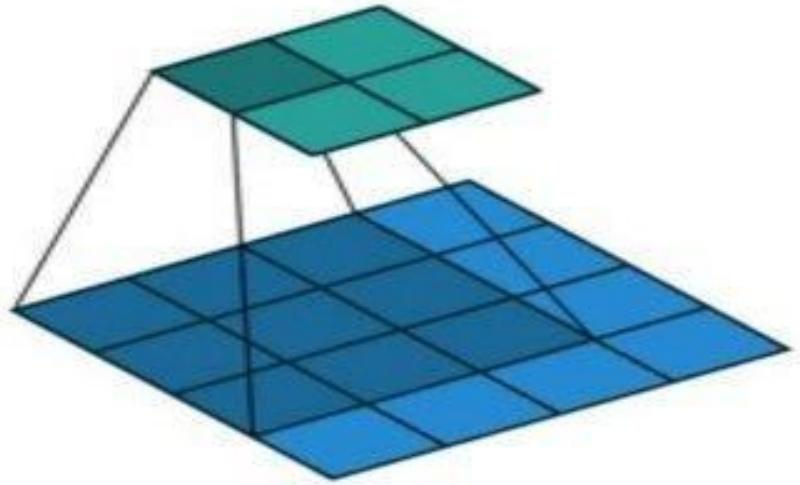
CNN Architectures

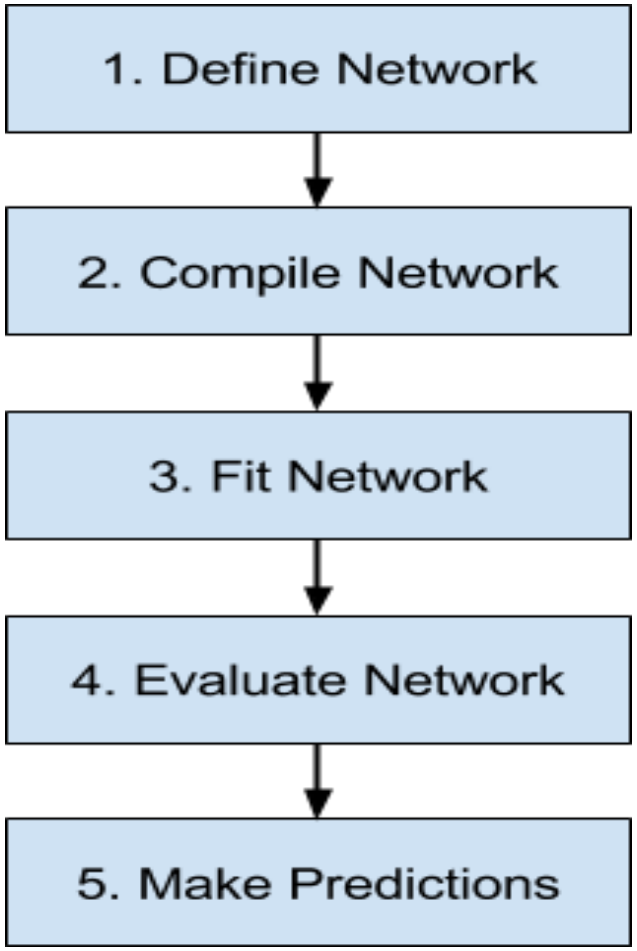
Building a Convolutional Neural Network (CNN) in Keras

- The Keras library in Python makes it pretty simple to build a CNN.
- A convolution multiplies a matrix of pixels with a filter matrix or 'kernel' and sums up the multiplication values. Then the convolution slides over to the next pixel and repeats the same process until all the image pixels have been covered.



