Empirical loss: measures the total loss over the entire dataset.

Binary cross entropy loss: can be used with models that output a probability between 0 and 1

$$J(W) = -\frac{1}{N}\sum_{i=1}^{N} \underbrace{y^{(i)}}_{\text{actual}} \underbrace{\log\left(f(x^{(i)};W)\right)}_{\log(\text{predicted})} + \underbrace{\left(1-y^{(i)}\right)}_{1-\text{actual}} \underbrace{\log\left(1-f(x^{(i)};W)\right)}_{\log(1-\text{predicted})}$$

Mean squared error loss: can be used with regression models that output continuous real numbers

$$J(W) = \frac{1}{N}\sum_{i=1}^{N} \left(\underbrace{y^{(i)}}_{\text{actual}} - \underbrace{f(x^{(i)};W)}_{\text{predicted}}\right)^2$$

Regularization: technique that constraints our optimization problem to discourage complex models. With it we can improve generalization of our model and reduce overfitting.

- **Dropout**: during training, randomly set some activations to 0. (consider for large fully-connected layers)
- **Early stopping**: stop training before we have a chance to overfit.
- **Data augmentation**: artificially increase the size and diversity of the training set (always a good idea)
  - horizontal flips
  - random crops and scales
  - color jitter
  - translation
  - rotation
  - stretching
  - cutout
  - **mixup**: randomly blend the pixels of a pair of training images (small classification datasets)

Batch normalization : usually inserted after fully connected or convolutional layers, it is a method used to make networks training faster and more stable. It works by normalizing the input of the layers by re-centering and re-scaling it. It reduces the internal covariate shift problem

$$x_i \leftarrow \frac{x_i - \mu_i}{\sqrt{\sigma_i^2 + \varepsilon}} \qquad \forall \text{ coefficient of an input feature } x$$

μ_i = mean
σ_i = variance
mean and variance are computed over a set S_i of coefficients, ε is a small constant added for numerical stability (avoid division by 0).
The difference on normalization method is how the set S_i is chosen.

feature $X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$ Batch of size $N = 3$

$H \times W = 1 \times 2$

first batch :

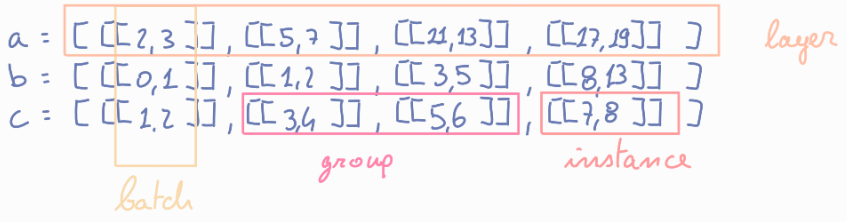
x_1 x_2 x_3 x_4

a = [[[2, 3]], [[5, 7]], [[11, 13]], [[17, 19]]]
b = [[[0, 1]], [[1, 2]], [[3, 5]], [[8, 13]]]
c = [[[1, 2]], [[3, 4]], [[5, 6]], [[7, 8]]]

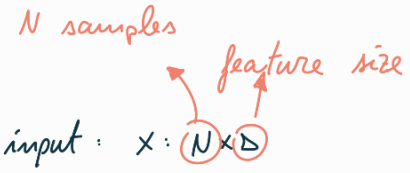
} 3

channel 1 channel 2 channel 3 channel 4

Batch normalization : normalizes by channels (depends on batch size) (e.g. $S_i = a_1 \cup b_1 \cup c_1$)
Layer normalization : S_i defined as all the coefficients that belong to the same input feature (e.g. $S_i = a$, $S_i = b$, $S_i = c$)
Instance normalization : S_i defined as all the coefficients that belong to the same input feature and the same channel (e.g. $S_i = a_1$, $S_i = b_3$, $S_i = c_2$, ...)
Group normalization : like instance normalization but for multiple instances (same feature, group of channels)



