Nº Réf : ....

Centre Universitaire
Abd Elhafid Boussouf Mila

Institut des Sciences et Technologie
Département de Mathématiques et Informatique

Mémoire préparé en vue de l'obtention du diplôme de Master
En : Informatique
Spécialité : Sciences et Technologies de l'Information et de la Communication (STIC)

Semantic Segmentation of Remote sensing image

Préparé par : Zahri chaima
Ferdi Ranya

Soutenue devant le jury

Encadré par	Dr. Aissa Boulmerka	C.U. Abd Elhafid Boussouf
Président	Dr.Sadek Benhammada	C.U. Abd Elhafid Boussouf
Examinatrice	Dr.Souheila Khalfi	C.U. Abd Elhafid Boussouf

Année Universitaire : 2021/2022

Abstract

هدف هذه الرسالة هو الحصول على فهم شامل لمجال التجزئة الدلالية للصور بالإضافة إلى الغوص في مجال التعلم العميق وكيف يمكننا استخدام شبكات مختلفة، لا سيما الشبكات العصبية التلافيفية، وبشكل أكثر تحديدًا ، تطبيق بنيات VGG و Resnet و Inception و U-Net في صور الاستشعار عن بُعد. تظهر التجارب والتقييمات الخاصة بالتصميمات المذكورة أعلاه فوائد استخدام هذه الشبكات بالإضافة إلى كفاءتها في حل المشكلات المتعلقة بالتجزئة الدلالية للصور.

الكلمات الرئيسية - التجزئة الدلالية، التعلم العميق، الشبكات العصبية التلافيفية، U-Net، Inception، Resnet، VGG، صور الاستشعار عن بعد ،صورة، الصور الجوية.

This dissertation's objective is to gain a thorough understanding of the field of image semantic segmentation as well as a deep dive into the field of deep learning and how we can use different networks, particularly convolutional neural networks and, more specifically, the application of VGG, Resnet, Inception, and U-Net architectures in the Remote Sensing imagery. The experiments and evaluations of the aforementioned models show the benefits of using such networks as well as their efficiency in resolving several problems with image semantic segmentation.

Keywords — Semantic segmentation, deep learning, convolutional neural networks CNN, U-Net, VGG, Resnet, Inception, Remote sensing image, Aerial image.

Acknowledgement

الحمد لله الذي منحني القدرة على الكتابة والتفكير، والصبر على تحقيق الحلم الذي يشرفني تكريس هذا العمل المتواضع. أهدي تخرجي إلى من كلله الله بالوقار وإلى من احمل اسمه بكل افتخار والدي العزيز **سليمان** وإلى بسمه الحياه وسر الوجود وإلى معنى الحب والحنان أمي الحبيبه **حملاوي صورية**،الى امي الثانية **بوعروج زينب** ،إلى من أخذ بيدي نحو ما أريد وأعاد إليّ ثقتي بقدرتي على التقدّم: إليك زوجي **موراس سفيان**

و أود أن أغتنم هذه الفرصة حتى أشكر اكبر مكسب في مسيرتي الدراسية الدكتور : **عيسى بو المرقة** على دعمك السخي والكبير لي للخروج بهذا البحث حتى النهاية، أنا ممتنة جدًا لك ولمساعدتك، والحمد لله الذي سخر هذه الفرصة لي، وألف شكر لقلبك الطيب وشخصك الكريم.

الي سندي ورفيقتي في هدا العمل المتواضع **زهري شيماء**

إلى عائلتي **فردي حملاوي موراس** والى كل صديقاتي واخواتي واخوتي الأعزاء، كما نود أن نشكر أعضاء لجنة التحكيم الذين شرفونا بفحص هذا العمل.

أخيرًا ، نجدد شكرنا لأولئك الذين ساعدونا للقيام بهذا العمل دون أن ننسى أساتذة معهد الرياضيات واعلام الي

اسال الله ان يمن علينا بالتوفيق والنجاح الدائم.

فردي رانية

الحمد والشكر لله رب العالمين الذي بنعمته تتم الصالحات الحمد لله الذي بتوفيقه و تسهيل منه حققت هذا الحلم في انجاز هذا العمل المتواضع. اهدي تخرجي الى قدوتي الاولى و سندي في الحياة الى من رفعت رأسي عاليا افتخارا به، أبي العزيز **زهري ياسين** وإلى تلك المرأة العظيمة التي ربت و عملت، أمي الحنونة **طالبي لطيفة** . وإلى من تمنى بلوغي هذا الهدف **جدي** رحمه الله. وإلى من دعمني و كان سندا لي زوجي **فرحات عبد الرزاق** , وإلى كل افراد عائلتي خاصة إخوتي و أخواتي الذين وقفو بجانبي طوال مسيرتي

كما يسرني ان اوجه خالص شكري الى اعظم استاذ في مسيرتي الدراسية مشرفي الدكتور **عيسى بوالمرقة** الذي يتمتع بخبرة عالية و انا ممتنة لك على دعمك و مساعدتك و ارشادك لنا و الحمد لله الدي سخر لنا هده الفرصة.

الى صديقتي و رفيقتي في هدا العمل **فردي رانية**

كما نتقدم بالشكر الى اعضاء لجنة التحكيم الذي نتشرف بفحصهم لهذا العمل، كما نشكر اساتذة معهد الراضيات و الاعلام الالي . والى كل من ساعدنا على اتمام هذه المذكرة من قريب و من بعيد و كل من وقفوا بجانبي في مسيرتي الدراسية و نسأل الله التوفيق و النجاح.

زهري شيماء

Contents

List of Figures

List of Tables

List of Abbreviations

AE	Auto-Encoder
ALUS	Arithmetic Logic Units
AMD	Advanced Micro Devices
ANN	Artificial Neural Network
API	Application Programming Interface
ARM	Automatic Restart Manager
ASICs	Application-Specific Integrated Circuits
ASPP	Atrous Spatial Pyramid Pooling
BN	Batch Normalization
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DNN	Deep Neural Network
DL	Deep Learning
ED	Encoder-Decoder
EMA	Exponential Moving Average
FC	Fully Connected
FCN	Fully Convolutional Network
GAN	Generative Adversarial Network
GLS	Gray Level Segmentation
GPU	Graphics processing Unit
GUI	Graphical User Interface
IBM	International Business Machines
IDE	Integrated Development Environment
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IoU	Intersection-Over-Union
LR	Learning Rate
LSTM	Long Short-Term Memory

MAC	Multiply and accumulate
MLP	Multi-Layer Perceptron
NN	Neural Network
RGB	Red Green Blue
ReLU	Rectified Linear Unit
ResNet	Residual Neural Network
RNN	Recurrent Neural Network
Tanh	Hyperbolic Tangent
TPU	Tensor Processing Unit
SVM	Machine Vertical Suport
RGB	Rouge,Vert,blue
DCNN	Depp Convolition Neural Network

General introduction

Many visual understanding systems, especially remote sensing imaging systems, rely on image segmentation and object detection to understand images (or video frames). Image segmentation has been a fundamental challenge in computer design from the beginning.

On the one hand, there are two approaches to image segmentation: semantic image segmentation (classifying pixels with semantic labels) and instance segmentation (partitioning of specific objects). Semantic segmentation is more difficult than whole-image classification, which assigns a single label to the entire image. Object categories (e.g., human, automobile, tree, sky) are applied at the pixel level, making semantic segmentation a more difficult task. Using instance segmentation, each object of interest in an image can be detected and separated, extending the semantic segmentation process (e.g., individual people). There are many approaches for image segmentation from simple algorithms like active contours, graph cuts, and Markov random fields to more advanced algorithms like histogram-based approaches, area growth, k-means clustering, watershed methods.

Deep learning has led to a new generation of image segmentation models that outperform prior approaches, achieving the best accuracy rates on established benchmarks. This has resulted in a paradigm shift in the area of image analysis, with advances in machine learning and artificial intelligence at the heart of the field.

In this dissertation, we have looked at some of the key concepts and approaches that allow the implementation of image semantic segmentation. The content of the present dissertation can be described as follows:

The content of first chapter begins by defining and explaining a few keywords and ideas relevant to deep learning (DL). Then we go through some fundamental deep learning architectures including CNN, RNN, LSTM, ED and AE models, and GANs. Following that, we go through some of the most popular deep learning frameworks, such as TensorFlow, PyTorch, and Keras. Finally, we describe and present certain DL-related hardware, such as CPUs, GPUs, and TPUs.

We begin chapter two by describing and comparing semantic segmentation to other computer vision problems, explaining methodologies and techniques that are comparable to semantic segmentation, and clarifying the link between the latter and DL. Following that, a look at some of CNN's most popular models, including LeNet-5, AlexNet, VGG-16, ResNet, GoogleNet, and MobileNet, is taken, with images and tables used to define the most significant elements and components that make them up. Finally, we look at the entire bypass networks, SegNet, DeepLab, Pyramid Landscape Analysis Network, and U-Net, and discuss DL-based semantic segmentation methods, including explanations and details.

In the final chapter, we explain the U-net Model for the semantic segmentation of remote-sensing images. We give some details about the network architecture and technology used to address this problem. Next, we describe our experiments, tests, and results. The predictions of each trained model are represented by a set of visual figures. Finally, we evaluate and compare the performance of the studied model.

Chapter 1

Deep Learning

1.1 Introduction

This chapter introduces the concepts of deep learning (DL) and describes some of the popular frameworks used in DL. Starting by general concepts of ANNs, perceptrons, MLPs, activation functions, cost functions, gradient descent, LR, and finally, forward propagation and backpropagation. In addition, we will describe DNN architectures used widely by the computer vision community, including CNN, RNN, and LSTM. Then we will explain some of the popular frameworks used in DL, such as TensorFlow, PyTorch, and Keras. At the end of this chapter, we will talk about the hardware used by DL techniques such as CPUs, GPUs, and TPUs.

1.2 General concepts

1.2.1 Definitions

DL is considered as a subset of ML and AI, and thus DL can be seen as an AI function that mimics the human brain's processing of data. The worldwide popularity of "Deep learning" is increasing day by day, as shown in [**1**].

DL technology originated from the artificial neural network (ANN), and has become one of the hot topics within the area of machine learning, artificial intelligence as well as data science and analytics, due to its learning capabilities from the given data, Many corporations including Google, Microsoft, Nokia, etc., study it actively as it can provide significant results in different classification and regression problems and datasets [**2**].

this technology uses multiple layers to represent the abstractions of data to build computational models. While deep learning takes a long time to train a model due to a large number of parameters, it takes a short amount of time to run during testing as compared to other machine learning algorithms.

By demonstrating how a machine's internal parameters are employed to generate the representation in each layer from the representation in the preceding layer, DL unveils intricate patterns in big data sets using the backpropagation technique.

One of the key differences between machine learning and DL models is feasibility. feature extraction; feature extraction is done by humans in machine learning.DL models, on the other hand, come to their own conclusions.

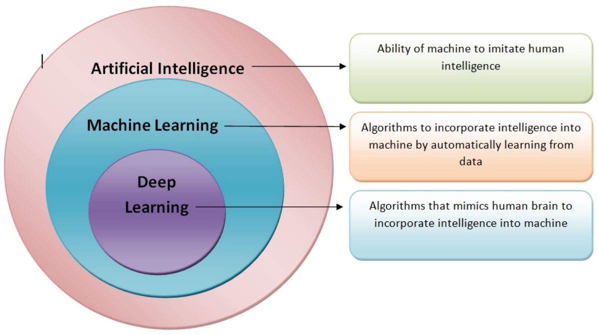

Artificial Intelligence
Machine Learning
Deep Learning
Ability of machine to imitate human intelligence
Algorithms to incorporate intelligence into machine by automatically learning from data
Algorithms that mimics human brain to incorporate intelligence into machine

Figure 1.1: Illustration showing the relationship between AI, ML, and DL [**44**].

1.2.2 Artificial neural networks

ANNs are influenced by the way biological neural systems process information, such as the brain [**4**].

The data processing system is made up of numerous highly interconnected processing elements called **neurons** that collaborate to fix targeted problems.

1.2.3 Perceptrons

The perceptron is the essential part of an Artificial Neural Network (ANN); a Perceptron is a learning approach for supervised binary classifiers. Binary classifiers use a series of vectors to determine if an input belongs to a particular class. A single-layer neural network is what a perceptron is, in a nutshell. Weights and bias are included for each of the input value[**5**], in addition to a net sum and activation function.

The first step is to multiply all the input values by their respective weights. Then, the weighted sum is calculated by multiplying each of the multiple values. Weighted sums are added together and used to a perceptron's activation function to produce its output[**6**]. Using the activation function, we can ensure that the output is mapped to values like 0 and 1 (or -1 and 1). The strength of a node can be gauged by looking at the weight of an input. Activation function curves can be shifted up or down by varying the bias value of an input in a similar way.

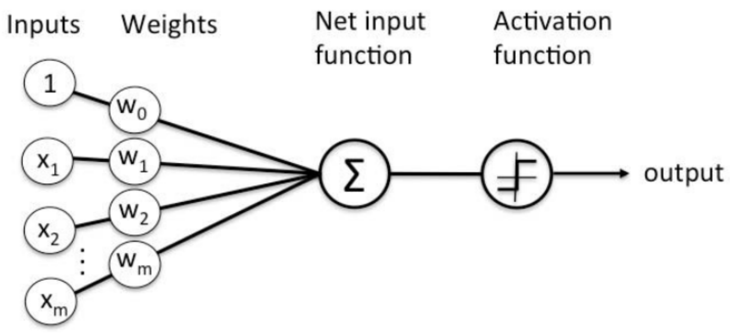


Figure 1.2: Illustration of Schematic representation of the mathematical model of an artificial neuron (processing element) [**10**]

ANNs are a logical progression from perceptrons. An example of a feed-forward neural network is a multi-layered perceptron. Input, output, and perceptron neurons (as well as synaptic weights) would all be part of it [**9**].

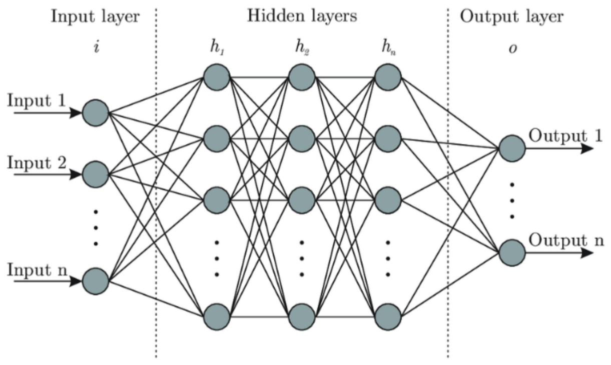


Figure 1.3: The architecture of ANN[**11**]

1.2.4 Activation functions

The activation function is critical for an artificial neural network to learn. It converts an input signal to an output signal like any other function. This output signal is the input to the next layer [**7**].

A neural network that lacks an activation function is a linear regression model; the weights and biases would perform a linear transformation without the activation feature. Among the most well-known activation functions, we find **Sigmoid** function, **Tanh**, **ReLU**, **Softmax**, etc.

1.2.4.1 Sigmoid

A sigmoid function is a mathematical function with a characteristic S-shaped curve or sigmoid curve that spans the range of 0 to 1, It is utilized in models where the outcome is required to estimate a probability [**8**].

It is defined as follows:

$$Sigmoid(x) = \frac{1}{1 + e^{-}x}$$

The sigmoid transformation produces a continuous range of values between 0 and 1.

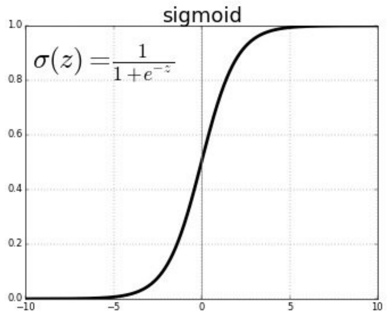


Figure 1.4: Sigmoid activation function

1.2.4.2 Tanh

The Hyperbolic Tan can also be called as symmetric sigmoid is, in fact, a scaled sigmoid function. Keep in mind that the slope for tanh is stronger than for sigmoid (derivatives are steeper).

It is defined as follows:

$$tanh(x) = 2sigmoid(2x) - 1$$

which is:

$$f(x) = tanh(x) = \frac{2}{1 + e^{-}2x} - 1$$

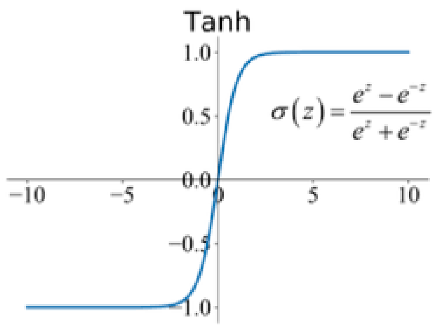


Figure 1.5: Hyperbolic Tangent activation function

1.2.4.3 ReLU

ReLU is another popular function, and it's preferred over sigmoid in recent networks. it is simply defined as follows:

$$f(x) = max(x, 0)$$

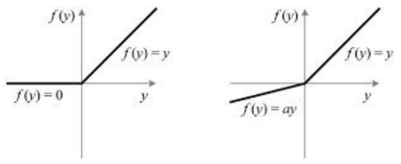


Figure 1.6: Rectified Linear Units activation function

The main advantage of using ReLU is that its derivative value is constant for all inputs greater than 0. The constant derivative value allows the network to train more quickly.

1.2.4.4 Softmax

It is a mathematical function that turns a vector of integers into a vector of probabilities, with the probability of each value proportional to the vector's relative scales, It can be defined as the sum of several sigmoid functions .[**12**].

$$S(x_i) = \frac{e_i^x}{\sum_{j=1}^{n} e_j^x}$$

The sigmoid function would be appropriate if we had a binary output; however, if we have a multiclass classification problem, softmax makes it extremely simple to assign values to each class that can be easily interpreted as probabilities.

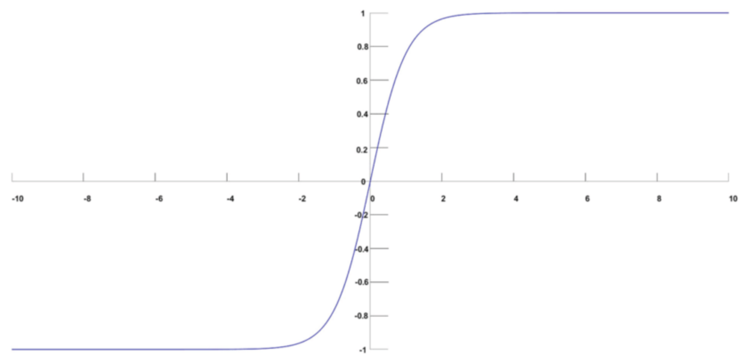


Figure 1.7: Multi-class classification with NN and softmax function

1.2.5 Batch normalization

As an algorithmic technique, batch-normalization speeds up and improves the stability of DNNs. After BN alters the signal at each hidden layer, it looks like this:

$$(1) \quad \mu = \frac{1}{n}\sum_i Z^{(i)} \qquad (2) \quad \sigma = \frac{1}{n}\sum_i (Z^{(i)} - \mu)$$

$$(3) \quad Z^{(i)}_{norm} = \frac{(Z^{(i)} - \mu)}{\sqrt{\sigma^2 - \varepsilon}} \qquad (4) \quad \breve{Z} = \gamma \, * \, Z^{(i)}_{norm} + \beta$$

Using (1) and (2), the BN layer first calculates the mean μ and standard deviation σ of the activation values throughout the batch (2). The activation vector $Z^{(i)}_{norm}$ is then normalized with (3). As a result, the output of each neuron follows a conventional normal distribution across the batch (For numerical stability, ε is utilized as a constant).

