remote sensing

MDPI

Review

Object Detection and Image Segmentation with Deep Learning on Earth Observation Data: A Review-Part I: Evolution and Recent Trends

Thorsten Hoeser [1,*] and Claudia Kuenzer [1,2]

[1] German Remote Sensing Data Center (DFD), German Aerospace Center (DLR), Münchner Straße 20, D-82234 Wessling, Germany; Claudia.Kuenzer@dlr.de

[2] Department of Remote Sensing, Institute of Geography and Geology, University Würzburg, Am Huband, D-97074 Wuerzburg, Germany

* Correspondence: Thorsten.Hoeser@dlr.de; Tel.: +49-8153-28-3966

Received: 27 April 2020; Accepted: 19 May 2020; Published: 22 May 2020

Abstract: Deep learning (DL) has great influence on large parts of science and increasingly established itself as an adaptive method for new challenges in the field of Earth observation (EO). Nevertheless, the entry barriers for EO researchers are high due to the dense and rapidly developing field mainly driven by advances in computer vision (CV). To lower the barriers for researchers in EO, this review gives an overview of the evolution of DL with a focus on image segmentation and object detection in convolutional neural networks (CNN). The survey starts in 2012, when a CNN set new standards in image recognition, and lasts until late 2019. Thereby, we highlight the connections between the most important CNN architectures and cornerstones coming from CV in order to alleviate the evaluation of modern DL models. Furthermore, we briefly outline the evolution of the most popular DL frameworks and provide a summary of datasets in EO. By discussing well performing DL architectures on these datasets as well as reflecting on advances made in CV and their impact on future research in EO, we narrow the gap between the reviewed, theoretical concepts from CV and practical application in EO.

Keywords: artificial intelligence; AI; machine learning; deep learning; neural networks; convolutional neural networks; CNN; image segmentation; object detection; Earth observation

1. Introduction

In recent years, deep learning (DL) has received a lot of attention, in both scientific research and practical application [1,2]. Two main factors are responsible for this growing attention: the accessibility of data and the increase in computational processing power, especially with graphics processing units [3–5]. Due to these developments, researchers were able to demonstrate working concepts for DL which could even outperform established approaches. Their fast improving insights were quickly applied in other disciplines and in practice. Therewith, a self-reinforcing research environment was created which today has significant impact on science and practice.

Increasing data accessibility can also be found in the field of Earth observation. The availability of high-resolution optical and multispectral imagery is particularly important. Due to the recent trend of opening archives of Earth observation data, it can be expected that this amount of high resolution remote sensing data will increase dramatically in near future. However, high resolution optical data have already paved the way for transferring DL concepts from computer vision to Earth observation application such as detecting or segmenting vehicles, roads and buildings from overhead images. With these proof-of-concepts of DL for Earth observation research, today, the applications are wide and

no longer limited to RGB images. The numbers of DL implementations in Earth observation are still growing and showing new trends and possibilities of analysing remotely-sensed data [6–9].

The importance of DL today reaches far across the scientific community. In 2019, Nature led the Google scholar metric h5-index, where its top three papers are all covered DL and can therefore be seen as part of the most relevant papers in 2019 overall. In the same list of h5-index journals of 2019, the IEEE Conference on Computer Vision and Pattern Recognition, well known for its contributions to the research on DL, reached the top ten for the first time [10]. Furthermore, interest from data driven companies such as Google and Facebook has been responsible for driving its recent popularity [1]. Their contributions are both theoretical and practical. For instance, Google is the leading affiliation for papers submitted between 2014 and 2018 during one of the world's most important conferences in the field, Neural Information Processing Systems Conference [11]. At the same time, both Google and Facebook are mainly developing the two most popular DL frameworks, TensorFlow [12] and Pytorch [13], respectively (see Section 4).

One proxy for the recently fast growing interest in research on DL are the publications submitted to arXiv, a distribution service of open access articles frequently used in computer science. Figure 1 shows the annual absolute number of publications concerning deep learning and also its share of all publications submitted to arXiv in the computer science (*cs*) and statistics (*stat*) categories in the same year. Both numbers are growing, which means that there is not just an absolute growth in DL research but also its share of research in computer science and statistics is getting bigger each year.

