinearities, for example by adding a ReLU function after 1x1 convolution (can preserve the channel numbers)

pooling: reduces the dimensionality (image size), n_H, n_W
convolution: modifies the depth (number of channels), n_C

[Andrew Ng, Convolutional neural network, https://www.coursera.org/learn/convolutional-neural-networks/lecture/ZTb8x/networks-in-networks-and-1x1-convolutions]
Lin, Min & Chen, Qiang & Yan, Shuicheng. (2013). Network In Network.

Inception network (v1 -2014) (we need to go deeper)

- When designing a layer for a ConvNet, you might have to pick:
  - do you want a 1 by 1 filter, or 3 by 3, or 5 by 5, or do you want a pooling layer?
- What the inception network says:
  - ✓ why should you not **do them all in one step**?
- This makes the network architecture more complicated, but it also works remarkably well.



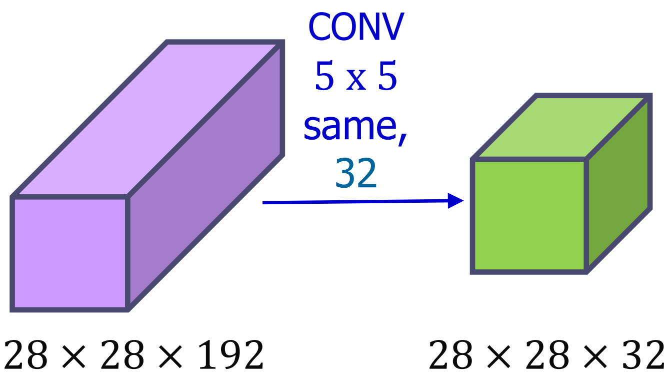
What about computation complexity?

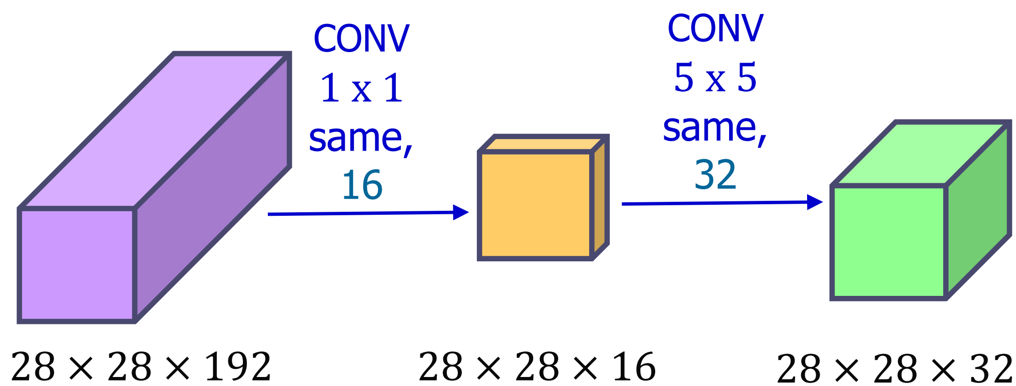
[C. Szegedy et al., "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.]

[Andrew Ng, Convolutional neural network, https://www.coursera.org/learn/convolutional-neural-networks/lecture/5WIZm/inception-network-motivation]

