Detection of forest fires using artificial intelligence

Auteur : Cajot, Antoine
Promoteur(s) : Van Droogenbroeck, Marc
Faculté : Faculté des Sciences appliquées
Diplôme : Master en ingénieur civil en informatique, à finalité spécialisée en "intelligent systems"
Année académique : 2020-2021
URI/URL : http://hdl.handle.net/2268.2/11670

Detection of forest fires using artificial intelligence

Master's thesis presented
by
Antoine Cajot
to the
School of Engineering and Computer Science

carried out to obtain the degree of
Master's Degree in Computer Science Engineering

University of Liège
Liège, Belgique
Academic Year 2020-2021

Thesis advisor: Professor Marc Van Droogenbroeck Antoine Cajot

Detection of forest fires using artificial intelligence

Abstract

With the advent of climate change comes the fear wildfires will become a rising concern in the near future as is hinted by several environmental studies. This fear has already become a reality for some parts of the globe. This work implements and compares different deep learning architectures for flame semantic segmentation on RGB images of fires occurring in a natural environment taken from the ground or from an unmanned aerial vehicle (UAV). The Corsican Fire Database is exploited after comparing it to other candidate public datasets. Results are compared in terms of the intersection over union (IoU), the mean squared error (MSE), the binary accuracy and the recall metrics as well as their number of network parameters. The implemented architectures are the FLAME U-Net, the DeepLabv3+ architecture considering the EfficientNet-B4 and the ResNet-50 backbones, the Squeeze U-Net as well as the ATT Squeeze U-Net. Notable among the evaluated architectures, the DeepLabV3+ with an EfficientNet backbone was the one that achieved the best results with an IoU of 0.93 and a recall of 0.967 while exploiting 22M parameters; and the ATT Squeeze U-Net that scored very decently with an IoU of 0.893, a recall of 0.928 and the least amount of network parameters (885K). All implementations were made public.

Contents

Listing of figures

Listing of tables

Listing of Abbreviations

ADALINE Adaptive Linear Neuron. 14

AHI Advanced Himawari Imager. 6

ASPP Atrous Spatial Pyramid Pooling. vii, 34–37

AVHRR Advanced Very High Resolution Radiometer. 6

CCTV closed-circuit television. 6, 9

CNN Convolutional Neural Network. 15–17, 19

DAG directed acyclic graph. vi, 22

DSC Dice similarity coefficient. 41, 42, 70

EASA European Union Aviation Safety Agency. 8

EOS Earth Observing System. 6

ER Entity-Relation. viii, 48, 49

FCN Fully Convolutional Network. 21, 23

FLAME Fire Luminosity Airborne-based Machine learning Evaluation. 44, 53, 58, 60, 66, 70, 75

FLOPS floating point operations per second. 54, 55

FPGA Field-programmable gate array. 24

FPS Frames Per Second. 45

GPU graphics processing unit. 23

IEEE Institute of Electrical and Electronics Engineers. 15

ILSVRC ImageNet Large Scale Visual Recognition Challenge. 21

INNS	International Neural Network Society. 15
IoU	Intersection over Union. viii, 39, 42, 51, 52, 57–61, 64–70, 75
IPCC	Intergovernmental Panel on Climate Change. vi, 4
IR	infrared. 45, 76
MACs	Multiplication Accumulation operations. 29
MetOp	Meteorological Operational. 6
MODIS	Moderate Resolution Imaging Spectroradiometer. 6, 7
MSE	Mean squared error. viii, 42, 51, 57–59, 61, 64–69, 75
NIR	near-infrared. viii, 7, 8, 44, 46–48
NOAA	National Oceanic and Atmospheric Administration. 6
PA	Pixel Accuracy. 38, 39
ReLU	rectified linear unit. 21, 23, 31
RGB	Red-Green-Blue. viii, 8, 10, 44–47, 76
SNARC	Stochastic Neural Analog Reinforcement Calculator. 13
UAV	Unmanned Aerial Vehicle. 6, 7, 9, 10, 44, 46
VIIRS	Visible Infrared Imaging Radiometer Suite. 6
WSN	Wireless Sensor Network. 6, 9

I dedicate this work to all who have suffered from wildfires, hoping it will be even only modestly useful in the fight against them.

Acknowledgments

First and foremost, this work would not have been possible without the precious advice and guidance of Professor Marc Van Droogenbroeck. The same goes for Renaud Vandeghen who also allowed me a great deal of time and cues. For their help and their time I owe them my sincerest and deepest gratitude.

I could not go on without also thanking both of my parents, Isabelle and Olivier, for their infallible support throughout my studies as well as their constant interest in my personal affairs. In addition, their ability to put up so tolerably with my nonsensical behavior should also be recognized.

I want to thank my brother, François, for letting me use and abuse his GPU, in turn making his bedroom warmer than the unburnt remnants of the Amazonian Forest (a climate propitious for fungi, for instance); I also have to thank him for barking louder than our dear canid, Rimbaud, himself with a coat darker than the hearts of men.

I would also have to thank very warmly the three hooligans who go by the catchy name of "*le Quatuor des zinzins*". Starting with Cédric "*Tonton*" Mullenders, who took the time — from inside of his control tower — to check the phrasing, structure and formatting of this work; continuing with Noémie "*DePain*" Lecocq and Mattias "*Ange*" Gabriel, gracious Easter bunnies who have both watched out a lot for me during my endeavors and offered their support on multiple occasions.

Then, I want to thank Johan Browet for his incredible roleplaying skills, his shared passion for coffee, and his unexpected though very welcome last-minute assistance.

Last but far from least, I want to bow down and express my heartfelt appreciation to Maira Valencia, who has been cheering me on daily for the full duration of this thesis as well as assisted me in making sure the language of Shakespeare was not being completely butchered by rereading this manuscript. This work is also hers.

0
Introduction

0.1 A rising concern

Forest fires — as opposed to other types of wildfires, such as brush fires or grass fire — are alleged to have been ravaging forests for almost as long as trees have existed,[87] id est at least 300 million years.[110] They are first and foremost a natural phenomenon which is far older than mankind itself. Civilisations that preceded our times had to cope with forest fires or even leveraged them to manipulate the landscape either for agriculture or town planning. For instance, the Amazon's ancient inhabitants used forest fires to clean patches of land for agriculture for thousands of years (a practice still used in the same area as of today). And ere Europe's settlers colonized the Americas, indigenous tribes leveraged controlled fires both to shape the landscape as well as to reduce the risk of wildfires as part of an ongoing maintenance of their natural reserves.[93] Forest fires are also known to help sustain biodiversity (by promoting evolution) and the health of ecosystems (by stimulating rejuvenation).[11 55 76 71] However, with the advent of the Anthropocene and climate change, many are concerned their frequency will soar in response to higher air temperature, shorter and scarcer rain seasons and more intensive land use, going beyond the natural resiliency of forests. This could potentially threaten biodiversity, human constructions and lives as well as constituting in itself an economical loss for countries as wood and other types of fuels are valuable resources. Having more forests turning to ashes is not helping to cut off carbon emissions either. Though these concerns still remain very uncertain predictions in some parts of the globe, they seem to already have become reality in many areas worldwide.

In the United States, there was a clear increase in forest fires over the recent years, especially along the west coast, as can be seen in Figures 1 and 2. In Europe, large forest fires have recently affected several western and northern countries in which there were not prevalent in the past. In those countries, it was projected there would be more severe fire weather and, as a consequence, substantial expansion of the fire-prone area and longer fire seasons, particularly for high emissions scenarios. The situation is especially concerning for the southern countries around the Mediterranean Sea.[29] In Asia, similar trends have been observed as it was suggested forest fires were increasing and requiring immediate attention specifically in India, Pakistan and Vietnam[103]; increased temperatures in Siberia's peatlands as well as increased human activity are also correlated with an increase in fires.[27] The Amazonian Forest could be at risk of increased vulnerability due to climate change[27] on top of the agricultural land pressure it is already facing. Predictions for Australia are not any more optimistic either since climate change might exacerbate the fire-weather risk of any given day,[66] leading to increased frequency or intensity of extreme fire weather days as depicted in Figure 3. In Africa, the situation seems less concerning since some models predict more humid regional conditions with temperature global increase. Also, many regions of Africa are already either occupied by deserts or in the grip of desertification; this lower amount of vegetation to burn of course implies it is less likely to observe fires in these areas. However, the land (including forested areas) is subjected to an increasing amount of agricultural pressure.[27]

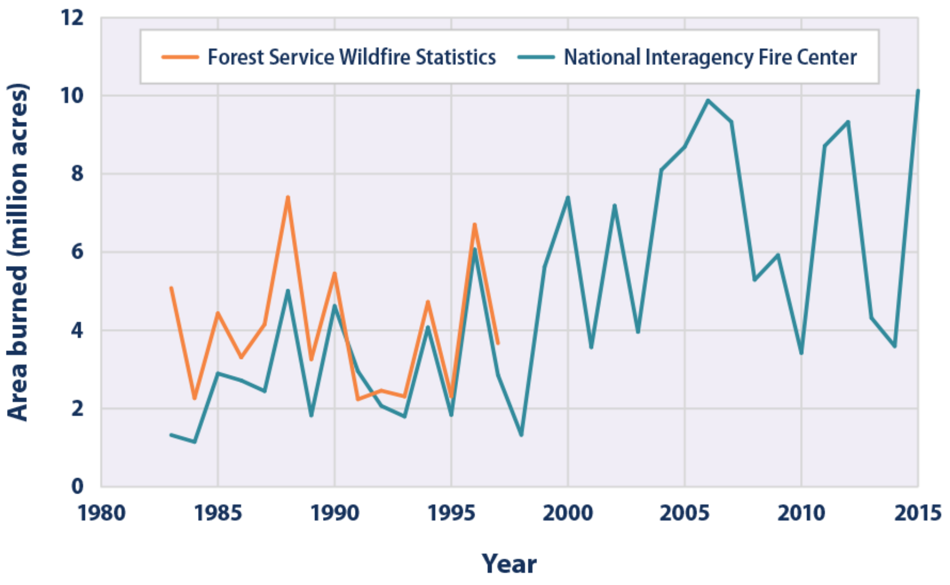


Figure 1: Annual wildfire-burned area (in millions of acres) in the United States from 1983 to 2015[28]

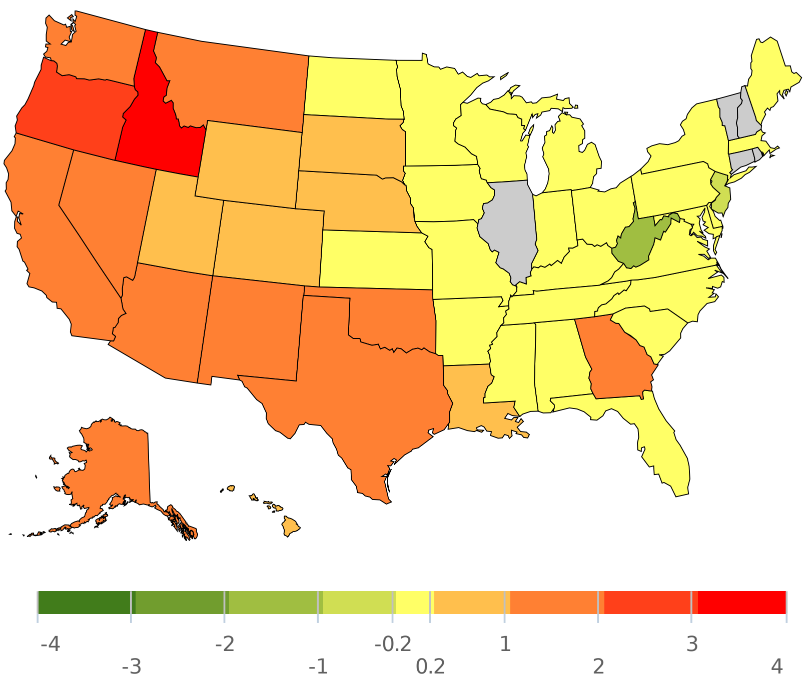
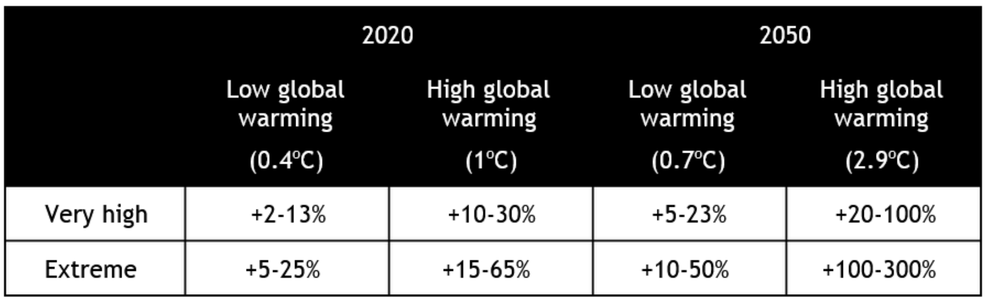


Figure 2: Change in Annual Burned Acreage by State between 1984-1999 and 2000-2014[28]



	2020		2050	
	Low global warming (0.4°C)	High global warming (1°C)	Low global warming (0.7°C)	High global warming (2.9°C)
Very high	+2-13%	+10-30%	+5-23%	+20-100%
Extreme	+5-25%	+15-65%	+10-50%	+100-300%

Figure 3: Predicted percent changes in the number of days with very high and extreme fire-weather in Australia— 2020 and 2050, relative to 1990, under the different scenarios hinted by IPCC[66]

0.2 The different types of forest fires

Forest fires are generally broadly classified in three categories: crown fires, ground fires and surface fires.[83] **Crown fires** are the most dangerous and violent type of fire for a forest: they spread swiftly over wide areas and often times kill the trees by burning them from their trunk to the crown and consuming their foliage entirely. **Ground fires** (also known as underground/subsurface fires) are the submarines of forest fires: they occur in the peaty and humus layers as well as other types of dead and dry vegetation that are beneath the litter of composed material laying on the forest floor. They produce a very intense heat but practically no visible flame. In spite of their very petty progression speed, their furtiveness may allow them to span wide areas before blazing up and becoming visible. There have been occurrences of ground fires smouldering underground for a whole winter before resurfacing in spring, after prolonged droughts. Eventually, **surface fires** are the most common type of forest fire but also the least damaging. They take place on or close to the ground and burn ground cover, scrub, saplings, etc. Because the combustible is sparse and generally thin (small branches, leaves and grass), they are easily put out. Their progression is influenced by wind's speed and direction. Naturally, these classifications are not definitive and a surface fire can evolve into a crown fire, for instance, or a fire could both be a ground and a crown fire at the same time.

0.3 Towards a problem statement

Prevention, prediction, forecasting and post-incident damage assessment are key components of the fight against forest fires; however, this work will focus essentially on detection and ongoing incident damage assessment. In particular, the methods exploiting neural networks will be given special attention since these families of algorithms have been known to show state-of-the-art performances at many tasks including detection and assessment. To detect and assess forest fires efficiently, multiple methods that leverage machine learning have been suggested in the past few years. They can first be distinguished in terms of the infrastructure that enables them to gather data to base their predictions on; e.g. satellite,[98,51,54,60,24,111,104,68,6,95,63,2,62,112] Unmanned Aerial Vehicle (UAV),[19,57,25] closed-circuit television (CCTV)[118,116,4] or Wireless Sensor Network (WSN).[1,5,90,79,7,45,46,26,115,85] Each of them has its own strengths and weaknesses which will be reviewed in the following subsections.

0.3.1 Satellites

Satellite-data is generally cheap and widely available since they have already been deployed in generous amount for various other purposes. A consequent amount of studies that have been conducted for the purpose of fire detection and damage assessment in the previous decades have used extensively satellite data originating mainly from the following sensors: *a.* the Moderate Resolution Imaging Spectroradiometer (MODIS) sensor (1995)[72] aboard the *Terra* and *Aqua* Earth Observing System (EOS) satellites, imaging the entire globe every one to two days together, *b.* the Advanced Very High Resolution Radiometer (AVHRR) sensors aboard the National Oceanic and Atmospheric Administration (NOAA) and the Meteorological Operational (MetOp) satellites (its first version, AVHRR/1, was first used in 1978 while its latest version, AVHRR/3, was first used in 1998[64]), *c.* the Visible Infrared Imaging Radiometer Suite (VIIRS) sensor (2011) designed to expand upon the data retrieved by the older MODIS and AVHRR sensor[73] and *d.* the Advanced Himawari Imager (AHI) sensor aboard Himawari 8 (2015).[53]

These sensors capture different bands of Earth's outgoing electromagnetic radiation. Each of these bands captures distinct information — for instance some bands are more suited to identify bodies of water, smoke or clouds — which can be used to decide whether or not to include them in the dataset. For reference, a table with the different bands captured by MODIS and their primary use is detailed in Table B.1. Therefore, their data can be leveraged to "see" past cloud and smoke covers and focus on wavelengths that are of interest (namely NIR in the case of forest fire).

The main issue that comes with using satellites for the purpose of detection and ongoing incident damage assessment is that in general, their revisit time — the time elapsed between their observations of the same point on Earth[56] — is too high to allow for consistency in the earliness of forest fire detection. For instance, the revisit time of MODIS (accounting for the combined data both of MODIS sensors) is 1-2 days[75] and in 1-2 days there is no telling how the fire incident situation might have worsened and spread before finally being detected and reported to firefighting units, depending on the direction and strength of the wind and the dryness of the land cover; every hour might count.

0.3.2 Unmanned Aerial Vehicles

A UAV, a.k.a. drone, is an aircraft that can be controlled either by a human pilot or by software that allows it to fly autonomously. This technology is used in many sectors including the military, the government and for commercial as well as by recreational purposes.[77] One of its main strengths is that they are widespread and their popularity is still rising. As can be seen in Figure 4, drone sale revenue has been soaring for a long time and is still expected to keep rising until at least 2025; likely way past 2025 thanks to their democratization and continuous improvement over the years. The fact there are more and more owners, including in the private sector, means it would be possible to promote their use for fire detection through incentives offered by the government (e.g. tax cuts, bonuses, ...). For instance, it was suggested during the 2019 edition of NASA's International Space Apps Challenge to use private drones lent to an association to help prevent and suppress wildfires in exchange for free battery charging; images collected by NASA satellites and by drones would be fed to a neural network model to predict higher-risk zones and the direction in which

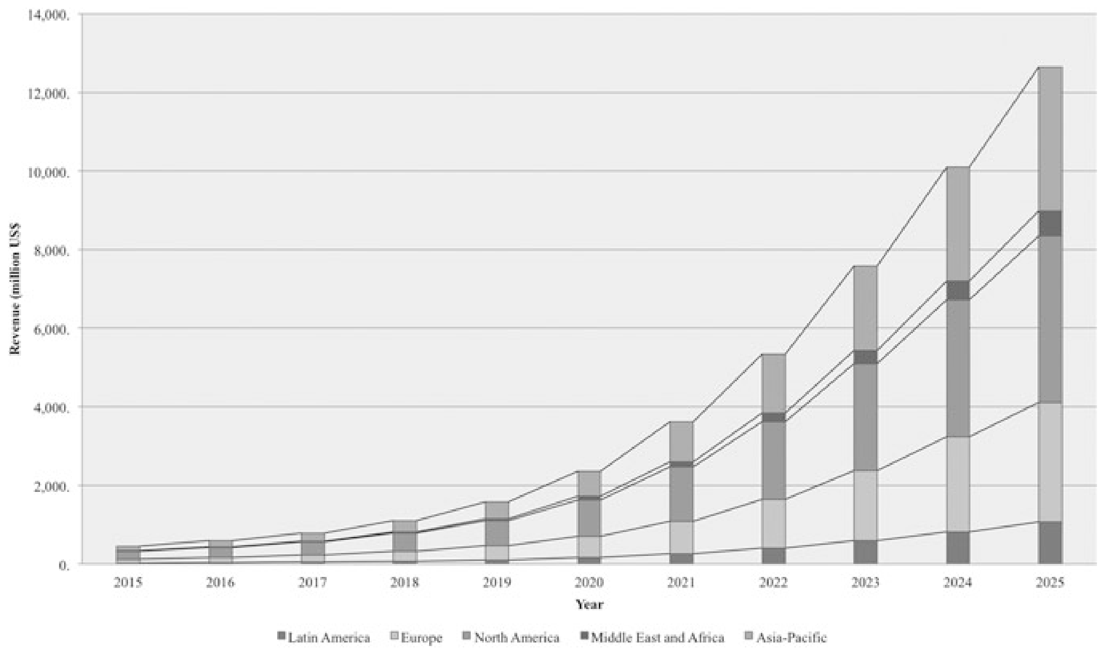


Figure 4: Projected commercial drone revenue from 2015 to 2025 (in million US dollars).[21] Source: *Drones for commercial application*, statista.com accessed 17 July 2017

fire would spread as well as help firefighters suppress the fires and restore forests by sowing.[74] Drones are cheap and versatile as many sensors can be easily equipped and unequipped including RGB or NIR sensors, plus their mobility makes them excellent candidates to patrol higher-risk areas.

On the other hand, drones have several weaknesses too. First, their popularity is still fairly new and legislation is still catching up. Legislation needs to take into account several fundamental issues drones may pose: trespassing, privacy and safety.[77] One would need to take a closer look at legislation before organizing such a fire monitoring system. Luckily, in Europe, this has been made easier by EU Regulations 2019/947 and 2019/945, who have set a unified framework for the safe operation of drones across the whole EU and European Union Aviation Safety Agency (EASA) Member States.[30]

0.3.3 Closed-circuit television

The case of CCTV, a.k.a. video surveillance, is easier to argue. CCTV is fairly cheap to deploy but it remains at a fixed position, increasing massively the required amount of hardware to cover an area of land that would more easily be covered by UAV. CCTV could also possibly be vandalized or damaged by natural hazards, which is certainly less likely for UAV and satellite.

0.3.4 Wireless Sensor Networks

WSN are carefully chosen sensors that have been disseminated around an area the condition of which they record and process, often forwarding the data they collect to one or several processing nodes called *sinks*. The conditions they observe can be temperature, sound, humidity or lighting levels, wind speed and direction, noise, the presence of certain gases, etc.[101] It is easy to see how this data could be leveraged by a neural network to predict how likely it is to have a fire (e.g. using humidity levels and temperature) or to detect to presence of fire rapidly. One very interesting study has even suggested using animals as biosensors, since their behaviors in case of fire is very useful to predict the spread and the direction of the spread of the fire (some of them hide underground, some fly or run away, ...).[85]

Compared to the other previously mentioned data-gathering infrastructures, WSNs are relatively expensive, both to deploy and to maintain. However they are arguably the ones that will provide the most data to work with. Another problem of WSNs is in case of fire, there are very high chances that nodes will be destroyed and then will have to be deployed again. They could also arguably be stolen or vandalized easily.

0.3.5 Problem statement

Since satellites have this inherently huge revisit time issue making them less valuable to detect forest fires as soon as possible, because of the huge deploying costs of wireless sensor networks plus the upkeep that comes with them and because UAVs are extremely versatile, widespread, cheap and can cover huge areas easily, this study will focus only on methods employing UAV data or data similar to UAV data.

Its aim will be to **implement and compare** different deep learning architectures for flame **semantic segmentation** on RGB images of fires occurring in a natural environment taken from the ground or from unmanned aerial vehicle (UAV); more specifically, the Corsican Fire database[99] will be exploited (the motives behind this decision will be detailed in Section 2.1.1). Results will be discussed in terms of the **intersection over union (IoU), the mean squared error (MSE), the binary accuracy and the recall** metrics. The implemented networks' **number of parameters** will also be compared.

Indeed, segmentation of flames would enable fire units to first assert if a fire is indeed present (for instance, a heuristic or algorithm for classification from the results of segmentation could be studied a posteriori) and alert fire units. Using the predicted mask paired with the altitude of the drone, fire units or another algorithm could deduce the spread of the fire to help organize fire suppression, assert damages, and possibly help decide whether to evacuate citizens in relevant areas.

0.4 Overview

Chapter 1 will focus on reviewing the different methods for image segmentation leveraging neural networks that have been researched in the recent years before giving special attention to the case of wildfire segmentation in particular. In Chapter 2, the methodology of the work presented in this manuscript will be detailed followed by Chapter 3, where results obtained will be discussed. Finally, a few steps back will be taken in the conclusion, where the shortcomings and contributions of this work will be discussed along with the questions it kindled.

1

Review of recent segmentation methods

1.1 Neural networks

1.1.1 Historical perspective

One of the earliest insightful theories about neurons was formulated in 1890 by American philosopher William James and stated that “*The amount of activity at any given point in the brain-cortex is the sum of tendencies of all other points to discharge into it, such tendencies being proportionate (1) to the number of times the excitement* *of each other point may have accompanied that of the point in question; (2) to the intensity of such excitement; and (3) to the absence of any rival point of functionality disconnected with the first point, into which the discharges might be diverted.*".[50] This inspired many researches afterwards but it was not until 1943 that McCulloch and Pitts elaborated the first credited neuron mathematical model, providing a baseline to evaluate the future state of a network or neurons, knowing its present state; "*Specification for any one time of afferent stimulation and of the activity of all constituent neurons, each an "all-or-none" affair, determines the state. Specification of the nervous net provides the law of necessary connection whereby one can compute from the description of any state that of the succeeding state [...]*".[67] Their work also proved that even simple types of neural networks could theoretically compute any arithmetic or logical function.[113] The idea of mimicking the functioning of the brain to design computers or applications was hinted subsequently by Norbert Wiener and von Neumann.[107,106] Influenced by additional works —such as Hebb's 1949 book *The Organization of Behaviour*[44], which notably drafted a learning scheme for the synapses of neurons— the first neurocomputer was built in 1951 by Marvin Minsky, christened the Stochastic Neural Analog Reinforcement Calculator (SNARC), using vacuum tubes and potentiometers (the latter ones had a role similar to weights in nowadays' neural networks); it was able to learn to solve a maze on its own. Besides simple tasks such as solving a maze, SNARC was however limited, and the first successful attempt at neurocomputing had to wait until 1957 when the Perceptron was invented by Frank Rosenblatt — an algorithm for supervised learning of binary classifiers primarily meant for pattern recognition.[81] The output of a neuron in the Perceptron algorithm could be described by Equation 1.1, where $\boldsymbol{w}$ is vector of real-valued weights that have been determined by a learning procedure, $\boldsymbol{x}$ is the real-valued input vector (for instance, an image) and b is the bias and controls how the decision boundary is shifted away from the origin.

$$f(x) = \begin{cases} 1 & \text{if} \quad \boldsymbol{w} \cdot \boldsymbol{x} + b > 0, \\ 0 & \text{otherwise} \end{cases} \tag{1.1}$$

With his Perceptron and his early book on neurocomputing, *Principles of Neurodynamics*,[82] Rosenblatt became the first pioneer of what came to be known as the first golden age of neural networks, soon followed by Bernard Widrow and his graduate student Ted Hoff in 1960, who developed Adaptive Linear Neuron (ADALINE) — which works very similarly to the Perceptron as described in Equation 1.1 but differs mainly in the learning procedure in that the weights of ADALINE are adjusted before being multiplied by the inputs and being passed through the activation function[109] — and then other researchers. This sparked a considerable enthusiasm within the research community until research hit a wall in 1969 when Minsky and Papert published *Perceptrons.* In it, they highlighted how Perceptrons were unable to compute, with efficiency, certain important predicates such as XOR. The Perceptron could only learn to separate linearly separable classes, making XOR a seemingly insurmountable barrier. In this work, they also highlighted how multilayer Perceptrons were not feasibly trainable since evaluating the weights of the neurons spread across the layers, based on the final output, would take ages to compute .[70] And that is how the ten to twelve quiet years that followed were remembered as "the AI winter", for neural computing. Neural computing research lost traction and went underground under other premises such as pattern recognition or biological modeling.[113]

It did not start thawing out before 1982 when John Hopfield published an article about a recurrent architecture now known as the Hopfield Net[47] —which consists of a single layer containing fully connected recurrent neurons— and Japan announced it would start funding Neural Network research again at the US-Japan Kyoto conference on Competition and Cooperation in Neural Nets.[3] Soon after in 1987, the Institute of Electrical and Electronics Engineers (IEEE) International Conference on Neural Networks took place in San Diego and lead to the formation of the International Neural Network Society (INNS). The INNS then founded the *Neural Networks* journal in 1988, followed by *Neural Computation* in 1989 and finally the IEEE *Transactions on Neural Networks* in 1990.[113] International interest in neurocomputing research had been rekindled and it has been in the spotlight since then.

1.1.2 Convolutional Neural Networks

The very first modern implementation of Convolutional Neural Network (CNN), called LeNet-5, was published in 1998 by Yann LeCun et al.[61] LeCun and his team were however inspired by Dr. Kunihiko Fukushima's research in the eighties, itself inspired by the work of David Hubel and Torsten Wiesel in 1962.

The latter had redefined simple and complex cells present in the primary visual cortex, the two types of cells for visual pattern recognition : "*while a simple cell can only respond to a vertical bar located in the upper section of a scene, a complex cell can respond to vertical scenes that are located anywhere in the scene.*".[14] Complex cells can achieve this location-independent capability by summing the information from several simple cells as represented in Figure 1.1.

So is it that Dr. Kunihiko Fukushima afterwards imagined an artificial neural network that mimicked the same concept of simple and complex cells which is described in his neocognitron research published in 1982.[36] It leveraged S-cells, the artificial simple cells, and C-cells, their complex counterpart. The main idea of his research is easy to grasp and again similar to the one introduced in the previous paragraph: capture

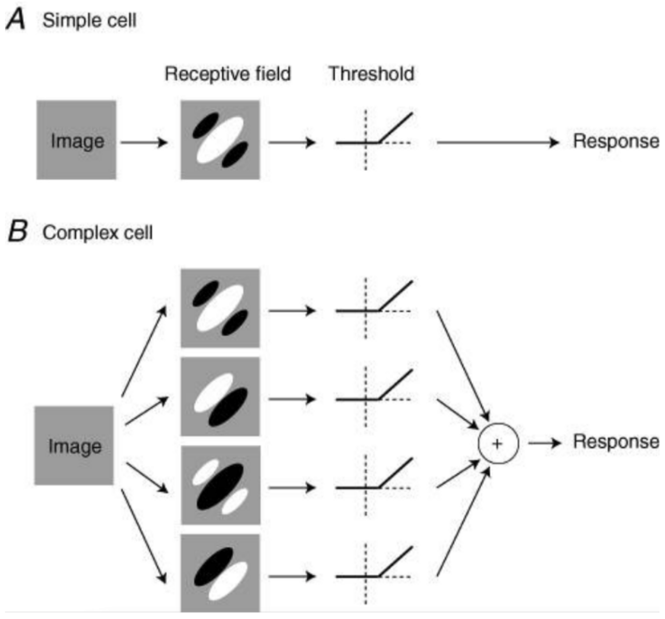


Figure 1.1: The models of simple and complex cells proposed by Movshon, Thompson and Tolhurst (1978) following the works of Hubel and Torsten[14]

complex patterns (for instance a parrot) using complex cells that aggregate data from lower-level complex cells or simple cells, themselves identifying simpler patterns (for instance feathers or a beak), organized in a hierarchy like portrayed in the Figure 1.2.

LeCun's CNN combines three main ideas to make extracted features invariant to shift, scale and distortion: local receptive fields (the equivalent of simple cells), shared weights and weight replication. It uses the same idea of combining the elementary features such as corners, edges, etc. coming from local receptive fields into higher order features: each unit in a layer receives inputs from a set of units that are located in a reduced neighborhood of the preceding layer. To make elementary features detectors useful not only on distinct small areas of the image but rather generalize better across its whole surface, they force a set of units whose receptive "cells" are perceiving different areas of the input image to share the same weight vectors.

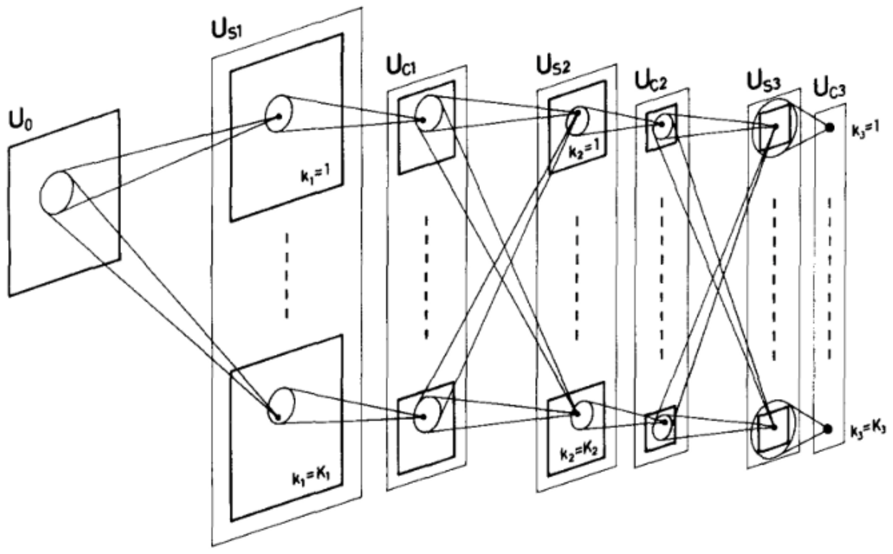


Figure 1.2: Schematic diagram illustrating the interconnections between layers in the neocognitron[36]

So, LeCun's CNN layers' units are organized in "flat" planes within, which all neurons share the same set of weights to make them work on detecting the same features, independently where they are in the image and if they were distorted or scaled. The output of such a plane of neurons is called a feature map: it is really the synthetic information computed by the plane. A **convolutional layer** is the composition of multiple feature maps (each of them with a distinct weight vector and bias) in a way that different types features can be extracted at each location of the image. This principle is illustrated in the first layer of Figure 1.3.

Therefore, it works as if each feature map "scanned" the input image with a single unit that has a local receptive field, and its resulting states while it combs the image are stored at the respective locations in the feature map. We can directly perceive why they are christened convolutional layers: they are the equivalent of a mathematical convolution that has been chained with an addition to a bias which result is passed

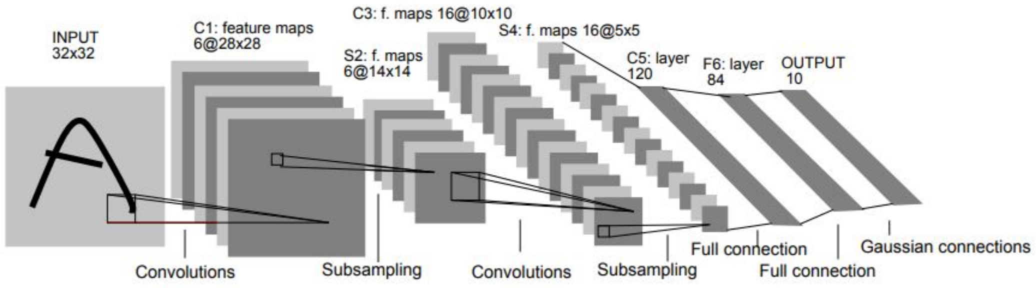


Figure 1.3: Architecture of LeNet-5, a convolutional NN, here used for digits recognition. Each plane is a feature map, i.e., a set of units whose weights are constrained to be identical.[61]

through a squashing function (such as a sigmoid for instance).

Subsequently, after features have been detected in a convolutional layer, their exact location becomes of less importance; what matters more is their relative positions compared to other features. To help the reader understand, in a handwritten number recognition system, if we understand our input image is made of a roughly horizontal segment in the upper left area, a corner in the upper right area, a roughly vertical segment stretching to the lower area then we can deduce the image corresponds to the number seven. Knowing the accurate position of all of these features is not only useless, but could also lead to not generalizing as well to other types of instances of the number seven since their exact position could vary. So, in order to reduce but not completely nullify the precision with which the position of different features is recorded in the feature map, a possible approach (the one taken here) is to reduce the spatial resolution of the feature map: this is the purpose of what are called **subsampling layers** (also sometimes referred to as **pooling layers**) in a convolutional network. Each of them conducts a local averaging and a subsampling, which has the effect of making the result more resilient to shifts and distortions. Also, in a subsampling layer, contrarily to a convolutional layer, the contiguous receptive fields of contiguous units do not overlap.

That is why convolutional and subsampling layers are generally alternated in CNNs: at each occurrence of a combination of convolutional and subsampling layer, the number of feature maps increases and the spatial resolution decreases. And thus, an important amount of invariance to geometric transformation of the input is achieved by compensating the progressive reduction of spatial resolution with a progressive increase of the richness in the representation (i.e. the number of feature maps).

CNNs have, since then, been the most used feature extractor for image tasks.

1.2 Semantic segmentation

Semantic segmentation, sometimes referred to as image segmentation or simply segmentation, is the task of determining which class (e.g. door, human, background, dog, ...) each pixel of an image belongs to.[38]

Using image segmentation to speak of semantic segmentation is, however, a misuse of language. Indeed, image segmentation could also be **instance segmentation**, for example, which goes a little further than semantic segmentation since it also distinguishes between different instances of the same object.[114] A comparative example is given in Figure 1.4.

Another widespread type of segmentation, especially in the recent years, is **panoptic segmentation** which combines the information of instance and semantic segmentation: the output of panoptic segmentation enables you to distinguish between different instances of the same object but also provides pixel-wise class information.[58]

This manuscript will solely focus on semantic segmentation, however, and will therefore use the terms segmentation and semantic segmentation interchangeably.

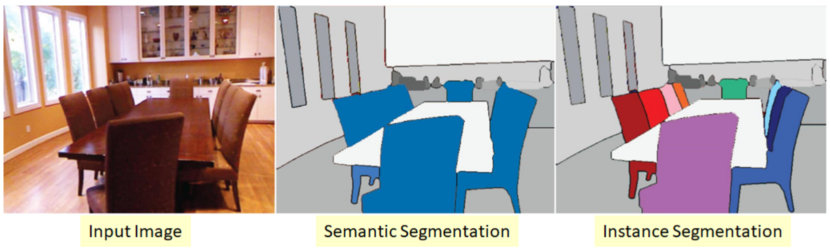


Figure 1.4: Comparison of semantic vs instance segmentation.[10] Instance segmentation distinguishes between different instances of the same object; in this case chairs.

1.3 Deep Learning architectures for Image Segmentation

Recent deep learning architectures that have been researched in the field of image segmentation include:[69]

- Fully convolutional networks
- Convolutional models with graphical models
- Encoder-Decoder based models
- Multi-scale and pyramid network based models
- R-CNN based models (for instance segmentation)
- Dilated convolutional models and DeepLab family
- Recurrent neural network based models
- Attention-based models
- Generative models and adversarial training
- Convolutional models with active contour models

It is worth mentioning that these categories are not absolute. For instance, the DeepLab family also uses an encoder-decoder structure as well as pyramid networks[16,17,15,18] or, some architectures based on encoder-decoder architectures also use attention mechanisms.[117]

This review will however focus on examining the segmentation architectures that have been studied on the Corsican Fire database[99] or similar datasets.

1.3.1 Fully Convolutional Network

Though the original Fully Convolutional Network (FCN) idea is older, it was first used for segmentation in 2015 by Long et al.[65] ImageNet Large Scale Visual Recognition Challenge (ILSVRC) classification architectures (namingly AlexNet,[59] GoogLeNet[96] and VGG-16[89]) were used as backbone they augmented with a prediction head consisting of in-network upsampling. They afterwards added skip connections between layers to merge information at local, coarse and semantic levels. This is illustrated in Figure 1.5.

