SAPIENZA
UNIVERSITÀ DI ROMA

Camouflaged People Detection from UAVs

Faculty of Ingegneria dell'informazione, informatica e statistica
Master's Degree In Computer Science

Davide Modenese
ID number 1665812

Thesis Supervisor
Prof. Danilo Avola
Danilo Avola

Thesis Co-Supervisor
Prof. Daniele Pannone

Academic Year 2023/2024

Camouflaged People Detection from UAV
Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: modenesedavide@gmail.com

Contents

Chapter 1

Introduction

This work focuses on the development of a system for the detection of camouflaged individuals using UAV (Unmanned Aerial Vehicle) technology leveraging state-of-the-art computer vision techniques. This chapter provides an overview of the research conducted. In particular, Section 1.1 outlines the scope and purpose of this study, detailing the specific objectives and potential applications across various sectors. Section 1.2 discusses the current state of the art in camouflaged object and people detection, including a review of the available literature and methodologies. Section 1.3 presents the contribution of this research, highlighting the limitations of existing detection systems and how this study can improve upon them using innovative techniques.

1.1 Scope and Purpose

The aim of this research is to develop a system for detecting camouflaged individuals through images captured by UAVs. Camouflage presents a challenge in various fields where objects or people blend into their surroundings, rendering common detection methods ineffective. This research seeks to contribute to the field of surveillance by providing an advanced computer vision model capable of detecting camouflaged individuals in real time. The proposed solution employs deep learning models, specifically CNNs such as U-Net and MobileNet, which can ensure accurate detection while maintaining low prediction times. The practical applications of this research can span various sectors including:

- Defense and security, detection of hostile targets or intruders
- Search and rescue, locating individuals in remote or dangerous areas
- Wildlife conservation: detecting illegal hunters or tracking wildlife in complex terrains

1.2 State of the Art

The detection of objects and people is a widely studied and well-known subject. However, the situation changes drastically when discussing the detection of camouflaged people, which is a more

specific topic with limited studies. Among the most valid works is *Detection of People With Camouflage Pattern Via Dense Deconvolution Network* [29]. In this research, the authors provide both a model for detecting camouflaged individuals and a new dataset (CPD1K) of camouflaged people to facilitate future research and test the proposed model. The detection method proposed by the authors uses a CNN for semantic feature extraction and a dense deconvolution network (DDCN) to more effectively fuse multi-scale semantic features. The results are excellent, with an F-measure score of 0.760. Another study that uses the CPD1K dataset is *Integrating Part-Object Relationship and Contrast for Camouflaged Object Detectio*n [14]. The authors propose a CNN (POCINet) dividing the task into two parts: the search phase and the identification phase. Each part adopts a specific scheme to incorporate contrast information and relational knowledge for decoding camouflage patterns. The two phases are connected by a Search-to-Guidance (SIG) module where the search and semantic decoding results merge to improve identification accuracy. This study is highly effective with an F-measure of 0.671 and an E-measure of 0.868. The results obtained by the POCINet model will be used as a reference in this research. The authors of *Camouflaged Object Detection* [2] propose the Search Identification Network (SINet), which includes two main modules, the search module (SM) and the identification module (IM). The first is responsible for searching for camouflaged objects, while the second attempts to detect them precisely. This model also leverages neural networks such as ResNet-50, to extract image features. This study also reports good results with an E-measure of 0.869 and an F-measure of 0.587.
From an analysis of the state of the art it is clear that neural networks are the obvious starting point but the specific solutions vary significantly.

1.3 Contribution

This study follows the state of the art by choosing Convolutional Neural Networks as the foundation but introduces new solutions for detecting camouflaged individuals while maintaining good prediction speed and without sacrificing high detection accuracy. The model will demonstrate these qualities through a series of tests and evaluations on datasets specifically designed for this purpose. Additionally, it will be tested with data outside its primary scope, showing a remarkable ability to adapt to detecting objects or people in more or less abstract environments.
The research is divided into the following chapters: Chapter 2 will provide a detailed explanation of what a Convolutional Neural Network is, its origins, its development, its critical points and its strengths, as well as the algorithms and functions used. Chapter 3 offers an explanation of the networks on which the proposed model is based and then focuses on the proposed architecture explaining its design and configuration details. Chapter 4 provides an explanation of the datasets used, their composition, and technical details, as well as the pre-processing operations performed. It also covers the details of the experiments, including optimizations and training processes, and finally, an analysis of the results and obtained metrics. Chapter 5 summarizes the research by discussing the work carried out.

Chapter 2

Machine Learning Background

2.1 Neural Network

A Neural Network is a computational model inspired by the structure and function of a biological neural network such as the human brain. It consists of layers of artificial neurons, also called nodes, connected to each other through edges, which model the synapses of a brain. Neurons receive information in the form of signals, process them and send a signal to other connected neurons. The strength of neural networks lies in their ability to learn which means adjusting the strength of each connection's signal based on the data received. Thanks to this quality, neural networks have become a powerful and often indispensable tool used in various fields and for complex tasks such as classification, prediction and pattern recognition.

2.1.1 Perceptron

The **Perceptron** is one of the simplest types of neural networks. It's a binary classifier that determines whether an input belongs to one of two classes. The perceptron can be mathematically represented as:

$$y = f(\sum_{i=1}^{n} w_i \cdot x_i + b) \tag{2.1}$$

where:

- x is the input vector
- y is the weight vector, with w_i being the weight of x_i
- $f()$ is the step function, which serves as the activation function
- b is the bias term that allows the model to shift the decision boundary
- y is the output value

In the perceptron, **weights** and **biases** are fundamental components in determining how the input data is processed to make a decision. Weights are numerical values assigned to each input and

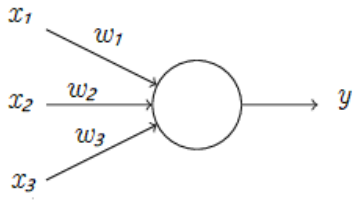


Figure 2.1. Representation of a Perceptron

determine its importance and influence on the final decision. When a perceptron receives input data, each input is multiplied by the corresponding weight. During the training process, the perceptron adjusts these weights to reduce prediction errors. The ultimate goal is to find the optimal set of weights to correctly classify the data. The bias is an additional value added to the inputs and it allows the perceptron to shift the boundaries further from the origin making the model more flexible. Like the weights, the bias is adjusted during training to improve accuracy and helps capture the determining factors for correct classification.

Activation functions are extremely important in a perceptron. They determine whether a neuron should be activated or not based on the weighted sum of the inputs. The activation function introduces non-linearity into the model allowing it to learn complex patterns and make more sophisticated decisions. Without an activation function the output would be a simple linear function of the inputs, limiting the model's power. There are several activation functions:

- **Step Function (Binary Activation)** is the simplest activation function. The output will be 1 if the input exceeds a certain threshold otherwise it will be 0.

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{2.2}$$

- **Sigmoid Function (Logistic Activation)** produces a value between 0 and 1 making it perfect for models that need to predict probabilities.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

- **TanH** is similar to the sigmoid function but the output is a value between -1 and 1.

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.4}$$

- **ReLU (Rectified Linear Unit)** is one of the most commonly used activation functions. Its output equals the input if positive otherwise it's 0.

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{2.5}$$

2.1.2 Multilayer Perceptron

A **Multilayer Perceptron (MLP)** is an extension of the basic perceptron and as the name suggests, it consists of multiple layers of neurons allowing it to model data that cannot be handled by a simple perceptron. It uses a more complex architecture with several layers where each layer is composed of multiple neurons interconnected with the neurons of the next layer. The outputs of one layer are passed as inputs to the next layer. By stacking multiple layers, the MLP gains the ability to model complex non-linear relationships in data. It's one of the foundational models of deep learning and consists of three main layers:

- **Input Layer**, the first layer of the network. Here the input data is received, processed and passed to the next layer. Each neuron represents a feature of the input data and the number of neurons depends on the dimensionality of the input data (e.g., the number of pixels in an image or values in a table).
- **Hidden Layers**, these layers are located between the input layer and the output layer. An MLP can have one or more hidden layers and it is these layers that give the network the ability to model complex non-linear relationships.
- **Output Layer**, the final layer that produces the network's output. The number of neurons in this layer corresponds to the number of classes or continuous values respectively for classification and regression.

