INTRODUCTION TO

INCEPTION NETWORK

By
Zarni Nway Oo
Phyo Myat Oo
Nu Wai Thet

MOTIVATION BEHIND INCEPTION CNN

The Problem: Bigger ≠ Better

Early CNN designs :

- Made deeper and wider networks to improve accuracy.
- But... More depth & width → more parameters → problems

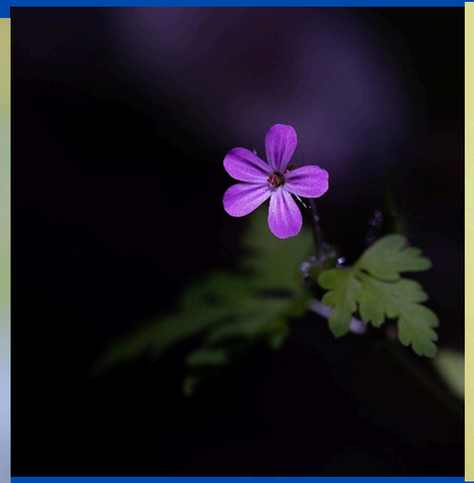
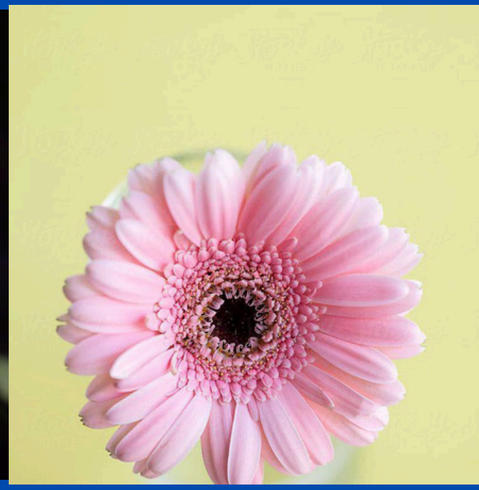
Problem	Why it matters
Overfitting	**Too many parameters + limited data = poor generalization**
Computational Cost	**Large filters like 5×5 or 7×7 are expensive**
Inefficient use of resources	**Many weights may learn "nothing useful”**

The Goal

“Can we design a CNN that is both accurate and efficient, without blindly stacking layers?”

MOTIVATION BEHIND INCEPTION CNN



Challenge

Real-world images contain features at multiple scales — small details and large objects may appear in the same scene.
Should we use small filters (like 1×1), medium filters (3×3), or large ones (5×5)?

Key Idea: multi-scale feature extraction

- Apply multiple filter sizes in parallel
- → 1×1, 3×3, 5×5 convolutions and pooling
- This allows the network to capture diverse features at different spatial resolutions

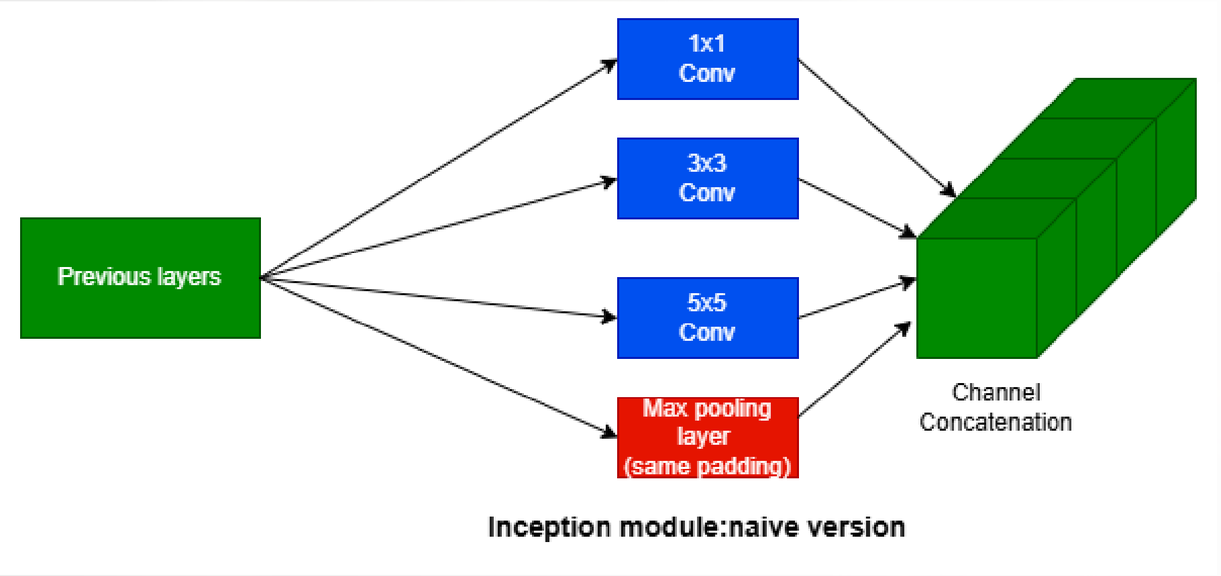
But... Computational Cost Is High!