Internal covariate shift : for a deep learning model, as the parameter of a lower layer changes, the input distribution of the upper layer also changes;
- the upper layer's parameters need to continuously adapt to new input data distribution, reducing learning speed;
- the input of upper layer may tend to become too large or too small (exploding/vanishing gradient)



$$\mu_j = \frac{1}{N} \sum_{i=1}^{N} x_{i,j}$$ per-channel mean, shape is D

$$\sigma_j^2 = \frac{1}{N} \sum_{i=1}^{N} (x_{i,j} - \mu_j)^2$$ per-channel variance, shape is D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$ normalized x, shape is N×D

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$ output, shape is N×D

γ, β are learnable scale and shift parameters

CNN

Stride : parameter that quantifies the movement of the filter over the image

$$output\_size = (image\_width - Kernel\_width)/stride + 1$$

$$n\_parameters = (Kernel\_width \cdot Kernel\_width \cdot Kernel\_depth + 1) \cdot num\_kernels$$

Pooling layer : used to reduce spatial dimensions of feature maps. The pooling operation consists on dividing the input feature map into windows (overlapping or not) and performing a pooling operation within each region to produce a downsampled output.

- **max pooling** : the maximum activation value within each pool is retained
- **avg pooling** : computes an average of the values in the pool

$$output\_size = (image\_width - pool\_window\_width)/stride + 1$$

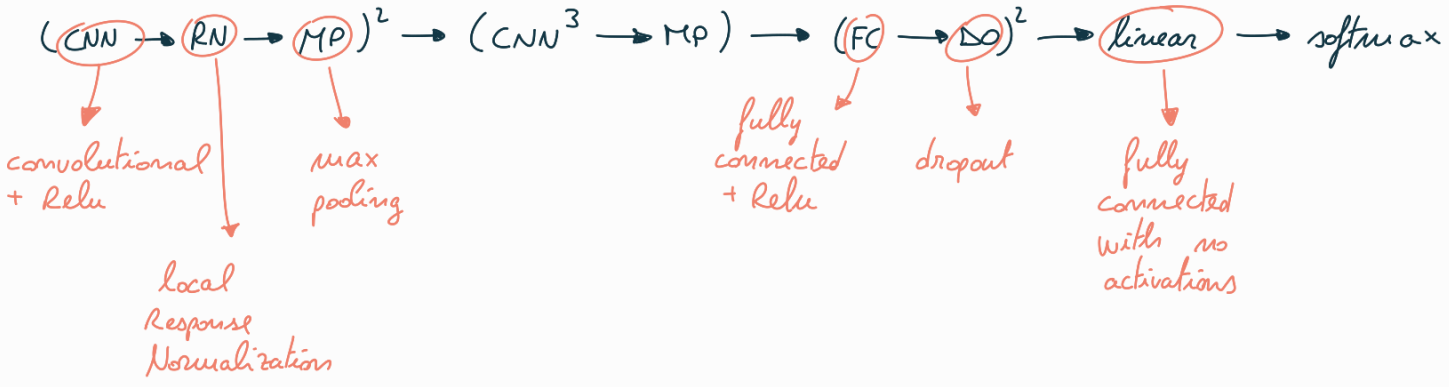
$$n\_parameters = 0$$

ImageNet Large-Scale Visual Recognition Challenge → proposed by Alex something

AlexNet (winner of INLSVRC 2012)

It consists of 8 layers: first 5 are convolutional, last 3 are fully connected.

CONV	1	
MAX POOL	1	
CONV	2	
MAX POOL	2	
CONV	3	
CONV	4	
CONV	5	
MAX POOL	3	
FC	6	Relu
FC	7	Relu
FC	8	Softmax



Key innovations:

1. ReLU activation were used instead of traditional ones, like sigmoid or tanh. RELU helps mitigate the vanishing gradient problem and accelerates training.
2. Overlapping pooling with stride 2 enable higher resolution and richer feature maps compared to non-overlapping pooling.
3. Local Response Normalization provide local competition between adjacent neurons and enhance the network's ability to generalize.
4. Dropout was used to mitigate overfitting; it randomly drops a fraction of neurons during training.
5. Large use of data augmentation

proposed by Visual Geometry Group (Oxford)

VGG Net (winner of ILSVRC 2014)

Compared to AlexNet, it uses smaller filters and is deeper

- smaller filters → stack of 3×3 conv layers has the same effective receptive field as one 7×7 conv layer
- deeper → more non-linearities

Effective Receptive Field: region of data that influences the output of a particular neuron or feature map in a network layer.