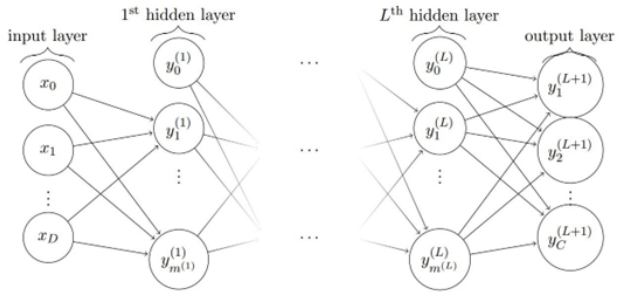


Figure 2.2. The structure of a multilayer perceptron

The multilayer perceptron is a special case of a Feedforward Neural Network. Let's look at it in detail along with Feedback Neural Networks:

- **Feedforward Neural Network**, this is the simplest neural network architecture where information moves in only one direction, from the input layer, through the hidden layers if present, to the output layer. There are no cycles in this network. It is suitable for tasks where the inputs are independent of each other such as image classification, pattern recognition, speech recognition and natural language processing. The limitation of this type of network is encountered when working with temporal patterns (e.g., video, time series) due to the lack of "memory" and feedback mechanisms.
- **Feedback Neural Network**, often known as a Recurrent Neural Network (RNN), the Feedback Neural Network allows cycles in the network meaning that information can flow both

forward and backward. These networks are designed to remember previous inputs and are particularly suitable for sequential data. Neurons are connected in such a way that the current input and the previous hidden state (memory) are used together to produce the output.

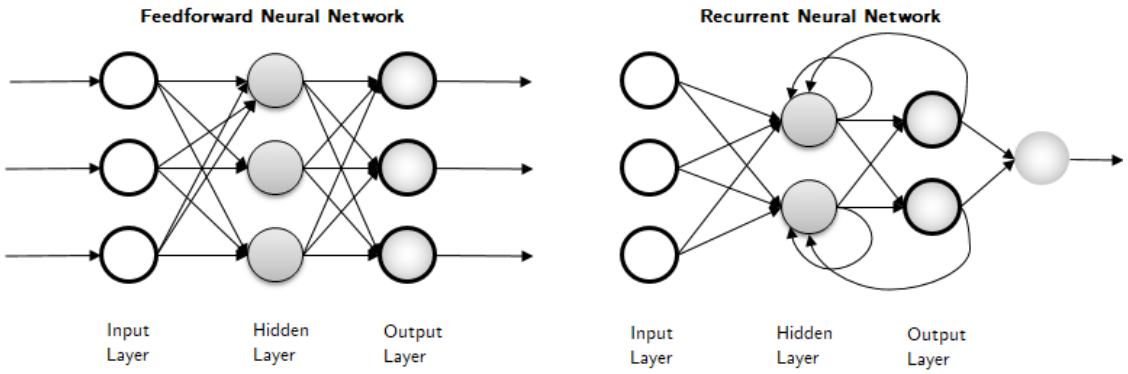


Figure 2.3. An example of Feedforward Neural Network and Feedback Neaural Network [16]

2.1.3 Learning

Learning in neural networks is the process of adjusting the network's parameters, mainly weights and biases, to minimize the difference between the predicted output and the actual target (known as the **ground truth**). This adjustment is achieved through a training algorithm.
The first step is **Forward Propagation**. The input data passes through the network, layer by layer. During this process the neurons compute the weighted sum of the inputs, add the bias and apply the activation function to produce an output. This continues until the network generates a prediction.
The second step is the **Loss Function** which calculates how far the prediction is from the actual target by providing a single value that represents the overall prediction error. Two of the most common loss functions are Mean Squared Error (MSE) and Cross-Entropy Loss.
The final step is **Backpropagation**, the main mechanism through which a neural network learns. It calculates the gradient of the loss function with respect to the network's weights and biases. Backpropagation works in reverse, moving from the output layer to the input layer, applying the chain rule of calculus to calculate how much each weight in the network contributed to the error. The result is a set of gradients for each weight and bias that tells the network how to adjust its parameters to reduce the loss.
Gradient Descent is the optimization algorithm used to update the weights and bias of the network.

$$\begin{aligned} w_{i+1} &= w_i - \eta \frac{\partial L(w,b)}{\partial w} \\ b_{i+1} &= w_i - \eta \frac{\partial L(w,b)}{\partial b} \end{aligned} \tag{2.6}$$

where:

- $\frac{\partial L(w,b)}{\partial w}$ and $\frac{\partial L(w,b)}{\partial b}$ are the gradients of the loss function with respect to the weight and bias.
- η is the learning rate, a small scalar value that controls how large the steps should be during the update of the weights.

There are different optimization algorithms, the most commonly used are:

- **Stochastic Gradient Descent (SGD)**, updates the weights after calculating the gradient for a single training sample. It is very fast but also introduces noise due to the high variance in individual samples, often leading it in the direction of the global minimum.
- **Full-Batch Gradient Descent**, the gradient is calculated over the entire training dataset. This approach ensures that each parameter is updated based on all available information but is computationally expensive.
- **Mini-Batch Gradient Descent**, a compromise between full-batch and stochastic gradient descent. Instead of calculating the gradient over the whole dataset or a single sample, it calculates it over a small batch of samples. This way it combines the efficiency of SGD with the stability of full-batch gradient descent.
- **Adam (Adaptive Moment Estimation)**, an extension of gradient descent that adapts the learning rate for each parameter based on estimates of the mean and centered variance of the gradient. It is widely used as it is very functional in practice converging faster compared to traditional gradient descent.

The final important concept in learning is the **epoch**. An epoch is a complete pass through the entire training dataset. The number of epochs, along with the learning rate, batch size, momentum and others, are **hyperparameters**, settings that are not learned from the data but are manually configured at the beginning of training.

2.2 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a specialized type of neural network designed to process grid-structured data like images but can also be used for video, audio, and text. CNNs are designed to automatically and adaptively learn spatial hierarchies of features, moving from simpler to more complex patterns. Additionally, they handle the high dimensionality of images making them particularly effective for tasks such as image classification, object detection and facial recognition.

2.2.1 CNN Layers

The typical architecture of a CNN consists of a series of convolutional layers, pooling layers and fully connected layers.
Convolutional layer is the fundamental layer of a CNN, responsible for detecting and extracting features (edges, textures, shapes in the case of images) from the input data. It applies convolution operations on the input data using filters called kernels. Let's examine the kernel and other fundamental aspects of the convolutional layer applied to an image:

- Kernel, a matrix (e.g., 3x3, 5x5) that moves across the input image or a feature map from the previous layer and performs element-wise multiplications. The result is summed to produce a single value, creating a feature map.
- Stride, the number of pixels by which the filter moves across the input. A stride of 1 means the filter moves one pixel at a time, while a stride of 2 reduces the size of the feature map more quickly.
- Padding, the process of adding extra pixels around the edges of the input.

The convolutional layer applies different filters, detecting low-level features like edges and corners in early layers or entire objects and people in deeper layers.

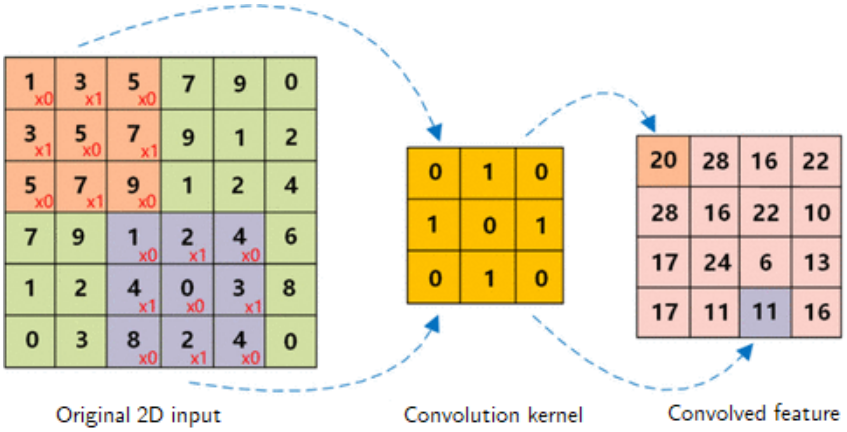


Figure 2.4. Convolution operation [28]

Pooling layer reduces the spatial dimensions (width and height) of the feature maps helping to decrease the computational cost, memory usage and overfitting. It also provides a form of spatial invariance meaning the network becomes less sensitive to small translations or distortions in the input. Pooling layers do not have parameters to learn but rely on the hyperparameters we have seen for the convolutional layer, filter size, stride, and padding. There are three main types of pooling:

- Max pooling, the most common type, takes the maximum value in the pooling region.
- Min pooling, takes the minimum value in the pooling region.
- Average pooling, calculates the average value in the pooling region.