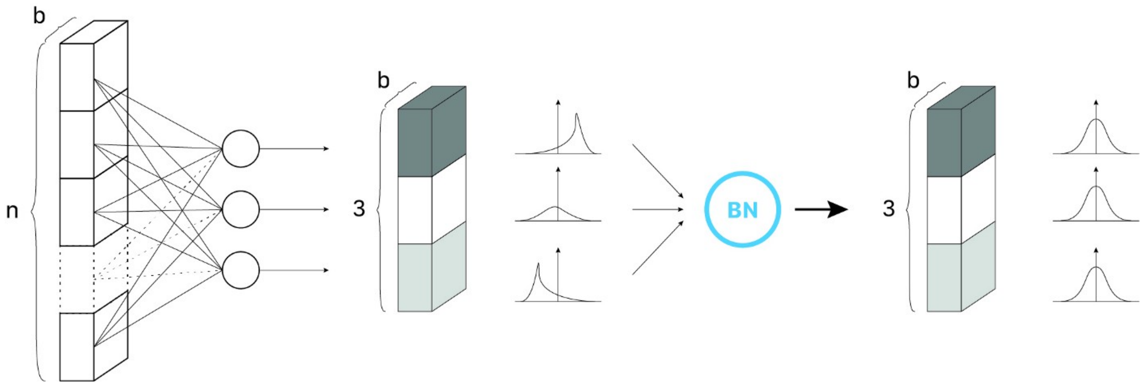


Figure 1.8: Batch normalization first step. Example of a 3-neurons hidden layer, with a batch of size b. Each neuron follows a standard normal distribution from .

By applying a linear transformation with two trainable parameters γ and β, it calculates the layer's output $Z^{(i)}_{norm}$ at the end (4). This phase allows the model to select the best distribution for each hidden layer by modifying two parameters: γ allows to alter the standard deviation; β permits to alter the bias by moving the curve to the right or left.

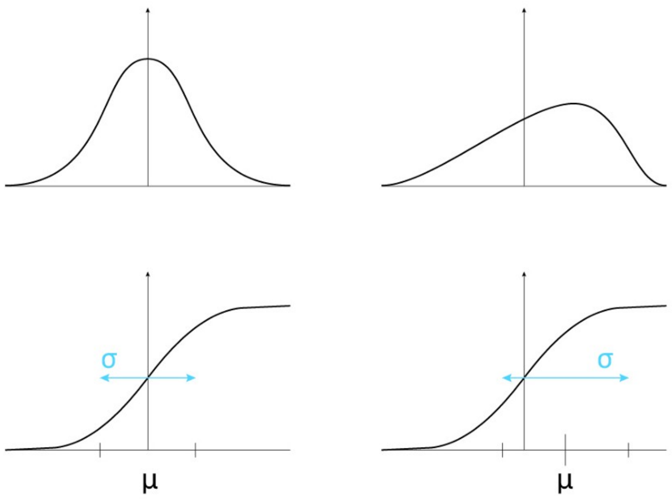


Figure 1.9: Benefits of γ and β parameters. Modifying the distribution (on the top) allows us to use different regimes of the nonlinear functions (on the bottom) from [**13**]

The network calculates the mean μ and standard deviation σ for the current batch at each iteration. When γ and β are ready, they are trained using gradient descent and an EMA to provide more weight to recent iterations.

1.2.6 Performance metrics

The following step is to determine how well a machine learning model can predict using categories. We must separate our data into a training set and a validation set in order to compute these measures.

1.2.6.1 Intersection-Over-Union

DNNs are frequently trained using simple loss functions (e.g., softmax loss). These loss functions are well suited to traditional classification problems in which overall classification accuracy is the primary objective. When it comes to picture segmentation, the two classes (foreground and background) are extremely imbalanced. The IoU method is frequently used to assess the performance of picture segmentation techniques [**14**]. Mean IoU is a typical semantic image segmentation assessment metric that computes the IoU for each semantic class before averaging over all classes. IoU is defined as follows:

$$IOU = \frac{True\ Positive}{(True\ Positive + False\ Positive + False\ Negative)}.$$

The predictions are accumulated in a confusion matrix, weighted by a variable, and the metric is then calculated.

1.2.6.2 Accuracy

Accuracy is one of the most significant factors to consider when evaluating machine learning models. Simply said, accuracy refers to our model's proportion of right predictions. The formal definition of accuracy is as follows:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

The following formula can be used to calculate accuracy if we had positive and negative numbers in a binary classification:

$$Accuracy = \frac{True\ Positives + True\ Negatives}{True\ Positives + True\ Negatives + False\ Positives + False\ Negatives}$$

1.2.6.3 Precision

Precision is a statistic that counts the number of correct positive predictions made. As a consequence, precision calculates the minority class's accuracy. It is calculated using the proportion of correctly predicted positive cases divided by the total number of positive examples expected.

$$Precision = \frac{True\ Positives}{False\ Positives + True\ Positives}$$

1.2.6.4 Recall

The recall is a statistic that counts how many valid positive predictions were made out of all the potential ones. Unlike precision, which only takes into account the most accurate positive predictions out of all positive forecasts, recall takes into account the positive predictions that were missed. In this technique, recollection provides some insight into the coverage of the positive class.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

1.2.6.5 F-Measure (F1 Score)

Because classification accuracy is a single metric used to describe model performance, it is extensively utilized. F-measure is a technique for combining precision and recall into a single metric.

Neither precision nor recall can give the complete picture on their own. We might have high precision but poor recall, or vice versa. With the F-measure, it may convey both worries with a single score.

Once precision and recall for a binary or multiclass classification problem have been determined, the two scores may be combined to calculate the F-Measure. This is how the conventional F measure is calculated:

$$F1\,Score = 2 * \frac{Precision\ *\ Recall}{Precision\ +\ Recall}$$

1.2.6.6 Jaccard similarity coefficient

The Jaccard similarity coefficient is a simple, intuitive formula that can be used for various applications, including image segmentation and other activities. Image segmentation quality is evaluated using this metric, which measures the similarity between the ground truth and segmentation results. The Jaccard similarity coefficient is defined as: let S and G denote the segmentation result and ground truth, respectively.

$$E = \frac{A(G \cap S)}{A(G \cup S)}$$

Where $A(x)$ is the operation of counting quantity. The numerator in the equation refers to the number of matching pixels or true positives[**15**]. The total number of matching and mismatched pixels is counted in the denominator.

1.2.7 Loss functions

When creating a neural network, the network tries to predict the output as similar to the existing value as possible. The loss or cost function also called the error function, is used to assess the network's accuracy. When the network makes mistakes, the cost or loss function attempts to penalize it.

While running the network, our goal is to improve prediction accuracy and reduce error, thereby minimizing the loss function. The most optimized output is the one with the lowest cost or loss function value.

The learning process is centered on reducing costs. Depending on the type of learning task, loss functions can be divided into two major categories — Classification and regression losses [**16**].

1.2.7.1 Regression losses

- Mean Square Error/Quadratic Loss/L2 Loss

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2)}{n}$$

- Mean Absolute Error/L1 Loss

$$MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n}$$

- Mean Bias Error

$$MBE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i))}{n}$$

1.2.7.2 Classification losses

- Hinge Loss/Multi class SVM Loss

$$SVMLOSS = \sum_{j \neq y_i} max(0, s_j - s_y i + 1)$$

- Cross Entropy Loss/Negative Log Likelihood

$$CrossEntropyLoss = -(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i))$$

- Focal Loss

$$FocalLoss = FL(p_t) = -(1 - p_t)^{\gamma} log(p_t)$$

The FL function for binary classification generalizes binary cross-entropy by incorporating a hyperparameter called the focusing parameter that penalizes hard-to-classify samples more strongly than easy-to-classify examples. [**17**].

1.2.8 Hyperparameters

There are various hyperparameters in DL models, and figuring out the best configuration for these parameters is difficult. Experience, as well as a lot of observation and experimenting, are required to set the hyperparameters [**3**]. Setting up hyperparameters such as gradient descent, LR, epochs, steps per epoch, and batch size is difficult. They act as controls that the model may adjust as she proceeds through her training. We must discover the perfect value for these hyperparameters in order for the model to produce the best outcomes.

1.2.8.1 Gradient descent algorithms

Gradient descent is among the most popular optimization algorithms, and it has always been the most common way to optimize neural networks [**18**].

Gradient descent has three variants that differ in how much data is used to compute the gradient of the objective function.

- Batch gradient descent
$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$
- Stochastic gradient descent
$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$
- Mini-batch gradient descent
$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$$

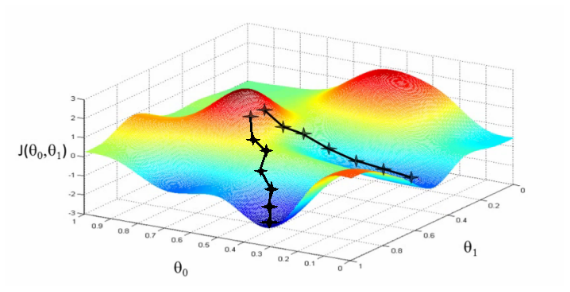


Figure 1.10: 3D Gradient Descent

However, vanilla mini-batch gradient descent does not guarantee good convergence and presents a few challenges, As a result, some algorithms are widely used by the DL community to address those challenges such as Momentum [**20**], Nesterov accelerated gradient [**21**] etc.

1.2.8.2 Learning rate

The LR is a much-needed parameter in gradient descent. As seen in the figures above (red curve), the steps are larger at first, indicating a higher LR, and as the point decreases, the LR decreases due to the smaller step size. In addition, the loss function is dropping (which is a good sign).

The learning rate is a configurable hyperparameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0 [**22**].

Multistep tuning of LR values at various stages of the training process is required for dynamic LRs, which provide high accuracy and rapid convergence. This hyperparameter's setting is a difficult balancing act between underfitting and overfitting. Textbfunderfitting occurs when a model is unable to reduce error for either the test or training set. An underfitting model cannot fit the data distributions because of their underlying complexity[**3**]. overfitting, on the other hand, occurs when a model is so powerful that it overfits the training set, increasing the generalization error. Overfitting may occur if the LR is too low. Large LRs aid in maintaining consistency in training, however excessively high LRs lead training to diverge.

In simple words, the LR is the rate at which we descend toward the cost function minima. We should choose the LR carefully because it should not be so high that the optimal solution is missed, nor should it be so low that the network takes forever to converge.

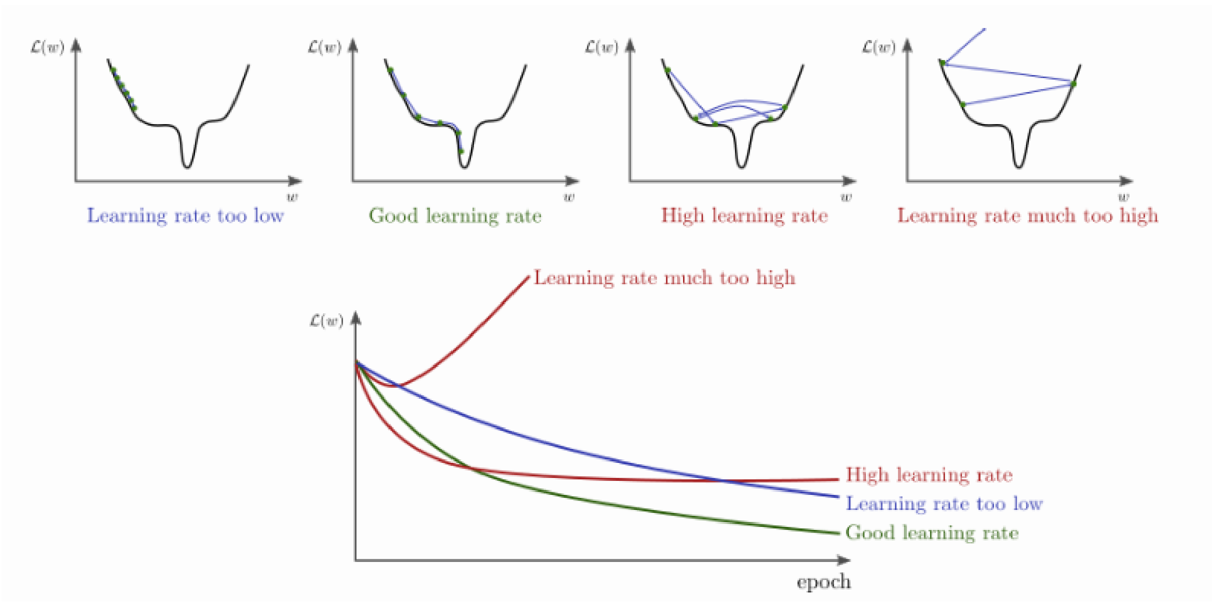


Figure 1.11: Learning Rate expletive illustration[**19**]

1.2.8.3 Batch size

The batch size is a hyperparameter that defines the number of samples to work through before updating the internal model parameters. Think of a batch as a for-loop iterating over one or more samples and making predictions. When the batch is over, the error value is calculated based on comparing the predictions and the expected output. From this error, the update algorithm is used to improve the model[**79**]. A training dataset can be divided into one or more batches.

1.2.8.4 Epochs

It is called an epoch when an entire dataset is only processed through the neural network once. We break the epoch into numerous smaller batches since one epoch is too large to provide the computer all at once. Because one epoch is insufficient for updating the weights, we employ numerous epochs. As the number of epochs grows, the weights in the neural network are modified more often, and the curve shifts from underfitting to the optimum to overfitting, as seen in (Figure 1.12). There is no set number of epochs. However, we can assume that the number of epochs is proportional to the diversity of the data[**79**].

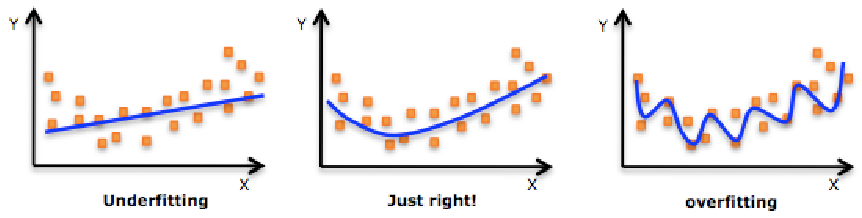


Figure 1.12: Underfitting, Optimum and Overfitting[**44**]

1.2.8.5 Steps per epoch

The number of times the training loop in the learning algorithm will run to update the parameters in the model is known as step per epoch. It will process a block of data, which is essentially a batch, at each loop iteration. The gradient descent technique is commonly used in this loop. Because this will use all the data points, one batch size worth at a time, the steps per epoch are traditionally calculated as train length divided by the batch size. In the case of augmented data, we multiply the previous operation by 2 or 3, and so on. However, if the training has been going on for too long, we'll just keep to the old method[**79**].

1.2.9 Forward propagation and backpropagation

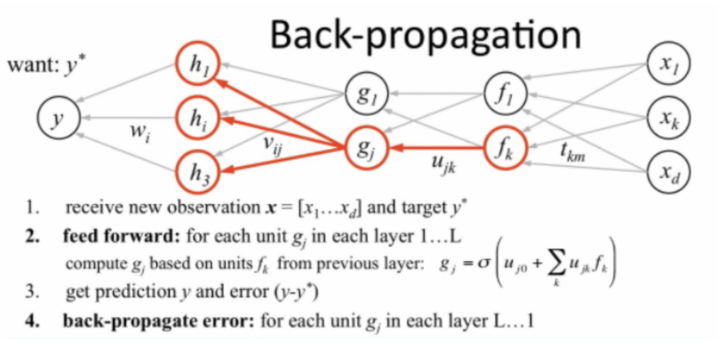
An analogy may help to understand brain network mechanisms. There are many similarities between neural network learning and human learning, such as learning in our daily lives and activities. For example, when we do an action and receive feedback from a trainer, we improve our performance. A trainer is also needed to explain what the output should have been when it comes to neural networks. Based on this discrepancy between actual and anticipated values, a cost function error value is calculated and sent back to the system. Cost functions are assessed and utilized to change thresholds and weights for the following input in the network at each layer. As a team, we are working to reduce the cost function. The closer the projected value to the actual value is the lower the cost function. In this approach, the network improves at analyzing values, and the error decreases over time. The results are fed back into the neural network and reprocessed. We can control the weighted synapses that connect input variables to the neuron. Adjusting the weights is necessary if there is a discrepancy between the actual and projected values[**23**]. If we tweak the parameters and repeat the neural network, a new cost function will be created, hopefully, less than the previous one. We must continue this approach in order to reduce the cost function to the lowest size achievable. Back-propagation is a method that is continually deployed throughout a network to maintain the error number as low as feasible[**23**].

1.2.9.1 Forward propagation

The input flow via the hidden layers to the output layers is referred to as forward propagation. It is the movement of information in only one direction. The layer input will provide information to hidden layers, generating output that keeps moving in the same direction without going backward.

1.2.9.2 Backpropagation

Backpropagation of error is a technique used to train feed-forward Neural network training relies on back-propagation to do its task. It is a technique for optimizing neural network weights based on the previous epoch's error rate (i.e., iteration). By modifying the weights, we may reduce error rates and improve model generalization[**24**].



1. receive new observation $\boldsymbol{x} = [x_1 \ldots x_d]$ and target y^*
2. **feed forward:** for each unit g_j in each layer 1…L
   compute g_j based on units f_k from previous layer: $g_j = \sigma\left(u_{j0} + \sum_k u_{jk} f_k\right)$
3. get prediction y and error $(y\text{-}y^*)$
4. **back-propagate error:** for each unit g_j in each layer L…1

Figure 1.13: Backpropagation expletive illustration

In neural networks, "backward propagation of mistakes" is known as "back-propagation." It is a common practice in artificial neural network training. A loss function's gradient can be calculated using this method for all network weights. This algorithm uses the chain rule to compute a single weight loss function gradient using back-propagation. A feed-forward neural network's weight-space gradient concerning a loss function is computed using back-propagation. The chain rule is crucial in back-propagation. Here is a partial differentiation of loss (L) in terms of weights/parameters (w).