- Larger filters (like 5×5) are expensive
- Adding many parallel paths can explode the number of computations

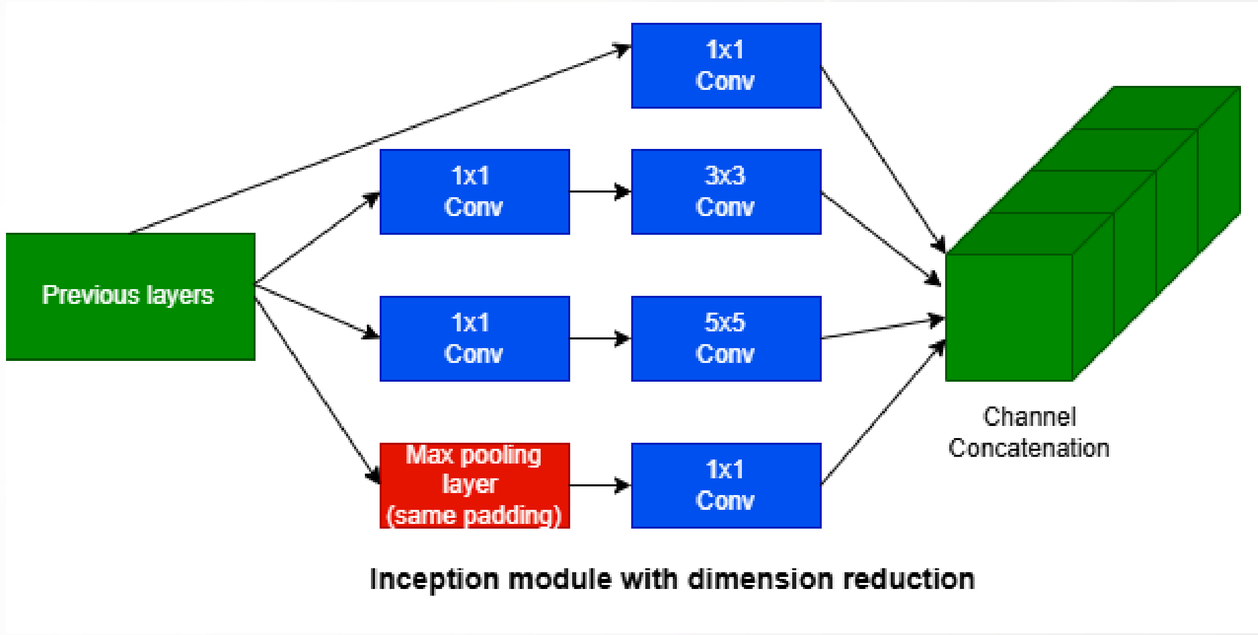
Solution: Inception Architecture

- Use multiple filter sizes in parallel
- Apply 1×1 convolutions to reduce input depth before expensive filters
- Combine all outputs into a richer, multi-scale feature set
- → This is the Inception Module (explained in next slide)

WHAT IS AN INCEPTION MODULE?