Fully connected layer takes the output from the previous layer and connects every neuron to every neuron in the next layer. It is typically used for high-level reasoning which is why it is often found at the end of the network and is followed by the final layer responsible for generating the desired output.

2.2.2 Normalization

Normalization in a CNN is a crucial technique to stabilize and accelerate the training process, improve generalization and sometimes help combat the vanishing and exploding gradient problem.

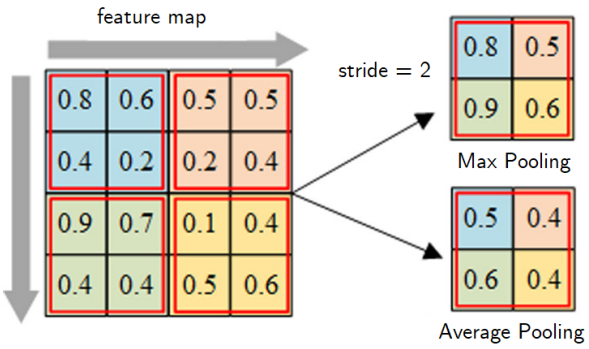


Figure 2.5. Pooling operation [10]

It helps the network converge faster by ensuring that the activations and gradients do not grow too large or too small. When this happens it becomes difficult for the model to learn efficiently and the problem becomes more severe in deeper networks, causing internal covariate shift or the aforementioned vanishing/exploding gradients.
There are various normalization techniques, the most popular are:

- **Batch Normalization (BN)** is the most commonly used technique. It normalizes the inputs for each layer by adjusting and scaling the activations during training. It ensures that the activations maintain a stable distribution with a mean of 0 and a variance of 1, helping the model train faster. For each mini-batch, the mean and variance are calculated, then the batch is normalized by subtracting the mean and dividing by the standard deviation. After normalization, the output is scaled and shifted using learnable scale and shift parameters.
- **Layer Normalization (LN)** normalizes across the features of each individual input, making it useful for recurrent neural networks and other structures where the mini-batch size is small or the input is sequential. The mean and variance are computed on the single sample.
- **Group Normalization (GN)** is an alternative to batch normalization and divides the channels of an activation into smaller groups to normalize them individually. It was designed to overcome the limitations of batch normalization on small batch sizes.
- **Instance Normalization (IN)** is similar to layer normalization, but here each feature map is normalized individually. The mean and variance are calculated over width and height for each individual channel. It is often used in style transfer applications.

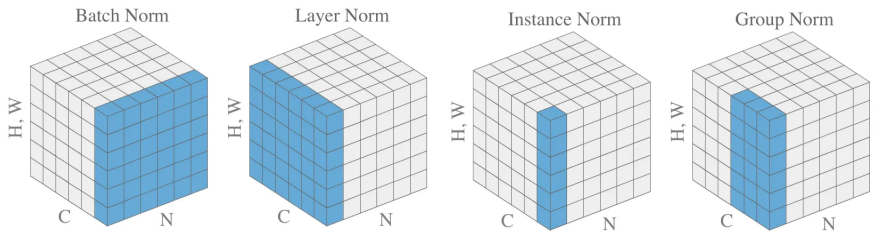


Figure 2.6. Visualization of the four types of normalization [17]

Chapter 3

Proposed Architecture

This chapter will provide an in-depth analysis of the techniques used, the algorithms and the machine learning models that form the foundation of the proposed architecture.

3.1 U-Net

The U-Net architecture [19] is a Convolutional Neural Network (CNN) commonly used for image segmentation in the medical field. One of its key advantages is the ability to work efficiently with small datasets while still producing highly accurate segmentation.

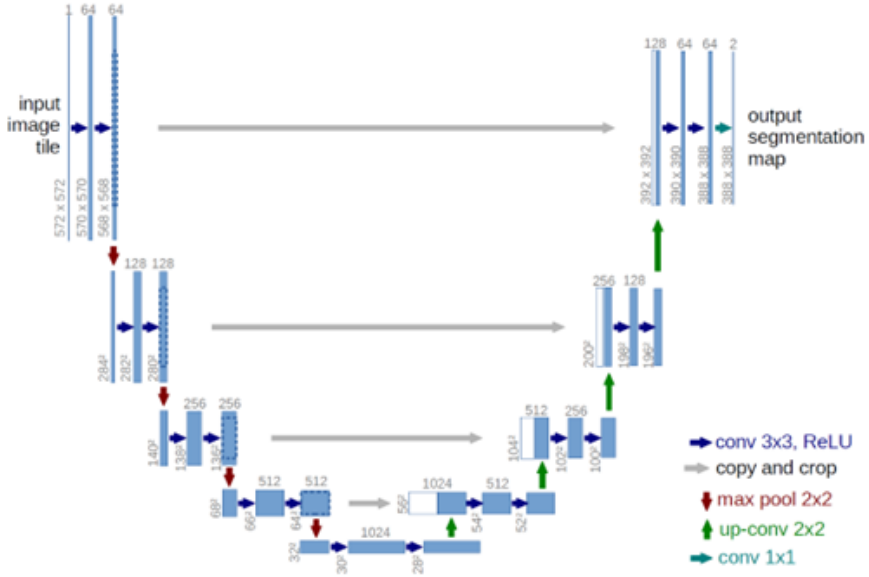


Figure 3.1. U-Net Architecture

U-Net gets the name from its U-shaped structure which is symmetric, with a "contracting path" (on the left side) and an "expansive path" (on the right side). The left side follows the typical structure of convolutional networks, performing multiple downsampling operations, each consisting of a 3x3 convolution followed by a rectifier (ReLU) and a 2x2 max pooling operation. With each downsampling step the number of feature channels is doubled. Each step in the expansive path involves upsampling the feature map using a 2x2 up-convolution, halving the number of feature

channels, followed by concatenation with the feature maps from the contracting path and two 3x3 convolutions followed by ReLU. The final step is a 1x1 convolution used to map the feature vector to the desired number of classes.
The concatenations of feature maps between the left and right sides, known as skip pathways, are extremely important because they help recover the full spatial resolution at the network's output. We will revisit them later with U-Net++ which aims to improve this aspect.

3.1.1 U-Net++

U-Net++ [30] is built on the solid foundation of U-Net and aims to reduce the semantic gap between the feature maps of the encoder and decoder before merging them. To achieve this, the skip connections between encoder and decoder have been redesigned and deep supervision has been incorporated into the architecture.

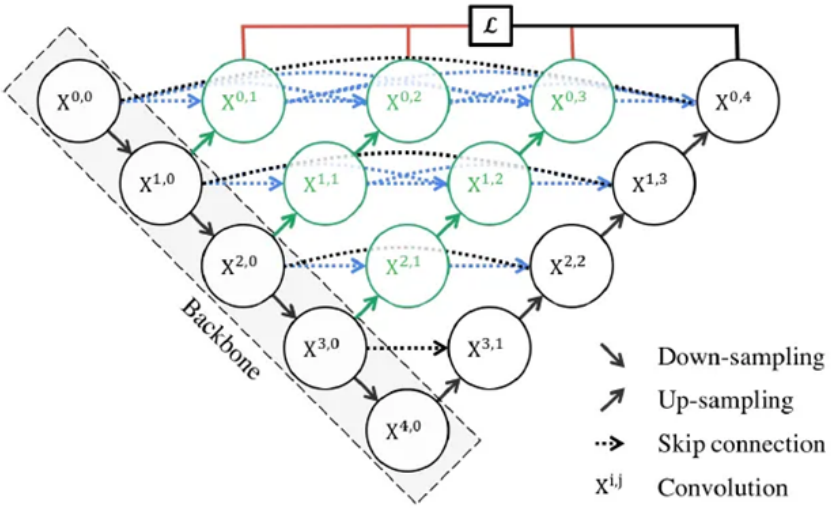


Figure 3.2. U-Net++ Architecture

In this case the skip pathways are nested pathways. Instead of directly linking the encoder to the decoder as in U-Net, intermediate convolutions are introduced, transforming the original skip connections into nested dense convolution blocks. This nesting helps to reduce the gap between the feature maps by processing and refining the information in stages before it reaches the decoder. Additionally, it smooths the sharp transitions between low- and high-level features, making the process more gradual. By reducing the semantic gap between encoder and decoder the skip pathways also help the model converge more easily.
Figure 3.3 shows an example of a skip pathway. We can formulate each convolutional layer as follows:

$$x^{i,j} = \begin{cases} \mathcal{H}(x^{i-1,j}), & \text{if } j = 0 \\ \mathcal{H}\left(\left[x^{i,k}\right]_{k=0}^{j-1}, \mathcal{U}(x^{i+1,j-1})\right), & \text{if } j > 0 \end{cases} \tag{3.1}$$

where:

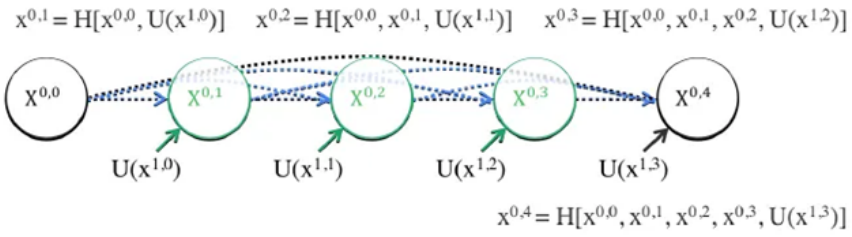


Figure 3.3. Nested Skip Pathway

- $X_{i,j}$ represents the feature map at position (i, j).
- $H()$ is a convolution operation followed by an activation function.
- $U()$ is the up-sampling layer.
- [] is the concatenation layer.

Another important concept is deep supervision at each layer of the decoder. Partial segmentation outputs are generated at different stages. These partial outputs are then combined to form the final prediction. The combination of deep supervision with the redesigned skip pathways improves the model's ability to segment objects at varying scales.
In the original work, tests were conducted using four medical datasets and the architecture consistently outperformed both U-Net and Wide U-Net.

Architecture	Params	Dataset			
		cell nuclei	colon polyp	liver	lung nodule
U-Net	7.76M	90.77	30.08	76.62	71.47
Wide U-Net	9.13M	90.92	30.14	76.58	73.38
UNet++ w/o DS	9.04M	**92.63**	**33.45**	79.70	76.44
UNet++ w/ DS	9.04M	92.52	32.12	**82.90**	**77.21**

Table 3.1. Performance comparison of different architectures on various datasets

3.2 MobileNet

MobileNetV3 [6] is a deep neural network architecture optimized for mobile devices and embedded systems. It is widely used in real-world applications including classification and object detection. MobileNetV3 builds upon previous versions (MobileNetV1 and V2), further improving model accuracy, latency and efficiency.
MobileNetV1 [7] introduced depthwise separable convolutions which separate the spatial filtering process from the feature generation mechanism. This is achieved by using two types of layers: a lightweight depthwise convolution for spatial filtering and a heavier 1x1 pointwise convolution for feature generation.
MobileNetV2 [20] builds upon V1 by introducing the linear bottleneck and inverted residual structure. The architecture consists of a 1x1 convolution expansion followed by depthwise convolutions

and ends with another 1x1 projection layer. This design allows maintaining a compact representation while enabling higher dimensional transformations improving the model's expressiveness.
MnasNet [21] is an architecture based on MobileNetV2, incorporating lightweight attention modules based on the squeeze and excitation mechanism. These modules are crucial because they are placed after the depthwise filters so that attention is applied to the larger representation.
The latest iteration MobileNetV3, combines these layers and enhances them with a modified version of the swish nonlinearity which uses hard sigmoid instead of the more common sigmoid function.

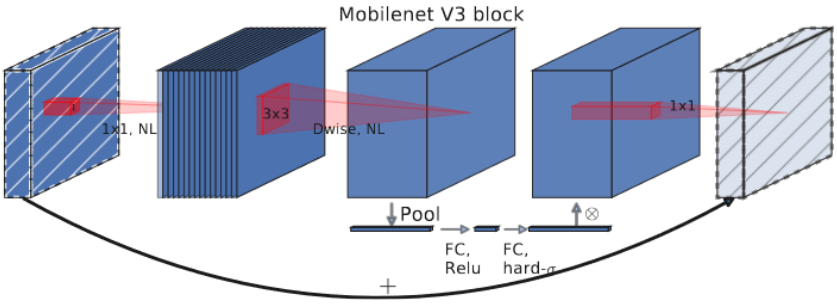


Figure 3.4. MobileNetV3 Block

To further improve the model architecture, two techniques were employed, platform-aware Neural Architecture Search (NAS) and NetAdapt.
Platform-Aware NAS was used to determine the global network structures. The process utilizes an RNN-based controller and a hierarchical search space, similar to the approach used in previous studies like MnasNet.
NetAdapt is a complementary technique used as a layer-wise optimization tool. It refines the individual sequential layers after establishing a base architecture through platform-aware NAS.
Further improvements were made by redesigning and subsequently optimizing some of the more expensive layers from MobileNetV2 as shown in the following image:

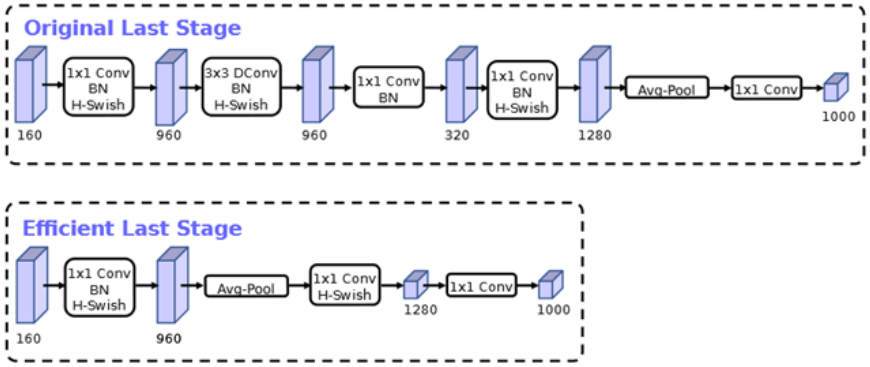


Figure 3.5. Last Stage Optimization

The final result consists of two models, MobileNetV3-Large and MobileNetV3-Small, designed for devices with high and low resource availability, respectively. The following are the specifications of the models:

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	-	RE	1
$112^2 \times 16$	bneck, 3x3	64	24	-	RE	2
$56^2 \times 24$	bneck, 3x3	72	24	-	RE	1
$56^2 \times 24$	bneck, 5x5	72	40	✓	RE	2
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 3x3	240	80	-	HS	2
$14^2 \times 80$	bneck, 3x3	200	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	480	112	✓	HS	1
$14^2 \times 112$	bneck, 3x3	672	112	✓	HS	1
$14^2 \times 112$	bneck, 5x5	672	160	✓	HS	2
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	conv2d, 1x1	-	960	-	HS	1
$7^2 \times 960$	pool, 7x7	-	-	-	-	1
$1^2 \times 960$	conv2d 1x1, NBN	-	1280	-	HS	1
$1^2 \times 1280$	conv2d 1x1, NBN	-	k	-	-	1

Table 3.2. MobileNetV3-Large

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d, 3x3	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	✓	RE	2
$56^2 \times 16$	bneck, 3x3	72	24	-	RE	2
$28^2 \times 24$	bneck, 3x3	88	24	-	RE	1
$28^2 \times 24$	bneck, 5x5	96	40	✓	HS	2
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	120	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	144	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	288	96	✓	HS	2
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	conv2d, 1x1	-	576	✓	HS	1
$7^2 \times 576$	pool, 7x7	-	-	-	-	1
$1^2 \times 576$	conv2d 1x1, NBN	-	1024	-	HS	1
$1^2 \times 1024$	conv2d 1x1, NBN	-	k	-	-	1

Table 3.3. MobileNetV3-Small

3.3 Proposed Model

As anticipated, the study quickly shifted towards a solution involving the use of Convolutional Neural Networks (CNN). The idea behind this work is to develop a model for detecting camouflaged people that is not very computationally intensive in order to be used on embedded device, such as a UAV, without excessively sacrificing detection quality. For this reason the model is based on previously seen networks. Starting with the solid and effective U-Net, the encoder is replaced with the more efficient and accurate MobileNetV3. Then the model is extended following the U-Net++ philosophy.

