Viktória Kabai

Application of object segmentation techniques (Objektum szegmentációs technikák alkalmazása)

Mathematics expert in data analytics and machine learning - Postgraduate specialist training thesis

Supervisors:
dr. András Zempléni
Department of Probability Theory and Statistics
Eötvös Loránd University, Faculty of Science
Institute of Mathematics

Szabolcs Szalánczi
Lead AI developer - B+N Referencia Zrt., Department of Robotics
Consultant - Hungarian Defence Forces, Medical Centre
Consultant - Smart Software Consulting Kft.
Consultant - Login Autonom Kft.

Eötvös Loránd University
Faculty of Science
2022

ACKNOWLEDGMENTS

This thesis was written in a quite short period and with this it is a testament to all of the lecturers' and teaching assistants' work through the whole year of the program. All of them did a remarkable job in explaining and teaching sophisticated and complex concepts of data science to a novice in the field.

I, also, want to express my deepest gratitude towards my mentor and thesis supervisor, Szabolcs Szalánczi, who showed the beauty and complexness of the field of deep learning computer vision, while teaching tremendous amounts of knowledge in the area in a very limited time.

Last, but not least, I want to thank my fiance, Péter Tóth, for reading through my notes and making suggestions on how to best present the knowledge I gathered in the process of writing this thesis as it always was the hardest part for me to do.

1. INTRODUCTION

Over the last couple of years advances in artificial intelligence and deep learning have been remarkable. All of this could happen because of the improvements in hardware and the before unimaginable amounts of data we create on a daily basis.

The image processing sector in itself was able to make great progress and in some cases surpass humans in some tasks related to detecting and labelling objects. Along with tremendous amounts of visual data (more than 3 billion images are shared online every day) [75] that made this progress possible, the computing power required to analyse this amount of data is now much more easily accessible than before. The accuracy of available models rose significantly too, what was a decade ago 50% accurate now is 99% , which makes the model more accurate than humans. [75]

In the thesis, we discuss the history of the image segmentation field and the stages it went through before landing on deep learning algorithms. Next, we cover in detail some of the milestone algorithms that shaped the research around image segmentation and the circumstances made these advances possible. In the last part, some experiments are made on implementing the discussed algorithms from scratch and importing them via transfer learning. The goal is to deepen the understanding of their structures and ways of adapting them to our use cases..

2. BASIC CONCEPTS OF IMAGE SEGMENTATION

2.1. Definitions of Computer Vision and its Applications

Computer vision is an interdisciplinary field that mainly focuses on problems around high-level understanding of human vision and replicating parts of its complexity by computer systems, enabling such systems to identify and process objects or full images and videos the same way humans do. It is the automatic extraction, analysis and understanding of information from images. It also involves scientific development of theoretical and algorithmic principles to achieve visual understanding by computers. [1, 2]

Computer vision began to develop in the 1960s at universities pioneering artificial intelligence development. To become a stepping stone for intelligent robots it was meant to replicate human vision. At that time, it seemed achievable through a student summer project. [3, 4] Research in the next decades laid down the basics of today's algorithms, including edge extraction from images, inference of shape from shading, application of graph cut variations, statistical learning for face recognition, etc. [1, 4]

In the more recent years the field saw a resurgence of feature-based models used together with machine learning algorithms and sophisticated optimization techniques. The advancement of deep learning capabilities brought a new life to the field of computer vision. These algorithms have surpassed all previously known computer vision techniques in accuracy for tasks in classification, segmentation and others. [1, 5, 6]

Examples of applications of computer vision:

- Automatic inspection of manufacturing applications;
- Assisting in animal species identification tasks;
- Visual surveillance or people counting in the tourism industry;
- Medical image analysis or topographical modelling;
- Autonomous vehicle navigation;
- Tasks related to Augmented Reality experiences.

One of the basic problems of computer vision is to determine whether or not the image contains a specific object, feature, activity. There are different subtypes of this problem, such as: object classification, where one or several learned objects are recognized with their position on the image; identification of individual instances of an object recognized (specific faces or fingerprints); detection of specific conditions, like possible abnormal cells or tissues in medical images, roadblocks for automatic vehicle navigation.

Some other types of computer vision problems are estimation of the orientation or position of an object relative to the camera, written or printed character recognition, QR code recognition, facial recognition, shape recognition, motion analysis for tracking movements of sets of points, such as humans or vehicles, 3D scene reconstruction, image restoration with the aim of removing noise, blur. [1]

Deep learning has become increasingly popular in the field of computer vision and we are at the stage where, in some cases, it works as accurately or even more accurately than humans. Currently, the most prevalent problems that computer vision researchers are most interested in solving are, in increasing order of difficulty, image classification, object detection and segmentation.

In the case of image classification we are interested in getting the labels of all the objects that are present in the image. In object detection we try to identify, along with all object labels present in the image, the approximate location of the objects present with the help of bounding boxes. Image segmentation takes it to a further step by looking to find accurately the exact boundary of the objects in the image.

Image segmentation has 2 main types:

1. Semantic segmentation is the process of classifying each pixel to a particular label. It doesn't identify different instances of the same object.
2. Instance segmentation, contrary to semantic segmentation, gives a unique label to every instance of a particular object in the image.

\+ Panoptic segmentation, unifies the 2 above, semantic and instance segmentation, it classifies all the pixels in the image as belonging to a class label and also identifies what instance of that class they belong to. [7]

2.2. Overview of Early Image Segmentation Approaches

In computer vision, image segmentation means the process of dividing an image into image segments, such as regions, objects or sets of pixels. Image segmentation is usually used to locate objects and their boundaries. More accurately, image segmentation is the process of assigning labels to an image’s every pixel such that pixels in the same class share certain characteristics. As a result of image segmentation we get segments that together cover the entire image, or contours extracted from the image. Pixels in a region share similarities in some characteristics or computed properties, it can be colour, intensity, texture, etc. Neighbouring regions have significant differences with respect to the same characteristics. When applied to stacks of images, like medical images (CT scans for example), the resulting boundaries of image segmentation can be used to create 3D reconstructions with the help of other computer vision algorithms [8, 9]

Practical applications of image segmentation include:

- Content-based image retrieval;
- Machine vision;
- Medical imaging: locate tumours and other pathologies, measure tissue volumes, diagnosis, study of anatomical structure, surgery planning and simulation, intra-surgery navigation;
- Object detection: pedestrian, face and brake light detection, locate objects in satellite images (roads, forests, etc.);
- Recognition tasks: face, fingerprint and iris recognition;
- Traffic control systems;
- Video surveillance.

There are two classes of segmentation techniques.

- Classical computer vision approaches
- AI based techniques

The simplest image segmentation method is the thresholding method. It is based on setting a threshold value on the original image's pixel intensity to turn it into a binary image or multi-color image. [10] Some popular methods used in industry include the maximum entropy method, balanced histogram thresholding, Otsu's method (maximum variance), and k-means clustering. [8]

The K-means algorithm is an iterative technique and an unsupervised machine learning technique that partitions an image into *K* clusters. [11] The basic algorithm is:

1. Pick *K* cluster centroids, either randomly or based on some heuristic method, for example K-means++
2. Assign each pixel in the image to the cluster that minimises the distance between the pixel and the cluster centre
3. Re-compute the cluster centres calculating the mean of all the pixels in the cluster
4. Repeat steps 2 and 3 until convergence, meaning no pixels change clusters. [12]

Motion based segmentation relies on motion in the images to perform segmentation. It looks at the difference in images and detects the object by the difference in the 2 images examined. [12]

Compression based methods try to find any patterns and regularity in the images to use it for compression. This way every segment of the image is described by the texture and boundary of the shape. [12]

Histogram-based methods are efficient because they go through an image only once. A histogram is computed from all pixels in an image and the peaks and valleys of the histogram are used to locate clusters of colour or intensity. This method can be iteratively to divide the image into smaller and smaller clusters. [12]

Edge detection is a well-developed image processing method, and is often used as a base for other segmentation techniques. Usually boundaries and edges of the regions in an image are closely related and there is often a sharp change in intensity on the edges of such regions. [12]

Region growing methods are based on the assumption that neighbouring pixels in the same region of the image have very similar values, so the usual way of region growing methods is to compare pixels with their neighbours. For defining similarity a criterion needs to be set and the whole result depends on it heavily. Several region growing methods exist: statistical region merging, seeded/unseeded region growing, lambda connectedness. [12]

Graph partitioning methods model the impact of one pixel neighbourhood on a cluster of pixels or one pixel assuming homogeneity of images. Here we think of images as weighted and undirected graphs. A group of pixels with their nodes and edge weights define the similarity or dissimilarity between neighbouring pixels. After this the graph is segmented based on some criteria defining a cluster and the output of this step is considered an object. [12]

3. DEEP LEARNING IMAGE SEGMENTATION

3.1. Milestone algorithms and their development

Most of the before mentioned segmentation methods are based on colour and/or intensity information of image pixels. Contrary to this, humans use a much broader spectrum of knowledge when performing image segmentation, but using this kind of knowledge would need considerable amounts of engineering and computational time, and would require a huge database with domain knowledge that does not exist currently. Trainable segmentation methods, such as methods based on different types of neural networks, help overcome these issues by modelling the knowledge from a dataset of labelled pixels. [8]

Neural networks in simple terms are algorithms that try to mimic the way human brains work to find relationships between data points. In the case of image segmentation some of the neural networks can try to understand small areas of the images to extract simple characteristics or features such as edges. Different neural networks can then try to combine these extracted features to label the pixels of the images accordingly.

Currently, the best semantic segmentation algorithms are based on convolutional neural networks. To understand their capabilities we are going to look through the development of algorithms by different computer vision competitions such as ImageNet Large Scale Visual Recognition Challenge that often drove the improvement of current methods. ImageNet Large Scale Visual Recognition Challenge is a benchmark in object classification and detection, with millions of images and 1000 object classes used in the competition. The performance of different types of convolutional neural networks on the ImageNet dataset is now close to human vision. [13]

Before 2000, we used several methods in image processing such as threshold segmentation, region based methods, edge detection, texture features, clustering etc that were mentioned in the previous section. From 2000 to 2010, there were four main methods in image processing: graph theory, clustering, classification and combination of clustering and classification. [8]

Since 2010, the rise of neural network models and the development of deep learning involve several popular models, discussed below.