Steps to Train and Build a CNN Model

- Import the modules
- Loading the dataset
- Exploratory data analysis
- Data pre-processing
- Building the model
- Compiling the model
- Training the model
- Using our model to make predictions



CNN Architectures

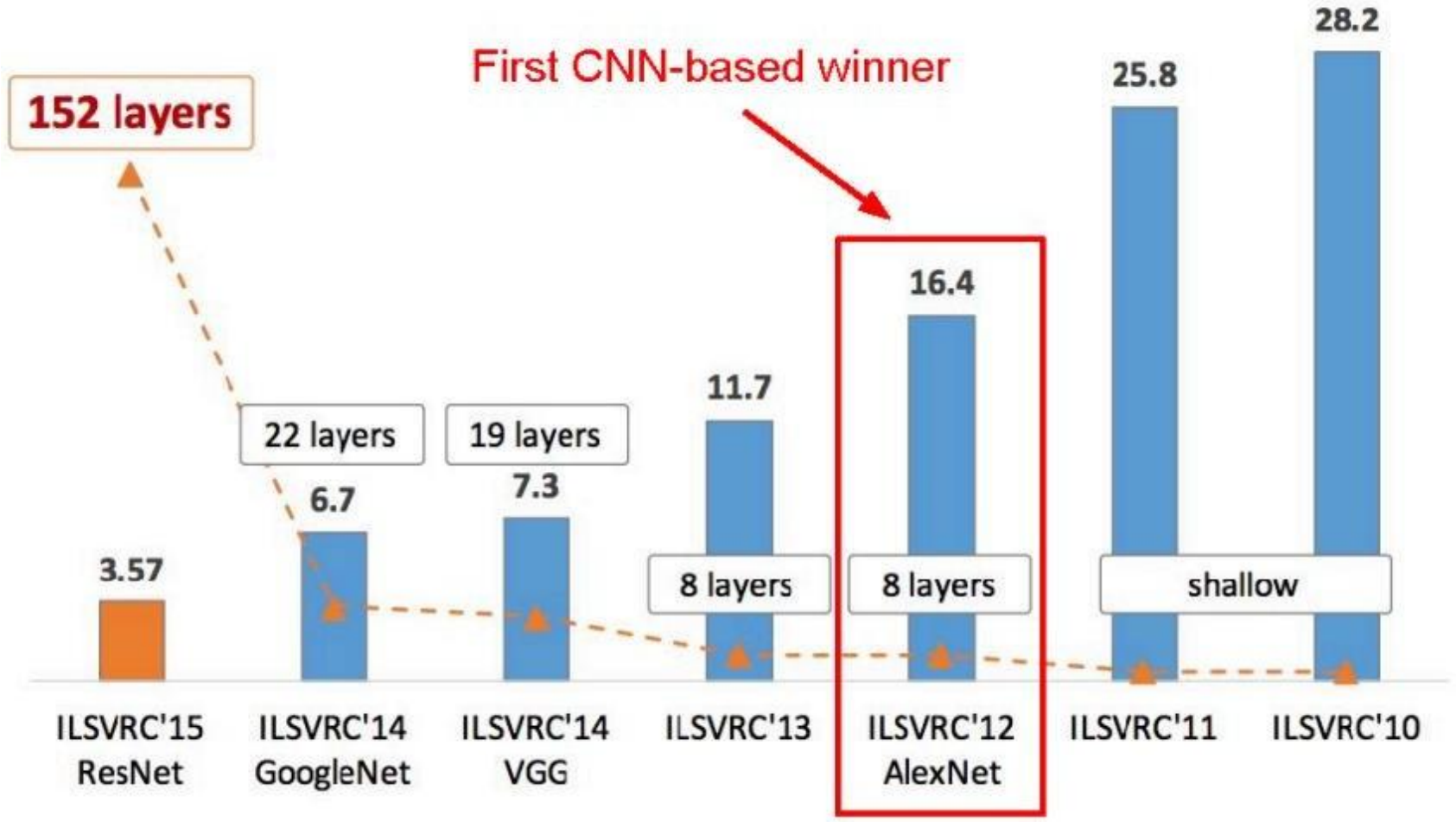
- The ImageNet project is a large visual database designed for use in visual object recognition software research.
- The ImageNet project runs an annual software contest, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), where software programs compete to correctly classify and detect objects and scenes.