Key innovations:

① Small convolutional filters

② Max pooling layers with fixed 2×2 filter and stride 2; the pooling layers progressively reduce spatial dimension while retaining important features

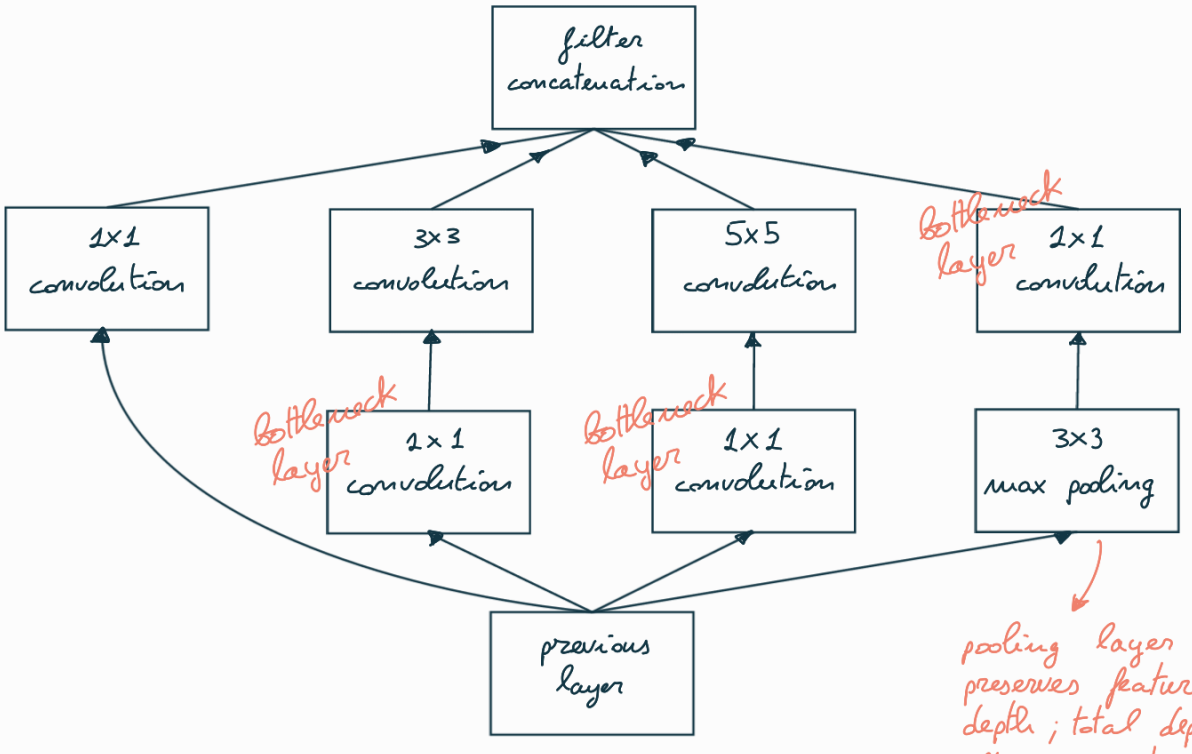
③ Used ensembles for best results

GoogleNet (winner of INLSVRC in 2014)

GoogleNet's primary innovation lies in its use of inception modules. It only has 5 million parameters (12x less than AlexNet, 27x less than VGG-16) in 22 layers.