In this period, convolutional neural networks (CNNs) became a very efficient visual processing tool because it can learn hierarchical features. The specifics of CNNs are discussed in the next section when outlining the building blocks of modern deep learning image processing algorithms. [14]

LeNet / AlexNet / ZFNet

LeNet was developed by LeCun in 1998 and it is a 7-level convolutional neural network. It was used to classify digits by several banks to recognise hand-written numbers on digitised cheques (32x32 pixel grayscale images). To process higher resolution images it needed a larger and more deeper CNN model, because of this the model was constrained by computing power, and it was difficult to implement the model until the 2010s. [15, 16, 17]

However, LeCun proposed a CNN model with back-propagation algorithm and his experiments created a state-of-the-art model. The model's architecture is known as LeNet-5 with 2 convolution layers, 2 sub-sampling layers and 2 fully connected layers with an output layer of Gaussian connection. [15]

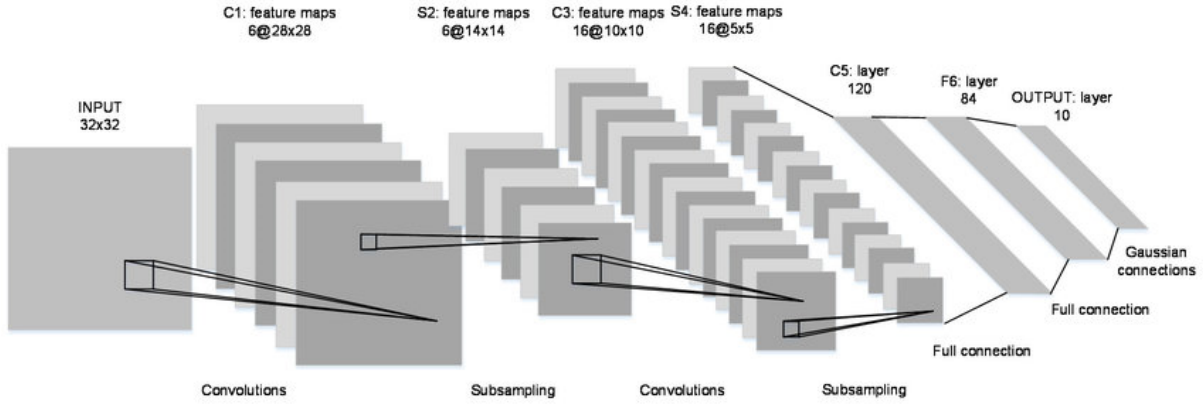

Fig. 1. Architecture of LeNet [16]

The number of weights trained are 431K and the number of Multiply and Accumulates (MACs) are 2.3M. [15]

AlexNet is believed to be the first model (paper) that sparked interest towards CNNs when it won the ImageNet 2012 challenge. It was developed by Alex Krizhevsky and others based on LeNet, but it was revised to be a more deeper and wider CNN model and achieved state-of-the-art accuracy against other, traditional machine learning and computer vision algorithms. [15, 16] The model was trained on ImageNet for 6 days straight on 2 GPUs. [17]

The architecture of AlexNet is shown in Fig. 2. The first convolution layer and max pooling consists of 96 features and is of size 11x11. Max pooling is performed with 3x3 filters and a stride of 2. The second layer consists of the same operations with 5x5 filters. In the third, fourth and fifth layers filters of size 3x3 are used with 384, 384 and 296 feature maps respectively. At the end 2 fully connected layers (FCs) are used with dropout and a softmax layer. The complete model consists of 2 similar structure networks with the same number of features trained in parallel. [15] It also uses ReLu activations and image augmentations.

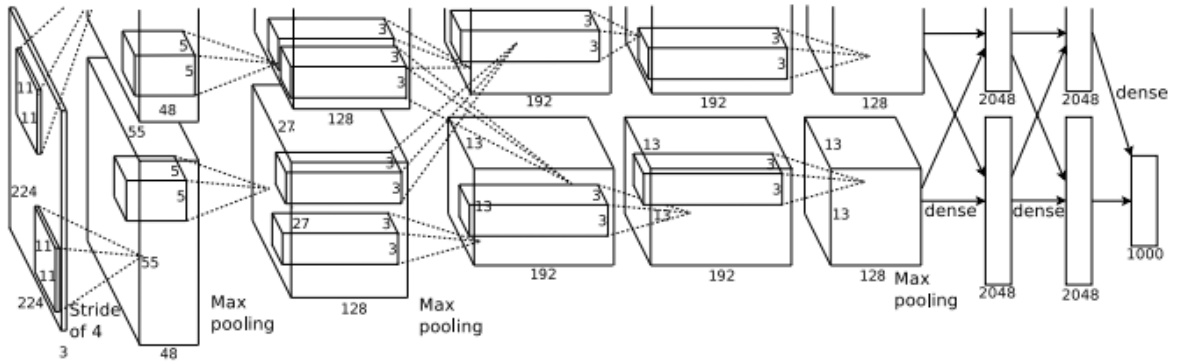

Fig. 2. Architecture of AlexNet [17]

The total number of weights and MACs trained and calculated for the whole network are 61M and 724M respectively. [17]

The next development stage of CNN structures was the ZFNet algorithm in 2013. Matthew Zeiler and Rob Fergue won the 2013 ILSVRC with this extended AlexNet architecture called ZFNet. It gives better accuracy compared to AlexNet by tweaking its parameters. [15] One of the main differences is that ZFNet uses 7x7 filters instead of 11x11 filters, with the idea that smaller filters retain more information about the image, it also reduced the number of parameters drastically and improves accuracy. This algorithm also uses ReLu activations and is trained by stochastic gradient descent. [16, 19]

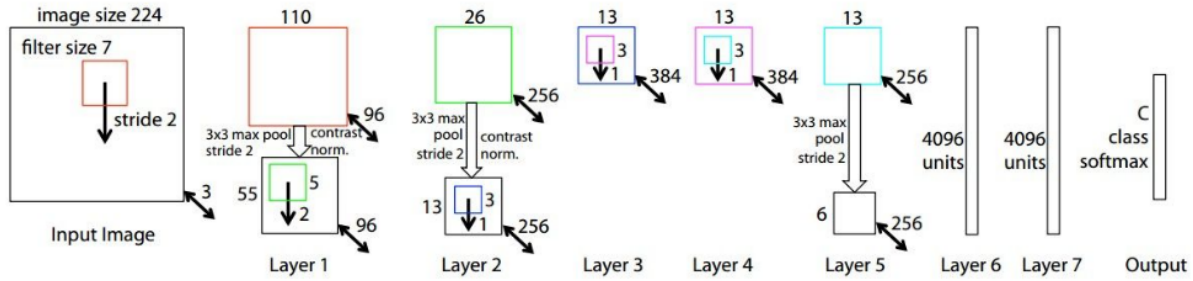

Fig. 3. Architecture of ZFNet [16]

GoogLeNet (Inception) / VGGNet

The winner of 2014 ILSVRC was a model proposed by Christian Szegedy from Google. The main focus was on reducing the complexity of computation needed for the algorithm compared to traditional CNNs. The GoogLeNet model proposed a new module called inception module that includes skip connections in the structure forming a mini module that are repeated throughout the network. [16] The inception modules contained variable receptive fields with operations that captured sparse correlation patterns in the new feature map stack. [15]

GoogLeNet consists of 22 layers, which was much higher than any previous network had, but the number of parameters was much lower than in AlexNet. [15] It uses 9 inception layers and it omits all fully connected layers by using average pooling. [16]

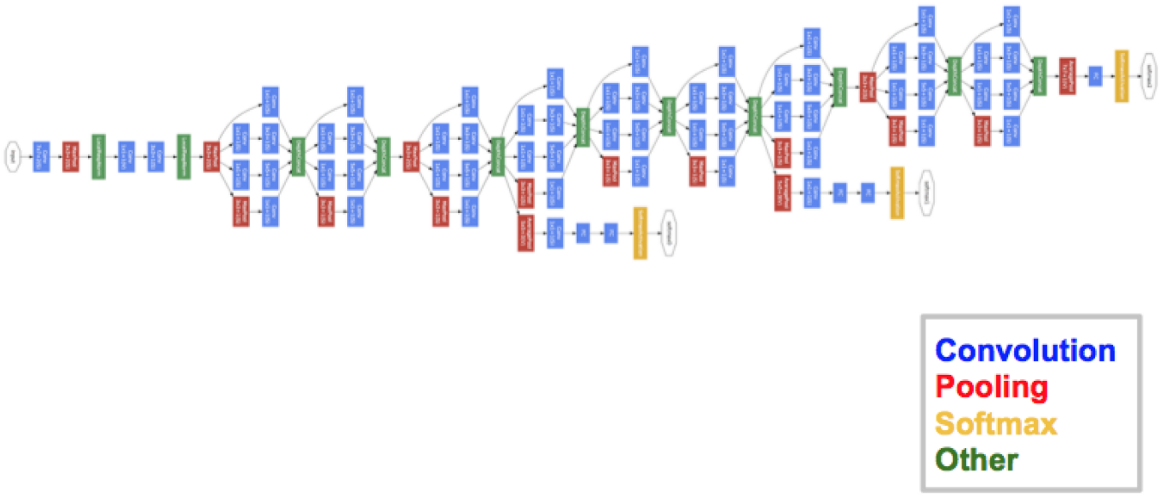

Fig. 4. Architecture of GoogLeNet (Inception V1) [17]

GoogLeNet had 7M parameters, which is much lower compared to AlexNet's 61M. [15]

2014 was very strong in producing state-of-the-art algorithms with ILSVRC 2014 runner-up VGGNet also introduced in that year. VGGNet was developed by Simonyan and Zisserman and it uses 3x3 filters compared to AlexNet's 11x11 and ZFNet's 7x7. [16] It has a much more appealing structure with very similar building blocks throughout the model. The authors intuition behind the smaller filter sizes was that 2 consecutive 3x3 filters give the effect of 5x5 field and 3 consecutive layers of 3x3 filters give a field of 7x7 filters. This way we have far fewer hyperparameters to train [15, 19] and the VGG versions (11, 16, 19) are currently one of the preferred choices for feature extraction in the image processing community. [17]

VGG architecture consists of several blocks of 2 or 3 convolutional layers with ReLu activations, followed by a max pooling layer and several fully connected layers also using ReLu activation. The final fully connected layer uses softmax activation. [15] The exact VGG16 and VGG19 architectures are described in more detail in the next chapter.

