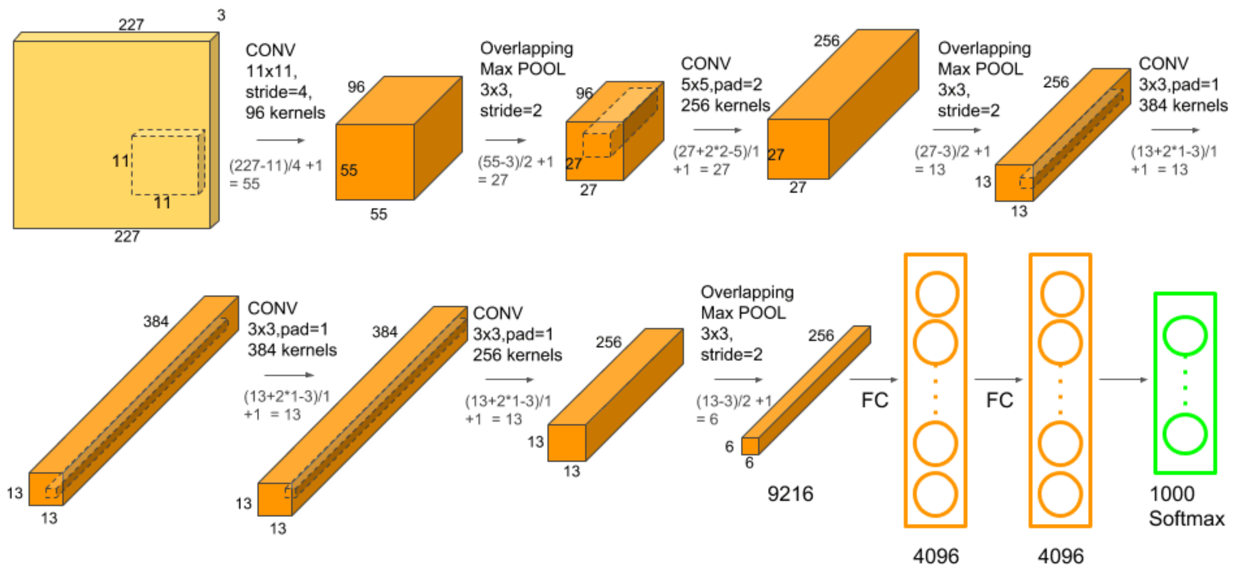
Convolutional Neural Networks (CNNs)

A convolutional neural network (CNN) is a type of deep learning algorithm primarily used for image recognition and processing due to its ability to recognize patterns in images. While CNNs are most commonly applied to visual tasks, they can also be used for other AI applications, including natural language processing and recommendation engines. These networks require large amounts of labeled data for training and must be trained on high-power processors, such as GPUs or NPUs, to achieve efficient and useful results.

What is AlexNet?

Introduction

AlexNet was designed by **Geoffery Hinton**, winner of the 2012 ImageNet competition, and his student **Alex Krizhevsky.** It was also after that year that more and deeper neural networks were proposed, such as the excellent VGG, GoogleLeNet. Its official data model has an accuracy rate of **57.1%** and top 1-5 reaches **80.2%**. This is already quite outstanding for traditional machine learning classification algorithms.



1. Input Layer

- **Input Image Size: 227 × 227 × 3** (RGB Image)
- The image is resized and normalized before being fed into the CNN.

2. Convolutional Layers

Conv Layer 1

- **Filter Size:** 11 × 11
- **Stride:** 4
- **Number of Filters:** 96
- **Output Size:** 55 × 55 × 96
- **Activation:** ReLU
- **Pooling:** 3 × 3 Max Pooling with **stride 2** → Output: **27 × 27 × 96**
- **Purpose:** Extracts low-level features such as edges and textures.

Conv Layer 2

- **Filter Size:** 5 × 5
- **Padding:** 2
- **Number of Filters:** 256
- **Output Size:** 27 × 27 × 256
- **Activation:** ReLU
- **Pooling:** 3 × 3 Max Pooling with **stride 2** → Output: **13 × 13 × 256**
- **Purpose:** Detects more complex patterns like shapes and structures.

Conv Layer 3

- **Filter Size:** 3 × 3
- **Padding:** 1
- **Number of Filters:** 384
- **Output Size:** 13 × 13 × 384
- **Activation:** ReLU
- **Purpose:** Extracts higher-level features.

Conv Layer 4

- **Filter Size:** 3 × 3
- **Padding:** 1
- **Number of Filters:** 384
- **Output Size:** 13 × 13 × 384
- **Activation:** ReLU
- **Purpose:** Further refines deep features.

Conv Layer 5

- **Filter Size:** 3 × 3
- **Padding:** 1
- **Number of Filters:** 256
- **Output Size:** 13 × 13 × 256
- **Activation:** ReLU
- **Pooling:** 3 × 3 Max Pooling with **stride 2** → Output: **6 × 6 × 256**
- **Purpose:** Final feature extraction before classification.