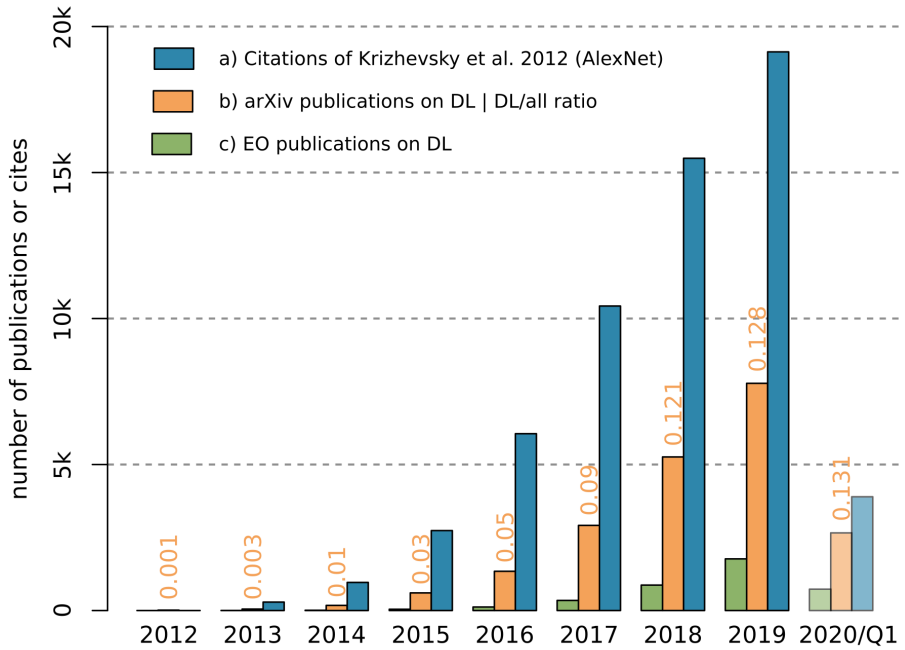

Figure 1. Evolution of deep learning related publications: (a) citations listed in Google scholar for Krizhevsky et al. 2012 [3]; (b) arXiv listed publications in the categories *cs* and *stat* including the terms *deep learning*, *convolutional neural networks*, *convolutional networks or fully convolutional* and their share of all publications listed in the two categories; and (c) publications in selected Earth observation journals, searched for with the same terms as in arXiv.

What makes DL so successful is its capacity to represent more abstract concepts [1,2,14] such as speech or images. DL models outperformed classical machine learning models and signal processing approaches [14], for instance in speech recognition [15–17] and image recognition for handwritten digits [18–20]. Finally, in 2012, Ciresan et al. [20] and Krizhevsky et al. [3] introduced convolutional neural networks (CNNs), the most representational DL models [2], for image recognition of natural images. The model of Krizhevsky et al. [3] called AlexNet, a CNN which extracts features from RGB

images to predict a single label, won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 [3]. ILSVRC is an annual competition, held since 2010 for computer vision tasks such as image recognition and object detection. The breakthrough of AlexNet in performance during the ILSVRC in 2012 can today be seen as the birth of recent developments in DL [21] and is used as starting point in this review.

From traditional computer science, where research is done on DL, the technique evolved into other disciplines which do research with DL. The paper of Krizhevsky et al. [3] is widely cited in both fields, those doing research on as well as with DL. Therefore, the number of citations can be interpreted as a proxy for how much attention DL and especially CNNs received over the last years, as well as outside the field of computer science. Figure 1 shows the increasing number of citations, thus demonstrating the relevance of not only AlexNet but also DL with CNNs for recent research.

Since imagery data are fundamental for Earth observation research, the application of DL seems likely. Nevertheless, looking at the combined publications of leading remote sensing journals and conference papers in Figure 1, one can see that DL reached the Earth observation community with an offset of three to four years, taking 2012 as the starting point. However, since then, the number of publications concerning DL and remote sensing has more than doubled each year. This trend continued in 2020, when in the first quarter the number of publications was nearly half the number for the entire year of 2019. Current reviews discuss the wide range of applications of DL on remotely-sensed data for super resolution, data fusion, denoising, weather forecasting, scene recognition, classification and object detection with optical, multispectral, hyperspectral and SAR sensors in different resolutions [6–9,22–24].