1.2.10 Data augmentation

data augmentation is the process of applying a sequence of deformations to a set of labeled training data to generate more diverse and extra training data. IT is needed to teach the network the required invariance, and resilience [**25**]. Geometric transformations (flipping and rotation, clipping and scaling), color space transformations (alteration of RGB channel intensities), kernel filters, mixing images, random erasing, adversarial training, neural style transfer, noise injection, and meta-learning schemes are just a few of the data augmentation techniques that have been proposed. The most fundamental premise of data augmentation is that the deformations used should not modify the labels' semantic meaning [**26**][**27**].

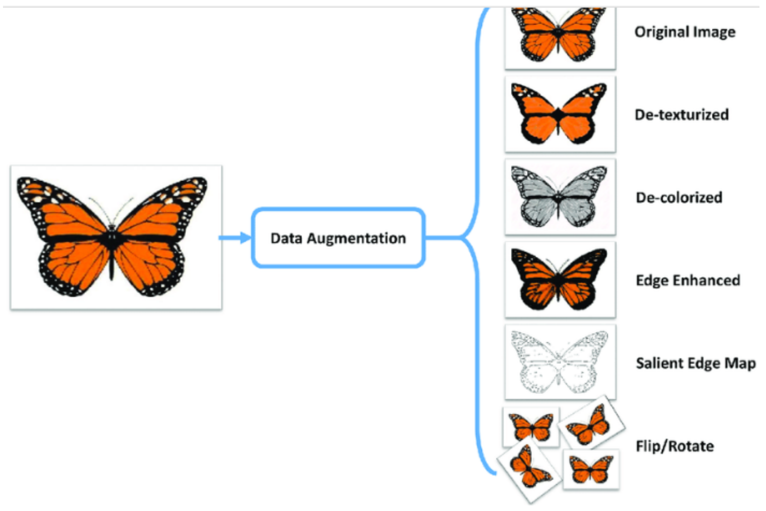


Figure 1.14: Representation of data augmentation[**3**]

1.3 Basic deep learning architectures

1.3.1 Convolutional neural networks

DL algorithms such as Convolutional Neural Networks (CNNs or ConvNets) can distinguish between distinct aspects and objects in an image and then use that knowledge to create new images based on that information [113]. There is far less pre-processing necessary when using a CNN than other classification methods[**28**]. CNN can learn certain filters/characteristics if they are given enough training.

A CNN's architecture is similar to the connectivity pattern of neurons in the human brain and was inspired by the visual cortex's arrangement. The receptive field is the area of the visual field in which individual neurons respond to stimuli. A collection of these fields covers the entire visual field. A CNN can capture the spatial and temporal dependencies in an image through relevant filtering techniques. The reduced number of parameters and reusability of weights allow the architecture to fit the picture collection better. In other words, the network may be trained to comprehend the image's complexity better. Figure 4.8 displays a generic CNN architecture [**30**].

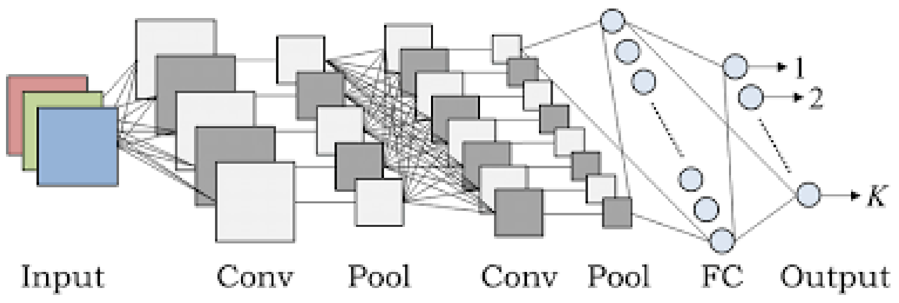


Figure 1.15: Architecture of a simple CNN from [**41**]

Convolutional Layers The convolutional layer is the fundamental block, as it is responsible for most of the computations. It comprises three components: input data, a filter, and a feature map. The kernel or filter traverses the image's receptive fields, checking for the presence of features; this is referred to as convolution. The kernel is a two-dimensional (2-D) weighted array representing a portion of the image. While filter sizes vary, they are commonly a 3×3 matrix; this also dictates the size of the receptive field. The dot product of the filter and the portion of the input array is calculated and loaded into an output array. Following that, the filter shifts by one stride, and the procedure is repeated until the kernel has swept across the entire image[**31**][**29**]. A feature map, activation map, or convolved feature is the ultimate result of a series of dot products from the input and the filter.

As illustrated , the convolution procedure does not require that each output pixel in the feature map be connected to each pixel value in the input array. For that, convolutional layers are frequently referred to as partially-connected layers. The filter weights remain constant as it traverses images, a phenomenon is known as parameter sharing. Specific parameters, such as the weight values, are adjusted during training using back-propagation and gradient descent. However, three hyper-parameters must be set before the neural network training begins. Among them are [**31**].

1) The number of filters: It affects the output's depth. n distinct filters, for example, would result from n different feature maps.

2) The stride parameter: specifies the distance or the number of pixels that the filter moves across the input array.

3) Zero-padding is typically used when the filters do not fit the input image; this reduces the size of all items outside the input matrix to zero, resulting in a larger or equal-sized output.

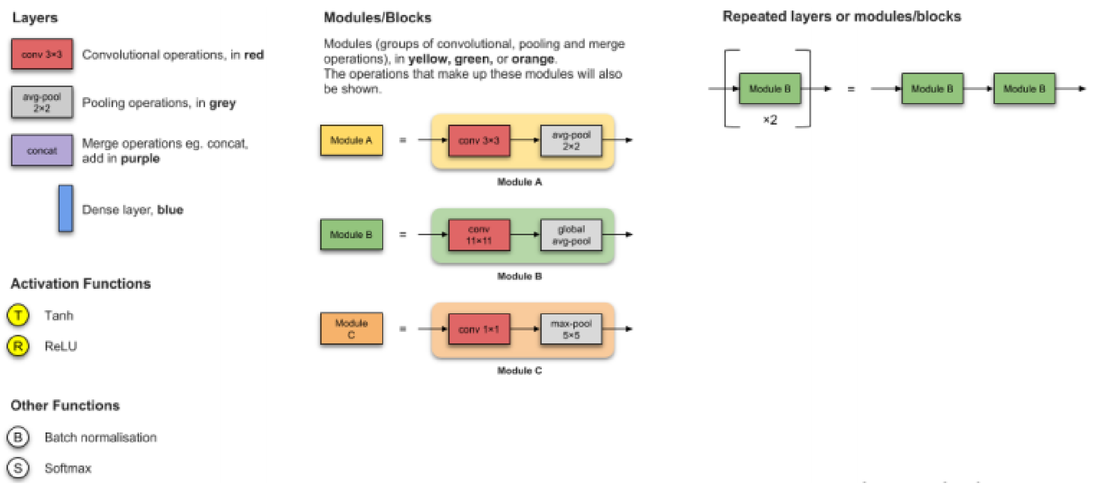
1.3.2 Common convolutional neural network architectures

As we previously stated, certain deep networks have made such significant contributions to the field that they have become widely known standards. It is the case of Le Net-5,AlexNet, VGG-16, GoogLeNet, and ResNet. Such was their importance that they are currently being used as building blocks for many segmentation architectures [**49**]. For that reason, we will devote this section to review them see (figure 1.16) .

We will use the following mathematical equation to calculate the output of convolutional layers:

$$\left[\frac{n+2p-f}{s}+1\right] * \left[\frac{n+2p-f}{s}+1\right]$$

Where we consider a $n * n$ image as an input, a $f * f$ filter, a padding p, and a stride s.



Layers
conv 3×3
Convolutional operations, in red
avg-pool 2×2
Pooling operations, in grey
concat
Merge operations eg. concat, add in purple
Dense layer, blue
Activation Functions
T Tanh
R ReLU
Other Functions
B Batch normalisation
S Softmax
Modules/Blocks
Modules (groups of convolutional, pooling and merge operations), in yellow, green, or orange.
The operations that make up these modules will also be shown.
Module A
conv 3×3
avg-pool 2×2
Module B
conv 11×11
global avg-pool
Module C
conv 1×1
max-pool 5×5
Repeated layers or modules/blocks
×2

Figure 1.16: Legend used for the various architectures from [52]

1.3.2.1 LeNet-5

LeNet-5 is a pioneering 7-level convolutional network develope[54] it is one of the earliest pre-trained models proposed by Yann LeCun and others in the year 1998, in the research paper Gradient-Based Learning Applied to Document Recognition. They used this architecture for recognizing the handwritten and machine-printed characters [53].

The main reason behind the popularity of this model was its simple and straightforward architecture. It is a multi-layer convolution neural network for image classification. The input to this model is a 32 X 32 grayscale image hence the number of channels is one[53] see (Table 1.1).

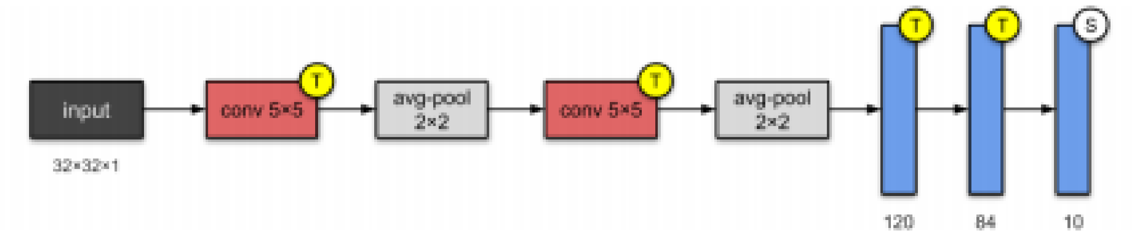


Figure 1.17: LeNet architecture from [52]

Layer		Feature Map	Size	Kernel Size	Stride	Activation	Parameters
Input	Image (Grayscale)	1	32x32	-	-	-	-
1	Convolution	6	28x28	5x5	1	tanh	156
2	Average Pooling	6	14x14	2x2	2	tanh	0
3	Convolution	16	10x10	5x5	1	tanh	2416
4	Average Pooling	16	5x5	2x2	2	tanh	0
5	Convolution	120	1x1	5x5	1	tanh	48120
6	FC	-	84	-	-	tanh	10164
Output	FC	-	10	-	-	softmax	850
Total number of parameters							61,706

Table 1.1: LeNet structural details

1.3.2.2 AlexNet

AlexNet was the pioneering deep CNN that won the ILSVRC-2012 with a TOP-5 test accuracy of 84.6The architecture presented by Krizhevsky et al[**55**].The network's architecture was quite similar to LeNet, but it was deeper, with 62M parameters and more filters per layer and layered convolutional layers was relatively simple. It consists of five convolutional layers, max-pooling ones, Rectified Linear Units (ReLUs) as non-linearities, three fully-connected layers.[**56**] see (Table 1.2)

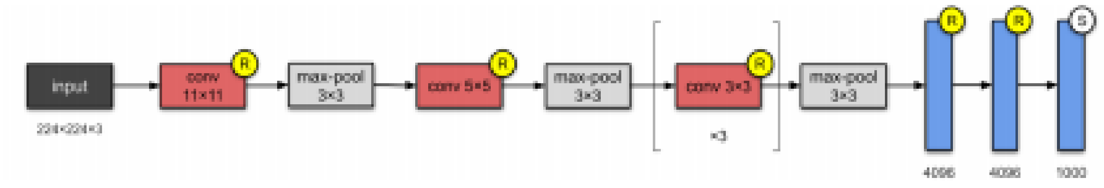


Figure 1.18: AlexNet architecture from [**52**]

Layer		Feature Map	Size	Kernel Size	Stride	Activation	Parameters
Input	Image (RGB)	1	227x227x3	-	-	-	-
1	Convolution	96	55x55x96	11x11	4	relu	34944
	Max Pooling	96	27x27x96	3x3	2	relu	0
2	Convolution	256	27x27x256	5x5	1	relu	614656
	Max Pooling	256	13x13x256	3x3	2	relu	0
3	Convolution	384	13x13x384	3x3	1	relu	885120
4	Convolution	384	13x13x384	3x3	1	relu	1327488
5	Convolution	256	13x13x256	3x3	1	relu	884992
	Max Pooling	256	6x6x256	3x3	2	relu	0
6	FC	-	4096	-	-	relu	37752832
7	FC	-	4096	-	-	relu	16781312
Output	FC	-	1000	-	-	softmax	4097000
Total number of parameters							62,378,344

Table 1.2: AlexNet structural details

1.3.2.3 VGG-16

It is a Convolutional Neural Network (CNN) model proposed by Karen Simonyan and Andrew Zisserman at the University of Oxford. The idea of the model was proposed in 2013, but the actual model was submitted during the ILSVRC ImageNet Challenge in 2014 [**57**]. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was a large-scale competition that assessed picture categorization (and object identification) systems. They fared well in the competition but were unable to win.

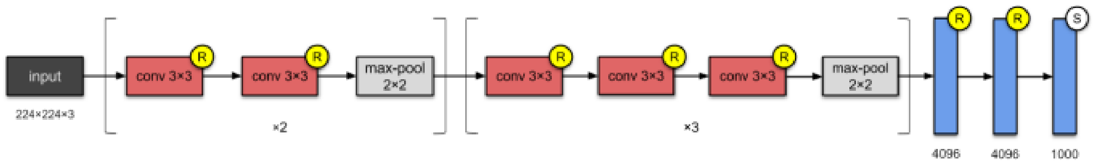


Figure 1.19: VGG-16 architecture from [**52**]

The distinction between VGG and AlexNet is that it has numerous characteristics that set it apart from other competing models:

- Rather than using big receptive fields like AlexNet (11x11 with a stride of 4), VGG employs extremely small receptive fields (3x3 with a stride of 1). The decision function is much more discriminative now that there are three ReLU units instead of simply one. There are also fewer parameters (27 channels as opposed to AlexNet's 49 channels).

- VGG employs 1x1 convolutional layers to increase the nonlinearity of the decision function without modifying the receptive fields.

- Because of the small dimensions of the convolution filters, VGG can have a large number of weight layers.

Layer		Feature Map	Size	Kernel Size	Stride	Activation	Parameters
Input	Image (RGB)	1	224x224x3	-	-	-	-
1	2xConvolution	64	224x224x64	3x3	1	relu	38720
	Max Pooling	64	112x122x64	2x2	2	relu	0
3	2xConvolution	128	112x122x128	3x3	1	relu	221440
	Max Pooling	128	56x56x128	2x2	2	relu	0
5	3xConvolution	256	56x56x256	3x3	1	relu	1475328
	Max Pooling	256	56x56x256	2x2	2	relu	0
7	3xConvolution	512	28x28x512	3x3	1	relu	5899776
	Max Pooling	512	14x14x512	2x2	2	relu	0
10	3xConvolution	512	14x14x512	3x3	1	relu	7079424
	Max Pooling	512	7x7x512	2x2	2	relu	0
14	FC	-	4096	-	-	relu	102764544
15	FC	-	4096	-	-	relu	16781312
Output	FC	-	1000	-	-	softmax	4097000
Total number of parameters							134,264,641

Table 1.3: VGG-16 structural details using padding = 1

1.3.2.4 ResNet

Microsoft's ResNet [58] is specially remarkable thanks to winning ILSVRC-2016 accuracy. Apart from that fact, the network is well-known due to its depth (152 layers) and the introduction of residual blocks . The residual blocks solve the challenge of training a deep architecture by incorporating identity skip links, which allow layers to replicate their inputs to the next layer.

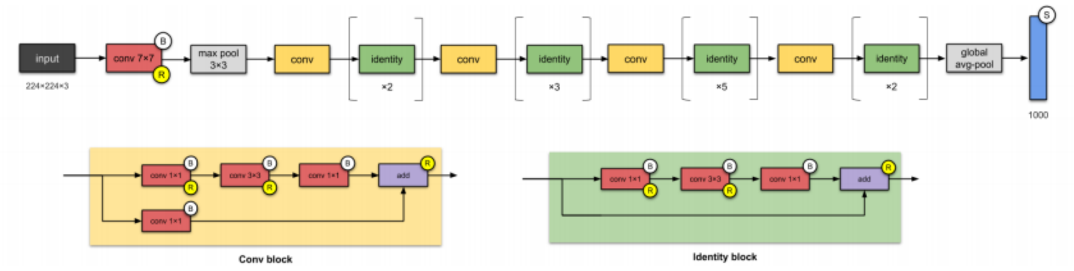


Figure 1.20: ResNet-50 architecture from [52]

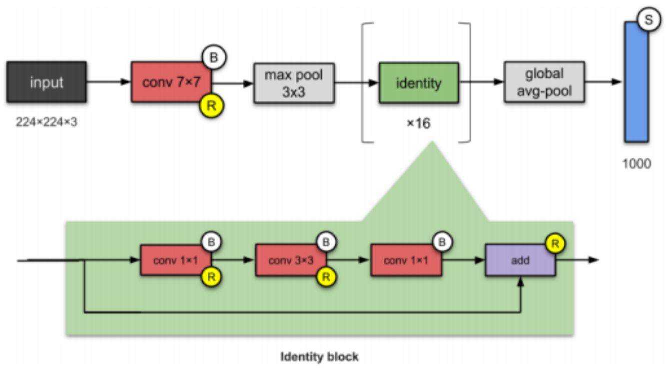


Figure 1.21: ResNet identity block from [52]

The intuitive idea behind this approach is that it ensures that the next layer learns something new and different from what the input has already encoded (since it is provided with both the output of the previous layer and its unchanged input)[56]. Furthermore, such linkages aid in the resolution of the vanishing gradients problem.

Layer	Input size	Output size	Filter	Parameters
Convolution1	224x224x3	112x122x64	7x7x64, stride = 2	9472
3xConvolution2 (Convolution Block + 2x(Identity block))	112x122x64	56x56x64	3x3 Max pooling, stride = 2	0
			1x1x64 3x3x64 1x1x256	214800
4xConvolution3 1x(Convolution Block) + 3x(Identity block)	56x56x64	28x28x128	1x1x128 3x3x128 1x1x512	1216000
6xConvolution4 1x(Convolution Block) + 5x(Identity block)	28x28x128	14x14x256	1x1x256 3x3x256 1x1x1024	7088128
3xConvolution5 1x(Convolution Block) + 2x(Identity block)	14x14x256	7x7x512	1x1x512 3x3x512 1x1x2048	14953472
FC	7x7x512	1x1x1000	Average pooling, 1000-d FC, Softmax	2049000
Total number of parameters				25,636,712

Table 1.4: ResNet structural details

1.3.2.5 GoogleNet

Google Net (or Inception V1) was proposed by research at Google (with the collaboration of various universities) in 2014 in the research paper titled "Going Deeper with Convolutions". This architecture was the winner at the ILSVRC 2014 image classification challenge.[**59**] It has a much lower error rate than previous winners AlexNet (Winner of ILSVRC 2012) and ZF-Net (Winner of ILSVRC 2013), as well as a significantly lower error rate than VGG (2014 runner up). 1×1 convolutions in the centre of the design and global average pooling are used in this architecture.

