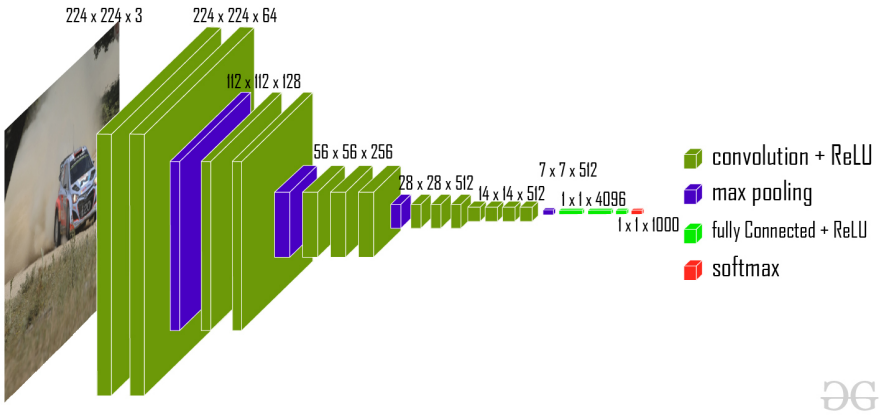

Figure 1: VGG-16

1 VGG-16

1.1 Introduction

The VGG-16 model is a convolutional neural network (CNN) architecture that was proposed by the Visual Geometry Group (VGG) at the University of Oxford. It is characterized by its depth, consisting of 16 layers, including 13 convolutional layers and 3 fully connected layers. VGG-16 is renowned for its simplicity and effectiveness, as well as its ability to achieve strong performance on various computer vision tasks, including image classification and object recognition. The model's architecture features a stack of convolutional layers followed by max-pooling layers, with progressively increasing depth. This design enables the model to learn intricate hierarchical representations of visual features, leading to robust and accurate predictions. Despite its simplicity compared to more recent architectures, VGG-16 remains a popular choice for many deep learning applications due to its versatility and excellent performance.

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an annual competition in computer vision where teams tackle tasks including object localization and image classification. VGG16, proposed by Karen Simonyan and Andrew Zisserman in 2014, achieved top ranks in both tasks, detecting objects from 200 classes and classifying images into 1000 categories.

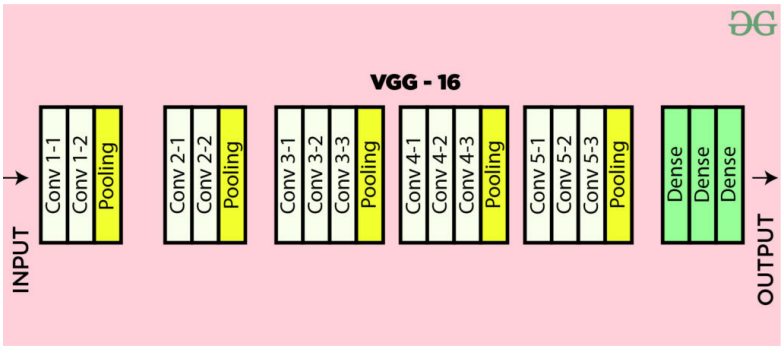

Figure 2: VGG Architecture

1.2 VGG Architecture

The VGG-16 architecture is a deep convolutional neural network (CNN) designed for image classification tasks. It was introduced by the Visual Geometry Group at the University of Oxford. VGG-16 is characterized by its simplicity and uniform architecture, making it easy to understand and implement.

The VGG-16 configuration typically consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. These layers are organized into blocks, with each block containing multiple convolutional layers followed by a max-pooling layer for downsampling.

Here's a breakdown of the VGG-16 architecture based on the provided details:

1.2.1 Input Layer:

1. Input dimensions: (224, 224, 3)
2. Convolutional Layers (64 filters, 3×3 filters, same padding):

   Two consecutive convolutional layers with 64 filters each and a filter size of 3×3. Same padding is applied to maintain spatial dimensions.
3. Max Pooling Layer (2×2, stride 2):

   Max-pooling layer with a pool size of 2×2 and a stride of 2.
4. Convolutional Layers (128 filters, 3×3 filters, same padding):

Two consecutive convolutional layers with 128 filters each and a filter size of 3×3.