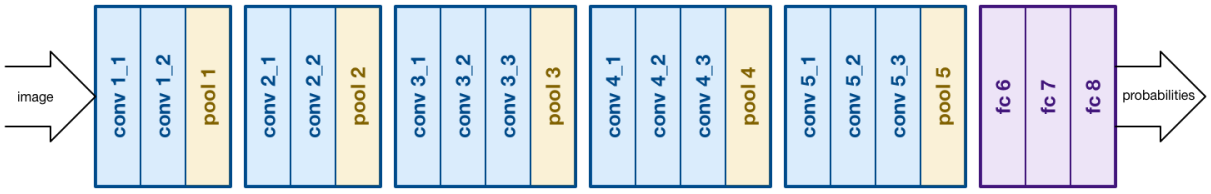

Fig. 5. Architecture of VGGNet [17]

ResNet

Next year's ILSVRC (2015) winner was the Residual Network Architecture (ResNet) by Kaiming He. The developers intent was to design an ultra-deep network that did not suffer from vanishing gradient problem, they also managed to show that adding new layers decreases the error rate. [15] It took 2-3 weeks to train the model on an 8 GPU machine, since the ResNet has several different setups (34, 50, 101, 152 and even 1202 layers deep). [16]

ResNet introduced the architecture with skip connections, authors proposed a pre-activation variant residual block in which the gradients can easily flow through the shortcut connection without obstruction during the back pass of back propagation[23]. Because of these they were able to train a 152 layer structure while having lower complexity than a VGGNet. [17]

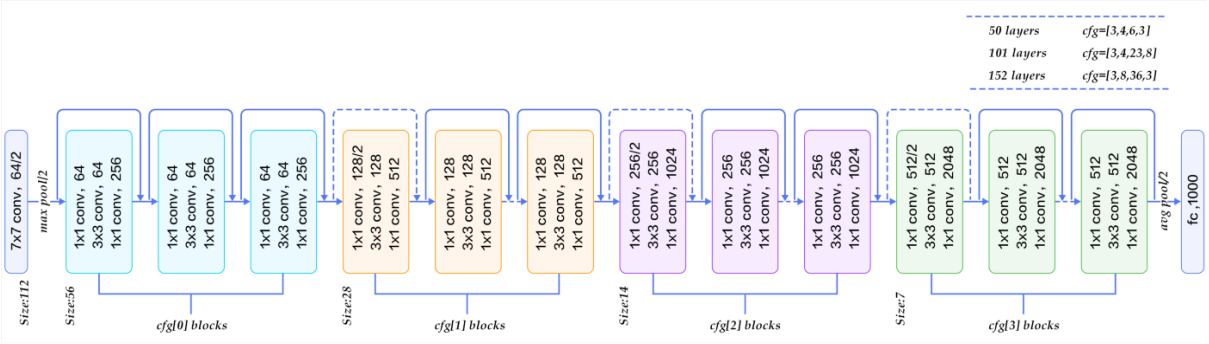

Fig. 6. Architecture of ResNet [17]

The total number of weights and MACs of the whole structure are 25.5M and 3.9M respectively. [15]

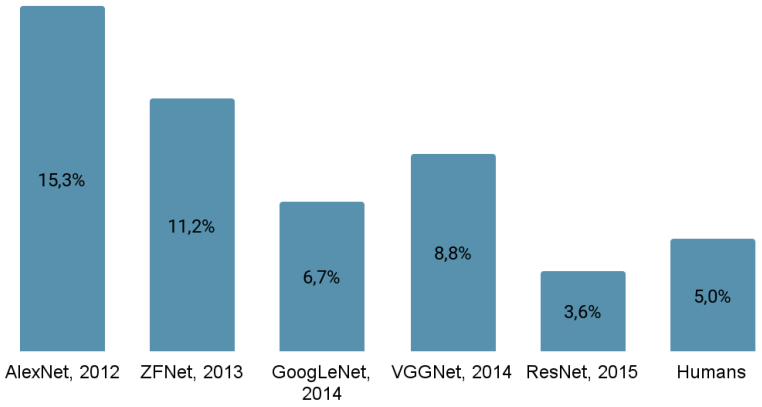

Fig. 7. Summary of Error Rates of discussed algorithms [20, 21, 22, 23, 24, 25]

The mentioned structures are just an extract from all of the existing algorithms and papers that developed the field of deep learning image processing. There are many newer and more accurate models developed every year, but these 5 were the stepping stones to the current state of the field. In the next section we will outline the basic parts of these algorithms and their functionality.

3.2. Building blocks of Image Segmentation Deep Learning algorithms

CNN

Convolutional neural network, in deep learning, is a version of artificial neural networks most often applied to analyse and extract information from images. CNNs are inspired by biological processes - animal visual cortex's architecture is very similar to the connectivity pattern between the network's neurons. Individual living neurons (cortical neurons) respond to stimuli in a restricted area of visual field known as the receptive field, these fields partially overlap for different neurons thus covering the whole image or visual field. [26]
CNNs have several advantages over other algorithms, like being more similar to human vision than any other structure, their max pooling layers can effectively keep shape variations. Most importantly, CNNs are trained with gradient-based learning algorithms and suffer less from the diminishing gradient. CNNs can produce highly optimised weights because it trains the whole network to minimise error criterion directly. [23]

The architecture of CNNs overall consists of 2 main parts: feature extractors and a classifier. In the first part, each layer's input is coming from the previous layer's output. The usual CNN architecture is a combination of 3 different types of layers: convolution, pooling and classification. The output nodes of convolution and pooling layers are collected into a group called feature mapping. Each node gathers features from the images by using convolution computation on the inputs. As features are propagated to the highest layers the sizes are reduced by the kernel size of the convolution and pooling. The output of the last CNN layer is used as input for a fully connected network called classification layer. Here, the desired number of features is selected as input keeping in mind the desired dimensions, but these layers are expensive in computations. Recently, different methods are used for classification alternatively to fully connected layers, like average pooling with softmax scoring after that. [23]

Convolution layer

The biological process of image recognition is very similar to what the CNNs do in their convolutional layer(s). The network performs a mathematical operation called convolution instead of matrix multiplications. Convolution is an operation on 2 functions that produces a third function; this third function expresses how the shape of one function is modified by the other. [27] The input of the first convolution layer is a tensor shaped: number of inputs+input height+input width+input channels. After going through the convolutional layer, the image is transformed into a feature map shaped: number of inputs+feature map height+feature map width+feature map channels. [26] After the first layer, feature maps are convolved with learnable kernels, their output goes through an activation function (sigmoid, softmax, relu etc) to form an output feature map that will be used as input for the subsequent layer. [23]
This way, using regularised weights with fewer parameters, we avoid vanishing or exploding gradients during backpropagation that happen with traditional neural nets. [26]

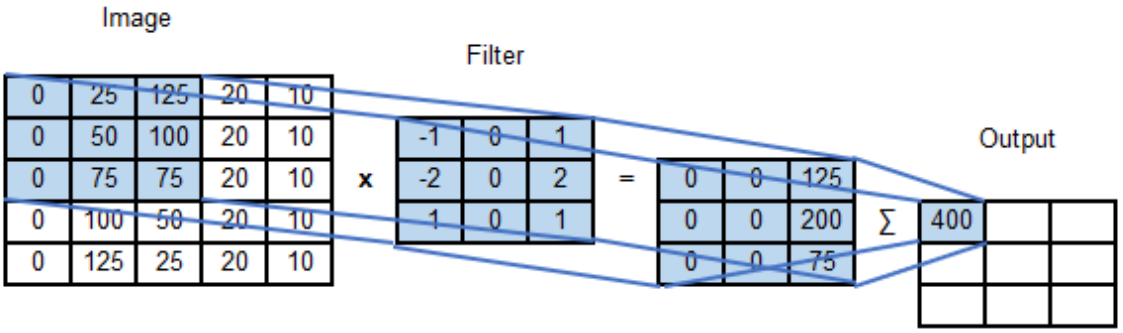

Fig. 8. Explanation of the convolution operation in CNNs

Sub-sampling layer (Pooling layer)

The sub-sampling layer performs the downsizing of the input maps. This layer is generally known as the pooling layer. In this layer the number of input and output feature maps do not change, however the size of each output map's dimension will be reduced due to the downsampling. The amount of this change is defined by the size of the downsampling mask (kernel). There are 2 main types of layers, according to the operations performed in this layer: average pooling and max pooling, but there are many more. In the case of average pooling, the function usually sums up the feature maps by the given kernel size and takes their average value. In the max pooling layer's case, the layer picks the highest value from that kernel-sized patch of feature maps. Both reduce the size of the output feature map by the size of the kernel. [23, 26] There are also many more pooling layers, like minimum pooling, adaptive pooling, weighted pooling and so on.

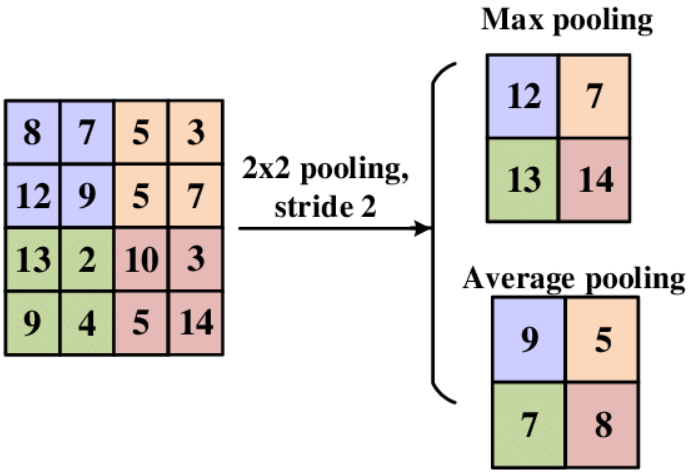

Fig.9. Explanation of the pooling layers [29]

Classification layer (Fully connected layer)

This is the fully connected layer that computes the score of each class from the features the network extracted in the previous layers and classifies the image or pixel. Fully connected layers connect every neuron in one layer with every neuron of the next layer. A flattened

matrix, which is the output of the last layer as a vector representing the feature maps, goes through this layer and classifies the image or the pixel. These layers are used as softmax layers. [23, 26]

