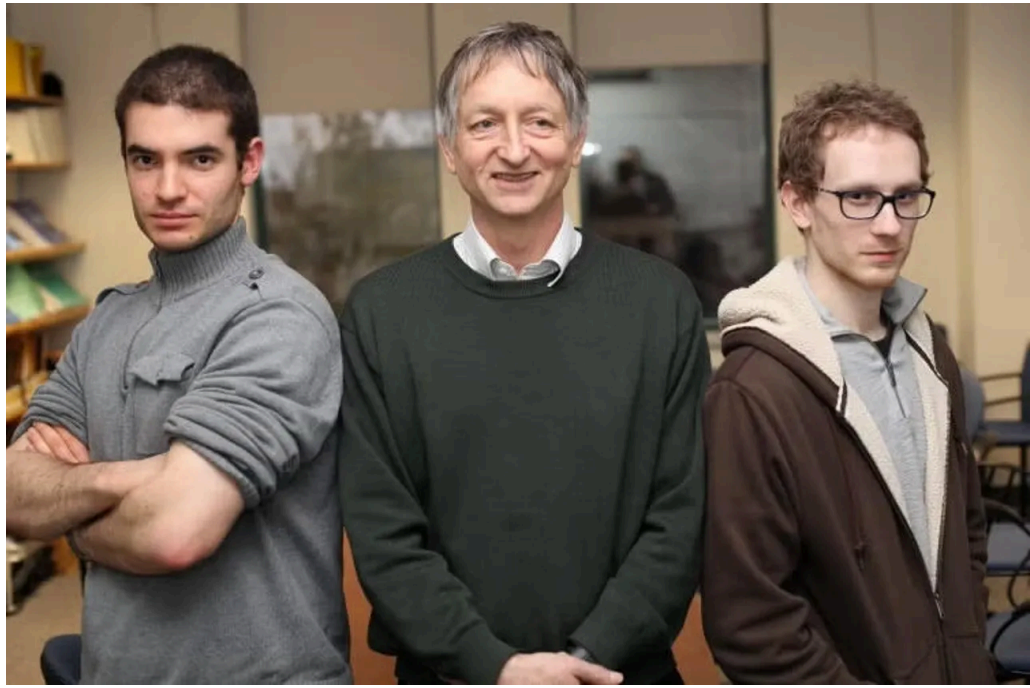
Image Processing 🧠

After AlexNet 🚀

- AlexNet (2012) proved deep ConvNets work for large-scale image recognition (ILSVRC winner).
- **The Big Question:** How do we build *even better* networks? What drives performance 🤔?
- Initial Hypothesis: *Maybe just go *deeper*?
- Two major contenders from ILSVRC 2014 who explored this: **VGG** and **GoogLeNet**.



VGG: Betting on Depth and Simplicity 🏛️

- From *Karen Simonyan & Andrew Zisserman* (Visual Geometry Group - VGG, Oxford)
- **Core Question:** How does network *depth* impact performance?
- **Strategy:** Isolate the effect of depth by keeping everything else extremely *simple* and *uniform*.
- Challenged the notion (in 2014) that very deep networks were too difficult to train. VGG's success paved the way.

VGG Architecture

The Uniform 3x3 Strategy 🧱

- **Radical Simplicity:** Exclusively uses *3x3 filters* (stride 1, padding 1) and *2x2 Max Pooling* (stride 2) for downsampling.