3. Fully Connected (FC) Layers

Flattening the Output

- The final **6 × 6 × 256** output is flattened into a **1D vector** of size **9216**.

Fully Connected Layer 1 (FC1)

- **Neurons:** 4096
- **Activation:** ReLU
- **Dropout:** Applied to reduce overfitting.

Fully Connected Layer 2 (FC2)

- **Neurons:** 4096
- **Activation:** ReLU
- **Dropout:** Applied to improve generalization.

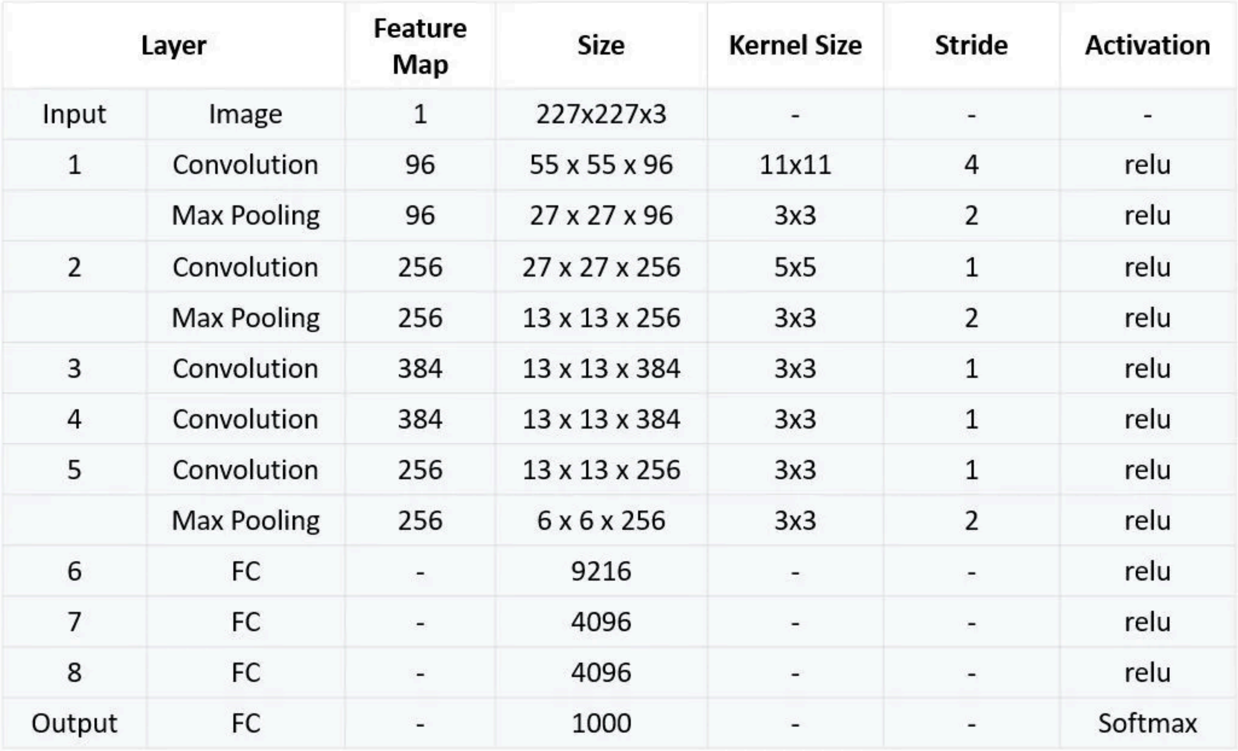
Fully Connected Layer 3 (Output Layer)

- **Neurons:** 1000 (for 1000 ImageNet classes)
- **Activation:** Softmax (for classification).

Performance and Impact

The AlexNet architecture dominated in 2012 by achieving a top-5 error rate of 15.3%, significantly lower than the runner-up's 26.2%.

This large reduction in error rate excited the researchers with the untapped potential of deep neural networks in handling large image datasets. Subsequently, various Deep Learning models were developed later.



Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	227x227x3	-	-	-
1	Convolution	96	55 x 55 x 96	11x11	4	relu
	Max Pooling	96	27 x 27 x 96	3x3	2	relu
2	Convolution	256	27 x 27 x 256	5x5	1	relu
	Max Pooling	256	13 x 13 x 256	3x3	2	relu
3	Convolution	384	13 x 13 x 384	3x3	1	relu
4	Convolution	384	13 x 13 x 384	3x3	1	relu
5	Convolution	256	13 x 13 x 256	3x3	1	relu
	Max Pooling	256	6 x 6 x 256	3x3	2	relu
6	FC	-	9216	-	-	relu
7	FC	-	4096	-	-	relu
8	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Applications of AlexNet

Developers created AlexNet for image classification. However, advances in its architecture and transfer learning (a technique where a model trained on one task is repurposed for a novel related task) opened up a new set of possibilities for AlexNet. Moreover, its convolutional layers form the foundation for object detection models such as Fast R-CNN and Faster R-CNN, and professionals have utilized them in fields like autonomous driving and surveillance.

