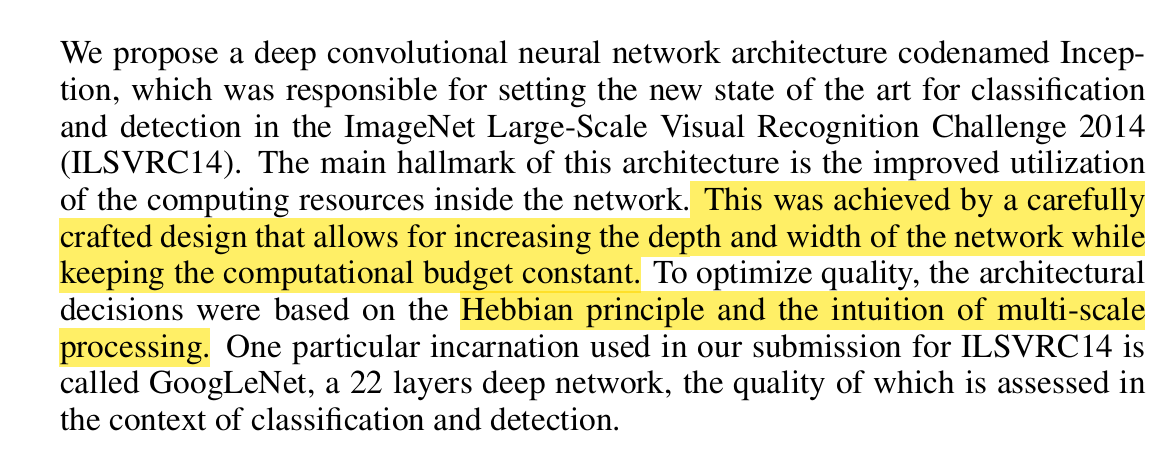
INCEPTION NETWORK

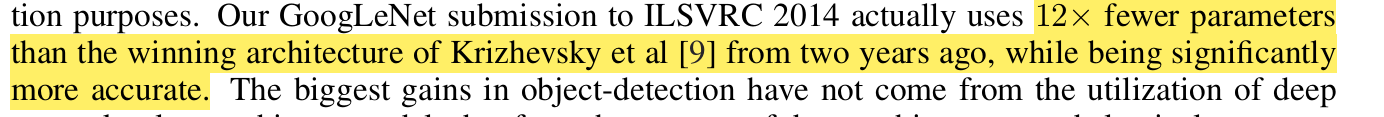
**ABSTRACT:**



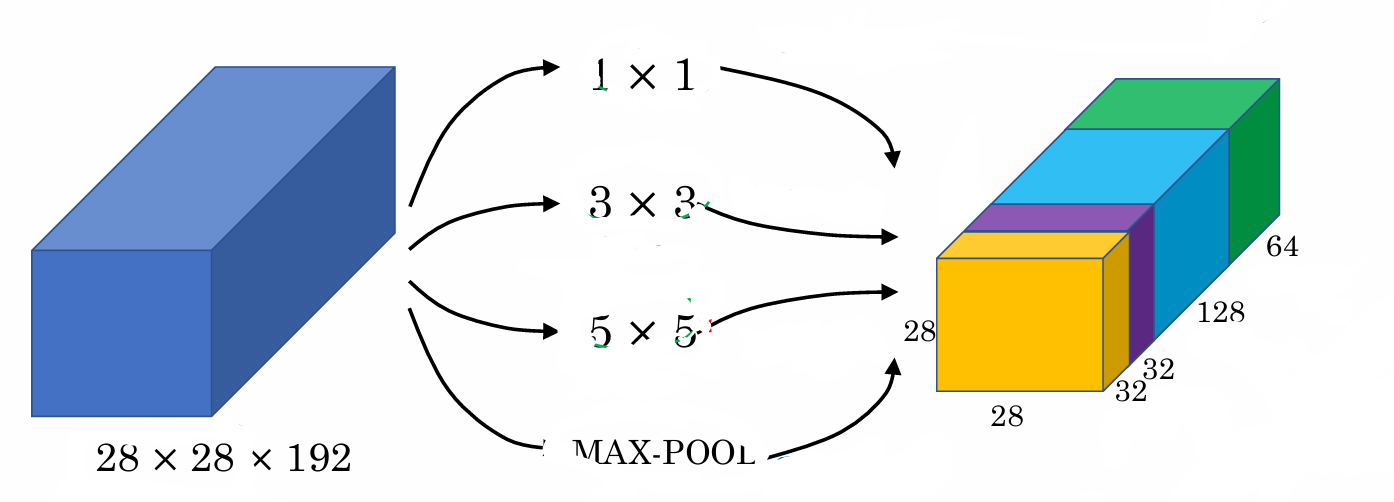
The Hebbian principle is a simple rule for how neurons in the brain learn. It can be summed up as:

"Neurons that fire together, wire together."