1.3.2 U-Net

The U-Net architecture was designed in 2015 by Ronneberger et al. originally with biomedical image segmentation in mind and gets its name from its shape as illustrated in Figure 1.7, where the expansive path (on the right) and the contracting path (on the left) are somewhat symmetric.[80] The U-Net was designed upon the foundations laid by Long et al. and their FCN segmentation architecture with the intent that it would yield more accurate results as well as require a lot less of training samples to reach this result compared to the FCN. The contracting path is similar to any other convolutional architecture: it consists of the repetition of blocks composed successively by two 3x3 convolutions (unpadded) with rectified linear unit (ReLU) activation and

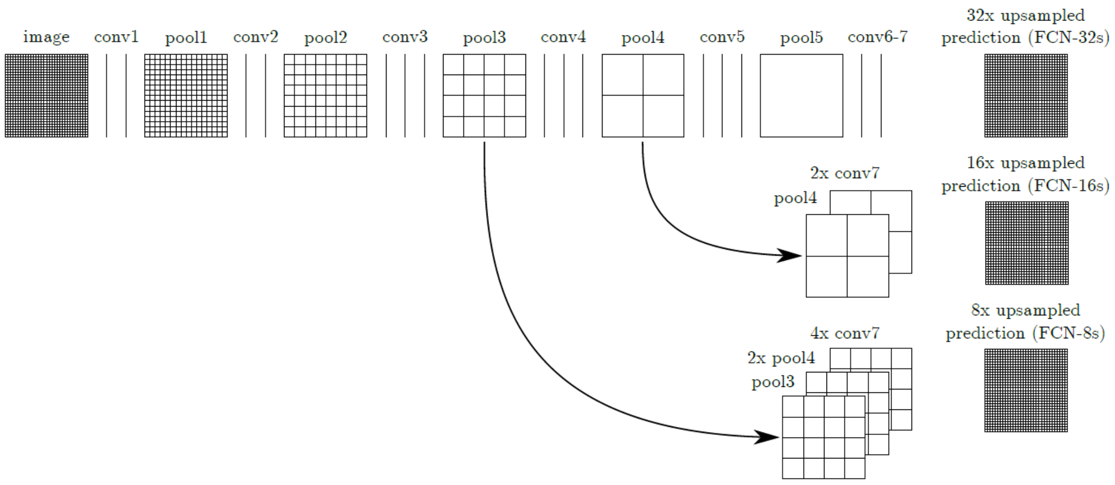


image
conv1
pool1
conv2
pool2
conv3
pool3
conv4
pool4
conv5
pool5
conv6-7
32x upsampled prediction (FCN-32s)
2x conv7
pool4
16x upsampled prediction (FCN-16s)
4x conv7
2x pool4
pool3
8x upsampled prediction (FCN-8s)

Figure 1.5: Long et al.'s DAG nets learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s): Their single stream net upsamples stride 32 predictions back to pixels in a single step. Second row (FCN-16s): Combining predictions from both the final layer and the pool4 layer, at stride 16, lets their net predict finer details, while retaining high-level semantic information. Third row (FCN-8s): Additional predictions from pool3, at stride 8, provide further precision.[65]

a 2x2 max pooling that is used for downsampling, with a stride of 2. After each of those downsampling blocks, the number of feature channels is doubled. The expansive path is also made of repetitions of an “upsampling block”. The latter is made of an upsampling of the feature map fed to a 2x2 convolution — which role is to divide the number of feature channels by two — followed by a concatenation with the relevant feature map in the contrating path and two 3x3 convolutions with ReLU activation again. The output of the expansive path is fed to a 1x1 convolutional layer to allow to map its feature vectors to the studied number of classes. The main difference between the U-Net and the FCN put forward by Long et al. is that in the U-Net’s upsampling path, a larger number of feature channels are used so that the context information can be propagated to higher resolution layers. Another important difference is that absence of fully connected layers in U-Net. Instead, only the valid part of each convolution is used, in other words, the segmentation map contains only the pixels the context of which is present in the input image; this approach enables to segment arbitrarily large images without having to worry about graphics processing unit (GPU) limitations. In order to predict masks for the areas along the border, the non-existing context data is hypothesized by mirroring the input image. This overlap-tile and mirroring process is illustrated in Figure 1.6. It is required that the input size of the network (the “tiles”) are such that every 2x2 max-pooling operation is applied to a layer with even x and y dimensions in order for the tiling of the output segmentation map to be seamless.

The U-Net, at the time of publication, outperformed all other architectures examined by the authors on various biomedical segmentation tasks. The authors were also convinced, which has been shown to be true, that their architecture could be used on numerous other tasks than their biomedical focus. Indeed, U-Net architectures remain, as of today, a popular architecture for countless image segmentation tasks

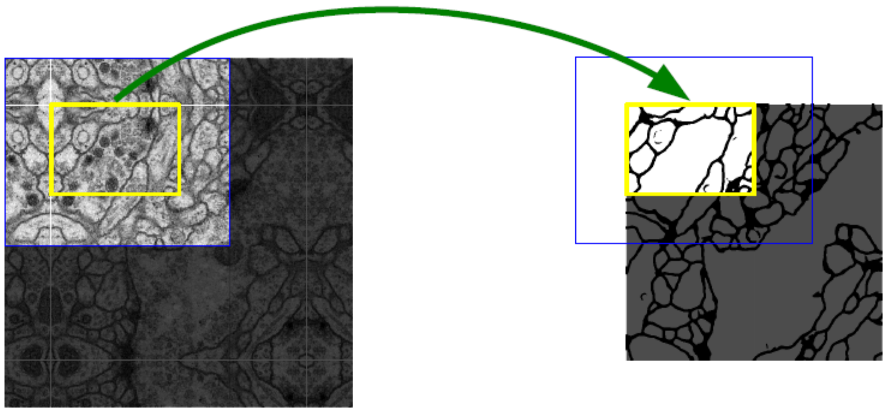


Figure 1.6: Overlap-tile strategy for seamless segmentation of arbitrary large images. Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring.[80]

such as road segmentation[119], audio separation[52,94] or, more interestingly relatively to the scope of this work, on vegetation segmentation[34,108] and fire/smoke segmentation.[33,100,117,49]

1.3.3 Squeeze U-Net

The Squeeze U-Net[8] is a hybrid between the U-Net introduced in the previous subsection and SqueezeNet drafted by F. Iandola et al.[49] Since we have already said a few words about U-Net, let us have a glimpse of the SqueezeNet architecture researched in 2016, which is originally a classification architecture.

The SqueezeNet design goals were to achieve a lightweight model size, inference time, and number of parameters, while still reaching near-state-of-the-art performances in order to make models more easily deployable over the network and on less powerful hardware such as Field-programmable gate array (FPGA). To achieve this level of compression and lightness, three strategies are leveraged. *First*, replace (most of) 3×3 filters with 1×1 filters (which results in $9\times$ less parameters). *Second*, decrease the

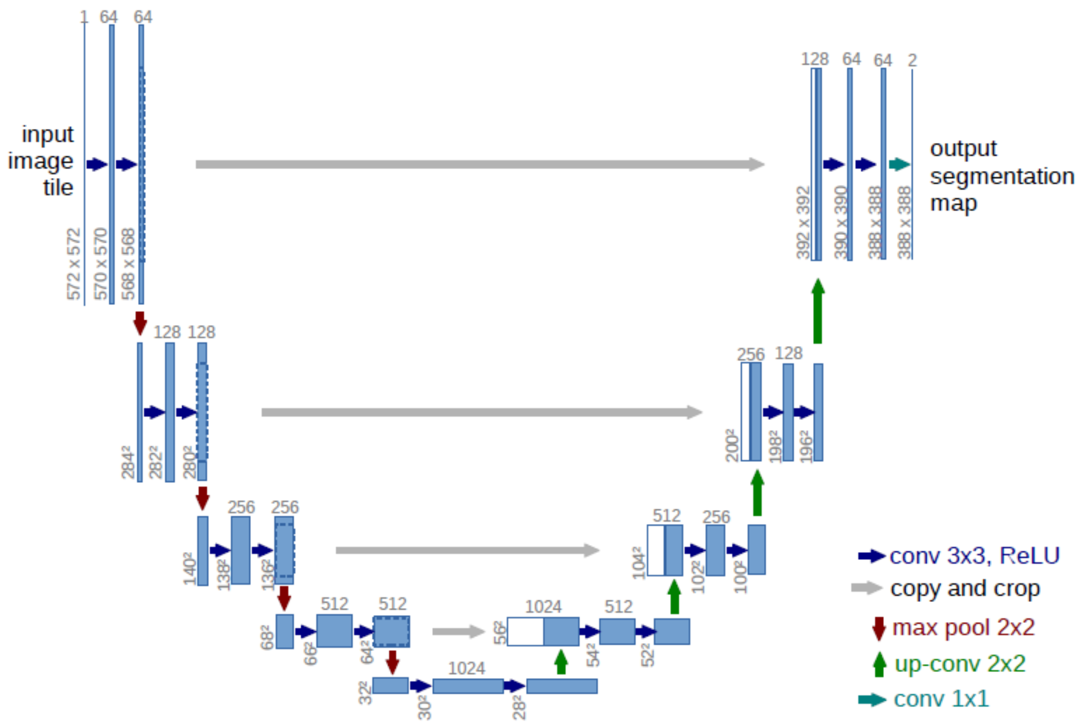


Figure 1.7: U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.[80]

number of input channels to the remaining 3×3 filters since total count of parameters in such a layer would be equal to

$$n_{input\_channels} \cdot n_{filters} \cdot 3^3 \tag{1.2}$$

The number of input channels to 3×3 filters is reduced, thanks to the *squeeze* layers described further on. These two first strategies' aim is merely to reduce the number of parameters in the network without sacrificing (too much of) accuracy. *Third*, downsample (with *strides* > 1 in convolution or pooling layers) as late as possible so that convolutional layers have larger activation maps. Their intuition is that maintaining large activation maps throughout the network by delaying downsampling (since delaying downsampling will lead to a greater number in the architecture having larger activation maps) is able to yield better accuracy, if all else is held equal; this phenomenon is indeed observed in the work of K. He and H. Sun, where they compared several CNN architecture by delaying downsampling further and further and in each case led to an improved accuracy.[41] This third and last strategy therefore aims to optimize the accuracy with a constrained number of parameters.

The two first strategies mentioned in the previous paragraph are combined in what F. Iandola et al. called the "Fire module", the core building block of the SqueezeNet architecture. The Fire module is represented in Figure 1.8. It is composed of a *squeeze* convolution layer, containing exclusively 1×1 filters (as per the *first* strategy), followed by an *expand* layer which leverages both 1×1 and 3×3 filters. The Fire module can be fine-tuned using three hyperparameters: $s_{1\times1}$ — the number of filters in the squeeze layer and $e_{1\times1}$ and $e_{3\times3}$ — the number of 1×1 and 3×3 filters in the expand layer,

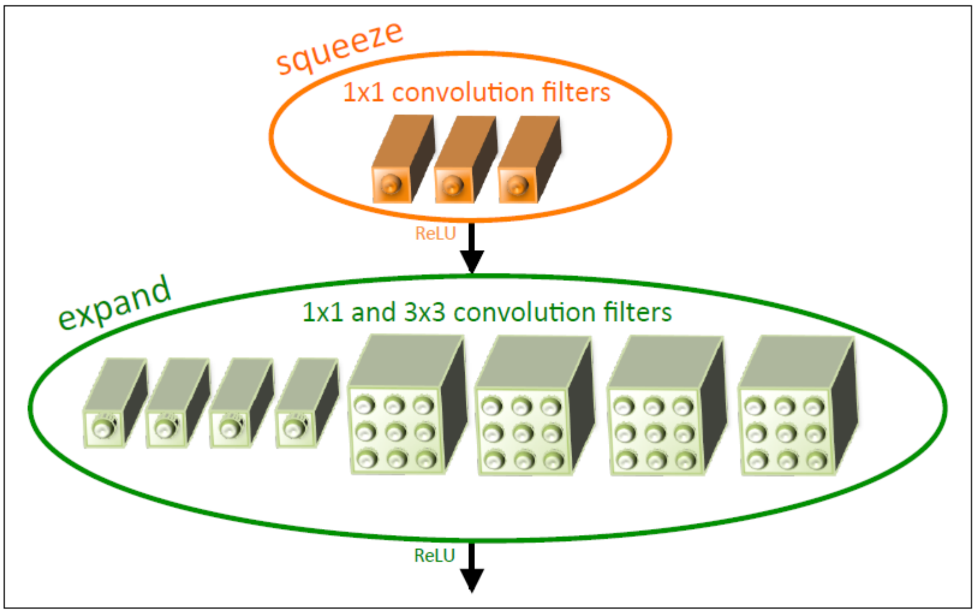


Figure 1.8: Organization of convolution filters in the Fire module.[49] In this case, hyperparameters $s_{1\times1} = 3,\ e_{1\times1} = 4$ and $e_{3\times3} = 4$ are used.

respectively. The second strategy is applied by enforcing the following criterion

$$s_{1\times1} < (e_{1\times1} + e_{3\times3}) \tag{1.3}$$

so that indeed the *squeeze* layer makes sure to limit the number of input channels to the 3×3 filters present in the *expand* layer.

The full SqueezeNet architecture is illustrated in Figure 1.9. It is compared with two variants where bypass connections have been included so that the outputs of two layers can be summed element-wise before being fed to the next. The only requirement in that case is that the the number of input channels has to be equal to the number of output channels. This case is referred to as *simple bypass* by the authors, since it does not include additional convolutions, it does not involve extra parameters. In order to bypass the same number of channels limitation, the authors imagined a

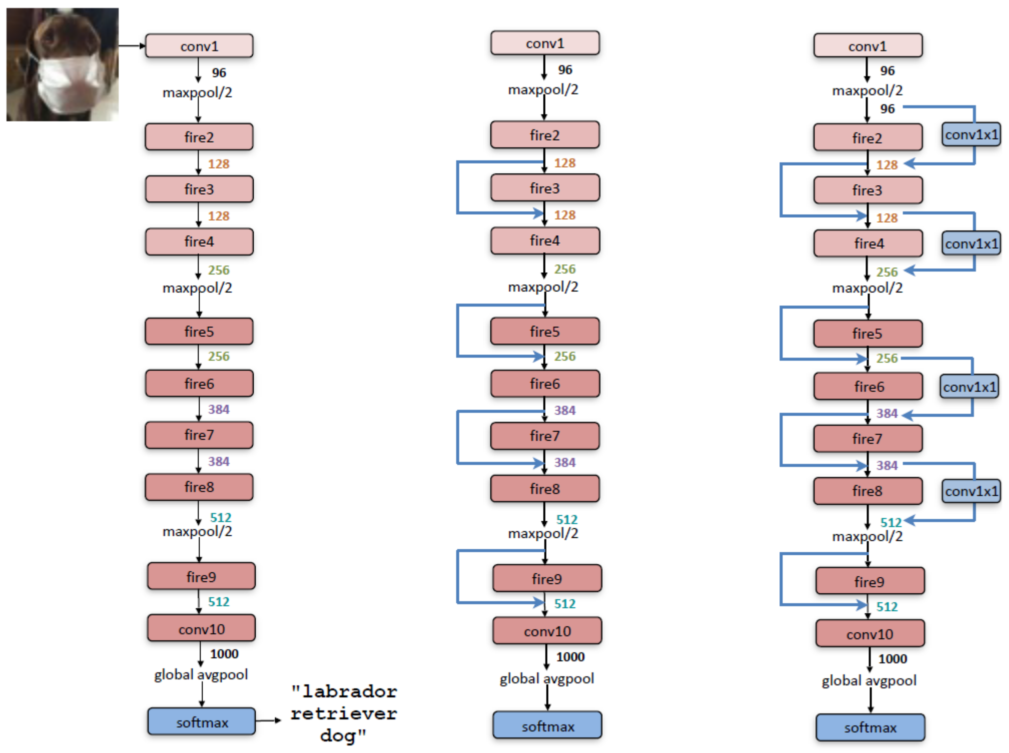


Figure 1.9: Macroarchitectural view of F. Iandola et al.'s SqueezeNet architecture. Left: SqueezeNet; Middle: SqueezeNet with simple bypass; Right: SqueezeNet with complex bypass[49]

complex bypass connection including a 1×1 convolution accustoming to the required number of output channels by employing an equal amount of filters. However, whereas their simple bypass did not clutter the architecture with additional parameters, the complex bypass does. Surprisingly enough, the complex bypass also did not perform better than the simple bypass, which scored the best performances amongst their three variations of the SqueezeNet.

The idea to adapt the SqueezeNet to a U-Net architecture (aptly christened Squeeze U-Net) had to hold until recently, in 2020.[8] SqueezeNet's Fire modules were used in the contracting path. For downsampling, instead on relying on max or average pooling like in the original U-Net architecture, stride 2 convolutions are used instead, since it is mentioned that striding improves the expressiveness of the network.[92] In the expansive path, their transposed version of the Fire module consists of a 1×1 transposed convolution, followed by two parallel transposed convolutions (2×2 and 1×1 respectively), each with half of the output channels of the transposed Fire module. Their result is concatenated to form the output. The full Squeeze U-Net architecture is represented in Figure 1.10 and a comparison of the upsampling and downsampling blocks between Squeeze U-Net and the original U-Net architecture is offered in Figure 1.11. Squeeze U-Net achieved 12× fewer parameters and 3× fewer Multiplication Accumulation operations (MACs) than U-Net. It also trained 69% faster and had a 17% shorter inference time. As a trade-off, the accuracy dropped from 86.9% for U-Net to 78% for Squeeze U-Net when evaluated on a fraction of the CamVid Database.[12,13]

1.3.4 ATT Squeeze U-Net

Another alteration of the SqueezeNet architecture, introduced independently from the Squeeze U-Net of Beheshti and Johnsson[8], also saw the light of day in 2020. This time it also involved attention mechanisms (drawing its inspiration from both the work of Oktay et al. in 2018 (Attention U-Net)[78] and the original SqueezeNet paper[49]); it was very appropriately called ATT Squeeze U-Net.[117]

The main idea of the ATT Squeeze U-Net is to use the SqueezeNet as a feature extractor and use additive attention gates at the skip connections of the U-Net architecture. The general architecture is represented in Figure 1.12 and the attention gate used at the skip connections is illustrated in Figure 1.15. However, Zhang et al. also modified the SqueezeNet's Fire module in order to improve feature communication and reduce computational cost by replacing the 3×3 convolutional layer in the original

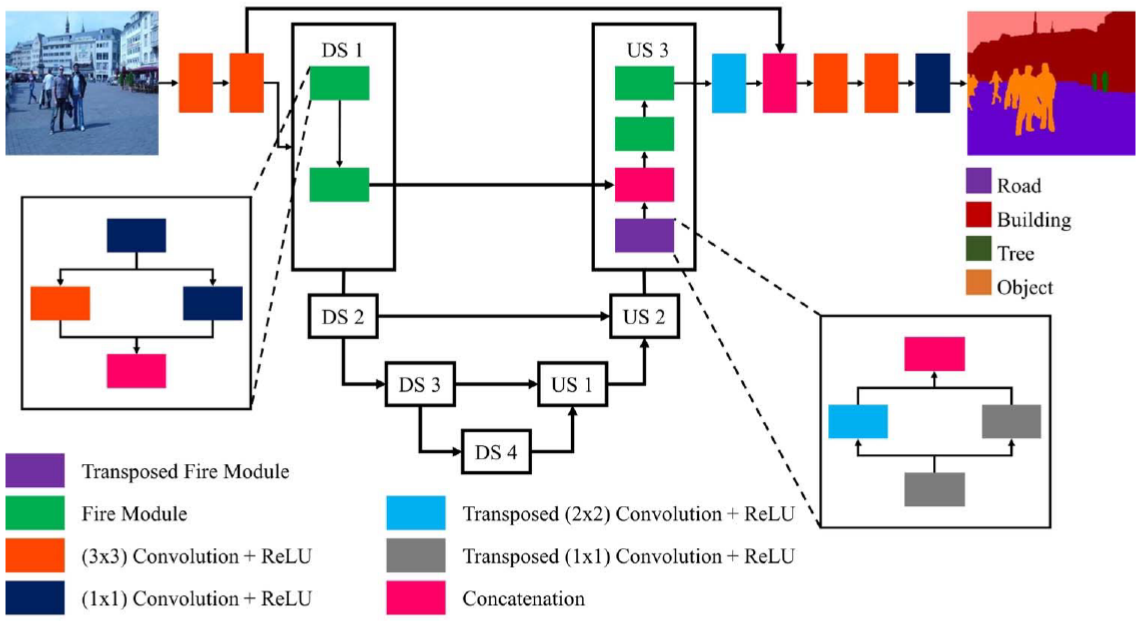


Figure 1.10: The Squeeze U-Net architecture. The downsampling units in the contracting path each consists of two fire modules which extract features. The extracted features are passed down to the next downsampling unit and to the corresponding upsampling unit. Every upsampling unit in the expansive path consists of a transposed fire module, a concatenation unit and two fire modules which in order upsamples their input, extract features, and concatenate features to construct the output. US and DS stand respectively for upsampling and downsampling.[8]

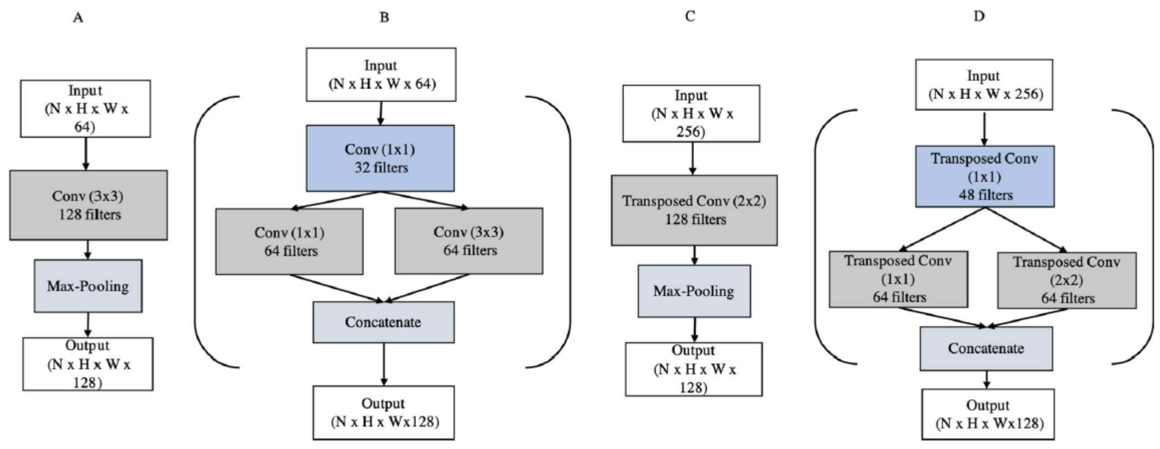


Figure 1.11: Comparison of convolutions in the contracting path (A and B) and transposed convolutions in the expansive path (C and D) between the original U-Net implementation (A and C) and the Squeeze U-Net implementation (B and D).[8]

Fire *expand* module by a depthwise convolution (which secures good feature learning abilities while decreasing required computations) followed by a channel shuffle (to allow inter-channel information flow). This is illustrated in Figure 1.13. The equivalent of the transposed Fire module present in the Squeeze U-Net architecture responsible for the upsampling steps in the U-Net was, in ATT Squeeze U-Net, called the DeFire module. Like the transposed fire module, it is made of an *extend* layer followed by a *squeeze* layer; the extend layer is composed of 1×1 and 3×3 filters in parallel and passed through a ReLU activation, the squeeze layer is made of a 1×1 convolution filter, followed by a resampling layer that is fed through a 3×3 filter. The DeFire module is illustrated in Figure 1.14.

The Corsican Fire DB was used as dataset in this study. They obtained an accuracy of 0.907 and a Dice coefficient of 0.8750 with a reduced number of parameters compared to other architectures evaluated in that paper (7.96M). Their implementation was not publicly disclosed.

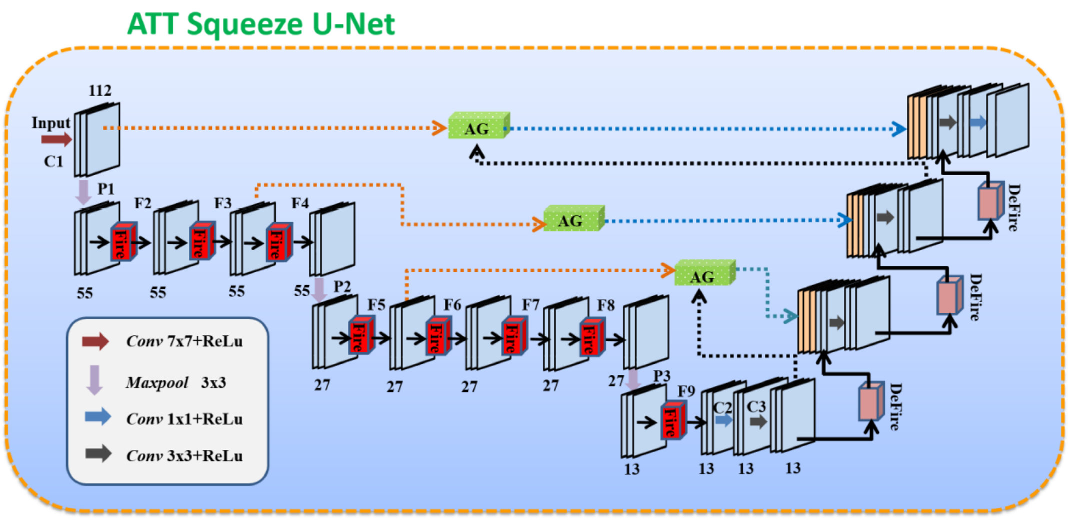


Figure 1.12: The ATT Squeeze U-Net architecture. The contracting path uses a SqueezeNet architecture with eight modified Fire modules. The expansive path incorporates three DeFire modules that take the same ideology as the modified Fire module. Three attention gates at the skip connections concatenate encoder and decoder features.[117]

1.3.5 DeepLabv3+

The DeepLab family of network architectures is a series of incremental improvements upon a first architecture called DeepLab first published in 2014,[16] followed by its second version in 2016,[17] and its third declination in 2017.[15] The latest flavor of DeepLab, called DeepLabV3+, arrived in 2018.[18] Only the latter version will be reviewed in this subsection. The different components of DeepLabv3+ will now first be described before its architecture is in turn presented.

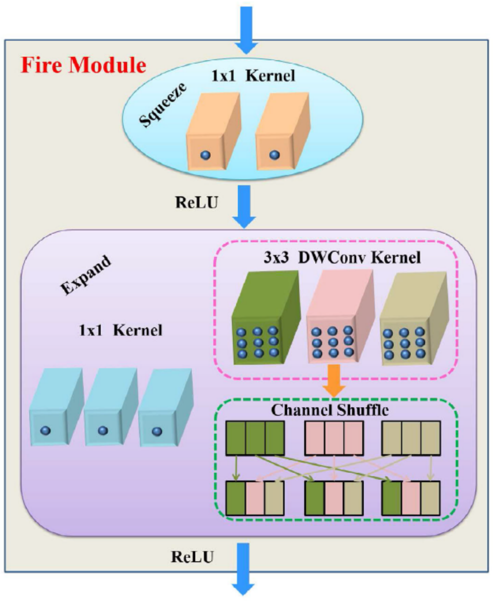


Figure 1.13: Structure of the the modified Fire module in ATT Squeeze U-Net.[117]

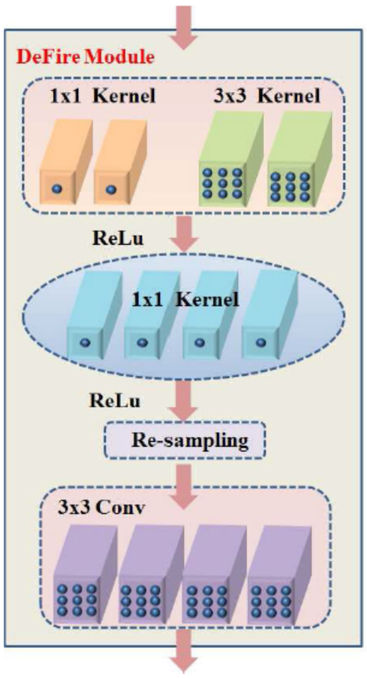


Figure 1.14: Structure of the DeFire module in ATT Squeeze U-Net.[117]

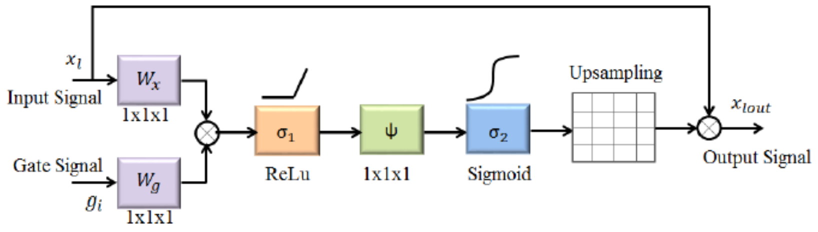


Figure 1.15: Schematic of the additive attention gate used in ATT Squeeze U-Net[117]

Atrous convolutions

Atrous convolutions (from French "à trous" which roughly translates to "with holes"), also sometimes referred to as dilated convolutions or hole algorithm, are a generalization of standard convolutions where it is possibly to use a distance, called the *rate*, between elements of the kernel. The output of such a convolution can be written

$$y[i] = \sum_k x[i + r \cdot k]w[k] \tag{1.4}$$

where y is the output, w is a filter of length K, x is the input feature map and r is the atrous rate, akin to a sampling stride applied on the input signal. Using r equal to 1 comes down to performing a vanilla convolution. Changing the value of r can be interpreted as adjusting the field-of-view of the filter. It also allows to compute the response, at any resolution desired, of any layer in a network, without changing the number of filter parameters or the number of operation per position.[17] A visualization of the functioning of atrous convolutions is given in Figure 1.16.

Atrous Spatial Pyramid Pooling

ASPP is the approach that helped DeepLabv3 (and therfore DeepLabv3+ of which it is a component) to better handle scale variability. Inspired by the R-CNN spatial

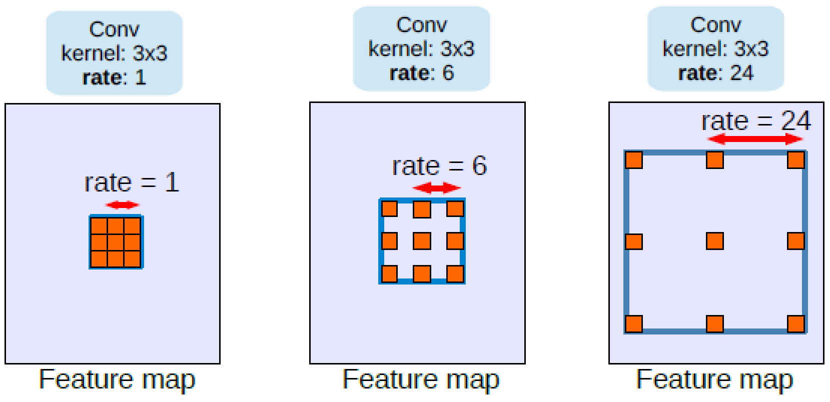


Figure 1.16: Example of atrous convolutions with different rates[15]

pyramid pooling method of He et al,[42] ASPP consists of several atrous convolutional layers performed in parallel with different rates (which allows to resample the input features at different scales), the result of each of which is handled independently in a separate branch before being merged all together to generate the output; this process is illustrated in Figure 1.17.

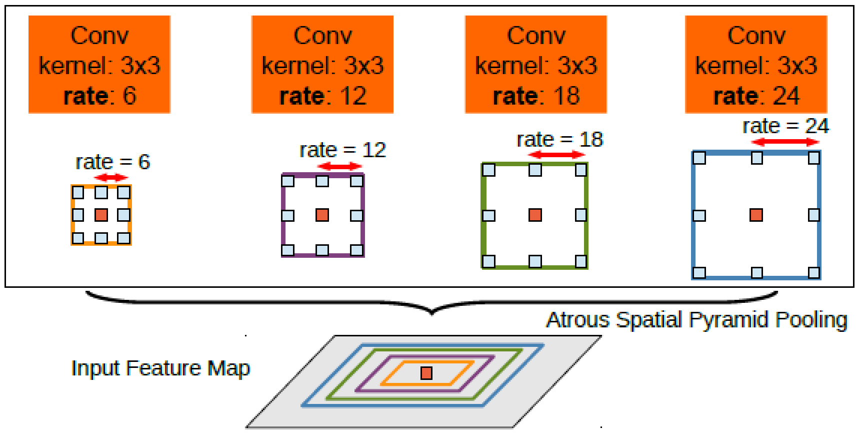


Figure 1.17: Atrous Spatial Pyramid Pooling (ASPP). To classify the center pixel (orange), ASPP exploits multi-scale features by employing multiple parallel filters with different rates. The effective Field-Of-Views are shown in different colors.[17]

Depthwise separable convolution

The depthwise separable convolution's aim is to reduce computation complexity of convolutions without harming the quality of the result. To do so, a vanilla convolution is emulated by combining in order a depthwise convolution with a pointwise convolution. A depthwise convolution is a type of convolution that keeps a separate filter for each input channel, the channel are therefore convolved independently.[48] A pointwise convolution is simply a 1×1 convolution that is leveraged to combine the output of the depthwise convolution (to, this time, use the information present in different channels). The authors use atrous convolution in the depthwise convolution, and christened the result atrous separable convolution. A representation for depthwise convolution, pointwise convolution and atrous separable convolution is provided in Figure 1.18.[18]

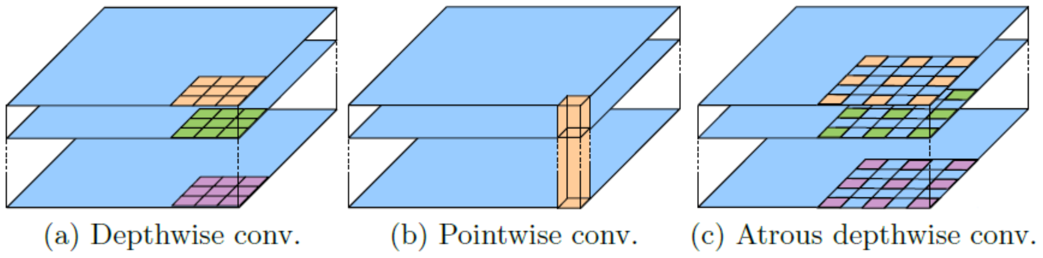


Figure 1.18: 3×3 Depthwise separable convolution decomposes a standard convolution into (a) a depthwise convolution (applying a single filter for each input channel) and (b) a pointwise convolution (combining the outputs from depthwise convolution across channels). In DeepLabv3+, atrous separable convolution are explored where atrous convolution is adopted in the depthwise convolution, as shown in (c) with rate = 2.[18]

DeepLabv3 encoder

The DeepLabv3 architecture is used as encoder in DeepLabv3+ and is represented in Figure 1.19. Their ASPP (modified in comparison to the first version of the ASPP

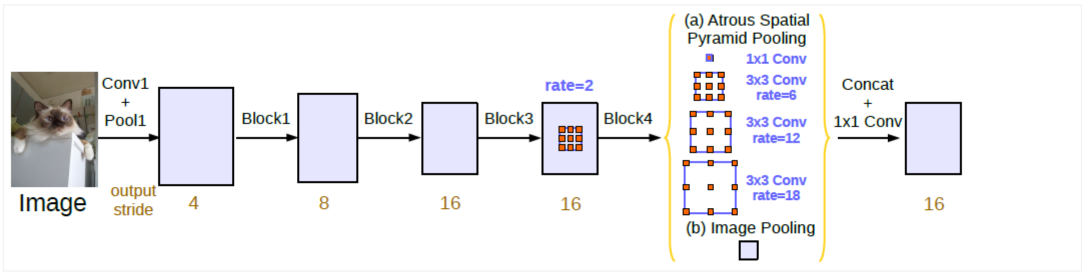


Figure 1.19: DeepLabv3 architecture. Parallel modules with ASPP (a), augmented with image-level features (b).[15]

drafted in DeepLabv2) is made of one 1×1 and three 3×3 convolutions with rates $(6, 12, 18)$, all with 256 filters and batch normalization as well as image-level features. Resulting features from all branches are then concatenated before being fed to another 1×1 convolutions right before the final 1×1 convolutions outputting the last logits. And that is how they employ atrous convolutions to extract dense features from their backbone (they used ResNet[43]). The map right before the final 1×1 convolutions is used as encoder output.

Decoder

The decoder is designed similarly to the U-Net expansive path, with upsampled features being concatenated with the corresponding low-level features from the backbone that share the same dimensions to help recover object segmentation details. After the concatenation, a few 3×3 convolutions are applied before jumping into the next upsampling block.[18] The final encoder/decoder resulting architecture is illustrated in Figure 1.20.

Application to the Corsican Fire Database

Even though the original DeepLabv3+ architecture was tested on the PASCAL VOC 2012 semantic segmentation benchmark,[31] it was also leveraged by Harkat et al. in

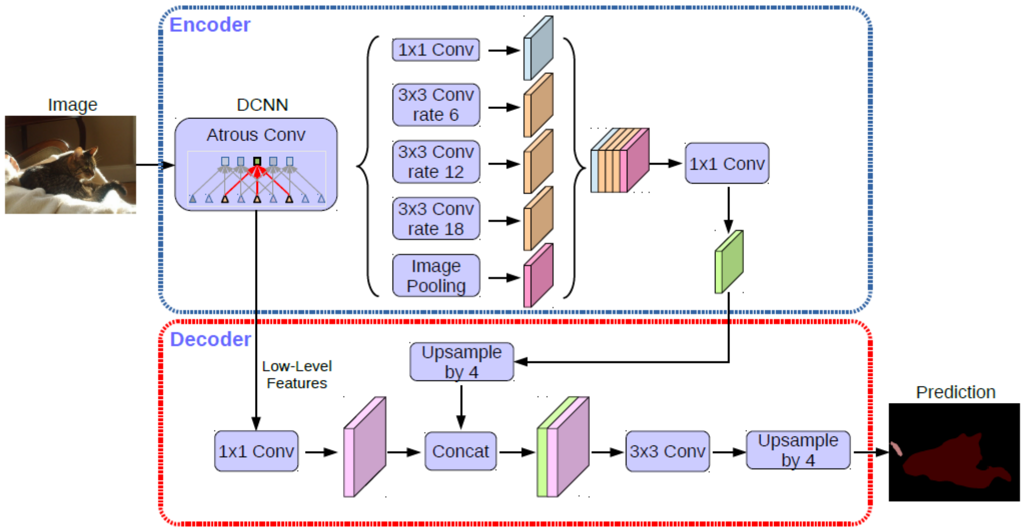


Figure 1.20: DeepLabv3+ extends DeepLabv3 by employing a encoder/decoder structure. The encoder module encodes multi-scale contextual information by applying atrous convolution at multiple scales, while the simple yet effective decoder module refines the segmentation results along object boundaries.[18]

2020 on the Corsican Fire Database using a ResNet-50 backbone.[40] However neither their research nor their implementation was made public and it was therefore impossible to compare it further with other works.