- **Autism Detection:** Gazal and their team developed a model using transfer learning, for early detection of autism in children. The model was first trained on ImageNet and then the pre-trained model was further trained on their dataset related to autism.

- **Video Classification:** For video classification, researchers have used AlexNet to extract critical features in videos for action recognition and event classification.

- **Agriculture**: AlexNet analyzes images to recognize the health and condition of plants, empowering farmers to take timely measures that improve crop yield and quality. Additionally, researchers have employed AlexNet for plant stress detection and weed and pest identification.

- **Disaster Management**: Rescue teams use the model for disaster assessment and making emergency relief decisions using images from satellites and drones.
- **Medical Images:** Doctors utilize AlexNet to diagnose various medical conditions. For example, they use it to analyze X-rays, MRIs (particularly brain MRIs), and CT scans for disease detection, including various types of cancers and organ-specific illnesses. Additionally, AlexNet assists in diagnosing and monitoring eye diseases by analyzing retinal images.

The success of AlexNet inspired the design and development of various neural network architectures. These include:

- **VGGNet**: It was developed by K. Simonyan and A. Zisserman at Oxford. The VGGNet incorporated ideas from AlexNet and was the next step taken after AlexNet.
- **GoogLeNet (Inception):** This was Introduced by Google, which further developed the architecture of AlexNet.
- **ResNets:** AlexNet started to face a vanishing gradient with deeper networks. To overcome this, ResNet was developed. These networks introduced residual learning, also called skip connections. The skip connection connects activations of lower layers to higher layers by skipping some layers in between.

What is GoogLeNet ?

Introduction

GoogLeNet, developed by Christian Szegedy and his team at Google, revolutionized deep learning by winning the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) with an impressive top-5 error rate of just 6.7%, nearly half the 11.7% error rate of the previous year's winner, ZFNet. This significant improvement demonstrated GoogLeNet's superior accuracy and computational efficiency over earlier architectures like AlexNet and VGG.

Unlike previous models that simply increased network depth, GoogLeNet introduced the Inception module, enabling multi-scale feature extraction without excessive computational overhead. With a total of **22 layers**, it was deeper than all prior CNN architectures. While increasing depth generally improves a model's learning capacity, it also introduces challenges like the vanishing gradient problem. GoogLeNet successfully addressed this issue, setting a new benchmark in object classification and detection while maintaining

Inception Module: Multi-Scale Feature Extraction

The core innovation of GoogLeNet is the Inception module, which processes input at multiple filter sizes simultaneously.Unlike traditional CNN layers that use a single filter size per layer, Inception applies 1×1, 3×3, and 5×5 convolutions in parallel, along with max pooling.The outputs are then concatenated along the depth axis, preserving rich spatial information.

22 Layers Deep but Computationally Efficient

GoogLeNet is a **22-layer deep** CNN, much deeper than AlexNet (8 layers) but computationally more efficient due to its design.Instead of fully connected layers, it uses global average pooling, reducing parameters and preventing overfitting.

1×1 Convolutions for Dimensionality Reduction

1×1 convolutions are used as bottleneck layers before applying expensive 3×3 and 5×5 convolutions, reducing the number of channels and computational cost.

This helps control overfitting by limiting the number of parameters.

Auxiliary Classifiers in GoogLeNet

GoogLeNet includes two auxiliary classifiers during training to combat vanishing gradients and improve gradient flow in deep networks. These are omitted during inference.