5. Max Pooling Layer (2×2, stride 2):

   Max-pooling layer with a pool size of 2×2 and a stride of 2.

6. Convolutional Layers (256 filters, 3×3 filters, same padding):

   Two consecutive convolutional layers with 256 filters each and a filter size of 3×3.

7. Convolutional Layers (512 filters, 3×3 filters, same padding):

   Two sets of three consecutive convolutional layers with 512 filters each and a filter size of 3×3.

8. Max Pooling Layer (2×2, stride 2):

   Max-pooling layer with a pool size of 2×2 and a stride of 2.

9. Stack of Convolutional Layers and Max Pooling:

   Two additional convolutional layers after the previous stack. Filter size: 3×3.

10. Flattening:

    Flatten the output feature map (7x7x512) into a vector of size 25088.

11. Fully Connected Layers:

    Three fully connected layers with ReLU activation. First layer with input size 25088 and output size 4096. Second layer with input size 4096 and output size 4096. Third layer with input size 4096 and output size 1000, corresponding to the 1000 classes in the ILSVRC challenge. Softmax activation is applied to the output of the third fully connected layer for classification.

2 Comparing Results

Confusion matrices show that the scratch model it's more benefit than vgg base model. Maybe, it's because of the first one trained just by mnist dataset and also test with them but VGG16 was pretrained by imagenet that is more complex than mnist.