1.4 Frequently employed metrics in Image Segmentation

1.4.1 Pixel Accuracy

Pixel Accuracy (PA) corresponds to the ratio of pixels that have been correctly classified averaged over the total number of pixels. If we consider K classes (including the background class), if we set down the number of pixels of class i that have been predicted has belonging to class j as p_{ij}, then PA can be written

$$PA = \frac{\sum_{i=0}^{K-1} p_{ii}}{\sum_{i=0}^{K-1} \sum_{j=0}^{K-1} p_{ij}} \tag{1.5}$$

From there, one can define mean PA (then usually written MPA) as the PA averaged over the total number of classes.

$$MPA = \frac{1}{K} \sum_{i=0}^{K-1} \frac{p_{ii}}{\sum_{j=0}^{K-1} p_{ij}} \tag{1.6}$$

1.4.2 Intersection over Union

The IoU, also sometimes referred to as the Jaccard Index, is one of most frequently metrics used in image segmentation tasks; it can be formulated as the overlapping area between the predicted mask and the ground truth weighted by the inverse of the area of union between the predicted mask and the ground truth. If the predicted mask area is set down as A_{pred} and the ground truth area as A_{GT}, then the IoU can be written as:

$$IoU = \frac{|A_{GT} \cap A_{pred}|}{|A_{GT} \cup A_{pred}|} \tag{1.7}$$

Just as for the MPA, the IoU (then called the mean IoU) can also be averaged over all classes.

1.4.3 Precision

Precision is the proportion of positive identifications that were actually correct. It can be formulated as

$$Precision = \frac{TP}{TP + FP} \tag{1.8}$$

where TP is the number of true positives and FP is the number of false positives. As can be seen more clearly in this formulation, only the number of false positives negatively impacts this metric.

1.4.4 Recall

Like precision, recall is a metric that is also derived from the confusion matrix and corresponds to the proportion of actual positives that were identified correctly. It can be written

$$Recall = \frac{TP}{TP + FN} \tag{1.9}$$

if TP denotes the number of true positives and FN the number of false negatives.

Maximizing recall leads to minimizing the number of omission of true positives. It is therefore clear that to assess properly how effective a model is, both precision and recall must be evaluated; however these metrics are conflicting with each other and it often happens that improvements on recall lead to reductions on precision and vice-versa.

1.4.5 Accuracy

Accuracy is a metric that is conceptually easy to grasp and therefore always nice-to-have, even if it does not suffice on itself to capture how effective a model is. It corresponds to the number of correct predictions over the total number of predictions. Or, if one musts formulate it using the confusion matrix again, then

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.10}$$

where TP, TN, FP and FN are respectively the number of true positives, true negatives, false positives and false negatives.

1.4.6 F_1-Score

The F_1-score, also called F_1-measure, is the harmonic mean of precision and recall.[86] It can be written

$$F_1 = \frac{2}{recall^{-1} + precision^{-1}} = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{1.11}$$

The F_1-score, though widespread, has been condemned for giving equal importance to precision and recall since their weight against one another is part of the formulation of the problem: a discussion to prioritize one or the other will lead to tailoring models that are more adapted to specific situations.[39]

1.4.7 Sørensen–Dice coefficient

The Sørensen–Dice coefficient, most commonly referred to as Dice's coefficient or Dice similarity coefficient (DSC), is a metric that was reimagined twice and independently by two botanists: Thorvald Sørensen and Lee Raymond Dice in 1948 and 1945 respectively; hence its name.[91,23] It can be used to assess how similar two sets are and, in the case of segmentation, can be formulated as

$$DSC = \frac{|A_{GT} \cap A_{pred}|}{|A_{GT}| + |A_{pred}|} \tag{1.12}$$

When applied to boolean data, the DSC is actually the same as the F_1-score, as indeed it can then be expressed as

$$DSC = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{1.13}$$

using the same notations as introduced earlier.

The DSC is very similar to the IoU: they both range from 0 to 1, are positively correlated and, what is more, one can be determined from the other using the following relation:

$$IoU = \frac{DSC}{2 - DSC} \tag{1.14}$$

They are still different metrics however and one of their main differences, in practice, is that IoU tends to penalize small classification errors more than DSC.

1.4.8 Mean squared error

The MSE is a metric originally introduced by Gauss[37] and literally measures the average of the squares of the errors: it is derived from the Euclidean distance between the predicted value and the correct value. In the case of pixel-wise binary classification, which corresponds to a segmentation task with only one class, if one sets down the predicted value of the i-th pixel of the j-th image as p_{ij} and its correct value as c_{ij}, then the MSE of the predictions over the images in the test set could be written as

$$MSE = \frac{1}{M} \sum_{i=0}^{N} \sum_{j=0}^{M} (p_{ij} - c_{ij})^2 \tag{1.15}$$

where M is the number of images in the test set and N is the fixed number of pixels in input images/predicted masks. It is a positive quantity that decreases with the error. The MSE is in fact the second central moment of the error and therefore accounts for the variance of the model as well as its bias, it is in this sense a valuable metric.

2 Methodology

This chapter will focus on detailing the methodology behind the implementations this manuscripts is based on. First, considerations pertaining to the dataset will be weighted in Section 2.1. Second, the evaluation framework that allows to compare results meaningfully will be described in Section 2.2. Third and last, the implementation of architectures will be discussed in Section 2.3.

2.1 Data

2.1.1 Choosing a dataset

According to the scope of this work, several candidate datasets were contemplated. The criteria used to select a relevant dataset in accordance with the problem statement (described in Section 0.3.5) are:

- the presence of RGB images of fires occurring in a natural environment;
- the inclusion of shots taken from UAV or at a distance from the fire equivalent to the altitude of one;
- the presence of manually delimited ground-truth masks.

There could also be other discriminating factors such as the number of pictures they contain, their resolution, the diversity of fires (in size and position) as well as of natural environments in which they occur or the presence of additional data in the dataset that could present an interest to help create more accurate predictions when used as input in the model such as UAV flight altitude, air temperature or NIR sensor data for instance.

The FLAME dataset

The Fire Luminosity Airborne-based Machine learning Evaluation (FLAME) dataset[9] contains several repositories of UAV image and video of a prescribed pile burn in Northern Arizona, USA. In particular, the nineth repository contains 2,003 3480×2160 video frames of the burn in JPEG format and the tenth repository contains the corresponding 2,003 manually segmented ground-truth flame masks in PNG format. These are per se very reasonable figures to work with to train and evaluate segmentation architectures but the drawback is that these 2,003 frames are highly correlated to one another since they are successive shots over a small of time of the same fire (it was captured at 29 Frames Per Second (FPS)). Also, the dataset does not capture a great diversity of fires nor of environments since it is only frames of the same fire taking place in the same environment and therefore a model trained on this dataset is bound to have a poor generalization ability.

Foggia dataset

The Foggia dataset[35] contains 31 videos of different fires, not all of them occurring in a natural environment. It also contains 149 non-fire videos that feature difficult situations that are often interpreted as fire by traditional color-based (non deep learning) approaches, such as a mountain at sunset, lens flares or red roof houses in a wide valley. Since no fire is present in the latter collection, the ground-truth mask can immediately be deduced and it would be interesting to see in which proportion the model trained on another dataset would misclassify some of these frames as fire and to compare this from one architecture to another. Access to this database was requested but no response was received and it could therefore not be studied further.

FireNet dataset

The FireNet dataset[25] contains short videos of wildfires of about 150 frames that have been collected over the previous years in various locations of the United States. It boasts around 400,000 annotated frames, 100,000 of which contain an active fire. All frames were captured in infrared (IR) since RGB sometimes makes it harder to see the whole fire perimeter because of various factors such as smoke for instance. Annotations have been performed with the help of experts from Californian Department of Forestry and Fire Protection and Air National Guard. Authors of this database were also contacted to request access to their dataset but no answer was returned.

Corsican Fire Database

The Corsican Fire Dabatase[99] was assembled after a call for wildland fire images was made. The images it is made of come from different parts of the world, and were taken by various researchers and partners. It also contains images in the NIR spectrum (with RGB in parallel). All images, including ground-truth and NIR images are in PNG format. It was assembled in 2017 after noticing there was no large public database for wildland fire images, bearing in mind it would be even more useful if it was an evolving database accepting outside contributions. Access to this database was requested and very rapidly granted. It was favored against the others because it has a large panel of images that have been well characterized, includes a greater variety of fires and environments, has NIR and RGB data in parallel as well as the manual segmentation mask, and contains images taken from UAV (though a great deal of these shots are also taken from the ground). Since it was the dataset that was selected to work on, it is further characterized in the following subsection.

2.1.2 Characterizing the dataset

The Corsican Fire Database contains 1135 RGB images, 634 of which have NIR data. Among the 634 images that have NIR data, 540 are part of 5 different video sequences of 5 different fires (which can trivially be explained by the fact those 634 images were captured by the same sensor, a JAI AD-080GE camera); this composition is illustrated in Figure 2.1. Additionally, there are 419 different image sizes present in the dataset; the number of pictures for the ten most represented dimensions is illustrated in Figure 2.2. The most represented image size can be justified since all of the 634 shots taken with the JAI AD-080GE camera have the same 1024×768 dimension.

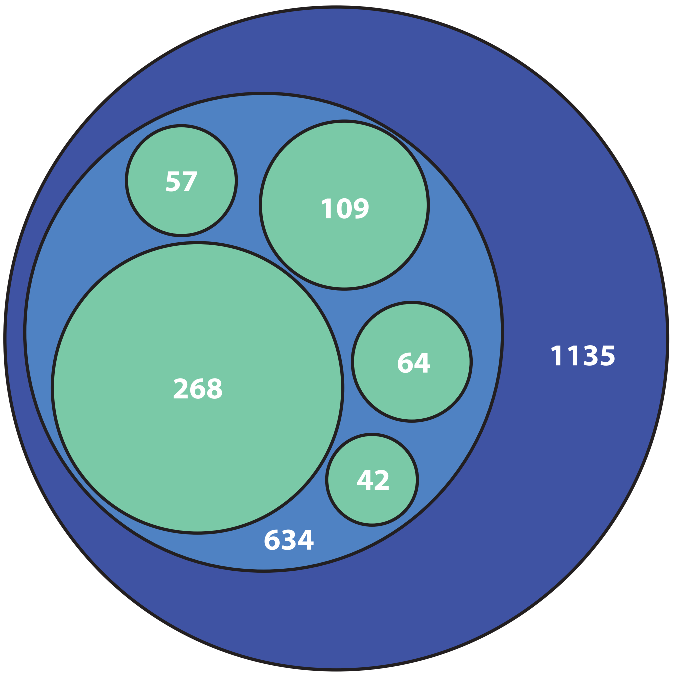


Figure 2.1: Composition of the Corsican Fire Database. Out of the **1135** RGB images composing the dataset, **634** of which have NIR data. Among the latter, **540** are part of 5 different video sequences of 5 different fires

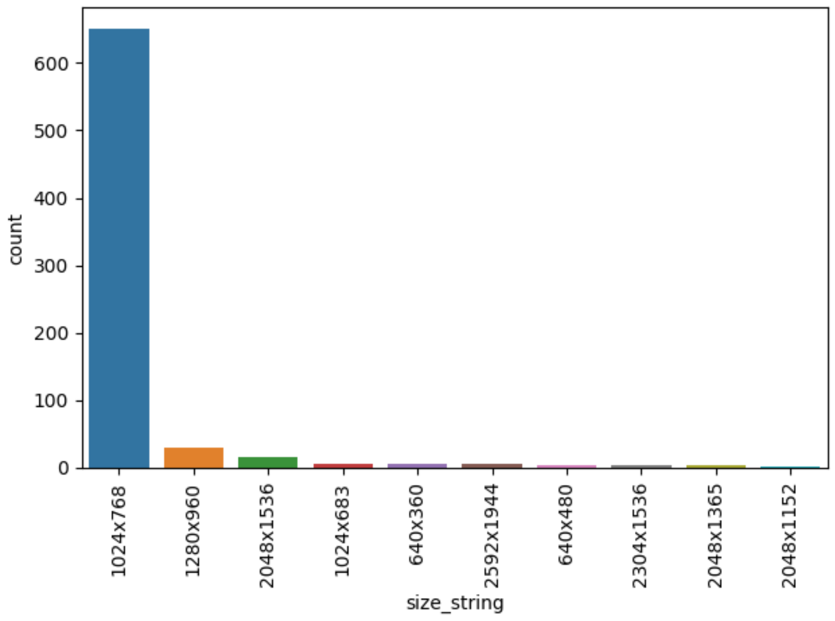


Figure 2.2: Top 10 most frequent image sizes represented in the Corsican Fire Database

2.1.3 Preprocessing

It was decided to preprocess the whole dataset by cropping samples of dimension 256×256 without overlap from the 1135 images, because some architectures encounter problems of layer shape compatibility when the input dimension is not a power of two. Crops that were smaller than 256×256 were simply discarded. Ground-truth masks and NIR images (when NIR was available) were cropped in parallel. This yielded 87672 samples that were indexed in a SQLite database with their corresponding ground-truth and NIR crop, to allow to browse them in a more efficient manner. The ER schema of the database used for indexing (though extremely straightforward) is represented in Figure 2.3. The `name` attribute refers to the file name, `nir` to the presence of NIR (1 or 0), `seq` refers to which video sequence the sample is part of, if any, `fire_pixels` is the count of fire pixels in the sample and `split` is the index of the

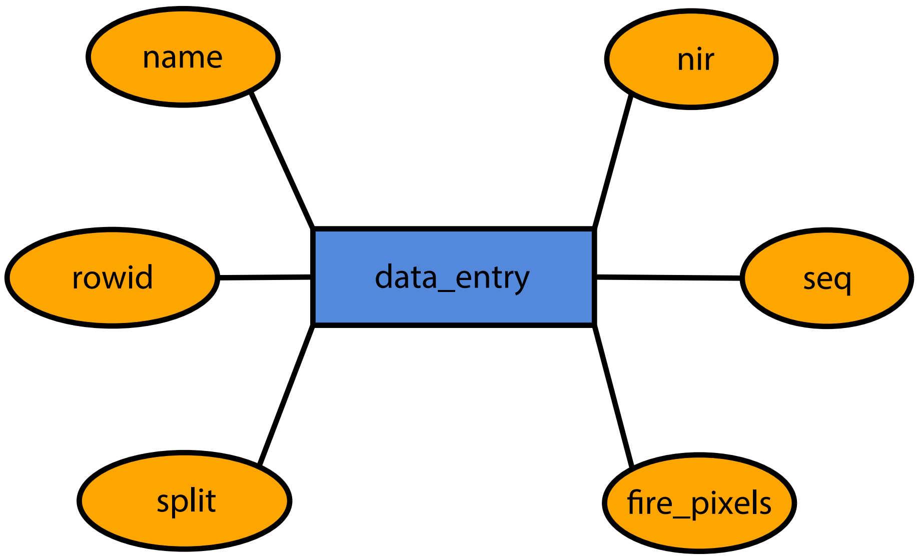


Figure 2.3: ER database schema used to index samples of the dataset. The only entity is represented in blue and its attributes in orange.

split to which the image belongs (this is detailed further in Section 2.2). Since this work focuses on segmentation and not on detection, the images without fire pixels can safely be removed from the dataset, even if it represents a good proportion of the images as represented in Figure 2.4. It also seemed reasonable to exclude samples with too small a number of pixels when the fire had been cropped near the borders. To choose this lower limit on the pixel count, samples were examined manually considering greater and greater value for this lower limit until an area of fire that seemed meaningful was encountered; all pictures with a number of fire pixels smaller than 20 were excluded from the dataset and 47520 samples remained.

2.2 Evaluation framework

To avoid overfitting and to allow to evaluate the model accurately, a standard train-validation-test split is generated (with proportions 0.7-0.15-0.15, respectively). To also allow the results of this research to be reproduced to some extent, this split is saved in

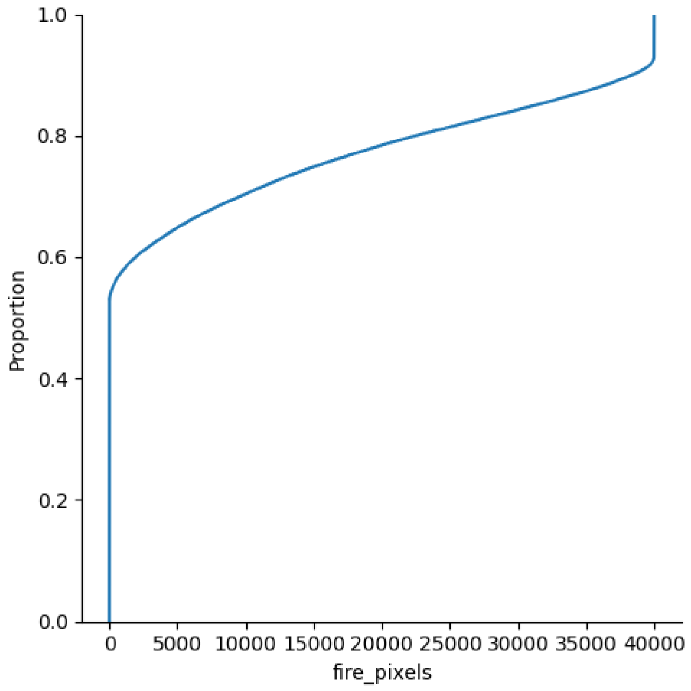


Figure 2.4: Empirical cumulative distribution of samples according to their count of flame pixels

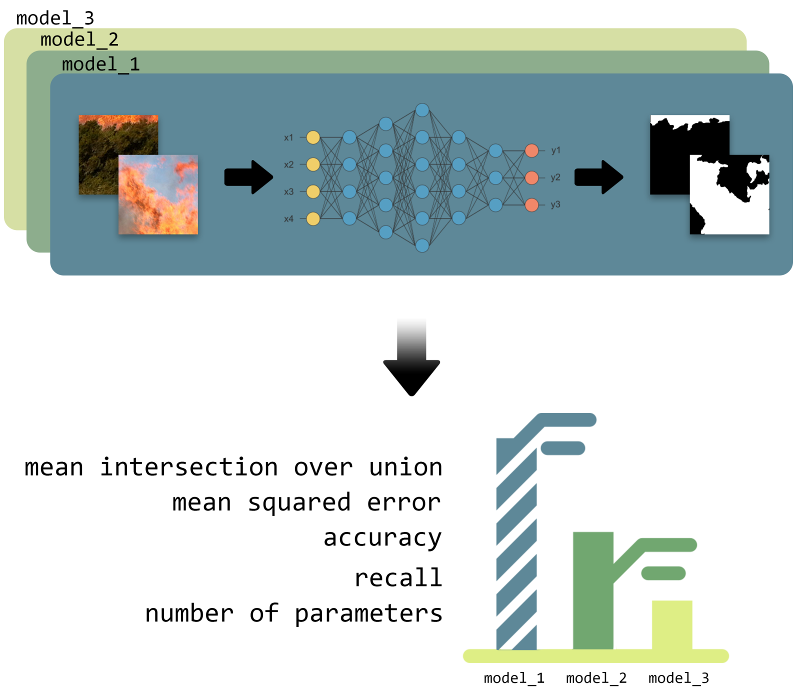


Figure 2.5: Evaluation methodology diagram

the SQLite database to allow the very same dataset and split to be reconstructed from it and is included in the repository. This is encoded as a simple integer: the `split` attribute is 0 when the sample is in the training set, 1 in the validation set and 2 in the testing set. All architectures implemented in this work were evaluated during training using the validation set to help choose the right hyperparameters (in this study, only the epoch count was tweaked), then retrained on the union of the training and validation sets before being evaluated for the first and last time on the test set. All models are then compared using the same sets of metrics. In this study, the IoU, the recall, the MSE and the accuracy were chosen. Recall was preferred over precision, since in the context of segmenting forest fires it seems more important to make sure we would favor missing less regions of fire over ensuring non-fire regions were not misclassified as fire. A diagram of the evaluation methodology is included in Figure 2.5.

Additionally, even if it was most of the time not included in the original implementations, shuffling of the training set between epochs was added for all architectures evaluated in this work, in order to make it less likely that the training algorithm gets entangled in a local minimum.

Finally, the implementation of the metrics chosen in this evaluation were the ones implemented in Keras. Only the IoU had to been partially adjusted to be used as it was needed to threshold the outputted predictions of the network that were all smaller than 1; more specifically a threshold of 0.5 was chosen (so that values below it are considered as non-fire and above it as fire).

2.3 Implementation

All architectures were implemented on Tensorflow 2.4.1 and were trained and tested on a NVIDIA GeForce RTX 3090. The full implementation is made public at

`https://github.com/PixelWeaver/ForestFireSegmentation`.

In this section, their implementation is briefly described and commented.

Before moving on to these specific implementations, let us first go through some generalities. All models were trained with an exponential decay learning schedule to avoid oscillations around minima during training. A decay rate of 0.96 was used. The chosen loss function was the binary cross-entropy, also called the log loss, which can be written as

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1-y_i)\cdot \log(1-p(y_i)) \tag{2.1}$$

if one sets down y_i as the label of the pixel (0 for non-fire, 1 for fire), and $p(y_i))$ as the probability that a pixel seen by the model is indeed fire or not. Minimizing the

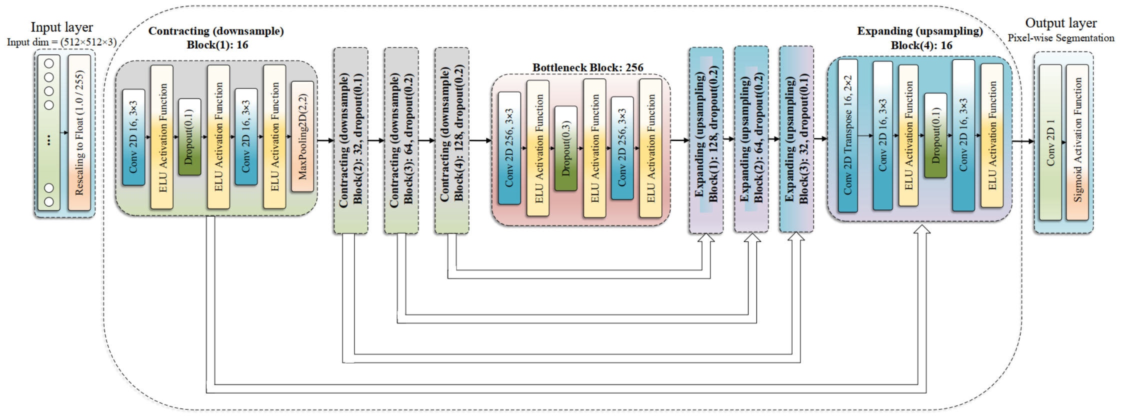


Figure 2.6: Customized version of the U-Net showcased in the FLAME paper.[88]

cross-entropy amounts to minimizing the dissimilarity between the y and $p(y)$ distributions so that our model (which distribution is the latter) better approximates the former. Other losses were not considered and this is indeed a limitation for this work. This is detailed further in the conclusion.

2.3.1 FLAME U-Net

The first architecture that was tested was the U-NET featured in the article associated with the FLAME dataset.[88] Along with the dataset is included suggested experiments and methodology that involve deep learning, including architectures for detection and segmentation. Their research and code are both made public. Since the scope of their research includes flame segmentation, their implementation was here tested on the Corsican Fire Database. Their exact architecture is represented in Figure 2.6. Only one thing needed to be changed in the architecture to make it compatible with our specific task: changing the input shape to 256×256.

The list of Tensorflow layers associated with their inputs and their number of parameters is included in Appendix E.

2.3.2 DeepLabV3+ w/ ResNet50 backbone

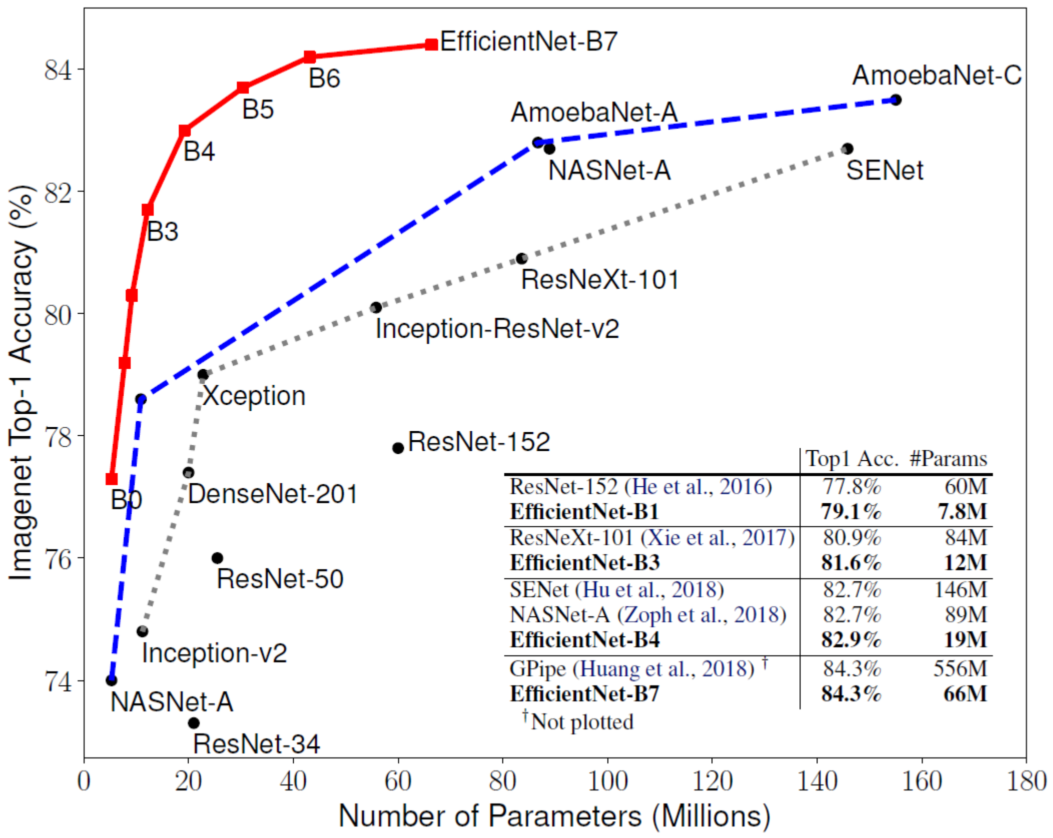
As mentioned in Section 1.3.5, Harkat et al. leveraged a DeepLabv3+ architecture with a ResNet50 backbone for flame segmentation on the dataset used in this study but neither their research, results or implementation was made public. The architecture was therefore reimplemeneted drawing inspiration from a public implementation of DeepLabv3+ found at

https://github.com/srihari-humbarwadi/DeepLabV3_Plus-Tensorflow2.0

though it was modified to have the correct input dimension, comply with the evaluation framework used in this work as well as to use the official Tensorflow ResNet50 implementation instead of a re-implementation. As a reminder, the DeepLabV3+ architecture is depicted in Section 1.3.5 and represented in Figure 1.19. The list of Tensorflow layers associated with their inputs and their number of parameters is included in Appendix G.

2.3.3 DeepLabV3+ w/ EfficientNet backbone

In 2019, M. Tan and Q. Le published their *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks* paper,[97] that defines a new scaling method that calibrates uniformly the depth, width and resolution dimensions of the network by the means of a mere compound coefficient. This compound coefficient is also expressed according to a user-defined coefficient that controls how many additional resources are available for model scaling. In addition to that scaling method, a new family of convolutional classification architectures called EfficientNets was also developped. The EfficientNet family's ImageNet accuracy is compared against the one of other famous architectures in Figure 2.7. They are more efficient in terms of parameters and floating point operations per second (FLOPS) (when benchmarked against ImageNet[22]) than previously studied ConvNets while also being more efficient (i.e. EfficientNets achieve a better accuracy than other convolution network with a similar number of parameters or FLOPS). This means that for instance, EfficientNet-B4 that has a better accuracy on ImageNet than ResNet50 — that was used as a feature extractor in the encoder part of DeepLabV3+ as mentioned previously — for a similar amount of parameters.



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

Figure 2.7: Model Size vs. ImageNet Accuracy. All numbers are for single-crop, single-model. EfficientNets significantly outperform other ConvNets. In particular, EfficientNet-B7 achieves new state-of-the-art 84.3% top-1 accuracy but being 8.4x smaller and 6.1x faster than GPipe. EfficientNet-B1 is 7.6x smaller and 5.7x faster than ResNet-152.[97]

This seems to hint that it would be likely that an EfficientNet-based DeepLabv3+ would also score better in terms of our metrics of interest. To verify so, the previous DeepLabv3+ implementation was modified to use official Tensorflow implementation of EfficientNet-B4 as a feature extractor instead of ResNet50. The detail of the Tensorflow layers and associated number of parameters can be found in Appendix H.

2.3.4 Squeeze U-Net

The Squeeze U-Net has not been used on the Corsican Fire Database or a similar dataset. But since it was chosen to implement the ATT Squeeze U-Net (which is mentioned in Section 2.3.5), it seemed preferable to also implement the Squeeze U-Net to see how their modified version of the Fire module, their modified version of the transposed Fire module and the added attention mechanisms would compare against the Squeeze U-Net implementation (described in detail in Sections 1.3.4 and 1.3.4, respectively). It was inspired by the public implementation found at https://github.com/lhelontra/squeeze-unet. Again, a few things needed to be adapted in order to work with the evaluation framework and the input size of images. The specifics of the Tensorflow layers as well as their respective number of parameters are detailed in Appendix F.

2.3.5 ATT Squeeze U-Net

As mentioned earlier in Section 1.3.4, the ATT Squeeze U-Net has been evaluated on the Corsican Fire Database in its original version but its implementation has not been made public. However, thanks to its accurate description in the paper, it was possible to reconstitute it. The list of the Tensorflow layers and associated number of parameters can be found in Appendix I.

3 Results

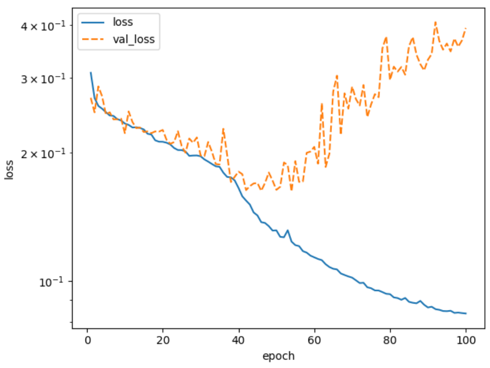
In this chapter, the results of the experiments conducted in this study are presented then briefly commented. For all trainings, a figure containing the evaluation of all metrics considered (i.e. accuracy, recall, IoU and MSE) as well as the loss (which is binary cross-entropy for all networks) is shown and used to motivate the choice of the considered hyper-parameters (here, only the number of epochs is considered) before a table with the final test values of these metrics (except for the loss which is only used for training) is reported. Like mentioned in Section 2.2, for all architectures, the test values are obtained after retraining on the whole union of the training and the testing set. All metrics were reported on a logarithmic scale to make smaller changes more noticeable.

3.1 FLAME U-Net

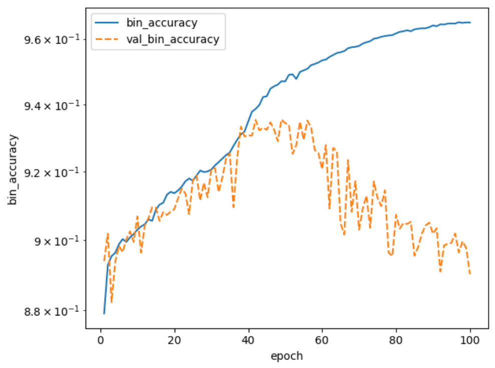
The FLAME U-Net was the first architecture tested. It was trained with a batch size of 16 and with the Adam optimizer like in the original FLAME U-Net implementation. During training, the model started to overfit very clearly after 41 epochs, which can be seen on every single metric we have at our disposition as shown in Figure 3.1. The metrics based on the training data continue to improve whereas the ones based on the validation set, an indication on the generalization ability of our model, start crashing (or soaring for the loss and the MSE) after 41 epochs. It was therefore decided to freeze training at 41 epochs.

The important variance of the validation-based metrics is somewhat troubling, even though the scale is relatively small.

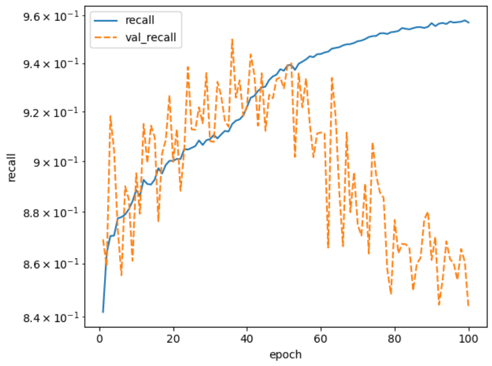
Results on the test sets are shown in Table 3.1. They are better than what was observed on the original FLAME dataset on which this model was first evaluated. For reference, they obtained a precision of 91.99%, a recall of 83.88% and an IoU of 78.17%. This could possibly be explained by the fact that flame segmentation on the Corsican Fire Database might be an easier task than on the FLAME dataset. Other factors that could explain this difference is that the authors used a different epoch count (30), a different input size (512×512) and did not rely on shuffling the training set between epochs. However, as it would already be difficult to compare results obtained on different splits of the same database, it goes without saying that it is utterly impossible to really compare results when using two distinct datasets.



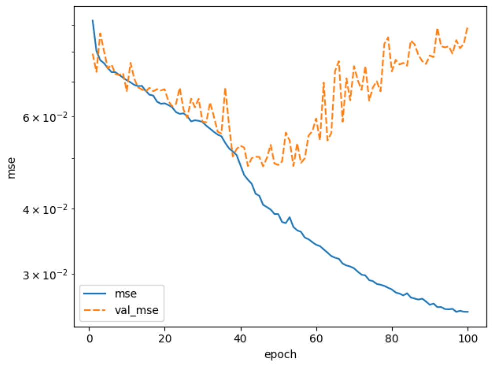
(a)



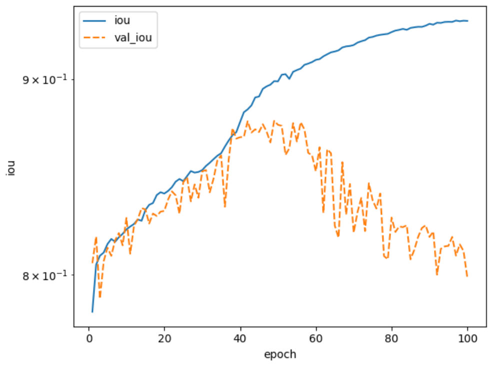
(b)



(c)



(d)



(e)

Figure 3.1: Training set and validation set -based metrics during training for FLAME U-Net: (a) loss (binary cross-entropy), (b) accuracy, (c) recall, (d) MSE, (e) IoU.

metric	score
recall	0.94
IoU	0.892
accuracy	0.943
MSE	0.042

Table 3.1: FLAME U-Net test results (41 epochs)

3.2 DeepLabv3+ with ResNet50 backbone

Second came the DeepLabv3+ architecture, first considered with a ResNet50 backbone, just like in the work of Harkat et al who also used the Corsican Fire Database.[40] Their research was however not in open access and it was impossible to compare their results with ours. A batch size of 16 was chosen, since our training set is relatively small, and the Adam optimizer was used once more. The ResNet50 was loaded with the ImageNet pretrained weights. The test and validation metrics are reported in Figure 3.2.

On the loss plot, it can clearly be seen when the network is starting to overfit (around 47 epochs). The plots of the other metrics seem to concur with the same conclusion, apart from the validation recall that seem to be decreasing after around 25 epochs. However, its variation is still somewhat small as it only goes down from around 0.98 before oscillating and then stabilizing around 0.96. The overfit is less visible on the accuracy and the IoU plot.

Results on the test set after freezing training at 47 epochs are reported in Table 3.2. They are better for every metric than the results of the FLAME U-Net architecture which can at least be explained by the huge difference in the number of parameters between these two: the FLAME U-Net has 2M parameters whereas the DeepLabv3+

with the ResNet50 backbone have 40M together as can be seen in Appendices E and G respectively.

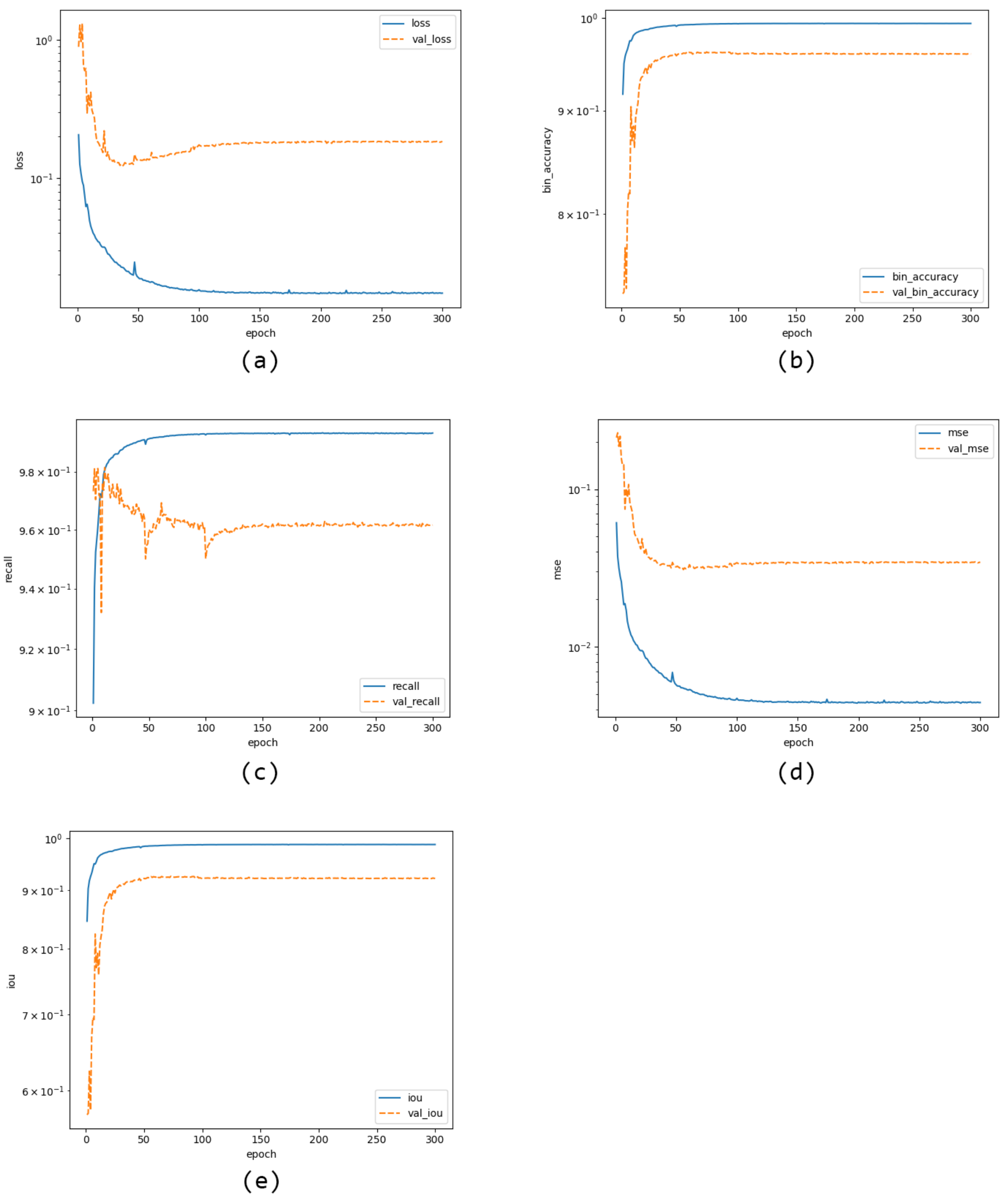


Figure 3.2: Training set and validation set -based metrics during training for DeepLabv3+ with a ResNet50 backbone: (a) loss (binary cross-entropy), (b) accuracy, (c) recall, (d) MSE, (e) IoU.

metric	score
recall	0.968
IoU	0.926
accuracy	0.962
MSE	0.031

Table 3.2: DeepLabV3+ w/ ResNet50 backbone test results (47 epochs)

3.3 DeepLabv3+ with EfficientNet backbone

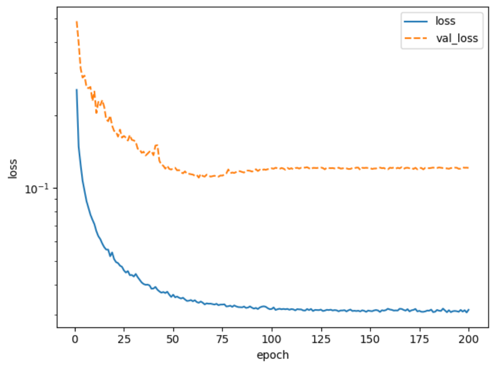
Because EfficientNet-B4 achieved better performances than ResNet50 on the ImageNet dataset with a similar or lower number of parameters, like shown in Figure 2.7, it felt important to investigate it as well on our task, since it seemed to herald increased performances. To do so, the previous architecture was modified with to replace the ResNet50 backbone with the EfficientNet-B4 backbone. The EfficientNet-B4 was already implemented in Tensorflow, therefore this implementation was rather straightforward. Akin to what has been done for the RestNet50, it was loaded with the ImageNet pretrained weights.