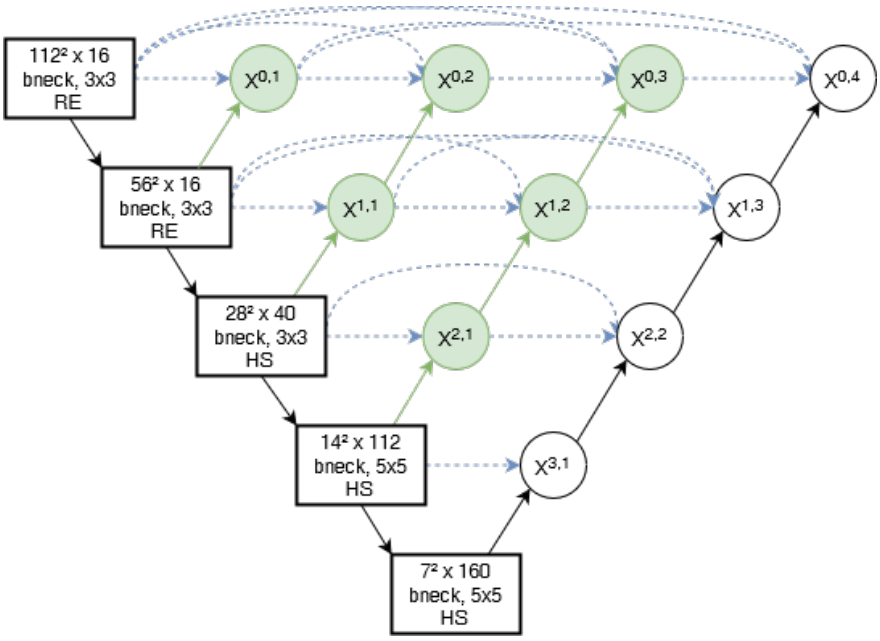


Figure 3.6. Proposed Model

The MobileNetV3 model remains identical, except for the removal of the last layers as they are

not necessary in this architecture. The decoder and nested skip pathways of U-Net++ are implemented from scratch following the original architecture. Specifically using the formula 3.1 for the convolution layers, we have:

- $H()$ and [] is a concatenation followed by a 2D convolution with a 3x3 kernel, then batch normalization and ReLU.
- $U()$ involves a transposed convolution with a 3x3 kernel and a 2x2 stride, followed by two 2D convolutions, batch normalization and finally ReLU.

An example of a nested skip pathway that uses the described layers. The number of filters for the convolution is defined by nf.

```
nf = 64
x01 = H([x00, U(x10, nf)], nf)
x02 = H([x00, x01, U(x11, nf)], nf)
x03 = H([x00, x01, x02, U(x12, nf)], nf)
x04 = H([x00, x01, x02, x03, U(x13, nf)], nf)
```

The activation function at the end of the network is a sigmoid which produces a value between 0 and 1. This way each pixel will indicate the probability of belonging to a person.

Chapter 4

Experiments

This chapter explores in detail the experiments conducted and the data obtained, starting from the chosen datasets, moving through the algorithms and metrics used and concluding with the analysis of the results produced by the model. In particular, Section 4.1 provides an overview of the datasets used, their composition and their key points. Section 4.2 describes the necessary steps for processing the datasets to ensure they are ready to be used in the experiments. In Section 4.3 the chosen algorithms, models, loss functions, metrics and finally the experiments carried out are presented. Section 4.4 provides a comprehensive analysis of the results obtained, discussing their implications and comparing them with those of other research in the same field.

4.1 Dataset

For this study, two datasets were used:

- CPD1K (`https://github.com/xfflyer/Camouflaged-Data`)
- CAMO (`https://sites.google.com/view/ltnghia/research/camo`)

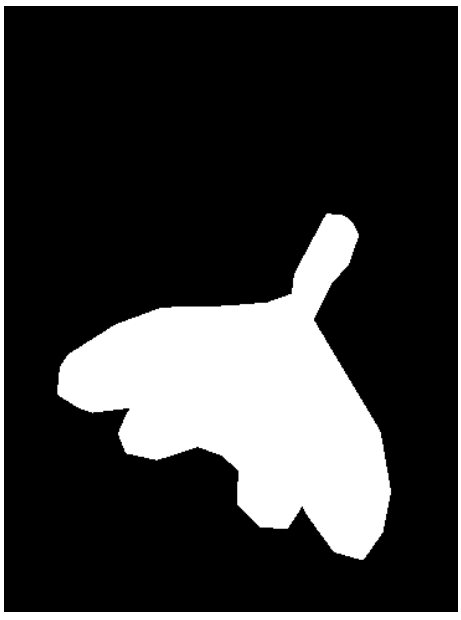
CPD1K was chosen because, after careful research, it proved to be the closest to the objectives of this study. It contains 1,000 images of camouflaged or concealed people, often in environments filled with vegetation that obstructs the view and partially covers the subjects. The dataset also includes corresponding ground truth images which are completely black with a white silhouette where a person appears in the original image. All images (4.1) are of a fixed size, 854x480. The dataset was created specifically for camouflage detection research [29].
The second dataset is CAMO, specifically designed for camouflaged object segmentation tasks. It was chosen for two reasons, first, to evaluate the performance of the proposed model on a less specific dataset and second, to compare the results with other works using a widely used dataset in this research domain. In this case, 1,250 images are provided along with their respective ground truth. The subjects vary, mostly animals and insects but also people and objects, always camouflaged with their surroundings. The images (4.2) do not have a fixed size, ranging from 340x140 to 3648x2736.

(a) Base Image

(b) Ground Truth

Figure 4.1. CPD1K example

(a) Base Image



(b) Ground Truth

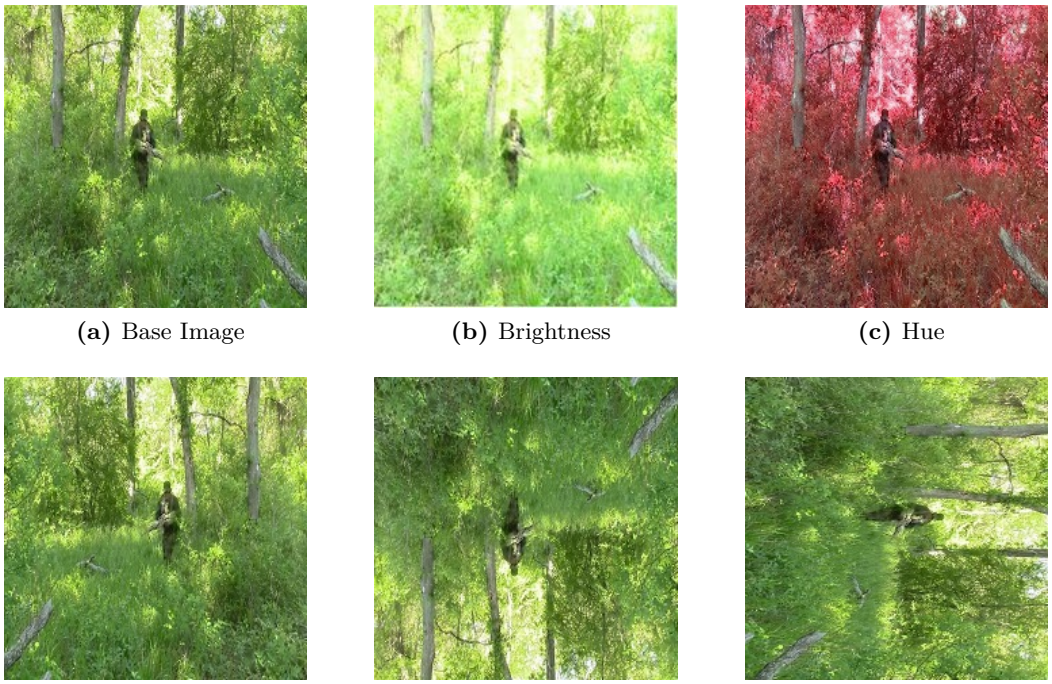
Figure 4.2. CAMO example

4.2 Data Pre-Processing

The datasets are already organized with filenames corresponding between the original images and the ground truth and there are no errors of any kind. The phase of preprocessing focused on compatibility between the input images and the chosen model. Specifically MobileNetV3 only accepts images of size 224x224 with 3 color channels. For this reason during loading it was necessary to resize all images to the correct size and remove the alpha channel where present. Another necessary step was converting the color representation scale from the more common 0-255 to the scale supported by MobileNet which ranges from -1 to 1. Both datasets have a limited number of images so to avoid easily overfitting during the training phase, a process of data augmentation was applied. Each image was copied multiple times once for each type of available modification:

- Brightness, randomly modifies brightness by increasing or decreasing its intensity.
- Hue, randomly modifies the color hue.
- Flip Hor, horizontally flips the image.
- Flip Vert, vertically flips the image.
- Rotate, randomly rotates the image by 90, 180, or 270 degrees.