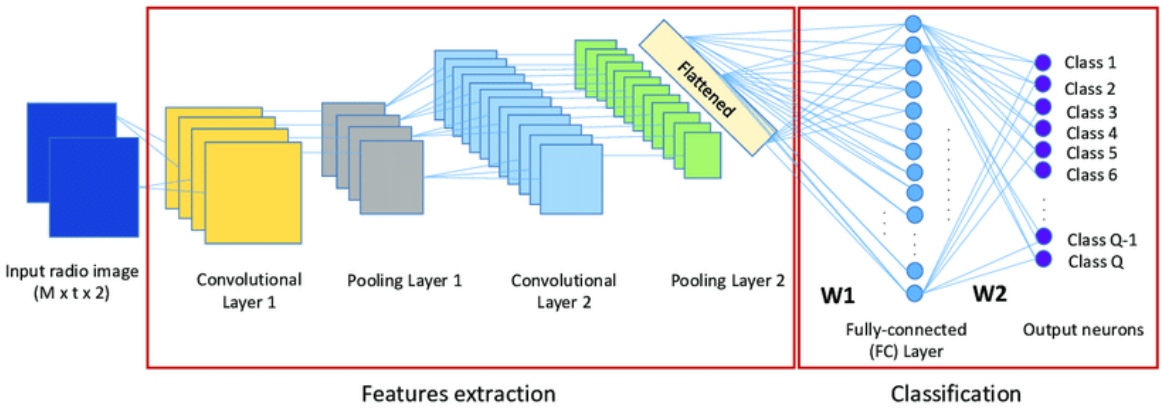

Fig. 10. Explanation of CNNs with feature extraction and classification parts highlighted [30]

Activation functions (ReLU, Softmax, Sigmoid)

Each neuron in a neural network passes on a value to the next layer as an output by applying a specific function which we call activation function. This value is often called a weight and a collection of weights and biases in a vector represent a feature map, mentioned above. These weights and biases are called filters, and learning consists of iteratively adjusting these biases and weights. One specific characteristic of CNNs is that many neurons can share the same filters, this reduces memory and computational needs as a single vector can represent all neurons of a receptive field. [26]

Each neuron creates a weighted sum of its inputs and passes it through an activation function. There are many different activation functions, but we are going to discuss only 3 used in deep learning algorithms VGGNet, ResNet and U-Net.

ReLU is short for rectified linear unit, which is often used for every layer in between the network, except for the last, classifier one. It was proposed in 2010 and solves the vanishing gradient problem for deep learning. It is based on the concept of keeping all values above zero and setting negative values to zero. [23]

$$g(x) = max\{0, x\}$$

For classification purposes sigmoid function is often used as an activation function. This function is specific to the cases of binary classification and its output is either 0 to 1 , or -1 to 1. These values can be interpreted as Yes/No answers or binary decisions. [26, 32]

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

Softmax activation function is a mathematical function that converts vectors of numbers into vectors of probabilities and is often used as an activation function of the classification layer of neural nets to normalise the output of the network into probability distribution over predicted output classes in a multiclass classification problem. Before applying softmax some values might be negative or not add up to 1, this is taken care of by the softmax activation. [26, 31]

$$g(x) = \frac{e^{x_i}}{\Sigma_{j=1}^{J} e^{x_j}}$$

Batch normalisation

Batch normalisation helps speed up deep learning by reducing internal covariance of the data. Input data is transformed to have zero mean and unit variance. By doing so the network converges faster and shows better regularisation during training and achieves higher accuracy. To improve the training process batch normalisation applied within the training process between layers of deep learning algorithms. [23]

Dropout

One of the methods to reduce overfitting, especially in fully connected layers, is dropout. During the training process specific nodes of the neural network are either dropped out with probability $1 - p$ or kept in with probability p. After this a reduced network is left, nodes coming out of and going into the dropped neuron are also removed. Only the reduced network is trained on the data in this stage and the dropped neurons are reinserted into the network with their previous weights after. Besides decreasing overfitting it also improves training speed by reducing the nodes trained at a time, and also leads to nodes picking up more robust features to better generalise images. [26]

Metrics (Pixel Accuracy, Intersection over Union)

Pixel accuracy is the most basic metric that is available to us in validating our segmentation results. Accuracy is obtained by taking the ratio of correctly classified pixels with respect to total pixels

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

where TP is true positive, TN is true negative, FP is false positive and FN is false negative.

The main disadvantage of using this metric is the result might look good in case of imbalanced data. Say for example the background class covers 85% of the input image we can get an accuracy of 85% by just classifying every pixel as background. [33]

IOU (Intersection Over Union) is defined as the ratio of intersection of the ground truth and the predicted segmentation results over their union. In case of multiclass classification IOU is calculated by taking the mean of IOU of the classes applicable. Compared to the pixel accuracy above it is a metric in case of binary classification (object and background) since the IOU value will be $\frac{\frac{90}{100}+\frac{0}{100}}{2} = 45\%$ IOU, which gives a better representation than 90% pixel accuracy. [34]

For imbalanced datasets, mean IOU might be misleading, so weighted IOU is introduced as an improvement. If one class dominates most of the images in a dataset (background), it needs to be weighed down compared to other classes. Thus instead of taking the mean of all the class results, a weighted mean is taken based on the frequency of the class in the dataset. [34]

Loss functions (Cross Entropy Loss, Dice Loss)

Loss function is used to guide the neural network towards optimization. It specifies how to penalise deviations of the predicted values from the ground truth.

Simple average of cross-entropy classification loss for every pixel in the image can be used as an overall function. Similarly to the metrics above, the simple average suffers from imbalanced data and it is advised to use weights. [33]

Dice loss function is a widely used metric to calculate similarity between 2 images. It addresses the imbalance problem in case of background/foreground imbalance problem. This loss function directly tries to optimise F1 score, that is accuracy based on the harmonic mean of precision (the number of true positives divided by the number of all positives, true and false) and recall (the number of true positives divided by the number true positives and false negatives). Similarly direct IOU score, it can be used to run optimization as well. [33, 35]

In the next chapter, experiments are conducted on open source data using algorithms and deep learning techniques described above.

4. APPLICATION OF SOME DEEP LEARNING ALGORITHMS

This chapter is dedicated to discussing the implementation of some of the above-mentioned algorithms and their building parts, reviewing the results of such experiments and drawing conclusions of the project.

4.1. Data Description

Nowadays, there are many different open source datasets that can be used for different machine learning problems. With the advance of modern computing and affordable cameras there are millions of pictures openly available on the internet and ready to use for solving computer vision problems with deep learning.

For our experimental purposes, we are using the COCO (Common Objects in Context) dataset that is a large scale object detection, segmentation and captioning dataset with among others object segmentation features, 330K images (>200K labelled), 1.5 million object instances and 80 categories. [36]

In this project we use COCO API [37, 38, 39] for accessing the 2014 train and validation data. For most of the algorithm experiments data is filtered down to the ‘cat’ category which has 2818 train images, 1480 validation images.

We load the data into the models with the help of data loaders, since the limited memory capabilities available at hand prevent us from loading the whole dataset into the GPUs memory and still have it train the model at the same time. We use data loaders to make batches of images that are easier processed by the model at one time, in our experiments we either use batches of 10 (for U-Net) or 5 (for VGGs and ResNets) images to train the models at a time. The images with their original masks are loaded into a generator object that is later used as train and validation data by the models.

For visualising test image results we are using self-created and labelled cat images. Labelling of the images was done by using the labelme API. [41, 42] The labelling process itself is quite time consuming given that the images have to be labelled one-by-one by selecting a closed area of the images as our class in question.

Fig. 11. Examples of training images of COCO dataset with their masks

Fig. 12. Examples of validation images of COCO dataset with their masks

Fig. 13. Examples of self-created and masked test images with their masks

4.2. VGGNet

The basic structure of VGGNet was already discussed in Section 3. Briefly, VGGNet consists of very similar building blocks over the whole structure. We are going to use VGG16 and VGG19, where the numbers denote the number of layers involved in the algorithm.

In the case of VGG16 with 2 convolutional layers with 3x3 kernel and 64 features, followed by a max pooling layer, then we repeat the block with the same layers and filters, only changing the features to 128. Next we add an additional convolutional layer of 3x3 kernel size to the block, meaning 3 conv layers per block + max pooling, with 3x3 kernel and 256, 512 and 512 features spread across 3 blocks. Lastly we have 2 fully connected layers with size 4096, 1 fully connected layer with size 1000 and a softmax layer with size equal to the number of classes we want to predict.

The VGG19 differs from the above only by adding an additional 3x3 convolutional layer to blocks 3-5, bringing the number of conv layers in those blocks up to 4, kernel sizes and number of features stay consistent across blocks and the same as VGG16. [21, 43]

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Fig. 14. Layer structure of all VGG networks [21]

First, we implement VGG16 building it from scratch in tensorflow's Keras API based on the original structure, later we do the same manipulations on the VGG19 structure to match our data and purpose.

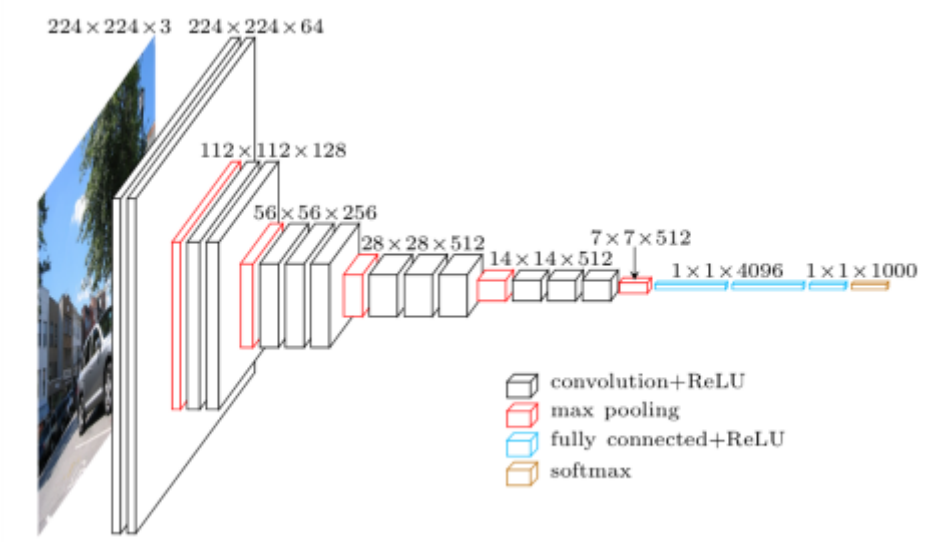

Fig. 15. Structure of VGG16 with image sizes [44]

We use 256x256x3 sized images as our inputs and our goal is to do binary semantic segmentation on cat images from the COCO dataset, which means we need to implement several changes to the structure. First, we adjust the input sizes to our needs, after all convolutional layers and flattening the output in the essence of available time, computing power and memory we use batch sizes of 5 and learning rate of 1e-5, we also contract the fully connected layers to size 1000, the last layer to 256*256 size fully connected layer to be able to reproduce the images pixel-by-pixel classification and reshape the output of the last layer to return to 2D matrix shape tensor instead of a vector of 65536 length for it to be comparable with the original masks of the images. [45, 46, 47]

We train the networks for 25 epochs, for the same reason of economy of time and availability of GPU and memory.