Inception network

Complexity reduction by using 1x1 convolution





$(5 \times 5 \times 192) \times (28 \times 28 \times 32) = \mathbf{120.4M}$

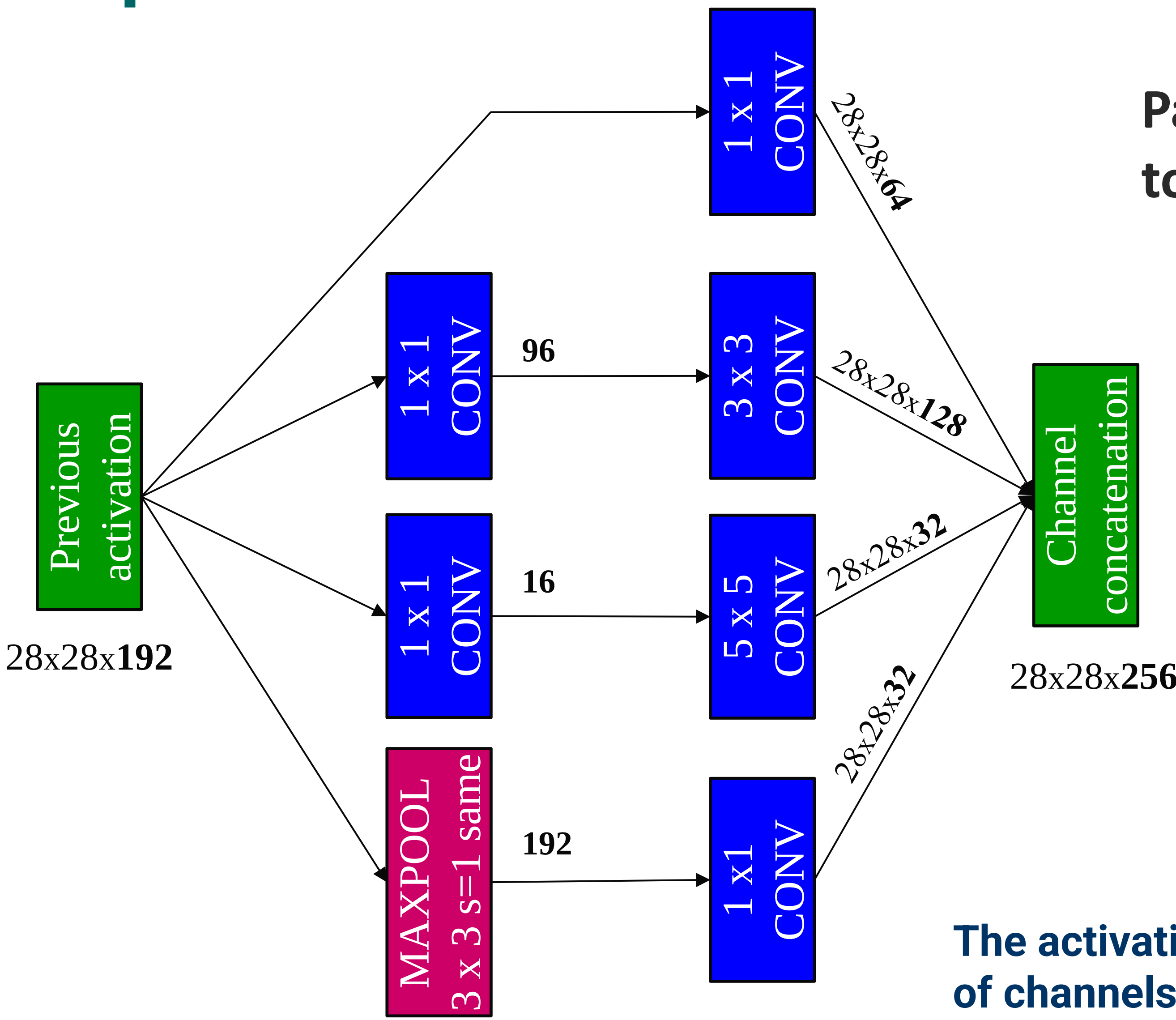
$(1 \times 1 \times 192) \times (28 \times 28 \times 16) = 2.4\text{M}$

$(5 \times 5 \times 16) \times (28 \times 28 \times 32) = 10.0\text{M}$

$8.65\text{M} + 3.28\text{ M} = \mathbf{12.4M}$

[C. Szegedy et al., "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.]
David White, Inception Network Overview, https://www.cs.colostate.edu/~dwhite54/InceptionNetworkOverview.pdf