However, due to the vast amount of insights in DL over the last eight years, this review aims to provide a more detailed overview about the progress primarily made in computer vision from 2012 to late 2019 by focussing on object detection and image segmentation with CNNs. Therefore, the review takes an evolutionary perspective to outline the major milestones and their interrelations. By running a thread through important DL publications, we specifically address Earth observation researchers who want to add DL to their toolbox, or want to reflect on the evolution of DL approaches to choose a matching model design for their own research questions. Providing this thorough introduction, we contribute to the open question number eight, "How to best handle high entry barriers to DL?" (p. 32) as stated by Ball et al. (2017) [8]. Overall, this review contributes to a better understanding of the principles of CNNs. This foundation will further be used in *Part II: Applications* to discuss applications of CNNs in Earth observation research by reviewing leading Earth observation journals.

2. Terminology and Basic Concepts of Deep Learning with CNNs

In supervised machine learning (ML), algorithms try to learn features from labelled training data to predict an accurate output from an unknown input. DL models are specific ML models, made of stacked layers, which enable those models to consecutively extract richer features from the input data. As more layers are stacked, the model becomes *deeper*, and more complex features can be *learned*, hence the name *deep learning*. ML itself is part of the wide field of artificial intelligence (AI), thus DL is not the same as AI but part of it. Before different DL models are introduced, reference is made to Table 1. It explains the intensive and commonly used abbreviations in DL literature, which are also used in the following sections in order to accustom an audience interested in DL to its terminology. Furthermore, the table is ordered thematically and chronologically to provide a thread through DL literature concerning image recognition, image segmentation and object detection with CNNs.

The classical DL model, the artificial neural network (ANN) pictured in Figure 2c, can briefly be described as a sequence of fully connected layers, from input over hidden to output layers of artificial neurons. All neurons within one layer are connected to each neuron of the previous layer via linear operations also called weights or parameters, hence the term fully connected. The connections transport the values from all neurons of the previous layer by multiplying them with the weights assigned to each connection and passing them to the neurons of the next layer. Here, each neuron sums

up the incoming values and performs a subsequent non-linear function, the activation. Such a network then transports values from the input layer via the weights over all hidden layers to the output layer and can be seen as a non-linear function of higher order, mapping input values to outputs [5,25].

Table 1. List of abbreviation, explanations and the original literature or further discussion. The sorting of the table reflects the structure of Sections 2 and 3. At the same time, it is a guiding thread through the milestone-publications on deep learning architectures of convolutional neural networks for image recognition, segmentation and object detection from 2012 to late 2019.