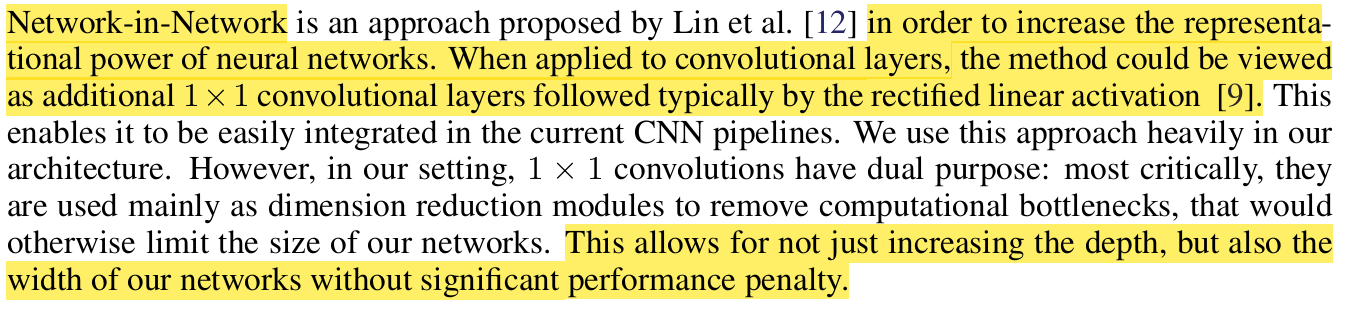
**INTRODUCTION:**







**RELATED WORK:**



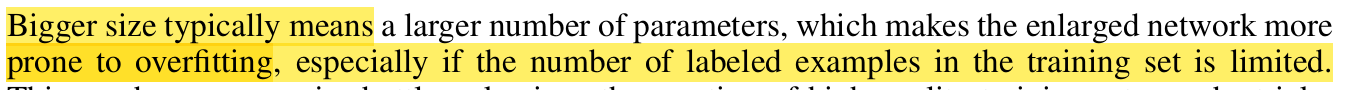
**SUMMARY:**

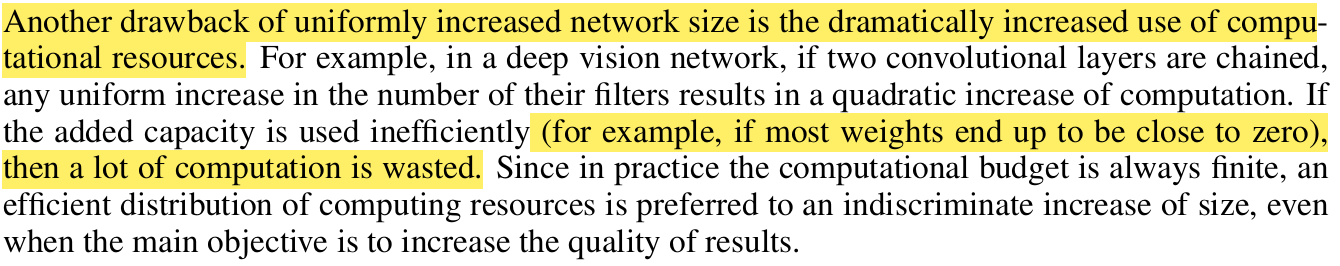
The Network-in-Network approach improves the representational power of neural networks by incorporating 1×1 convolutional layers followed by activation functions. These layers serve a dual purpose:

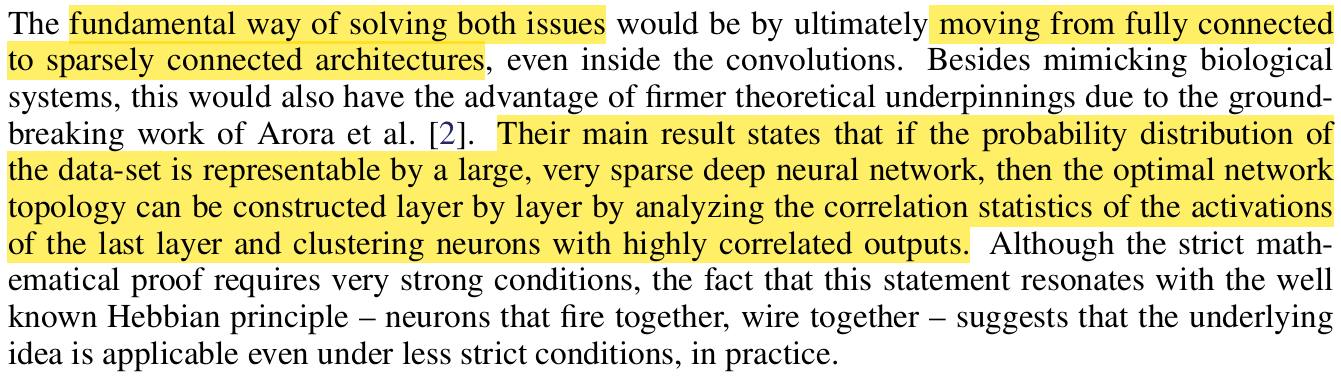
1. Acting as additional non-linear transformations to enhance feature extraction.
2. Functioning as dimension reduction modules, reducing computational bottlenecks.

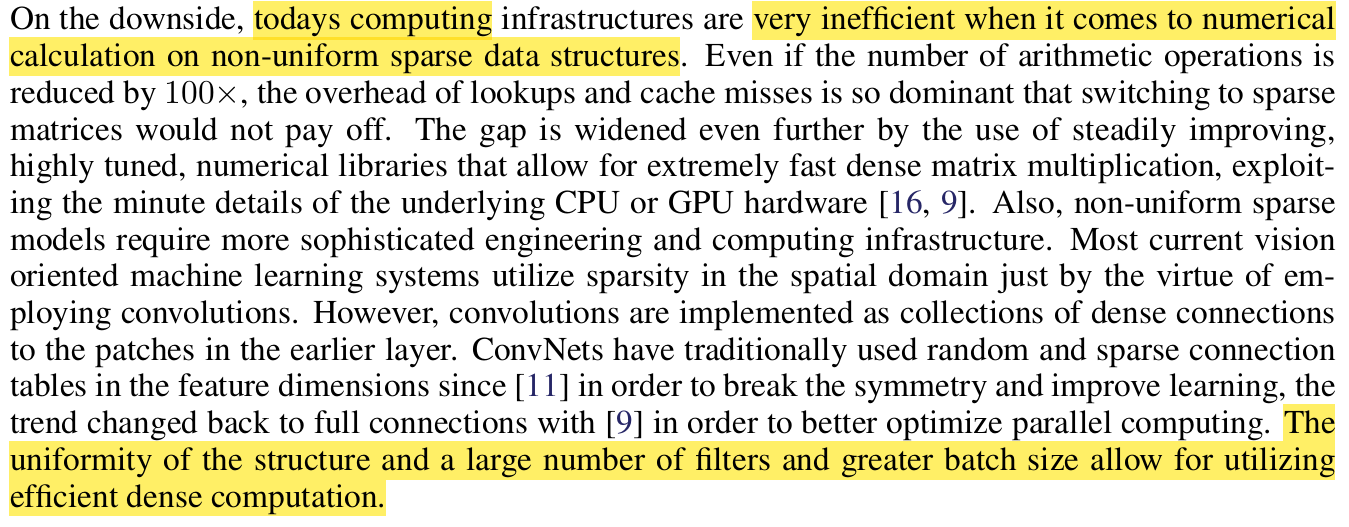
This method enables networks to efficiently increase both depth and width without a significant performance penalty, making it easily integrable into CNN architectures.

**MOTIVATION:**









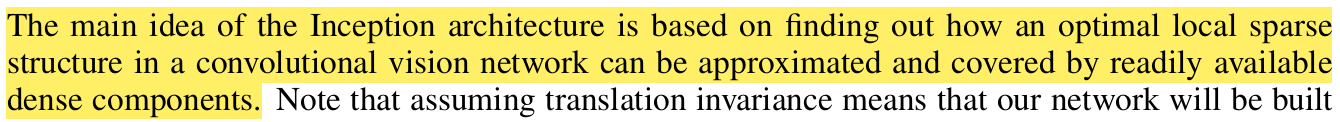
**SUMMARY:**

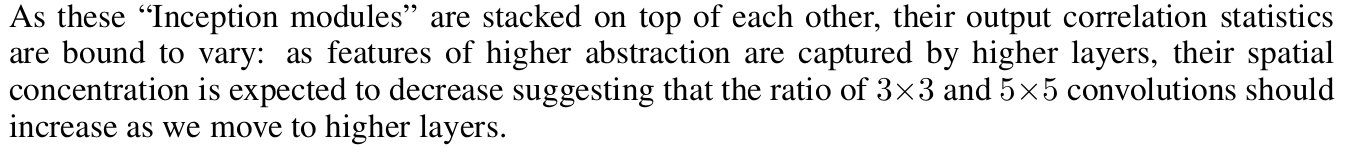
Increasing a neural network’s size (depth and width) improves performance but leads to overfitting and high computational costs. Larger models require extensive labelled data, which is expensive, and increasing parameters leads to quadratic growth in computation, often inefficiently utilized.