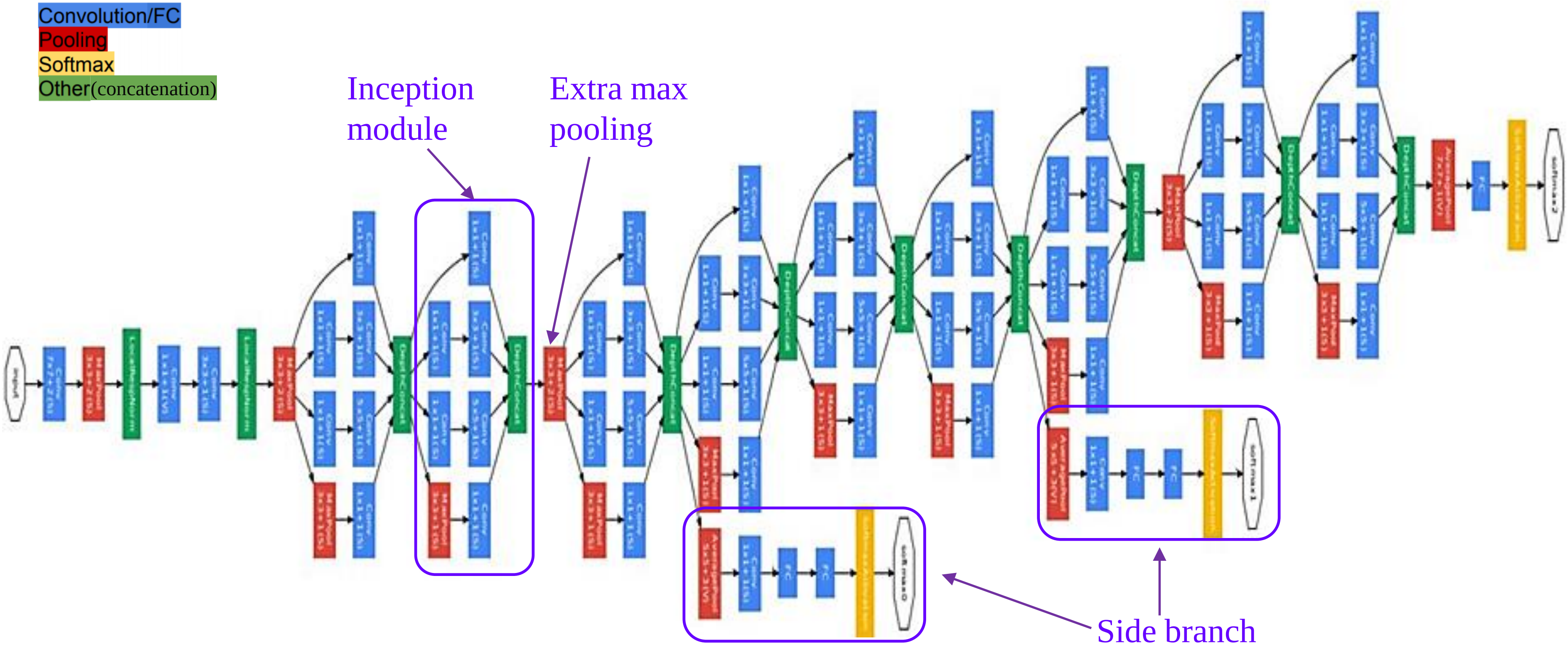
Inception module



Parallel towers

The activation size and number of channels are illustrative, for a particular module.

Inception network – Full Inception- V1



Side branch
Takes a hidden layer and try to use it to make a prediction

22-layer architecture with **5M parameters**

network in a (convolutional neural) network

improved utilisation of the computing resources inside the network

Building networks using modules/blocks.
Instead of stacking convolutional layers, we stack modules or blocks, within which are convolutional layers.
Hence the name Inception.

[Andrew Ng, Convolutional neural network, https://www.coursera.org/learn/convolutional-neural-networks/lecture/5WIZm/inception-network-motivation]

[C. Szegedy et al., "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.]

David White, Inception Network Overview, https://www.cs.colostate.edu/~dwhite54/InceptionNetworkOverview.pdf

Inception – V3 (2015)

Inception-v3 is a successor to Inception-v1, with **24M parameters**

The motivation for 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.

1. Factorising *n*×*n* convolutions into asymmetric convolutions: 1×*n* and *n*×1 convolutions
2. Factorise 5×5 convolution to two 3×3 convolution operations
3. Replace 7×7 to a series of 3×3 convolutions
4. Uses batch normalization

Illustrated: 10 CNN Architectures. A compiled visualisation of the common convolutional neural networks, Raimi Karim, Jul 29, 2019, https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

Inception-v4 (2016)

Anas BRITAS, Inception V4 CNN Architecture Explained. Inception-V4 CNN Architecture illustrated and Implemented in both Keras and PyTorch , 24.10.2021,
Inception V4 CNN Architecture Explained . | by Anas BRITAL | Oct, 2021 | Medium

Xception (2016)

Francois Chollet , Xception: Deep Learning with Depthwise Separable Convolutions, arXiv:1610.02357v3 [cs.CV] 4 Apr 2017, https://arxiv.org/pdf/1610.02357.pdf

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).

- 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.

Illustrated: 10 CNN Architectures. A compiled visualisation of the common convolutional neural networks, Raimi Karim, Jul 29, 2019, https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