- LeNet-5 (1998)
- AlexNet(2012)
- ZefNet (2013)
- Visual geometry group (VGG) (2014)
- GoogLeNet (2014)
- Highway network (2015)

- ResNet (2015)
- DenseNet (2017)
- ResNext (2016 Runnerup)
- WideResNet (2016)
- Pyramidal Net (2017)
- Xception (2017)

2023-
BASIC-L (Lion, fine-tuned)

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) Winners

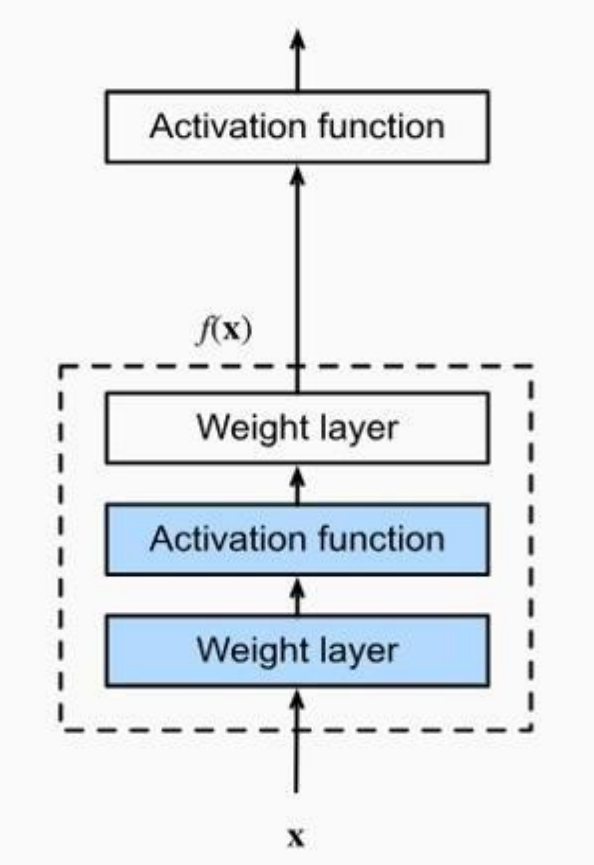


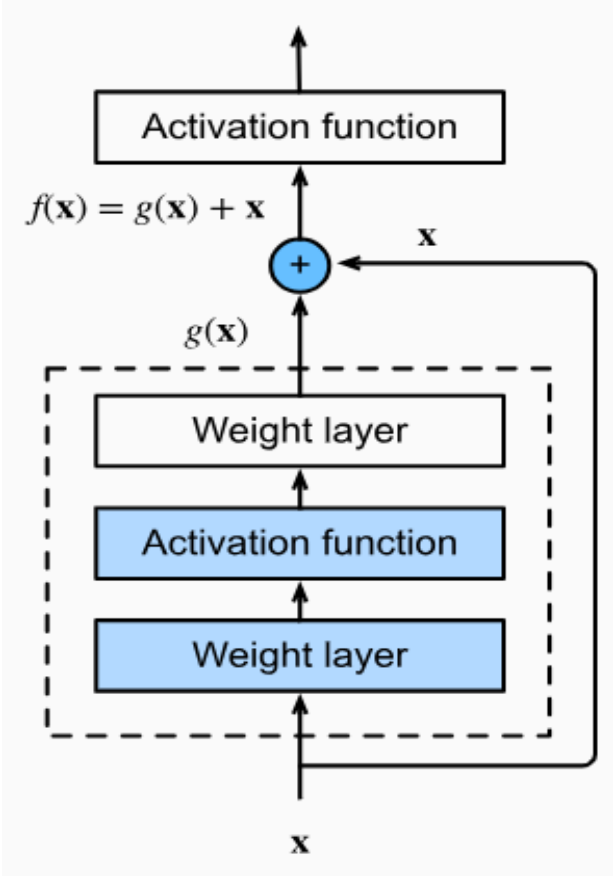
ResNet (Residual Network)

- Residual connections were introduced in 2015 by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.
- This architecture introduced the concept called Residual Blocks. That solves the problem of the **vanishing/exploding gradient.**
- In this network, use a technique called **skip connections.**
- The **skip connection** connects **activations of a layer** to further layers by skipping some layers in between. This forms a residual block. Results are made by stacking these residual blocks together.
- The approach behind this network is instead of layers learning the underlying mapping, to allow the network to fit the residual mapping. So, instead of say H(x), initial mapping, let the network fit.

ResNet (Residual Network)

$F(x) := H(x) - x$ which gives $H(x) := F(x) + x$.