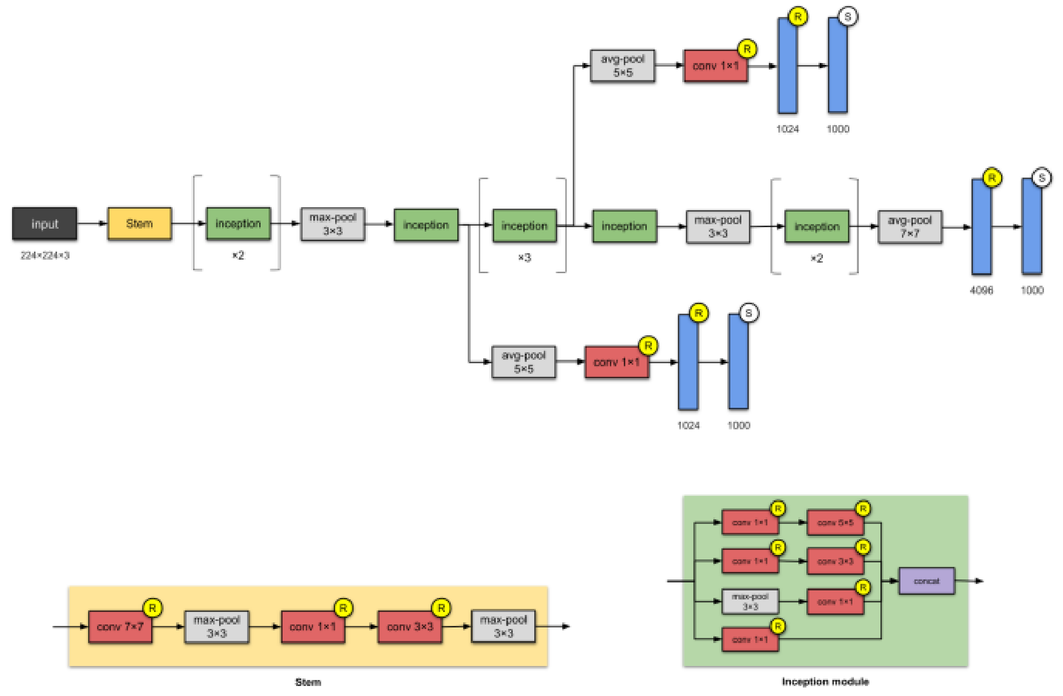


Figure 1.22: GoogleNet architecture (Inception V1) from [**52**]

The GoogLeNet architecture is very different from previous state-of-the-art architectures such as AlexNet . It uses many different kinds of methods such as 1×1 convolution and global average pooling that enables it to create deeper architecture [**59**]. GoogLeNet contains about 6.8 million parameters (without auxiliary layers), which is 9 times less than AlexNet and 20 times less than VGG-16.

Type	Patch size / Stride	Output size	Depth	#1x1	#3x3 Reduce	#3x3	#5x5 Reduce	#5x5	Pool Proj	Parameters
Convolution	7x7/2	112x112x64	1	-	-	-	-	-	-	2.7K
Max Pooling	3x3/2	56x56x64	0	-	-	-	-	-	-	-
Convolution	3x3/1	56x56x192	2	-	64	192	-	-	-	112K
Max Pooling	3x3/2	28x28x192	0	-	-	-	-	-	-	-
Inception (3a)	-	28x28x256	2	64	96	128	16	32	32	159K
Inception (3b)	-	28x28x480	2	128	128	192	32	96	64	380K
Max Pooling	3x3/2	14x14x480	0	-	-	-	-	-	-	-
Inception (4a)	-	14x14x512	2	192	96	208	16	48	64	364K
Inception (4b)	-	14x14x512	2	160	112	224	24	64	64	437K
Inception (4c)	-	14x14x512	2	128	128	256	24	64	64	463K
Inception (4d)	-	14x14x528	2	112	144	288	32	64	64	580K
Inception (4e)	-	14x14x832	2	256	160	320	32	128	128	840K
Max Pooling	3x3/2	7x7x832	0	-	-	-	-	-	-	-
Inception (5a)	-	7x7x832	2	256	160	320	32	128	128	1072K
Inception (5b)	-	7x7x1024	2	384	192	384	48	128	128	1388K
Average Pooling	7x7/1	1x1x1024	0	-	-	-	-	-	-	-
Dropout (40%)	-	1x1x1024	0	-	-	-	-	-	-	-
Linear	-	1x1x1000	1	-	-	-	-	-	-	1000K
Softmax	-	1x1x1000	0	-	-	-	-	-	-	-
Total number of parameters										6,8M

Table 1.5: GoogleNet structural details

1.3.2.6 MobileNet

MobileNet is a CNN architecture that is both efficient and portable and is employed in real-world applications. To develop lighter models, MobileNets typically employ depthwise separable convolutions instead of the usual convolutions used in previous designs. MobileNets adds two new global hyperparameters (width multiplier and resolution multiplier) that let model creators trade off latency or accuracy for speed and small size, depending on their needs.

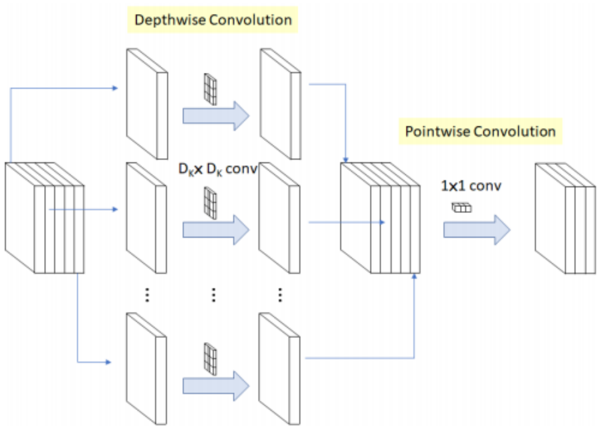


Figure 1.23: MobileNet architecture from

Convolution layers that are depthwise separable are used to build MobileNets. A depthwise convolution and a pointwise convolution are included in each depthwise separable convolution layer. A MobileNet contains 28 layers if you count depthwise and pointwise convolutions separately. The width multiplier hyperparameter can be adjusted to reduce the number of parameters in a conventional MobileNet to 4.2 million. The size of the input image is $224 \times 224 \times 3$. The detailed architecture of a MobileNet is given below :

Layer	Input size	Stride	Filter	Parameters
Convolution1	224x224x3	2	3x3x3x32	992
Convolution_dw1	112x112x32	1	3x3x32 dw	416
Convolution_pw1	112x112x32	1	1x1x32x64	2304
Convolution_dw2	112x112x32	2	3x3x64 dw	832
Convolution_pw2	56x56x64	1	1x1x46x128	8704
Convolution_dw3	56x56x128	1	3x3x128 dw	1664
Convolution_pw3	56x56x128	1	1x1x128x128	16896
Convolution_dw4	56x56x128	2	3x3x128 dw	1664
Convolution_pw4	56x56x128	1	1x1x128x256	33792
Convolution_dw5	56x56x256	1	3x3x256 dw	3328
Convolution_pw5	56x56x256	1	1x1x256x256	66560
Convolution_dw6	56x56x256	2	3x3x256 dw	3328
Convolution_pw6	14x14x256	1	1x1x256x512	133120
5xConvolution_dw7	14x14x512	1	3x3x512 dw	33280
5xConvolution_pw7	14x14x512	1	1x1x512x512	1320960
Convolution_dw8	14x14x512	2	3x3x512 dw	6656
Convolution_pw8	7x7x512	1	1x1x512x1024	528384
Convolution_dw9	7x7x1024	2	3x3x1024 dw	13312
Convolution_pw10	7x7x1024	1	1x1x1024x1024	1052672
Average pooling	7x7x1024	1	7x7x1024	0
FC	7x7x1024	1	1024x1000	1025000
Softmax	1x1x1000	1	-	0
Total number of parameters				4,253,864

Table 1.6: MobileNet structural details

1.3.3 Pooling Layer

is a technique for lowering input dimensionality by reducing the number of factors. The pooling operation sweeps an unweighted filter across the entire input using an aggregation function to populate the output array with the values contained within the receptive field. It can be classified into two broad categories [**32**]:

• Pooling to a maximum: The filter traverses the input array. It selects the pixel with the highest value for the output array.

• Average pooling: The filter determines the average value contained inside the receptive field. While the pooling layer loses much information, it also provides several benefits for the CNN. They contribute to the reduction of complexity, the enhancement of efficiency, and the avoidance of over-fitting [**33**].

1.3.4 Fully-Connected Layer

The Full-Connected (FC) layer is what its name implies. Layers that are only partially connected have no direct connection between the input image and the output layer. Nevertheless, in a completely interconnected layer, every node in the output layer is directly linked to a node in the previous layer [**34**]. Classification is done here using the information from the previous layers and their various filters. FC layers often employ a softmax activation function to categorize inputs adequately, whereas convolutional and pooling layers typically use ReLu functions, resulting in a probability ranging from zero to one.

1.3.5 Recurrent neural networks and the LSTM

We focus on the RNN first, because the LSTM network is a type of an RNN, Recurrent neural networks are one of the powerful models for data. RNNs can use their internal state (memory) to process sequences of inputs.

RNNs are used in many fields such as speech recognition, machine translation, music composition, gamma learning, stock prediction, self-driving cars, to name a few.
RNNs are widely used in the following domains/ applications:
Prediction problems;Language Modelling and Generating Text; Machine Translation;Speech Recognition;Generating Image Descriptions;Video Tagging and Other applications like Music composition
RNN model consists of a system of three layers: an input layer, a hidden layer (the core network), and an output layer;the hidden layer or middleware is the layer that contains a complex form of neural networks ,that are interconnected through weighted synapses (connection weights),The input and output layers are connected to the hidden layer, again through weighted synapses.. (Figure 1.24).

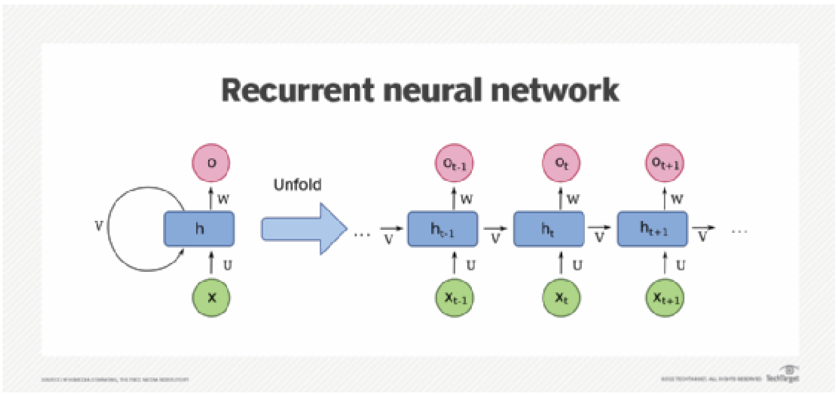


Figure 1.24: Architecture of a simple RNN

But we have e problem that RNNs have a very short-term memory, and they will have terrible results with long sequences and often suffer from problems such as vanishing/exploding gradients. what is LSTM or in the sense the Long Short-Term Memory network?,this last one was the key to avoid these problems, with an appropriate gradient-based learning algorithm[**35**].
The architecture of LSTM includes three gates (input gate, output gate, forget gate)enforcing constant error flow through internal states of special units called memory cells that store values for arbitrary time intervals, see (Figure 1.25)

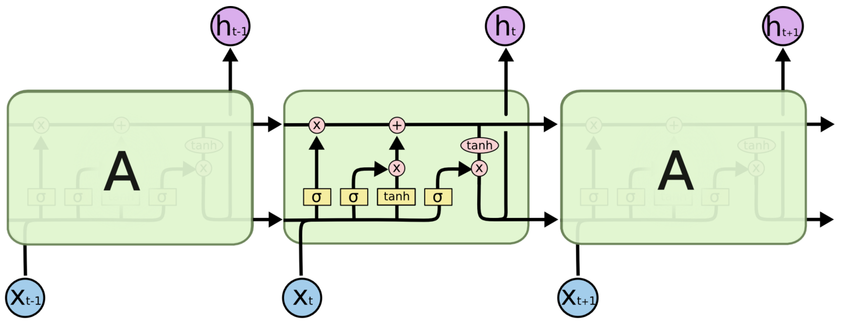


Figure 1.25: Architecture of LSTM from [**38**]

1.3.6 Encoder-Decoder and Auto-Encoder Models

A decoder in data processing retrieves the information in its original form, whereas an encoder can compress digital information for transport or storage.

The encoder compresses the information contained in the four inputs into two outputs. The inputs are light of four different wavelengths that photoisomerize the fulgimide, dithienylethene, or both. The outputs are absorbance at two wavelengths[**36**].

Excitation at two wavelengths is used as a decoder input, while absorbance at two wavelengths, transmittance at one wavelength, and fluorescence emission are used as outputs to retrieve the information compressed within the inputs.

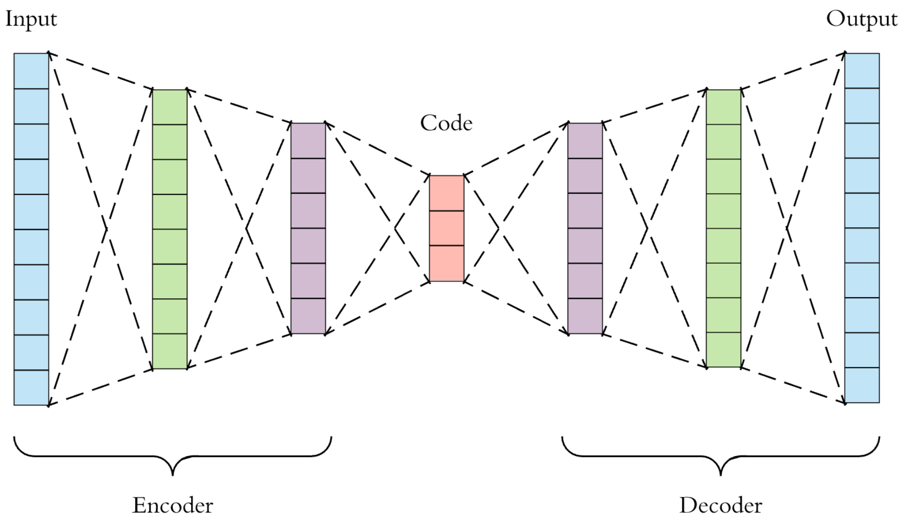


Figure 1.26: Architecture of an Encoder-Decoder from [**37**]

Auto-encoders are a specific type of feedforward neural networks where the input is the same as the output. 1.27).

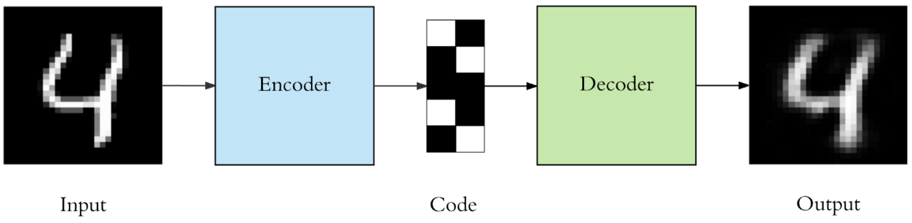


Figure 1.27: an exemple of Encoder-Decoder from [**37**]

An auto-encoder consists of 3 components: encoder, code and decoder. the encoder takes a sequence as the input and compresses it into a fixed-length numeric vector (feature), the decoder then predicts the output from that vector see (Figure 1.27).

An AE is a particular variant of the ED models with a distinguished change. The number of neurons is the same in the input and the output (called reconstructed input). Therefore we can expect that the input and the output to be the same sequence of data.

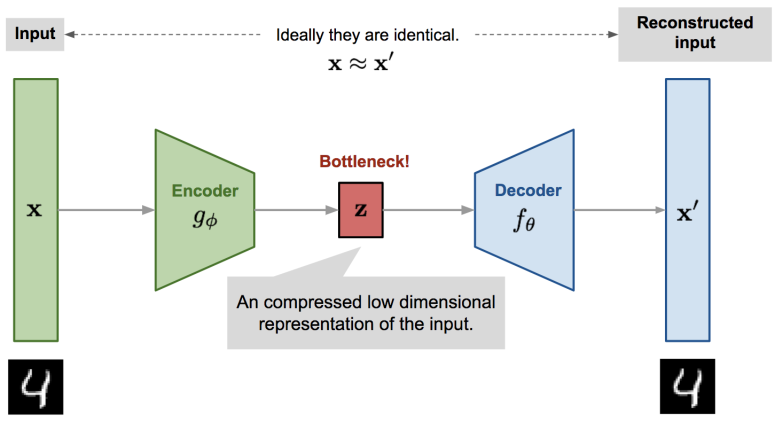

Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
$\mathbf{x}$
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
$\mathbf{x}'$
An compressed low dimensional representation of the input.

Figure 1.28: Architecture of Auto-Encoder from

1.3.7 Generative Adversarial Networks

GANs, or Generative Adversarial Networks are one of the most modern DL models. The use of the GAN for conditionally producing an output is a significant extension.

The generative model may be trained to produce new instances from the input domain, where the input, a random vector from the latent space, is (conditionally) given.

In the case of creating photos of handwritten numbers, the extra input may be a class value, such as male or female in the case of generating photographs of humans, or a digit in the case of generating photographs of handwritten digits[**27**].

Generative adversarial networks may be enlarged to a conditional model if both the generator and the discriminator are conditioned on some extra information y, such as class labels or input from other modalities. Conditioning can be accomplished in a number of ways.

One of the many major advancements in the use of deep learning algorithms in disciplines like computer vision is a method known as data augmentation.

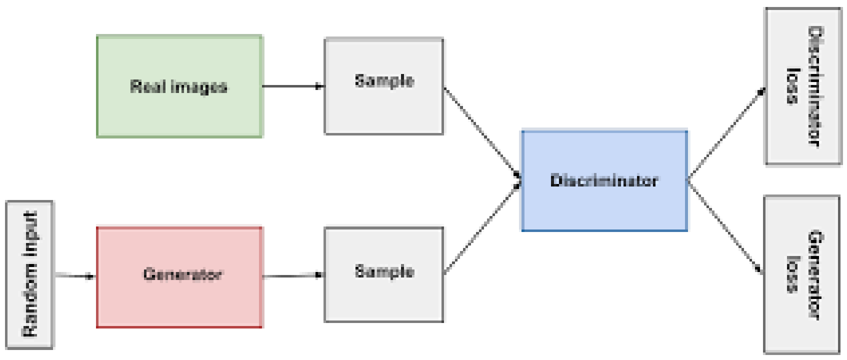


Figure 1.29: structure of GANs

They are made up of two key elements: a generator network **G** and a discriminator network **D**. It works by feeding a random noise vector z into the system.to **G** so that it may utilize it to create fictitious samples to feed to**D** Meanwhile, real life continues. **D** is given photos, and its duty is to try to discern between actual and fraudulent data . If discriminator **D** fails to discriminate between actual and bogus data,[**37**] then we receive ModelG, a well-trained generator that has learnt the distribution of real numbers.