To point out, these experiments are solely used for deeper understanding of the structure of different image segmentation algorithms, the goal wasn't to outbest them in any way, hence the mediocre model accuracies in some cases that we were, in fact, expecting.

Our VGG16 results are not too high with validation accuracy of around 0.813 and test accuracy of 0.914 and image results shown below.

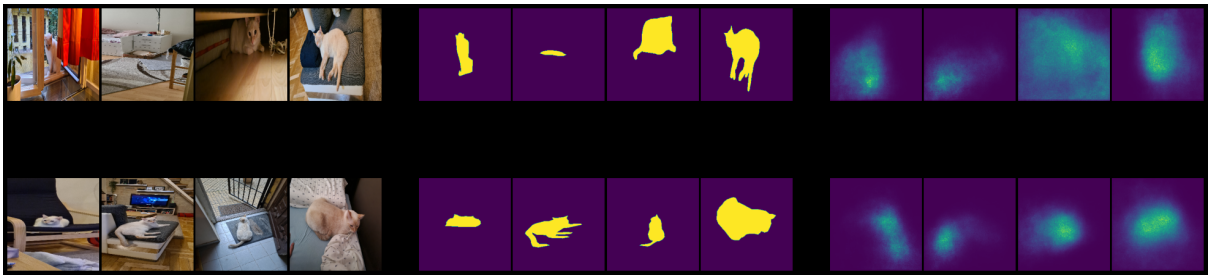

Fig. 16. Results of VGG16 model built from scratch

The VGG19 results are almost identical to the VGG16 results seen above at least in terms of accuracy numbers, with validation accuracy of 0.814 and test accuracy of 0.914.

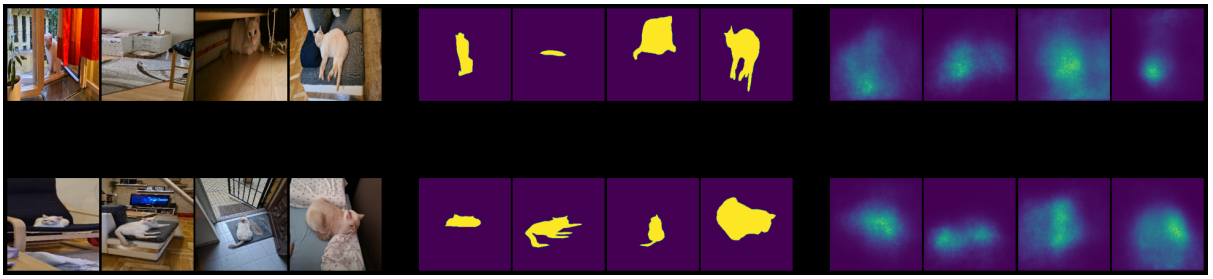

Fig. 17. Results of VGG19 model built from scratch

As we can see the accuracy here does what we discussed in the previous section, since the image masks are unequally favouring the background, misclassifying a lot of pixels still can produce high accuracy. The VGG19 visually does a little better job at identifying the area in which the cat is situated, VGG16 selects too wide ranges of pixels.

Amongst solutions for improving the performance of the models can be implementing a higher number of training epochs, using more training images. The models already implement dropout and batch normalisation layers after each convolutional layer.

4.3. ResNet

The general structure of ResNet was already discussed in the previous section, here we are going to focus on specific versions of ResNet, namely ResNet 50 and ResNet101. The number in the names refers to the number of layers included in the network.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig. 18. Layer structure of all ResNet versions [20]

These models are also compiled by different block convolutional layers, the difference compared to the VGG networks is that the blocks contain different kernel sizes and feature numbers, and also in the fact that this model has skip connections (or shortcut connections as in the original paper). [20]

For our purposes of binary segmentation and relatively small of computing power we change the fully connected layer from size 1000 to 125 and the last softmax layer to be of size 256*256 to get back the size of our original images. Here we also reshape the output to a 2D tensor from the vector output of the fully connected layer, to make it comparable to the original image masks. [20, 48, 49, 50, 51, 52]

Here we also operate with 5 image batches, 1e-5 learning rate, 25 epochs. As previously we use binary cross entropy loss and accuracy metric.

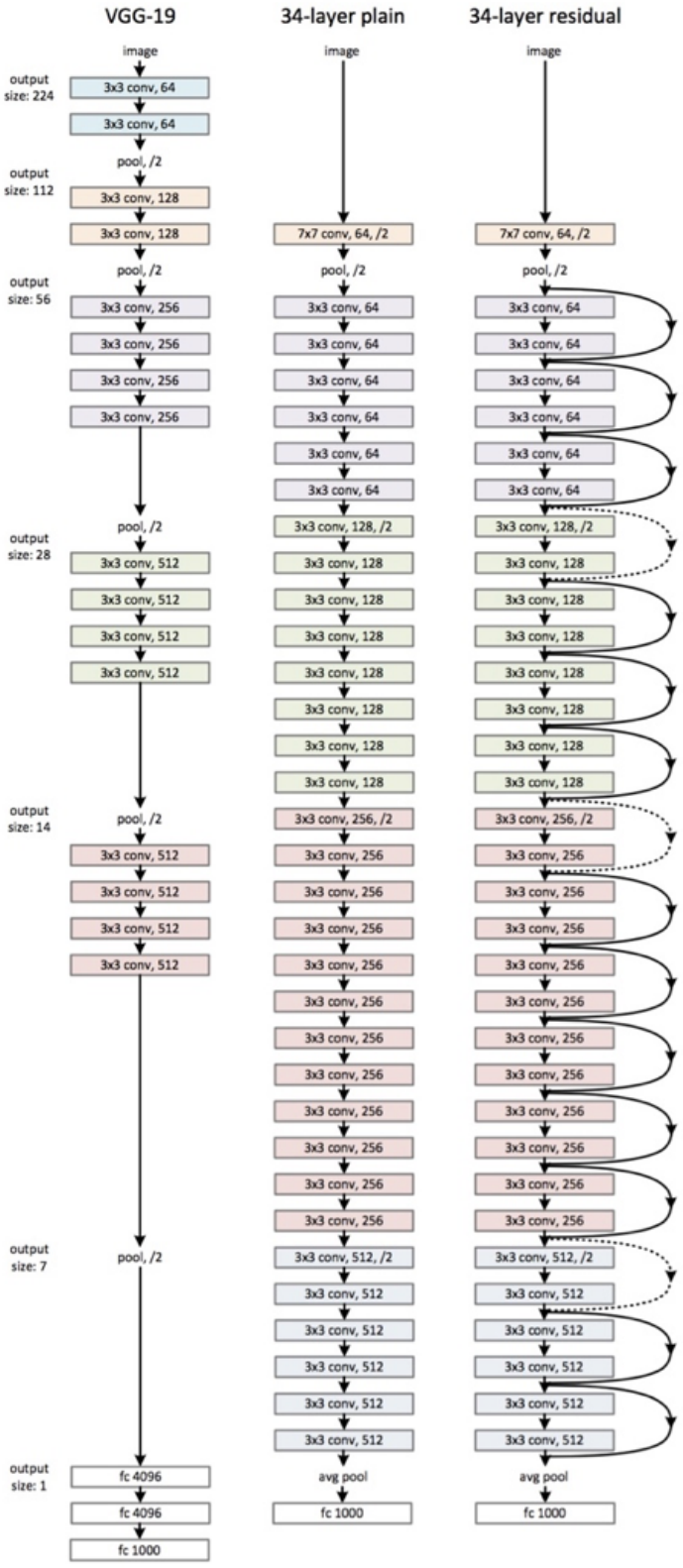

Fig. 19. Structure of ResNet34 compared to VGG19 and a simple 34 layer CNN [48]

ResNet50 achieves almost the same accuracy as our VGG models, with 0.812 validation and 0.914 test accuracy.

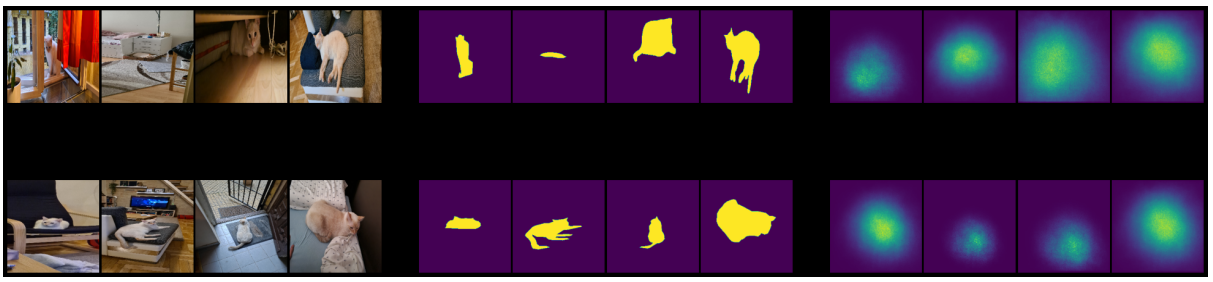

Fig. 20. Results of ResNet50 model built from scratch

ResNet101 based on accuracy numbers improves the performance compared to previous models, but the change is not meaningful in terms of visual appearance, validation accuracy reaches 0.815 at one point and test accuracy is still 0.914.