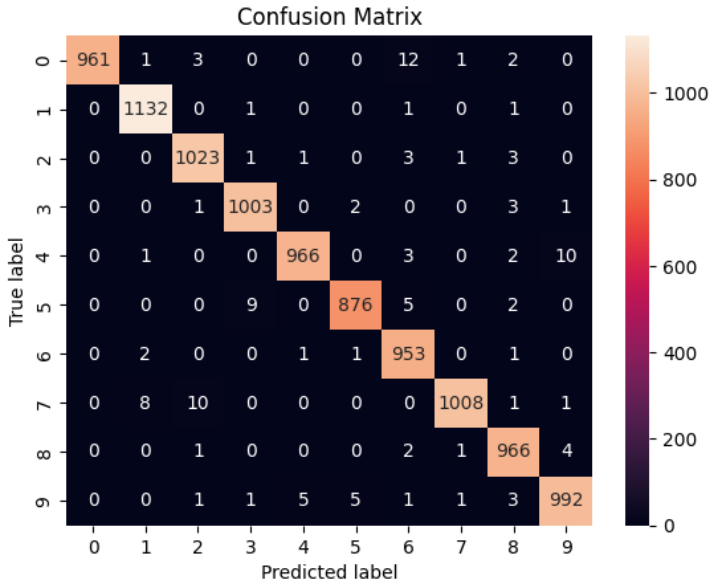

Figure 3: CNN model

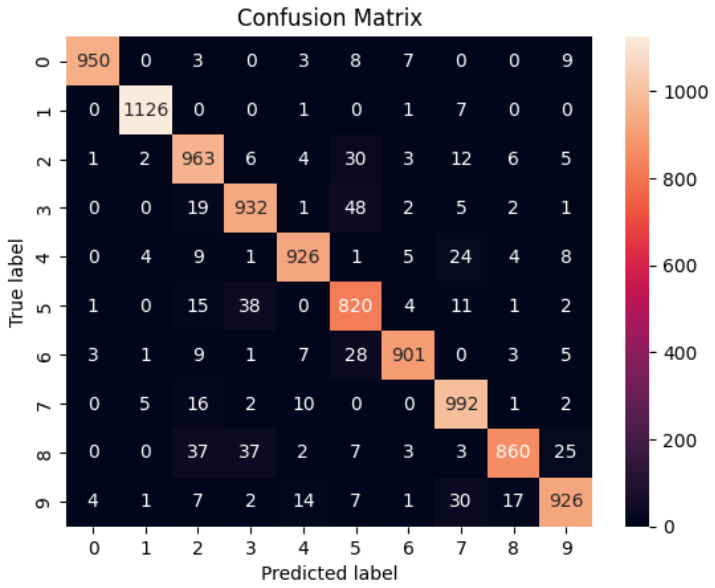

Figure 4: VGG16 base model

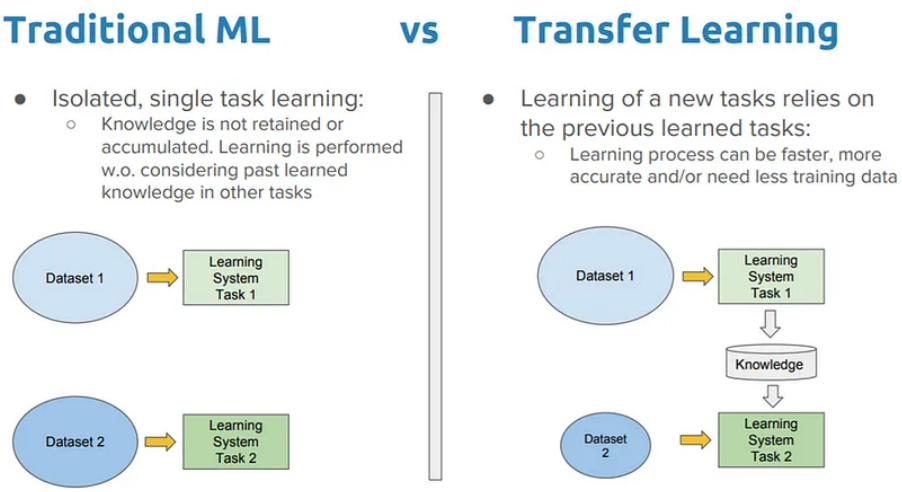

Figure 5: Traditional Learning vs Transfer Learning

3 Transfer learning

3.1 Motivation

When trained from scratch, a model's parameters are initialized randomly and updated by some optimization algorithm like stochastic gradient descent or Adam So, if there are two different tasks to be solved, no matter how similar, this process will be repeated separately both times However, does it really need to be? Consider the case when we humans learn something new: do we ALWAYS start from the ground up

3.2 What Is Transfer Learning?

In transfer learning, the knowledge of an already trained machine learning model is applied to a different but related problem. For example, if you trained a simple classifier to predict whether an image contains a backpack, you could use the knowledge that the model gained during its training to recognize other objects like sunglasses.

With transfer learning, we basically try to exploit what has been learned in one task to improve generalization in another. We transfer the weights that a network has learned at "task A" to a new "task B."

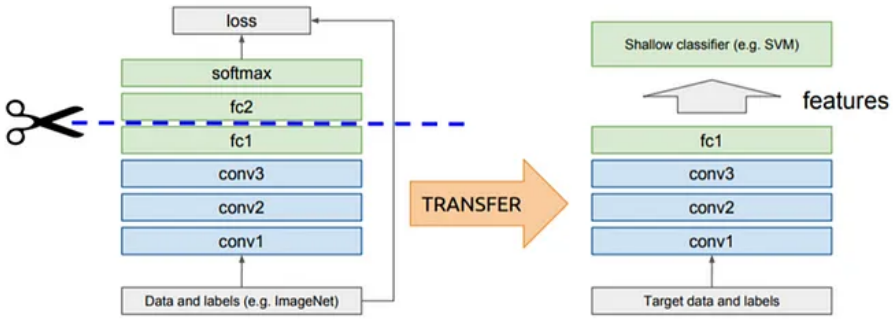

Figure 6: Transfer Learning with Pre-trained Deep Learning Models as Feature Extractors

3.3 Freezing Layers

- Take the pretrained network from task 1
- Freeze some of the initial layers (i.e. disable gradient updates to them) and treat them as a fixed feature extractor — take the activations from these layers as some intermediate representation of your input
- Discard the remaining un-frozen (possibly none) later layers
- Optionally, attach a new custom network (often only a few linear layers that are initialized randomly) that is not frozen to the frozen layers