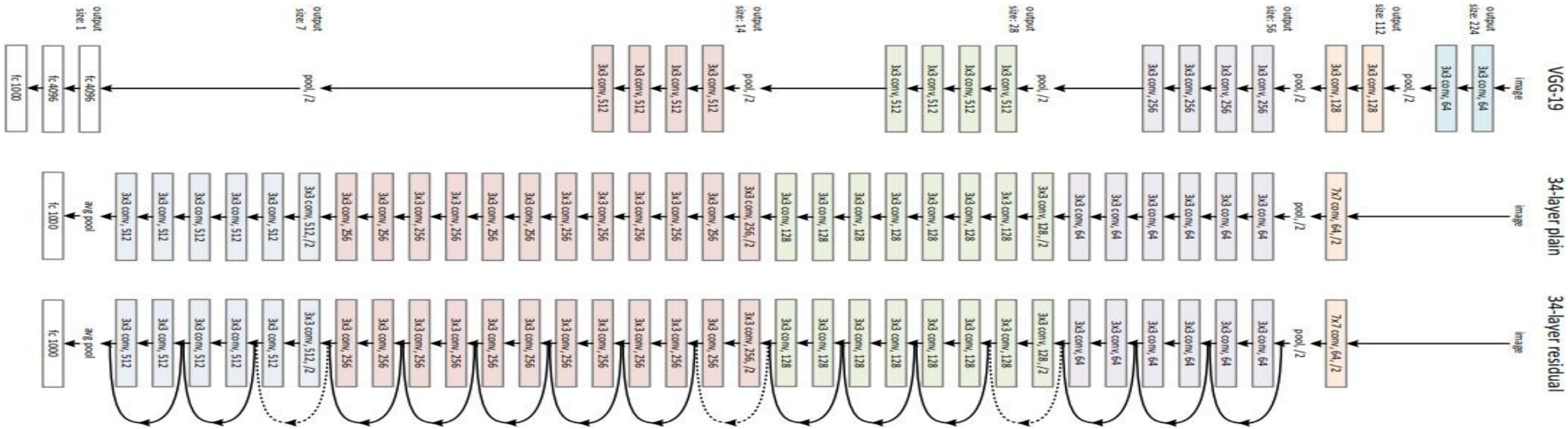




The advantage of adding this type of skip connection is that if any layer hurt the performance of architecture then it will be skipped by regularization.

ResNet (Residual Network)

- Network Architecture: This network uses a **34-layer plain network architecture** inspired by **VGG-19** in which then the **shortcut connection is added**. These shortcut connections then **convert the architecture into a residual network.**



ResNet (Residual Network)

- Even though ResNet is much deeper than VGG16 and VGG19, the model size is actually substantially smaller due to the usage of **global average pooling** rather than **fully-connected layers** — this reduces the model size down to 102MB for ResNet50.

Variants of ResNets

ResNet-18
ResNet-50
ResNet-101
ResNet-152
ResNet-1000

Summary

Comparison					
Network	Year	Salient Feature	top5 accuracy	Parameters	FLOP
AlexNet	2012	Deeper	84.70%	62M	1.5B
VGGNet	2014	Fixed-size kernels	92.30%	138M	19.6B
Inception	2014	Wider - Parallel kernels	93.30%	6.4M	2B
ResNet-152	2015	Shortcut connections	95.51%	60.3M	11B

VGG

The **VGG (Visual Geometry Group) network** is a **deep convolutional neural network (CNN)** introduced in 2014. It became famous for achieving **high accuracy** in image classification tasks, especially in the **ImageNet Challenge**.

1. **Deep Architecture:**
   - VGG-16 and VGG-19 have **16 and 19 layers**, respectively.
   - Uses **small 3×3 convolution filters** but stacks them deep.
2. **Simple & Uniform Design:**
   - All convolutional layers use **3×3 filters with stride 1**.
   - Max pooling layers use **2×2 filters with stride 2**.
   - Fully connected (FC) layers at the end.
3. **High Computational Cost:**
   - Requires more memory and processing power.
   - Large number of parameters (~138 million in VGG-16).