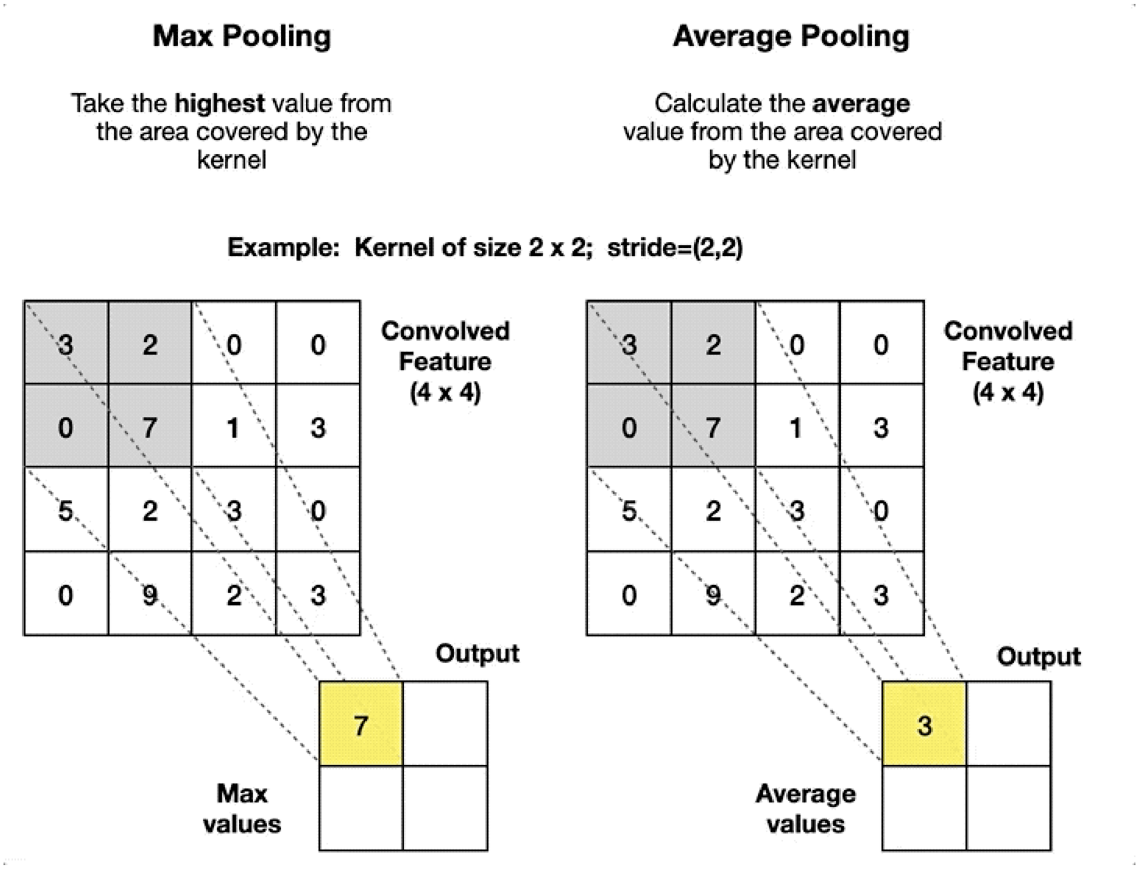
Structure of Each Auxiliary Classifier:

- 5×5 average pooling (stride 3)
- 1×1 convolution (128 filters) for dimension reduction
- Two fully connected layers (1024 units + dropout, final output layer)
- Softmax for prediction probabilities

The auxiliary loss (weighted at 0.3) is added to the total loss, helping stabilize training and improve regularization, reducing overfitting by distributing learning across the network.

Global Average Pooling Instead of Fully Connected Layers

Global Average Pooling is a technique used in CNNs to reduce parameters and prevent overfitting. It is typically placed at the end of the CNN architecture.



Max Pooling

Take the **highest** value from the area covered by the kernel

Average Pooling

Calculate the **average** value from the area covered by the kernel

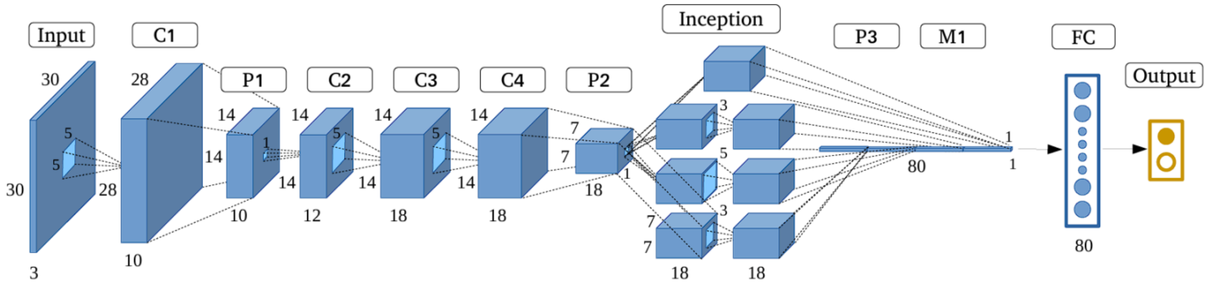
Example: Kernel of size 2 x 2; stride=(2,2)

3	2	0	0
0	7	1	3
5	2	3	0
0	9	2	3

- **Averages each feature map** across its spatial dimensions (Width & Height).
- **Outputs a vector** with a size equal to the number of channels.
- **Summarizes activation** across the feature map, retaining essential information while reducing complexity.

Architecture of GoogLeNet

GoogLeNet consists of **nine Inception modules**, interspersed with **max-pooling layers** and auxiliary classifiers.