Two possibilities were considered to encode the image features using EfficientNet-B4 (cf. DeepLabv3+ encoder-decoder structure described in Section 1.3.5), after discarding the classification head: the output of the seventh convolutional block, resulting in a somewhat shallower architecture, henceforth referred to as `dlv3_efficientnet`; or taking the top activation right before the classification head of EfficientNet-B4, therefore resulting in a slightly deeper architecture, henceforth referred to as `dlv3_efficientnet_2`. Both were were implemented, trained on the training set and evaluated using the validation set. They are compared using the same metrics as previously in Figure 2.7. It can be seen that for all validation-based metrics considered, the shallower `dlv3_efficientnet` variant scored better (and converged faster towards better scores), which seems to indicate that features right at the output of the last convolutional block capture better the semantics of the images than the ones at the top activation right before the classification head of EfficientNet-B4.

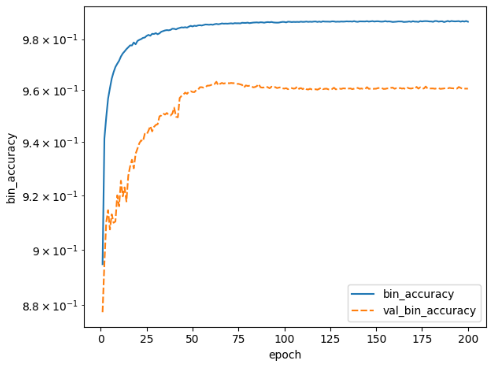
Therefore, it was chosen to stick with the `dlv3_efficientnet` variant. A batch size of 24 was chosen as well as the Adam optimizer again. For the sake of clarity in order to be able to choose as best as possible the epoch hyperparameter, the training plots of this variant are displayed on their own in Figure 3.3. The loss plot seems to indicate that the model begins to overfit after the 65th epoch. The history of the other validation metrics seem to concur once more with this conclusion. The training was therefore pegged at 65 epochs and the test results are displayed in Table 3.3. Results are all slightly better than those obtained with the ResNet50 backbone, except for the recall that is right below. What is more, the EfficientNet based DeepLabv3+ architecture has way less parameters than the ResNet flavor, it is brought down from a whopping 40M to 22M as can be seen in Appendix H, which constitutes a 55% decrease in parameters.

metric	score
recall	0.967
IoU	0.93
accuracy	0.964
MSE	0.028

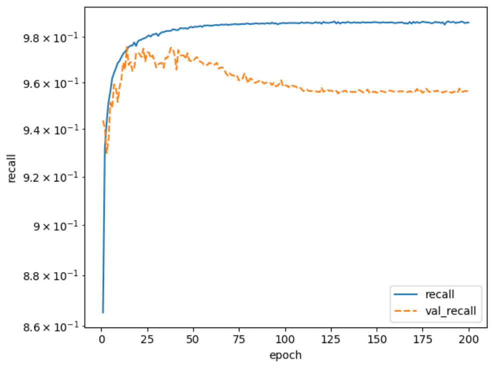
Table 3.3: DeepLabV3+ w/ EfficientNet backbone test results (65 epochs)



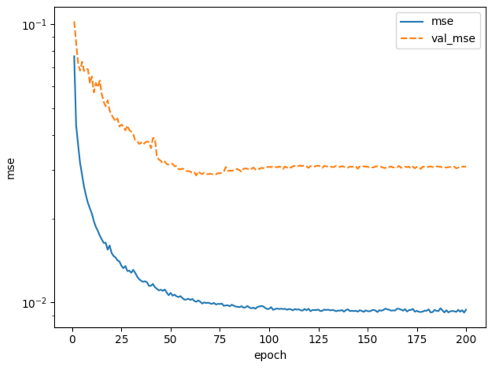
(a)



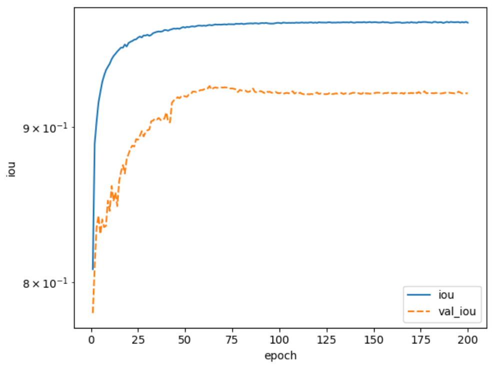
(b)



(c)

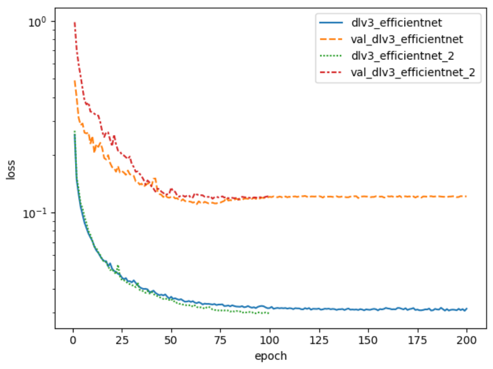


(d)

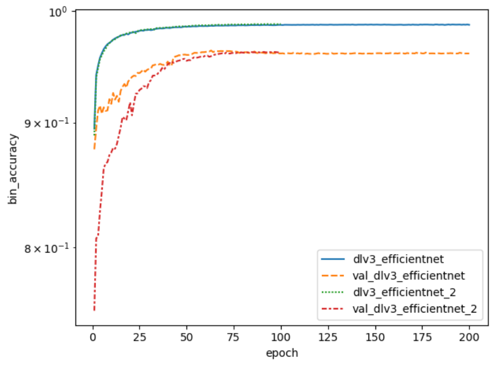


(e)

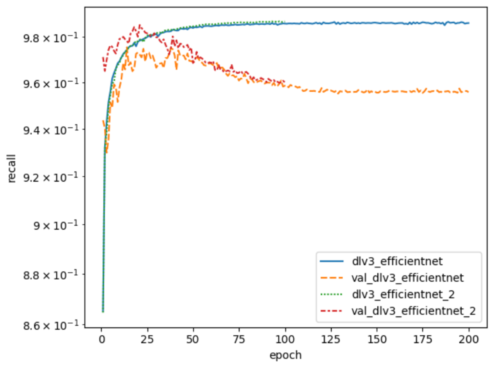
Figure 3.3: Training set and validation set -based metrics during training for DeepLabv3+ with an EfficientNet backbone (`dlv3_efficientnet` variant): (a) loss (binary cross-entropy), (b) accuracy, (c) recall, (d) MSE, (e) IoU.



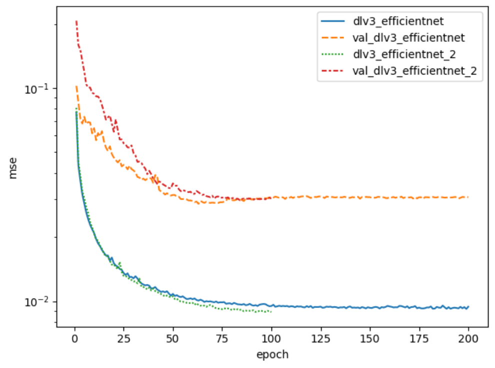
(a)



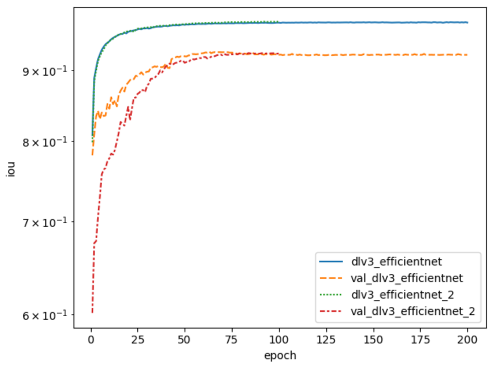
(b)



(c)



(d)



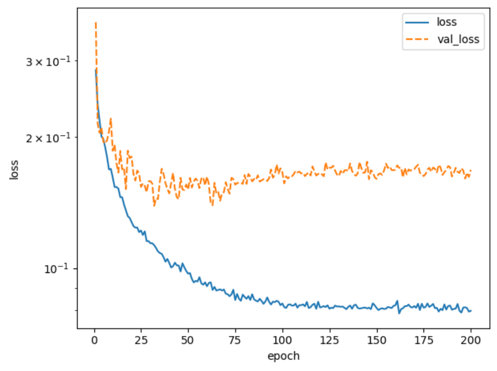
(e)

Figure 3.4: Comparison of train/val-based metrics during training between `dlv3_efficientnet` and `dlv3_efficientnet_2` described in Section 3.3: (a) loss (binary cross-entropy), (b) accuracy, (c) recall, (d) MSE, (e) IoU.

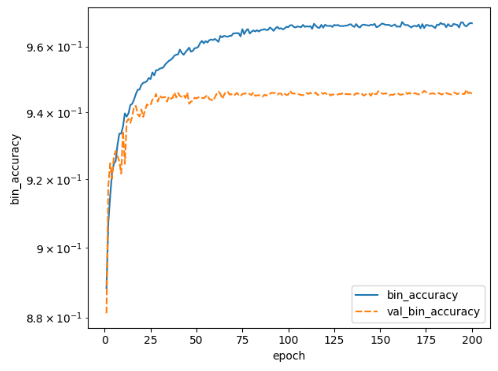
3.4 Squeeze U-Net

Next, the Squeeze U-Net was evaluated in order to compare it with the ATT Squeeze U-Net and assess the added value of its modified Fire and transposed Fire module as well as the attention mechanisms. To be able to compare this with the results obtained in the original ATT Squeeze U-Net paper,[117] a batch size of 12 was chosen (like theirs). The Adam optimizer was once more used. The training and validation set-based metrics are plotted in Figure 3.5. Overfitting can once more clearly be seen on the loss plot but its exact start is harder to pinpoint. Using the other metrics, however, it becomes clearer that validation performances do not increase any further after approximately 50 epochs.

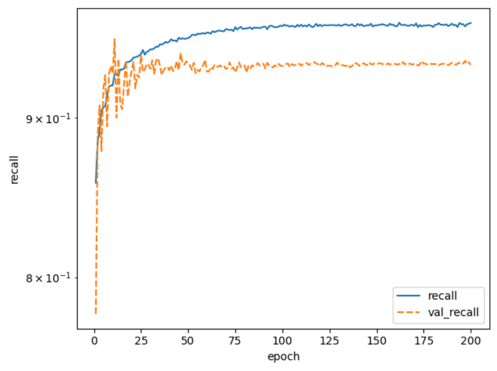
The results on the testing set after pegging training at 50 epochs are reported in Table 3.4. It goes without saying performances are lesser than the ones of both of the DeepLabv3+ based architectures. The complexity and number of parameters is not even comparable. It makes more sense to compare it with the FLAME U-Net: their results are really similar to one another. The recall is slightly lower than for the FLAME U-Net but the IoU, accuracy and MSE are somewhat better, in a small measure. The difference in parameters between the Squeeze U-Net and the FLAME U-Net is very reasonable with the Squeeze U-Net having 500K additional parameters, a 125% increase compared to the FLAME U-Net. This 125% increase in parameters is still questionable if it does not come with noticeably increased performances. However, the Squeeze U-Net is very noticeably less prone to overfit than the FLAME U-Net.



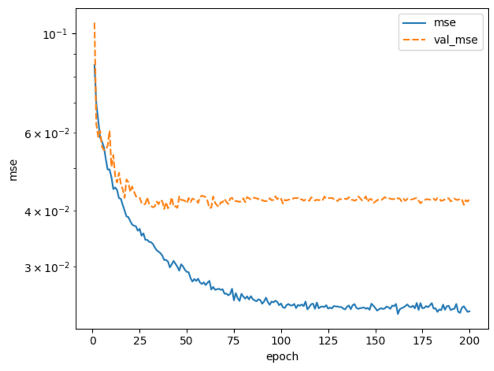
(a)



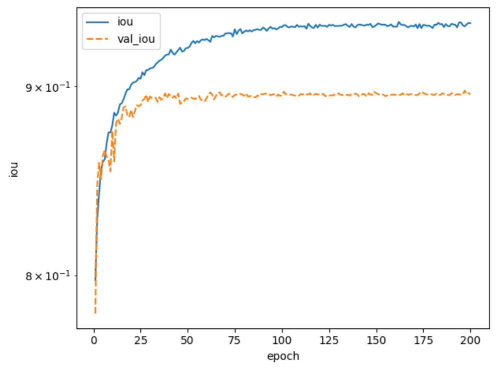
(b)



(c)



(d)



(e)

Figure 3.5: Training set and validation set -based metrics during training for Squeeze U-Net: (a) loss (binary cross-entropy), (b) accuracy, (c) recall, (d) MSE, (e) IoU.

metric	score
recall	0.93
IoU	0.896
accuracy	0.946
MSE	0.04

Table 3.4: Squeeze U-Net test results (50 epochs)

3.5 ATT Squeeze U-Net

Finally, the ATT Squeeze U-Net architecture is evaluated. Like mentioned earlier, this implementation had to be replicated from scratch because it had not been disclosed by the authors.[117] Exactly as in the original paper, a batch size of 12 and the Adam optimizer were used. The train/val set-based metrics during training are plotted in Figure 3.6. What is blingly obvious when looking at those trends compared to previously considered architectures is that the validation and training metrics are extremely close all along, especially for the accuracy, the recall and the IoU. The model does not seem to overfit greatly, similarly to what was observed for the Squeeze U-Net. However, the model performances do not increase anymore past 50 epochs, which is especially noticeable on the loss and MSE plots.

After freezing the training at 50 epochs, the model was evaluated on the testing sets. Its performances are illustrated in Table 3.5. In this implementation, a number of 885K parameters was leveraged, which is greatly lower than all the other architectures investigated. The ATT Squeeze U-Net scored very similarly to the Squeeze U-Net with values that are right below (yet very close). Also, it scored better than the FLAME U-Net for all metrics but recall, with a decrease of 226% in the number of parameters.

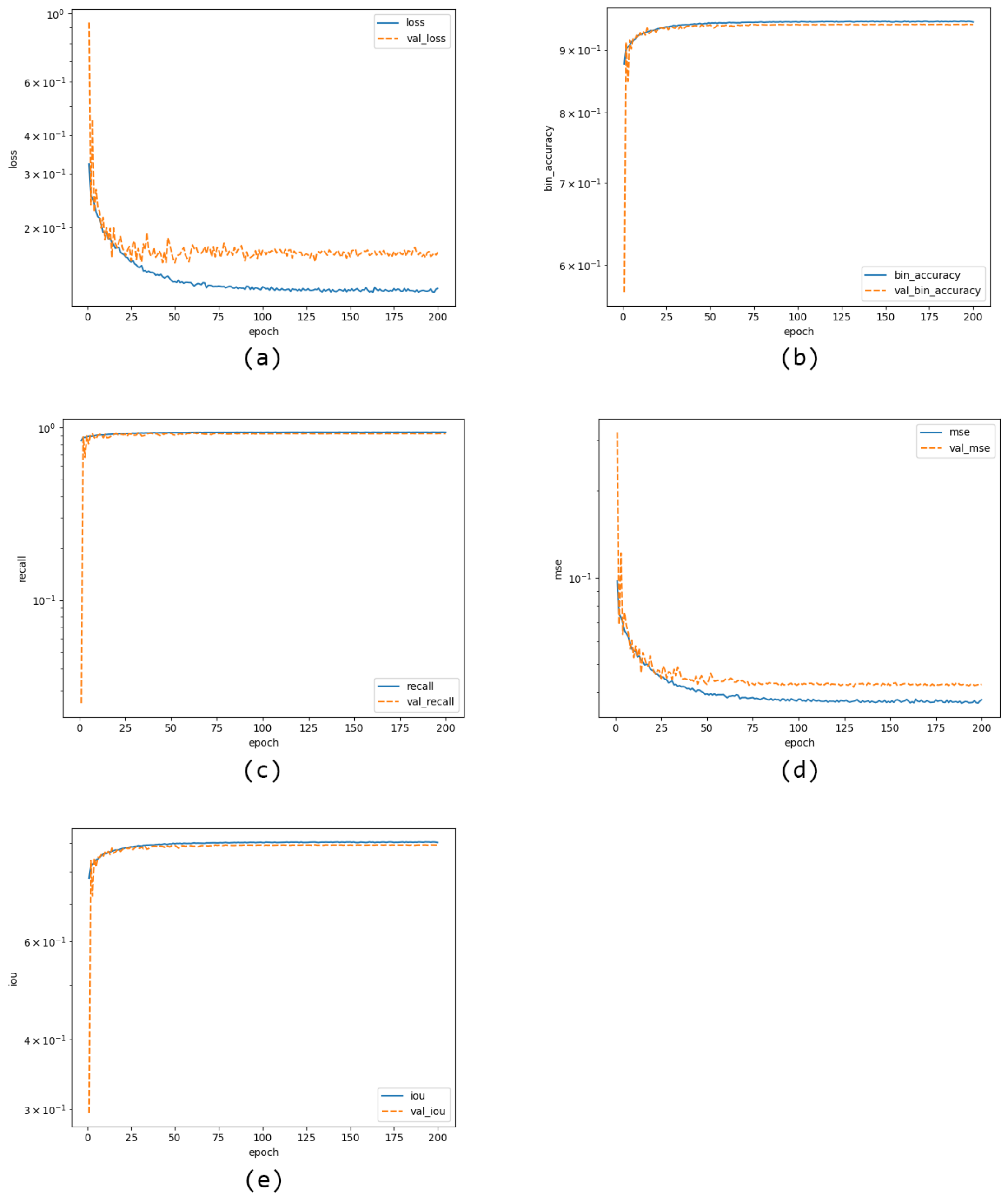


Figure 3.6: Training set and validation set -based metrics during training for ATT Squeeze U-Net: (a) loss (binary cross-entropy), (b) accuracy, (c) recall, (d) MSE, (e) IoU.

In their original ATT Squeeze U-Net paper, the authors were able to score an accuracy of 0.907 and a DSC of 0.875 (i.e. an IoU of 0.778 using Equation 1.14) using 7.96M parameters, on the same dataset (but not the same split). Those scores are significantly lower than the ones obtained in this study, whereas it uses 899% of the parameters used in our case. This is quite peculiar indeed and the fact that our implementation leveraged training set shuffling between epochs while theirs make no mention of it does not seem enough to explain this gap. Two hypothesis remain. On the one hand, they used the peculiar input size of 224×224 though it is unsure how that would make their performance lower. On the other hand, they did not disclose the number of epochs they have used to evaluate their model, nor how they had chosen that number; they could have used a lower number of epochs, leading to not finding an actual minimum in the loss function, or they could have used a higher number of epochs leading to overfitting. There is no mention of that in their work, however, so these two hypotheses will remain unconfirmed.

metric	score
recall	0.928
IoU	0.893
accuracy	0.944
MSE	0.042

Table 3.5: ATT Squeeze U-Net test results (50 epochs)

3.6 Perspectives

The results of all experiments conducted in this work are reported in Table 3.6. It can be seen that ATT Squeeze U-Net has a significantly lower number of parameters than all other architectures while achieving very decent performance, especially compared to the FLAME U-Net (compared to which it performs better except for the recall) and the Squeeze U-Net. The DeepLabv3+ architectures have the best results overall but at the cost of a huge amount of additional parameters. The EfficientNet-B4 flavor scored the most while also using a lot less parameters than the ResNet50 flavor, which was expected from the results of the work of M. Tan and Q. Le.[97]

architecture	recall	IoU	accuracy	MSE	# parameters
FLAME U-Net	0.94	0.892	0.943	0.043	2M
DLV3+ w/ ResNet50	**0.968**	0.926	0.962	0.031	**40M**
DLV3+ w/ EfficientNetB4	0.967	**0.93**	**0.964**	**0.028**	22M
Squeeze U-Net	0.930	0.897	0.946	0.042	2.5M
ATT Squeeze U-Net	0.928	0.893	0.944	0.042	**885K**

Table 3.6: Metric comparison table

In order to be able to compare better what increase in performance is achieved with what overhead in the number of parameters, the results of Table 3.6 have been normalized by the ones of the ATT Squeeze U-Net since it was the one with the lowest number of parameters. These normalized values are presented in Table 3.7. In the simple task of fire segmentation, it does not seem that it is worth considering architectures that are more complex and come with a higher price to pay in training time and memory usage as well as in hardware. To better assess the competitiveness of more advanced networks such as the DeepLabv3+ architectures, then a more difficult task would be required.

architecture	normalized recall	normalized IoU	normalized accuracy	normalized MSE	normalized # parameters
FLAME U-Net	1.013	0.999	0.943	1.023	2.3
DLV3+ w/ ResNet50	1.043	1.037	1.019	0.74	45.2
DLV3+ w/ EfficientNetB4	1.042	1.041	1.02	0.67	24.8
Squeeze U-Net	1.002	1.004	1.002	1	2.8

Table 3.7: Metric comparison table normalized by the scores of the ATT Squeeze U-Net

Eventually, in Figure 3.7, the outputs of the different networks are compared for a set of representative inputs chosen from the dataset, which comprises both shots taken from the ground and from a higher altitude, along with these inputs and their ground-truth mask. On these, it is much more obvious than on Table 3.6 that the expressive power of the DeepLabv3+ architectures is more important. They are able to more accurately predict the shapes and their contour. In particular, smaller regions of fire also seem to be more accurately segmented but this should be the object of a separate experiment to be able to quantify this phenomenon more appropriately. The ATT Squeeze U-Net also seem to have the smoothest predicted fire area contours overall but this should also be more properly measured using a metric such as the fractal dimension, for instance.[32,84,105] It is also unsure that having smoother prediction contours is a desirable property for a fire segmentation model used by a fire department.

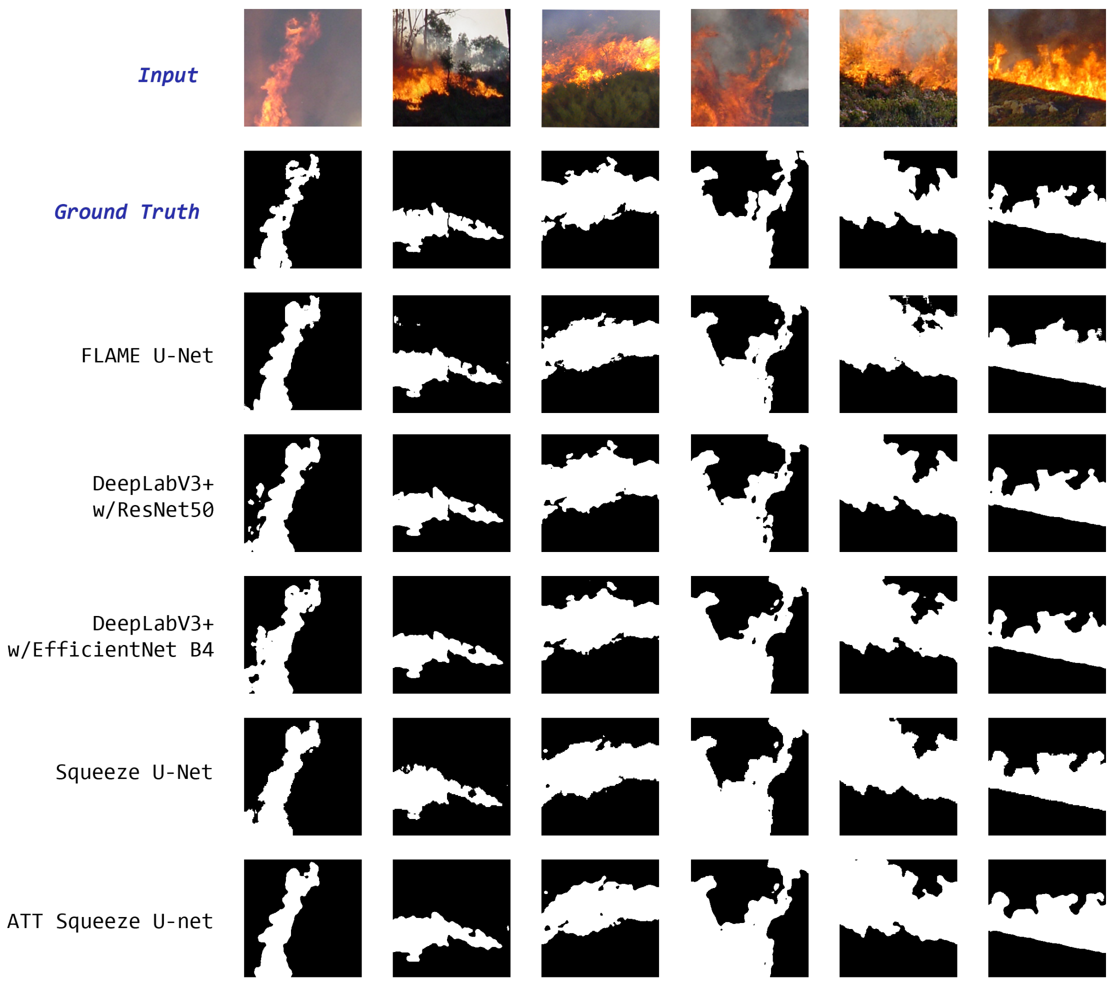


Figure 3.7: Comparison of outputs between the different implemented network, along with the input image and associated ground truth

4 Conclusion

In this work, the problem of flame segmentation in a natural environment is considered, more specifically, on the Corsican Fire Database which is an evolving dataset of fires in a natural environment from around the world.[99] Despite the fact that they were not numerous, it was still possible to find some architectures that had either been evaluated on the Corsican Fire DB or on similar datasets. A comparison between different candidate datasets was established before motivating why the Corsican Fire DB was preferred in this study, and several architectures were implemented, either inspired by existing from implementation or from scratch, chosen among the reviewed architectures. These architectures are: the FLAME U-Net, the DeepLabv3+ architecture with the EfficientNet-B4 and the ResNet-50 backbones, the Squeeze U-Net as well as the ATT Squeeze U-Net. The algorithms were then evaluated using a proper evaluation framework and compared using four commonly used metrics that seemed appropriate for the scope of this work namingly recall, accuracy, IoU and MSE. Notable among the evaluated architectures, the DeepLabV3+ with an EfficientNet backbone was the one that achieved the best results and the ATT Squeeze U-Net scored very decently with the least amount of network parameters. What transpired clearly from the results is that considering deeper and more complex architecture coming inherently coming with a higher number of parameters is not always justified, especially for a task that is not too intricate, such as the one considered in this manuscript.

The contributions of this work are in order:

- the comparison between multiple datasets that can be leveraged for segmentation of fires occurring in a natural environment,
- laying the groundwork for an evaluation framework of segmentation methods on the Corsican Fire Database,
- the evaluation of the performances of several architectures on the Corsican Fire Database,
- sharing all its implementations publicly.

Naturally, this work is very far from being exhaustive and one could picture multiple ways in which it could be extended. Some ideas of extensions follow. *(a)* An ablation study was not conducted on the considered architectures. *(b)* More architectures could be implemented and evaluated, including architectures that were not tested on a similar dataset in the past. *(c)* Different metrics for detection using the results of segmentation could be implemented and compared. *(d)* Other loss functions than the binary cross-entropy could be compared. *(e)* More datasets that could be useful in the context of early detection and segmentation of forest fires could be reviewed. *(f)* More hyperparameters than only the epoch count could be considered. *(g)* The metric comparison between architectures could be nuanced not only using the number of parameters but also by weighting other important performance considerations such as inference time for instance. *(h)* It would be interesting to compare how more relevant and more efficient the use of IR data might be instead of RGB in this type of application where visibility can be obstructed by several conditions such as plumes of smoke.

Appendix A. AHI bands

Band	Central Wavelength (μm)	Spatial resolution (km)
1 (B)	0.47063	1
2 (G)	0.51000	
3 (R)	0.63914	0.5
4	0.85670	1
5	1.6101	2
6	2.2568	
7	3.8853	
8	6.2429	
9	6.9410	
10	7.3467	
11	8.5926	
12	9.6372	
13	10.4073	
14	11.2395	
15	12.3806	
16	13.2807	

Table A.1: AHI bands. Natural-color component bands are noted with R, G and B[53]

Appendix B. MODIS bands

Band	Bandwidth	Spectral Radiance	Spatial Resolution	Primary Use
1	620 - 670 nm	21.8	250 m	Land/Cloud/Aerosols Boundaries
2	841 - 876 nm	24.7		
3	459 - 479 nm	35.3	500 m	Land/Cloud/Aerosols Properties
4	545 - 565 nm	29.0		
5	1230 - 1250 nm	5.4		
6	1628 - 1652 nm	7.3		
7	2105 - 2155 nm	1.0		
8	405 - 420 nm	44.9	1000 m	Ocean Color/Phytoplankton/ Biogeochemistry
9	438 - 448 nm	41.9		
10	483 - 493 nm	32.1		
11	526 - 536 nm	27.9		
12	546 - 556 nm	21.0		
13	662 - 672 nm	9.5		
14	673 - 683 nm	8.7		
15	743 - 753 nm	10.2		
16	862 - 877 nm	6.2		
17	890 - 920 nm	10.0		Atmospheric Water Vapor
18	931 - 941 nm	3.6		
19	915 - 965 nm	15.0		
20	3.660 - 3.840 µm	0.45		Surface/Cloud Temperature
21	3.929 - 3.989 µm	2.38		
22	3.929 - 3.989 µm	0.67		
23	4.020 - 4.080 µm	0.79		
24	4.433 - 4.498 µm	0.17		Atmospheric Temperature
25	4.482 - 4.549 µm	0.59		
26	1.360 - 1.390 µm	6.00		Cirrus Clouds Water Vapor
27	6.535 - 6.895 µm	1.16		
28	7.175 - 7.475 µm	2.18		
29	8.400 - 8.700 µm	9.58		Cloud Properties
30	9.580 - 9.880 µm	3.69		Ozone
31	10.780 - 11.280 µm	9.55		Surface/Cloud Temperature
32	11.770 - 12.270 µm	8.94		
33	13.185 - 13.485 µm	4.52		Cloud Top Altitude
34	13.485 - 13.785 µm	3.76		
35	13.785 - 14.085 µm	3.11		
36	14.085 - 14.385 µm	2.08		

Table B.1: MODIS bands. Spectral Radiance expressed in $Wm^{-2}sr^{-1}\mu m^{-1}$.[72]

Appendix C. VIIRS bands

Band	Wavelength Range (μm)	Band Explanation	Spatial Resolution (m)
M1	0.402 - 0.422	Visible/Reflective	750
M2	0.436 - 0.454		
M3	0.478 - 0.488		
M4	0.545 - 0.565		
M5(B)	0.662 - 0.682		
M6	0.739 - 0.754	Near IR	
M7 (G)	0.846 - 0.885		
M8	1.23 - 1.25	Shortwave IR	
M9	1.371 - 1.386		
M10 (R)	1.58 - 1.64		
M11	2.23 - 2.28		
M12	3.61 - 3.79	Medium-wave IR	
M13	3.97 - 4.13		
M14	8.4 - 8.7	Longwave IR	
M15	10.26 - 11.26		
M16	11.54 - 12.49		
DNB	0.5 - 0.9	Visible/Reflective	750 across full scan
I1 (B)	0.6 - 0.68	Visible/Reflective	375
I2 (G)	0.85 - 0.88	Near IR	
I3 (R)	1.58 - 1.64	Shortwave IR	
I4	3.55 - 3.93	Medium-wave IR	
I5	10.5 - 12.4	Longwave IR	

Table C.1: VIIRS bands. M stands for moderate resolution bands, I for imagery, DNB for Day-Night Band (or Near Constant Contrast (NCC) band) and natural-color component bands are noted with R, G and B.[20]

Appendix D. AVHRR bands

Band	Central Wavelength (μm)	Primary Uses
1	0.63	Visible cloud and surface features
2	0.86	Visible aerosols over water, vegetation
3	3.74	Infrared low-level cloud/fog, fire detection
4	10.8	Infrared surface/cloud-top temperature
5	12.0	Infrared surface/cloud temperature, low-level water vapor
6	1.61	Near-infrared surface, cloud phase

Table D.1: AVHRR/3 bands.[102]

Appendix E. FLAME U-Net layers and parameters

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 256, 256, 3) 0
__________________________________________________________________________________________________
lambda (Lambda)                 (None, 256, 256, 3)  0           input_1[0][0]
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 256, 256, 16) 448         lambda[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 256, 256, 16) 0           conv2d[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 256, 256, 16) 2320        dropout[0][0]
__________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 128, 128, 16) 0           conv2d_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 128, 128, 32) 4640        max_pooling2d[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 128, 128, 32) 0           conv2d_2[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 128, 128, 32) 9248        dropout_1[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 64, 64, 32)   0           conv2d_3[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 64, 64, 64)   18496       max_pooling2d_1[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 64, 64, 64)   0           conv2d_4[0][0]
__________________________________________________________________________________________________
conv2d_5 (Conv2D)               (None, 64, 64, 64)   36928       dropout_2[0][0]
__________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, 32, 32, 64)   0           conv2d_5[0][0]
__________________________________________________________________________________________________
conv2d_6 (Conv2D)               (None, 32, 32, 128)  73856       max_pooling2d_2[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 32, 32, 128)  0           conv2d_6[0][0]
__________________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 32, 32, 128)  147584      dropout_3[0][0]
__________________________________________________________________________________________________
```

```
max_pooling2d_3 (MaxPooling2D)  (None, 16, 16, 128)  0           conv2d_7[0][0]

__________________________________________________________________________________________________
conv2d_8 (Conv2D)               (None, 16, 16, 256)  295168      max_pooling2d_3[0][0]

__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 16, 16, 256)  0           conv2d_8[0][0]

__________________________________________________________________________________________________
conv2d_9 (Conv2D)               (None, 16, 16, 256)  590080      dropout_4[0][0]

__________________________________________________________________________________________________
conv2d_transpose (Conv2DTranspo (None, 32, 32, 128)  131200      conv2d_9[0][0]

__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 32, 32, 256)  0           conv2d_transpose[0][0]

                                                                 conv2d_7[0][0]

__________________________________________________________________________________________________
conv2d_10 (Conv2D)              (None, 32, 32, 128)  295040      concatenate[0][0]

__________________________________________________________________________________________________
dropout_5 (Dropout)             (None, 32, 32, 128)  0           conv2d_10[0][0]

__________________________________________________________________________________________________
conv2d_11 (Conv2D)              (None, 32, 32, 128)  147584      dropout_5[0][0]

__________________________________________________________________________________________________
conv2d_transpose_1 (Conv2DTrans (None, 64, 64, 64)   32832       conv2d_11[0][0]

__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 64, 64, 128)  0           conv2d_transpose_1[0][0]

                                                                 conv2d_5[0][0]

__________________________________________________________________________________________________
conv2d_12 (Conv2D)              (None, 64, 64, 64)   73792       concatenate_1[0][0]

__________________________________________________________________________________________________
dropout_6 (Dropout)             (None, 64, 64, 64)   0           conv2d_12[0][0]

__________________________________________________________________________________________________
conv2d_13 (Conv2D)              (None, 64, 64, 64)   36928       dropout_6[0][0]

__________________________________________________________________________________________________
conv2d_transpose_2 (Conv2DTrans (None, 128, 128, 32) 8224        conv2d_13[0][0]

__________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 128, 128, 64) 0           conv2d_transpose_2[0][0]

                                                                 conv2d_3[0][0]

__________________________________________________________________________________________________
conv2d_14 (Conv2D)              (None, 128, 128, 32) 18464       concatenate_2[0][0]

__________________________________________________________________________________________________
```

```
dropout_7 (Dropout)             (None, 128, 128, 32) 0           conv2d_14[0][0]

__________________________________________________________________________________________________
conv2d_15 (Conv2D)              (None, 128, 128, 32) 9248        dropout_7[0][0]

__________________________________________________________________________________________________
conv2d_transpose_3 (Conv2DTrans (None, 256, 256, 16) 2064        conv2d_15[0][0]

__________________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 256, 256, 32) 0           conv2d_transpose_3[0][0]

                                                                 conv2d_1[0][0]

__________________________________________________________________________________________________
conv2d_16 (Conv2D)              (None, 256, 256, 16) 4624        concatenate_3[0][0]

__________________________________________________________________________________________________
dropout_8 (Dropout)             (None, 256, 256, 16) 0           conv2d_16[0][0]

__________________________________________________________________________________________________
conv2d_17 (Conv2D)              (None, 256, 256, 16) 2320        dropout_8[0][0]

__________________________________________________________________________________________________
conv2d_18 (Conv2D)              (None, 256, 256, 1)  17          conv2d_17[0][0]
==================================================================================================
Total params: 1,941,105