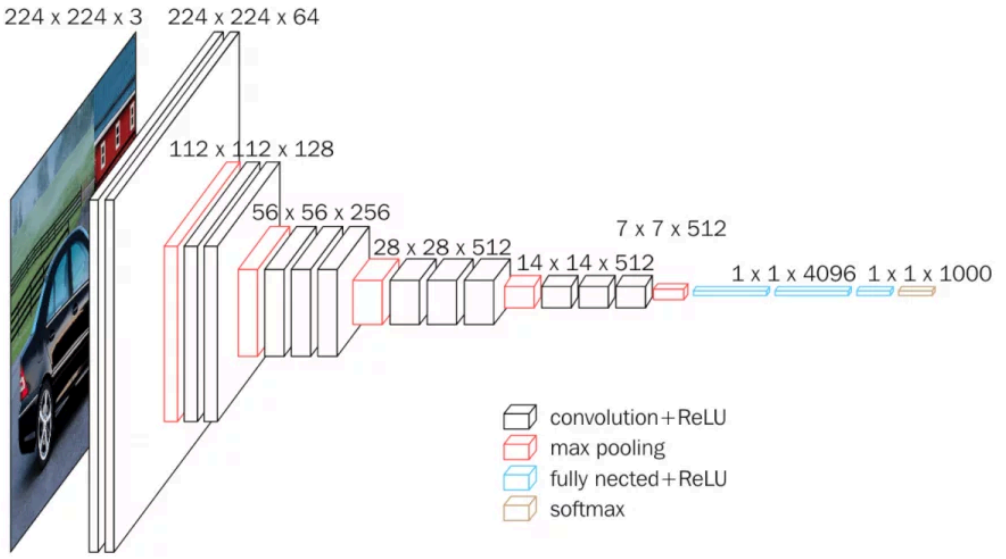
Why only 3x3 filters?

1. Smallest size capturing spatial context (up/down, left/right, center).
2. **Stacking Benefit:**:

- `Stack of two 3x3` -> 5x5 receptive field
- `Stack of three 3x3` -> 7x7 receptive field

A Deep, Uniform Stack 🏗️

- **Input:** 224x224x3 image
- **Structure:** Repeating blocks of 3x3 convs + MaxPool.
- **Channel Doubling:** `64 -> 128 -> 256 -> 512 -> 512`
- **Spatial Reduction:** `224 -> 112 -> 56 -> 28 -> 14 -> 7` (Creates **"pyramid structure"**)
- **Ends with Large Fully Connected Layers:** (e.g., 4096, 4096, 1000 for ImageNet)



🤔 Thinking Point: Design Trade-offs

🤔 Question

VGG uses *only* 3x3 filters for simplicity and to study depth.

Question: What might be the potential *disadvantages* of strictly using only one small filter size throughout such a deep network?

(Hint: Think about features at different scales in an image).

💡 Answer

💡 Answer: Using only 3x3 filters in VGG has several disadvantages:

1. **No Multi-Scale Processing**: VGG can't process different scales simultaneously within a layer.
2. **Computational Depth**: Requires many layers to achieve large receptive fields, increasing computational cost.

The uniform approach, while elegant, isn't optimal for capturing diverse feature scales in natural images compared to architectures with explicit multi-scale processing.

VGG: Results ✅

- **Success:** Excellent ILSVRC 2014 performance (Runner-up Class., Winner Localization).
- **Confirmed:** Depth is crucial.
- **Legacy:** Simple design, strong baseline, very popular for transfer learning.

Other Enhancements

- **Key Technique: VGG-style Data Augmentation:**
  - Resized images to *multiple scales* (e.g., 256px, 384px heights) *before* random 224x224 cropping.
  - Learned features robust to scale variations.