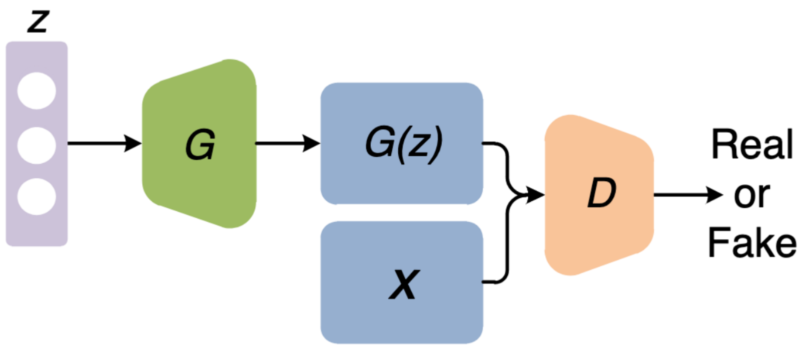


Figure 1.30: Architecture of Generative Adversarial Networks

1.4 Deep learning frameworks

1.4.1 TensorFlow

TensorFlow, a Google open-source platform, is likely the most widely used tool for Machine Learning and Deep Learning. TensorFlow is a JavaScript-based framework that includes a variety of tools and community resources for quickly training and deploying machine learning and deep learning models. Learn more about the best deep learning software [**38**]. TensorFlow is the greatest tool for creating Deep Learning models and experimenting with architectures.

It is used to do data integration tasks, such as combining graphs, SQL tables, and photos.

1.4.2 PyTorch

PyTorch is a complete machine learning platform with a user-friendly front-end, distributed training, and an ecosystem of tools and modules that allow for rapid, modular exploration and development[**39**]. PyTorch has a rich ecosystem of software and libraries for expanding it and promoting growth in fields ranging from machine vision to reinforcement learning, thanks to an engaged group of researchers and developers. PyTorch is well-supported on major cloud platforms, allowing for frictionless deployment and scaling through pre-configured images, wide GPU training, and the ability to execute models in a manufacturing environment, among other features.

1.4.3 Keras

Another open-source Deep Learning framework on our list is Keras, Keras is a human-centric API, not a machine-centric one. Keras adheres to best practices for minimizing cognitive burden, such as providing reliable and easy APIs[**40**], decreasing the amount of user activities required for common use cases, and giving clear and meaningful error messages It includes several documentation and developer instructions.

keras is excellent for beginners who have just started their journey in this field. It allows for easy learning and prototyping simple concepts.

Keras provides the opportunity to do new experiments, test more hypotheses, and do better than the competition. Keras is an industrial-strength architecture built on top of TensorFlow 2.0 that can scale to massive clusters of GPUs or an entire TPU pod. It is not only feasible; it is also easy.

Keras is a crucial component of the TensorFlow 2.0 ecosystem, and it covers every aspect of the machine learning process, from data management to hyperparameter training to deployment solutions.

Keras is used by CERN, NASA, the National Institutes of Health, and several other research institutions worldwide. it has the low-level stability to execute any research hypothesis while still providing high-level convenience functionality to shorten experimentation intervals.

Keras is the DL solution of choice for many university courses due to its ease of use and emphasis on user experience. It is generally viewed as one of the most effective methods for studying DL.

1.5 Hardware used in deep learning

1.5.1 Central processing units

The majority of deep learning models now run on GPU, with the CPU being used solely for data prepossessing. If you're working with terabytes of data and running analytics queries, you'll need a powerful CPU. They have a lot of programmability and can handle a lot of different workloads. In comparison to other CPU manufacturers, Intel is the most prominent (ARM,AMD,IBM POWER,ORACLE SPARC,Fujitsu etc). In datacenters, Intel's Xeon and Xeon Phi are examples, while Qualcomm's Snapdragon in mobile devices are instances of CPUs,any CPUs, including AMD Ryzen 9 3900X, Intel Core i9-9900K, AMD Ryzen Threadripper 3990X, AMD Ryzen 5 2600, are currently considered the most appropriate when it comes to training DL models[**41**].

A CPU is a computer that consists of numerous ALUs, a Control Unit that controls particular ALUs, a Cache Memory, and DRAM. Computers can accomplish every work with precision and variety because CPUs are more powerful. The CPU's memory power, which may surpass 1TB of RAM, provides this adaptability by allowing it to fetch memory packages in the RAM quicker and with reduced latency. Unlike GPUs, CPUs do not yet match the DL's strict requirements. CPUs, on the other hand, help GPUs prepare data by giving them adequate data and reading/writing files from/to RAM/HDD.

1.5.2 Graphics processing units

GPUs are the most extensively utilized machine learning and neural network hardware. These were created with high parallelism and memory bandwidth in mind. GPUs are thought to be the greatest choice for training [**42**]. This is used in 2D or 3D graphics rendering to speed up a lot of MAC (Multiply and accumulate) processes.

GPUs are graphics processors that create polygon-based computer graphics. GPUs have acquired huge computing powers in recent years, because of the complexity and desire for realism in recent video games and graphic engines [**42**]. NVIDIA is the industry leader, with processors that have hundreds of cores and are designed to compute at about 100% efficiency. These processors can also conduct neural network calculations and matrix multiplications, as it turns out.

GPUs are currently the standard for training DL systems, whether they are CNNs (CNN) or RNNs (RNN). In only a few milliseconds, they will practice on massive batches of images, such as 128 or 256 images.

However, they absorb 250 W and need a full PC with an additional 150 W of power to run. A greater GPU system can use up to 400 watts of power.

1.5.3 Tensor processing units

Google's Tensor Processing Unit came in 2015 as one of the specialized architecture for Machine Learning and AI applications. It focus on fast inference via high throughput 8-bit arithmetic.Most of the chip is dedicated to perform dense-matrix multiplication, and it uses 32-bit accumulator to sum up the result. Google supports running tensorflow code on TPUs, this makes it easy to train and infer deep neural networks.You can run your own programs on TPUs on google cloud.

From Google-designed ASICs we find TPUs that accelerate machine learning caseloads[**43**]. TPUs were created from the ground up with the considerable machine learning know-how and guidance of Google. Cloud TPU tools speed up linear algebra computing, which is widely utilized in machine learning applications.

While training large, complex neural network models, TPUs decrease the time-to-accuracy. Models that took weeks to master on other hardware platforms will now take only hours to learn. TPUs in the cloud are best for specific workloads. You might want to use GPUs or CPUs on Compute in various situations. In some cases, you might want to run machine learning tasks on Compute Engine instances with GPUs or CPUs. In general, you should choose equipment that is suited for the job at hand.

1.6 Conclusion

We explored the most common deep learning concepts and delved deeper into their architectures and the main frameworks used by the community. In the next chapter, we will examine applications in computer vision, specifically semantic image segmentation.

Chapter 2

Semantic Segmentation

2.1 Introduction

In computer vision, semantic segmentation is considered a challenging task. DL techniques have immensely enhanced the performance of semantic segmentation in recent years. Several innovative methods have been proposed, making segmentation algorithms more efficient and precise, and various new applications have widely used them.
In this chapter, we will go over some of the most typical CNN architectures by giving information about its production and content. We will also go through the state-of-the-art methods in DL-based semantic image segmentation. These methods include FCN, SegNet, DeepLab, and U-net.

2.2 General concepts of semantic segmentation

2.2.1 Definition

Semantic segmentation is the process of classifying each pixel belonging to a particular label. It doesn't differ across different instances of the same object [**46**]. Semantic segmentation plays a vital role in computer vision. For example, if an image contains two women, semantic segmentation assigns the same label to all the pixels for both women see figure 2.1.

Handwriting recognition, medical imaging, autonomous driving, industrial robotics, indoor navigation, virtual or augmented reality systems, portrait mode in modern smartphones, and even social media filters and virtual make-up are just a few of the innovative applications that have evolved from this discipline.

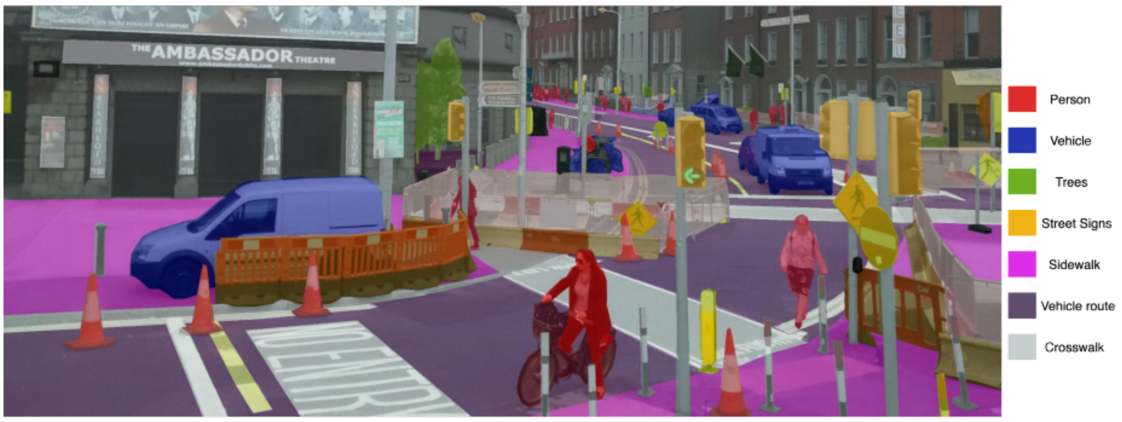


Figure 2.1: An example of semantic segmentation[**46**]

2.2.2 Comparison with other computer vision tasks

Semantic segmentation differentiates itself from other familiar computer vision tasks like image classification, object detection, and instance segmentation.

The most fundamental building block in Computer Vision is the Image classification problem were given an image, we expect the computer to output a discrete label, which is the main object in the image. In image classification, we assume that there is only one (and not multiple) objects in the image [**47**]see (figure 2.2).

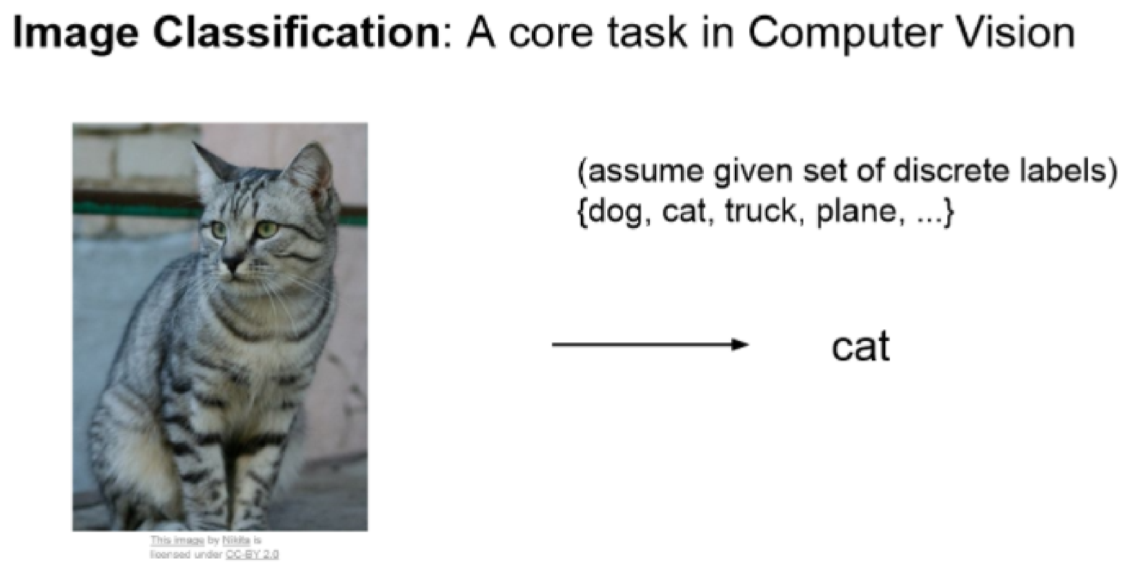


Image Classification: A core task in Computer Vision
(assume given set of discrete labels)
{dog, cat, truck, plane, ...}
cat
This image by Nikita is licensed under CC-BY 2.0

Figure 2.2: An example of Image classification

In localization along with the discrete label, we also expect the computer to localize where exactly the object is present in the image. This localization is typically implemented using a bounding box which can be identified by some numerical parameters with respect to the image boundary [**47**]. Even in this case, the assumption is to have only one object per image, see (figure 2.3).

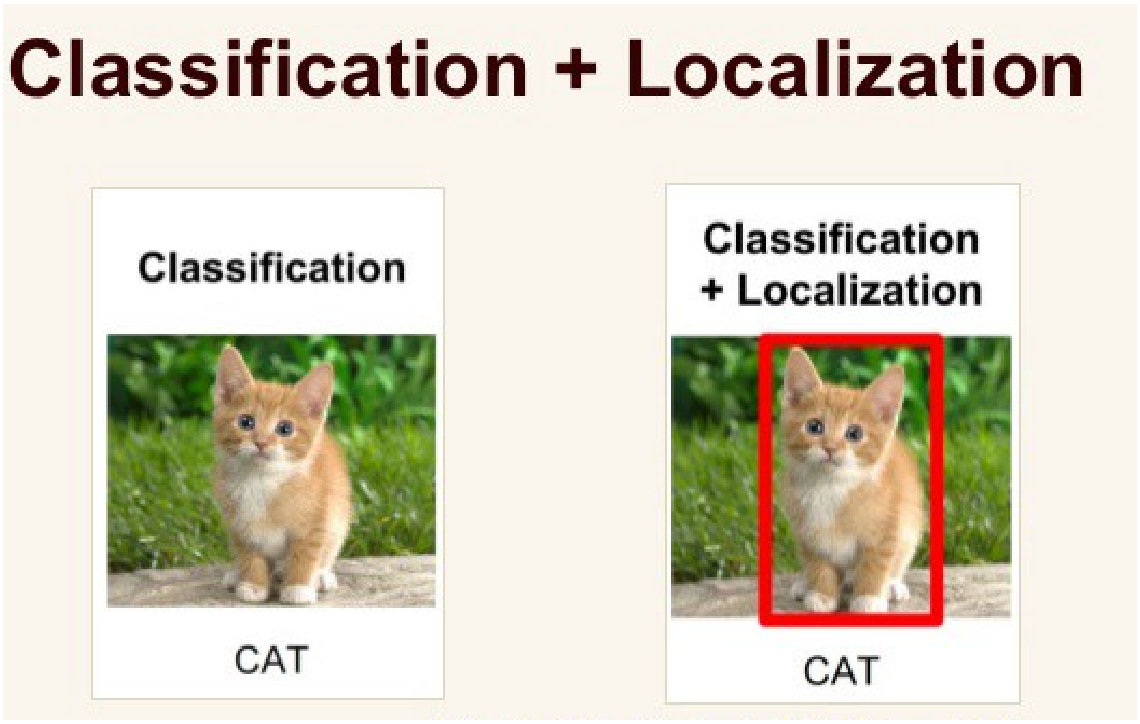


Figure 2.3: Classification with Localization

Object Detection extends localization to the next level where now the image is not constrained to have only one object but can contain multiple objects. The task is to classify and localize all the objects in the image [**47**]. Localization is done utilizing the bounding box notion once more, see (figure 2.4).

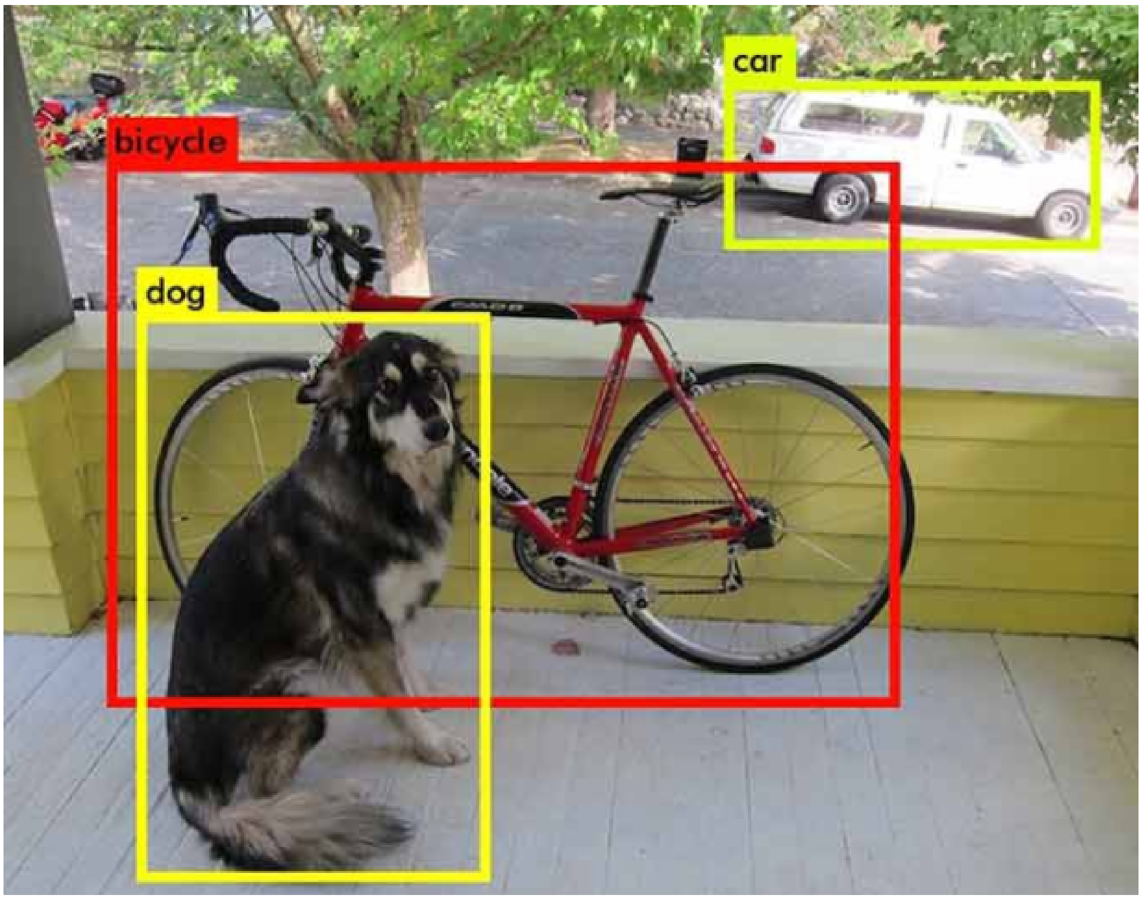


Figure 2.4: An example of Object Detection

The goal of semantic image segmentation is to label each pixel of an image with a corresponding class of what is being represented. Because we're predicting for every pixel in the image, this task is commonly referred to as dense prediction [**47**] see (figure 2.5).