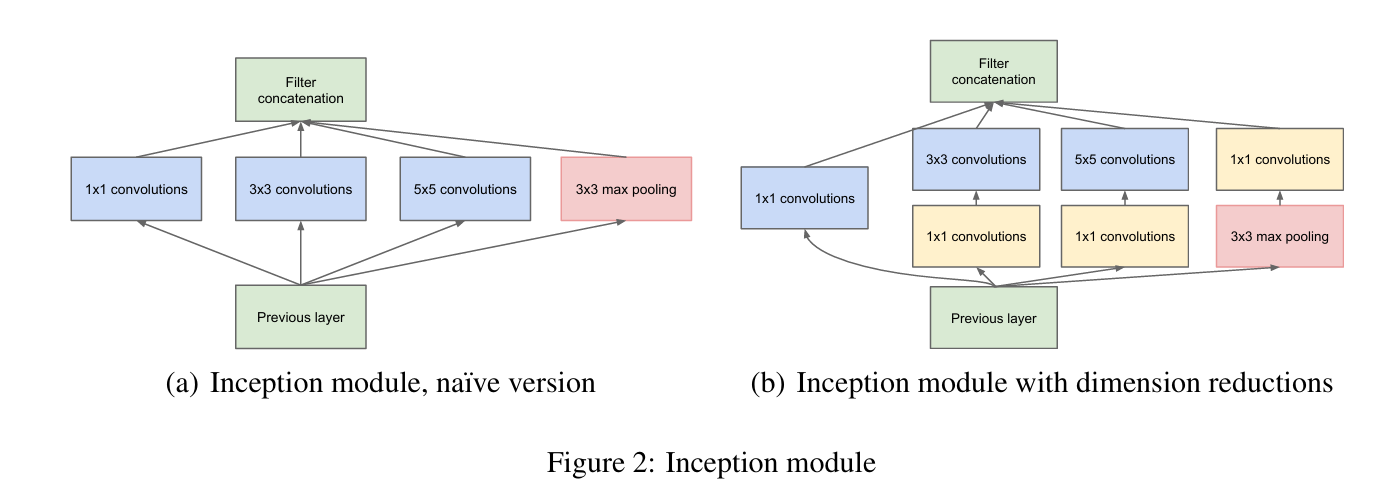
A solution is sparsely connected architectures, which optimize resource distribution while maintaining performance. However, current hardware Favors dense matrix operations, making sparse models less efficient due to memory overhead.

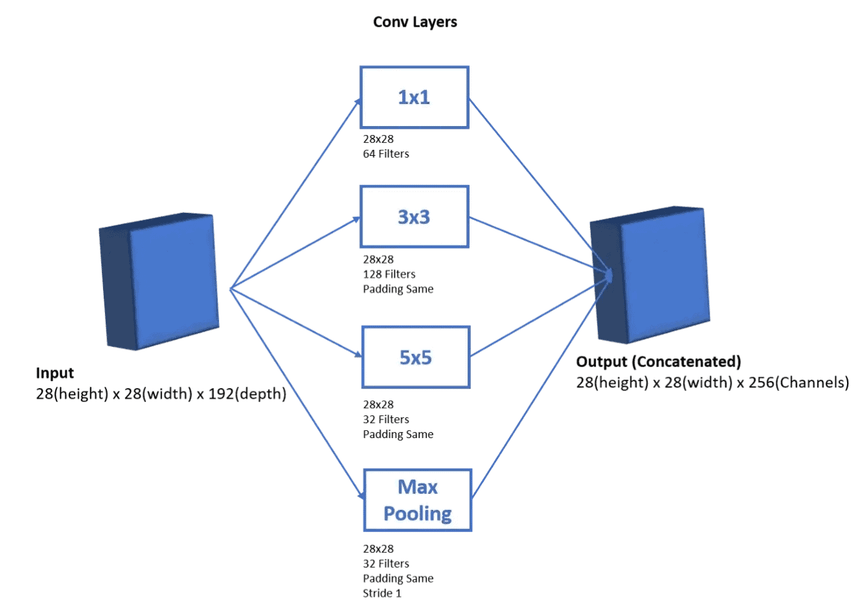
The Inception architecture was developed to approximate sparse structures using dense components, balancing efficiency and accuracy. Through iterative improvements, it became highly effective in object detection and localization. While its success is evident, further research is needed to determine whether its performance is due to its design principles or other factors. Its development encourages continued exploration of efficient deep learning architectures.