Trainable params: 1,941,105

Non-trainable params: 0
```

Appendix F. Squeeze U-Net layers and parameters

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 256, 256, 3) 0

__________________________________________________________________________________________________
lambda (Lambda)                 (None, 256, 256, 3)  0           input_1[0][0]

__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 128, 128, 64) 1792        lambda[0][0]

__________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 64, 64, 64)   0           conv2d[0][0]

__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 64, 64, 16)   1040        max_pooling2d[0][0]

__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 64, 64, 16)   64          conv2d_1[0][0]

__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 64, 64, 64)   1088        batch_normalization[0][0]

__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 64, 64, 64)   9280        batch_normalization[0][0]

__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 64, 64, 128)  0           conv2d_2[0][0]
                                                                 conv2d_3[0][0]

__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 64, 64, 16)   2064        concatenate[0][0]

__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 64, 64, 16)   64          conv2d_4[0][0]

__________________________________________________________________________________________________
conv2d_5 (Conv2D)               (None, 64, 64, 64)   1088        batch_normalization_1[0][0]

__________________________________________________________________________________________________
conv2d_6 (Conv2D)               (None, 64, 64, 64)   9280        batch_normalization_1[0][0]

__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 64, 64, 128)  0           conv2d_5[0][0]
                                                                 conv2d_6[0][0]

__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 32, 32, 128)  0           concatenate_1[0][0]

__________________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 32, 32, 32)   4128        max_pooling2d_1[0][0]

__________________________________________________________________________________________________
```

```
batch_normalization_2 (BatchNor (None, 32, 32, 32)   128         conv2d_7[0][0]

____________________________________________________________________________________________________
conv2d_8 (Conv2D)               (None, 32, 32, 128)  4224        batch_normalization_2[0][0]

____________________________________________________________________________________________________
conv2d_9 (Conv2D)               (None, 32, 32, 128)  36992       batch_normalization_2[0][0]

____________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 32, 32, 256)  0           conv2d_8[0][0]

                                                                 conv2d_9[0][0]

____________________________________________________________________________________________________
conv2d_10 (Conv2D)              (None, 32, 32, 32)   8224        concatenate_2[0][0]

____________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 32, 32, 32)   128         conv2d_10[0][0]

____________________________________________________________________________________________________
conv2d_11 (Conv2D)              (None, 32, 32, 128)  4224        batch_normalization_3[0][0]

____________________________________________________________________________________________________
conv2d_12 (Conv2D)              (None, 32, 32, 128)  36992       batch_normalization_3[0][0]

____________________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 32, 32, 256)  0           conv2d_11[0][0]

                                                                 conv2d_12[0][0]

____________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, 16, 16, 256)  0           concatenate_3[0][0]

____________________________________________________________________________________________________
conv2d_13 (Conv2D)              (None, 16, 16, 48)   12336       max_pooling2d_2[0][0]

____________________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, 16, 16, 48)   192         conv2d_13[0][0]

____________________________________________________________________________________________________
conv2d_14 (Conv2D)              (None, 16, 16, 192)  9408        batch_normalization_4[0][0]

____________________________________________________________________________________________________
conv2d_15 (Conv2D)              (None, 16, 16, 192)  83136       batch_normalization_4[0][0]

____________________________________________________________________________________________________
concatenate_4 (Concatenate)     (None, 16, 16, 384)  0           conv2d_14[0][0]

                                                                 conv2d_15[0][0]

____________________________________________________________________________________________________
conv2d_16 (Conv2D)              (None, 16, 16, 48)   18480       concatenate_4[0][0]

____________________________________________________________________________________________________
batch_normalization_5 (BatchNor (None, 16, 16, 48)   192         conv2d_16[0][0]

____________________________________________________________________________________________________
```

```
conv2d_17 (Conv2D)              (None, 16, 16, 192)  9408        batch_normalization_5[0][0]

__________________________________________________________________________________________________
conv2d_18 (Conv2D)              (None, 16, 16, 192)  83136       batch_normalization_5[0][0]

__________________________________________________________________________________________________
concatenate_5 (Concatenate)     (None, 16, 16, 384)  0           conv2d_17[0][0]

                                                                 conv2d_18[0][0]

__________________________________________________________________________________________________
conv2d_19 (Conv2D)              (None, 16, 16, 64)   24640       concatenate_5[0][0]

__________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 16, 16, 64)   256         conv2d_19[0][0]

__________________________________________________________________________________________________
conv2d_20 (Conv2D)              (None, 16, 16, 256)  16640       batch_normalization_6[0][0]

__________________________________________________________________________________________________
conv2d_21 (Conv2D)              (None, 16, 16, 256)  147712      batch_normalization_6[0][0]

__________________________________________________________________________________________________
concatenate_6 (Concatenate)     (None, 16, 16, 512)  0           conv2d_20[0][0]

                                                                 conv2d_21[0][0]

__________________________________________________________________________________________________
conv2d_22 (Conv2D)              (None, 16, 16, 64)   32832       concatenate_6[0][0]

__________________________________________________________________________________________________
batch_normalization_7 (BatchNor (None, 16, 16, 64)   256         conv2d_22[0][0]

__________________________________________________________________________________________________
conv2d_23 (Conv2D)              (None, 16, 16, 256)  16640       batch_normalization_7[0][0]

__________________________________________________________________________________________________
conv2d_24 (Conv2D)              (None, 16, 16, 256)  147712      batch_normalization_7[0][0]

__________________________________________________________________________________________________
concatenate_7 (Concatenate)     (None, 16, 16, 512)  0           conv2d_23[0][0]

                                                                 conv2d_24[0][0]

__________________________________________________________________________________________________
dropout (Dropout)               (None, 16, 16, 512)  0           concatenate_7[0][0]

__________________________________________________________________________________________________
conv2d_transpose (Conv2DTranspo (None, 16, 16, 192)  884928      dropout[0][0]

__________________________________________________________________________________________________
concatenate_8 (Concatenate)     (None, 16, 16, 576)  0           conv2d_transpose[0][0]

                                                                 concatenate_5[0][0]

__________________________________________________________________________________________________
conv2d_25 (Conv2D)              (None, 16, 16, 48)   27696       concatenate_8[0][0]
```

```
__________________________________________________________________________________________________
batch_normalization_8 (BatchNor (None, 16, 16, 48)   192         conv2d_25[0][0]
__________________________________________________________________________________________________
conv2d_26 (Conv2D)              (None, 16, 16, 192)  9408        batch_normalization_8[0][0]
__________________________________________________________________________________________________
conv2d_27 (Conv2D)              (None, 16, 16, 192)  83136       batch_normalization_8[0][0]
__________________________________________________________________________________________________
concatenate_9 (Concatenate)     (None, 16, 16, 384)  0           conv2d_26[0][0]
                                                                 conv2d_27[0][0]
__________________________________________________________________________________________________
conv2d_transpose_1 (Conv2DTrans (None, 16, 16, 128)  442496      concatenate_9[0][0]
__________________________________________________________________________________________________
concatenate_10 (Concatenate)    (None, 16, 16, 384)  0           conv2d_transpose_1[0][0]
                                                                 max_pooling2d_2[0][0]
__________________________________________________________________________________________________
conv2d_28 (Conv2D)              (None, 16, 16, 32)   12320       concatenate_10[0][0]
__________________________________________________________________________________________________
batch_normalization_9 (BatchNor (None, 16, 16, 32)   128         conv2d_28[0][0]
__________________________________________________________________________________________________
conv2d_29 (Conv2D)              (None, 16, 16, 128)  4224        batch_normalization_9[0][0]
__________________________________________________________________________________________________
conv2d_30 (Conv2D)              (None, 16, 16, 128)  36992       batch_normalization_9[0][0]
__________________________________________________________________________________________________
concatenate_11 (Concatenate)    (None, 16, 16, 256)  0           conv2d_29[0][0]
                                                                 conv2d_30[0][0]
__________________________________________________________________________________________________
conv2d_transpose_2 (Conv2DTrans (None, 32, 32, 64)   147520      concatenate_11[0][0]
__________________________________________________________________________________________________
concatenate_12 (Concatenate)    (None, 32, 32, 192)  0           conv2d_transpose_2[0][0]
                                                                 max_pooling2d_1[0][0]
__________________________________________________________________________________________________
conv2d_31 (Conv2D)              (None, 32, 32, 16)   3088        concatenate_12[0][0]
__________________________________________________________________________________________________
batch_normalization_10 (BatchNo (None, 32, 32, 16)   64          conv2d_31[0][0]
__________________________________________________________________________________________________
conv2d_32 (Conv2D)              (None, 32, 32, 64)   1088        batch_normalization_10[0][0]
__________________________________________________________________________________________________
```

```
conv2d_33 (Conv2D)              (None, 32, 32, 64)   9280        batch_normalization_10[0][0]

----------------------------------------------------------------------------------------------------
concatenate_13 (Concatenate)    (None, 32, 32, 128)  0           conv2d_32[0][0]

                                                                 conv2d_33[0][0]

----------------------------------------------------------------------------------------------------
conv2d_transpose_3 (Conv2DTrans (None, 64, 64, 32)   36896       concatenate_13[0][0]

----------------------------------------------------------------------------------------------------
concatenate_14 (Concatenate)    (None, 64, 64, 96)   0           conv2d_transpose_3[0][0]

                                                                 max_pooling2d[0][0]

----------------------------------------------------------------------------------------------------
conv2d_34 (Conv2D)              (None, 64, 64, 16)   1552        concatenate_14[0][0]

----------------------------------------------------------------------------------------------------
batch_normalization_11 (BatchNo (None, 64, 64, 16)   64          conv2d_34[0][0]

----------------------------------------------------------------------------------------------------
conv2d_35 (Conv2D)              (None, 64, 64, 32)   544         batch_normalization_11[0][0]

----------------------------------------------------------------------------------------------------
conv2d_36 (Conv2D)              (None, 64, 64, 32)   4640        batch_normalization_11[0][0]

----------------------------------------------------------------------------------------------------
concatenate_15 (Concatenate)    (None, 64, 64, 64)   0           conv2d_35[0][0]

                                                                 conv2d_36[0][0]

----------------------------------------------------------------------------------------------------
up_sampling2d (UpSampling2D)    (None, 128, 128, 64) 0           concatenate_15[0][0]

----------------------------------------------------------------------------------------------------
concatenate_16 (Concatenate)    (None, 128, 128, 128 0           up_sampling2d[0][0]

                                                                 conv2d[0][0]

----------------------------------------------------------------------------------------------------
conv2d_37 (Conv2D)              (None, 128, 128, 64) 73792       concatenate_16[0][0]

----------------------------------------------------------------------------------------------------
up_sampling2d_1 (UpSampling2D)  (None, 256, 256, 64) 0           conv2d_37[0][0]

----------------------------------------------------------------------------------------------------
conv2d_38 (Conv2D)              (None, 256, 256, 1)  65          up_sampling2d_1[0][0]
====================================================================================================
Total params: 2,503,889

Trainable params: 2,503,025

Non-trainable params: 864
```

Appendix G. DeepLabv3+ with ResNet50 layers and parameters

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to                     
==================================================================================================
input_1 (InputLayer)            [(None, 256, 256, 3) 0                                            
__________________________________________________________________________________________________
conv1_pad (ZeroPadding2D)       (None, 262, 262, 3)  0           input_1[0][0]                    
__________________________________________________________________________________________________
conv1_conv (Conv2D)             (None, 128, 128, 64) 9472        conv1_pad[0][0]                  
__________________________________________________________________________________________________
conv1_bn (BatchNormalization)   (None, 128, 128, 64) 256         conv1_conv[0][0]                 
__________________________________________________________________________________________________
conv1_relu (Activation)         (None, 128, 128, 64) 0           conv1_bn[0][0]                   
__________________________________________________________________________________________________
pool1_pad (ZeroPadding2D)       (None, 130, 130, 64) 0           conv1_relu[0][0]                 
__________________________________________________________________________________________________
pool1_pool (MaxPooling2D)       (None, 64, 64, 64)   0           pool1_pad[0][0]                  
__________________________________________________________________________________________________
conv2_block1_1_conv (Conv2D)    (None, 64, 64, 64)   4160        pool1_pool[0][0]                 
__________________________________________________________________________________________________
conv2_block1_1_bn (BatchNormali (None, 64, 64, 64)   256         conv2_block1_1_conv[0][0]        
__________________________________________________________________________________________________
conv2_block1_1_relu (Activation (None, 64, 64, 64)   0           conv2_block1_1_bn[0][0]          
__________________________________________________________________________________________________
conv2_block1_2_conv (Conv2D)    (None, 64, 64, 64)   36928       conv2_block1_1_relu[0][0]        
__________________________________________________________________________________________________
conv2_block1_2_bn (BatchNormali (None, 64, 64, 64)   256         conv2_block1_2_conv[0][0]        
__________________________________________________________________________________________________
conv2_block1_2_relu (Activation (None, 64, 64, 64)   0           conv2_block1_2_bn[0][0]          
__________________________________________________________________________________________________
conv2_block1_0_conv (Conv2D)    (None, 64, 64, 256)  16640       pool1_pool[0][0]                 
__________________________________________________________________________________________________
conv2_block1_3_conv (Conv2D)    (None, 64, 64, 256)  16640       conv2_block1_2_relu[0][0]        
__________________________________________________________________________________________________
conv2_block1_0_bn (BatchNormali (None, 64, 64, 256)  1024        conv2_block1_0_conv[0][0]        
__________________________________________________________________________________________________
conv2_block1_3_bn (BatchNormali (None, 64, 64, 256)  1024        conv2_block1_3_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_add (Add)          (None, 64, 64, 256)  0           conv2_block1_0_bn[0][0]
                                                                 conv2_block1_3_bn[0][0]
__________________________________________________________________________________________________
conv2_block1_out (Activation)   (None, 64, 64, 256)  0           conv2_block1_add[0][0]
__________________________________________________________________________________________________
conv2_block2_1_conv (Conv2D)    (None, 64, 64, 64)   16448       conv2_block1_out[0][0]
__________________________________________________________________________________________________
conv2_block2_1_bn (BatchNormali (None, 64, 64, 64)   256         conv2_block2_1_conv[0][0]
__________________________________________________________________________________________________
conv2_block2_1_relu (Activation (None, 64, 64, 64)   0           conv2_block2_1_bn[0][0]
__________________________________________________________________________________________________
conv2_block2_2_conv (Conv2D)    (None, 64, 64, 64)   36928       conv2_block2_1_relu[0][0]
__________________________________________________________________________________________________
conv2_block2_2_bn (BatchNormali (None, 64, 64, 64)   256         conv2_block2_2_conv[0][0]
__________________________________________________________________________________________________
conv2_block2_2_relu (Activation (None, 64, 64, 64)   0           conv2_block2_2_bn[0][0]
__________________________________________________________________________________________________
conv2_block2_3_conv (Conv2D)    (None, 64, 64, 256)  16640       conv2_block2_2_relu[0][0]
__________________________________________________________________________________________________
conv2_block2_3_bn (BatchNormali (None, 64, 64, 256)  1024        conv2_block2_3_conv[0][0]
__________________________________________________________________________________________________
conv2_block2_add (Add)          (None, 64, 64, 256)  0           conv2_block1_out[0][0]
                                                                 conv2_block2_3_bn[0][0]
__________________________________________________________________________________________________
conv2_block2_out (Activation)   (None, 64, 64, 256)  0           conv2_block2_add[0][0]
__________________________________________________________________________________________________
conv2_block3_1_conv (Conv2D)    (None, 64, 64, 64)   16448       conv2_block2_out[0][0]
__________________________________________________________________________________________________
conv2_block3_1_bn (BatchNormali (None, 64, 64, 64)   256         conv2_block3_1_conv[0][0]
__________________________________________________________________________________________________
conv2_block3_1_relu (Activation (None, 64, 64, 64)   0           conv2_block3_1_bn[0][0]
__________________________________________________________________________________________________
conv2_block3_2_conv (Conv2D)    (None, 64, 64, 64)   36928       conv2_block3_1_relu[0][0]
__________________________________________________________________________________________________
```

```
conv2_block3_2_bn (BatchNormali (None, 64, 64, 64)   256         conv2_block3_2_conv[0][0]
__________________________________________________________________________________________________
conv2_block3_2_relu (Activation (None, 64, 64, 64)   0           conv2_block3_2_bn[0][0]
__________________________________________________________________________________________________
conv2_block3_3_conv (Conv2D)    (None, 64, 64, 256)  16640       conv2_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv2_block3_3_bn (BatchNormali (None, 64, 64, 256)  1024        conv2_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv2_block3_add (Add)          (None, 64, 64, 256)  0           conv2_block2_out[0][0]
                                                                 conv2_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv2_block3_out (Activation)   (None, 64, 64, 256)  0           conv2_block3_add[0][0]
__________________________________________________________________________________________________
conv3_block1_1_conv (Conv2D)    (None, 32, 32, 128)  32896       conv2_block3_out[0][0]
__________________________________________________________________________________________________
conv3_block1_1_bn (BatchNormali (None, 32, 32, 128)  512         conv3_block1_1_conv[0][0]
__________________________________________________________________________________________________
conv3_block1_1_relu (Activation (None, 32, 32, 128)  0           conv3_block1_1_bn[0][0]
__________________________________________________________________________________________________
conv3_block1_2_conv (Conv2D)    (None, 32, 32, 128)  147584      conv3_block1_1_relu[0][0]
__________________________________________________________________________________________________
conv3_block1_2_bn (BatchNormali (None, 32, 32, 128)  512         conv3_block1_2_conv[0][0]
__________________________________________________________________________________________________
conv3_block1_2_relu (Activation (None, 32, 32, 128)  0           conv3_block1_2_bn[0][0]
__________________________________________________________________________________________________
conv3_block1_0_conv (Conv2D)    (None, 32, 32, 512)  131584      conv2_block3_out[0][0]
__________________________________________________________________________________________________
conv3_block1_3_conv (Conv2D)    (None, 32, 32, 512)  66048       conv3_block1_2_relu[0][0]
__________________________________________________________________________________________________
conv3_block1_0_bn (BatchNormali (None, 32, 32, 512)  2048        conv3_block1_0_conv[0][0]
__________________________________________________________________________________________________
conv3_block1_3_bn (BatchNormali (None, 32, 32, 512)  2048        conv3_block1_3_conv[0][0]
__________________________________________________________________________________________________
conv3_block1_add (Add)          (None, 32, 32, 512)  0           conv3_block1_0_bn[0][0]
                                                                 conv3_block1_3_bn[0][0]
__________________________________________________________________________________________________
conv3_block1_out (Activation)   (None, 32, 32, 512)  0           conv3_block1_add[0][0]
```

```
__________________________________________________________________________________________________
conv3_block2_1_conv (Conv2D)    (None, 32, 32, 128)  65664       conv3_block1_out[0][0]
__________________________________________________________________________________________________
conv3_block2_1_bn (BatchNormali (None, 32, 32, 128)  512         conv3_block2_1_conv[0][0]
__________________________________________________________________________________________________
conv3_block2_1_relu (Activation (None, 32, 32, 128)  0           conv3_block2_1_bn[0][0]
__________________________________________________________________________________________________
conv3_block2_2_conv (Conv2D)    (None, 32, 32, 128)  147584      conv3_block2_1_relu[0][0]
__________________________________________________________________________________________________
conv3_block2_2_bn (BatchNormali (None, 32, 32, 128)  512         conv3_block2_2_conv[0][0]
__________________________________________________________________________________________________
conv3_block2_2_relu (Activation (None, 32, 32, 128)  0           conv3_block2_2_bn[0][0]
__________________________________________________________________________________________________
conv3_block2_3_conv (Conv2D)    (None, 32, 32, 512)  66048       conv3_block2_2_relu[0][0]
__________________________________________________________________________________________________
conv3_block2_3_bn (BatchNormali (None, 32, 32, 512)  2048        conv3_block2_3_conv[0][0]
__________________________________________________________________________________________________
conv3_block2_add (Add)          (None, 32, 32, 512)  0           conv3_block1_out[0][0]
                                                                 conv3_block2_3_bn[0][0]
__________________________________________________________________________________________________
conv3_block2_out (Activation)   (None, 32, 32, 512)  0           conv3_block2_add[0][0]
__________________________________________________________________________________________________
conv3_block3_1_conv (Conv2D)    (None, 32, 32, 128)  65664       conv3_block2_out[0][0]
__________________________________________________________________________________________________
conv3_block3_1_bn (BatchNormali (None, 32, 32, 128)  512         conv3_block3_1_conv[0][0]
__________________________________________________________________________________________________
conv3_block3_1_relu (Activation (None, 32, 32, 128)  0           conv3_block3_1_bn[0][0]
__________________________________________________________________________________________________
conv3_block3_2_conv (Conv2D)    (None, 32, 32, 128)  147584      conv3_block3_1_relu[0][0]
__________________________________________________________________________________________________
conv3_block3_2_bn (BatchNormali (None, 32, 32, 128)  512         conv3_block3_2_conv[0][0]
__________________________________________________________________________________________________
conv3_block3_2_relu (Activation (None, 32, 32, 128)  0           conv3_block3_2_bn[0][0]
__________________________________________________________________________________________________
conv3_block3_3_conv (Conv2D)    (None, 32, 32, 512)  66048       conv3_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv3_block3_3_bn (BatchNormali (None, 32, 32, 512)  2048        conv3_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv3_block3_add (Add)          (None, 32, 32, 512)  0           conv3_block2_out[0][0]
                                                                 conv3_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv3_block3_out (Activation)   (None, 32, 32, 512)  0           conv3_block3_add[0][0]
__________________________________________________________________________________________________
conv3_block4_1_conv (Conv2D)    (None, 32, 32, 128)  65664       conv3_block3_out[0][0]
__________________________________________________________________________________________________
conv3_block4_1_bn (BatchNormali (None, 32, 32, 128)  512         conv3_block4_1_conv[0][0]
__________________________________________________________________________________________________
conv3_block4_1_relu (Activation (None, 32, 32, 128)  0           conv3_block4_1_bn[0][0]
__________________________________________________________________________________________________
conv3_block4_2_conv (Conv2D)    (None, 32, 32, 128)  147584      conv3_block4_1_relu[0][0]
__________________________________________________________________________________________________
conv3_block4_2_bn (BatchNormali (None, 32, 32, 128)  512         conv3_block4_2_conv[0][0]
__________________________________________________________________________________________________
conv3_block4_2_relu (Activation (None, 32, 32, 128)  0           conv3_block4_2_bn[0][0]
__________________________________________________________________________________________________
conv3_block4_3_conv (Conv2D)    (None, 32, 32, 512)  66048       conv3_block4_2_relu[0][0]
__________________________________________________________________________________________________
conv3_block4_3_bn (BatchNormali (None, 32, 32, 512)  2048        conv3_block4_3_conv[0][0]
__________________________________________________________________________________________________
conv3_block4_add (Add)          (None, 32, 32, 512)  0           conv3_block3_out[0][0]
                                                                 conv3_block4_3_bn[0][0]
__________________________________________________________________________________________________
conv3_block4_out (Activation)   (None, 32, 32, 512)  0           conv3_block4_add[0][0]
__________________________________________________________________________________________________
conv4_block1_1_conv (Conv2D)    (None, 16, 16, 256)  131328      conv3_block4_out[0][0]
__________________________________________________________________________________________________
conv4_block1_1_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block1_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block1_1_relu (Activation (None, 16, 16, 256)  0           conv4_block1_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block1_2_conv (Conv2D)    (None, 16, 16, 256)  590080      conv4_block1_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block1_2_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block1_2_conv[0][0]
__________________________________________________________________________________________________
```

```
conv4_block1_2_relu (Activation (None, 16, 16, 256)  0           conv4_block1_2_bn[0][0]

__________________________________________________________________________________________________
conv4_block1_0_conv (Conv2D)    (None, 16, 16, 1024) 525312      conv3_block4_out[0][0]

__________________________________________________________________________________________________
conv4_block1_3_conv (Conv2D)    (None, 16, 16, 1024) 263168      conv4_block1_2_relu[0][0]

__________________________________________________________________________________________________
conv4_block1_0_bn (BatchNormali (None, 16, 16, 1024) 4096        conv4_block1_0_conv[0][0]

__________________________________________________________________________________________________
conv4_block1_3_bn (BatchNormali (None, 16, 16, 1024) 4096        conv4_block1_3_conv[0][0]

__________________________________________________________________________________________________
conv4_block1_add (Add)          (None, 16, 16, 1024) 0           conv4_block1_0_bn[0][0]

                                                                 conv4_block1_3_bn[0][0]

__________________________________________________________________________________________________
conv4_block1_out (Activation)   (None, 16, 16, 1024) 0           conv4_block1_add[0][0]

__________________________________________________________________________________________________
conv4_block2_1_conv (Conv2D)    (None, 16, 16, 256)  262400      conv4_block1_out[0][0]

__________________________________________________________________________________________________
conv4_block2_1_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block2_1_conv[0][0]

__________________________________________________________________________________________________
conv4_block2_1_relu (Activation (None, 16, 16, 256)  0           conv4_block2_1_bn[0][0]

__________________________________________________________________________________________________
conv4_block2_2_conv (Conv2D)    (None, 16, 16, 256)  590080      conv4_block2_1_relu[0][0]

__________________________________________________________________________________________________
conv4_block2_2_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block2_2_conv[0][0]

__________________________________________________________________________________________________
conv4_block2_2_relu (Activation (None, 16, 16, 256)  0           conv4_block2_2_bn[0][0]

__________________________________________________________________________________________________
conv4_block2_3_conv (Conv2D)    (None, 16, 16, 1024) 263168      conv4_block2_2_relu[0][0]

__________________________________________________________________________________________________
conv4_block2_3_bn (BatchNormali (None, 16, 16, 1024) 4096        conv4_block2_3_conv[0][0]

__________________________________________________________________________________________________
conv4_block2_add (Add)          (None, 16, 16, 1024) 0           conv4_block1_out[0][0]

                                                                 conv4_block2_3_bn[0][0]

__________________________________________________________________________________________________
conv4_block2_out (Activation)   (None, 16, 16, 1024) 0           conv4_block2_add[0][0]

__________________________________________________________________________________________________
conv4_block3_1_conv (Conv2D)    (None, 16, 16, 256)  262400      conv4_block2_out[0][0]
```

```
__________________________________________________________________________________________________
conv4_block3_1_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block3_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block3_1_relu (Activation (None, 16, 16, 256)  0           conv4_block3_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block3_2_conv (Conv2D)    (None, 16, 16, 256)  590080      conv4_block3_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block3_2_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block3_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block3_2_relu (Activation (None, 16, 16, 256)  0           conv4_block3_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block3_3_conv (Conv2D)    (None, 16, 16, 1024) 263168      conv4_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block3_3_bn (BatchNormali (None, 16, 16, 1024) 4096        conv4_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block3_add (Add)          (None, 16, 16, 1024) 0           conv4_block2_out[0][0]
                                                                 conv4_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block3_out (Activation)   (None, 16, 16, 1024) 0           conv4_block3_add[0][0]
__________________________________________________________________________________________________
conv4_block4_1_conv (Conv2D)    (None, 16, 16, 256)  262400      conv4_block3_out[0][0]
__________________________________________________________________________________________________
conv4_block4_1_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block4_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block4_1_relu (Activation (None, 16, 16, 256)  0           conv4_block4_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block4_2_conv (Conv2D)    (None, 16, 16, 256)  590080      conv4_block4_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block4_2_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block4_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block4_2_relu (Activation (None, 16, 16, 256)  0           conv4_block4_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block4_3_conv (Conv2D)    (None, 16, 16, 1024) 263168      conv4_block4_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block4_3_bn (BatchNormali (None, 16, 16, 1024) 4096        conv4_block4_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block4_add (Add)          (None, 16, 16, 1024) 0           conv4_block3_out[0][0]
```

```
                                                                 conv4_block4_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block4_out (Activation)   (None, 16, 16, 1024) 0           conv4_block4_add[0][0]
__________________________________________________________________________________________________
conv4_block5_1_conv (Conv2D)    (None, 16, 16, 256)  262400      conv4_block4_out[0][0]
__________________________________________________________________________________________________
conv4_block5_1_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block5_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block5_1_relu (Activation (None, 16, 16, 256)  0           conv4_block5_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block5_2_conv (Conv2D)    (None, 16, 16, 256)  590080      conv4_block5_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block5_2_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block5_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block5_2_relu (Activation (None, 16, 16, 256)  0           conv4_block5_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block5_3_conv (Conv2D)    (None, 16, 16, 1024) 263168      conv4_block5_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block5_3_bn (BatchNormali (None, 16, 16, 1024) 4096        conv4_block5_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block5_add (Add)          (None, 16, 16, 1024) 0           conv4_block4_out[0][0]
                                                                 conv4_block5_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block5_out (Activation)   (None, 16, 16, 1024) 0           conv4_block5_add[0][0]
__________________________________________________________________________________________________
conv4_block6_1_conv (Conv2D)    (None, 16, 16, 256)  262400      conv4_block5_out[0][0]
__________________________________________________________________________________________________
conv4_block6_1_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block6_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block6_1_relu (Activation (None, 16, 16, 256)  0           conv4_block6_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block6_2_conv (Conv2D)    (None, 16, 16, 256)  590080      conv4_block6_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block6_2_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block6_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block6_2_relu (Activation (None, 16, 16, 256)  0           conv4_block6_2_bn[0][0]
__________________________________________________________________________________________________
```

```
conv4_block6_3_conv (Conv2D)    (None, 16, 16, 1024) 263168      conv4_block6_2_relu[0][0]

__________________________________________________________________________________________________
conv4_block6_3_bn (BatchNormali (None, 16, 16, 1024) 4096        conv4_block6_3_conv[0][0]

__________________________________________________________________________________________________
conv4_block6_add (Add)          (None, 16, 16, 1024) 0           conv4_block5_out[0][0]

                                                                 conv4_block6_3_bn[0][0]

__________________________________________________________________________________________________
conv4_block6_out (Activation)   (None, 16, 16, 1024) 0           conv4_block6_add[0][0]

__________________________________________________________________________________________________
conv5_block1_1_conv (Conv2D)    (None, 8, 8, 512)    524800      conv4_block6_out[0][0]

__________________________________________________________________________________________________
conv5_block1_1_bn (BatchNormali (None, 8, 8, 512)    2048        conv5_block1_1_conv[0][0]

__________________________________________________________________________________________________
conv5_block1_1_relu (Activation (None, 8, 8, 512)    0           conv5_block1_1_bn[0][0]

__________________________________________________________________________________________________
conv5_block1_2_conv (Conv2D)    (None, 8, 8, 512)    2359808     conv5_block1_1_relu[0][0]

__________________________________________________________________________________________________
conv5_block1_2_bn (BatchNormali (None, 8, 8, 512)    2048        conv5_block1_2_conv[0][0]

__________________________________________________________________________________________________
conv5_block1_2_relu (Activation (None, 8, 8, 512)    0           conv5_block1_2_bn[0][0]

__________________________________________________________________________________________________
conv5_block1_0_conv (Conv2D)    (None, 8, 8, 2048)   2099200     conv4_block6_out[0][0]

__________________________________________________________________________________________________
conv5_block1_3_conv (Conv2D)    (None, 8, 8, 2048)   1050624     conv5_block1_2_relu[0][0]

__________________________________________________________________________________________________
conv5_block1_0_bn (BatchNormali (None, 8, 8, 2048)   8192        conv5_block1_0_conv[0][0]

__________________________________________________________________________________________________
conv5_block1_3_bn (BatchNormali (None, 8, 8, 2048)   8192        conv5_block1_3_conv[0][0]

__________________________________________________________________________________________________
conv5_block1_add (Add)          (None, 8, 8, 2048)   0           conv5_block1_0_bn[0][0]

                                                                 conv5_block1_3_bn[0][0]

__________________________________________________________________________________________________
conv5_block1_out (Activation)   (None, 8, 8, 2048)   0           conv5_block1_add[0][0]

__________________________________________________________________________________________________
conv5_block2_1_conv (Conv2D)    (None, 8, 8, 512)    1049088     conv5_block1_out[0][0]

__________________________________________________________________________________________________
conv5_block2_1_bn (BatchNormali (None, 8, 8, 512)    2048        conv5_block2_1_conv[0][0]
__________________________________________________________________________________________________
conv5_block2_1_relu (Activation (None, 8, 8, 512)    0           conv5_block2_1_bn[0][0]
__________________________________________________________________________________________________
conv5_block2_2_conv (Conv2D)    (None, 8, 8, 512)    2359808     conv5_block2_1_relu[0][0]
__________________________________________________________________________________________________
conv5_block2_2_bn (BatchNormali (None, 8, 8, 512)    2048        conv5_block2_2_conv[0][0]
__________________________________________________________________________________________________
conv5_block2_2_relu (Activation (None, 8, 8, 512)    0           conv5_block2_2_bn[0][0]
__________________________________________________________________________________________________
conv5_block2_3_conv (Conv2D)    (None, 8, 8, 2048)   1050624     conv5_block2_2_relu[0][0]
__________________________________________________________________________________________________
conv5_block2_3_bn (BatchNormali (None, 8, 8, 2048)   8192        conv5_block2_3_conv[0][0]
__________________________________________________________________________________________________
conv5_block2_add (Add)          (None, 8, 8, 2048)   0           conv5_block1_out[0][0]
                                                                 conv5_block2_3_bn[0][0]
__________________________________________________________________________________________________
conv5_block2_out (Activation)   (None, 8, 8, 2048)   0           conv5_block2_add[0][0]
__________________________________________________________________________________________________
conv5_block3_1_conv (Conv2D)    (None, 8, 8, 512)    1049088     conv5_block2_out[0][0]
__________________________________________________________________________________________________
conv5_block3_1_bn (BatchNormali (None, 8, 8, 512)    2048        conv5_block3_1_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_1_relu (Activation (None, 8, 8, 512)    0           conv5_block3_1_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_2_conv (Conv2D)    (None, 8, 8, 512)    2359808     conv5_block3_1_relu[0][0]
__________________________________________________________________________________________________
conv5_block3_2_bn (BatchNormali (None, 8, 8, 512)    2048        conv5_block3_2_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_2_relu (Activation (None, 8, 8, 512)    0           conv5_block3_2_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_3_conv (Conv2D)    (None, 8, 8, 2048)   1050624     conv5_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv5_block3_3_bn (BatchNormali (None, 8, 8, 2048)   8192        conv5_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_add (Add)          (None, 8, 8, 2048)   0           conv5_block2_out[0][0]
                                                                 conv5_block3_3_bn[0][0]
__________________________________________________________________________________________________
```

```
conv5_block3_out (Activation)   (None, 8, 8, 2048)   0           conv5_block3_add[0][0]
__________________________________________________________________________________________________
average_pooling (AveragePooling (None, 1, 1, 2048)   0           conv5_block3_out[0][0]
__________________________________________________________________________________________________
pool_1x1conv2d (Conv2D)         (None, 1, 1, 256)    524288      average_pooling[0][0]
__________________________________________________________________________________________________
bn_1 (BatchNormalization)       (None, 1, 1, 256)    1024        pool_1x1conv2d[0][0]
__________________________________________________________________________________________________
ASPP_conv2d_d1 (Conv2D)         (None, 8, 8, 256)    524288      conv5_block3_out[0][0]
__________________________________________________________________________________________________
ASPP_conv2d_d6 (Conv2D)         (None, 8, 8, 256)    4718592     conv5_block3_out[0][0]
__________________________________________________________________________________________________
ASPP_conv2d_d12 (Conv2D)        (None, 8, 8, 256)    4718592     conv5_block3_out[0][0]
__________________________________________________________________________________________________
ASPP_conv2d_d18 (Conv2D)        (None, 8, 8, 256)    4718592     conv5_block3_out[0][0]
__________________________________________________________________________________________________
relu_1 (Activation)             (None, 1, 1, 256)    0           bn_1[0][0]
__________________________________________________________________________________________________
bn_2 (BatchNormalization)       (None, 8, 8, 256)    1024        ASPP_conv2d_d1[0][0]
__________________________________________________________________________________________________
bn_3 (BatchNormalization)       (None, 8, 8, 256)    1024        ASPP_conv2d_d6[0][0]
__________________________________________________________________________________________________
bn_4 (BatchNormalization)       (None, 8, 8, 256)    1024        ASPP_conv2d_d12[0][0]
__________________________________________________________________________________________________
bn_5 (BatchNormalization)       (None, 8, 8, 256)    1024        ASPP_conv2d_d18[0][0]
__________________________________________________________________________________________________
relu_1_upsample (Lambda)        (None, 8, 8, 256)    0           relu_1[0][0]
__________________________________________________________________________________________________
relu_2 (Activation)             (None, 8, 8, 256)    0           bn_2[0][0]
__________________________________________________________________________________________________
relu_3 (Activation)             (None, 8, 8, 256)    0           bn_3[0][0]
__________________________________________________________________________________________________
relu_4 (Activation)             (None, 8, 8, 256)    0           bn_4[0][0]
__________________________________________________________________________________________________
relu_5 (Activation)             (None, 8, 8, 256)    0           bn_5[0][0]
__________________________________________________________________________________________________
ASPP_concat (Concatenate)       (None, 8, 8, 1280)   0           relu_1_upsample[0][0]
```

```
                                                                relu_2[0][0]

                                                                relu_3[0][0]

                                                                relu_4[0][0]

                                                                relu_5[0][0]

__________________________________________________________________________________________________
ASPP_conv2d_final (Conv2D)      (None, 8, 8, 256)    327680      ASPP_concat[0][0]

__________________________________________________________________________________________________
bn_final (BatchNormalization)   (None, 8, 8, 256)    1024        ASPP_conv2d_final[0][0]

__________________________________________________________________________________________________
low_level_projection (Conv2D)   (None, 64, 64, 48)   12288       conv2_block3_out[0][0]

__________________________________________________________________________________________________
relu_final (Activation)         (None, 8, 8, 256)    0           bn_final[0][0]

__________________________________________________________________________________________________
bn_low_level_projection (BatchN (None, 64, 64, 48)   192         low_level_projection[0][0]

__________________________________________________________________________________________________
relu_final_upsample (Lambda)    (None, 64, 64, 256)  0           relu_final[0][0]

__________________________________________________________________________________________________
low_level_activation (Activatio (None, 64, 64, 48)   0           bn_low_level_projection[0][0]

__________________________________________________________________________________________________
decoder_concat (Concatenate)    (None, 64, 64, 304)  0           relu_final_upsample[0][0]

                                                                 low_level_activation[0][0]

__________________________________________________________________________________________________
decoder_conv2d_1 (Conv2D)       (None, 64, 64, 256)  700416      decoder_concat[0][0]

__________________________________________________________________________________________________
bn_decoder_1 (BatchNormalizatio (None, 64, 64, 256)  1024        decoder_conv2d_1[0][0]

__________________________________________________________________________________________________
activation_decoder_1 (Activatio (None, 64, 64, 256)  0           bn_decoder_1[0][0]

__________________________________________________________________________________________________
decoder_conv2d_2 (Conv2D)       (None, 64, 64, 256)  589824      activation_decoder_1[0][0]

__________________________________________________________________________________________________
bn_decoder_2 (BatchNormalizatio (None, 64, 64, 256)  1024        decoder_conv2d_2[0][0]

__________________________________________________________________________________________________
activation_decoder_2 (Activatio (None, 64, 64, 256)  0           bn_decoder_2[0][0]

__________________________________________________________________________________________________
activation_decoder_2_upsample ( (None, 256, 256, 256 0           activation_decoder_2[0][0]

__________________________________________________________________________________________________
output_layer (Conv2D)           (None, 256, 256, 1)  257         activation_decoder_2_upsample[0][
==================================================================================================
Total params: 40,430,913