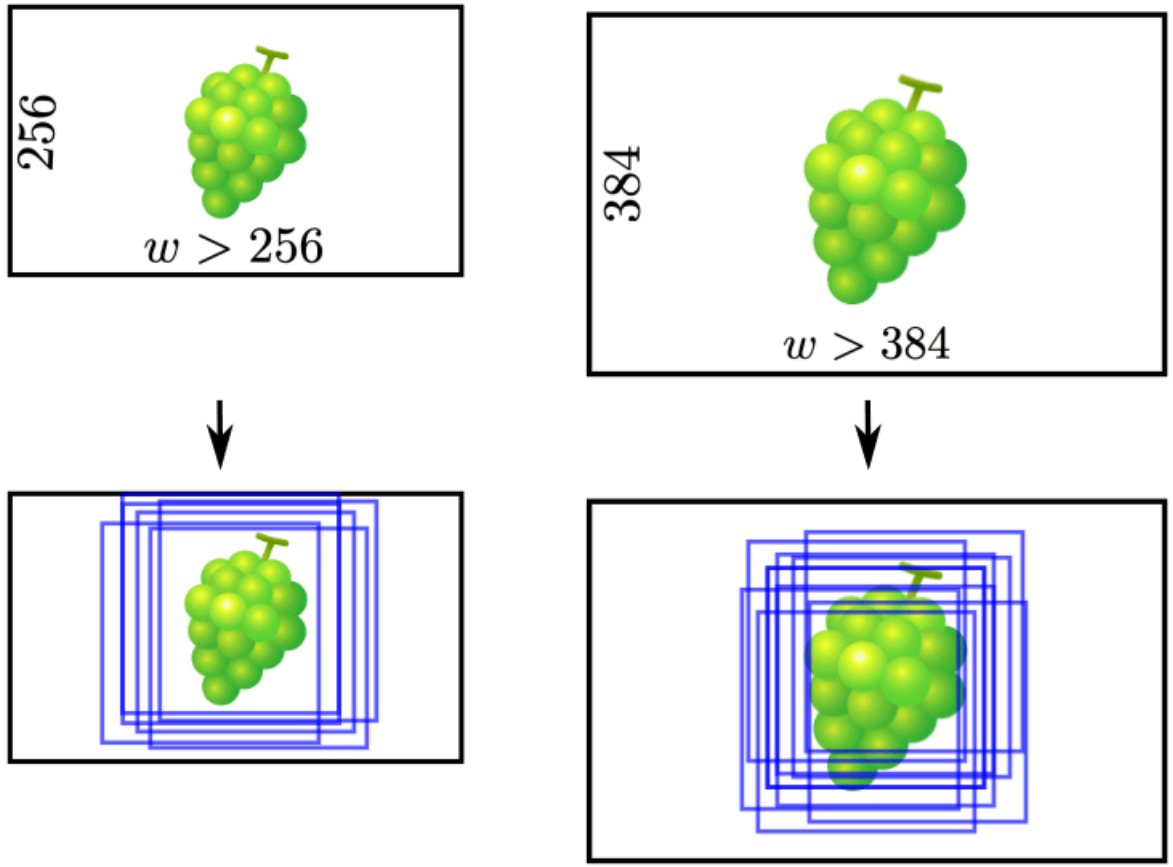


Image: cvml-expertguide.net

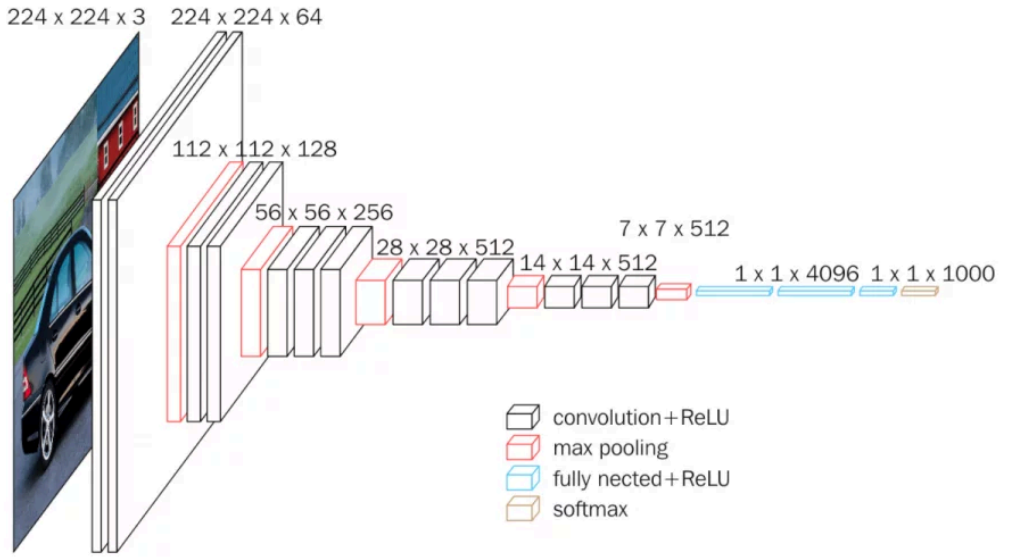
The VGG Catch: Elephant in the Room 🐘

- **Major Drawback:** Very computationally expensive.
- **~140 Million Parameters (VGG16):**
  - The vast **majority (~102 Million!)**.
  - This issue was later addressed.

🤔 Question

VGG16 has ~140 million parameters. We saw that stacked 3x3 convolutions are relatively parameter-efficient compared to larger filters.

Question: Looking at the VGG architecture again, where do you hypothesize the *vast majority* of these parameters are located?



GoogLeNet: "Going Deeper with Convolutions!"

- From Google (**Szegedy et al.**, 2014). *inner ILSVRC 2014 Classification*.
- Going Deeper with Convolutions
- **Core Idea:** Don't just go deeper, go *smarter*. Design a better, multi-scale building block – the **Inception Module**.
- **Analogy:** Like viewing a painting up close (details) and from at distance (composition) *simultaneously*.

Key Idea: Multi-Scale Processing 🔭🔬

- **Strategy (Naive):** Apply multiple filter sizes (**1x1, 3x3, 5x5**) *in parallel* on the same input. Also include parallel *Max Pooling*.
- **Goal:** Capture fine details (1x1) and broader context (3x3, 5x5) simultaneously.
- **Output:** Concatenate the channels from all branches.
- **This is computationally expensive!** 🤔

🤔 Question

The "naive" Inception module applies 1x1, 3x3, 5x5 filters (and pooling) in parallel and concatenates the outputs.

Question: Imagine stacking several of these naive modules. Consider the number of *output channels* after concatenation at each layer. What is the major computational problem that arises?

(Hint: Think about the input channels for the next module).

💡 Answer

The main problem is **Channel Explosion**. Concatenating outputs from all parallel branches creates many more output channels than inputs. When stacked, this makes 3x3 and 5x5 convolutions computationally intractable due to the rapidly growing feature dimensions.