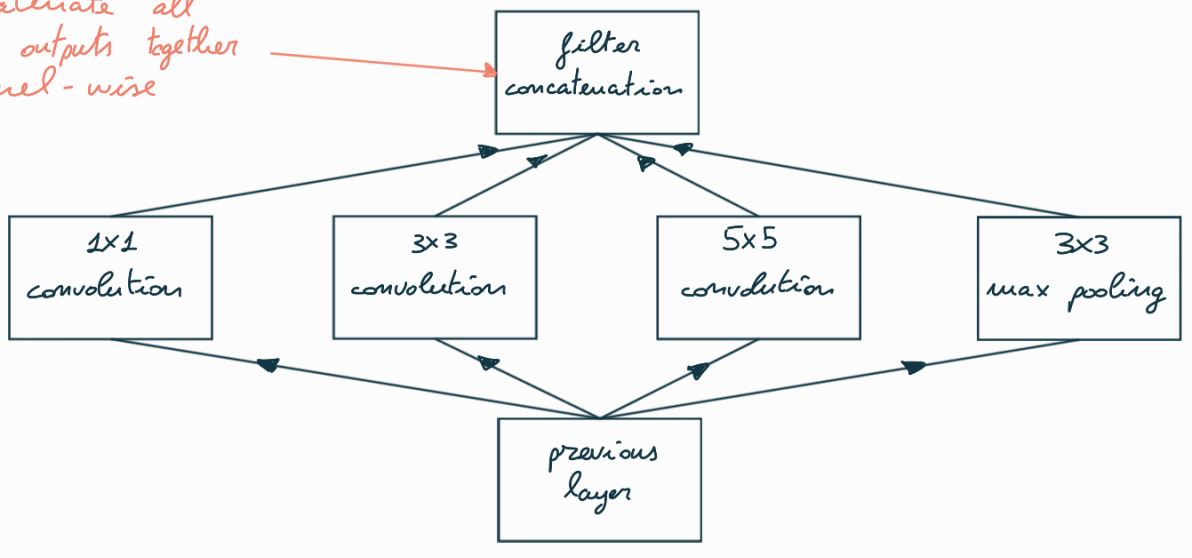
Key innovations:

① Inception modules: the inception module is the building block of GoogleNet and consists of multiple parallel convolutional layers of different sizes (1x1, 3x3, 5x5, ...). These parallel operations capture features at various spacial scales, enabling the network to learn both local and global information efficiently. Inception modules are local networks (network within a network).



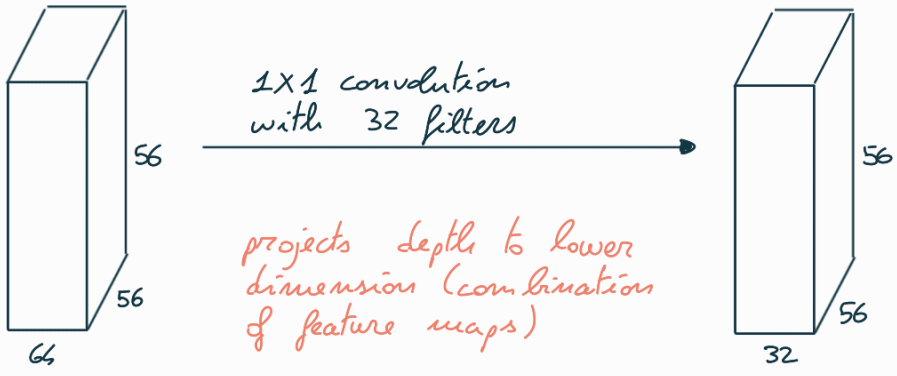
Inception Module

concatenate all filter outputs together channel-wise



Naive Inception Module

The problem with this approach is that this is very expensive to compute.
Solution: Bottleneck layers that use 1x1 convolutions to reduce feature channel size



After the last convolutional layer a global average pooling layer is used that spatially averages across each feature map before the final FC classification layer.
No longer multiple expensive FC layers.