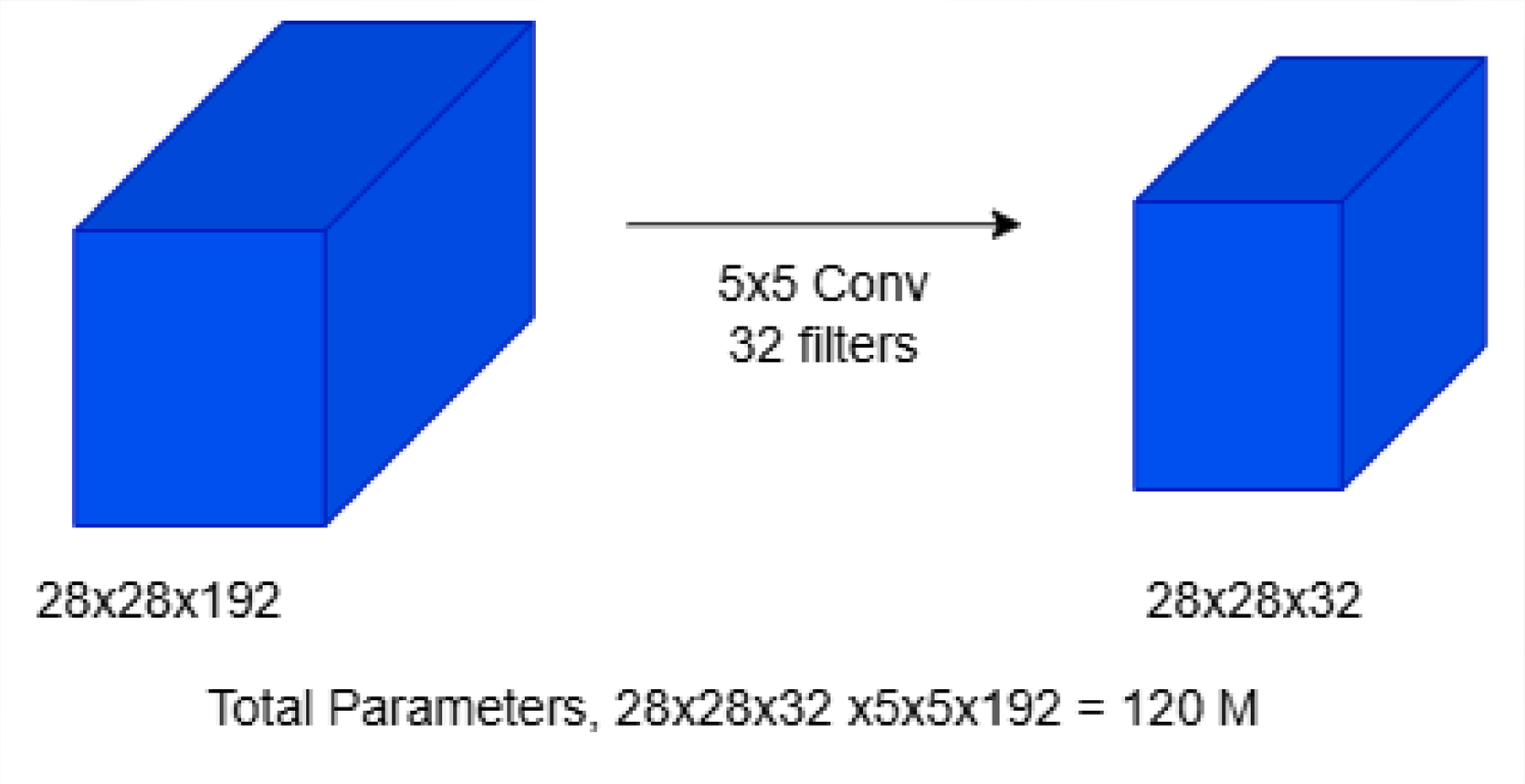
Inception module:naive version



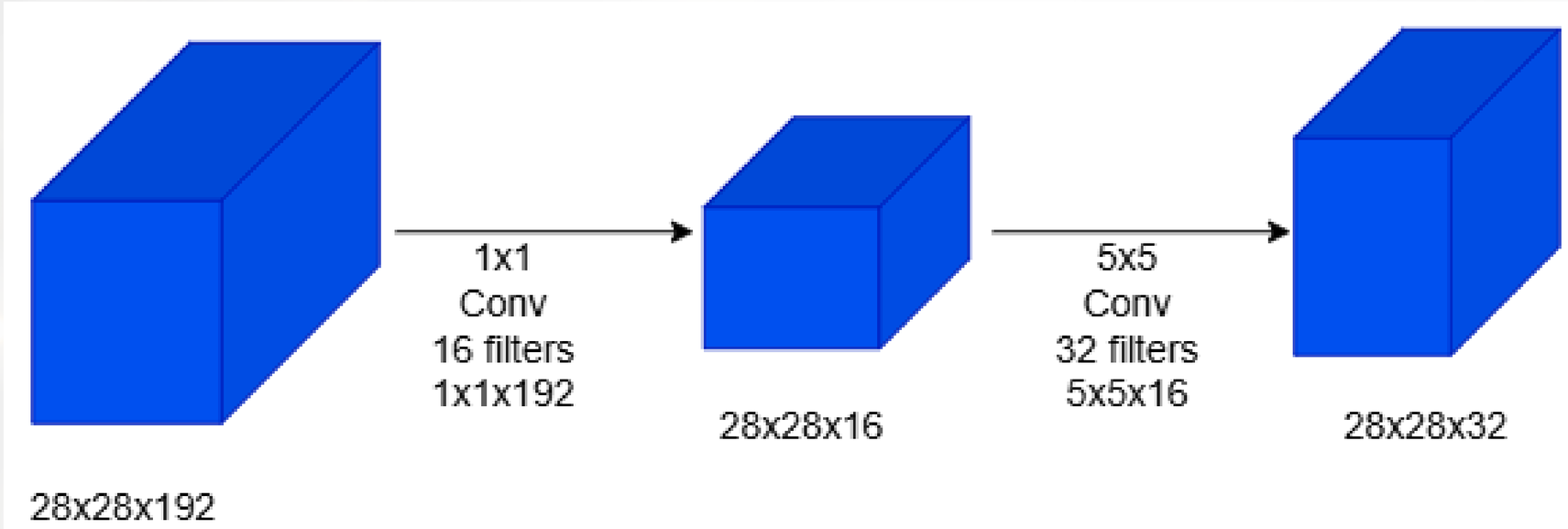
Inception module with dimension reduction

Instead of choosing one filter size, why not apply multiple filter sizes at once, in parallel?
Each kernel captures features at a different scale:
1×1 → fine local detail
3×3 → medium patterns
5×5 → large patterns
Max Pooling → spatial context / robustness
All outputs are concatenated (depth-wise) into one output tensor.