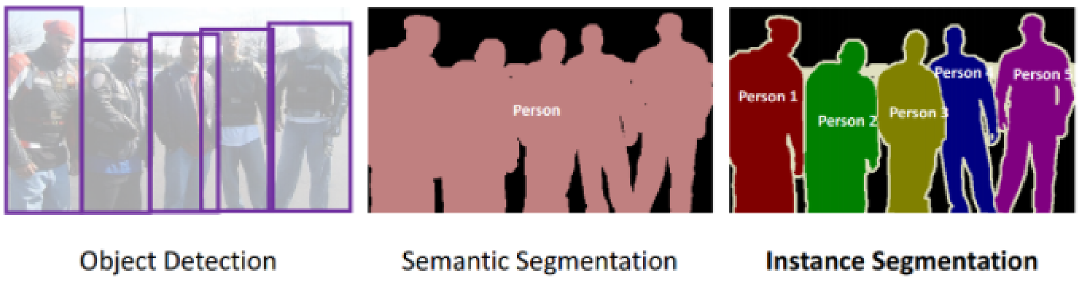
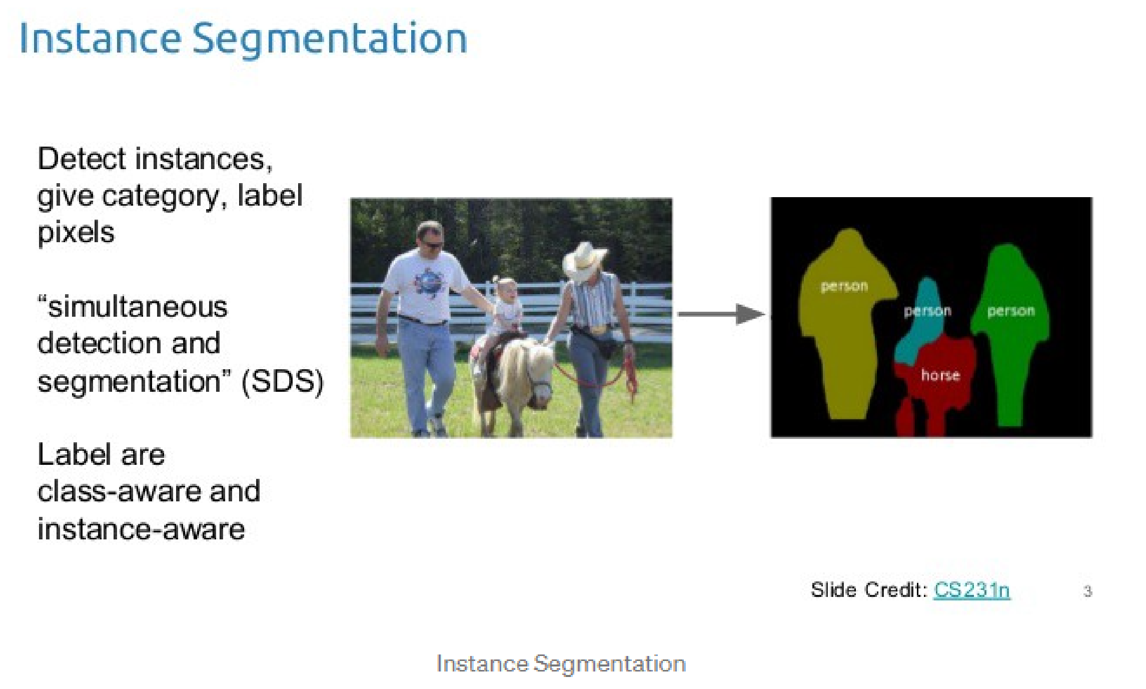


Figure 2.5: An example of semantic image segmentation

Instance segmentation is one step ahead of semantic segmentation wherein along with pixel-level classification, we expect the computer to classify each instance of a class separately. For example, in the image above there are 3 people, technically 3 instances of the class "Person". All the 3 are classified separately (in a different color). But semantic segmentation does not differentiate between the instances of a particular class[**47**] see (figure 2.6).



Instance Segmentation

Detect instances, give category, label pixels

"simultaneous detection and segmentation" (SDS)

Label are class-aware and instance-aware

person
person
person
horse

Slide Credit: CS231n 3

Instance Segmentation

Figure 2.6: An example of Instance segmentation

2.2.3 Traditional Techniques for Semantic segmentation

Before the advent of deep learning, classical machine learning techniques like SVM, Random Forest, GLS [**50**], [**51**] or K-means Clustering were used to solve the problem of image segmentation [**46**]. However, like with other image-related problem statements, deep learning has outperformed earlier solutions and has now become the standard for dealing with Semantic Segmentation.

2.2.4 Deep learning Methods

The significance of scene understanding as a core computer vision problem is illustrated by the fact that a growing number of applications benefit from inferring knowledge from visual images. Various conventional computer vision and machine learning techniques have been used in the past to tackle this problem.

Despite their prominence, the DL revolution has reversed conditions so that many computer vision problems, including semantic segmentation, are now being solved using deep architectures, typically CNNs, which outperform other approaches in terms of accuracy and efficiency.

2.3 Deep learning-based semantic segmentation methods

The constant superiority of deep learning methods in a variety of high-level computer vision tasks – mostly supervised approaches like CNNs for image classification or object recognition – motivated researchers to investigate their possibilities for challenges like semantic segmentation. We will highlight only a few of the over a hundred DL-based segmentation approaches proposed through 2021 that have had a substantial influence on the domain.

2.3.1 Fully convolutional networks

Fully Convolutional Networks (FCN) was one of the earliest DL methods for semantic picture segmentation 61. Equivalently, an FCN is a CNN without fully connected layers allowing it to take an image of any size and generate a segmentation map of the same size. The model converts all FC layers to convolutional layers and outputs spatial maps rather than classification scores.

2.3.2 SegNet

SegNet is another promising work proposed by V. Badrinarayanan et al. [**62**], SegNet is a semantic segmentation model. This core trainable segmentation architecture consists of an encoder network, and a corresponding decoder network followed by a pixel-wise classification layer. The architecture of the encoder network is topologically identical to the 13 convolutional layers in the VGG16 network.

Its encoder part is topologically identical to the VGG-16[**63**]. The unique aspect of SegNet is how the decoder upsamples the lower-resolution input feature maps. To conduct non-linear upsampling, the decoder leverages pooling indices obtained in the matching encoder's max-pooling phase.

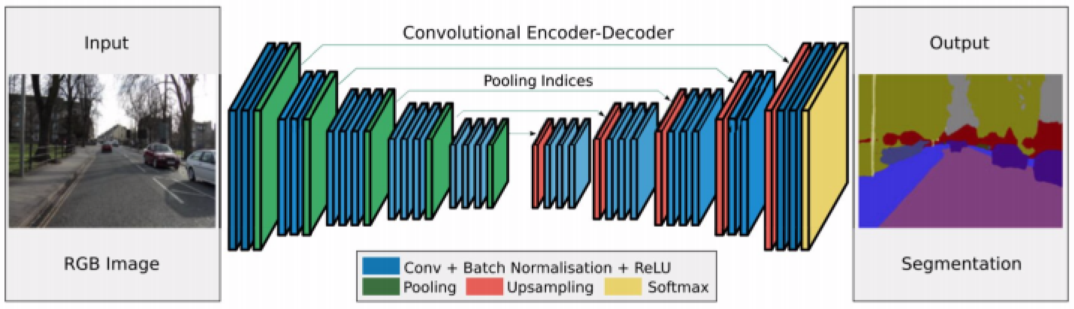


Figure 2.7: SegNet architecture from [**62**]

2.3.3 DeepLab

DeepLabv1[**64**] and DeepLabv2[**65**] were created by Chen et al., which are two of the most widely used image segmentation methods. DeepLabv1 and DeepLabv2 are reviewed together because they both use Atrous Convolution and Fully Connected Conditional Random Field (CRF) except that DeepLabv2 has one additional technology called Atous Spatial Pyramid Pooling (ASPP), which is the main difference from DeepLabv1 [**66**]. (Of course, there are other differences as well, e.g.: DeepLabv2 uses ResNet and VGGNet for experiment but DeepLabv1 only uses VGGNet.)

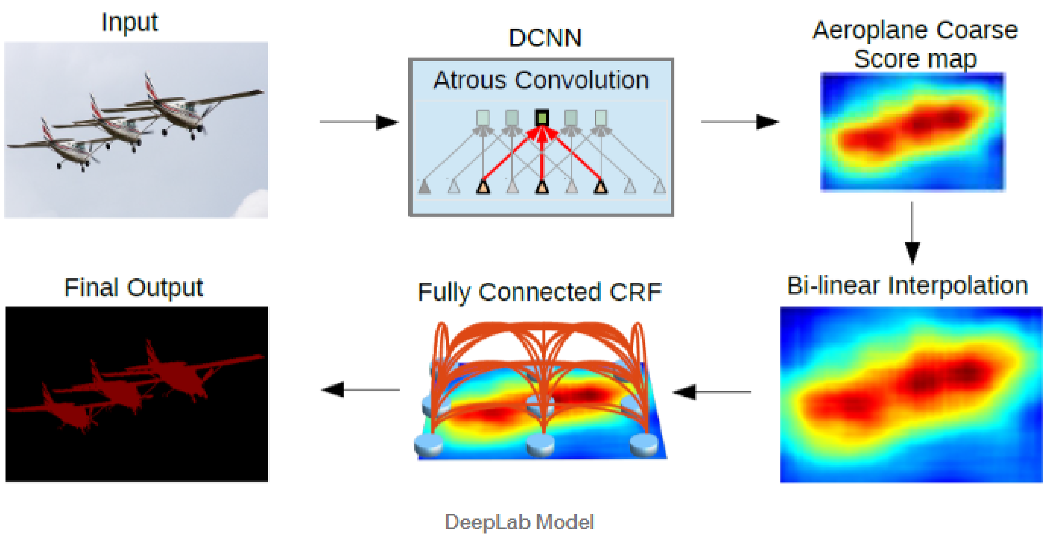


Figure 2.8: The DeepLab model[**66**]

The above figure is the DeepLab model architecture. First, the input image goes through the network with the use of atrous convolution and ASPP. Then the output from the network is bilinearly interpolated and goes through the fully connected CRF to fine-tune the result and get the final output[**66**].

DeepLabv3, a semantic segmentation architecture, makes a number of improvements over DeepLabv2. Modules that use atrous convolution in cascade or parallel to capture multi-scale context by adopting numerous atrous rates are aimed to address the issue of segmenting objects at different scales.

2.3.4 U-net

U-net, which was initially developed for medical/biomedical image segmentation, is also based on FCN and is inspired by ED architecture [**68**]. Data augmentation is a key component of the network and training process for getting the most out of the labeled data that is already available. A contracting path for context capture and a symmetric extending path for precise localization make up the U-Net design. Fully Convolutional Networks (FCN) was one of the earliest DL methods for semantic picture segmentation.

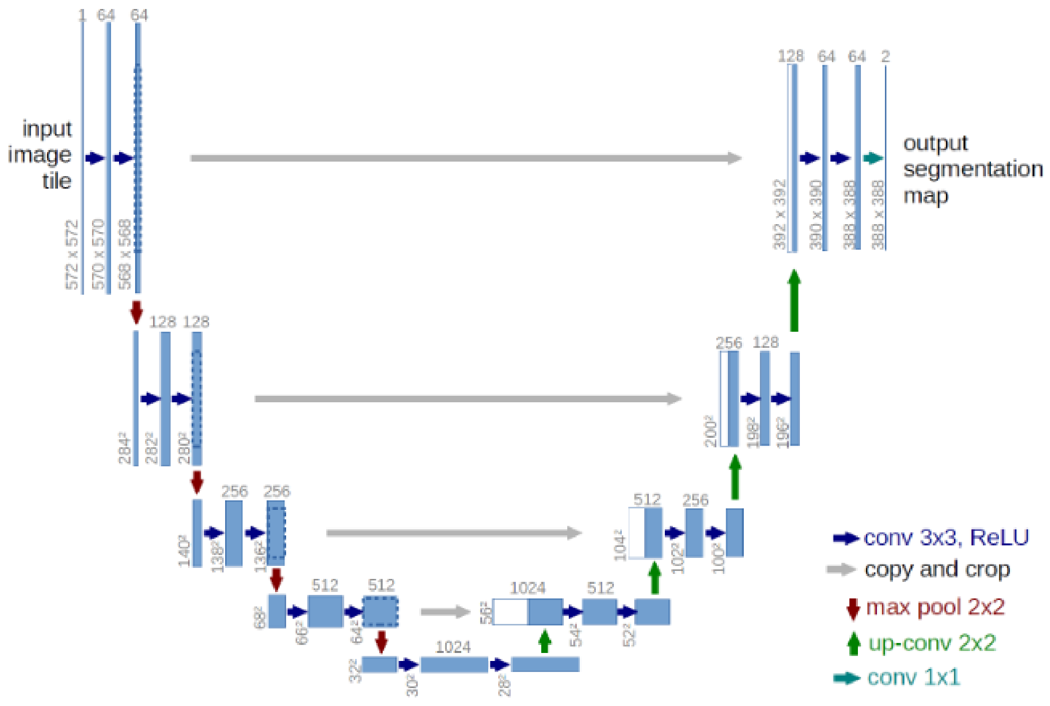


Figure 2.9: U-Net architecture

An FCN-like architecture is used during the downsampling stage to extract features using 3x3 convolutions. During the up-sampling process, deconvolution is used to increase the size of the feature maps while reducing the number of feature maps. To avoid losing pattern information, feature maps from the network's down-sampling portion are replicated to the network's up-sampling portion. The feature maps are then processed using a 1x1 convolution to create a segmentation map that categorizes each pixel in the input picture.

2.4 Conclusion

In this chapter, we discussed a set of principles related to semantic segmentation. We defined and connected it to DL and also compared it to other tasks. After that, we went over various prominent CNN architectures before introducing a set of DL-based image segmentation methods.

Chapter 3

Semantic segmentation of remote sensing images

3.1 Introduction

In this chapter, we start by defining the problem of remote sensing image segmentation. Then we give more details about the dataset used to test and evaluate the proposed model. After that, we explain how we use data augmentation to have more data available for the training and test phases. Subsequently, we present the model building blocks, where we explain the construction of the network architecture of the proposed model and how to train it. Next, we discuss the experiments, tests, and results, starting by explaining the method of conducting the experiments and the environment used, then we show the test results, and then compare those results. Finally, we evaluate our results using the standard IoU and accuracy metrics, suitable for semantic segmentation problems.

3.2 Problem definition

Aerial and satellite imagery allows monitoring of different types of terrain using images and videos taken by satellites or aerial vehicles. It can be used to measure forest monitoring, and deforestation, to map damaged areas after natural disasters, in addition to other not exploited use cases. On the other hand, the enormous growing amount of images presents a challenge: how to extract relevant knowledge from all this big data? Human beings are unable to look at all the pictures all the time. This is why we are looking for tools and techniques to improve computer vision capability and to see beyond human beings.

Our project aims to develop techniques based on deep learning approaches for the semantic segmentation of remote-sensing images. This feature is intended to aid users to gain insight into the geospatial data quickly and automatically. The semantic segmentation methods developed within this project are based on several deep convolutional neural network architectures such as the U-Net architecture. To implement the proposed approaches, the use of open-source libraries such as Tensorflow and Keras and scientific datasets offered by the scientific community will be privileged.

3.3 Dataset

The MBRSC dataset is freely available for download under the CC0 license. It's made up of aerial footage of Dubai captured by MBRSC satellites and tagged with pixel-by-pixel semantic segmentation into six groups. With the dataset, there are three major challenges:

1- The mask pictures are in RGB, but the class colors are in hex.

2- The dataset contains 72 photos, which are divided into six bigger tiles. A neural network may be trained using 72 photos, which is a pretty little dataset.

3-Images of various heights and widths are found on each tile, and some images within the same tile vary in size. Inputs of identical spatial dimensions are expected by the neural network model. [70].

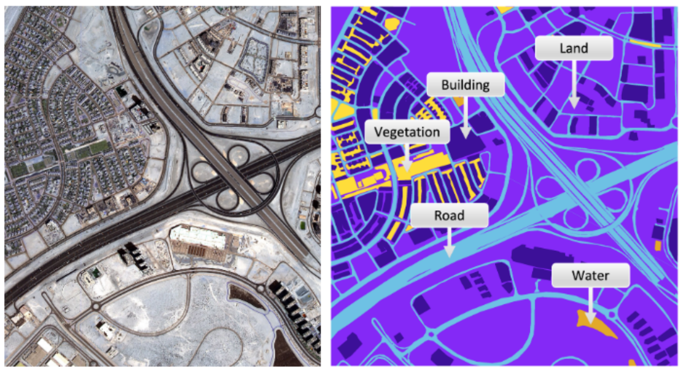


Figure 3.1: Example of Satellite image and its corresponding segmentation

3.4 Data preparation and Data augmentation

The dataset contains just 72 photos (of various resolutions), of which 65 images (90 percent) were utilized for training and 7 images (10 percent) were used for validation. Because there is such a little quantity of data, we prefer to use data augmentation to artificially enhance the amount of data and minimize overfitting [**70**]. Data augmentation is done by the following techniques.

- vertical/horizontal flipping.
- Rotation.

Here are some sample augmented images and masks from the dataset:

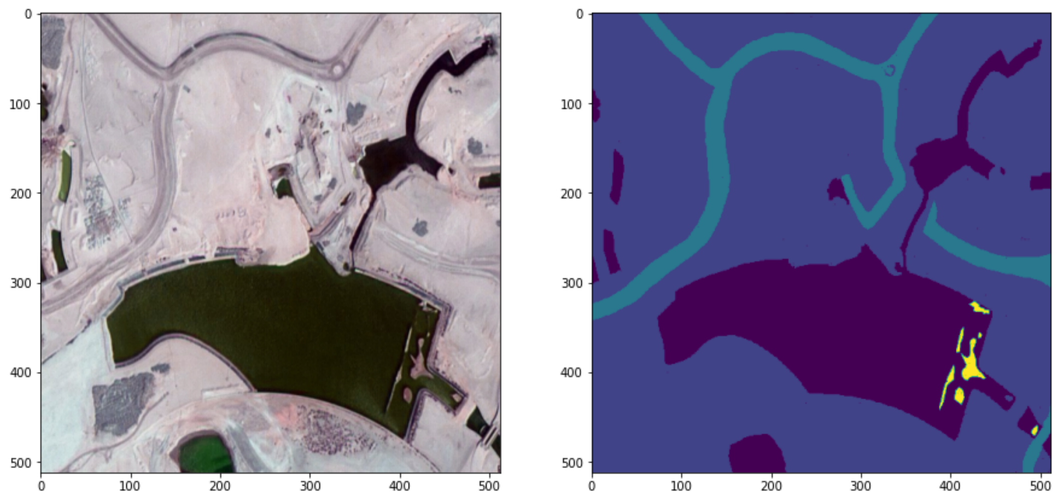


Figure 3.2: samples example of augmented images and masks from the dataset .