Trainable params: 40,373,601

Non-trainable params: 57,312
```

Appendix H. DeepLabv3+ with EfficientNet layers and parameters

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 256, 256, 3) 0
__________________________________________________________________________________________________
rescaling (Rescaling)           (None, 256, 256, 3)  0           input_1[0][0]
__________________________________________________________________________________________________
normalization (Normalization)   (None, 256, 256, 3)  7           rescaling[0][0]
__________________________________________________________________________________________________
stem_conv_pad (ZeroPadding2D)   (None, 257, 257, 3)  0           normalization[0][0]
__________________________________________________________________________________________________
stem_conv (Conv2D)              (None, 128, 128, 48) 1296        stem_conv_pad[0][0]
__________________________________________________________________________________________________
stem_bn (BatchNormalization)    (None, 128, 128, 48) 192         stem_conv[0][0]
__________________________________________________________________________________________________
stem_activation (Activation)    (None, 128, 128, 48) 0           stem_bn[0][0]
__________________________________________________________________________________________________
block1a_dwconv (DepthwiseConv2D (None, 128, 128, 48) 432         stem_activation[0][0]
__________________________________________________________________________________________________
block1a_bn (BatchNormalization) (None, 128, 128, 48) 192         block1a_dwconv[0][0]
__________________________________________________________________________________________________
block1a_activation (Activation) (None, 128, 128, 48) 0           block1a_bn[0][0]
__________________________________________________________________________________________________
block1a_se_squeeze (GlobalAvera (None, 48)           0           block1a_activation[0][0]
__________________________________________________________________________________________________
block1a_se_reshape (Reshape)    (None, 1, 1, 48)     0           block1a_se_squeeze[0][0]
__________________________________________________________________________________________________
block1a_se_reduce (Conv2D)      (None, 1, 1, 12)     588         block1a_se_reshape[0][0]
__________________________________________________________________________________________________
block1a_se_expand (Conv2D)      (None, 1, 1, 48)     624         block1a_se_reduce[0][0]
__________________________________________________________________________________________________
block1a_se_excite (Multiply)    (None, 128, 128, 48) 0           block1a_activation[0][0]
                                                                 block1a_se_expand[0][0]
__________________________________________________________________________________________________
```

```
block1a_project_conv (Conv2D)   (None, 128, 128, 24) 1152        block1a_se_excite[0][0]

__________________________________________________________________________________________________
block1a_project_bn (BatchNormal (None, 128, 128, 24) 96          block1a_project_conv[0][0]

__________________________________________________________________________________________________
block1b_dwconv (DepthwiseConv2D (None, 128, 128, 24) 216         block1a_project_bn[0][0]

__________________________________________________________________________________________________
block1b_bn (BatchNormalization) (None, 128, 128, 24) 96          block1b_dwconv[0][0]

__________________________________________________________________________________________________
block1b_activation (Activation) (None, 128, 128, 24) 0           block1b_bn[0][0]

__________________________________________________________________________________________________
block1b_se_squeeze (GlobalAvera (None, 24)           0           block1b_activation[0][0]

__________________________________________________________________________________________________
block1b_se_reshape (Reshape)    (None, 1, 1, 24)     0           block1b_se_squeeze[0][0]

__________________________________________________________________________________________________
block1b_se_reduce (Conv2D)      (None, 1, 1, 6)      150         block1b_se_reshape[0][0]

__________________________________________________________________________________________________
block1b_se_expand (Conv2D)      (None, 1, 1, 24)     168         block1b_se_reduce[0][0]

__________________________________________________________________________________________________
block1b_se_excite (Multiply)    (None, 128, 128, 24) 0           block1b_activation[0][0]

                                                                 block1b_se_expand[0][0]

__________________________________________________________________________________________________
block1b_project_conv (Conv2D)   (None, 128, 128, 24) 576         block1b_se_excite[0][0]

__________________________________________________________________________________________________
block1b_project_bn (BatchNormal (None, 128, 128, 24) 96          block1b_project_conv[0][0]

__________________________________________________________________________________________________
block1b_drop (Dropout)          (None, 128, 128, 24) 0           block1b_project_bn[0][0]

__________________________________________________________________________________________________
block1b_add (Add)               (None, 128, 128, 24) 0           block1b_drop[0][0]

                                                                 block1a_project_bn[0][0]

__________________________________________________________________________________________________
block2a_expand_conv (Conv2D)    (None, 128, 128, 144 3456        block1b_add[0][0]

__________________________________________________________________________________________________
block2a_expand_bn (BatchNormali (None, 128, 128, 144 576         block2a_expand_conv[0][0]

__________________________________________________________________________________________________
block2a_expand_activation (Acti (None, 128, 128, 144 0           block2a_expand_bn[0][0]

__________________________________________________________________________________________________
block2a_dwconv_pad (ZeroPadding (None, 129, 129, 144 0           block2a_expand_activation[0][0]
__________________________________________________________________________________________________
block2a_dwconv (DepthwiseConv2D (None, 64, 64, 144)  1296        block2a_dwconv_pad[0][0]
__________________________________________________________________________________________________
block2a_bn (BatchNormalization) (None, 64, 64, 144)  576         block2a_dwconv[0][0]
__________________________________________________________________________________________________
block2a_activation (Activation) (None, 64, 64, 144)  0           block2a_bn[0][0]
__________________________________________________________________________________________________
block2a_se_squeeze (GlobalAvera (None, 144)          0           block2a_activation[0][0]
__________________________________________________________________________________________________
block2a_se_reshape (Reshape)    (None, 1, 1, 144)    0           block2a_se_squeeze[0][0]
__________________________________________________________________________________________________
block2a_se_reduce (Conv2D)      (None, 1, 1, 6)      870         block2a_se_reshape[0][0]
__________________________________________________________________________________________________
block2a_se_expand (Conv2D)      (None, 1, 1, 144)    1008        block2a_se_reduce[0][0]
__________________________________________________________________________________________________
block2a_se_excite (Multiply)    (None, 64, 64, 144)  0           block2a_activation[0][0]
                                                                 block2a_se_expand[0][0]
__________________________________________________________________________________________________
block2a_project_conv (Conv2D)   (None, 64, 64, 32)   4608        block2a_se_excite[0][0]
__________________________________________________________________________________________________
block2a_project_bn (BatchNormal (None, 64, 64, 32)   128         block2a_project_conv[0][0]
__________________________________________________________________________________________________
block2b_expand_conv (Conv2D)    (None, 64, 64, 192)  6144        block2a_project_bn[0][0]
__________________________________________________________________________________________________
block2b_expand_bn (BatchNormali (None, 64, 64, 192)  768         block2b_expand_conv[0][0]
__________________________________________________________________________________________________
block2b_expand_activation (Acti (None, 64, 64, 192)  0           block2b_expand_bn[0][0]
__________________________________________________________________________________________________
block2b_dwconv (DepthwiseConv2D (None, 64, 64, 192)  1728        block2b_expand_activation[0][0]
__________________________________________________________________________________________________
block2b_bn (BatchNormalization) (None, 64, 64, 192)  768         block2b_dwconv[0][0]
__________________________________________________________________________________________________
block2b_activation (Activation) (None, 64, 64, 192)  0           block2b_bn[0][0]
__________________________________________________________________________________________________
block2b_se_squeeze (GlobalAvera (None, 192)          0           block2b_activation[0][0]
__________________________________________________________________________________________________
block2b_se_reshape (Reshape)    (None, 1, 1, 192)    0           block2b_se_squeeze[0][0]
```

```
-----------------------------------------------------------------------------------------------
block2b_se_reduce (Conv2D)      (None, 1, 1, 8)      1544        block2b_se_reshape[0][0]

-----------------------------------------------------------------------------------------------
block2b_se_expand (Conv2D)      (None, 1, 1, 192)    1728        block2b_se_reduce[0][0]

-----------------------------------------------------------------------------------------------
block2b_se_excite (Multiply)    (None, 64, 64, 192)  0           block2b_activation[0][0]
                                                                 block2b_se_expand[0][0]

-----------------------------------------------------------------------------------------------
block2b_project_conv (Conv2D)   (None, 64, 64, 32)   6144        block2b_se_excite[0][0]

-----------------------------------------------------------------------------------------------
block2b_project_bn (BatchNormal (None, 64, 64, 32)   128         block2b_project_conv[0][0]

-----------------------------------------------------------------------------------------------
block2b_drop (Dropout)          (None, 64, 64, 32)   0           block2b_project_bn[0][0]

-----------------------------------------------------------------------------------------------
block2b_add (Add)               (None, 64, 64, 32)   0           block2b_drop[0][0]
                                                                 block2a_project_bn[0][0]

-----------------------------------------------------------------------------------------------
block2c_expand_conv (Conv2D)    (None, 64, 64, 192)  6144        block2b_add[0][0]

-----------------------------------------------------------------------------------------------
block2c_expand_bn (BatchNormali (None, 64, 64, 192)  768         block2c_expand_conv[0][0]

-----------------------------------------------------------------------------------------------
block2c_expand_activation (Acti (None, 64, 64, 192)  0           block2c_expand_bn[0][0]

-----------------------------------------------------------------------------------------------
block2c_dwconv (DepthwiseConv2D (None, 64, 64, 192)  1728        block2c_expand_activation[0][0]

-----------------------------------------------------------------------------------------------
block2c_bn (BatchNormalization) (None, 64, 64, 192)  768         block2c_dwconv[0][0]

-----------------------------------------------------------------------------------------------
block2c_activation (Activation) (None, 64, 64, 192)  0           block2c_bn[0][0]

-----------------------------------------------------------------------------------------------
block2c_se_squeeze (GlobalAvera (None, 192)          0           block2c_activation[0][0]

-----------------------------------------------------------------------------------------------
block2c_se_reshape (Reshape)    (None, 1, 1, 192)    0           block2c_se_squeeze[0][0]

-----------------------------------------------------------------------------------------------
block2c_se_reduce (Conv2D)      (None, 1, 1, 8)      1544        block2c_se_reshape[0][0]

-----------------------------------------------------------------------------------------------
block2c_se_expand (Conv2D)      (None, 1, 1, 192)    1728        block2c_se_reduce[0][0]

-----------------------------------------------------------------------------------------------
```

```
block2c_se_excite (Multiply)    (None, 64, 64, 192)  0           block2c_activation[0][0]
                                                                 block2c_se_expand[0][0]
__________________________________________________________________________________________________
block2c_project_conv (Conv2D)   (None, 64, 64, 32)   6144        block2c_se_excite[0][0]
__________________________________________________________________________________________________
block2c_project_bn (BatchNormal (None, 64, 64, 32)   128         block2c_project_conv[0][0]
__________________________________________________________________________________________________
block2c_drop (Dropout)          (None, 64, 64, 32)   0           block2c_project_bn[0][0]
__________________________________________________________________________________________________
block2c_add (Add)               (None, 64, 64, 32)   0           block2c_drop[0][0]
                                                                 block2b_add[0][0]
__________________________________________________________________________________________________
block2d_expand_conv (Conv2D)    (None, 64, 64, 192)  6144        block2c_add[0][0]
__________________________________________________________________________________________________
block2d_expand_bn (BatchNormali (None, 64, 64, 192)  768         block2d_expand_conv[0][0]
__________________________________________________________________________________________________
block2d_expand_activation (Acti (None, 64, 64, 192)  0           block2d_expand_bn[0][0]
__________________________________________________________________________________________________
block2d_dwconv (DepthwiseConv2D (None, 64, 64, 192)  1728        block2d_expand_activation[0][0]
__________________________________________________________________________________________________
block2d_bn (BatchNormalization) (None, 64, 64, 192)  768         block2d_dwconv[0][0]
__________________________________________________________________________________________________
block2d_activation (Activation) (None, 64, 64, 192)  0           block2d_bn[0][0]
__________________________________________________________________________________________________
block2d_se_squeeze (GlobalAvera (None, 192)          0           block2d_activation[0][0]
__________________________________________________________________________________________________
block2d_se_reshape (Reshape)    (None, 1, 1, 192)    0           block2d_se_squeeze[0][0]
__________________________________________________________________________________________________
block2d_se_reduce (Conv2D)      (None, 1, 1, 8)      1544        block2d_se_reshape[0][0]
__________________________________________________________________________________________________
block2d_se_expand (Conv2D)      (None, 1, 1, 192)    1728        block2d_se_reduce[0][0]
__________________________________________________________________________________________________
block2d_se_excite (Multiply)    (None, 64, 64, 192)  0           block2d_activation[0][0]
                                                                 block2d_se_expand[0][0]
__________________________________________________________________________________________________
block2d_project_conv (Conv2D)   (None, 64, 64, 32)   6144        block2d_se_excite[0][0]
__________________________________________________________________________________________________
```

```
block2d_project_bn (BatchNormal (None, 64, 64, 32)   128         block2d_project_conv[0][0]

__________________________________________________________________________________________________
block2d_drop (Dropout)          (None, 64, 64, 32)   0           block2d_project_bn[0][0]

__________________________________________________________________________________________________
block2d_add (Add)               (None, 64, 64, 32)   0           block2d_drop[0][0]
                                                                 block2c_add[0][0]

__________________________________________________________________________________________________
block3a_expand_conv (Conv2D)    (None, 64, 64, 192)  6144        block2d_add[0][0]

__________________________________________________________________________________________________
block3a_expand_bn (BatchNormali (None, 64, 64, 192)  768         block3a_expand_conv[0][0]

__________________________________________________________________________________________________
block3a_expand_activation (Acti (None, 64, 64, 192)  0           block3a_expand_bn[0][0]

__________________________________________________________________________________________________
block3a_dwconv_pad (ZeroPadding (None, 67, 67, 192)  0           block3a_expand_activation[0][0]

__________________________________________________________________________________________________
block3a_dwconv (DepthwiseConv2D (None, 32, 32, 192)  4800        block3a_dwconv_pad[0][0]

__________________________________________________________________________________________________
block3a_bn (BatchNormalization) (None, 32, 32, 192)  768         block3a_dwconv[0][0]

__________________________________________________________________________________________________
block3a_activation (Activation) (None, 32, 32, 192)  0           block3a_bn[0][0]

__________________________________________________________________________________________________
block3a_se_squeeze (GlobalAvera (None, 192)          0           block3a_activation[0][0]

__________________________________________________________________________________________________
block3a_se_reshape (Reshape)    (None, 1, 1, 192)    0           block3a_se_squeeze[0][0]

__________________________________________________________________________________________________
block3a_se_reduce (Conv2D)      (None, 1, 1, 8)      1544        block3a_se_reshape[0][0]

__________________________________________________________________________________________________
block3a_se_expand (Conv2D)      (None, 1, 1, 192)    1728        block3a_se_reduce[0][0]

__________________________________________________________________________________________________
block3a_se_excite (Multiply)    (None, 32, 32, 192)  0           block3a_activation[0][0]
                                                                 block3a_se_expand[0][0]

__________________________________________________________________________________________________
block3a_project_conv (Conv2D)   (None, 32, 32, 56)   10752       block3a_se_excite[0][0]

__________________________________________________________________________________________________
block3a_project_bn (BatchNormal (None, 32, 32, 56)   224         block3a_project_conv[0][0]

__________________________________________________________________________________________________
block3b_expand_conv (Conv2D)    (None, 32, 32, 336)  18816       block3a_project_bn[0][0]
```

```
__________________________________________________________________________________________________
block3b_expand_bn (BatchNormali (None, 32, 32, 336)  1344        block3b_expand_conv[0][0]
__________________________________________________________________________________________________
block3b_expand_activation (Acti (None, 32, 32, 336)  0           block3b_expand_bn[0][0]
__________________________________________________________________________________________________
block3b_dwconv (DepthwiseConv2D (None, 32, 32, 336)  8400        block3b_expand_activation[0][0]
__________________________________________________________________________________________________
block3b_bn (BatchNormalization) (None, 32, 32, 336)  1344        block3b_dwconv[0][0]
__________________________________________________________________________________________________
block3b_activation (Activation) (None, 32, 32, 336)  0           block3b_bn[0][0]
__________________________________________________________________________________________________
block3b_se_squeeze (GlobalAvera (None, 336)          0           block3b_activation[0][0]
__________________________________________________________________________________________________
block3b_se_reshape (Reshape)    (None, 1, 1, 336)    0           block3b_se_squeeze[0][0]
__________________________________________________________________________________________________
block3b_se_reduce (Conv2D)      (None, 1, 1, 14)     4718        block3b_se_reshape[0][0]
__________________________________________________________________________________________________
block3b_se_expand (Conv2D)      (None, 1, 1, 336)    5040        block3b_se_reduce[0][0]
__________________________________________________________________________________________________
block3b_se_excite (Multiply)    (None, 32, 32, 336)  0           block3b_activation[0][0]
                                                                 block3b_se_expand[0][0]
__________________________________________________________________________________________________
block3b_project_conv (Conv2D)   (None, 32, 32, 56)   18816       block3b_se_excite[0][0]
__________________________________________________________________________________________________
block3b_project_bn (BatchNormal (None, 32, 32, 56)   224         block3b_project_conv[0][0]
__________________________________________________________________________________________________
block3b_drop (Dropout)          (None, 32, 32, 56)   0           block3b_project_bn[0][0]
__________________________________________________________________________________________________
block3b_add (Add)               (None, 32, 32, 56)   0           block3b_drop[0][0]
                                                                 block3a_project_bn[0][0]
__________________________________________________________________________________________________
block3c_expand_conv (Conv2D)    (None, 32, 32, 336)  18816       block3b_add[0][0]
__________________________________________________________________________________________________
block3c_expand_bn (BatchNormali (None, 32, 32, 336)  1344        block3c_expand_conv[0][0]
__________________________________________________________________________________________________
block3c_expand_activation (Acti (None, 32, 32, 336)  0           block3c_expand_bn[0][0]
__________________________________________________________________________________________________
```

```
block3c_dwconv (DepthwiseConv2D (None, 32, 32, 336)  8400        block3c_expand_activation[0][0]

__________________________________________________________________________________________________
block3c_bn (BatchNormalization) (None, 32, 32, 336)  1344        block3c_dwconv[0][0]

__________________________________________________________________________________________________
block3c_activation (Activation) (None, 32, 32, 336)  0           block3c_bn[0][0]

__________________________________________________________________________________________________
block3c_se_squeeze (GlobalAvera (None, 336)          0           block3c_activation[0][0]

__________________________________________________________________________________________________
block3c_se_reshape (Reshape)    (None, 1, 1, 336)    0           block3c_se_squeeze[0][0]

__________________________________________________________________________________________________
block3c_se_reduce (Conv2D)      (None, 1, 1, 14)     4718        block3c_se_reshape[0][0]

__________________________________________________________________________________________________
block3c_se_expand (Conv2D)      (None, 1, 1, 336)    5040        block3c_se_reduce[0][0]

__________________________________________________________________________________________________
block3c_se_excite (Multiply)    (None, 32, 32, 336)  0           block3c_activation[0][0]

                                                                 block3c_se_expand[0][0]

__________________________________________________________________________________________________
block3c_project_conv (Conv2D)   (None, 32, 32, 56)   18816       block3c_se_excite[0][0]

__________________________________________________________________________________________________
block3c_project_bn (BatchNormal (None, 32, 32, 56)   224         block3c_project_conv[0][0]

__________________________________________________________________________________________________
block3c_drop (Dropout)          (None, 32, 32, 56)   0           block3c_project_bn[0][0]

__________________________________________________________________________________________________
block3c_add (Add)               (None, 32, 32, 56)   0           block3c_drop[0][0]

                                                                 block3b_add[0][0]

__________________________________________________________________________________________________
block3d_expand_conv (Conv2D)    (None, 32, 32, 336)  18816       block3c_add[0][0]

__________________________________________________________________________________________________
block3d_expand_bn (BatchNormali (None, 32, 32, 336)  1344        block3d_expand_conv[0][0]

__________________________________________________________________________________________________
block3d_expand_activation (Acti (None, 32, 32, 336)  0           block3d_expand_bn[0][0]

__________________________________________________________________________________________________
block3d_dwconv (DepthwiseConv2D (None, 32, 32, 336)  8400        block3d_expand_activation[0][0]

__________________________________________________________________________________________________
block3d_bn (BatchNormalization) (None, 32, 32, 336)  1344        block3d_dwconv[0][0]

__________________________________________________________________________________________________
block3d_activation (Activation) (None, 32, 32, 336)  0           block3d_bn[0][0]
```

```
__________________________________________________________________________________________________
block3d_se_squeeze (GlobalAvera (None, 336)          0           block3d_activation[0][0]
__________________________________________________________________________________________________
block3d_se_reshape (Reshape)    (None, 1, 1, 336)    0           block3d_se_squeeze[0][0]
__________________________________________________________________________________________________
block3d_se_reduce (Conv2D)      (None, 1, 1, 14)     4718        block3d_se_reshape[0][0]
__________________________________________________________________________________________________
block3d_se_expand (Conv2D)      (None, 1, 1, 336)    5040        block3d_se_reduce[0][0]
__________________________________________________________________________________________________
block3d_se_excite (Multiply)    (None, 32, 32, 336)  0           block3d_activation[0][0]
                                                                 block3d_se_expand[0][0]
__________________________________________________________________________________________________
block3d_project_conv (Conv2D)   (None, 32, 32, 56)   18816       block3d_se_excite[0][0]
__________________________________________________________________________________________________
block3d_project_bn (BatchNormal (None, 32, 32, 56)   224         block3d_project_conv[0][0]
__________________________________________________________________________________________________
block3d_drop (Dropout)          (None, 32, 32, 56)   0           block3d_project_bn[0][0]
__________________________________________________________________________________________________
block3d_add (Add)               (None, 32, 32, 56)   0           block3d_drop[0][0]
                                                                 block3c_add[0][0]
__________________________________________________________________________________________________
block4a_expand_conv (Conv2D)    (None, 32, 32, 336)  18816       block3d_add[0][0]
__________________________________________________________________________________________________
block4a_expand_bn (BatchNormali (None, 32, 32, 336)  1344        block4a_expand_conv[0][0]
__________________________________________________________________________________________________
block4a_expand_activation (Acti (None, 32, 32, 336)  0           block4a_expand_bn[0][0]
__________________________________________________________________________________________________
block4a_dwconv_pad (ZeroPadding (None, 33, 33, 336)  0           block4a_expand_activation[0][0]
__________________________________________________________________________________________________
block4a_dwconv (DepthwiseConv2D (None, 16, 16, 336)  3024        block4a_dwconv_pad[0][0]
__________________________________________________________________________________________________
block4a_bn (BatchNormalization) (None, 16, 16, 336)  1344        block4a_dwconv[0][0]
__________________________________________________________________________________________________
block4a_activation (Activation) (None, 16, 16, 336)  0           block4a_bn[0][0]
__________________________________________________________________________________________________
block4a_se_squeeze (GlobalAvera (None, 336)          0           block4a_activation[0][0]
__________________________________________________________________________________________________
```

```
block4a_se_reshape (Reshape)    (None, 1, 1, 336)    0           block4a_se_squeeze[0][0]

__________________________________________________________________________________________________
block4a_se_reduce (Conv2D)      (None, 1, 1, 14)     4718        block4a_se_reshape[0][0]

__________________________________________________________________________________________________
block4a_se_expand (Conv2D)      (None, 1, 1, 336)    5040        block4a_se_reduce[0][0]

__________________________________________________________________________________________________
block4a_se_excite (Multiply)    (None, 16, 16, 336)  0           block4a_activation[0][0]

                                                                 block4a_se_expand[0][0]

__________________________________________________________________________________________________
block4a_project_conv (Conv2D)   (None, 16, 16, 112)  37632       block4a_se_excite[0][0]

__________________________________________________________________________________________________
block4a_project_bn (BatchNormal (None, 16, 16, 112)  448         block4a_project_conv[0][0]

__________________________________________________________________________________________________
block4b_expand_conv (Conv2D)    (None, 16, 16, 672)  75264       block4a_project_bn[0][0]

__________________________________________________________________________________________________
block4b_expand_bn (BatchNormali (None, 16, 16, 672)  2688        block4b_expand_conv[0][0]

__________________________________________________________________________________________________
block4b_expand_activation (Acti (None, 16, 16, 672)  0           block4b_expand_bn[0][0]

__________________________________________________________________________________________________
block4b_dwconv (DepthwiseConv2D (None, 16, 16, 672)  6048        block4b_expand_activation[0][0]

__________________________________________________________________________________________________
block4b_bn (BatchNormalization) (None, 16, 16, 672)  2688        block4b_dwconv[0][0]

__________________________________________________________________________________________________
block4b_activation (Activation) (None, 16, 16, 672)  0           block4b_bn[0][0]

__________________________________________________________________________________________________
block4b_se_squeeze (GlobalAvera (None, 672)          0           block4b_activation[0][0]

__________________________________________________________________________________________________
block4b_se_reshape (Reshape)    (None, 1, 1, 672)    0           block4b_se_squeeze[0][0]

__________________________________________________________________________________________________
block4b_se_reduce (Conv2D)      (None, 1, 1, 28)     18844       block4b_se_reshape[0][0]

__________________________________________________________________________________________________
block4b_se_expand (Conv2D)      (None, 1, 1, 672)    19488       block4b_se_reduce[0][0]

__________________________________________________________________________________________________
block4b_se_excite (Multiply)    (None, 16, 16, 672)  0           block4b_activation[0][0]

                                                                 block4b_se_expand[0][0]

__________________________________________________________________________________________________
block4b_project_conv (Conv2D)   (None, 16, 16, 112)  75264       block4b_se_excite[0][0]
```

```
--------------------------------------------------------------------------------------------------
block4b_project_bn (BatchNormal (None, 16, 16, 112)  448         block4b_project_conv[0][0]

--------------------------------------------------------------------------------------------------
block4b_drop (Dropout)          (None, 16, 16, 112)  0           block4b_project_bn[0][0]

--------------------------------------------------------------------------------------------------
block4b_add (Add)               (None, 16, 16, 112)  0           block4b_drop[0][0]

                                                                 block4a_project_bn[0][0]

--------------------------------------------------------------------------------------------------
block4c_expand_conv (Conv2D)    (None, 16, 16, 672)  75264       block4b_add[0][0]

--------------------------------------------------------------------------------------------------
block4c_expand_bn (BatchNormali (None, 16, 16, 672)  2688        block4c_expand_conv[0][0]

--------------------------------------------------------------------------------------------------
block4c_expand_activation (Acti (None, 16, 16, 672)  0           block4c_expand_bn[0][0]

--------------------------------------------------------------------------------------------------
block4c_dwconv (DepthwiseConv2D (None, 16, 16, 672)  6048        block4c_expand_activation[0][0]

--------------------------------------------------------------------------------------------------
block4c_bn (BatchNormalization) (None, 16, 16, 672)  2688        block4c_dwconv[0][0]

--------------------------------------------------------------------------------------------------
block4c_activation (Activation) (None, 16, 16, 672)  0           block4c_bn[0][0]

--------------------------------------------------------------------------------------------------
block4c_se_squeeze (GlobalAvera (None, 672)          0           block4c_activation[0][0]

--------------------------------------------------------------------------------------------------
block4c_se_reshape (Reshape)    (None, 1, 1, 672)    0           block4c_se_squeeze[0][0]

--------------------------------------------------------------------------------------------------
block4c_se_reduce (Conv2D)      (None, 1, 1, 28)     18844       block4c_se_reshape[0][0]

--------------------------------------------------------------------------------------------------
block4c_se_expand (Conv2D)      (None, 1, 1, 672)    19488       block4c_se_reduce[0][0]

--------------------------------------------------------------------------------------------------
block4c_se_excite (Multiply)    (None, 16, 16, 672)  0           block4c_activation[0][0]

                                                                 block4c_se_expand[0][0]

--------------------------------------------------------------------------------------------------
block4c_project_conv (Conv2D)   (None, 16, 16, 112)  75264       block4c_se_excite[0][0]

--------------------------------------------------------------------------------------------------
block4c_project_bn (BatchNormal (None, 16, 16, 112)  448         block4c_project_conv[0][0]

--------------------------------------------------------------------------------------------------
block4c_drop (Dropout)          (None, 16, 16, 112)  0           block4c_project_bn[0][0]

--------------------------------------------------------------------------------------------------
```

```
block4c_add (Add)              (None, 16, 16, 112)  0           block4c_drop[0][0]

                                                                block4b_add[0][0]

__________________________________________________________________________________________________
block4d_expand_conv (Conv2D)    (None, 16, 16, 672)  75264       block4c_add[0][0]

__________________________________________________________________________________________________
block4d_expand_bn (BatchNormali (None, 16, 16, 672)  2688        block4d_expand_conv[0][0]

__________________________________________________________________________________________________
block4d_expand_activation (Acti (None, 16, 16, 672)  0           block4d_expand_bn[0][0]

__________________________________________________________________________________________________
block4d_dwconv (DepthwiseConv2D (None, 16, 16, 672)  6048        block4d_expand_activation[0][0]

__________________________________________________________________________________________________
block4d_bn (BatchNormalization) (None, 16, 16, 672)  2688        block4d_dwconv[0][0]

__________________________________________________________________________________________________
block4d_activation (Activation) (None, 16, 16, 672)  0           block4d_bn[0][0]

__________________________________________________________________________________________________
block4d_se_squeeze (GlobalAvera (None, 672)          0           block4d_activation[0][0]

__________________________________________________________________________________________________
block4d_se_reshape (Reshape)    (None, 1, 1, 672)    0           block4d_se_squeeze[0][0]

__________________________________________________________________________________________________
block4d_se_reduce (Conv2D)      (None, 1, 1, 28)     18844       block4d_se_reshape[0][0]

__________________________________________________________________________________________________
block4d_se_expand (Conv2D)      (None, 1, 1, 672)    19488       block4d_se_reduce[0][0]

__________________________________________________________________________________________________
block4d_se_excite (Multiply)    (None, 16, 16, 672)  0           block4d_activation[0][0]

                                                                block4d_se_expand[0][0]

__________________________________________________________________________________________________
block4d_project_conv (Conv2D)   (None, 16, 16, 112)  75264       block4d_se_excite[0][0]

__________________________________________________________________________________________________
block4d_project_bn (BatchNormal (None, 16, 16, 112)  448         block4d_project_conv[0][0]

__________________________________________________________________________________________________
block4d_drop (Dropout)          (None, 16, 16, 112)  0           block4d_project_bn[0][0]

__________________________________________________________________________________________________
block4d_add (Add)               (None, 16, 16, 112)  0           block4d_drop[0][0]

                                                                block4c_add[0][0]

__________________________________________________________________________________________________
block4e_expand_conv (Conv2D)    (None, 16, 16, 672)  75264       block4d_add[0][0]

__________________________________________________________________________________________________
```

```
block4e_expand_bn (BatchNormali (None, 16, 16, 672)  2688        block4e_expand_conv[0][0]
__________________________________________________________________________________________________
block4e_expand_activation (Acti (None, 16, 16, 672)  0           block4e_expand_bn[0][0]
__________________________________________________________________________________________________
block4e_dwconv (DepthwiseConv2D (None, 16, 16, 672)  6048        block4e_expand_activation[0][0]
__________________________________________________________________________________________________
block4e_bn (BatchNormalization) (None, 16, 16, 672)  2688        block4e_dwconv[0][0]
__________________________________________________________________________________________________
block4e_activation (Activation) (None, 16, 16, 672)  0           block4e_bn[0][0]
__________________________________________________________________________________________________
block4e_se_squeeze (GlobalAvera (None, 672)          0           block4e_activation[0][0]
__________________________________________________________________________________________________
block4e_se_reshape (Reshape)    (None, 1, 1, 672)    0           block4e_se_squeeze[0][0]
__________________________________________________________________________________________________
block4e_se_reduce (Conv2D)      (None, 1, 1, 28)     18844       block4e_se_reshape[0][0]
__________________________________________________________________________________________________
block4e_se_expand (Conv2D)      (None, 1, 1, 672)    19488       block4e_se_reduce[0][0]
__________________________________________________________________________________________________
block4e_se_excite (Multiply)    (None, 16, 16, 672)  0           block4e_activation[0][0]
                                                                 block4e_se_expand[0][0]
__________________________________________________________________________________________________
block4e_project_conv (Conv2D)   (None, 16, 16, 112)  75264       block4e_se_excite[0][0]
__________________________________________________________________________________________________
block4e_project_bn (BatchNormal (None, 16, 16, 112)  448         block4e_project_conv[0][0]
__________________________________________________________________________________________________
block4e_drop (Dropout)          (None, 16, 16, 112)  0           block4e_project_bn[0][0]
__________________________________________________________________________________________________
block4e_add (Add)               (None, 16, 16, 112)  0           block4e_drop[0][0]
                                                                 block4d_add[0][0]
__________________________________________________________________________________________________
block4f_expand_conv (Conv2D)    (None, 16, 16, 672)  75264       block4e_add[0][0]
__________________________________________________________________________________________________
block4f_expand_bn (BatchNormali (None, 16, 16, 672)  2688        block4f_expand_conv[0][0]
__________________________________________________________________________________________________
block4f_expand_activation (Acti (None, 16, 16, 672)  0           block4f_expand_bn[0][0]
__________________________________________________________________________________________________
block4f_dwconv (DepthwiseConv2D (None, 16, 16, 672)  6048        block4f_expand_activation[0][0]
__________________________________________________________________________________________________
block4f_bn (BatchNormalization) (None, 16, 16, 672)  2688        block4f_dwconv[0][0]

__________________________________________________________________________________________________
block4f_activation (Activation) (None, 16, 16, 672)  0           block4f_bn[0][0]

__________________________________________________________________________________________________
block4f_se_squeeze (GlobalAvera (None, 672)          0           block4f_activation[0][0]

__________________________________________________________________________________________________
block4f_se_reshape (Reshape)    (None, 1, 1, 672)    0           block4f_se_squeeze[0][0]

__________________________________________________________________________________________________
block4f_se_reduce (Conv2D)      (None, 1, 1, 28)     18844       block4f_se_reshape[0][0]

__________________________________________________________________________________________________
block4f_se_expand (Conv2D)      (None, 1, 1, 672)    19488       block4f_se_reduce[0][0]

__________________________________________________________________________________________________
block4f_se_excite (Multiply)    (None, 16, 16, 672)  0           block4f_activation[0][0]

                                                                 block4f_se_expand[0][0]

__________________________________________________________________________________________________
block4f_project_conv (Conv2D)   (None, 16, 16, 112)  75264       block4f_se_excite[0][0]

__________________________________________________________________________________________________
block4f_project_bn (BatchNormal (None, 16, 16, 112)  448         block4f_project_conv[0][0]

__________________________________________________________________________________________________
block4f_drop (Dropout)          (None, 16, 16, 112)  0           block4f_project_bn[0][0]

__________________________________________________________________________________________________
block4f_add (Add)               (None, 16, 16, 112)  0           block4f_drop[0][0]

                                                                 block4e_add[0][0]

__________________________________________________________________________________________________
block5a_expand_conv (Conv2D)    (None, 16, 16, 672)  75264       block4f_add[0][0]

__________________________________________________________________________________________________
block5a_expand_bn (BatchNormali (None, 16, 16, 672)  2688        block5a_expand_conv[0][0]

__________________________________________________________________________________________________
block5a_expand_activation (Acti (None, 16, 16, 672)  0           block5a_expand_bn[0][0]

__________________________________________________________________________________________________
block5a_dwconv (DepthwiseConv2D (None, 16, 16, 672)  16800       block5a_expand_activation[0][0]

__________________________________________________________________________________________________
block5a_bn (BatchNormalization) (None, 16, 16, 672)  2688        block5a_dwconv[0][0]

__________________________________________________________________________________________________
block5a_activation (Activation) (None, 16, 16, 672)  0           block5a_bn[0][0]

__________________________________________________________________________________________________
```

```
block5a_se_squeeze (GlobalAvera (None, 672)         0           block5a_activation[0][0]

__________________________________________________________________________________________________
block5a_se_reshape (Reshape)    (None, 1, 1, 672)   0           block5a_se_squeeze[0][0]

__________________________________________________________________________________________________
block5a_se_reduce (Conv2D)      (None, 1, 1, 28)    18844       block5a_se_reshape[0][0]

__________________________________________________________________________________________________
block5a_se_expand (Conv2D)      (None, 1, 1, 672)   19488       block5a_se_reduce[0][0]

__________________________________________________________________________________________________
block5a_se_excite (Multiply)    (None, 16, 16, 672) 0           block5a_activation[0][0]

                                                                block5a_se_expand[0][0]

__________________________________________________________________________________________________
block5a_project_conv (Conv2D)   (None, 16, 16, 160) 107520      block5a_se_excite[0][0]

__________________________________________________________________________________________________
block5a_project_bn (BatchNormal (None, 16, 16, 160) 640         block5a_project_conv[0][0]

__________________________________________________________________________________________________
block5b_expand_conv (Conv2D)    (None, 16, 16, 960) 153600      block5a_project_bn[0][0]

__________________________________________________________________________________________________
block5b_expand_bn (BatchNormali (None, 16, 16, 960) 3840        block5b_expand_conv[0][0]

__________________________________________________________________________________________________
block5b_expand_activation (Acti (None, 16, 16, 960) 0           block5b_expand_bn[0][0]

__________________________________________________________________________________________________
block5b_dwconv (DepthwiseConv2D (None, 16, 16, 960) 24000       block5b_expand_activation[0][0]

__________________________________________________________________________________________________
block5b_bn (BatchNormalization) (None, 16, 16, 960) 3840        block5b_dwconv[0][0]

__________________________________________________________________________________________________
block5b_activation (Activation) (None, 16, 16, 960) 0           block5b_bn[0][0]

__________________________________________________________________________________________________
block5b_se_squeeze (GlobalAvera (None, 960)         0           block5b_activation[0][0]

__________________________________________________________________________________________________
block5b_se_reshape (Reshape)    (None, 1, 1, 960)   0           block5b_se_squeeze[0][0]

__________________________________________________________________________________________________
block5b_se_reduce (Conv2D)      (None, 1, 1, 40)    38440       block5b_se_reshape[0][0]

__________________________________________________________________________________________________
block5b_se_expand (Conv2D)      (None, 1, 1, 960)   39360       block5b_se_reduce[0][0]

__________________________________________________________________________________________________
block5b_se_excite (Multiply)    (None, 16, 16, 960) 0           block5b_activation[0][0]

                                                                block5b_se_expand[0][0]
```

```
__________________________________________________________________________________________________
block5b_project_conv (Conv2D)   (None, 16, 16, 160)  153600      block5b_se_excite[0][0]
__________________________________________________________________________________________________
block5b_project_bn (BatchNormal (None, 16, 16, 160)  640         block5b_project_conv[0][0]
__________________________________________________________________________________________________
block5b_drop (Dropout)          (None, 16, 16, 160)  0           block5b_project_bn[0][0]
__________________________________________________________________________________________________
block5b_add (Add)               (None, 16, 16, 160)  0           block5b_drop[0][0]
                                                                 block5a_project_bn[0][0]
__________________________________________________________________________________________________
block5c_expand_conv (Conv2D)    (None, 16, 16, 960)  153600      block5b_add[0][0]
__________________________________________________________________________________________________
block5c_expand_bn (BatchNormali (None, 16, 16, 960)  3840        block5c_expand_conv[0][0]
__________________________________________________________________________________________________
block5c_expand_activation (Acti (None, 16, 16, 960)  0           block5c_expand_bn[0][0]
__________________________________________________________________________________________________
block5c_dwconv (DepthwiseConv2D (None, 16, 16, 960)  24000       block5c_expand_activation[0][0]
__________________________________________________________________________________________________
block5c_bn (BatchNormalization) (None, 16, 16, 960)  3840        block5c_dwconv[0][0]
__________________________________________________________________________________________________
block5c_activation (Activation) (None, 16, 16, 960)  0           block5c_bn[0][0]
__________________________________________________________________________________________________
block5c_se_squeeze (GlobalAvera (None, 960)          0           block5c_activation[0][0]
__________________________________________________________________________________________________
block5c_se_reshape (Reshape)    (None, 1, 1, 960)    0           block5c_se_squeeze[0][0]
__________________________________________________________________________________________________
block5c_se_reduce (Conv2D)      (None, 1, 1, 40)     38440       block5c_se_reshape[0][0]
__________________________________________________________________________________________________
block5c_se_expand (Conv2D)      (None, 1, 1, 960)    39360       block5c_se_reduce[0][0]
__________________________________________________________________________________________________
block5c_se_excite (Multiply)    (None, 16, 16, 960)  0           block5c_activation[0][0]
                                                                 block5c_se_expand[0][0]
__________________________________________________________________________________________________
block5c_project_conv (Conv2D)   (None, 16, 16, 160)  153600      block5c_se_excite[0][0]
__________________________________________________________________________________________________
block5c_project_bn (BatchNormal (None, 16, 16, 160)  640         block5c_project_conv[0][0]
__________________________________________________________________________________________________
```

```
block5c_drop (Dropout)          (None, 16, 16, 160)  0           block5c_project_bn[0][0]

__________________________________________________________________________________________________
block5c_add (Add)               (None, 16, 16, 160)  0           block5c_drop[0][0]

                                                                 block5b_add[0][0]

__________________________________________________________________________________________________
block5d_expand_conv (Conv2D)    (None, 16, 16, 960)  153600      block5c_add[0][0]

__________________________________________________________________________________________________
block5d_expand_bn (BatchNormali (None, 16, 16, 960)  3840        block5d_expand_conv[0][0]

__________________________________________________________________________________________________
block5d_expand_activation (Acti (None, 16, 16, 960)  0           block5d_expand_bn[0][0]

__________________________________________________________________________________________________
block5d_dwconv (DepthwiseConv2D (None, 16, 16, 960)  24000       block5d_expand_activation[0][0]

__________________________________________________________________________________________________
block5d_bn (BatchNormalization) (None, 16, 16, 960)  3840        block5d_dwconv[0][0]

__________________________________________________________________________________________________
block5d_activation (Activation) (None, 16, 16, 960)  0           block5d_bn[0][0]

__________________________________________________________________________________________________
block5d_se_squeeze (GlobalAvera (None, 960)          0           block5d_activation[0][0]

__________________________________________________________________________________________________
block5d_se_reshape (Reshape)    (None, 1, 1, 960)    0           block5d_se_squeeze[0][0]

__________________________________________________________________________________________________
block5d_se_reduce (Conv2D)      (None, 1, 1, 40)     38440       block5d_se_reshape[0][0]

__________________________________________________________________________________________________
block5d_se_expand (Conv2D)      (None, 1, 1, 960)    39360       block5d_se_reduce[0][0]

__________________________________________________________________________________________________
block5d_se_excite (Multiply)    (None, 16, 16, 960)  0           block5d_activation[0][0]

                                                                 block5d_se_expand[0][0]

__________________________________________________________________________________________________
block5d_project_conv (Conv2D)   (None, 16, 16, 160)  153600      block5d_se_excite[0][0]

__________________________________________________________________________________________________
block5d_project_bn (BatchNormal (None, 16, 16, 160)  640         block5d_project_conv[0][0]

__________________________________________________________________________________________________
block5d_drop (Dropout)          (None, 16, 16, 160)  0           block5d_project_bn[0][0]

__________________________________________________________________________________________________
block5d_add (Add)               (None, 16, 16, 160)  0           block5d_drop[0][0]

                                                                 block5c_add[0][0]

__________________________________________________________________________________________________
```

```
block5e_expand_conv (Conv2D)    (None, 16, 16, 960)  153600      block5d_add[0][0]

__________________________________________________________________________________________________
block5e_expand_bn (BatchNormali (None, 16, 16, 960)  3840        block5e_expand_conv[0][0]

__________________________________________________________________________________________________
block5e_expand_activation (Acti (None, 16, 16, 960)  0           block5e_expand_bn[0][0]

__________________________________________________________________________________________________
block5e_dwconv (DepthwiseConv2D (None, 16, 16, 960)  24000       block5e_expand_activation[0][0]

__________________________________________________________________________________________________
block5e_bn (BatchNormalization) (None, 16, 16, 960)  3840        block5e_dwconv[0][0]

__________________________________________________________________________________________________
block5e_activation (Activation) (None, 16, 16, 960)  0           block5e_bn[0][0]

__________________________________________________________________________________________________
block5e_se_squeeze (GlobalAvera (None, 960)          0           block5e_activation[0][0]

__________________________________________________________________________________________________
block5e_se_reshape (Reshape)    (None, 1, 1, 960)    0           block5e_se_squeeze[0][0]

__________________________________________________________________________________________________
block5e_se_reduce (Conv2D)      (None, 1, 1, 40)     38440       block5e_se_reshape[0][0]

__________________________________________________________________________________________________
block5e_se_expand (Conv2D)      (None, 1, 1, 960)    39360       block5e_se_reduce[0][0]

__________________________________________________________________________________________________
block5e_se_excite (Multiply)    (None, 16, 16, 960)  0           block5e_activation[0][0]

                                                                 block5e_se_expand[0][0]

__________________________________________________________________________________________________
block5e_project_conv (Conv2D)   (None, 16, 16, 160)  153600      block5e_se_excite[0][0]

__________________________________________________________________________________________________
block5e_project_bn (BatchNormal (None, 16, 16, 160)  640         block5e_project_conv[0][0]

__________________________________________________________________________________________________
block5e_drop (Dropout)          (None, 16, 16, 160)  0           block5e_project_bn[0][0]

__________________________________________________________________________________________________
block5e_add (Add)               (None, 16, 16, 160)  0           block5e_drop[0][0]

                                                                 block5d_add[0][0]

__________________________________________________________________________________________________
block5f_expand_conv (Conv2D)    (None, 16, 16, 960)  153600      block5e_add[0][0]

__________________________________________________________________________________________________
block5f_expand_bn (BatchNormali (None, 16, 16, 960)  3840        block5f_expand_conv[0][0]

__________________________________________________________________________________________________
block5f_expand_activation (Acti (None, 16, 16, 960)  0           block5f_expand_bn[0][0]
__________________________________________________________________________________________________
block5f_dwconv (DepthwiseConv2D (None, 16, 16, 960)  24000       block5f_expand_activation[0][0]
__________________________________________________________________________________________________
block5f_bn (BatchNormalization) (None, 16, 16, 960)  3840        block5f_dwconv[0][0]
__________________________________________________________________________________________________
block5f_activation (Activation) (None, 16, 16, 960)  0           block5f_bn[0][0]
__________________________________________________________________________________________________
block5f_se_squeeze (GlobalAvera (None, 960)          0           block5f_activation[0][0]
__________________________________________________________________________________________________
block5f_se_reshape (Reshape)    (None, 1, 1, 960)    0           block5f_se_squeeze[0][0]
__________________________________________________________________________________________________
block5f_se_reduce (Conv2D)      (None, 1, 1, 40)     38440       block5f_se_reshape[0][0]
__________________________________________________________________________________________________
block5f_se_expand (Conv2D)      (None, 1, 1, 960)    39360       block5f_se_reduce[0][0]
__________________________________________________________________________________________________
block5f_se_excite (Multiply)    (None, 16, 16, 960)  0           block5f_activation[0][0]
                                                                 block5f_se_expand[0][0]
__________________________________________________________________________________________________
block5f_project_conv (Conv2D)   (None, 16, 16, 160)  153600      block5f_se_excite[0][0]
__________________________________________________________________________________________________
block5f_project_bn (BatchNormal (None, 16, 16, 160)  640         block5f_project_conv[0][0]
__________________________________________________________________________________________________
block5f_drop (Dropout)          (None, 16, 16, 160)  0           block5f_project_bn[0][0]
__________________________________________________________________________________________________
block5f_add (Add)               (None, 16, 16, 160)  0           block5f_drop[0][0]
                                                                 block5e_add[0][0]
__________________________________________________________________________________________________
block6a_expand_conv (Conv2D)    (None, 16, 16, 960)  153600      block5f_add[0][0]
__________________________________________________________________________________________________
block6a_expand_bn (BatchNormali (None, 16, 16, 960)  3840        block6a_expand_conv[0][0]
__________________________________________________________________________________________________
block6a_expand_activation (Acti (None, 16, 16, 960)  0           block6a_expand_bn[0][0]
__________________________________________________________________________________________________
block6a_dwconv_pad (ZeroPadding (None, 19, 19, 960)  0           block6a_expand_activation[0][0]
__________________________________________________________________________________________________
block6a_dwconv (DepthwiseConv2D (None, 8, 8, 960)    24000       block6a_dwconv_pad[0][0]
__________________________________________________________________________________________________
```

```
block6a_bn (BatchNormalization) (None, 8, 8, 960)    3840        block6a_dwconv[0][0]

__________________________________________________________________________________________________
block6a_activation (Activation) (None, 8, 8, 960)    0           block6a_bn[0][0]

__________________________________________________________________________________________________
block6a_se_squeeze (GlobalAvera (None, 960)          0           block6a_activation[0][0]

__________________________________________________________________________________________________
block6a_se_reshape (Reshape)    (None, 1, 1, 960)    0           block6a_se_squeeze[0][0]

__________________________________________________________________________________________________
block6a_se_reduce (Conv2D)      (None, 1, 1, 40)     38440       block6a_se_reshape[0][0]

__________________________________________________________________________________________________
block6a_se_expand (Conv2D)      (None, 1, 1, 960)    39360       block6a_se_reduce[0][0]

__________________________________________________________________________________________________
block6a_se_excite (Multiply)    (None, 8, 8, 960)    0           block6a_activation[0][0]

                                                                 block6a_se_expand[0][0]

__________________________________________________________________________________________________
block6a_project_conv (Conv2D)   (None, 8, 8, 272)    261120      block6a_se_excite[0][0]

__________________________________________________________________________________________________
block6a_project_bn (BatchNormal (None, 8, 8, 272)    1088        block6a_project_conv[0][0]

__________________________________________________________________________________________________
block6b_expand_conv (Conv2D)    (None, 8, 8, 1632)   443904      block6a_project_bn[0][0]

__________________________________________________________________________________________________
block6b_expand_bn (BatchNormali (None, 8, 8, 1632)   6528        block6b_expand_conv[0][0]

__________________________________________________________________________________________________
block6b_expand_activation (Acti (None, 8, 8, 1632)   0           block6b_expand_bn[0][0]

__________________________________________________________________________________________________
block6b_dwconv (DepthwiseConv2D (None, 8, 8, 1632)   40800       block6b_expand_activation[0][0]

__________________________________________________________________________________________________
block6b_bn (BatchNormalization) (None, 8, 8, 1632)   6528        block6b_dwconv[0][0]

__________________________________________________________________________________________________
block6b_activation (Activation) (None, 8, 8, 1632)   0           block6b_bn[0][0]

__________________________________________________________________________________________________
block6b_se_squeeze (GlobalAvera (None, 1632)         0           block6b_activation[0][0]

__________________________________________________________________________________________________
block6b_se_reshape (Reshape)    (None, 1, 1, 1632)   0           block6b_se_squeeze[0][0]

__________________________________________________________________________________________________
block6b_se_reduce (Conv2D)      (None, 1, 1, 68)     111044      block6b_se_reshape[0][0]

__________________________________________________________________________________________________
```

```
block6b_se_expand (Conv2D)      (None, 1, 1, 1632)   112608      block6b_se_reduce[0][0]

__________________________________________________________________________________________________
block6b_se_excite (Multiply)    (None, 8, 8, 1632)   0           block6b_activation[0][0]

                                                                 block6b_se_expand[0][0]

__________________________________________________________________________________________________
block6b_project_conv (Conv2D)   (None, 8, 8, 272)    443904      block6b_se_excite[0][0]

__________________________________________________________________________________________________
block6b_project_bn (BatchNormal (None, 8, 8, 272)    1088        block6b_project_conv[0][0]

__________________________________________________________________________________________________
block6b_drop (Dropout)          (None, 8, 8, 272)    0           block6b_project_bn[0][0]

__________________________________________________________________________________________________
block6b_add (Add)               (None, 8, 8, 272)    0           block6b_drop[0][0]

                                                                 block6a_project_bn[0][0]

__________________________________________________________________________________________________
block6c_expand_conv (Conv2D)    (None, 8, 8, 1632)   443904      block6b_add[0][0]

__________________________________________________________________________________________________
block6c_expand_bn (BatchNormali (None, 8, 8, 1632)   6528        block6c_expand_conv[0][0]

__________________________________________________________________________________________________
block6c_expand_activation (Acti (None, 8, 8, 1632)   0           block6c_expand_bn[0][0]

__________________________________________________________________________________________________
block6c_dwconv (DepthwiseConv2D (None, 8, 8, 1632)   40800       block6c_expand_activation[0][0]

__________________________________________________________________________________________________
block6c_bn (BatchNormalization) (None, 8, 8, 1632)   6528        block6c_dwconv[0][0]

__________________________________________________________________________________________________
block6c_activation (Activation) (None, 8, 8, 1632)   0           block6c_bn[0][0]

__________________________________________________________________________________________________
block6c_se_squeeze (GlobalAvera (None, 1632)         0           block6c_activation[0][0]

__________________________________________________________________________________________________
block6c_se_reshape (Reshape)    (None, 1, 1, 1632)   0           block6c_se_squeeze[0][0]

__________________________________________________________________________________________________
block6c_se_reduce (Conv2D)      (None, 1, 1, 68)     111044      block6c_se_reshape[0][0]

__________________________________________________________________________________________________
block6c_se_expand (Conv2D)      (None, 1, 1, 1632)   112608      block6c_se_reduce[0][0]

__________________________________________________________________________________________________
block6c_se_excite (Multiply)    (None, 8, 8, 1632)   0           block6c_activation[0][0]

                                                                 block6c_se_expand[0][0]

__________________________________________________________________________________________________
```

```
block6c_project_conv (Conv2D)   (None, 8, 8, 272)    443904      block6c_se_excite[0][0]

________________________________________________________________________________________________
block6c_project_bn (BatchNormal (None, 8, 8, 272)    1088        block6c_project_conv[0][0]

________________________________________________________________________________________________
block6c_drop (Dropout)          (None, 8, 8, 272)    0           block6c_project_bn[0][0]

________________________________________________________________________________________________
block6c_add (Add)               (None, 8, 8, 272)    0           block6c_drop[0][0]

                                                                 block6b_add[0][0]

________________________________________________________________________________________________
block6d_expand_conv (Conv2D)    (None, 8, 8, 1632)   443904      block6c_add[0][0]

________________________________________________________________________________________________
block6d_expand_bn (BatchNormali (None, 8, 8, 1632)   6528        block6d_expand_conv[0][0]

________________________________________________________________________________________________
block6d_expand_activation (Acti (None, 8, 8, 1632)   0           block6d_expand_bn[0][0]

________________________________________________________________________________________________
block6d_dwconv (DepthwiseConv2D (None, 8, 8, 1632)   40800       block6d_expand_activation[0][0]

________________________________________________________________________________________________
block6d_bn (BatchNormalization) (None, 8, 8, 1632)   6528        block6d_dwconv[0][0]

________________________________________________________________________________________________
block6d_activation (Activation) (None, 8, 8, 1632)   0           block6d_bn[0][0]

________________________________________________________________________________________________
block6d_se_squeeze (GlobalAvera (None, 1632)         0           block6d_activation[0][0]

________________________________________________________________________________________________
block6d_se_reshape (Reshape)    (None, 1, 1, 1632)   0           block6d_se_squeeze[0][0]

________________________________________________________________________________________________
block6d_se_reduce (Conv2D)      (None, 1, 1, 68)     111044      block6d_se_reshape[0][0]

________________________________________________________________________________________________
block6d_se_expand (Conv2D)      (None, 1, 1, 1632)   112608      block6d_se_reduce[0][0]

________________________________________________________________________________________________
block6d_se_excite (Multiply)    (None, 8, 8, 1632)   0           block6d_activation[0][0]

                                                                 block6d_se_expand[0][0]

________________________________________________________________________________________________
block6d_project_conv (Conv2D)   (None, 8, 8, 272)    443904      block6d_se_excite[0][0]

________________________________________________________________________________________________
block6d_project_bn (BatchNormal (None, 8, 8, 272)    1088        block6d_project_conv[0][0]

________________________________________________________________________________________________
block6d_drop (Dropout)          (None, 8, 8, 272)    0           block6d_project_bn[0][0]
```

```
____________________________________________________________________________________________________
block6d_add (Add)               (None, 8, 8, 272)    0           block6d_drop[0][0]
                                                                 block6c_add[0][0]
____________________________________________________________________________________________________
block6e_expand_conv (Conv2D)    (None, 8, 8, 1632)   443904      block6d_add[0][0]
____________________________________________________________________________________________________
block6e_expand_bn (BatchNormali (None, 8, 8, 1632)   6528        block6e_expand_conv[0][0]
____________________________________________________________________________________________________
block6e_expand_activation (Acti (None, 8, 8, 1632)   0           block6e_expand_bn[0][0]
____________________________________________________________________________________________________
block6e_dwconv (DepthwiseConv2D (None, 8, 8, 1632)   40800       block6e_expand_activation[0][0]
____________________________________________________________________________________________________
block6e_bn (BatchNormalization) (None, 8, 8, 1632)   6528        block6e_dwconv[0][0]
____________________________________________________________________________________________________
block6e_activation (Activation) (None, 8, 8, 1632)   0           block6e_bn[0][0]
____________________________________________________________________________________________________
block6e_se_squeeze (GlobalAvera (None, 1632)         0           block6e_activation[0][0]
____________________________________________________________________________________________________
block6e_se_reshape (Reshape)    (None, 1, 1, 1632)   0           block6e_se_squeeze[0][0]
____________________________________________________________________________________________________
block6e_se_reduce (Conv2D)      (None, 1, 1, 68)     111044      block6e_se_reshape[0][0]
____________________________________________________________________________________________________
block6e_se_expand (Conv2D)      (None, 1, 1, 1632)   112608      block6e_se_reduce[0][0]
____________________________________________________________________________________________________
block6e_se_excite (Multiply)    (None, 8, 8, 1632)   0           block6e_activation[0][0]
                                                                 block6e_se_expand[0][0]
____________________________________________________________________________________________________
block6e_project_conv (Conv2D)   (None, 8, 8, 272)    443904      block6e_se_excite[0][0]
____________________________________________________________________________________________________
block6e_project_bn (BatchNormal (None, 8, 8, 272)    1088        block6e_project_conv[0][0]
____________________________________________________________________________________________________
block6e_drop (Dropout)          (None, 8, 8, 272)    0           block6e_project_bn[0][0]
____________________________________________________________________________________________________
block6e_add (Add)               (None, 8, 8, 272)    0           block6e_drop[0][0]
                                                                 block6d_add[0][0]
____________________________________________________________________________________________________
block6f_expand_conv (Conv2D)    (None, 8, 8, 1632)   443904      block6e_add[0][0]
```

```
__________________________________________________________________________________________________
block6f_expand_bn (BatchNormali (None, 8, 8, 1632)   6528        block6f_expand_conv[0][0]
__________________________________________________________________________________________________
block6f_expand_activation (Acti (None, 8, 8, 1632)   0           block6f_expand_bn[0][0]
__________________________________________________________________________________________________
block6f_dwconv (DepthwiseConv2D (None, 8, 8, 1632)   40800       block6f_expand_activation[0][0]
__________________________________________________________________________________________________
block6f_bn (BatchNormalization) (None, 8, 8, 1632)   6528        block6f_dwconv[0][0]
__________________________________________________________________________________________________
block6f_activation (Activation) (None, 8, 8, 1632)   0           block6f_bn[0][0]
__________________________________________________________________________________________________
block6f_se_squeeze (GlobalAvera (None, 1632)         0           block6f_activation[0][0]
__________________________________________________________________________________________________
block6f_se_reshape (Reshape)    (None, 1, 1, 1632)   0           block6f_se_squeeze[0][0]
__________________________________________________________________________________________________
block6f_se_reduce (Conv2D)      (None, 1, 1, 68)     111044      block6f_se_reshape[0][0]
__________________________________________________________________________________________________
block6f_se_expand (Conv2D)      (None, 1, 1, 1632)   112608      block6f_se_reduce[0][0]
__________________________________________________________________________________________________
block6f_se_excite (Multiply)    (None, 8, 8, 1632)   0           block6f_activation[0][0]
                                                                 block6f_se_expand[0][0]
__________________________________________________________________________________________________
block6f_project_conv (Conv2D)   (None, 8, 8, 272)    443904      block6f_se_excite[0][0]
__________________________________________________________________________________________________
block6f_project_bn (BatchNormal (None, 8, 8, 272)    1088        block6f_project_conv[0][0]
__________________________________________________________________________________________________
block6f_drop (Dropout)          (None, 8, 8, 272)    0           block6f_project_bn[0][0]
__________________________________________________________________________________________________
block6f_add (Add)               (None, 8, 8, 272)    0           block6f_drop[0][0]
                                                                 block6e_add[0][0]
__________________________________________________________________________________________________
block6g_expand_conv (Conv2D)    (None, 8, 8, 1632)   443904      block6f_add[0][0]
__________________________________________________________________________________________________
block6g_expand_bn (BatchNormali (None, 8, 8, 1632)   6528        block6g_expand_conv[0][0]
__________________________________________________________________________________________________
block6g_expand_activation (Acti (None, 8, 8, 1632)   0           block6g_expand_bn[0][0]
__________________________________________________________________________________________________
```

```
block6g_dwconv (DepthwiseConv2D (None, 8, 8, 1632)   40800       block6g_expand_activation[0][0]
__________________________________________________________________________________________________
block6g_bn (BatchNormalization) (None, 8, 8, 1632)   6528        block6g_dwconv[0][0]
__________________________________________________________________________________________________
block6g_activation (Activation) (None, 8, 8, 1632)   0           block6g_bn[0][0]
__________________________________________________________________________________________________
block6g_se_squeeze (GlobalAvera (None, 1632)         0           block6g_activation[0][0]
__________________________________________________________________________________________________
block6g_se_reshape (Reshape)    (None, 1, 1, 1632)   0           block6g_se_squeeze[0][0]
__________________________________________________________________________________________________
block6g_se_reduce (Conv2D)      (None, 1, 1, 68)     111044      block6g_se_reshape[0][0]
__________________________________________________________________________________________________
block6g_se_expand (Conv2D)      (None, 1, 1, 1632)   112608      block6g_se_reduce[0][0]
__________________________________________________________________________________________________
block6g_se_excite (Multiply)    (None, 8, 8, 1632)   0           block6g_activation[0][0]
                                                                 block6g_se_expand[0][0]
__________________________________________________________________________________________________
block6g_project_conv (Conv2D)   (None, 8, 8, 272)    443904      block6g_se_excite[0][0]
__________________________________________________________________________________________________
block6g_project_bn (BatchNormal (None, 8, 8, 272)    1088        block6g_project_conv[0][0]
__________________________________________________________________________________________________
block6g_drop (Dropout)          (None, 8, 8, 272)    0           block6g_project_bn[0][0]
__________________________________________________________________________________________________
block6g_add (Add)               (None, 8, 8, 272)    0           block6g_drop[0][0]
                                                                 block6f_add[0][0]
__________________________________________________________________________________________________
block6h_expand_conv (Conv2D)    (None, 8, 8, 1632)   443904      block6g_add[0][0]
__________________________________________________________________________________________________
block6h_expand_bn (BatchNormali (None, 8, 8, 1632)   6528        block6h_expand_conv[0][0]
__________________________________________________________________________________________________
block6h_expand_activation (Acti (None, 8, 8, 1632)   0           block6h_expand_bn[0][0]
__________________________________________________________________________________________________
block6h_dwconv (DepthwiseConv2D (None, 8, 8, 1632)   40800       block6h_expand_activation[0][0]
__________________________________________________________________________________________________
block6h_bn (BatchNormalization) (None, 8, 8, 1632)   6528        block6h_dwconv[0][0]
__________________________________________________________________________________________________
block6h_activation (Activation) (None, 8, 8, 1632)   0           block6h_bn[0][0]
```

```
__________________________________________________________________________________________________
block6h_se_squeeze (GlobalAvera (None, 1632)         0           block6h_activation[0][0]
__________________________________________________________________________________________________
block6h_se_reshape (Reshape)    (None, 1, 1, 1632)   0           block6h_se_squeeze[0][0]
__________________________________________________________________________________________________
block6h_se_reduce (Conv2D)      (None, 1, 1, 68)     111044      block6h_se_reshape[0][0]
__________________________________________________________________________________________________
block6h_se_expand (Conv2D)      (None, 1, 1, 1632)   112608      block6h_se_reduce[0][0]
__________________________________________________________________________________________________
block6h_se_excite (Multiply)    (None, 8, 8, 1632)   0           block6h_activation[0][0]
                                                                 block6h_se_expand[0][0]
__________________________________________________________________________________________________
block6h_project_conv (Conv2D)   (None, 8, 8, 272)    443904      block6h_se_excite[0][0]
__________________________________________________________________________________________________
block6h_project_bn (BatchNormal (None, 8, 8, 272)    1088        block6h_project_conv[0][0]
__________________________________________________________________________________________________
block6h_drop (Dropout)          (None, 8, 8, 272)    0           block6h_project_bn[0][0]
__________________________________________________________________________________________________
block6h_add (Add)               (None, 8, 8, 272)    0           block6h_drop[0][0]
                                                                 block6g_add[0][0]
__________________________________________________________________________________________________
block7a_expand_conv (Conv2D)    (None, 8, 8, 1632)   443904      block6h_add[0][0]
__________________________________________________________________________________________________
block7a_expand_bn (BatchNormali (None, 8, 8, 1632)   6528        block7a_expand_conv[0][0]
__________________________________________________________________________________________________
block7a_expand_activation (Acti (None, 8, 8, 1632)   0           block7a_expand_bn[0][0]
__________________________________________________________________________________________________
block7a_dwconv (DepthwiseConv2D (None, 8, 8, 1632)   14688       block7a_expand_activation[0][0]
__________________________________________________________________________________________________
block7a_bn (BatchNormalization) (None, 8, 8, 1632)   6528        block7a_dwconv[0][0]
__________________________________________________________________________________________________
block7a_activation (Activation) (None, 8, 8, 1632)   0           block7a_bn[0][0]
__________________________________________________________________________________________________
block7a_se_squeeze (GlobalAvera (None, 1632)         0           block7a_activation[0][0]
__________________________________________________________________________________________________
block7a_se_reshape (Reshape)    (None, 1, 1, 1632)   0           block7a_se_squeeze[0][0]
__________________________________________________________________________________________________
```

```
block7a_se_reduce (Conv2D)      (None, 1, 1, 68)     111044      block7a_se_reshape[0][0]

__________________________________________________________________________________________________
block7a_se_expand (Conv2D)      (None, 1, 1, 1632)   112608      block7a_se_reduce[0][0]

__________________________________________________________________________________________________
block7a_se_excite (Multiply)    (None, 8, 8, 1632)   0           block7a_activation[0][0]

                                                                 block7a_se_expand[0][0]

__________________________________________________________________________________________________
block7a_project_conv (Conv2D)   (None, 8, 8, 448)    731136      block7a_se_excite[0][0]

__________________________________________________________________________________________________
block7a_project_bn (BatchNormal (None, 8, 8, 448)    1792        block7a_project_conv[0][0]

__________________________________________________________________________________________________
block7b_expand_conv (Conv2D)    (None, 8, 8, 2688)   1204224     block7a_project_bn[0][0]

__________________________________________________________________________________________________
block7b_expand_bn (BatchNormali (None, 8, 8, 2688)   10752       block7b_expand_conv[0][0]

__________________________________________________________________________________________________
block7b_expand_activation (Acti (None, 8, 8, 2688)   0           block7b_expand_bn[0][0]

__________________________________________________________________________________________________
block7b_dwconv (DepthwiseConv2D (None, 8, 8, 2688)   24192       block7b_expand_activation[0][0]

__________________________________________________________________________________________________
block7b_bn (BatchNormalization) (None, 8, 8, 2688)   10752       block7b_dwconv[0][0]

__________________________________________________________________________________________________
block7b_activation (Activation) (None, 8, 8, 2688)   0           block7b_bn[0][0]

__________________________________________________________________________________________________
block7b_se_squeeze (GlobalAvera (None, 2688)         0           block7b_activation[0][0]

__________________________________________________________________________________________________
block7b_se_reshape (Reshape)    (None, 1, 1, 2688)   0           block7b_se_squeeze[0][0]

__________________________________________________________________________________________________
block7b_se_reduce (Conv2D)      (None, 1, 1, 112)    301168      block7b_se_reshape[0][0]

__________________________________________________________________________________________________
block7b_se_expand (Conv2D)      (None, 1, 1, 2688)   303744      block7b_se_reduce[0][0]

__________________________________________________________________________________________________
block7b_se_excite (Multiply)    (None, 8, 8, 2688)   0           block7b_activation[0][0]

                                                                 block7b_se_expand[0][0]

__________________________________________________________________________________________________
block7b_project_conv (Conv2D)   (None, 8, 8, 448)    1204224     block7b_se_excite[0][0]

__________________________________________________________________________________________________
block7b_project_bn (BatchNormal (None, 8, 8, 448)    1792        block7b_project_conv[0][0]
--------------------------------------------------------------------------------------------------
block7b_drop (Dropout)          (None, 8, 8, 448)    0           block7b_project_bn[0][0]

--------------------------------------------------------------------------------------------------
block7b_add (Add)               (None, 8, 8, 448)    0           block7b_drop[0][0]

                                                                 block7a_project_bn[0][0]

--------------------------------------------------------------------------------------------------
average_pooling (AveragePooling (None, 1, 1, 448)    0           block7b_add[0][0]

--------------------------------------------------------------------------------------------------
pool_1x1conv2d (Conv2D)         (None, 1, 1, 256)    114688      average_pooling[0][0]

--------------------------------------------------------------------------------------------------
bn_1 (BatchNormalization)       (None, 1, 1, 256)    1024        pool_1x1conv2d[0][0]

--------------------------------------------------------------------------------------------------
ASPP_conv2d_d1 (Conv2D)         (None, 8, 8, 256)    114688      block7b_add[0][0]

--------------------------------------------------------------------------------------------------
ASPP_conv2d_d6 (Conv2D)         (None, 8, 8, 256)    1032192     block7b_add[0][0]

--------------------------------------------------------------------------------------------------
ASPP_conv2d_d12 (Conv2D)        (None, 8, 8, 256)    1032192     block7b_add[0][0]

--------------------------------------------------------------------------------------------------
ASPP_conv2d_d18 (Conv2D)        (None, 8, 8, 256)    1032192     block7b_add[0][0]

--------------------------------------------------------------------------------------------------
relu_1 (Activation)             (None, 1, 1, 256)    0           bn_1[0][0]

--------------------------------------------------------------------------------------------------
bn_2 (BatchNormalization)       (None, 8, 8, 256)    1024        ASPP_conv2d_d1[0][0]

--------------------------------------------------------------------------------------------------
bn_3 (BatchNormalization)       (None, 8, 8, 256)    1024        ASPP_conv2d_d6[0][0]

--------------------------------------------------------------------------------------------------
bn_4 (BatchNormalization)       (None, 8, 8, 256)    1024        ASPP_conv2d_d12[0][0]

--------------------------------------------------------------------------------------------------
bn_5 (BatchNormalization)       (None, 8, 8, 256)    1024        ASPP_conv2d_d18[0][0]

--------------------------------------------------------------------------------------------------
relu_1_upsample (Lambda)        (None, 8, 8, 256)    0           relu_1[0][0]

--------------------------------------------------------------------------------------------------
relu_2 (Activation)             (None, 8, 8, 256)    0           bn_2[0][0]

--------------------------------------------------------------------------------------------------
relu_3 (Activation)             (None, 8, 8, 256)    0           bn_3[0][0]

--------------------------------------------------------------------------------------------------
relu_4 (Activation)             (None, 8, 8, 256)    0           bn_4[0][0]
```

```
__________________________________________________________________________________________________
relu_5 (Activation)             (None, 8, 8, 256)    0           bn_5[0][0]
__________________________________________________________________________________________________
ASPP_concat (Concatenate)       (None, 8, 8, 1280)   0           relu_1_upsample[0][0]
                                                                 relu_2[0][0]
                                                                 relu_3[0][0]
                                                                 relu_4[0][0]
                                                                 relu_5[0][0]
__________________________________________________________________________________________________
ASPP_conv2d_final (Conv2D)      (None, 8, 8, 256)    327680      ASPP_concat[0][0]
__________________________________________________________________________________________________
bn_final (BatchNormalization)   (None, 8, 8, 256)    1024        ASPP_conv2d_final[0][0]
__________________________________________________________________________________________________
low_level_projection (Conv2D)   (None, 64, 64, 48)   1536        block2b_add[0][0]
__________________________________________________________________________________________________
relu_final (Activation)         (None, 8, 8, 256)    0           bn_final[0][0]
__________________________________________________________________________________________________
bn_low_level_projection (BatchN (None, 64, 64, 48)   192         low_level_projection[0][0]
__________________________________________________________________________________________________
relu_final_upsample (Lambda)    (None, 64, 64, 256)  0           relu_final[0][0]
__________________________________________________________________________________________________
low_level_activation (Activatio (None, 64, 64, 48)   0           bn_low_level_projection[0][0]
__________________________________________________________________________________________________
decoder_concat (Concatenate)    (None, 64, 64, 304)  0           relu_final_upsample[0][0]
                                                                 low_level_activation[0][0]
__________________________________________________________________________________________________
decoder_conv2d_1 (Conv2D)       (None, 64, 64, 256)  700416      decoder_concat[0][0]
__________________________________________________________________________________________________
bn_decoder_1 (BatchNormalizatio (None, 64, 64, 256)  1024        decoder_conv2d_1[0][0]
__________________________________________________________________________________________________
activation_decoder_1 (Activatio (None, 64, 64, 256)  0           bn_decoder_1[0][0]
__________________________________________________________________________________________________
decoder_conv2d_2 (Conv2D)       (None, 64, 64, 256)  589824      activation_decoder_1[0][0]
__________________________________________________________________________________________________
bn_decoder_2 (BatchNormalizatio (None, 64, 64, 256)  1024        decoder_conv2d_2[0][0]
__________________________________________________________________________________________________
activation_decoder_2 (Activatio (None, 64, 64, 256)  0           bn_decoder_2[0][0]
____________________________________________________________________________________________________
activation_decoder_2_upsample ( (None, 256, 256, 256 0          activation_decoder_2[0][0]
____________________________________________________________________________________________________
output_layer (Conv2D)           (None, 256, 256, 1)  257        activation_decoder_2_upsample[0][
====================================================================================================
Total params: 21,817,888
Trainable params: 21,692,073
Non-trainable params: 125,815
```