Breakthrough: 1x1 Convolutions

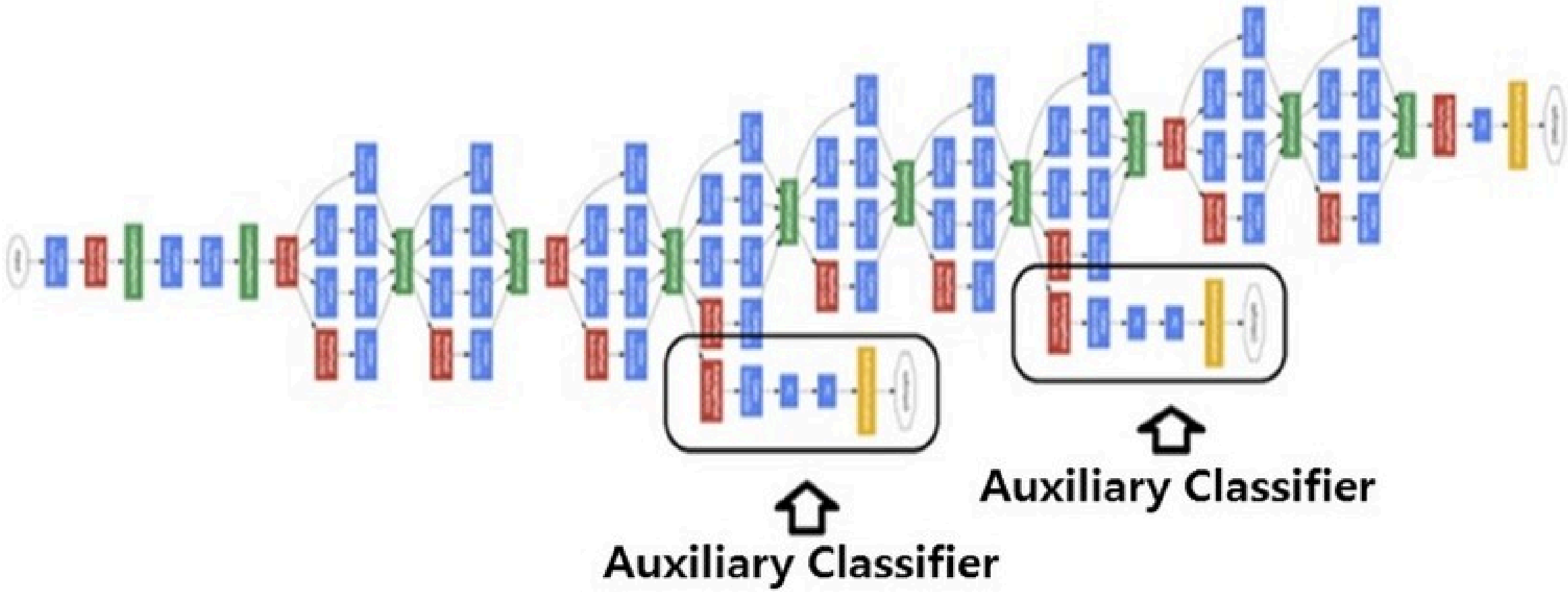
- Use **1x1 Convolutions *before*** the convolutions.
  - "Cross Channel Down sampling" by Min Lin et al. 2013
- **Dimensionality Reduction:** Reduces channel depth fed into larger filters.
  - `Input (H x W x C_in) -> 1x1 Conv (K filters) -> Output (H x W x K)` where `K < C_in`.
- **Approximating Sparsity:** Efficiently processes info across channels (high correlation) before spatial convs (lower correlation).
- **Parameter Reduction:** Saves computation significantly vs. direct 3x3/5x5 on full convolution.

The Final Inception Module v1 ✨

Combines multi-scale processing with 1x1 bottlenecks for efficiency:

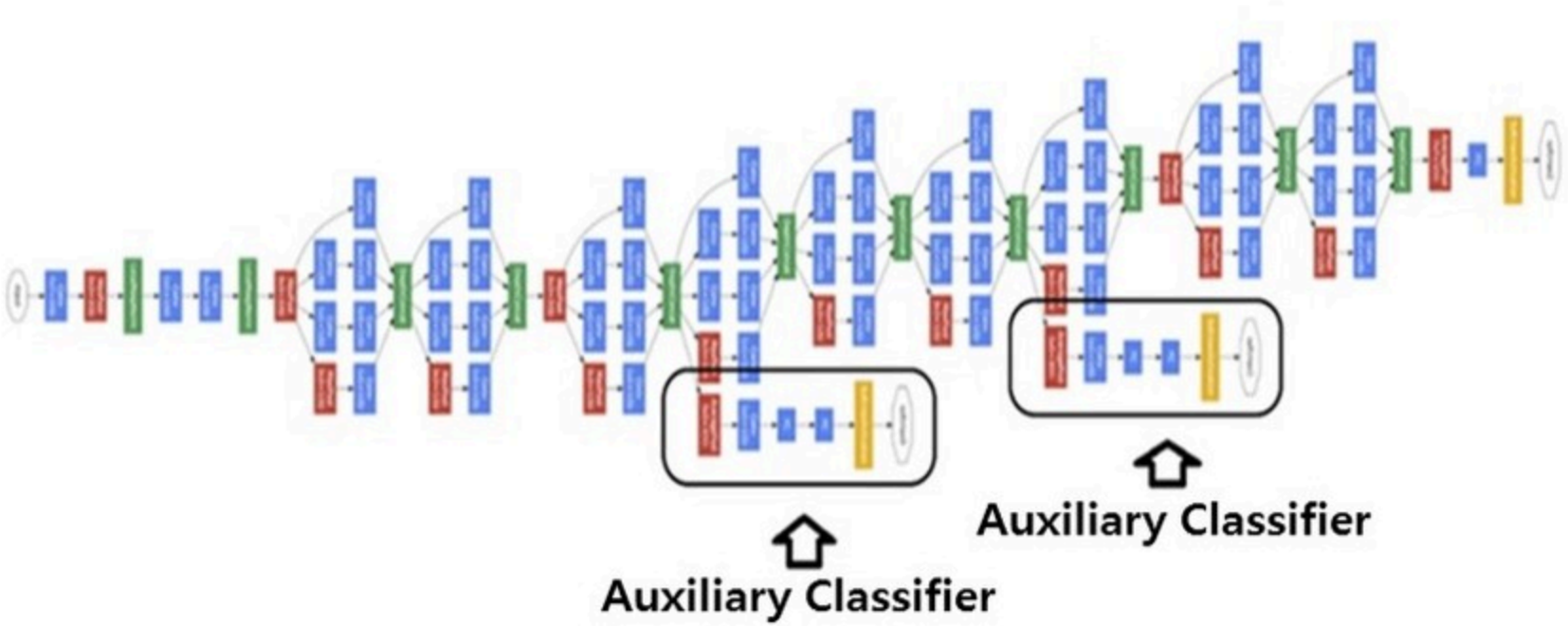
1. **1x1 Conv path** (Direct)
2. **1x1 Conv (reduce) -> 3x3 Conv path**
3. **1x1 Conv (reduce) -> 5x5 Conv path**
4. **3x3 Max Pool -> 1x1 Conv path**
   (Projection after pooling)

Output: Concatenate feature maps from all 4 branches along the channel dimension.