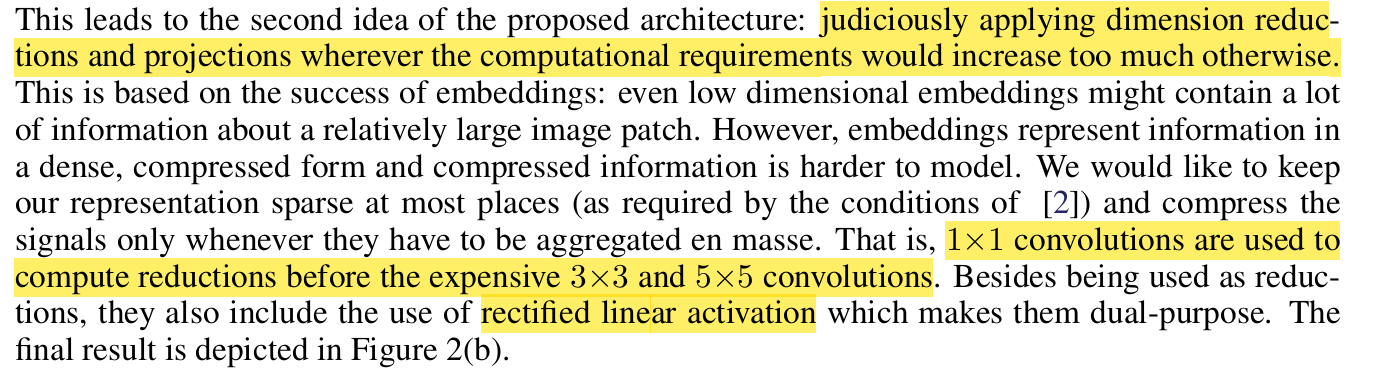
**ARCHITECTURE DETAILS:**

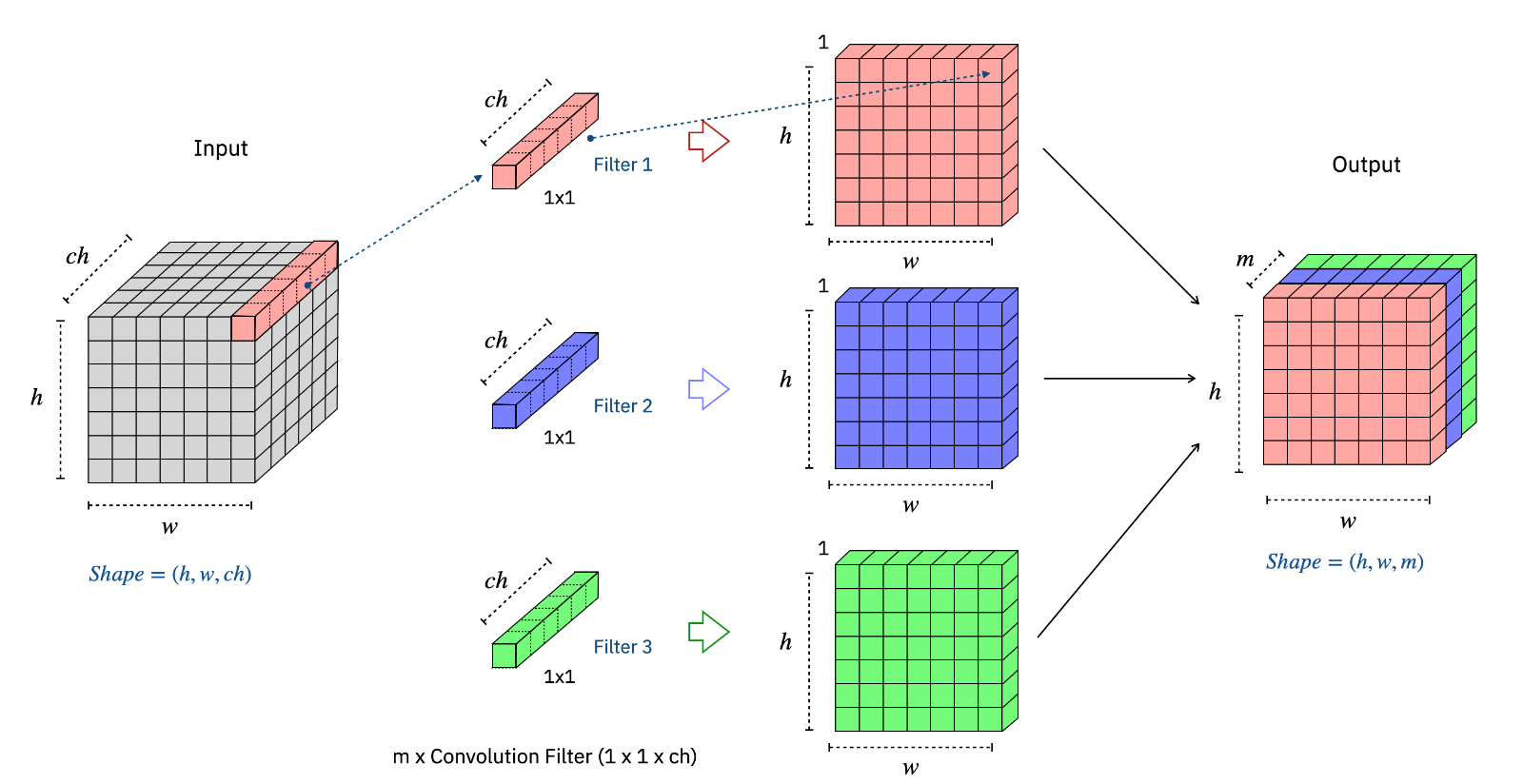


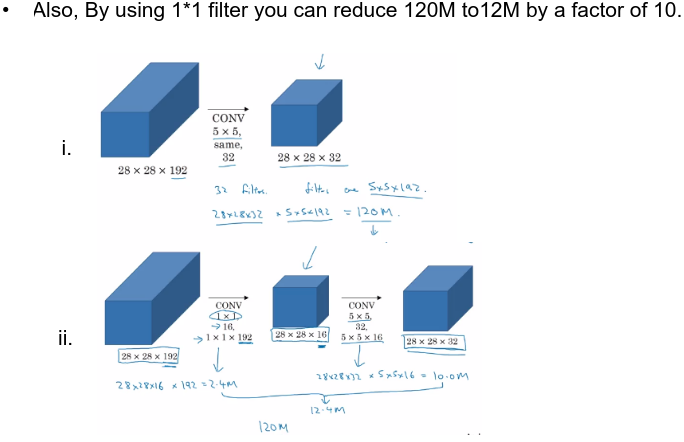












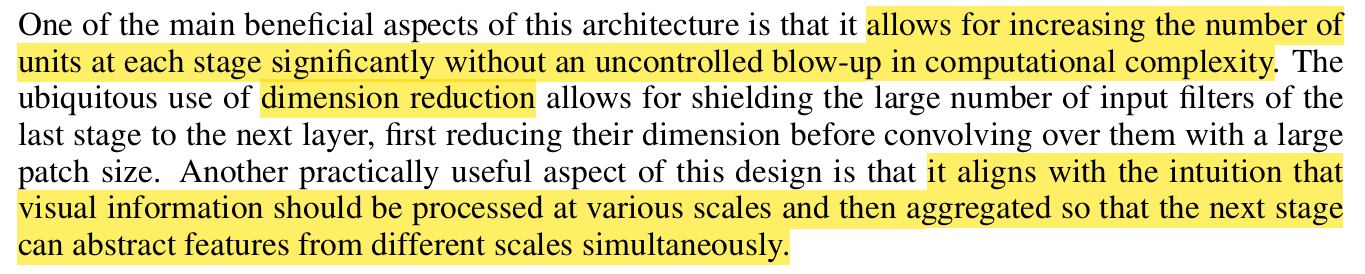
Sparsity:

The problem of sparsity arises in deep networks as the depth increases some neurons become dead wasting computational power and memory.

The Inception model effectively handles sparsity by structuring its architecture in a way that makes computation more efficient while preserving important features. It does this using three main techniques:

1. 1x1 convolutions
2. Multiple orientation of filters applied to the same layer capturing fine and broader details at the same time.
3. Use of auxiliary classifiers to prevent overfitting.

Polyak Averaging is a technique used in optimization to stabilize training and improve generalization by averaging model weights over multiple iterations.