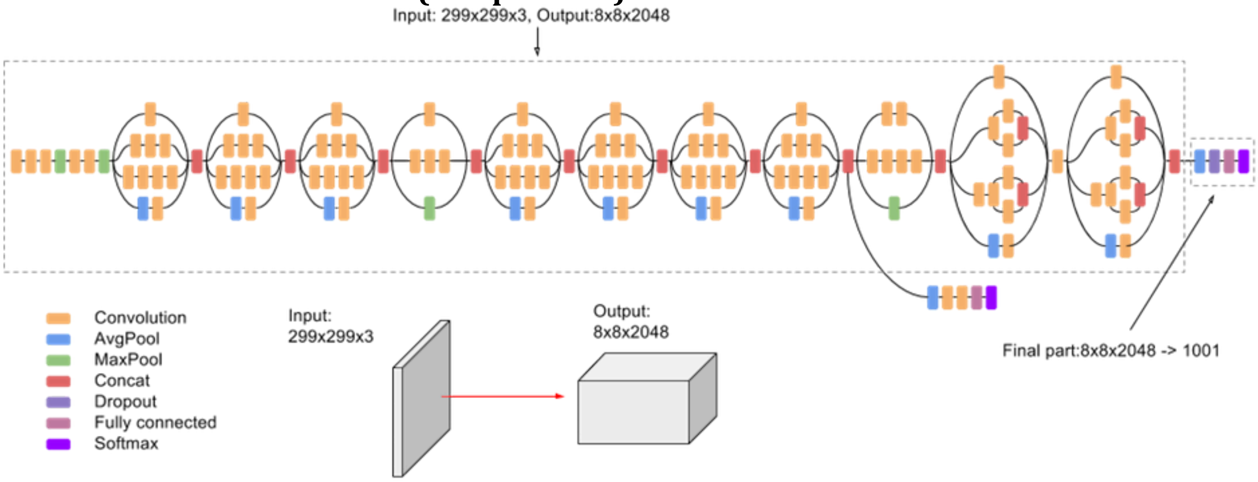


GoogLeNet Variants and Successors

Following the success of the original Google Net (Inception v1), several variants and successors enhanced its architecture. These include Inception v2, v3, v4, and the Inception-ResNet hybrids. Each of these models introduced key improvements and optimizations. These addressed various challenges and pushed the boundaries of what was possible with the CNN architectures.

- **Inception v2 (2015):** The second version of Inception included improvements such as batch normalization and shortcut connections. It also refined the inception modules by replacing larger convolutions with smaller, more efficient ones. These changes improved accuracy and reduced training time.
- **Inception v3 (2015):** The v3 model further refined Inception v2 by using atrous convolution (dilated convolutions that expand the network's receptive field without sacrificing resolution and significantly increasing network parameters).

(Inception v3)



- **Inception v4, Inception-ResNet v2 (2016):** This version of Inception introduced residual connections (inspired by ResNet) into the Inception lineage, which led to further performance improvements.
- **Xception (2017):** Xception replaced Inception modules with depth-wise separable convolutions.
- **MobileNet (2017):** This architecture is for mobile and embedded devices. The network uses depth-wise separable convolutions and linear bottleneck layers.
- **EfficientNet (2019):** This is a family of models that scales both model size and accuracy strategically by using Neural Architecture Search (NAS).

What is VGGNET ?

Introduction

VGGNet, developed by the **Visual Geometry Group** at the **University of Oxford**, is a seminal Convolutional Neural Network (CNN) architecture that emphasizes depth through the use of small convolutional filters. Introduced in **2014**, VGGNet has significantly influenced subsequent CNN designs.

It is a typical deep Convolutional Neural Network (CNN) design with numerous layers, and the abbreviation VGG stands for Visual Geometry Group. The term “deep” describes the number of layers, with VGG-16 or VGG-19 having 16 or 19 convolutional layers, respectively.

Innovative object identification models are built using the VGG architecture. The VGGNet, created as a deep neural network, outperforms benchmarks on a variety of tasks and datasets outside of ImageNet. It also remains one of the most often used image recognition architectures today.