ResNets (2015) Residual networks

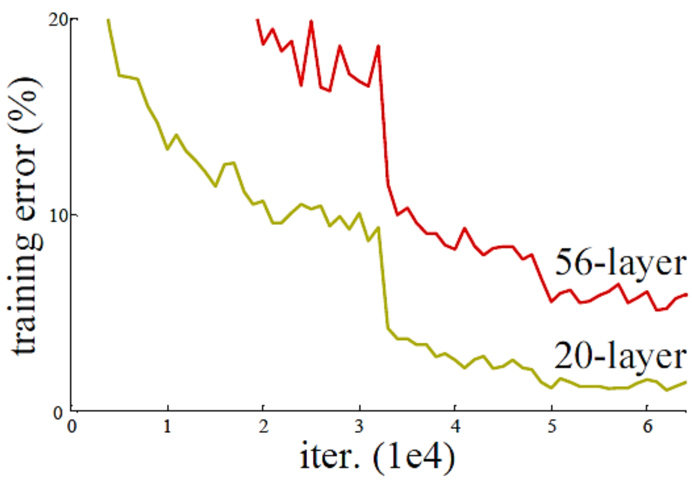
- According to the universal approximation theorem, given enough capacity, a feedforward network with a single layer is sufficient to represent any function.
- However, the layer might be massive, and the network is prone to overfitting the data.
- Therefore, a common trend is that the network architecture needs to go deeper.

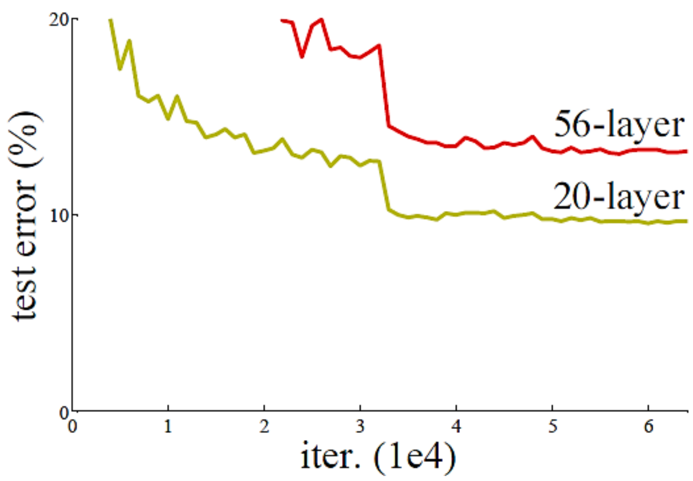
- Increasing network depth does not work by simply stacking layers together.
- Deep networks are hard to train because of the notorious vanishing (or exploding) gradient problem:
  - As the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient infinitively small (or large).
- As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly.

[Vincent Fung, An Overview of ResNet and its Variants, 2017, https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035]

Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "**plain**" networks. The deeper network has higher training error, and thus test error.

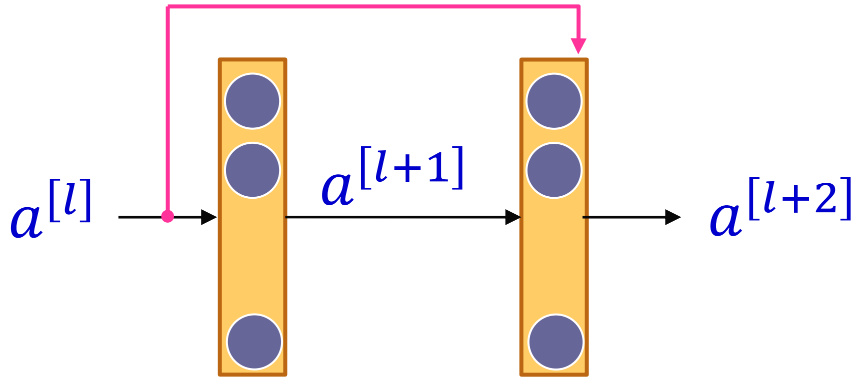
[He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2015-12-10). "Deep Residual Learning for ImageRecognition". arXiv:1512.03385]





ResNets

Residual networks - cont.



shortcut (skip connection)

$$a^{[l]} \rightarrow \text{Linear} \rightarrow \text{ReLU} \rightarrow \text{Linear} \rightarrow \oplus \rightarrow \text{ReLU} \rightarrow a^{[l+2]}$$

main path

$$z^{[l+1]} = W^{[l+1]}a^{[l]} + b^{[l+1]} \qquad a^{[l+1]} = g(z^{[l+1]}) \qquad z^{[l+2]} = W^{[l+2]}a^{[l+1]} + b^{[l+2]} \qquad \cancel{a^{[l+2]} = g(z^{[l+2]})}$$

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

makes the residual network

Avoids the problem of vanishing gradients by reusing activations from a previous layer until the adjacent layer learns its weights.