Appendix I. ATT Squeeze U-Net layers and parameters

```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 256, 256, 3) 0
__________________________________________________________________________________________________
lambda (Lambda)                 (None, 256, 256, 3)  0           input_1[0][0]
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 128, 128, 64) 9472        lambda[0][0]
__________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 64, 64, 64)   0           conv2d[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 64, 64, 16)   1040        max_pooling2d[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 64, 64, 16)   64          conv2d_1[0][0]
__________________________________________________________________________________________________
depthwise_conv2d (DepthwiseConv (None, 64, 64, 16)   160         batch_normalization[0][0]
__________________________________________________________________________________________________
tf.compat.v1.shape (TFOpLambda) (4,)                 0           depthwise_conv2d[0][0]
__________________________________________________________________________________________________
tf.__operators__.getitem (Slici ()                   0           tf.compat.v1.shape[0][0]
__________________________________________________________________________________________________
tf.convert_to_tensor (TFOpLambd (5,)                 0           tf.__operators__.getitem[0][0]
__________________________________________________________________________________________________
tf.reshape (TFOpLambda)         (None, 64, 64, 1, 16 0           depthwise_conv2d[0][0]
                                                                 tf.convert_to_tensor[0][0]
__________________________________________________________________________________________________
tf.compat.v1.transpose (TFOpLam (None, 64, 64, 16, 1 0           tf.reshape[0][0]
__________________________________________________________________________________________________
tf.compat.v1.shape_1 (TFOpLambd (5,)                 0           tf.compat.v1.transpose[0][0]
__________________________________________________________________________________________________
tf.__operators__.getitem_1 (Sli ()                   0           tf.compat.v1.shape_1[0][0]
__________________________________________________________________________________________________
tf.convert_to_tensor_1 (TFOpLam (4,)                 0           tf.__operators__.getitem_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 64, 64, 64)   1088        batch_normalization[0][0]
__________________________________________________________________________________________________
tf.reshape_1 (TFOpLambda)       (None, 64, 64, 16)   0           tf.compat.v1.transpose[0][0]
```

```
                                                                tf.convert_to_tensor_1[0][0]

__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 64, 64, 80)   0           conv2d_2[0][0]
                                                                 tf.reshape_1[0][0]

__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 64, 64, 16)   1296        concatenate[0][0]

__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 64, 64, 16)   64          conv2d_3[0][0]

__________________________________________________________________________________________________
depthwise_conv2d_1 (DepthwiseCo (None, 64, 64, 16)   160         batch_normalization_1[0][0]

__________________________________________________________________________________________________
tf.compat.v1.shape_2 (TFOpLambd (4,)                 0           depthwise_conv2d_1[0][0]

__________________________________________________________________________________________________
tf.__operators__.getitem_2 (Sli ()                   0           tf.compat.v1.shape_2[0][0]

__________________________________________________________________________________________________
tf.convert_to_tensor_2 (TFOpLam (5,)                 0           tf.__operators__.getitem_2[0][0]

__________________________________________________________________________________________________
tf.reshape_2 (TFOpLambda)       (None, 64, 64, 1, 16 0           depthwise_conv2d_1[0][0]
                                                                 tf.convert_to_tensor_2[0][0]

__________________________________________________________________________________________________
tf.compat.v1.transpose_1 (TFOpL (None, 64, 64, 16, 1 0           tf.reshape_2[0][0]

__________________________________________________________________________________________________
tf.compat.v1.shape_3 (TFOpLambd (5,)                 0           tf.compat.v1.transpose_1[0][0]

__________________________________________________________________________________________________
tf.__operators__.getitem_3 (Sli ()                   0           tf.compat.v1.shape_3[0][0]

__________________________________________________________________________________________________
tf.convert_to_tensor_3 (TFOpLam (4,)                 0           tf.__operators__.getitem_3[0][0]

__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 64, 64, 64)   1088        batch_normalization_1[0][0]

__________________________________________________________________________________________________
tf.reshape_3 (TFOpLambda)       (None, 64, 64, 16)   0           tf.compat.v1.transpose_1[0][0]
                                                                 tf.convert_to_tensor_3[0][0]

__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 64, 64, 80)   0           conv2d_4[0][0]
                                                                 tf.reshape_3[0][0]

__________________________________________________________________________________________________
conv2d_5 (Conv2D)               (None, 64, 64, 16)   1296        concatenate_1[0][0]
```

```
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 64, 64, 16)   64          conv2d_5[0][0]
__________________________________________________________________________________________________
depthwise_conv2d_2 (DepthwiseCo (None, 64, 64, 16)   160         batch_normalization_2[0][0]
__________________________________________________________________________________________________
tf.compat.v1.shape_4 (TFOpLambd (4,)                 0           depthwise_conv2d_2[0][0]
__________________________________________________________________________________________________
tf.__operators__.getitem_4 (Sli ()                   0           tf.compat.v1.shape_4[0][0]
__________________________________________________________________________________________________
tf.convert_to_tensor_4 (TFOpLam (5,)                 0           tf.__operators__.getitem_4[0][0]
__________________________________________________________________________________________________
tf.reshape_4 (TFOpLambda)       (None, 64, 64, 1, 16 0           depthwise_conv2d_2[0][0]
                                                                 tf.convert_to_tensor_4[0][0]
__________________________________________________________________________________________________
tf.compat.v1.transpose_2 (TFOpL (None, 64, 64, 16, 1 0           tf.reshape_4[0][0]
__________________________________________________________________________________________________
tf.compat.v1.shape_5 (TFOpLambd (5,)                 0           tf.compat.v1.transpose_2[0][0]
__________________________________________________________________________________________________
tf.__operators__.getitem_5 (Sli ()                   0           tf.compat.v1.shape_5[0][0]
__________________________________________________________________________________________________
tf.convert_to_tensor_5 (TFOpLam (4,)                 0           tf.__operators__.getitem_5[0][0]
__________________________________________________________________________________________________
conv2d_6 (Conv2D)               (None, 64, 64, 64)   1088        batch_normalization_2[0][0]
__________________________________________________________________________________________________
tf.reshape_5 (TFOpLambda)       (None, 64, 64, 16)   0           tf.compat.v1.transpose_2[0][0]
                                                                 tf.convert_to_tensor_5[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 64, 64, 80)   0           conv2d_6[0][0]
                                                                 tf.reshape_5[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 32, 32, 80)   0           concatenate_2[0][0]
__________________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 32, 32, 32)   2592        max_pooling2d_1[0][0]
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 32, 32, 32)   128         conv2d_7[0][0]
__________________________________________________________________________________________________
depthwise_conv2d_3 (DepthwiseCo (None, 32, 32, 32)   320         batch_normalization_3[0][0]
__________________________________________________________________________________________________
tf.compat.v1.shape_6 (TFOpLambd (4,)                 0           depthwise_conv2d_3[0][0]
__________________________________________________________________________________________________
tf.__operators__.getitem_6 (Sli ()                   0           tf.compat.v1.shape_6[0][0]
__________________________________________________________________________________________________
tf.convert_to_tensor_6 (TFOpLam (5,)                 0           tf.__operators__.getitem_6[0][0]
__________________________________________________________________________________________________
tf.reshape_6 (TFOpLambda)       (None, 32, 32, 1, 32 0           depthwise_conv2d_3[0][0]
                                                                 tf.convert_to_tensor_6[0][0]
__________________________________________________________________________________________________
tf.compat.v1.transpose_3 (TFOpL (None, 32, 32, 32, 1 0           tf.reshape_6[0][0]
__________________________________________________________________________________________________
tf.compat.v1.shape_7 (TFOpLambd (5,)                 0           tf.compat.v1.transpose_3[0][0]
__________________________________________________________________________________________________
tf.__operators__.getitem_7 (Sli ()                   0           tf.compat.v1.shape_7[0][0]
__________________________________________________________________________________________________
tf.convert_to_tensor_7 (TFOpLam (4,)                 0           tf.__operators__.getitem_7[0][0]
__________________________________________________________________________________________________
conv2d_8 (Conv2D)               (None, 32, 32, 128)  4224        batch_normalization_3[0][0]
__________________________________________________________________________________________________
tf.reshape_7 (TFOpLambda)       (None, 32, 32, 32)   0           tf.compat.v1.transpose_3[0][0]
                                                                 tf.convert_to_tensor_7[0][0]
__________________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 32, 32, 160)  0           conv2d_8[0][0]
                                                                 tf.reshape_7[0][0]
__________________________________________________________________________________________________
conv2d_9 (Conv2D)               (None, 32, 32, 32)   5152        concatenate_3[0][0]
__________________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, 32, 32, 32)   128         conv2d_9[0][0]
__________________________________________________________________________________________________
depthwise_conv2d_4 (DepthwiseCo (None, 32, 32, 32)   320         batch_normalization_4[0][0]
__________________________________________________________________________________________________
tf.compat.v1.shape_8 (TFOpLambd (4,)                 0           depthwise_conv2d_4[0][0]
__________________________________________________________________________________________________
tf.__operators__.getitem_8 (Sli ()                   0           tf.compat.v1.shape_8[0][0]
__________________________________________________________________________________________________
tf.convert_to_tensor_8 (TFOpLam (5,)                 0           tf.__operators__.getitem_8[0][0]
________________________________________________________________________________________________
tf.reshape_8 (TFOpLambda)       (None, 32, 32, 1, 32 0           depthwise_conv2d_4[0][0]
                                                                 tf.convert_to_tensor_8[0][0]
________________________________________________________________________________________________
tf.compat.v1.transpose_4 (TFOpL (None, 32, 32, 32, 1 0           tf.reshape_8[0][0]
________________________________________________________________________________________________
tf.compat.v1.shape_9 (TFOpLambd (5,)                 0           tf.compat.v1.transpose_4[0][0]
________________________________________________________________________________________________
tf.__operators__.getitem_9 (Sli ()                   0           tf.compat.v1.shape_9[0][0]
________________________________________________________________________________________________
tf.convert_to_tensor_9 (TFOpLam (4,)                 0           tf.__operators__.getitem_9[0][0]
________________________________________________________________________________________________
conv2d_10 (Conv2D)              (None, 32, 32, 128)  4224        batch_normalization_4[0][0]
________________________________________________________________________________________________
tf.reshape_9 (TFOpLambda)       (None, 32, 32, 32)   0           tf.compat.v1.transpose_4[0][0]
                                                                 tf.convert_to_tensor_9[0][0]
________________________________________________________________________________________________
concatenate_4 (Concatenate)     (None, 32, 32, 160)  0           conv2d_10[0][0]
                                                                 tf.reshape_9[0][0]
________________________________________________________________________________________________
conv2d_11 (Conv2D)              (None, 32, 32, 32)   5152        concatenate_4[0][0]
________________________________________________________________________________________________
batch_normalization_5 (BatchNor (None, 32, 32, 32)   128         conv2d_11[0][0]
________________________________________________________________________________________________
depthwise_conv2d_5 (DepthwiseCo (None, 32, 32, 32)   320         batch_normalization_5[0][0]
________________________________________________________________________________________________
tf.compat.v1.shape_10 (TFOpLamb (4,)                 0           depthwise_conv2d_5[0][0]
________________________________________________________________________________________________
tf.__operators__.getitem_10 (Sl ()                   0           tf.compat.v1.shape_10[0][0]
________________________________________________________________________________________________
tf.convert_to_tensor_10 (TFOpLa (5,)                 0           tf.__operators__.getitem_10[0][0]
________________________________________________________________________________________________
tf.reshape_10 (TFOpLambda)      (None, 32, 32, 1, 32 0           depthwise_conv2d_5[0][0]
                                                                 tf.convert_to_tensor_10[0][0]
________________________________________________________________________________________________
tf.compat.v1.transpose_5 (TFOpL (None, 32, 32, 32, 1 0           tf.reshape_10[0][0]
________________________________________________________________________________________________
```

```
tf.compat.v1.shape_11 (TFOpLamb (5,)                 0           tf.compat.v1.transpose_5[0][0]

__________________________________________________________________________________________________
tf.__operators__.getitem_11 (Sl ()                   0           tf.compat.v1.shape_11[0][0]

__________________________________________________________________________________________________
tf.convert_to_tensor_11 (TFOpLa (4,)                 0           tf.__operators__.getitem_11[0][0]

__________________________________________________________________________________________________
conv2d_12 (Conv2D)              (None, 32, 32, 128)  4224        batch_normalization_5[0][0]

__________________________________________________________________________________________________
tf.reshape_11 (TFOpLambda)      (None, 32, 32, 32)   0           tf.compat.v1.transpose_5[0][0]

                                                                 tf.convert_to_tensor_11[0][0]

__________________________________________________________________________________________________
concatenate_5 (Concatenate)     (None, 32, 32, 160)  0           conv2d_12[0][0]

                                                                 tf.reshape_11[0][0]

__________________________________________________________________________________________________
conv2d_13 (Conv2D)              (None, 32, 32, 32)   5152        concatenate_5[0][0]

__________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 32, 32, 32)   128         conv2d_13[0][0]

__________________________________________________________________________________________________
depthwise_conv2d_6 (DepthwiseCo (None, 32, 32, 32)   320         batch_normalization_6[0][0]

__________________________________________________________________________________________________
tf.compat.v1.shape_12 (TFOpLamb (4,)                 0           depthwise_conv2d_6[0][0]

__________________________________________________________________________________________________
tf.__operators__.getitem_12 (Sl ()                   0           tf.compat.v1.shape_12[0][0]

__________________________________________________________________________________________________
tf.convert_to_tensor_12 (TFOpLa (5,)                 0           tf.__operators__.getitem_12[0][0]

__________________________________________________________________________________________________
tf.reshape_12 (TFOpLambda)      (None, 32, 32, 1, 32 0           depthwise_conv2d_6[0][0]

                                                                 tf.convert_to_tensor_12[0][0]

__________________________________________________________________________________________________
tf.compat.v1.transpose_6 (TFOpL (None, 32, 32, 32, 1 0           tf.reshape_12[0][0]

__________________________________________________________________________________________________
tf.compat.v1.shape_13 (TFOpLamb (5,)                 0           tf.compat.v1.transpose_6[0][0]

__________________________________________________________________________________________________
tf.__operators__.getitem_13 (Sl ()                   0           tf.compat.v1.shape_13[0][0]

__________________________________________________________________________________________________
tf.convert_to_tensor_13 (TFOpLa (4,)                 0           tf.__operators__.getitem_13[0][0]

__________________________________________________________________________________________________
```

```
conv2d_14 (Conv2D)              (None, 32, 32, 128)  4224        batch_normalization_6[0][0]

--------------------------------------------------------------------------------------------------
tf.reshape_13 (TFOpLambda)      (None, 32, 32, 32)   0           tf.compat.v1.transpose_6[0][0]

                                                                 tf.convert_to_tensor_13[0][0]

--------------------------------------------------------------------------------------------------
concatenate_6 (Concatenate)     (None, 32, 32, 160)  0           conv2d_14[0][0]

                                                                 tf.reshape_13[0][0]

--------------------------------------------------------------------------------------------------
max_pooling2d_2 (MaxPooling2D)  (None, 16, 16, 160)  0           concatenate_6[0][0]

--------------------------------------------------------------------------------------------------
conv2d_15 (Conv2D)              (None, 16, 16, 48)   7728        max_pooling2d_2[0][0]

--------------------------------------------------------------------------------------------------
batch_normalization_7 (BatchNor (None, 16, 16, 48)   192         conv2d_15[0][0]

--------------------------------------------------------------------------------------------------
depthwise_conv2d_7 (DepthwiseCo (None, 16, 16, 48)   480         batch_normalization_7[0][0]

--------------------------------------------------------------------------------------------------
tf.compat.v1.shape_14 (TFOpLamb (4,)                 0           depthwise_conv2d_7[0][0]

--------------------------------------------------------------------------------------------------
tf.__operators__.getitem_14 (Sl ()                   0           tf.compat.v1.shape_14[0][0]

--------------------------------------------------------------------------------------------------
tf.convert_to_tensor_14 (TFOpLa (5,)                 0           tf.__operators__.getitem_14[0][0]

--------------------------------------------------------------------------------------------------
tf.reshape_14 (TFOpLambda)      (None, 16, 16, 1, 48 0           depthwise_conv2d_7[0][0]

                                                                 tf.convert_to_tensor_14[0][0]

--------------------------------------------------------------------------------------------------
tf.compat.v1.transpose_7 (TFOpL (None, 16, 16, 48, 1 0           tf.reshape_14[0][0]

--------------------------------------------------------------------------------------------------
tf.compat.v1.shape_15 (TFOpLamb (5,)                 0           tf.compat.v1.transpose_7[0][0]

--------------------------------------------------------------------------------------------------
tf.__operators__.getitem_15 (Sl ()                   0           tf.compat.v1.shape_15[0][0]

--------------------------------------------------------------------------------------------------
tf.convert_to_tensor_15 (TFOpLa (4,)                 0           tf.__operators__.getitem_15[0][0]

--------------------------------------------------------------------------------------------------
conv2d_16 (Conv2D)              (None, 16, 16, 192)  9408        batch_normalization_7[0][0]

--------------------------------------------------------------------------------------------------
tf.reshape_15 (TFOpLambda)      (None, 16, 16, 48)   0           tf.compat.v1.transpose_7[0][0]

                                                                 tf.convert_to_tensor_15[0][0]
```

```
__________________________________________________________________________________________________
concatenate_7 (Concatenate)     (None, 16, 16, 240)  0           conv2d_16[0][0]
                                                                  tf.reshape_15[0][0]
__________________________________________________________________________________________________
conv2d_17 (Conv2D)              (None, 16, 16, 64)   15424       concatenate_7[0][0]
__________________________________________________________________________________________________
conv2d_18 (Conv2D)              (None, 16, 16, 64)   36928       conv2d_17[0][0]
__________________________________________________________________________________________________
conv2d_19 (Conv2D)              (None, 16, 16, 64)   10304       concatenate_3[0][0]
__________________________________________________________________________________________________
conv2d_20 (Conv2D)              (None, 16, 16, 64)   4160        conv2d_18[0][0]
__________________________________________________________________________________________________
add (Add)                       (None, 16, 16, 64)   0           conv2d_19[0][0]
                                                                  conv2d_20[0][0]
__________________________________________________________________________________________________
re_lu (ReLU)                    (None, 16, 16, 64)   0           add[0][0]
__________________________________________________________________________________________________
conv2d_22 (Conv2D)              (None, 16, 16, 192)  12480       conv2d_18[0][0]
__________________________________________________________________________________________________
conv2d_23 (Conv2D)              (None, 16, 16, 192)  110784      conv2d_18[0][0]
__________________________________________________________________________________________________
conv2d_21 (Conv2D)              (None, 16, 16, 1)    65          re_lu[0][0]
__________________________________________________________________________________________________
concatenate_8 (Concatenate)     (None, 16, 16, 384)  0           conv2d_22[0][0]
                                                                  conv2d_23[0][0]
__________________________________________________________________________________________________
activation (Activation)         (None, 16, 16, 1)    0           conv2d_21[0][0]
__________________________________________________________________________________________________
conv2d_24 (Conv2D)              (None, 16, 16, 48)   18480       concatenate_8[0][0]
__________________________________________________________________________________________________
up_sampling2d (UpSampling2D)    (None, 32, 32, 1)    0           activation[0][0]
__________________________________________________________________________________________________
up_sampling2d_1 (UpSampling2D)  (None, 32, 32, 48)   0           conv2d_24[0][0]
__________________________________________________________________________________________________
multiply (Multiply)             (None, 32, 32, 160)  0           concatenate_3[0][0]
                                                                  up_sampling2d[0][0]
__________________________________________________________________________________________________
```

```
conv2d_25 (Conv2D)              (None, 32, 32, 48)   20784       up_sampling2d_1[0][0]

__________________________________________________________________________________________________
concatenate_9 (Concatenate)     (None, 32, 32, 208)  0           multiply[0][0]

                                                                 conv2d_25[0][0]

__________________________________________________________________________________________________
conv2d_transpose (Conv2DTranspo (None, 32, 32, 128)  239744      concatenate_9[0][0]

__________________________________________________________________________________________________
conv2d_26 (Conv2D)              (None, 32, 32, 32)   2592        concatenate_1[0][0]

__________________________________________________________________________________________________
conv2d_27 (Conv2D)              (None, 32, 32, 32)   4128        conv2d_transpose[0][0]

__________________________________________________________________________________________________
add_1 (Add)                     (None, 32, 32, 32)   0           conv2d_26[0][0]

                                                                 conv2d_27[0][0]

__________________________________________________________________________________________________
re_lu_1 (ReLU)                  (None, 32, 32, 32)   0           add_1[0][0]

__________________________________________________________________________________________________
conv2d_29 (Conv2D)              (None, 32, 32, 128)  16512       conv2d_transpose[0][0]

__________________________________________________________________________________________________
conv2d_30 (Conv2D)              (None, 32, 32, 128)  147584      conv2d_transpose[0][0]

__________________________________________________________________________________________________
conv2d_28 (Conv2D)              (None, 32, 32, 1)    33          re_lu_1[0][0]

__________________________________________________________________________________________________
concatenate_10 (Concatenate)    (None, 32, 32, 256)  0           conv2d_29[0][0]

                                                                 conv2d_30[0][0]

__________________________________________________________________________________________________
activation_1 (Activation)       (None, 32, 32, 1)    0           conv2d_28[0][0]

__________________________________________________________________________________________________
conv2d_31 (Conv2D)              (None, 32, 32, 32)   8224        concatenate_10[0][0]

__________________________________________________________________________________________________
up_sampling2d_2 (UpSampling2D)  (None, 64, 64, 1)    0           activation_1[0][0]

__________________________________________________________________________________________________
up_sampling2d_3 (UpSampling2D)  (None, 64, 64, 32)   0           conv2d_31[0][0]

__________________________________________________________________________________________________
multiply_1 (Multiply)           (None, 64, 64, 80)   0           concatenate_1[0][0]

                                                                 up_sampling2d_2[0][0]

__________________________________________________________________________________________________
conv2d_32 (Conv2D)              (None, 64, 64, 32)   9248        up_sampling2d_3[0][0]
```

```
__________________________________________________________________________________________________
concatenate_11 (Concatenate)    (None, 64, 64, 112)  0           multiply_1[0][0]
                                                                 conv2d_32[0][0]
__________________________________________________________________________________________________
conv2d_transpose_1 (Conv2DTrans (None, 64, 64, 64)   64576       concatenate_11[0][0]
__________________________________________________________________________________________________
conv2d_33 (Conv2D)              (None, 64, 64, 32)   2080        conv2d[0][0]
__________________________________________________________________________________________________
conv2d_34 (Conv2D)              (None, 64, 64, 32)   2080        conv2d_transpose_1[0][0]
__________________________________________________________________________________________________
add_2 (Add)                     (None, 64, 64, 32)   0           conv2d_33[0][0]
                                                                 conv2d_34[0][0]
__________________________________________________________________________________________________
re_lu_2 (ReLU)                  (None, 64, 64, 32)   0           add_2[0][0]
__________________________________________________________________________________________________
conv2d_36 (Conv2D)              (None, 64, 64, 64)   4160        conv2d_transpose_1[0][0]
__________________________________________________________________________________________________
conv2d_37 (Conv2D)              (None, 64, 64, 64)   36928       conv2d_transpose_1[0][0]
__________________________________________________________________________________________________
conv2d_35 (Conv2D)              (None, 64, 64, 1)    33          re_lu_2[0][0]
__________________________________________________________________________________________________
concatenate_12 (Concatenate)    (None, 64, 64, 128)  0           conv2d_36[0][0]
                                                                 conv2d_37[0][0]
__________________________________________________________________________________________________
activation_2 (Activation)       (None, 64, 64, 1)    0           conv2d_35[0][0]
__________________________________________________________________________________________________
conv2d_38 (Conv2D)              (None, 64, 64, 16)   2064        concatenate_12[0][0]
__________________________________________________________________________________________________
up_sampling2d_4 (UpSampling2D)  (None, 128, 128, 1)  0           activation_2[0][0]
__________________________________________________________________________________________________
up_sampling2d_5 (UpSampling2D)  (None, 128, 128, 16) 0           conv2d_38[0][0]
__________________________________________________________________________________________________
multiply_2 (Multiply)           (None, 128, 128, 64) 0           conv2d[0][0]
                                                                 up_sampling2d_4[0][0]
__________________________________________________________________________________________________
conv2d_39 (Conv2D)              (None, 128, 128, 16) 2320        up_sampling2d_5[0][0]
__________________________________________________________________________________________________
```

```
concatenate_13 (Concatenate)    (None, 128, 128, 80) 0           multiply_2[0][0]
                                                                 conv2d_39[0][0]
__________________________________________________________________________________________________
conv2d_transpose_2 (Conv2DTrans (None, 256, 256, 32) 23072       concatenate_13[0][0]
__________________________________________________________________________________________________
conv2d_40 (Conv2D)              (None, 256, 256, 64) 18496       conv2d_transpose_2[0][0]
__________________________________________________________________________________________________
conv2d_41 (Conv2D)              (None, 256, 256, 1)  65          conv2d_40[0][0]
==================================================================================================
Total params: 884,932
Trainable params: 884,484
Non-trainable params: 448
```