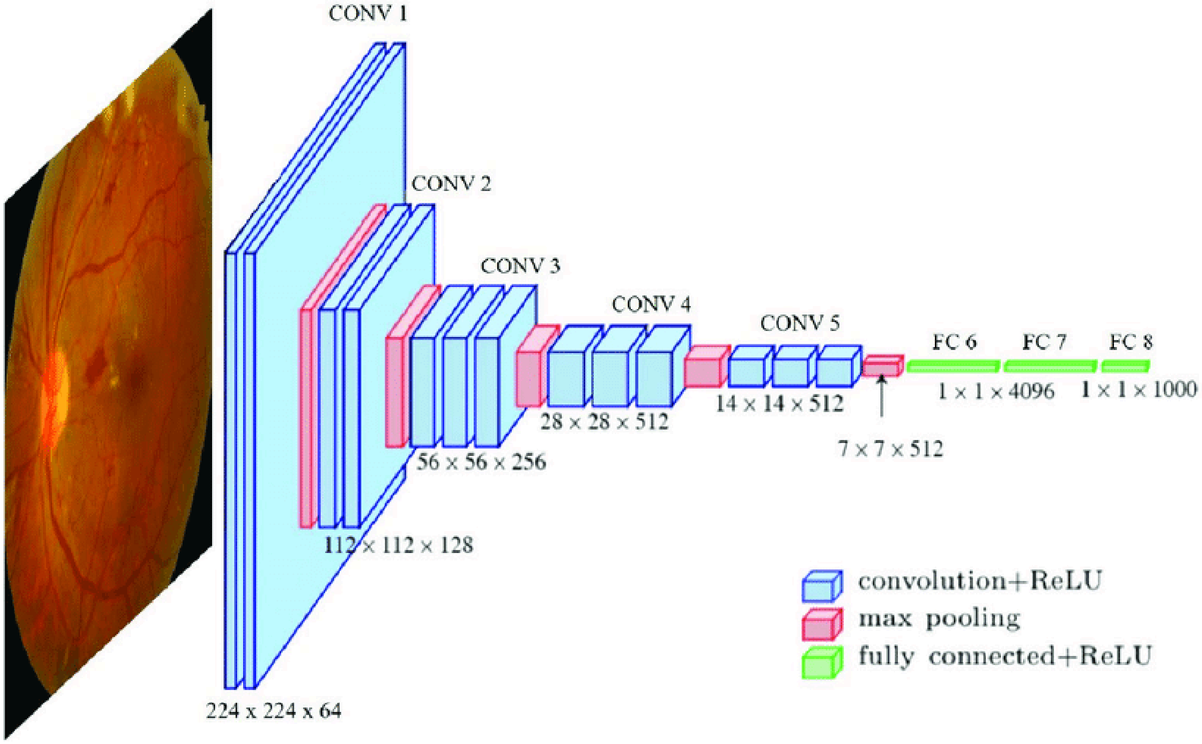
- **Depth:** VGGNet comes in various configurations, notably VGG-16 and VGG-19, with 16 and 19 weight layers respectively.

- **Convolutional Layers:** Utilizes 3×3 convolutional filters consistently, allowing the network to capture intricate patterns while maintaining computational efficiency.

- **Pooling Layers:** Incorporates 2×2 max-pooling layers with a stride of 2 after certain convolutional blocks to reduce spatial dimensions.

- **Fully Connected Layers:** Concludes with three fully connected layers, the first two with 4096 neurons each, followed by a final layer corresponding to the number of classes in the dataset.

Architecture of VGG-16:



Input Layer:

★ The input is a 224×224 image with three color channels.

★ The first convolutional layer extracts 64 feature maps.

Convolutional Layers:

★ Uses 3×3 convolutional filters to capture spatial hierarchies.

★ Includes 1×1 convolution filters for linear transformations.

★ Stride = 1 pixel to maintain spatial resolution.

★ Each convolution is followed by a ReLU activation function, which accelerates training.

★ Conv 1: Two convolutional layers with 64 filters each, followed by max-pooling.

★ Conv 2: Two convolutional layers with 128 filters each, followed by max-pooling.

★ Conv 3: Three convolutional layers with 256 filters each, followed by max-pooling.

★ Conv 4: Three convolutional layers with 512 filters each, followed by max-pooling.

★ Conv 5: Three convolutional layers with 512 filters each, followed by max-pooling.

Hidden Layers:

- All hidden layers use **ReLU activation** for non-linearity.
- Unlike some earlier models, **Local Response Normalization (LRN)** is avoided, as it increases memory usage and training time without improving accuracy.

Fully Connected Layers

- The feature maps are flattened into a vector (7×7×512).
- **FC 6 & FC 7:** Two fully connected layers with 4096 neurons, using ReLU activation.
- **FC 8:** The final classification layer with 1000 neurons (for ImageNet classes) or task-specific output.