Auxiliary Classifier
Auxiliary Classifier

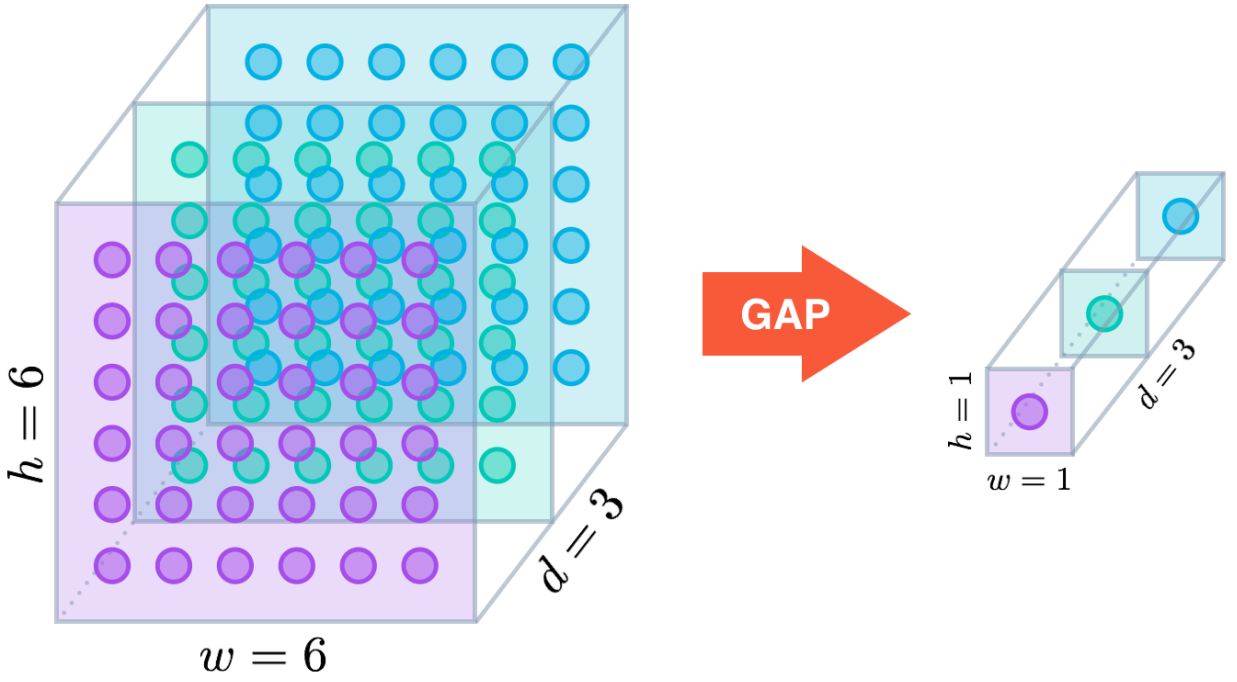
Auxiliary Classifiers

- Small classifiers added partway through during *training*.
- **Purpose:** Combat vanishing gradients by injecting loss/gradient deeper.
- Removed during inference.



Global Average Pooling (GAP)

- Replaces heavy Fully Connected layers at the end (Lin et al., 2013).
- Averages each final feature map spatially to a single value (`H x W x C_out -> 1 x 1 x C_out`).
- Drastically cuts parameters & overfitting risk (compare to VGG's FC layers!).



🤔 Question

GoogLeNet used auxiliary classifiers to help gradients reach early layers during training, addressing the "vanishing gradient" problem in deep networks.

Question: Can you think of *other* techniques (that came later) designed to solve the same problem of training very deep networks effectively?

(Hint: Think about how information could bypass layers or blocks).

GoogLeNet: Winning Efficiently 🏆🍃

- **Result:** Won ILSVRC 2014 Classification task.
- **Remarkable Efficiency:**
  - Only **~7 Million parameters** (~20x fewer than VGG16!).
  - Faster inference, less memory required.
- **Key Message:** Proved that sophisticated architectural design could achieve state-of-the-art results more efficiently than brute-force depth alone.

VGG vs. GoogLeNet: Head-to-Head 🥊

Feature	VGG16	GoogLeNet (Inception v1)
Core Idea	Depth via **Uniformity**	Efficiency via **Multi-Scale Blocks**
Key Innovation	Stacking simple **3x3 convs**	**Inception Module**, 1x1 Bottlenecks, GAP
Parameters	**~140 Million**	**~7 Million**
Main Bottleneck	Heavy **Fully Connected** Layers	Module Design Complexity
Strengths	Simplicity, Good for Transfer Learning	High Efficiency, State-of-the-Art Perf.
Weaknesses	Very Heavy, Slow Inference	More Complex Architecture
ILSVRC '14 Class.	Runner-Up	**Winner**

Legacy and Evolution 📈

- **VGG Legacy:**
  - Confirmed value of **depth**.
  - Became a standard **baseline** and feature extractor due to **simplicity**.
- **GoogLeNet Legacy:**
  - Showcased power of **efficient architectural design**.
  - Introduced influential concepts: **Inception module, 1x1 bottlenecks, Global Average Pooling**.
- **Evolution of Inception:** The core idea inspired further improvements:
  - **Inception v2/v3:** Batch Normalization, Filter Factorization.
  - **Inception v4 / Inception-ResNet:** Added Residual Connections.
  - **Xception:** Pushed towards Depthwise Separable Convolutions ("Extreme Inception").

Conclusion: Key Takeaways takeaways 🔑

- Both VGG and GoogLeNet were landmark achievements in 2014, significantly advancing deep learning for vision.
- **VGG:** Demonstrated the power of **depth** achieved through **architectural simplicity and uniformity**, despite high computational cost.
- **GoogLeNet:** Showed that **clever architectural design** (Inception modules, 1x1 bottlenecks, GAP) could yield state-of-the-art performance with remarkable **computational efficiency**.
- These contrasting philosophies (simplicity vs. engineered efficiency) heavily influenced subsequent network designs.