With this operation the CPD1K dataset grows to 6,000 images while the CAMO dataset grows to 7,500 images.



(a) Base Image **(b)** Brightness **(c)** Hue

(d) Flip horizontally **(e)** Flip vertically **(f)** Rotate

Figure 4.3. Examples of data augmentation

4.3 Experiments and Results

4.3.1 Batch Size and Optimization

The dataset was split into 80% training set and 20% testing set, this split proved to be a good compromise, showing the best loss curve. Training was conducted on a high-performance machine, allowing for the evaluation of several parameters including batch size. A large batch size allows for faster model training through process parallelization but also requires a significant amount of memory especially when working with images. For this reason several batch sizes were tested, even going up to 64 which was immediately discarded due to excessive memory usage. The final choice fell on a batch size of 32, a compromise between memory usage and training speed.
Another important factor was finding the right algorithm to optimize the model's parameters to ensure rapid and efficient convergence. The Adam optimizer was the first to be tested and immediately proved ideal for the proposed CNN. It is an algorithm that combines the benefits of adaptive learning rates and momentum. It is based on the combination of two Gradient Descent methodologies allowing it to further optimize the descending gradient by overcoming local minimums and reaching the global minimum efficiently. Ultimately the algorithm chosen was AdamW, a slight variation of Adam.

4.3.2 Training Process

As mentioned earlier, several parameters were tested during the training phase. In addition to the already mentioned data split, batch size and optimization algorithm, experiments also focused on the number of filters which settled at 32 for the first layer of the encoder, the nested skip pathway and the last layer of the decoder, doubling at each subsequent level. The number of training epochs was also evaluated. Initially set at 20 it proved to be insufficient for complete training and was therefore increased to 30.
Another fundamental aspect on which several tests were conducted was the loss function. The first function evaluated was Binary Cross Entropy which, although producing decent results, was not sufficient for this research. Other functions were tested, including Dice Loss but the final choice fell on the IoU (Intersection over Union), also known as the Jaccard Index (4.4). As the name implies, the calculation of this value is based on the ratio between the area of intersection of the prediction and the ground truth and the union of the areas covered by the prediction and ground truth. The loss is defined as:

$$L_{IoU} = 1 - IoU \tag{4.1}$$

It is important to highlight that the experiments conducted focused on training the model excluding the MobileNetV3. In fact for this research, a pre-trained network on the "imagenet" dataset was used which significantly reduced training time. The dataset is designed for use in computer vision, particularly object recognition, and consists of numerous images (14 million) and includes more than 20,000 categories such as objects, plants and people. The network is therefore trained for a broader purpose which is why some fine-tuning tests were conducted. The U-Net part was frozen, the MobileNet part was unlocked and training was relaunched. However, the results obtained were

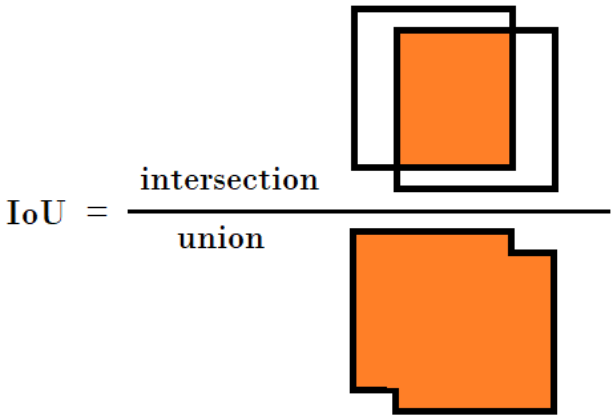


Figure 4.4. Visualization of the IoU formula

insufficient as the loss curve showed significant overfitting and this option was discarded.

4.3.3 Evaluation Metrics

The quality of the model was evaluated based on metrics commonly used in the field of image segmentation:

- **IoU** (Intersection over Union)
- **E-Measure** evaluates the alignment between a predicted saliency map and the ground truth focusing on the quality of detection across various aspects. It combines pixel-level accuracy and spatial distribution into a unified score making it more sensitive to both region-wise and overall structural errors. The E-measure is defined as:

$$E_\phi = \frac{1}{h \times w} \sum_{x=1}^{h} \sum_{y=1}^{w} \phi_{FM}(x, y) \tag{4.2}$$

  where:

  - h and w are the height and width of the saliency map
  - ϕ_{FM} is the enhanced alignment matrix reflecting the correlation between P (predicted saliency) and G (ground truth) after subtracting their global means
  - P and G are the probability maps of saliency detection and the corresponding ground truth

- **F-Measure** evaluates both precision and recall simultaneously:

$$\text{Precision}^w = \frac{TP^w}{TP^w + FP^w} \tag{4.3}$$

$$\text{Recall}^w = \frac{TP^w}{TP^w + FN^w} \tag{4.4}$$

$$F_\beta^w = (1 + \beta^2) \cdot \frac{\text{Precision}^w \cdot \text{Recall}^w}{\beta^2 \cdot \text{Precision}^w + \text{Recall}^w} \tag{4.5}$$

where:

  - β determines the weight given to precision or recall.
  - TP (True Positive) is the number of instances correctly classified as positive.
  - TN (True Negative) is the number of instances correctly classified as negative.
  - FP (False Positive) is the number of instances incorrectly classified as positive.
  - FN (False Negative) is the number of instances incorrectly classified as negative.

- **MAE (Mean Absolute Error)** represents the average pixel-wise difference between a predicted saliency map and its respective ground truth mask. The formula is:

$$MAE = \frac{1}{H \times W} \sum_{x=1}^{H} \sum_{y=1}^{W} |P(x,y) - G(x,y)| \tag{4.6}$$

where:

  - H and W are the height and width of the saliency map, respectively.
  - x and y are the pixel coordinates.
  - P and G are the probability maps of saliency detection and the corresponding ground truth.

4.4 Results and Comparison

The proposed model trained on the CPD1K dataset performed very well in detecting camouflaged or hidden people achieving excellent values for IoU, E-measure, F-measure and MAE. The model scored 0.741 for IoU, 0.839 for E-measure, 0.632 for F-measure and 0.009 for MAE. Regarding the loss value it can be said that the model converged successfully and the parameter optimization was effective. The loss value is 0.410 which is a great score considering the loss function used.
From the graphs, it can be seen that the values of the IoU, E-measure and F-measure metrics tend to increase correctly as the number of epochs rises and stabilize around the 25th epoch. It is also evident that the curves are not perfectly linear but show some fluctuations likely due to a dataset that is not sufficiently varied. Despite efforts to expand the dataset to include all possible cases, some image configurations may be underrepresented and thus harder to train. When these particular configurations are encountered in testing, a slight peak can be observed in the graphs. In the loss and MAE graphs the issue is less evident. Here the curves correctly decrease until stabilizing around the 25th epoch.
Prediction speed is also crucial. TensorFlow Lite tools were used to perform speed tests. The network was converted from Keras to TFLite without any type of extra optimization that would have affected the prediction quality. The network was tested with the TFLite benchmark software on a Xiaomi Redmi Note 8 mobile device which is equipped with a modest Mediatek Helio G85 CPU. The speed stands at 1.1s per prediction, a very good result considering the non-trivial task and the quality of the segmentation.