Combining features from different scales → richer and more robust representations.

THE PROBLEM OF COMPUTATIONAL COST



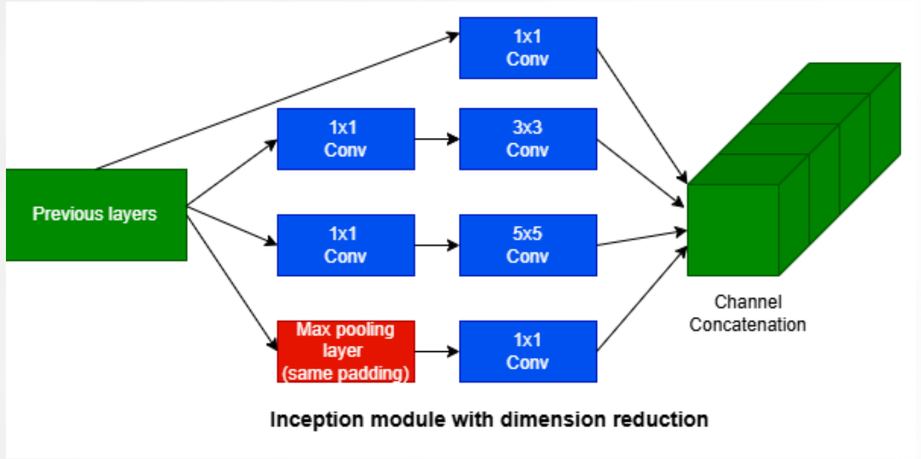
USING 1 X 1 CONVOLUTION



Without 1×1:
Input (192) → 5×5 conv →
Computationally Expensive
With 1×1:
Input (192) → 1×1 conv →
Computationally Much cheaper

- Parallel Processing = multiple perspectives of the same image region.
- 1×1 Conv acts as:
  - Dimension reduction before expensive 3×3 and 5×5 convolutions
  - Non-linearity injector (ReLU after 1×1 conv)