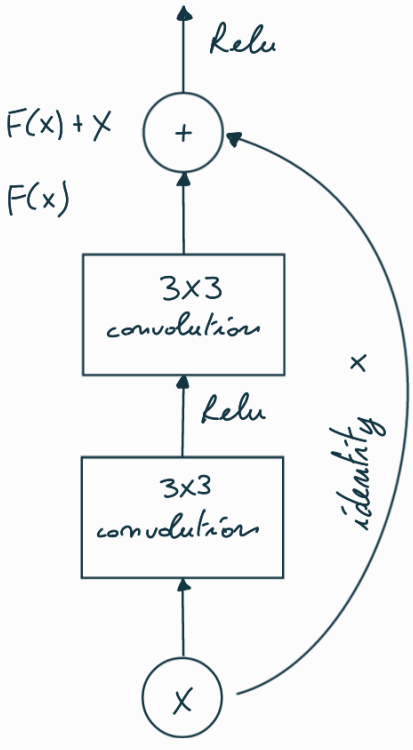
② Another notable aspect is the use of auxiliary classifiers. In addition to the main classification output, there are auxiliary classifiers attached to intermediate layers of the network used during training to combat the vanishing gradient problem and provide additional regularization. (common in deep neural networks)
By introducing these auxiliary classifiers the network encourages the flow of gradients through different depths of the network. The intermediate layers receive gradients not only from the subsequent layers but also from the auxiliary classifiers. The auxiliary classifiers are connected to the main loss function of the network. Their gradients contribute to the overall gradient update.

Microsoft

ResNet (winner of ILSVRC 215)

ResNet is a very deep network using residual connections.
152 layers

Key innovations:



Residual Block

① Residual connections (also known as skip connections) enable successful training of very deep neural networks.

Traditional networks learn by stacking layers on top of each other, where each layer applies a transformation to its input to produce an output. As the network gets deeper the vanishing gradient problem becomes more pronounced.

Residual connections address this problem by bypassing one or more layers, directly connecting the input of a layer to the output of a later layer. This connection creates a shortcut path.

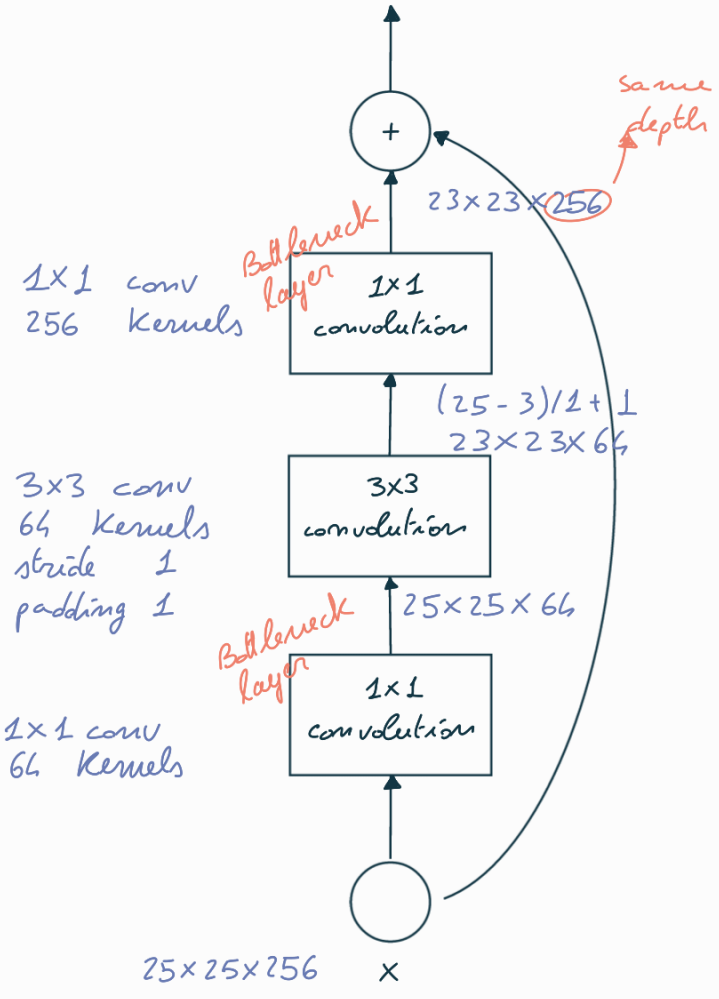
Output = Input + Residual

The network more effectively learns the residual more effectively, rather than the complete transformation from scratch. This residual learning enables the network to focus on refining the already learned information, which tends to be easier for the network to optimize.