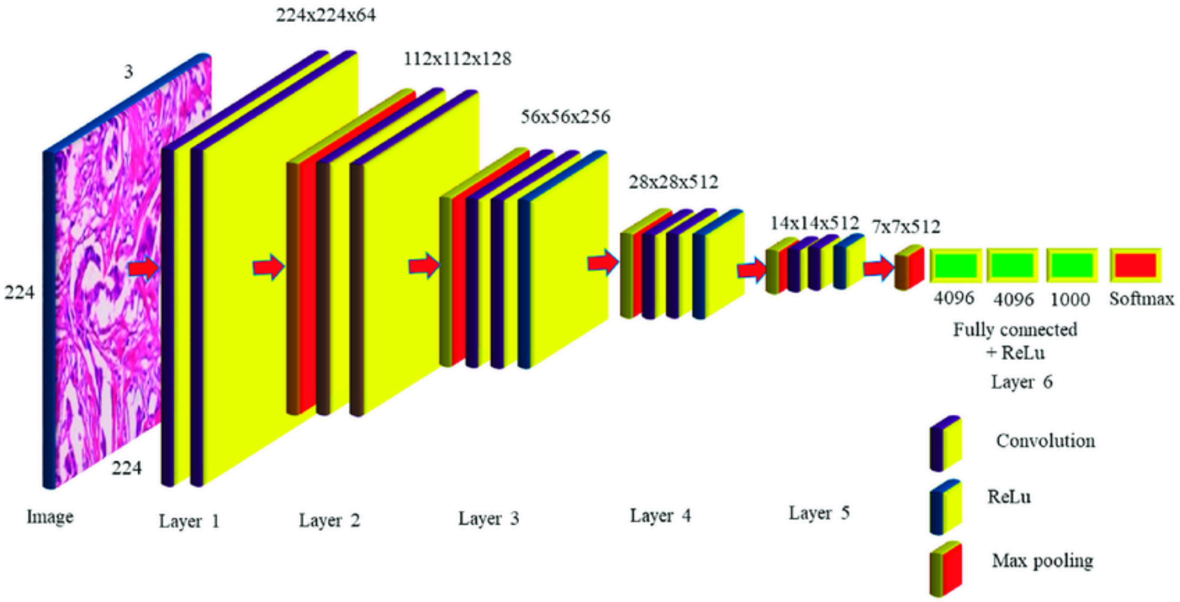
VGG-16

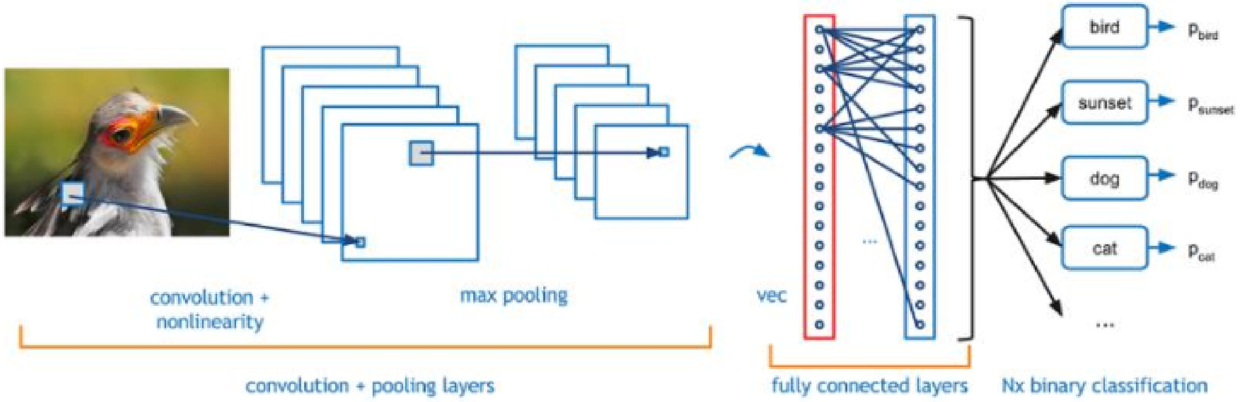
The deep neural network's 16 layers are indicated by the number 16 in their name, which is VGG (VGGNet). This indicates that the VGG16 network is quite large, with a total of over 138 million parameters. Even by today's high standards, it is a sizable network. The network is more appealing due to the simplicity of the VGGNet16 architecture, nevertheless. Its architecture alone can be used to describe how uniform it is.

The height and width are decreased by a pooling layer that comes after a few convolution layers. There are around 64 filters available, which we can then multiply by two to get about 128 filters, and so on up to 256 filters. In the last layer, we can use 512 filters.

Limitations Of VGG 16:

- It is very slow to train (the original VGG model was trained on Nvidia Titan GPU for 2–3 weeks).
- The size of VGG-16 trained imageNet weights is *528* MB. So, it takes quite a lot of disk space and bandwidth which makes it inefficient.
- 138 million parameters lead to exploding gradients problem.





VGG

UNIVERSITY OF OXFORD

Evolution of Convolutional Neural Networks (CNNs) Models 2017-2024

DenseNet (2017): DenseNet, or Densely Connected Convolutional Networks, introduced a novel connectivity pattern where each layer receives inputs from all preceding layers and passes its feature maps to all subsequent layers. This design promotes feature reuse, mitigates the vanishing gradient problem, and reduces the number of parameters compared to traditional CNNs.

MobileNet Series:

- ***MobileNetV1 (2017):*** Introduced depthwise separable convolutions to reduce computation, making it suitable for mobile and embedded vision applications.
- ***MobileNetV2 (2018):*** Enhanced the original by incorporating inverted residuals and linear bottlenecks, further improving mobile efficiency.
- ***MobileNetV3 (2019):*** Combined neural architecture search (NAS) with network redesign, offering optimized performance for mobile devices.

NASNet (2018): Developed using Neural Architecture Search (NAS), NASNet optimized both accuracy and computational cost, paving the way for automated design of neural network architectures.