VGG Variants

- **VGG-16**: 16 layers (13 conv + 3 FC)
- **VGG-19**: 19 layers (16 conv + 3 FC)

Architecture of VGG-16

VGG-16 consists of **5 convolutional blocks**, followed by **fully connected layers**:

Layer Type	Filters	Kernel Size	Activation	Output Size
Conv Layer (x2)	64	3×3	ReLU	224×224×64
Max Pooling	-	2×2	-	112×112×64
Conv Layer (x2)	128	3×3	ReLU	112×112×128
Max Pooling	-	2×2	-	56×56×128
Conv Layer (x3)	256	3×3	ReLU	56×56×256
Max Pooling	-	2×2	-	28×28×256
Conv Layer (x3)	512	3×3	ReLU	28×28×512
Max Pooling	-	2×2	-	14×14×512
Conv Layer (x3)	512	3×3	ReLU	14×14×512
Max Pooling	-	2×2	-	7×7×512
Flatten	-	-	-	25088
Fully Connected	4096	-	ReLU	4096
Fully Connected	4096	-	ReLU	4096
Output Layer	#Classes	-	Softmax	#Classes

GoogleNet (Inception v1)

GoogleNet (also called Inception v1) was introduced by **Szegedy et al. in 2014** and **won the ILSVRC-2014** ImageNet competition. It is one of the first deep CNN architectures designed to be **efficient while maintaining high accuracy.**

1. **Inception Modules → "Multi-scale Feature Extraction"**
    - Instead of using a single type of convolution, GoogleNet **combines multiple filter sizes (1×1, 3×3, 5×5) at the same time.**
    - This allows the network to **capture both fine and coarse details.**
2. **1×1 Convolutions for Dimensionality Reduction**
    - Instead of directly applying expensive 3×3 or 5×5 convolutions, GoogleNet first applies **1×1 convolutions** to **reduce the number of channels**, saving computation.

3. **Deeper but More Efficient**
    - **22 layers deep**, but uses **fewer parameters (~5 million)** compared to **VGG-16 (~138 million)**.
4. **Global Average Pooling Instead of Fully Connected Layers**
    - Instead of adding **fully connected layers** at the end (which have lots of parameters), GoogleNet uses **Global Average Pooling (GAP)**.
    - This reduces overfitting and improves efficiency.

The Inception module applies multiple filters **at the same time** and concatenates the results:

Layer Type	Filter Size	Purpose
1×1 Conv	1×1	Reduces depth (fewer parameters)
3×3 Conv	3×3	Captures medium-scale features
5×5 Conv	5×5	Captures large-scale features
Max Pooling	3×3	Helps with spatial reduction

Architecture

Layer	Details
Conv Layer 1	7×7, 64 filters
Max Pooling	3×3
Conv Layer 2	1×1 + 3×3
Max Pooling	3×3
Inception Module 1	(1×1, 3×3, 5×5, Max Pool)
Inception Module 2	(1×1, 3×3, 5×5, Max Pool)
Max Pooling	3×3
Inception Module 3-5	More stacked modules
Max Pooling	3×3
Inception Module 6-9	More stacked modules
Global Average Pooling	Reduces features
Fully Connected	1000 classes (ImageNet)
Softmax	Final classification

Comparison of Pre-trained models

Feature	GoogleNet (Inception v1)	VGG-16	ResNet-50
Introduced By	Szegedy et al. (2014)	Simonyan & Zisserman (2014)	Kaiming He et al. (2015)
Depth	22 layers	16 layers	50 layers
Parameters	5M (Very Efficient)	138M (Heavy Model)	25M (Moderate)
Key Feature	Inception Modules (Multi-scale features)	Deep Stacked Convolutions	Residual Connections (Skip Connections)
Fully Connected Layers?	No (Uses Global Average Pooling)	Yes (Fully Connected Dense Layers)	No (Global Average Pooling)
Training Time	Medium	Slow (Very Large Model)	Fast (Efficient due to Residual Learning)
Accuracy (ImageNet)	Higher than VGG-16	Good but outdated	Best among the three
Gradient Vanishing Problem?	Yes (Deep but no residual connections)	Yes (Very deep, vanishing gradients)	No (Skip connections solve it)