BENEFITS OF **INCEPTION MODULE**



Inception module with dimension reduction

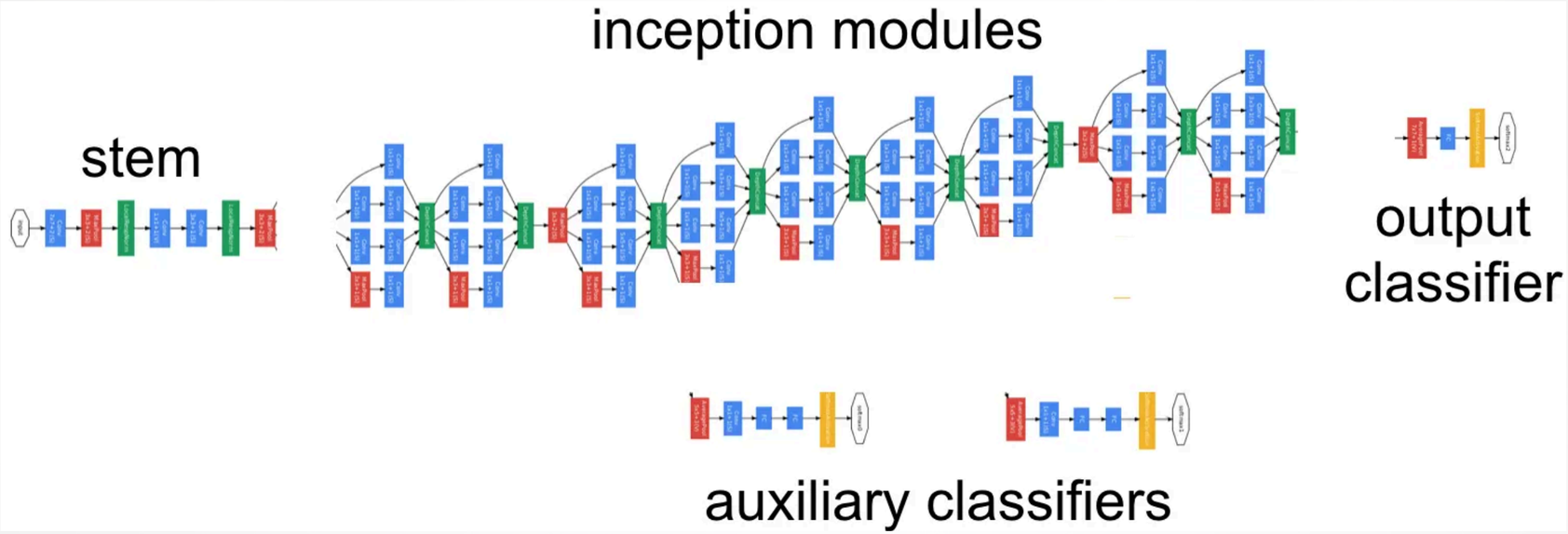
Multiple kernel sizes **1** Capture multi-scale features

1x1 Conv **2** Reduces computation & adds depth

Parallel paths **3** Richer representation

Flexible design **4** Easy to balance accuracy vs speed

INCEPTION ARCHITECTURE – GOOGLENET



Overall Structure
GoogLeNet = Stacked Inception modules with some traditional conv & pooling layers.

- GoogLeNet is a deep convolutional neural network built using 9 Inception modules.
- Total Layers: 22
- Key Feature: Uses stacked Inception modules to increase depth and width without computational blow-up
- Introduces global average pooling before the final fully connected layer, reducing the number of parameters compared to traditional deep FC layers.

AUXILIARY CLASSIFIERS IN GOOGLENET

Deep networks (like GoogLeNet with 22 layers) can suffer from:

- Vanishing gradients during backpropagation
- Slow or unstable training

Solution: Add Small Classifiers in the Middle

- Acts like "intermediate supervision"
- Helps improve gradient flow to lower layers
- Adds regularization during training

What Are Auxiliary Classifiers?

An auxiliary classifier is like adding a mini neural network in the middle of the big one. It:

- Takes the output of an intermediate layer
- Applies some layers: pooling → 1x1 conv → FC → softmax.
- Tries to predict the final class — just like the main classifier.