MnasNet (2018): An AutoML-designed lightweight model tailored for mobile devices, MnasNet balances latency and accuracy, emphasizing efficient performance on resource-constrained hardware.

AmoebaNet (2019): A NAS-based model, AmoebaNet achieved state-of-the-art accuracy on ImageNet, demonstrating the potential of automated architecture search methods in designing high-performance CNNs.

EfficientNet Series:

- ***EfficientNet (2019):*** Introduced a scalable CNN using compound scaling (depth, width, resolution) for improved efficiency, achieving better accuracy with fewer parameters.
- ***EfficientNetV2 (2021):*** Offered faster training and better parameter efficiency than the original EfficientNet, further enhancing performance.

EfficientDet (2020): Focused on object detection, EfficientDet extends EfficientNet with bidirectional feature fusion, providing a scalable and efficient solution for detection tasks.

RegNet (2020): Introduced a network design space that facilitates the creation of simple, high-performance models, emphasizing regularization and scalability.

ResNeSt (2020): Enhanced ResNet architectures with split-attention blocks, improving feature representation and achieving better performance on various vision tasks.

NFNets (2021): Normalizer-Free ResNets, or NFNets, achieved state-of-the-art results without relying on batch normalization, simplifying training and improving robustness.

ConvNeXt (2022): Modernized ConvNets by incorporating design principles from Vision Transformers (ViTs), bridging the gap between CNNs and transformer models in vision tasks.

Recent Developments (2023-2024):

- ***MobileOne (2023):*** Introduced by Apple, optimized for mobile and edge devices, achieving state-of-the-art speed-accuracy trade-offs with reparameterized convolutions. Notably, it runs over 1x faster than previous MobileNet variants on iOS devices.
- ***RepViT (2023):*** Merged CNNs with Vision Transformers (ViTs) through reparameterization, maintaining CNN efficiency while leveraging ViT-like global modeling, ideal for edge devices requiring both speed and high accuracy.
- ***FasterViT (2023)****:* A hybrid architecture combining hierarchical ViTs with CNN-like locality, focusing on reducing computational overhead in vision transformers.
- ***EdgeNeXt (2023):*** Designed for edge computing, EdgeNeXt is a lightweight model utilizing split depth-wise convolutions and adaptive feature fusion.
- ***InternImage (2023):*** Modernized CNNs with deformable convolutions and large-kernel attention, enhancing object detection and segmentation tasks.

CNN Models Used in Smartphones

Smartphones require CNN models that are lightweight, efficient, and optimized for low-power hardware while maintaining high accuracy for tasks like image recognition, object detection, and augmented reality. Here are some CNN models commonly used in mobile and edge AI applications:

1. MobileNet Series (2017 - Present)

Optimized for mobile devices using depthwise separable convolutions.

- **MobileNetV1 (2017):** Introduced efficient convolutions for real-time applications.

- **MobileNetV2 (2018):** Added **inverted residuals** and **linear bottlenecks** to improve performance.
- **MobileNetV3 (2019):** Used **Neural Architecture Search (NAS)** to optimize speed and accuracy.

Face recognition, gesture control, augmented reality (AR) filters, Google Lens.

2. EfficientNet Series (2019 - Present)

Uses compound scaling to improve efficiency across different mobile hardware.

- **EfficientNet-B0 to B7 (2019):** Scales width, depth, and resolution systematically.
- **EfficientNetV2 (2021):** Optimized for faster training and inference on mobile devices.

AI-powered camera enhancements, Google Photos search, object detection.

3. NASNet (2018) & MnasNet (2018)

These models were designed using Neural Architecture Search (NAS) for optimal trade-offs between accuracy and latency on mobile hardware.
Low-power AI processing in smartphones.

4. MobileOne (2023)

Designed by Apple, this CNN model is optimized for iPhones and iPads, offering 1x faster inference compared to previous MobileNet versions.
Apple's AI-powered features like Face ID, object detection, and scene recognition.

5. RepViT (2023)

Combines CNNs with Vision Transformers (ViTs) while maintaining high efficiency.
Image segmentation, scene detection in Android AI assistants.

6. FasterViT (2023)

Hybrid model with CNN-like locality and Vision Transformer efficiency, reducing power consumption.
Real-time image enhancements, on-device AI acceleration.

7. EdgeNeXt (2023)

Designed specifically for low-power edge devices, making it ideal for smartphones.
AI-driven photography, battery-optimized image classification.

Which Smartphone Brands Use CNNs?

- **Apple:** Uses MobileOne, EfficientNet, and custom AI models for Face ID, Photos, and Siri.
- **Google:** Uses MobileNet, EfficientNet, and Vision Transformers in Android and Google Lens.
- **Samsung:** Integrates MobileNet and EfficientNet for camera AI and Bixby Vision.
- **Huawei & Xiaomi:** Use MnasNet, EfficientDet, and proprietary CNNs for on-device AI processing.