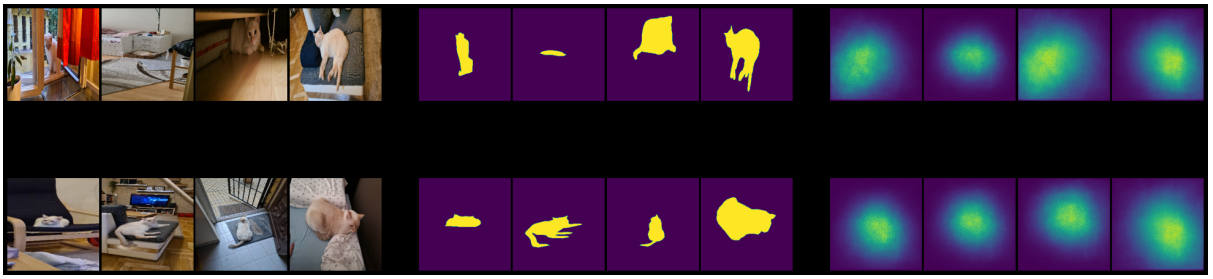

Fig. 21. Results of ResNet101 model built from scratch

These 2 models also use batch normalisation throughout their building blocks and produce a visibly more rounded edge area detection compared to the VGG models above. Significant improvements can be made by longer training periods and having a lot more training images than ~3000, the limited accuracy is partially due to the fact that we needed to contract the dense layer's size, as mentioned above. As mentioned in the VGG model section, these results were expected.

4.4. U-Net

U-Net is the only model not discussed in the previous section, we are describing it here. U-Net is a convolutional neural network that was developed for medical image segmentation (to find tumours in the brain and lungs) at the Computer Science Department of the University of Freiburg. It is based on the fully convolutional network (only convolutional layers added, no dense layers as previously seen in the models) and the architecture specifically was modified to work with fewer images and make more accurate predictions. [53] The network trains very fast on modern GPUs, with an image of size 512x512 it takes less then a seconds to do the segmentation. [54]

The U-Net architecture was developed by Olaf Ronneberger et al and the original architecture has 2 parts, one is the contraction or encoder part which tries to capture the context of the image with traditional stack of convolutional layers and max pooling layers, the second part is a symmetric expanding or decoder part which is responsible for localization of object using convolutions and upsampling. There are also concatenations added between the respective encoder and decoder blocks to keep as much information about the images as possible and

fight the information loss downsampling may have caused in other modells. [54, 55] It also makes possible to use pixel position information more precisely as dense layers use a vector of features they lose information regarding pixels above or below and their relations to the current pixel. This model is bypassing the issue with using all convolution layers with down- and upsampling.

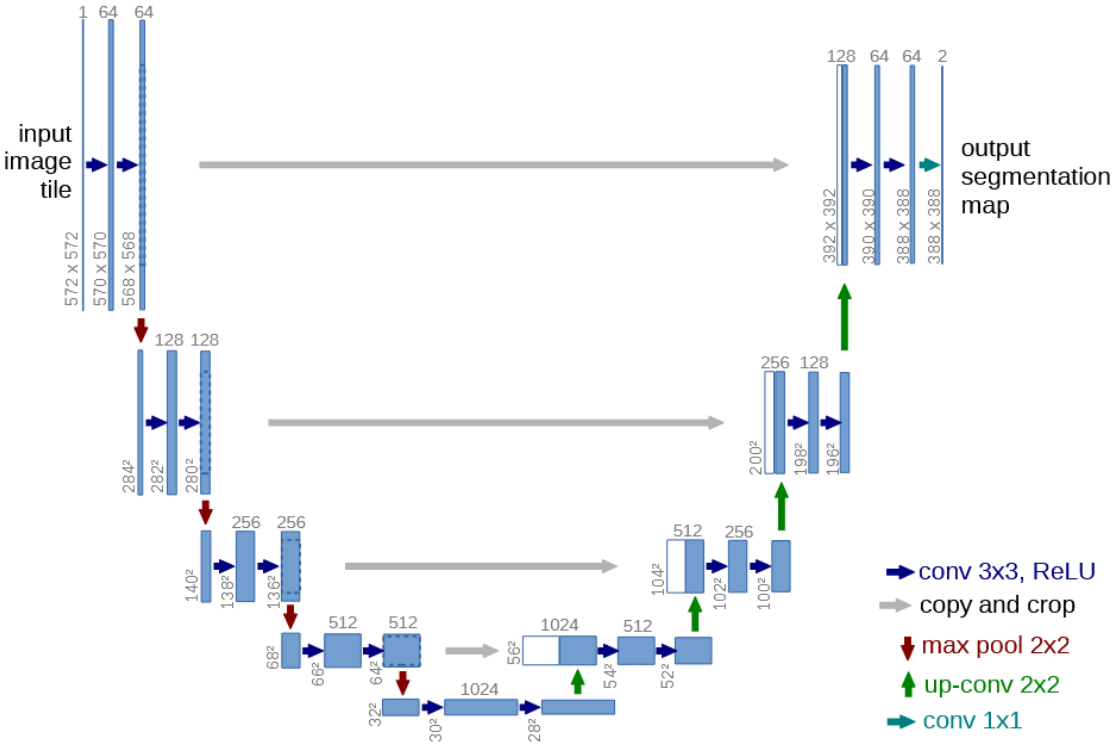

Fig. 22. Structure of U-Net model [54]

In the original architecture the encoder (downsampling) part contains 4 blocks of 2 convolutional layers with 3x3 kernels and 1 max pooling layer with size 2x2, we start with 64 features and increase them to 128, 256 and 512 as we go through the blocks. At the bottom of the U-Net we arrive at the bottleneck that serves as the turning point and transition to the decoder part, this only contains 2 convolution layers of 3x3 size without max pooling layer and feature number 1024. From here we enter the decoder or expansion part of the model with the same number of blocks and within those blocks the same number of convolution layers, to keep the symmetry, but we start with upsampling the size of the feature map to match the respective block in the encoder part and concatenate its output to our blocks input. We start with 512 features, going slowly down to 256, 128 and 64 as the encoder part. The output layer in this case has features equal to the number of classes, which is 1 for binary classification and sigmoid activation function; all previous layers had ReLU activation. [54, 55]

During our experiments with the U-Net architecture we do binary segmentation on a low number of train images. We adjust the input image sizes to 256*256 and start the number of features at 16. We use dropout and batch normalisation through the whole model. For training we use batches of 10 images and a learning rate of 1e-3 through 25 epochs. As in previous experiments we use binary cross entropy loss and pixel accuracy metric. [56, 57, 58, 59]
In terms of accuracy we can see definite improvements on the validation set with a maximum of 0.934 and test accuracy of 0.965.

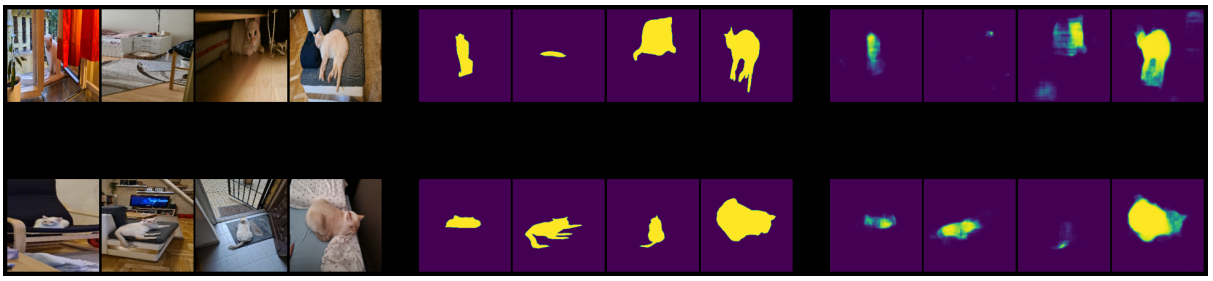

Fig. 23. Results of U-Net model built from scratch

The test image masks in all previous cases used a 0.5 threshold for classifying results into pixels with or without cats. To showcase the technique of changing that threshold, below is an example of using 0.9 instead of 0.5 threshold value, which shows where the algorithm is very confident in its predictions.

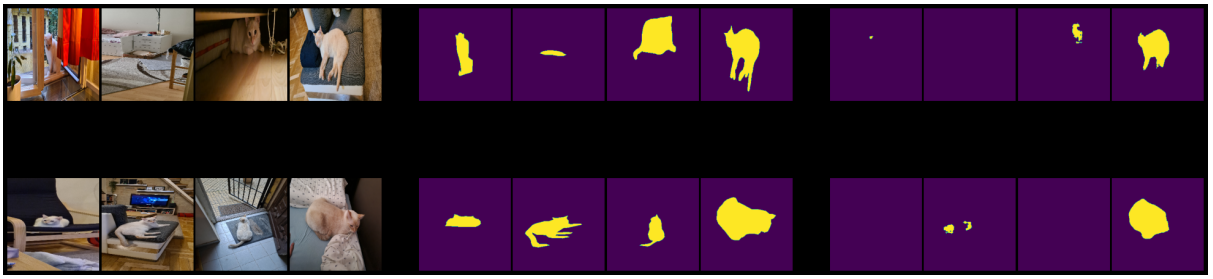

Fig. 24. Results of U-Net model built from scratch with mask threshold 0.9 (previously 0.5)

We can clearly see improvements in visual performance as well, having more precise areas selected by the model even with this low training setup. To improve performance even more we could do image augmentations to increase the training data size, hyperparameter tuning, apply longer training periods and deepen the structure of the model by adding layers.

4.5. Transfer Learning

Transfer learning is a field of research in machine learning and its focus is storing knowledge acquired while solving other problems and then applying it to a different but similar problem. For example if we have a successful algorithm recognizing cars the knowledge gained there can be used to recognize trucks as well. [60]

This approach is very popular in deep learning where pre-trained models are used as starting points in for example image segmentation task (but other fields of deep learning research

apply) given the great amounts of data, time and computing power required to develop these models and the huge head start these models provide in solving other, related problems. [61]

It is very common to use pre-trained weights in image segmentation problems, these trained models can be directly downloaded and added into our own models directly. There are several of these pre-trained models, we are going to use the Oxford VGG model and the Microsoft ResNet model. Both of them were trained on the ImageNet dataset for the ILSVRC competitions in 2014 and 2015 respectively. [62, 63]

For our VGG16 and ResNet50 models with pre-trained weights we use every convolutional layer, but build our own dense (fully connected) layers (4096 and 1000 neurons in VGG and 500 neurons in ResNet) with the last one sized 256*256 and softmax activation function, then reshape it as in the original VGG and ResNet models built from scratch. We use the same batch size, epoch number, learning rate, optimizer, loss and metric as in the VGG and ResNet models above. [64, 65, 66, 67, 68, 69, 70]

We freeze the weights of the pre-trained models on all layers and only train the layer we added ourselves, which is the proper way of using transfer learning. Otherwise, the whole model, even the layers imported from VGG or ResNet will be retrained.