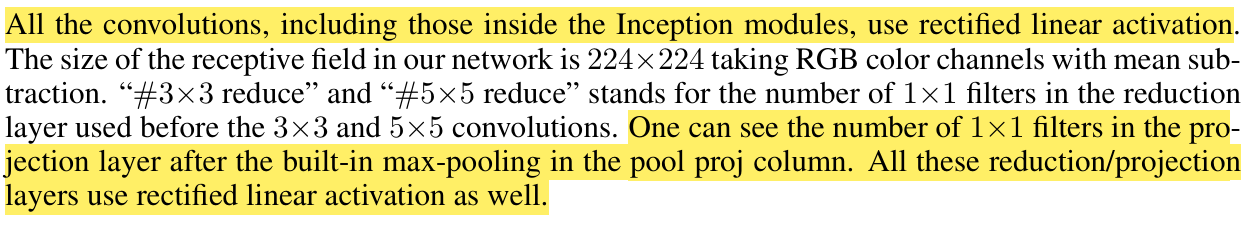
**SUMMARY:**

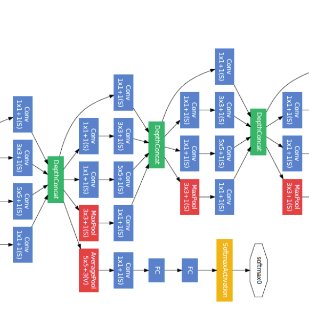
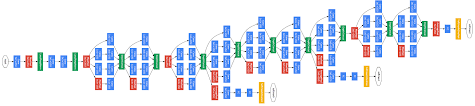
The Inception architecture improves convolutional neural networks by approximating optimal sparse structures with efficient dense components. It follows the principle that correlated activations in one layer can be grouped into multi-scale convolutional operations (1×1, 3×3, 5×5) and pooling layers applied in parallel. The outputs are then concatenated to form the input for the next layer, allowing the network to capture features at different scales simultaneously.

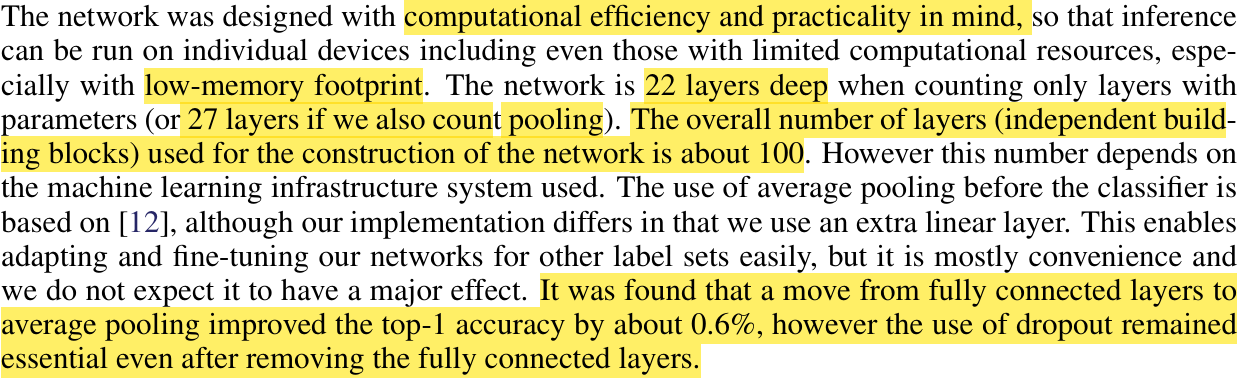
A major challenge is that larger filter sizes (e.g., 5×5) drastically increase computational cost, especially in deeper layers. To mitigate this, 1×1 convolutions are used as dimension reduction bottlenecks before applying larger convolutions. These reduce the number of input channels, making the network significantly more efficient while still preserving useful information.