Benefits:

🌊 Better gradient flow : Helps train earlier layers
🧩 Regularization: Reduces overfitting
⚡ Faster convergence: Training stabilizes sooner

Think of them as “training helpers,” not part of the final model

TRAINING METHODOLOGY & RESULTS (ILSVRC 2014)

Training Setup

- Framework: DistBelief (CPU-based distributed training)
- Optimizer: Asynchronous SGD + 0.9 momentum
- Learning rate: Decreased 4% every 8 epochs
- Final model: Polyak Averaging (smooths model parameters)

Data Augmentation Techniques

- Multi-scale random crops (8%–100% of image area)
- Random aspect ratios (from ¾ to 4/3)
- Photometric distortions (color jittering)
- Random interpolation (bilinear, area, nearest, cubic)

Testing Strategy

- Ensemble of 7 GoogLeNet models
- Aggressive multi-crop testing (144 crops/image)
- Simple averaging of softmax scores across models & crops

🏆 Results: ILSVRC 2014

Metric	Value
Top-5 Error	6.67% (1st place)
Training Images	~1.2 million
No external data used	✅

EVOLUTION OF INCEPTION MODELS (V1 → INCEPTION-RESNET-V2)

Model	Key Innovations	Architectural Highlights	Training Improvements	Impact & Benefits
Inception-V1 (GoogLeNet, 2014)	Introduced the Inception Module: parallel convolutions (1×1, 3×3, 5×5) + pooling	Used 1×1 convolutions to reduce depth before expensive ops; included auxiliary classifiers	Used Dropout, global average pooling instead of FC layers	Reduced parameters vs. VGG, efficient computation, top-5 ImageNet accuracy: 93.3%
Inception-V2 (2015)	Factorized convolutions: replaced 5×5 with two 3×3 layers; introduced asymmetric convolutions (1×n + n×1)	Removed 5×5 filters entirely; widened inception modules for better capacity	Added Batch Normalization to improve convergence and stability	Lower computation cost, faster training, better regularization
Inception-V3 (2015)	Factorized 7×7 convolutions into 1×7 + 7×1; improved module design	Deeper and more efficient modules with extensive factorization	Added Label Smoothing, switched to RMSprop optimizer	Significant accuracy gains; top-1 ImageNet error reduced to ~22%
Inception-V4 (2016)	Modular redesign: separated into Inception-A/B/C and Reduction blocks	More structured and deeper architecture	Standardized modules, improved network clarity	State-of-the-art performance with more depth, better suited for large-scale training
Inception-ResNet-V2 (2016)	Merged Inception blocks with Residual Connections from ResNet	Used skip connections inside inception modules	Faster training, deeper networks with less vanishing gradient	Combined benefits of ResNet + Inception; one of the best models on ImageNet

CONCLUSION & KEY TAKEAWAYS – INCEPTION (GOOGLENET)

🔑 Why Inception Was a Breakthrough

- Introduced multi-scale processing within a single module
- Used 1×1 convolutions to reduce cost without losing performance
- Achieved depth & width efficiently without computational explosion

📊 Performance Highlights

- 1st place in ILSVRC 2014 (Classification: 6.67% Top-5 Error)
- 1st place in Detection (43.9% mAP) — even without bounding box refinement

🧠 Smart Design Choices

- Auxiliary classifiers improve gradient flow + regularization
- Ensembles + aggressive crop testing boost final accuracy
- Carefully balanced depth, width, and compute cost

🌱 Legacy & Impact

- Inspired later models (e.g., Inception v2, v3, v4)
- Sparked a shift toward modular, efficient architectures
- Proof that dense approximations of sparse structures can scale

💡 Inception showed it's not just about going deeper — it's about going smarter.

THE INSPIRATION BEHIND THE NAME: WE NEED TO GO DEEPER

The meme “We need to go deeper” comes from the movie Inception — specifically from the character Dom Cobb (Leonardo DiCaprio) — and became a popular internet meme.

Your paragraph text

🎓 “Yes, **the original paper literally references the movie and the meme**: ‘We need to go deeper.’”
📚 Meme on KnowYourMeme: https://knowyourmeme.com/memes/we-need-to-go-deeper

THANK YOU