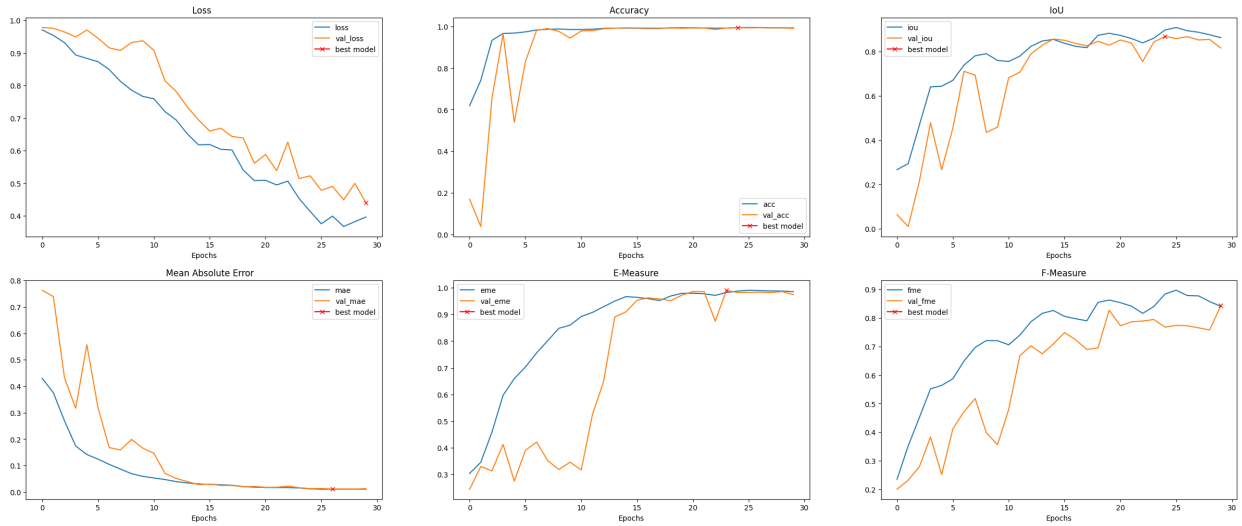


Figure 4.5. Learning Curves on CPD1K dataset

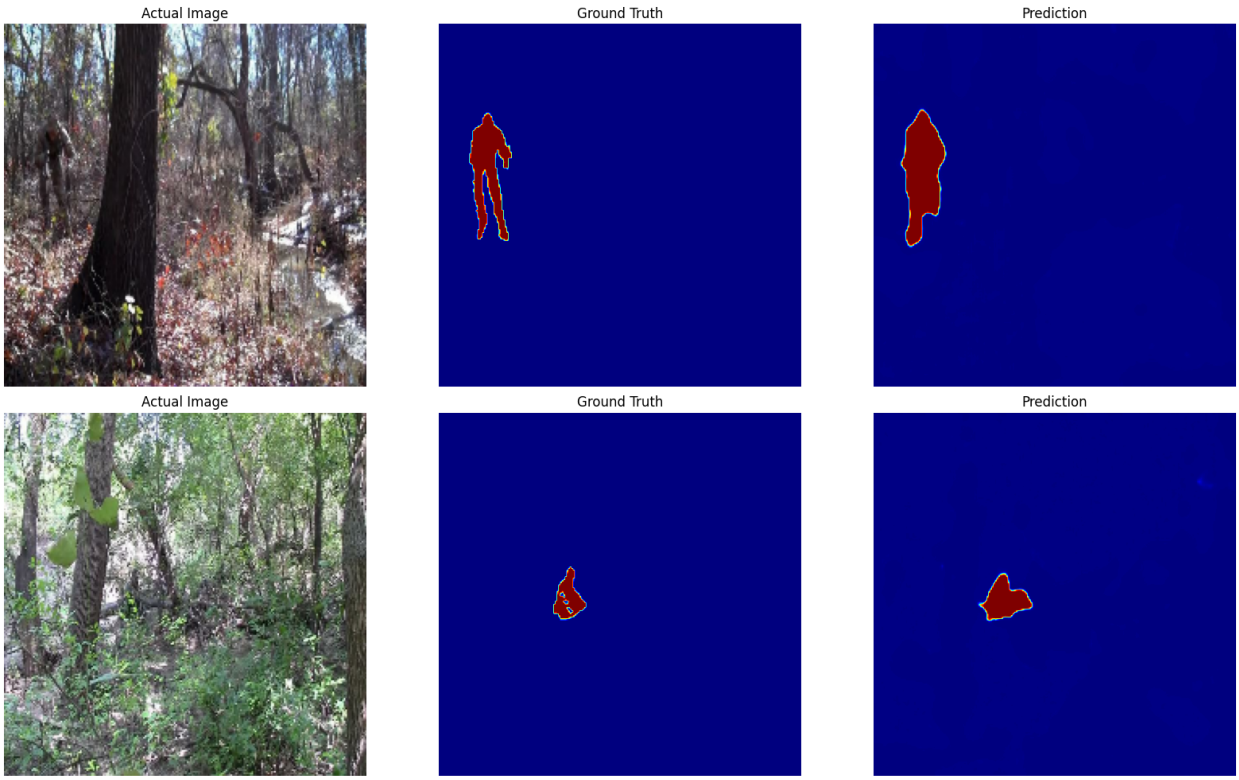


Figure 4.6. Prediction after 30 epochs on CPD1K dataset

The results were surprisingly good even when the model was trained on the CAMO dataset. The reported values are 0.509 for IoU, 0.092 for MAE, 0.840 for E-measure and 0.684 for F-measure. The graphs show similar trends to the training on CPD1K with the same issues encountered.

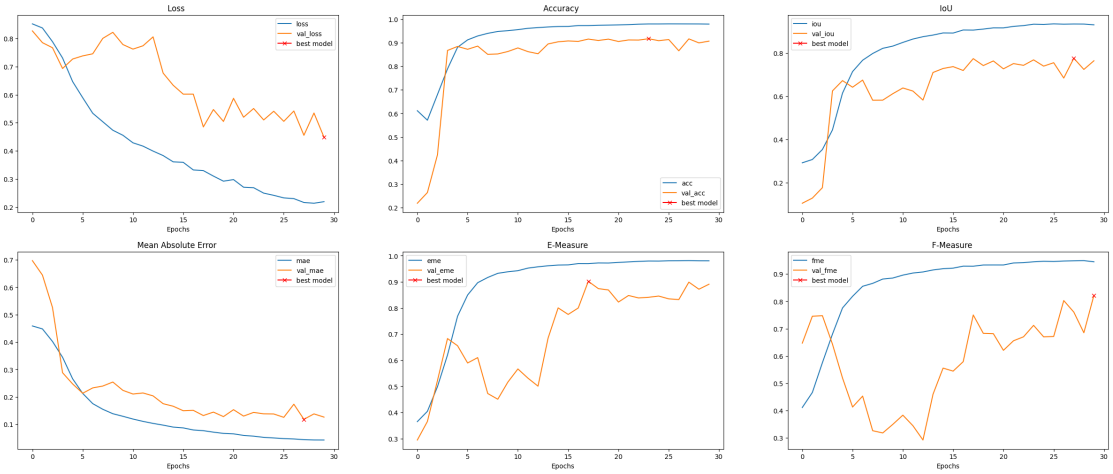


Figure 4.7. Learning Curves on CAMO dataset

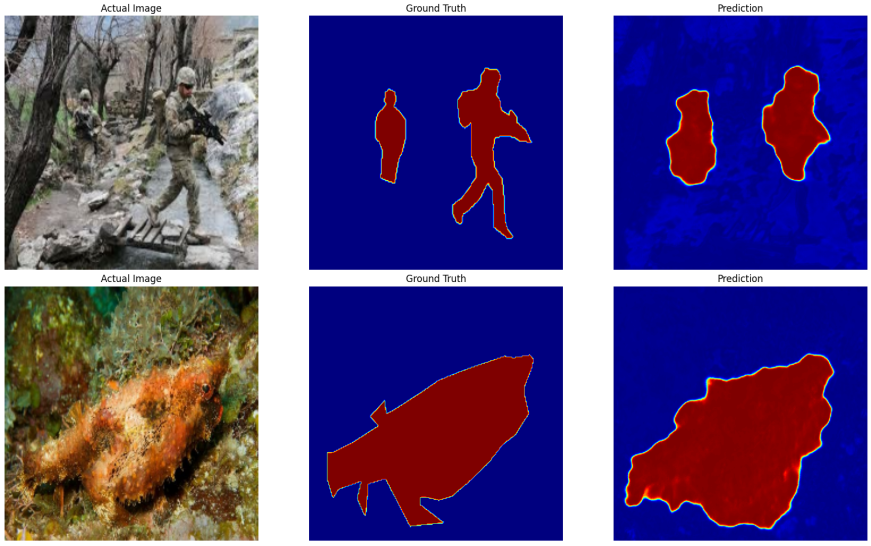


Figure 4.8. Prediction after 30 epochs on CAMO dataset

In conclusion, the proposed model proves to be effective in detecting camouflaged people and fast on low-performance devices. Equally important is the quality of the model trained on a less specific dataset which, as we will see below, achieves results comparable to the state-of-the-art. The comparison between the results of this work and the state-of-the-art highlights how the proposed model is often superior or at least comparable to those of other researchers, with a significant advantage in speed. Table 4.1 shows a comparison with state-of-the-art networks, particularly the scores obtained on the CPD1K dataset.
In Table 4.2 results are compared with other state-of-the-art models obtained on the CAMO dataset. The comparison demonstrates the validity of the proposed network, surpassing many state-of-the-art solutions and compensating with prediction speed in cases where it falls slightly behind.