3.5 Model building blocks

3.5.1 Unet

U-Net, derived from the traditional CNN neural network, was designed and first applied in 2015 to process biomedical images. It is able to locate and distinguish the borders of the regions composing a certain image by doing the classification on each pixel.

Visually, it has a "U" shape. The architecture is symmetrical and consists of two main parts The left part is called Encoder, which is the general convolution process. The right part is a Decoder, which consists of transposed 2d convolutional layers.

The first block consists of an assembly of convolution layers and max pooling layers to capture the characteristics of an image and reduce its size. It consists of the repeated application of two 3x3 convolution layers. Then a 2x2 max pooling operation is applied to reduce the spatial dimensions.

The bridge connects the encoder and the network of decoders. It consists of two layers of 3x3 convolutions. The second decoder block begins with an upsampling of the feature map followed by a transposed 2x2 convolution layer. Next, two layers of 3x3 convolutions are used, where each convolution is followed by a ReLU activation function. The output of the last decoder goes through a 1x1 convolution layer.

	1 epoch	100 epochs
U-Net	83 S	2h 18min

Table 3.1: Example of execution time results for the form Unet

3.5.2 Backbone

The backbone is the architectural element that defines how the layers are arranged in the encoder network and determines how the decoder network is constructed. Often, the backbone is made up of CNNs such as VGG, ResNet, Inception, etc.

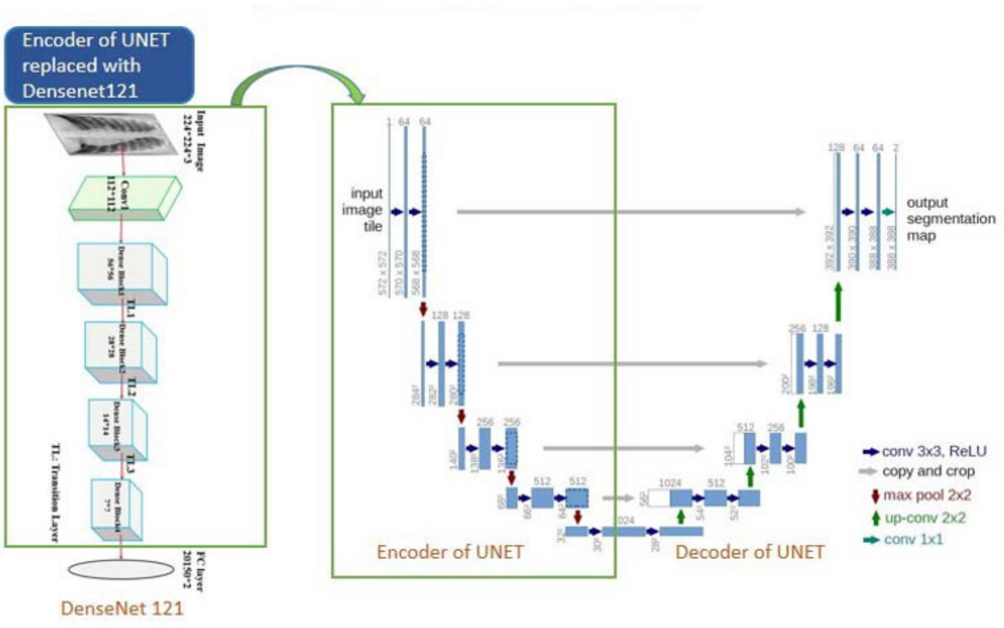


Figure 3.3: samples example of Back Bone

3.5.3 Training

To train our models, we use **90%** of all of the training stack's available slices, which is 72 images of the membrane with **256x256** resolution, and **10%** we use it for the validation process. we use the binary focal loss function since there are only two classes; white for the pixels of segmented objects and black for the rest of pixels (which correspond mostly to membranes). We use a batch size of **16**, with **200** epochs (which we can control as we like) and the default number of steps per epoch which is 22.5 calculated by dividing the number of training images by the batch size $(65/4 = 16.25\ 17)$, and it was trained using a learning rate of 10^{-2}.

We also compute semantic segmentation standard performance measures, which are the accuracy and IoU metrics. We trained the models on a Google Colab GPU (NVIDIA ® Tesla ® K80),

3.6 Experiments, tests, and results

3.6.1 Evaluation and Comparison

The obtained performance metrics such as accuracy and IoU are given in table 3.2. In the process of training our models, we have fixed the Hyperparameters of training, so we can put all the models in the same training environment and evaluate them.

Model	**IoU**	**Accuracy**
U-Net	**0.51**	**0.80**
Unet+ Resnet50	**0.53**	**0.83**
Unet+VGG19	**0.64**	**0.88**
Unet+InceptionV3	**0.65**	**0.89**

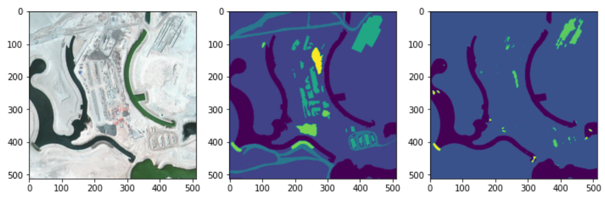
Table 3.2: Results of the performance metrics for each model

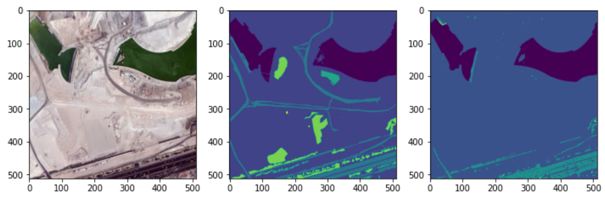
From table 3.2, we can observe that the best results are given by the approach composed of U-net architecture combined with InceptionV3 as the backbone. This approach gives an IOU metric of 0.65 which is the best among all the studied models. The same remark can be noted with the evaluation using the accuracy metric, where the best value is given by the U-net with the InceptionV3 backbone which gives the best accuracy value of 0.89. Also, we can observe that Unet+InceptionV3 is followed by Unet+VGG19, Unet+ Resnet50, and simple U-net, respectively.

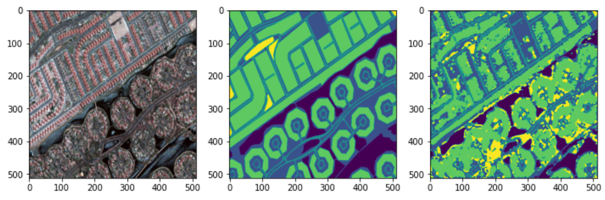
These results show the importance of combining U-net architecture with pre-trained backbones to improve the performance of semantic segmentation.

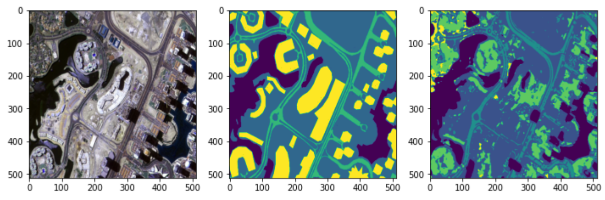
3.6.2 Visual results

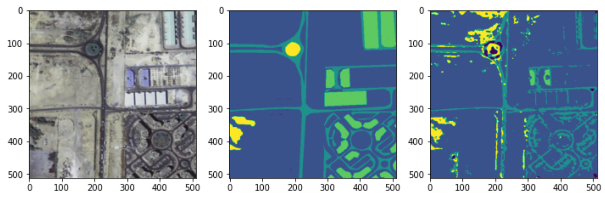
In this section, we will calculate and predict the labels of seven specific test images. The test images, the ground truth masks, and the obtained masks are given in the following figures.

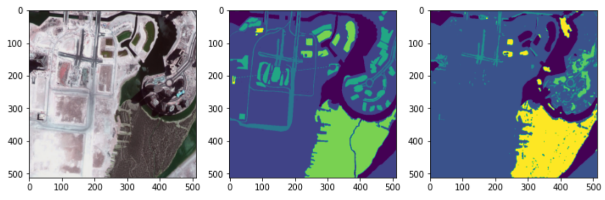

(a) U-Net prediction (Image1)


(b) U-Net prediction (Image2)


(c) U-Net prediction (Image3)


(d) U-Net prediction (Image4)


(e) U-Net prediction (Image5)


(f) U-Net prediction (Image6)

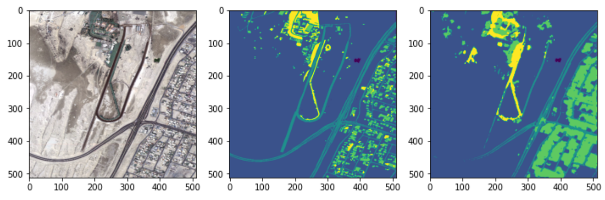
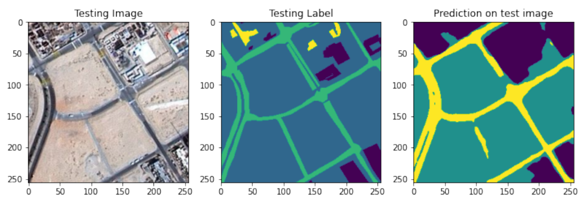
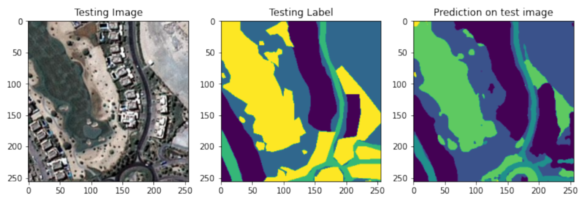

(g) U-Net prediction (Image7)

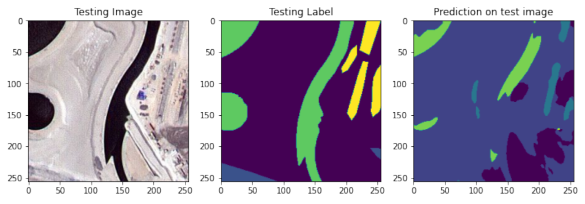
Figure 3.4: Prediction results on test images (U-net)



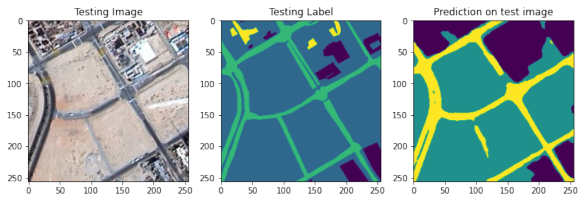
(a) Unet+Resnet50 prediction (Image1)



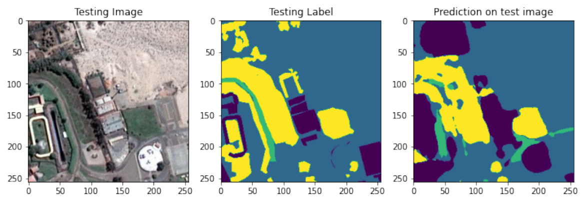
(b) Unet+Resnet50 prediction (Image2)



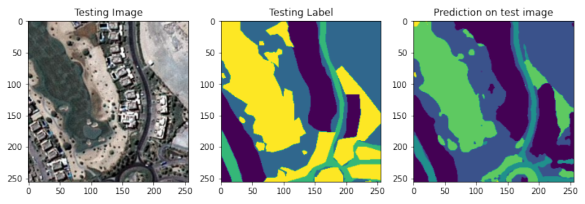
(c) Unet+resnet50 prediction (Image3)



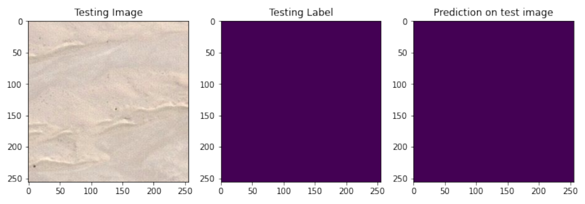
(d) Unet+Resnet50 prediction (Image4)



(e) Unet+Resnet50 prediction (Image5)

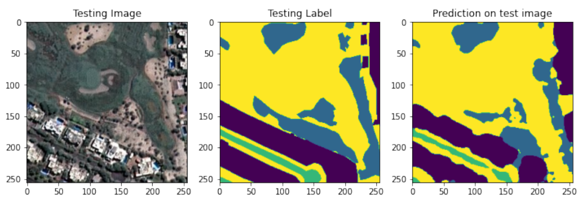


(f) Unet+Resnet50 prediction (Image6)

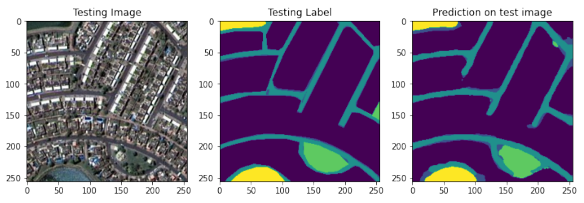


(g) Unet+Resnet50 prediction (Image7)

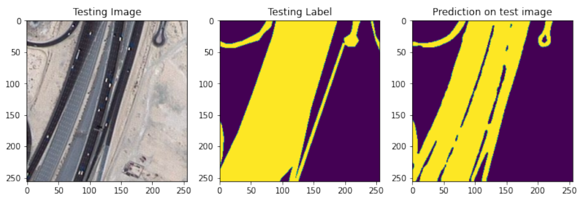
Figure 3.5: Prediction results on test images (Resnet50 backbone)



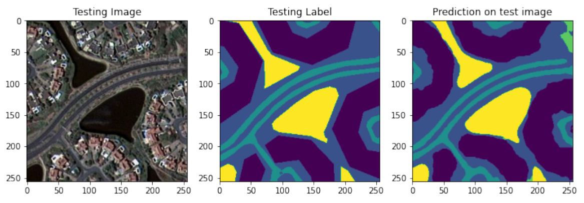
(a) Unet+Vgg 19 prediction (Image1)



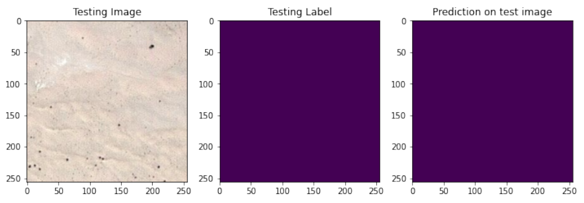
(b) Unet+Vgg 19 prediction (Image2)



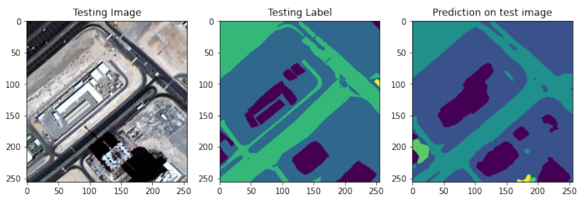
(c) Unet+Vgg 19 prediction (Image3)



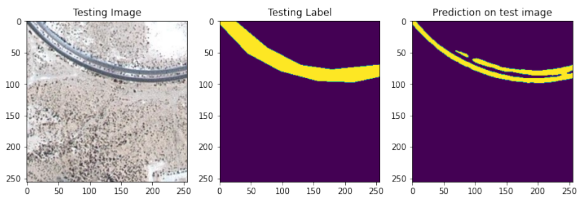
(d) Unet+Vgg 19 prediction (Image4)



(e) Unet+Vgg 19 prediction (Image5)

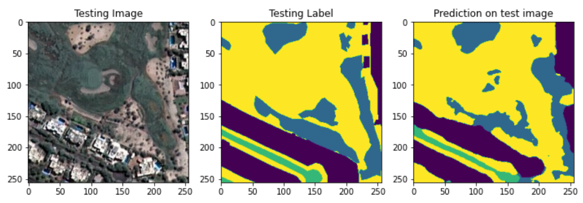


(f) Unet+Vgg 19 prediction (Image6)

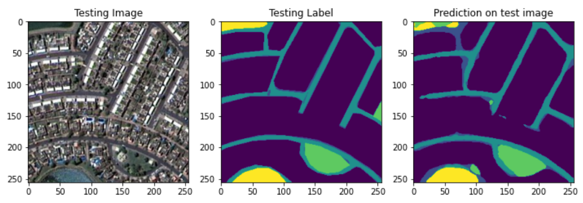


(g) Unet+ Vgg 19 prediction (Image7)

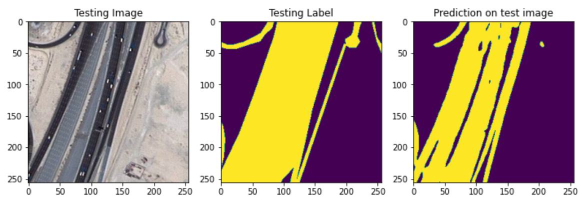
Figure 3.6: Prediction results on test images(VGG19 backbone)



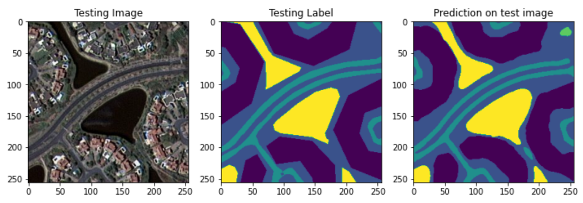
(a) Unet+inception v2 prediction (Image1)



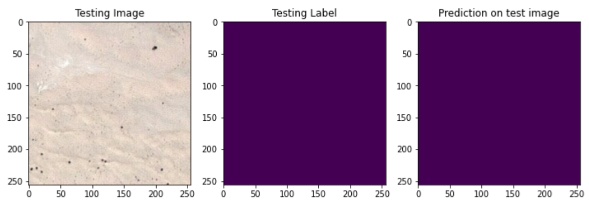
(b) Unet+inception v2 prediction (Image2)



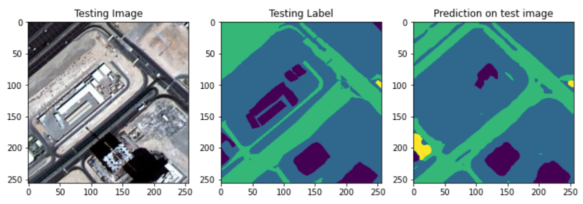
(c) Unet+inception v2 prediction (Image3)



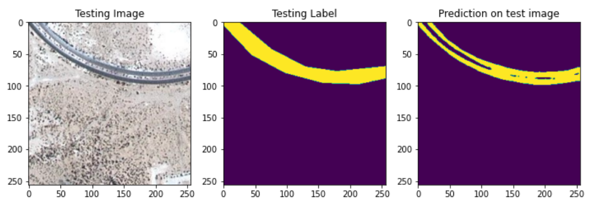
(d) Unet+inception v2 prediction (Image4)



(e) Unet+inception v2 prediction (Image5)



(f) Unet+inception v2 prediction (Image6)



(g) Unet+inception v2 prediction (Image7)

Figure 3.7: Prediction results on test images (InceptionV3 backbone)

In (figures 3.3, 3.4, 3.5 and 3.6) we show the segmentation created using the four studied networks. We notice some differences between the obtained results, especially between the predictions given by the U-net+GG model (figures 3.5) and the U-net+InceptionV3 model (figures 3.6). We can observe that the best results are given by the application of U-Net+InceptionV3 model.