References

[1] Ahlawat, H. D. & Chauhan, R. (2020). Detection and monitoring of forest fire using serial communication and wi-fi wireless sensor network. In *Handbook of Wireless Sensor Networks: Issues and Challenges in Current Scenario's* (pp. 464–492). Springer.

[2] Al-Rawi, K., Casanova, J., & Calle, A. (2001). Burned area mapping system and fire detection system, based on neural networks and noaa-avhrr imagery. *International Journal of Remote Sensing*, 22(10), 2015–2032.

[3] Amari, S.-i. & Arbib, M. A. (2013). *Competition and Cooperation in Neural Nets: Proceedings of the US-Japan Joint Seminar Held at Kyoto, Japan February 15–19, 1982*, volume 45. Springer Science & Business Media.

[4] Angayarkkani, K. & Radhakrishnan, N. (2010). An intelligent system for effective forest fire detection using spatial data. *arXiv preprint arXiv:1002.2199*.

[5] Aslan, Y. E., Korpeoglu, I., & Ulusoy, Ö. (2012). A framework for use of wireless sensor networks in forest fire detection and monitoring. *Computers, Environment and Urban Systems*, 36(6), 614–625.

[6] Ba, R., Chen, C., Yuan, J., Song, W., & Lo, S. (2019). Smokenet: Satellite smoke scene detection using convolutional neural network with spatial and channel-wise attention. *Remote Sensing*, 11(14), 1702.

[7] Balasubramanian, A., Sathick, M., & Senthamaran, K. (2012). An efficient method of forest fire detection using wireless sensor network with yager's modified dempster's rule. *Int J Emerg Technol Adv Eng IJETAE*, 2(1), 222–227.

[8] Beheshti, N. & Johnsson, L. (2020). Squeeze u-net: A memory and energy efficient image segmentation network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[9] Blasch, A. S. F. A. A. R. L. Z. P. F. E. (2020). The flame dataset: Aerial imagery pile burn detection using drones (uavs).

[10] Blenz (2019). What is the difference between semantic segmentation, object detection and instance segmentation? https://datascience.stackexchange.com/questions/52015/what-is-the-difference-between-semantic-segmentation-object-detection-and-insta. [Online; accessed 2021-5-25].

[11] Bowman, D. M., Perry, G. L., Higgins, S. I., Johnson, C. N., Fuhlendorf, S. D., & Murphy, B. P. (2016). Pyrodiversity is the coupling of biodiversity and fire regimes in food webs. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 371(1696), 20150169.

[12] Brostow, G. J., Fauqueur, J., & Cipolla, R. (2008a). Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, xx(x), xx–xx.

[13] Brostow, G. J., Fauqueur, J., & Cipolla, R. (2008b). Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, xx(x), xx–xx.

[14] Carandini, M. (2006). What simple and complex cells compute. *The Journal of physiology*, 577(Pt 2), 463.

[15] Chen, L., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587.

[16] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*.

[17] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018a). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4), 834–848.

[18] Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018b). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 801–818).

[19] Chen, Y., Zhang, Y., Xin, J., Wang, G., Mu, L., Yi, Y., Liu, H., & Liu, D. (2019). Uav image-based forest fire detection approach using convolutional neural network. In *2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA)* (pp. 2118–2123).: IEEE.

[20] Colorado State University (2017). Viirs bands and bandwidths. `http://rammb.cira.colostate.edu/projects/npp/VIIRS_bands_and_bandwidths.pdf`. [Online; accessed 2021-02-17].

[21] de Miguel Molina, B. & Oña, M. S. (2018). The drone sector in europe. In *Ethics and civil drones* (pp. 7–33). Springer, Cham.

[22] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248–255).: Ieee.

[23] Dice, L. R. (1945). Measures of the amount of ecologic association between species. *Ecology*, 26(3), 297–302.

[24] Doshi, J., Basu, S., & Pang, G. (2018). From satellite imagery to disaster insights. *arXiv preprint arXiv:1812.07033*.

[25] Doshi, J., Garcia, D., Massey, C., Llueca, P., Borensztein, N., Baird, M., Cook, M., & Raj, D. (2019). Firenet: Real-time segmentation of fire perimeter from aerial video. *arXiv preprint arXiv:1910.06407*.

[26] Dubey, V., Kumar, P., & Chauhan, N. (2019). Forest fire detection system using iot and artificial neural network. In *International Conference on Innovative Computing and Communications* (pp. 323–337).: Springer.

[27] Earth Policy Institute (November 2009). Wildfires by region: Observations and future prospects. http://www.earth-policy.org/images/uploads/graphs_tables/fire.htm. [Online; accessed 2021-1-18].

[28] Environmental Protection Agency (2016). Climate change indicators: Wildfires. www.epa.gov/climate-indicators/climate-change-indicators-wildfires. [Online; accessed 2021-1-18].

[29] European Environment Agency (2019). Forest fires indicator assessment. https://www.eea.europa.eu/data-and-maps/indicators/forest-fire-danger-3/assessment. [Online; accessed 2021-1-18].

[30] European Union Aviation Safety Agency (2020). Civil drones (unmanned aircraft). https://www.easa.europa.eu/domains/civil-drones-rpas. [Online; accessed 2021-5-24].

[31] Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1), 98–136.

[32] Falconer, K. (2004). *Fractal geometry: mathematical foundations and applications*. John Wiley & Sons.

[33] Farasin, A., Colomba, L., & Garza, P. (2020). Double-step u-net: A deep learning-based approach for the estimation of wildfire damage severity through sentinel-2 satellite data. *Applied Sciences*, 10(12), 4332.

[34] Flood, N., Watson, F., & Collett, L. (2019). Using a u-net convolutional neural network to map woody vegetation extent from high resolution satellite imagery across queensland, australia. *International Journal of Applied Earth Observation and Geoinformation*, 82, 101897.

[35] Foggia, P., Saggese, A., & Vento, M. (2015). Real-time fire detection for video-surveillance applications using a combination of experts based on color, shape, and motion. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(9), 1545–1556.

[36] Fukushima, K. & Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets* (pp. 267–285). Springer.

[37] Gauss, C. F. & Bertrand, J. (1957). *Gauss's Work (1803-1826) on the Theory of Least Squares.*

[38] Guo, D., Pei, Y., Zheng, K., Yu, H., Lu, Y., & Wang, S. (2019). Degraded image semantic segmentation with dense-gram networks. *IEEE Transactions on Image Processing*, 29, 782–795.

[39] Hand, D. & Christen, P. (2018). A note on using the f-measure for evaluating record linkage algorithms. *Statistics and Computing*, 28(3), 539–547.

[40] Harkat, H., Nascimento, J., & Bernardino, A. (2020). Fire segmentation using a deeplabv3+ architecture. In *Image and Signal Processing for Remote Sensing XXVI*, volume 11533 (pp. 115330M).: International Society for Optics and Photonics.

[41] He, K. & Sun, J. (2015). Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5353–5360).

[42] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9), 1904–1916.

[43] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

[44] Hebb, D. (2002). The organization of behavior. 1949. *New York Wiely*, 2(7), 8.

[45] Hefeeda, M. & Bagheri, M. (2009). Forest fire modeling and early detection using wireless sensor networks. *Ad Hoc Sens. Wirel. Networks*, 7(3-4), 169–224.

[46] Herutomo, A., Abdurohman, M., Suwastika, N. A., Prabowo, S., & Wijiutomo, C. W. (2015). Forest fire detection system reliability test using wireless sensor network and openmtc communication platform. In *2015 3rd International conference on information and communication technology (ICoICT)* (pp. 87–91).: IEEE.

[47] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8), 2554–2558.

[48] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861.*

[49] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360.*

[50] James, W., Burkhardt, F., Bowers, F., & Skrupskelis, I. K. (1890). *The principles of psychology*, volume 1. Macmillan London.

[51] Jang, E., Kang, Y., Im, J., Lee, D.-W., Yoon, J., & Kim, S.-K. (2019). Detection and monitoring of forest fires using himawari-8 geostationary satellite data in south korea. *Remote Sensing*, 11(3), 271.

[52] Jansson, A., Humphrey, E., Montecchio, N., Bittner, R., Kumar, A., & Weyde, T. (2017). Singing voice separation with deep u-net convolutional networks.

[53] Japan Meteorological Agency (2014). New geostationary meteorological satellites — himawari-8/9 —. http://www.jma.go.jp/jma/jma-eng/satellite/news/himawari89/himawari89_leaflet.pdf. [Online; accessed 2021-02-19].

[54] Kalinicheva, E., Ienco, D., Sublime, J., & Trocan, M. (2020). Unsupervised change detection analysis in satellite image time series using deep learning combined with graph-based approaches. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13, 1450–1466.

[55] Keeley, J. E., Pausas, J. G., Rundel, P. W., Bond, W. J., & Bradstock, R. A. (2011). Fire as an evolutionary pressure shaping plant traits. *Trends in plant science*, 16(8), 406–411.

[56] Kim, H.-D., Jung, O.-C., & Bang, H. (2007). A computational approach to reduce the revisit time using a genetic algorithm. In *2007 International Conference on Control, Automation and Systems* (pp. 184–189).: IEEE.

[57] Kinaneva, D., Hristov, G., Raychev, J., & Zahariev, P. (2019). Early forest fire detection using drones and artificial intelligence. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (pp. 1060–1065).: IEEE.

[58] Kirillov, A., He, K., Girshick, R., Rother, C., & Dollár, P. (2019). Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 9404–9413).

[59] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 1097–1105.

[60] Larsen, A., Hanigan, I., Reich, B. J., Qin, Y., Cope, M., Morgan, G., & Rappold, A. G. (2021). A deep learning approach to identify smoke plumes in satellite imagery in near-real time for health risk communication. *Journal of exposure science & environmental epidemiology*, 31(1), 170–176.

[61] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

[62] Li, X., Song, W., Lian, L., & Wei, X. (2015). Forest fire smoke detection using back-propagation neural network based on modis data. *Remote Sensing*, 7(4), 4473–4498.

[63] Li, Z., Khananian, A., Fraser, R. H., & Cihlar, J. (2001). Automatic detection of fire smoke using artificial neural networks and threshold approaches applied to avhrr imagery. *IEEE Transactions on geoscience and remote sensing*, 39(9), 1859–1870.

[64] Liang, S. (2017). *Comprehensive Remote Sensing*. Elsevier.

[65] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431–3440).

[66] Lucas, C., Hennessy, K., Mills, G., & Bathols, J. (2007). Bushfire weather in southeast australia: recent trends and projected climate change impacts.

[67] McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4), 115–133.

[68] Miller, J., Borne, K., Thomas, B., Huang, Z., & Chi, Y. (2013). Automated wildfire detection through artificial neural networks. In *Remote Sensing and Modeling Applications to Wildland Fires* (pp. 293–304). Springer.

[69] Minaee, S., Boykov, Y. Y., Porikli, F., Plaza, A. J., Kehtarnavaz, N., & Terzopoulos, D. (2021). Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[70] Minsky, M. & Papert, S. (1969). Perceptrons.

[71] Moritz, M. A., Batllori, E., Bradstock, R. A., Gill, A. M., Handmer, J., Hessburg, P. F., Leonard, J., McCaffrey, S., Odion, D. C., Schoennagel, T., et al. (2014). Learning to coexist with wildfire. *Nature*, 515(7525), 58–66.

[72] National Aeronautics and Space Administration (2015). Modis components. https://modis.gsfc.nasa.gov/about/components.php. [Online; accessed 2021-1-19].

[73] National Aeronautics and Space Administration (2017). Viirs land. https://viirsland.gsfc.nasa.gov/. [Online; accessed 2021-1-26].

[74] National Aeronautics and Space Administration (2019). Honeycomb platform. https://2019.spaceappschallenge.org/challenges/living-our-world/spot-fire-v20/teams/et760/project. [Online; accessed 2021-5-24].

[75] National Aeronautics and Space Administration (2021). Modis. https://terra.nasa.gov/about/terra-instruments/modis. [Online; accessed 2021-5-23].

[76] North, M., Stephens, S., Collins, B., Agee, J., Aplet, G., Franklin, J., & Fule, P. Z. (2015). Reform forest fire management. *Science*, 349(6254), 1280–1281.

[77] Ohio University (2021). 7 pros and cons of drones and unmanned aerial vehicles. https://onlinemasters.ohio.edu/blog/the-pros-and-cons-of-unmanned-aerial-vehicles-uavs/. [Online; accessed 2021-5-23].

[78] Oktay, O., Schlemper, J., Folgoc, L. L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N. Y., Kainz, B., et al. (2018). Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*.

[79] Pant, D., Verma, S., & Dhuliya, P. (2017). A study on disaster detection and management using wsn in himalayan region of uttarakhand. In *2017 3rd International conference on advances in computing, communication & automation (ICACCA)(Fall)* (pp. 1–6).: IEEE.

[80] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597.

[81] Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

[82] Rosenblatt, F. (1961). *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Technical report, Cornell Aeronautical Lab Inc Buffalo NY.

[83] Roy, P. (2003). Forest fire and degradation assessment using satellite remote sensing and geographic information system. *Satellite Remote sensing and GIS applications in agricultural meteorology*, (pp. 361).

[84] Sagan, H. (2012). *Space-filling curves*. Springer Science & Business Media.

[85] Sahin, Y. G. (2007). Animals as mobile biological sensors for forest fire detection. *Sensors*, 7(12), 3084–3099.

[86] Sasaki, Y. (2007). The truth of the f-measure. *Teach Tutor Mater*.

[87] Scott, A. C. & Glasspool, I. J. (2006). The diversification of paleozoic fire systems and fluctuations in atmospheric oxygen concentration. *Proceedings of the National Academy of Sciences*, 103(29), 10861–10865.

[88] Shamsoshoara, A., Afghah, F., Razi, A., Zheng, L., Fulé, P. Z., & Blasch, E. (2021). Aerial imagery pile burn detection using deep learning: The flame dataset. *Computer Networks*, 193, 108001.

[89] Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[90] Soliman, H., Sudan, K., & Mishra, A. (2010). A smart forest-fire early detection sensory system: Another approach of utilizing wireless sensor and neural networks. In *SENSORS, 2010 IEEE* (pp. 1900–1904).: IEEE.

[91] Sørenson, T. (1948). *A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species Content and Its Application to Analyses of the Vegetation on Danish Commons*. Biologiske skrifter. I kommission hos E. Munksgaard.

[92] Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

[93] Stewart, O. C. (2002). *Forgotten fires: Native Americans and the transient wilderness*. University of Oklahoma Press.

[94] Stoller, D., Ewert, S., & Dixon, S. (2018). Wave-u-net: A multi-scale neural network for end-to-end audio source separation. *arXiv preprint arXiv:1806.03185*.

[95] Sunar, F. & Özkan, C. (2001). Forest fire analysis with remote sensing data. *International Journal of Remote Sensing*, 22(12), 2265–2277.

[96] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1–9).

[97] Tan, M. & Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning* (pp. 6105–6114).: PMLR.

[98] Toan, N. T., Cong, P. T., Hung, N. Q. V., & Jo, J. (2019). A deep learning approach for early wildfire detection from hyperspectral satellite images. In *2019 7th International Conference on Robot Intelligence Technology and Applications (RiTA)* (pp. 38–45).: IEEE.

[99] Toulouse, T., Rossi, L., Campana, A., Celik, T., & Akhloufi, M. A. (2017). Computer vision for wildfire research: An evolving image dataset for processing and analysis. *Fire Safety Journal*, 92, 188–194.

[100] Ulku, I., Barmpoutis, P., Stathaki, T., & Akagunduz, E. (2020). Comparison of single channel indices for u-net based segmentation of vegetation in satellite images. In *Twelfth International Conference on Machine Vision (ICMV 2019)*, volume 11433 (pp. 1143319).: International Society for Optics and Photonics.

[101] Ullo, S. L. & Sinha, G. (2020). Advances in smart environment monitoring systems using iot and sensors. *Sensors*, 20(11), 3113.

[102] University of Wisconsin (2020). Mcidas-x user's guide. https://www.ssec.wisc.edu/mcidas/doc/users_guide/current/app_d-12.html. [Online; accessed 2021-02-17].

[103] Vadrevu, K. P., Lasko, K., Giglio, L., Schroeder, W., Biswas, S., & Justice, C. (2019). Trends in vegetation fires in south and southeast asian countries. *Scientific reports*, 9(1), 1–13.

[104] Vani, K. et al. (2019). Deep learning based forest fire classification and detection in satellite images. In *2019 11th International Conference on Advanced Computing (ICoAC)* (pp. 61–65).: IEEE.

[105] Vicsek, T. (1992). *Fractal growth phenomena*. World scientific.

[106] Von Neumann, J. (2016). Probabilistic logics and the synthesis of reliable organisms from unreliable components. In *Automata Studies.(AM-34), Volume 34* (pp. 43–98). Princeton university press.

[107] Von Neumann, J. (2017). The general and logical theory of automata. In *Systems Research for Behavioral Sciencesystems Research* (pp. 97–107). Routledge.

[108] Wagner, F. H., Sanchez, A., Tarabalka, Y., Lotte, R. G., Ferreira, M. P., Aidar, M. P., Gloor, E., Phillips, O. L., & Aragao, L. E. (2019). Using the u-net convolutional network to map forest types and disturbance in the atlantic rainforest with very high resolution images. *Remote Sensing in Ecology and Conservation*, 5(4), 360–375.

[109] Widrow, B. et al. (1960). *Adaptive" adaline" Neuron Using Chemical" memistors.".*

[110] Willis, K. & McElwain, J. (2014). *The evolution of plants*. Oxford University Press.

[111] Wyniawskyj, N. S., Napiorkowska, M., Petit, D., Podder, P., & Marti, P. (2019). Forest monitoring in guatemala using satellite imagery and deep learning. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium* (pp. 6598–6601).: IEEE.

[112] Xie, Z., Song, W., Ba, R., Li, X., & Xia, L. (2018). A spatiotemporal contextual model for forest fire detection using himawari-8 satellite data. *Remote Sensing*, 10(12), 1992.

[113] Yadav, N., Yadav, A., & Kumar, M. (2015). History of neural networks. In *An Introduction to Neural Network Methods for Differential Equations* (pp. 13–15). Springer.

[114] Yi, J., Wu, P., Jiang, M., Huang, Q., Hoeppner, D. J., & Metaxas, D. N. (2019). Attentive neural cell instance segmentation. *Medical image analysis*, 55, 228–240.

[115] Yu, L., Wang, N., & Meng, X. (2005). Real-time forest fire detection with wireless sensor networks. In *Proceedings. 2005 International Conference on Wireless Communications, Networking and Mobile Computing, 2005.*, volume 2 (pp. 1214–1217).: Ieee.

[116] Zhang, D., Han, S., Zhao, J., Zhang, Z., Qu, C., Ke, Y., & Chen, X. (2009). Image based forest fire detection using dynamic characteristics with artificial neural networks. In *2009 International Joint Conference on Artificial Intelligence* (pp. 290–293).: IEEE.

[117] Zhang, J., Zhu, H., Wang, P., & Ling, X. (2021). Att squeeze u-net: A lightweight network for forest fire detection and recognition. *IEEE Access*, 9, 10858–10870.

[118] Zhang, Q.-x., Lin, G.-h., Zhang, Y.-m., Xu, G., & Wang, J.-j. (2018a). Wildland forest fire smoke detection based on faster r-cnn using synthetic smoke images. *Procedia engineering*, 211, 441–446.

[119] Zhang, Z., Liu, Q., & Wang, Y. (2018b). Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5), 749–753.