Residual blocks also facilitates the flow of the gradients during backpropagation since the shortcut path directly connects the input to the output potentially bypassing numerous layers. This alleviates the vanishing gradient problem.

Residual connections also provide stability to the learning process. In the absence of a residual connection, if a layer fails to learn the desired transformation it may completely disrupt the information flow in subsequent layers. However, with the residual connections, even if a layer does not learn the desired transformation well it can still contribute by learning the residual mapping, ensuring that the information flow is not completely lost.

② Residual + Bottleneck



Residual Block
+
Bottleneck
for very deep
networks

Squeeze-and-Exitation Network

SENet (winner of INLSVRC in 2017)

This network aims to enhance the representation power of convolutional neural networks. It adds a Feature Recalibration module that learns to adaptively reweight feature maps enabling the network to adaptively emphasize informative features and suppress less relevant ones.

Key innovation : Attention mechanism

Attention = Squeeze + Extraction

① Squeeze : extract information from features maps by performing global average pooling across spatial dimensions. Global average pooling computes the average value for each channel, resulting in a channel descriptor that captures the overall importance of each channel.

② Extraction : the squeeze module is followed by two fully connected layers which introduce non-linearities and model channel interdependencies. We obtain coefficients for each channel that represent how much each channel should be emphasized or suppressed.

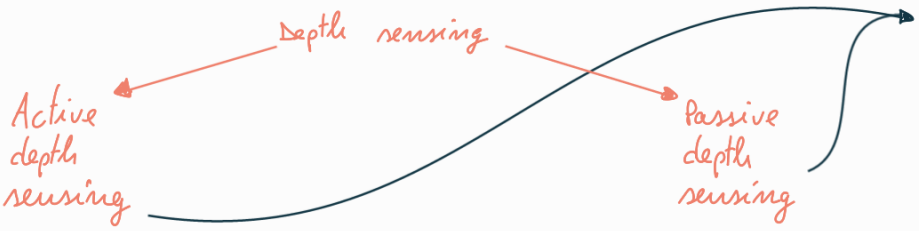
Identity Mappings in Deep Residual Networks
- improved ResNet block

Wide Residual Networks
- argues that residuals are the important factor, not depth
- use wider residual blocks
- increasing width instead of depth is more computationally efficient (parallelizable)

Aggregated Residual Transformations for Deep Neural Networks
- multiple parallel pathways

Densely Connected Convolutional Networks
- each block is connected to every other layer in a feedforward fashion
- alleviates vanishing gradient, strengthens feature propagation, encourages feature reuse

MobileNets: Efficient Convolutional Neural Networks for Mobile Applications



two methods used to determine depth information in various applications. They differ on how they capture and infer depth

actively emitting some form of energy or signal and measure its reflection or interaction with the surrounding objects; depth is perceived by perturbing the environment

- Time of Flight
- Structured light (Kinect)
- Laser triangulation (LIDAR)

capturing and analyzing the light in the environment to infer depth deformation; depth is reconstructed or estimated

- Binocular stereo (epipolar geometry)
- Monocular
- Multi-view (structure from motion)

In computer vision, existing solutions to depth estimation from a single image usually rely on Deep learning based approaches

Self supervised learning

- Deep unsupervised learning refers to training deep neural networks using unlabeled data without any explicit supervision signal. The objective is to learn useful representations or features that capture the underlying structure and patterns in the data.
  - Autoencoders
  - GANs
- Self-supervised learning is a specific form of unsupervised learning where the learning task is created from the input data itself. It leverages the inherent structure or information within the data to define a pretext task, which is used as a supervision signal for training (generate artificial labels.
  - Denoising Autoencoders
  - In-Painting
  - Instance Discrimination

Domain adaptation : transfering knowledge learned from a source domain to a target domain. Domain adaptation is crucial in situations where labeled data is abundant in the source domain but limited or unavailable in the target domain.

GANs

Generative : learn a generative model
Adversarial : trained in an adversarial setting
Network : use deep neural networks