3.7 Software and tools

This section will provide the definitions of the languages, software, and tools we have used to develop our application.

3.7.1 Python programming language

Python is a dynamically semantic high-level programming language that is interpreted and object-oriented. Combined with dynamic type and dynamic binding, its high-level built-in data structures make it ideal for Faster Development and for usage as a scripting or glue language to bring existing components together. Python's straightforward, easy-to-learn syntax prioritizes readability, lowering the cost of program maintenance. Python has support for modules and packages, which promotes program modularity and code reuse. Python's interpreter and substantial standard library are freely accessible in source or binary form for all major platforms. It can be open to public distribution [**72**].

Figure 3.8: Python logo

3.7.2 PyCharm IDE

PyCharm is a specialized python IDE that offers a wide range of necessary python developer tools that are deeply intertwined to offer a pleasant environment for productive python, data science development, and the web [**73**].

Figure 3.9: PyCharm logo

3.7.3 Google Colaboratory

Google research's Collaboratory, or **Colab** for short, is a product. Colab is a web-based Python editor that allows anybody to create and run arbitrary Python code. It is notably helpful for machine learning, data analysis, and teaching. Colab is a hosted Jupyter notebook service that does not require any setup and offers free access to computational resources, including GPUs, for free [**74**].

Figure 3.10: Google Colaboratory logo

3.7.4 PySimpleGUI

PySimpleGUI is a python library that allows Python programmers of all abilities to construct GUIs. PySimpleGUI defines the GUI window using a **layout** that consists of widgets (Elements). Using the layout, one of the four supporting frameworks creates a window that may be displayed and interacted with. Frameworks that are supported include Tkinter, Qt, WxPython, and Remi. These packages are commonly referred to as "wrappers" [75].

Figure 3.11: PysimpleGUI logo

3.7.5 NumPy

NumPy is a python package that is essential for numerical computation. It offers a multidimensional array object, derivative objects (including masked arrays and matrices), and a variety of routines for quick array operations such as mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation, and more [76].

Figure 3.12: NumPy logo

3.7.6 Matplotlib

Matplotlib is a python 2D plotting framework that generates publication-quality figures in a range of hard copy and interactive formats across several platforms. It is compatible with python scripts, the python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits. Among the visualizations that can be created using matplotlib are bar graph and pie chart, box plot and histogram plots, scatter plot, as well as figures [77].

Figure 3.13: Matplotlib logo

3.7.7 Pandas

Pandas is a popular open-source python library for data science, data analysis, and machine learning activities. It is based on the NumPy library, which supports multidimensional arrays. Pandas, as one of the most popular data-wrangling packages, integrates well with many other data science modules within the python ecosystem and is typically included in every python distribution [**67**].

Figure 3.14: Pandas logo

3.7.8 Scikit-learn

Scikit-learn is an open-source python machine learning library. It is regarded as a straightforward and effective technique for analyzing predictive data. It is based on the NumPy, SciPy, and matplotlib libraries. This library can be used in a variety of uses, such as classification, regression, clustering, dimensionality reduction, model selection, and preprocessing [**78**].

Figure 3.15: Scikit-learn logo

General conclusion

The main goal was to develop a number of networks that would help with the image segmentation problem in remote sensing. Deep learning and neural networks, in particular, were the focus of this dissertation.

The dissertation was divided into three different chapters for structure. The first chapter dealt with definitions and general concepts about deep learning, the second was a deep dive into the theoretical notions for semantic segmentation and deep learning approaches, and the third was the experimental part where we had our tests and results.

The first chapter presents the basic principles of deep learning and the fundamental deep learning architectures, along with the most popular deep learning frameworks, as well as the benefits and drawbacks of each one.

In the second chapter, we discussed semantic segmentation, which is an important element of our dissertation. Also, we went over some of the most well-known CNN architectures. In addition, we've covered all of the deep-learning-based semantic segmentation algorithms.

In the last chapter, we present the remote sensing image segmentation problem, the dataset used for the evaluation and the obtained results given by the application of four methods, namely; simple Unet, Unet+VGG19, Unet+ResNet50 and Unet+InceptionV3. At the last of his chapter, we came up with the main result that the Unet+InceptionV3 is the best model compared to the four studied models in term of IoU and Accuracy metrics.

Finally, we think that we are far from obtaining the optimal results for semantic segmentation of remote sensing images. Future works will improve the current models and add new elements to them to make them work more efficiently.

Bibliography

[1] Dr. Iqbal H. ,Sarker. , Deep Cybersecurity: A Comprehensive Overview from Neural Network and Deep Learning Perspective ,SN Computer,vol 1 ,(2021). 1–16 .Doi : https://10.1007/s42979-021-00535-6.

[2] Karhunen J, Raiko T, Cho KH. Unsupervised deep learning: a short review. In: Advances in independent component analysisand learning machines. (2015); p. 125–42. https://doi.org/10.1016/B978-0-12-802806-3.00007.

[3] Jamil Ahmad ,Khan Muhammad,Sung Wook Baik, Data augmentation-assisted deep learning of hand-drawn partially colored sketches for visual search .12(8). (August 2017),doi=10.1371/journal.pone.0183838

[4] Crina Grosan, Ajith Abraham,.'Intelligent Systems',A Modern Approach. (2011),Doi :https://link.springer.com/book/10.1007/978-3-642 -21004-4.

[5] Sarker IH, Colman A, Han J, Khan AI, Abushark YB, Salah K. Behavdt: a behavioral decision tree learning to build usercentric context-aware predictive model. Mob Netw Appl. 2020;25(3):1151–61.

[6] Sarker IH, Colman A, Kabir MA, Han J. Individualized timeseries segmentation for mining mobile phone user behavior. Comput J. 2018;61(3):349–68.

[7] Sarker IH, Hoque MM, Uddin MK. Mobile data science and intelligent apps: concepts, ai-based modeling and research directions. Mob Netw Appl. 2021;26(1):285–303 doi : https://link.springer.com/article/10.1007/s11036-020-01650-z.

[8] Sarker IH, Kayes ASM, Badsha S, Alqahtani H, Watters P, Ng A. Cybersecurity data science: an overview from machine learning perspective. J Big data. 2020;7(1):1–29 doi : https://10.1186/s40537-020-00318-5.

[9] Sarker IH, Furhad MH, Nowrozy R. Ai-driven cybersecurity: an overview, security intelligence modeling and research directions. SN Computer. Science. 2021;2(3):1–18.}

[10] Hashim Qamar,'Perceptron Learning'. Information Security Engineer at Rewterz .

(2020). URL :h//www.linkedin.com/pulse/perceptron-learning-hashim-qamar.

[11] Facundo BreJuan M. Gimenez aVíctor, D.Fachinotti.Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks.

(Nov 2017), URL :https://www.researchgate.net/figure/Artificial-neural-network-architecture-ANN-i-h-1-h-2-h-n-o_fig1_321259051.

[12] Sebastian Ruder. "An overview of gradient descent optimization algorithms".In: CoRR abs/1609.04747 (2016). arXiv: 1609.04747. url: http://arxiv.org/abs/1609.04747 .

[13] Scikit-learn. Scikit-learn - Machine Learning in Python. url: https://scikit-

learn.org/stable/. url :https://scikit-learn.org/stable/.

[14] Jieneng Chen et al. TransUNet: Transformers Make Strong Encoders for Med-

ical Image Segmentation. 2021. arXiv: 2102.04306 [cs.CV].}

[15] NumPy. What is NumPy? url: https://numpy.org/doc/stable/user/ whatisnumpy.html whatisnumpy.html.

[16] C.M. Bishop. Pattern Recognition and Machine Learning. Springer, Berlin: Information Science and Statistics, 2006. isbn: 0387310738.

[17] PysimpleGUI. Python GUIs for Humans. url: https://pypi.org/project/ PySimpleGUI/ . url: https://pypi.org/project/PySimpleGUI/.

[18] Ravindra Parmar. Common Loss functions in machine learning. Ed. by Towards Data Science. Sept. 2018. url: https://towardsdatascience.com/ common-loss-functions-in-machine-learning-46af0ffc4d23.

[19] B. D. Hammel. Common Loss functions in machine learning. To- wards Data Science .(2018).URL :https://towardsdatascience.com/.

[20] Yurii Nesterov. "A method for unconstrained convex minimization problem with the rate of convergence o(1/k2)". In: Doklady AN USSR 269 (1983), pp. 543–547. url: https://ci.nii.ac.jp/naid/20001173129/en/.

[21] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: Journal of Machine Learning Research 12.61 (2011), pp. 2121–2159. url: http : / / jmlr . org / papers/v12/duchi11a.html.

[22] Timothy Dozat. "Incorporating Nesterov Momentum into Adam". In: ICLR Workshop (2016).

[23] Abdul Mueed Hafiz and Ghulam Mohiuddin Bhat. "A survey on instance segmentation: state of the art". In: International Journal of Multimedia Information Retrieval 9.3 (July 2020), pp. 171–189. issn: 2192-662X. doi: 10.1007/ s13735- 020- 00195- x. url: http://dx.doi.org/10.1007/s13735- 020- 00195-x.

[24] Sevakula RK, Singh V, Verma NK, Kumar C, Cui Y. Transfer learning for molecular cancer classification using deep neural networks. IEEE/ACM Trans Comput Biol Bioinf. 2018;16(6):2089–100.

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015. arXiv: 1505.04597 [cs.CV].

[26] Andrew G. Howard et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 2017. arXiv: 1704.04861 [cs.CV].

[27] Kim K et al. “A Deep Learning-Based Automatic Mosquito Sensing and Con-trol System for Urban Mosquito Habitats”. In: Sensors 2785.12 (2019). doi: https://doi.org/10.3390/s19122785

[28] Siami-Namini S, Tavakoli N, Namin AS. The performance of

lstm and bilstm in forecasting time series. In: 2019 IEEE International Conference on Big Data (Big Data), 2019; p. 3285–292. IEEE.

[29] Tan C, Sun F, Kong T, Zhang W, Yang C, Liu C. A survey on deep transfer learning. In:

International Conference on artificial neural networks, 2018; p. 270–279. Springer.

[30] Ślusarczyk B. Industry 4.0: are we ready? Pol J Manag Stud.2018; p. 17

[31] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, ErhanD, Vanhoucke V, Rabinovich A. Going deeper with convolutions.In: Proceedings of the IEEE Conference on computer vision and pattern recognition, 2015; p. 1–9.

[32] Vesanto J, Alhoniemi E. Clustering of the self-organizing map. IEEE Trans Neural Netw. 2000;11(3):586–600.

[33] Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol P-A, Bottou L. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J Mach Learn Res. 2010;11(12).

[34] Wang W, Zhao M, Wang J. Effective android malware detection

with a hybrid model based on deep autoencoder and convolutional neural network. J Ambient Intell Humaniz Comput. 2019;10(8):3035–43.

[35] S. Hochreiter and J. Schmidhuber. “Long short-term memory”. In: Neural com-putation, vol. 9, no. 8 (1997), pp. 1735–1780

[36] Liu W, Wang Z, Liu X, Zeng N, Liu Y, Alsaadi FE. A survey of

deep neural network architectures and their applications. Neurocomputing.

[37] Laurence Moroney. AI and Machine Learning for Coders. O’Reilly Media, Inc., Oct. 2020. isbn: 9781492078197.

[38] S Tokui, K Oono, S Hido, J Clayton.

workshop on machine learning,learningsys.org .(2015),

[39] Keras. Mar. 2015. url: https://keras.io/

https://www.oreilly.com/library/view/hands-on-machine-learnig/9781492032632/.

[40] Aurélien Géron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition. O'Reilly Media, Inc., Sept. 2019. isbn: 9781492032649. https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/}.

[41] Sourav Samantaa et al. Multilevel Threshold Based Gray Scale Image Segmentation using Cuckoo Search. 2013. arXiv: 1307.0277 [cs.CV].

[42] Anurag Arnab et al. "Conditional Random Fields Meet Deep Neural Networks for Semantic Segmentation: Combining Probabilistic Graphical Models with Deep Learning for Structured Prediction". In: IEEE Signal Processing Magazine 35.1 (2018), pp. 37–52. doi: 10.1109/MSP.2017.2762355.

[43] Google Cloud. Cloud Tensor Processing Units (TPUs). May 2016. url: https://cloud.google.com/tpu/docs/tpus.

[44] jOm Prakash.Associate Data Scientist at CEIPAL.(2019).

Doi :https://www.quora.com/What-are-the-key-trade-offs-between-overfitting-and-underfitting.

[45] H. Yu, Z. Yang, L. Tan, Y. Wang, W. Sun, M. Sun and Y. Tang.Methods and datasets on semantic segmentation: A review. Methods and datasets on semantic segmentation: A review. 304}.Aug. 2018. 82–103.

[46] Anil Chandra Naidu Matcha.A 2021 guide to Semantic Segmentation.2021.

Doi :https://nanonets.com/blog/semantic-image-segmentation-2020/.

[47] Harshall Lamba.A Salt Identification Case Study.Understanding Semantic Segmentation with UNET. Feb 17, 2019. doi= :https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47.

[48] TechTarget Contributor.convolutional neural network.2021.

Doi :https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network.

[49] A Review on Deep Learning Techniques Applied to Semantic Segmentation.Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Jose Garcia-Rodriguez.22 Apr 2017.arXiv : 1704.06857 .(cs.CV). Doi : https://arxiv.org/abs/1704.06857.

[50] Multilevel Threshold Based Gray Scale Image Segmentation using Cuckoo Search.

Sourav Samantaa et al.2013.arXiv : 1307.0277 .(cs.CV).

[51] Anurag Arnab et al.IEEE Signal Processing Magazine.Conditional Random Fields Meet Deep Neural Networks .for Semantic Segmentation: Combining Probabilistic Graphica

Models withDeep Learning for Structured Prediction.35.1.2018.37–52.   Doi : 10.1109/MSP.2017.2762355.

[52] Medium –Towards.Renu Khandelwa. Anomaly Detection using Autoencoders.35.1.Jan. 2021.

Doi :https://towardsdatascience.com/

anomaly-detection-using-autoencoders-5b032178a1ea.

[53] Shipra Saxena . The Architecture of Lenet-5.March 18, 2021},

Doi : https://www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5/.

[54] P. Vincent et al. Journal ofmachine learning research. Stacked denoising autoencoders: Learning usefu representations in a deep network with a local denoising criterion.11. Dec 2010.

3371–3408.

[55] A. Krizhevsky, I. Sutskever, and G. E. Hinton.in Advances in neural information processing systems. Imagenet classification with deep convolutional neural networks(2012). 1097–1105.

[56] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Jose Garcia-Rodriguez.A Review on Deep Learning Techniques Applied to Semantic Segmentation.22 Apr 2017},

arXiv :1704.06857.(cs.CV).

[57] Avinash Thite.By Great Learning Team. Introduction to VGG16 | What is VGG16?.Oct 1, 2021.

Doi :https://www.mygreatlearning.com/blog/introduction-to-vgg16/.

[58] K. He, X. Zhang, S. Ren, and J. Sun.Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.Deep residual learning for image recognition.(2016)770–778.

[59] geeksforgeeks.Understanding GoogLeNet Model.CNN Architecture.18 Nov, 2021},

Doi :https://www.geeksforgeeks.org/understanding-googlenet-model-cnn-architectur/.

[60] Proceedings of the IEEE Conference on Computer Vision and Pattern .MobileNet V1 Architecture.Doi :https://iq.opengenus.org/mobilenet-v1-architecture/#:~:text=MobileNet%20is%20an%20efficient%20and,architectures%20to%20build%20lighter%20models.

[61] Jonathan Long, Evan Shelhamer, and Trevor Darrell.Fully Convolutional Networks for Semantic Segmentation.

2015. arXiv :1411.4038 . cs.CV.

[62] V. Badrinarayanan, A. Kendall, and R. Cipolla. IEEE Transactions on Pattern Analysis and Machine Intelligence. SegNet: A deep convolutional encoder-decoder architecture for image segmentation.(2017). 11. 2481–2495.

[63] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition.(2015).arXiv 1409.1556.(cs.CV).

[64] Keras .Mar. 2015. doi :https://keras.io/.

[65] Yasmeen Khaled and Amr Kayid.In: Artificial General Intelligence . Performance of CPUs/GPUs for Deep Learning workloads.May 2018.doi : 10.13140/RG.2.2.22603.54563.

[66,] Sik-Ho Tsang.Semantic Segmentation. Review: DeepLabv1 & DeepLabv2 — Atrous Convolution.Nov 9, 2018.

Doi :https://towardsdatascience.com/review-deeplabv1-deeplabv2-atrous-convolution-semantic-segmentation-b51c5fbde92d.

[67] Matplotlib. Pandas Documentation. url: https : / / pandas . pydata . org /

docs/.

[68] Olaf Ronneberger, Philipp Fischer, and Thomas Brox.U-Net: Convolutional Networks for Biomedical Image Segmentation.2015. arXiv : 1505.04597 .(cs.CV).

[69] Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media. Aurélien Géron.Sept. 2019. Doi :9781492032649.

[70] Andrew Joseph Davies. Semantic Segmentation of Aerial Imagery Using U-Net in Python

Semantic Segmentation of MBRSC Aerial Imagery of Dubai Using a TensorFlow U-Net Model in Python. https://towardsdatascience.com/semantic-segmentation-of-aerial-imagery-using-u-net-in-python-552705238514.

[71] Liang-Chieh Chen et al.Atrous Convolution, and Fully Connected CRFs.DeepLab: Semantic Image Segmentation with DeepConvolutional Nets.(2017).

arXiv :1606.00915 .(cs.CV).

[72] Python. What is Python? Executive Summary. url: https://www.python.

org/doc/essays/blurb/.

[73] Jetbrains. PyCharm documentation (Quick start guide). url: https://www.

jetbrains.com/help/pycharm/quick-start-guide.html.

[74] Google Research. Colaboratory - Frequently Asked Questions. url: https :

//research.google.com/colaboratory/faq.html.

[75] PysimpleGUI. Python GUIs for Humans. url: https://pypi.org/project/

PySimpleGUI/.

[76] NumPy. What is NumPy? url: https://numpy.org/doc/stable/user/

whatisnumpy.html.

[77] Hu Cao et al. Swin-Unet: Unet-like Pure Transformer for Medical Image Segmentation. 2021. arXiv: 2105.05537 [eess.IV].

[78] Scikit-learn. Scikit-learn - Machine Learning in Python. url: https://scikitlearn.org/stable/.

[79] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks

for Large-Scale Image Recognition. 2015. arXiv: 1409.1556 [cs.CV].