Our accuracy doesn't improve much in the numbers we see, with maximum validation accuracy of 0.814 and test accuracy of 0.915, but as seen on the below images the localization of cats on the test images is far better.

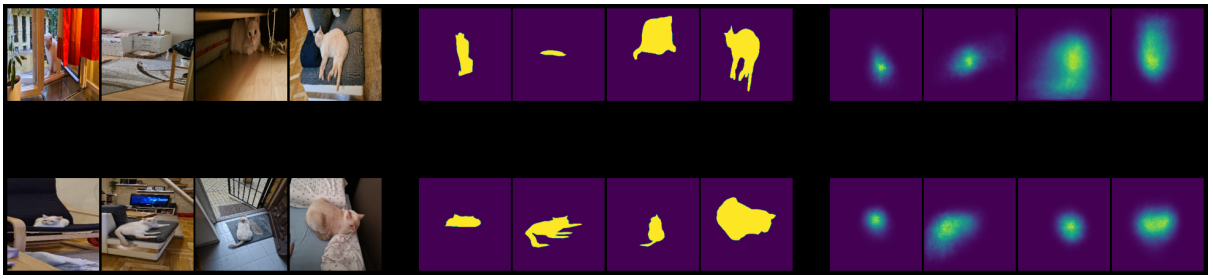

Fig. 25. Results of VGG16 with pre-trained weights

The ResNet50 model with pre-trained weights we achieve very similar accuracies of 0.816 and 0.915 for validation and test respectively. But the images also show great improvements in localising the objects (cats).

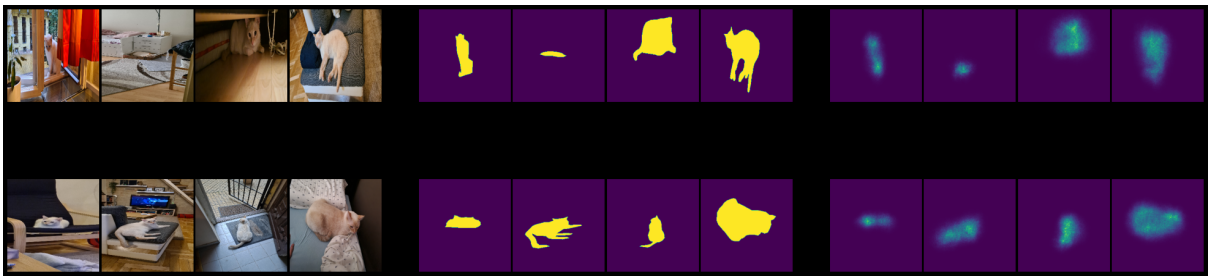

Fig. 26. Results of ResNet50 with pre-trained weights

These views show us clearly the power of transfer learning even in such constrained circumstances and advocates for using these abilities in general purposes.

4.6. VGG16 - U-Net

Our final experiment with the different image segmentation deep learning algorithms is combining the VGG16 model with U-Net, where VGG16 is used as the encoder part and the rest of the U-Net as the decoder part of the algorithm. Since we already talked about both structures in detail and also discussed the changes we applied for customization, this section will only show the current setup and the results it produced.

We use in the decoder part the architecture of our VGG16 models, with 2 blocks of 2 convolutional layers and a max pooling layer and 2 blocks of 3 convolutional layers and a max pooling layer, the kernel size stays 3x3 through the whole decoder part, the feature number starts from 64 and goes up to 512. At the bottom we have 3 convolutional layers with the same kernel size and 512 features without max pooling. In the decoder we have 4 blocks of 3 convolutional layers with upsampling and 3x3 kernels, with the feature number going up from 512 to 64 block-by-block. We use dropout and batch normalisation though the whole model and our output is created by a convolutional layer of size 1 and sigmoid activation. [71, 72, 73, 74]

Accuracy of this model tops at 0.942 on validation set and 0.929 on test data, which is a drop from all previous models.

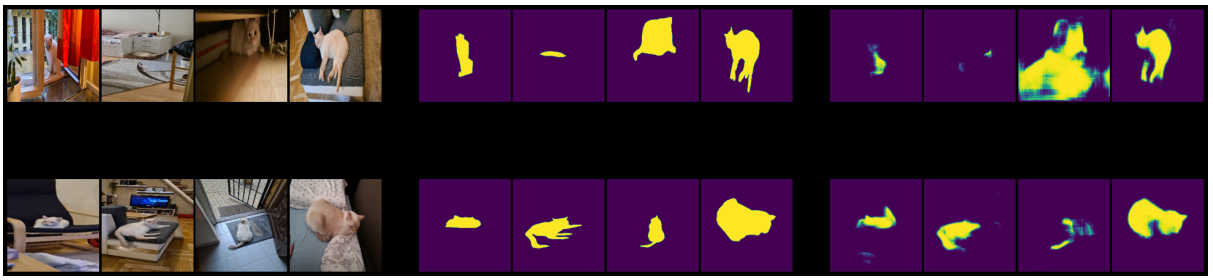

Fig. 27. Results of VGG16 - U-Net model

As we can see from the visual results presented it has very different localisation compared to other models tried out above. The intensity of pixels tends to be more visible around the patches where the cats are located, but we can definitely see very blurry edges and pixels scattered across the whole image. The model is having a hard time identifying one of the cats in the first row.

As all previous models, this one also can be optimised with hyperparameter tuning, longer training periods, as well as used with transfer learning, which we saw produced very nice results.

4.7. Summary of results

The goal of this thesis from the beginning was to explore and understand more deeply the available state-of-the-art deep learning image segmentation algorithms. For that we went through several milestone models that shaped the field of research behind it. We discussed the structures of AlexNet and related models, VGG and ResNet models, which all share the fact that they are combined with CNNs and fully connected layers. From there we went on to experiment with these algorithms by building them up from scratch to see on hand how they behave and what are their specific characteristics. Here we showed, in line with our expectations, that models with fully connected layers need a lot more training time and data size (ResNet and VGG models) to perform better, while U-Net that was specifically designed for small amounts of data and is fully convolutional is working very well with limited amounts of data and can achieve great accuracy.

In the process of working through the knowledge and best-practices, we also learned ways of loading data effectively into models to have enough memory space for training as well and used applications for labelling our own images that have been used to visualise test results.

To sum up, the work that has been done during the preparation and writing of this thesis is in line with the goals set out in the beginning. With that, it was an introduction (and a modern idea) into the vast field of deep learning image processing (segmentation) and it set out a solid basis for future development and exploration of the field.

5. REFERENCES

1. Wikipedia. (2022). *Computer vision*. [online] Available at: https://en.wikipedia.org/wiki/Computer_vision#cite_note-bmva-9.
2. Ilija Mihajlovic (2019a). *Everything You Ever Wanted To Know About Computer Vision. Here's A Look Why It's So Awesome.* [online] Medium. Available at: https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e.
3. Papert, S.A. (1966). The Summer Vision Project. *dspace.mit.edu*. [online] Available at: https://dspace.mit.edu/handle/1721.1/6125.
4. Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*Springer Science & Business Media.
5. Sebe, N., Cohen, I., Garg, A. and Huang, T.S. (2005). *Machine Learning in Computer Vision*. Springer Science & Business Media.
6. Freeman, W., Perona, P. and Schölkopf, B. (2008). Guest Editorial. *International Journal of Computer Vision*, 77(1-3), pp.1–1. doi:10.1007/s11263-008-0127-7.
7. AI & Machine Learning Blog. (2021a). *A 2021 guide to Semantic Segmentation*. [online] Available at: https://nanonets.com/blog/semantic-image-segmentation-2020/.
8. Wikipedia Contributors (2019b). *Digital image processing*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Digital_image_processing.
9. Gonzalez, R.C. and Woods, Richard E (2018). *Digital image processing*. [online] *Open WorldCat*. Available at: https://www.worldcat.org/title/digital-image-processing/oclc/966609831 [Accessed 30 May 2022].
10. Tyagi, M. (2021a). *Image Segmentation : Part 1*. [online] Medium. Available at: https://towardsdatascience.com/image-segmentation-part-1-9f3db1ac1c50.
11. Tyagi, M. (2021b). *Image Segmentation: Part 2*. [online] Medium. Available at: https://towardsdatascience.com/image-segmentation-part-2-8959b609d268.
12. Stanford.edu. (2022). [online] Available at: https://ai.stanford.edu/~syyeung/cvweb/tutorial3.html [Accessed 30 May 2022].
13. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C. and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, [online] 115(3), pp.211–252. doi:10.1007/s11263-015-0816-y.

14. EAI Fund Offical. (2018). *History of image segmentation*. [online] Medium. Available at: https://medium.com/@eaifundoffical/history-of-image-segmentation-655eb793559a [Accessed 30 May 2022].
15. Raza, H. (2020). *CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more*. [online] coderz.py. Available at: https://coderzpy.com/cnn-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more/ [Accessed 30 May 2022].
16. OpenGenus IQ: Learn Computer Science. (2019). *Evolution of CNN Architectures: LeNet, AlexNet, ZFNet, GoogleNet, VGG and ResNet*. [online] Available at: https://iq.opengenus.org/evolution-of-cnn-architectures/.
17. Siddharth Das (2017). *CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more*. [online] Medium. Available at: https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5.
18. Tsang, S.-H. (2020). *Review: ZFNet — Winner of ILSVRC 2013 (Image Classification)*. [online] Medium. Available at: https://medium.com/coinmonks/paper-review-of-zfnet-the-winner-of-ilsvlc-2013-image-classification-d1a5a0c45103.
19. Sik-Ho Tsang (2018). *Review: VGGNet — 1st Runner-Up (Image Classification), Winner (Localization) in ILSVRC 2014*. [online] Medium. Available at: https://medium.com/coinmonks/paper-review-of-vggnet-1st-runner-up-of-ilsvlc-2014-image-classification-d02355543a11.
20. He, K., Zhang, X., Ren, S. and Sun, J. (2015). *Deep Residual Learning for Image Recognition*. [online] Available at: https://arxiv.org/pdf/1512.03385.pdf.
21. Simonyan, K. and Zisserman, A. (2015). *Published as a conference paper at ICLR 2015 VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION*. [online] Available at: https://arxiv.org/pdf/1409.1556.pdf.
22. Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), pp.84–90. doi:10.1145/3065386.
23. Alom, Z., Taha, T., Yakopcic, C., Westberg, S., Nasrin, S. and Asari, V. (n.d.). *The History Began from AlexNet: A Comprehensive Survey on Deep Learning*

Approaches. [online] Available at: https://arxiv.org/ftp/arxiv/papers/1803/1803.01164.pdf.

24. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. (2014). *Going Deeper with Convolutions*. [online] arXiv.org. Available at: https://arxiv.org/abs/1409.4842.
25. Zeiler, M.D. and Fergus, R. (2013). *Visualizing and Understanding Convolutional Networks*. [online] arXiv.org. Available at: https://arxiv.org/abs/1311.2901.
26. Wikipedia. (2020a). *Convolutional neural network*. [online] Available at: https://en.wikipedia.org/wiki/Convolutional_neural_network#Architecture [Accessed 30 Jun. 2020].
27. Wikipedia. (2019a). *Convolution*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Convolution.
28. LaptrinhX (2020). *Getting Started With CNN*. [online] LaptrinhX. Available at: https://laptrinhx.com/getting-started-with-cnn-153816276/ [Accessed 30 May 2022].
29. ResearchGate. (2020). *Pooling layer operation approaches*. [online] Available at: https://www.researchgate.net/figure/Pooling-layer-operation-oproaches-1-Pooling-layers-For-the-function-of-decreasing-the_fig4_340812216.
30. ResearchGate. (2019). *An example of a CNN architecture with two convolution layers and one fully connected*. [online] Available at: https://www.researchgate.net/figure/An-example-of-a-CNN-architecture-with-two-convolution-layers-and-one-fully-connected_fig4_334486302.
31. Wikipedia (2019c). *Softmax function*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Softmax_function.
32. Wikipedia (2018). *Activation function*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Activation_function.
33. AI & Machine Learning Blog. (2021b). *A 2021 guide to Semantic Segmentation*. [online] Available at: https://nanonets.com/blog/semantic-image-segmentation-2020/.
34. PyImageSearch. (2016). *Intersection over Union (IoU) for object detection*. [online] Available at: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.
35. Wikipedia. (2021). *F-score*. [online] Available at: https://en.wikipedia.org/wiki/F-score.

36. cocodataset.org. (n.d.). *COCO - Common Objects in Context*. [online] Available at: https://cocodataset.org/#home.
37. GitHub. (2022b). *cocodataset/cocoapi*. [online] Available at: https://github.com/cocodataset/cocoapi [Accessed 30 May 2022].
38. Viraf (2020a). *Master the COCO Dataset for Semantic Image Segmentation*. [online] Medium. Available at: https://towardsdatascience.com/master-the-coco-dataset-for-semantic-image-segmentation-part-1-of-2-732712631047 [Accessed 30 May 2022].
39. Viraf (2020b). *Master the COCO Dataset for Semantic Image Segmentation*. [online] Medium. Available at: https://towardsdatascience.com/master-the-coco-dataset-for-semantic-image-segmentation-part-2-of-2-c0d1f593096a [Accessed 30 May 2022].
40. Patrawala, V. (2022). *COCO-Semantic-Segmentation*. [online] GitHub. Available at: https://github.com/virafpatrawala/COCO-Semantic-Segmentation/blob/master/COCOdataset_SemanticSegmentation_Demo.ipynb [Accessed 30 May 2022].
41. labelme.csail.mit.edu. (n.d.). *LabelMe. The Open annotation tool*. [online] Available at: http://labelme.csail.mit.edu/Release3.0/ [Accessed 30 May 2022].
42. wkentaro (2019). *wkentaro/labelme*. [online] GitHub. Available at: https://github.com/wkentaro/labelme.
43. Jamil, N., N Redzuan, N., Ismail, M. and Ramli, W. (2019). Evaluation of VGG Networks for Semantic Image Segmentation of Malaysian Meals. *Proceedings of the Proceedings of the 1st International Conference on Informatics, Engineering, Science and Technology, INCITEST 2019, 18 July 2019, Bandung, Indonesia*. doi:10.4108/eai.18-7-2019.2287943.
44. paperswithcode.com. (n.d.). *Papers with Code - VGG Explained*. [online] Available at: https://paperswithcode.com/method/vgg.
45. Rohit Thakur (2019). *Step by step VGG16 implementation in Keras for beginners*. [online] Medium. Available at: https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c.
46. Mohan, S. (2020b). *Keras Implementation of VGG16 Architecture from Scratch with Dogs Vs Cat Data Set*. [online] MLK - Machine Learning Knowledge. Available at: https://machinelearningknowledge.ai/keras-implementation-of-vgg16-architecture-from-scratch-with-dogs-vs-cat-data-set/.

47. AI & Machine Learning Blog. (2021c). *How to do Semantic Segmentation using Deep learning*. [online] Available at: https://nanonets.com/blog/how-to-do-semantic-segmentation-using-deep-learning/.
48. Ruiz, P. (2018). *Understanding and visualizing ResNets*. [online] Medium. Available at: https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8.
49. Versloot, C. (2022). *Machine learning articles*. [online] GitHub. Available at: https://github.com/christianversloot/machine-learning-articles/blob/main/how-to-build-a-resnet-from-scratch-with-tensorflow-2-and-keras.md.
50. GitHub. (2022a). *Automatic Detection of Solar Panels on High Resolution Imagery.* [online] Available at: https://github.com/dbaofd/solar-panels-detection/blob/master/segnet_resnet_v1.ipynb [Accessed 30 May 2022].
51. Mohan, S. (2020a). *Keras Implementation of ResNet-50 (Residual Networks) Architecture from Scratch*. [online] MLK - Machine Learning Knowledge. Available at: https://machinelearningknowledge.ai/keras-implementation-of-resnet-50-architecture-from-scratch/.
52. Analytics Vidhya. (2021). *How to code your ResNet from scratch in Tensorflow?* [online] Available at: https://www.analyticsvidhya.com/blog/2021/08/how-to-code-your-resnet-from-scratch-in-tensorflow/.
53. Wikipedia. (2020b). *U-Net*. [online] Available at: https://en.wikipedia.org/wiki/U-Net.
54. Ronneberger, O., Fischer, P. and Brox, T. (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation*. [online] arXiv.org. Available at: https://arxiv.org/abs/1505.04597.
55. Harshall Lamba (2019). *Understanding Semantic Segmentation with UNET*. [online] Medium. Available at: https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47.
56. Paul, J. (2022). *Skeyenet - Read Our Planet Like a Book*. [online] GitHub. Available at: https://github.com/Paulymorphous/skeyenet/blob/master/Src/Road_Detection_GPU.ipynb [Accessed 30 May 2022].

57. Bhattiprolu, D.S. (2022a). *Introductory python tutorials for image processing*. [online] GitHub. Available at: https://github.com/bnsreenu/python_for_image_processing_APEER.
58. Bhattiprolu, D.S. (2022b). *Introductory python tutorials for image processing*. [online] GitHub. Available at: https://github.com/bnsreenu/python_for_image_processing_APEER/blob/master/tutorial118_binary_semantic_segmentation_using_unet.ipynb [Accessed 30 May 2022].
59. Bhattiprolu, D.S. (2022c). *Introductory python tutorials for image processing*. [online] GitHub. Available at: https://github.com/bnsreenu/python_for_image_processing_APEER/blob/master/tutorial117_building_unet_using_encoder_decoder_blocks.ipynb [Accessed 30 May 2022].
60. Wikipedia. (2019d). *Transfer learning*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Transfer_learning.
61. Jason Brownlee (2019b). *A Gentle Introduction to Transfer Learning for Deep Learning*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/transfer-learning-for-deep-learning/.
62. Visual Geometry Group. (2014). *Visual Geometry Group - University of Oxford*. [online] Available at: https://www.robots.ox.ac.uk/~vgg/research/very_deep/.
63. KaimingHe (2016). *KaimingHe/deep-residual-networks*. [online] GitHub. Available at: https://github.com/KaimingHe/deep-residual-networks.
64. Cassimiro, G. (2021). *Transfer Learning with VGG16 and Keras*. [online] Medium. Available at: https://towardsdatascience.com/transfer-learning-with-vgg16-and-keras-50ea161580b4.
65. Keras. (n.d.). *Keras documentation: VGG16 and VGG19*. [online] keras.io. Available at: https://keras.io/api/applications/vgg/.
66. Brownlee, J. (2019a). *Transfer Learning in Keras with Computer Vision Models*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/.
67. www.learndatasci.com. (n.d.). *Hands-on Transfer Learning with Keras and the VGG16 Model*. [online] Available at: https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras/.

68. Chronicles of AI. (2021). *Transfer Learning With Keras(Resnet-50)*. [online] Available at: https://chroniclesofai.com/transfer-learning-with-keras-resnet-50/.
69. Aguas, K.C. (2020). *A guide to transfer learning with Keras using ResNet50*. [online] Medium. Available at: https://medium.com/@kenneth.ca95/a-guide-to-transfer-learning-with-keras-using-resnet50-a81a4a28084b.
70. Keras. (n.d.). *Keras documentation: ResNet and ResNetV2*. [online] keras.io. Available at: https://keras.io/api/applications/resnet/.
71. ResearchGate. (2018). *The VGG16-UNet architecture which is U-Net models with VGG16 encoder*. [online] Available at: https://www.researchgate.net/figure/The-VGG16-UNet-architecture-which-is-U-Net-models-with-VGG-16-encoder_fig1_325893881.
72. Gazali, S. (2021). *Cell Nuclei Segmentation using VGG16-UNET And Double-UNET*. [online] Medium. Available at: https://saifgazali.medium.com/cell-nuclei-segmentation-using-vgg16-unet-and-double-unet-eafb65bb959a [Accessed 30 May 2022].
73. Patel, A. (2022). *Satellight Image Segmentation with UNet and Its Variants*. [online] GitHub. Available at: https://github.com/ashishpatel26/satellite-Image-Semantic-Segmentation-Unet-Tensorflow-keras [Accessed 30 May 2022].
74. Nawaz, A., Akram, U., Salam, A.A., Ali, A.R., Ur Rehman, A. and Zeb, J. (2021). *VGG-UNET for Brain Tumor Segmentation and Ensemble Model for Survival Prediction*. [online] IEEE Xplore. doi:10.1109/ICRAI54018.2021.9651367.
75. Ilija Mihajlovic (2019b). *Everything You Ever Wanted To Know About Computer Vision. Here's A Look Why It's So Awesome.* [online] Medium. Available at: https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e.