	CPD1K		
	$E_\phi \uparrow$	$F_\beta^\omega \uparrow$	$M \downarrow$
Proposed Model	0.839	0.633	0.009
POCINet [14]	0.868	0.671	0.007
TSPOANet [15]	0.727	0.483	0.009
PFANet [27]	0.454	0.218	0.042
SINet [3]	0.724	0.506	0.010
PoolNet [12]	0.536	0.263	0.025
HTC [1]	0.665	0.472	0.019
EGNet [26]	0.692	0.405	0.018
BASNet [18]	0.750	0.484	0.017
PiCANet [13]	0.543	0.296	0.024
FPN [11]	0.621	0.391	0.012
MaskRCNN [5]	0.672	0.321	0.037
MSRCNN [8]	0.791	0.578	0.009
PSPNet [25]	0.552	0.317	0.016

Table 4.1. Performance comparison with state-of-the-art models that use CPD1K dataset, the best value is highlighted in blue, the second best is highlighted in red

	CAMO		
	$E_\phi \uparrow$	$F_\beta^\omega \uparrow$	$M \downarrow$
Proposed Model	0.840	0.684	0.092
UGTR [24]	0.859	0.686	0.086
SINet [3]	0.771	0.606	0.100
PraNet [4]	0.833	0.663	0.094
MirrorNet [23]	0.804	0.652	0.100
ANet-SRM [9]	0.685	0.484	0.126
EGNet [26]	0.768	0.583	0.104
HTC [1]	0.442	0.174	0.172
CPD [22]	0.729	0.550	0.115
PFANet [27]	0.622	0.391	0.172
BASNet [18]	0.661	0.413	0.159
PoolNet [12]	0.698	0.494	0.129
MSRCNN [8]	0.669	0.454	0.133
PiCANet [13]	0.548	0.356	0.156
UNet++ [30]	0.653	0.392	0.149
PSPNet [25]	0.659	0.455	0.139
MaskRCNN [5]	0.715	0.430	0.151
FPN [11]	0.677	0.483	0.131

Table 4.2. Performance comparison with state-of-the-art models that use CAMO dataset, the best value is highlighted in blue, the second best is highlighted in red

Chapter 5

Conclusions

In conclusion, the study focused on the UAV detection of camouflaged people, an important task in the military and security fields which can also provide useful insights in the more general object detection field. To solve this challenge it was crucial to consider the UAV element since the hardware inside is generally not very powerful. Thus, computation speed is a determining factor which is why a Convolutional Neural Network was used and it is built from two networks, U-Net and MobileNetV3, designed specifically for mobile and embedded devices. The encoder of the U-Net is replaced by MobileNet and the resulting model is expanded by modifying the skip pathways into nested skip pathways.
The final result is a fast network that achieves excellent scores in the metrics most commonly used to evaluate image segmentation, demonstrating its ability to detect camouflaged people without significant issues. This is especially evident in the generated images which show a silhouette very close to that of the ground truth. The results obtained using a more generic dataset are also notable, showing comparable numbers with other state-of-the-art solutions in the detection of more abstract objects and shapes.

Bibliography

[1] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. C. Loy, and D. Lin. Hybrid task cascade for instance segmentation, 2019.

[2] D.-P. Fan, G.-P. Ji, G. Sun, M.-M. Cheng, J. Shen, and L. Shao. Camouflaged object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2774–2784, 2020.

[3] D.-P. Fan, G.-P. Ji, G. Sun, M.-M. Cheng, J. Shen, and L. Shao. Camouflaged object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[4] D.-P. Fan, G.-P. Ji, T. Zhou, G. Chen, H. Fu, J. Shen, and L. Shao. Pranet: Parallel reverse attention network for polyp segmentation, 2020.

[5] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn, 2018.

[6] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam. Searching for mobilenetv3, 2019.

[7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[8] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang. Mask scoring r-cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[9] T.-N. Le, T. V. Nguyen, Z. Nie, M.-T. Tran, and A. Sugimoto. Anabranch network for camouflaged object segmentation. *Computer Vision and Image Understanding*, 184:45–56, July 2019.

[10] Y. Li, M. Lei, Y. Cheng, R. Wang, and M. Xu. Convolutional neural network with huffman pooling for handling data with insufficient categories: A novel method for anomaly detection and fault diagnosis. *Science progress*, 105:368504221135457, 11 2022.

[11] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection, 2017.

[12] J.-J. Liu, Q. Hou, M.-M. Cheng, J. Feng, and J. Jiang. A simple pooling-based design for real-time salient object detection, 2019.

[13] N. Liu, J. Han, and M.-H. Yang. Picanet: Learning pixel-wise contextual attention for saliency detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3089–3098, 2018.

[14] Y. Liu, D. Zhang, Q. Zhang, and J. Han. Integrating part-object relationship and contrast for camouflaged object detection. *IEEE Transactions on Information Forensics and Security*, 16:5154–5166, 2021.

[15] Y. Liu, Q. Zhang, D. Zhang, and J. Han. Employing deep part-object relationships for salient object detection. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1232–1241, 2019.

[16] E. Pekel and S. Kara. A comprehensive review for artificial neural network application to public transportation. *Sigma Journal of Engineering and Natural Sciences*, 35:157–179, 03 2017.

[17] S. Qiao, H. Wang, C. Liu, W. Shen, and A. Yuille. Micro-batch training with batch-channel normalization and weight standardization, 2020.

[18] X. Qin, Z. Zhang, C. Huang, C. Gao, M. Dehghan, and M. Jagersand. Basnet: Boundary-aware salient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[19] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[20] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[21] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le. Mnasnet: Platform-aware neural architecture search for mobile, 2019.

[22] Z. Wu, L. Su, and Q. Huang. Cascaded partial decoder for fast and accurate salient object detection, 2019.

[23] J. Yan, T.-N. Le, K.-D. Nguyen, M.-T. Tran, T.-T. Do, and T. V. Nguyen. Mirrornet: Bio-inspired camouflaged object segmentation. *IEEE Access*, 9:43290–43300, 2021.

[24] F. Yang, Q. Zhai, X. Li, R. Huang, A. Luo, H. Cheng, and D.-P. Fan. Uncertainty-guided transformer reasoning for camouflaged object detection. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4126–4135, 2021.

[25] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network, 2017.

[26] J.-X. Zhao, J.-J. Liu, D.-P. Fan, Y. Cao, J. Yang, and M.-M. Cheng. Egnet: Edge guidance network for salient object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[27] T. Zhao and X. Wu. Pyramid feature attention network for saliency detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[28] J. Zheng, L. Ma, Y. Wu, L. Ye, and F. Shen. Nonlinear dynamic soft sensor development with a supervised hybrid cnn-lstm network for industrial processes. *ACS Omega*, 7, 05 2022.

[29] Y. Zheng, X. Zhang, F. Wang, T. Cao, M. Sun, and X. Wang. Detection of people with camouflage pattern via dense deconvolution network. *IEEE Signal Processing Letters*, 26(1):29–33, 2019.

[30] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang. Unet++: A nested u-net architecture for medical image segmentation, 2018.

Acknowledgements

I would like to thank my supervisor, Danilo Avola, and my co-supervisor, Daniele Pannone, who guided me by offering valuable insights and advice, and above all, were always available. A heartfelt thanks goes to my parents who have always supported me both morally and financially throughout my entire academic journey. I also thank my brother, my friends, and my cousins who played an important role in helping me balance study and leisure. Finally, the biggest thanks go to my girlfriend, Federica, who encouraged and supported me especially during the most difficult moments. Her help was essential for the entire writing of this thesis.

Davide Modenese, Rome, October 2024