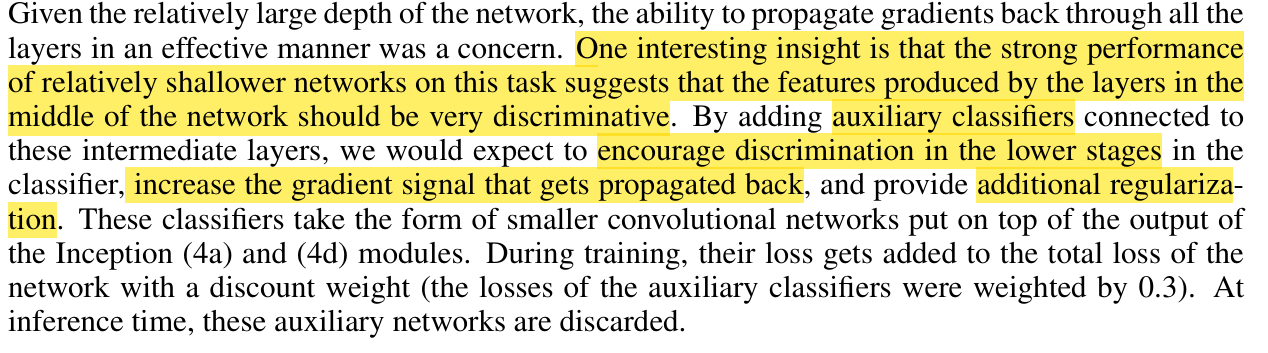
By stacking Inception modules, the architecture allows deep and wide networks without an uncontrolled increase in computational complexity. This approach balances efficiency and performance, making it superior to traditional deep networks. Additionally, it enables controlled scaling of model size and computational requirements, allowing trade-offs between speed and accuracy.

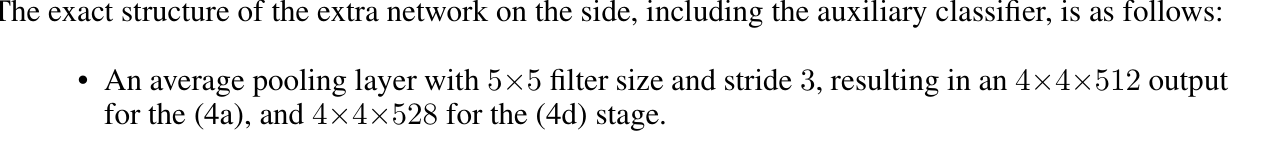
**GOOGLE NET:**

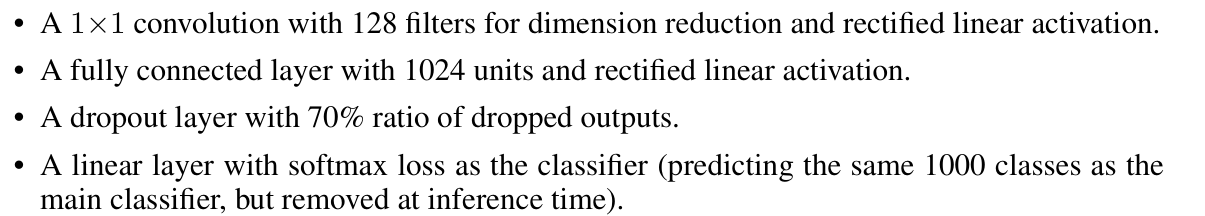










**SUMMARY:**

GoogLeNet, developed for the ILSVRC14 competition, is a deep convolutional network based on the Inception architecture. It was named in homage to LeNet-5 by Yann LeCun. The network focuses on computational efficiency, enabling deep architectures without excessive computational costs.

Key Features of GoogLeNet

1. Deep Inception-based Architecture
   * Composed of 22 parameterized layers (or 27 including pooling layers).
   * Uses multi-scale convolutions (1×1, 3×3, 5×5) and pooling layers in parallel within Inception modules.
   * Allows for efficient feature extraction at multiple scales while maintaining computational efficiency.
2. Use of 1×1 Convolutions for Dimension Reduction
   * Reduces the number of channels before applying 3×3 and 5×5 convolutions, lowering computational costs.
   * Preserves important feature information while making the network more efficient.
3. Auxiliary Classifiers for Regularization and Gradient Flow
   * Two auxiliary classifiers are added at intermediate layers (Inception 4a & 4d).
   * Each auxiliary classifier consists of:
     + 5×5 average pooling
     + 1×1 convolution (128 filters)
     + Fully connected layer (1024 units)
     + Dropout (70%)
     + Softmax classifier (removed at inference)
   * Helps with gradient propagation and regularization, reducing overfitting.
4. Transition from Fully Connected Layers to Global Average Pooling
   * Instead of using fully connected layers, global average pooling (GAP) is applied before the final classification layer.
   * This change improves top-1 accuracy by ~0.6% and reduces the number of parameters, making the network more efficient.
5. Computational Efficiency and Scalability
   * Despite being deep and wide, GoogLeNet is designed to run on resource-limited devices.
   * The ensemble of multiple models (6 out of 7 models used the same topology) helped improve competition results.

**CHALLENGE SETUP AND RESULTS:**