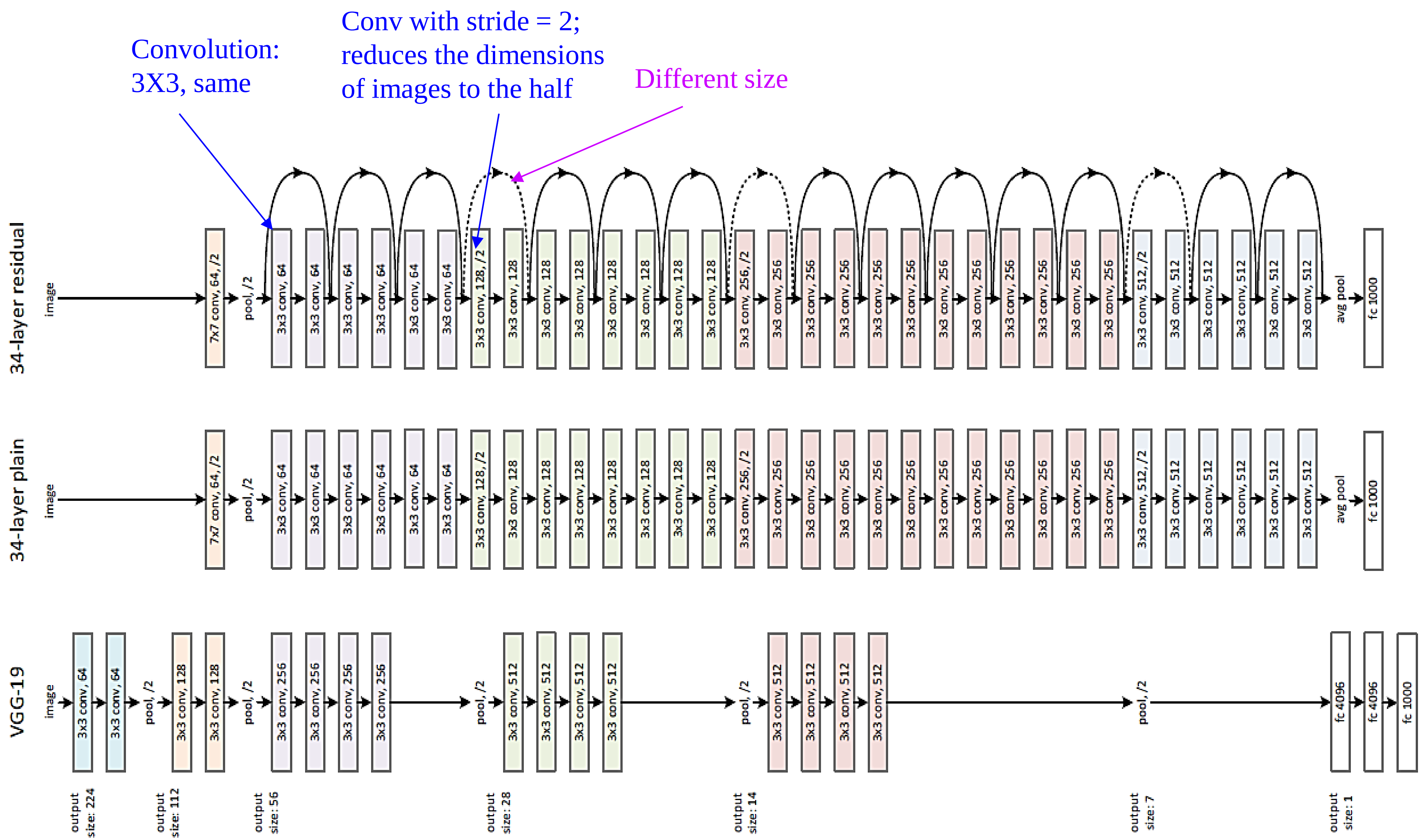
Skipping effectively simplifies the network, using fewer layers in the initial training stages.
This speeds learning by reducing the impact of vanishing gradients, as there are fewer layers to propagate through.
The network then gradually restores the skipped layers as it learns the feature space.

Towards the end of training, when all layers are expanded, it stays closer to the manifold and thus learns faster.

A neural network without residual parts explores more of the feature space. This makes it more vulnerable to perturbations that cause it to leave the manifold and necessitates extra training data to recover.

[https://en.wikipedia.org/wiki/Residual_neural_network]

ResNets Residual networks - cont.



[He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2015-12-10). "Deep Residual Learning for Image Recognition". arXiv:1512.03385]

Number of Layers	Number of Parameters
ResNet 18	11.174M
ResNet 34	21.282M
ResNet 50	23.521M
ResNet 101	42.513M
ResNet 152	58.157M

Pablo Ruiz, Understanding and visualizing ResNets, Oct 8, 2018,
https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8