Abbreviation	Reference	Explanation
DL	[2,25]	**Deep Learning**
ANN	[25]	Artificial Neural Network
SGD	[26–29]	Stochastic Gradient Descent
BP	[25,26,30]	Backpropagation
ReLU	[31–33]	Rectified Linear Unit
CNN	[2]	Convolutional Neural Network
RNN	[34]	Recurrent Neural Network
LSTM	[35]	Long Short Term Memory
GAN	[36]	Generative Adversarial Network
IR		**Image Recognition**
ImageNet	[37]	DL dataset
ILSVRC	[3]	ImageNet Large Scale Vision Recognition Challenge
AlexNet	[3]	CNN by Alex Krizhevsky et al. [3]
ZFNet	[38]	CNN by Zeiler and Fergus [38]
VGG-16/19	[39]	CNN by members of the Visual Geometry Group
LRN	[3,38]	Local Response Normalisation
Inception V1-3	[40–42]	CNN architectures with Inception modules
BN	[41]	Batch Normalisation
ResNet	[43]	CNN architecture with residual connections
ResNeXt	[44]	Advanced CNN architecture with residual connections
Xception	[45]	ResNet-Inception combined CNN architecture
DenseNet	[46]	Very deep, ResNet based CNN
SENet	[47]	Squeeze and Excitation Network
NAS	[48]	Neural Architecture Search
NASNet	[49]	CNN architecture drafted with NAS
MobileNet	[50]	Efficient CNN architecture
MnasNet	[51]	Efficient CNN architecture drafted with NAS
EfficientNet	[52]	Efficient CNN architecture drafted with NAS
IS		**Image Segmentation**
PASCAL-VOC	[53,54]	Pattern Analysis, Statistical modeling and Computational Learning—Visual Object Classes dataset
FCN	[55]	Fully Convolutional Network
DeepLabV1-V3+	[56–60]	CNN architectures for IS
CRF	[61]	Conditional Random Field
ASPP	[58,59]	Atrous Spatial Pyramid Pooling
DPC	[62]	Dense Prediction Cell, CNN module drafted with NAS
AutoDeepLab	[63]	CNN architecture drafted with NAS
DeconvNet	[64]	Deconvolutional Network CNN
ParseNet	[65]	Parsing image context CNN
PSPNet	[66]	Pyramid Scene Parsing Network
U-Net	[67]	U-shaped encoder–decoder CNN
Tiramisu	[68]	U-Net-DenseNet combined CNN
RefineNet	[69]	Encoder–decoder CNN
HRNetV1-2	[70,71]	High Resolution Networks CNNs for IS

Table 1. *Cont.*

Abbreviation	Reference	Explanation
OD		**Object Detection**
MS-COCO	[72]	Microsoft-Common Object in Context dataset
R-CNN	[73]	Region based CNN
SPPNet	[74]	Spatial Pyramid Pooling Network
RoI pooling	[75]	Discretised pooling of Regions of Interest
Fast R-CNN	[75]	R-CNN + RoI pooling based CNN
RPN	[76]	Region (RoI) Proposal Network
Faster R-CNN	[76]	R-CNN + RPN + RoI pooling based CNN
FPN	[77]	Feature Pyramid Network
IoU	[53,78]	Intersection over Union (metric)
Cascade R-CNN	[79]	Faster R-CNN based cascading detector for less noisy detections
RoI align	[80]	Floating point variant of RoI pooling for higher accuracy
Mask R-CNN	[80]	Faster R-CNN + FCN based instance segmentation
CBNet	[81]	Composite Backbone Network for R-CNN based networks
PANet	[82]	Path Aggregation Network
SNIP(ER)	[83,84]	Scale Normalisation for Image Pyramids (with Efficient Resampling)
TridentNet	[85]	CNN using atrous convolution
fps		frames per second (metric)
YOLO-V1-3	[86–88]	You Only Look Once
NMS		Non-Maximum Suppression
DarkNet	[87,88]	CNN backbone for YOLO-V2+3
SSD	[89]	Single Shot MultiBox Detector
RetinaNet	[90]	CNN using an adaptive loss function for OD
RefineDet	[91]	CNN performing anchor refinement before detection
NAS-FPN	[92]	FPN variant drafted with NAS
EfficientDet	[93]	Efficient CNN for OD based on EfficientNet
BiFPN	[93]	Bi-directional FPN

In supervised learning, to calibrate or train the parameters within an ANN, a set of inputs with known outputs, the labels, is passed forward through the ANN. The predicted outcome is compared with the label and a measure is calculated representing the performance of the ANN, the loss. To increase the models performance, the loss function is minimised. This is done by partially deriving its gradients for each contributing parameter. The opposite of those gradients is then used to change the parameters iteratively until a minimum of the loss function is reached. In DL, this is known as stochastic gradient descent with back propagation [25–30,94]. In this way, an ANN gradually learns from labelled training data which combination of parameters best represents the variance of the training dataset. The trained model can then be used with a known accuracy to infer on unknown data coming from a distribution represented by the training data. Thereby, the order of inputs in an ANN is not necessarily important, since the model does not especially exploit the underlying structure of the input data.

The main characteristics of a DL model can be summarised as stacked layers of non-linear functions, trained to extract features from input data to predict an output based on those automatically found instead of hand crafted features. The model type and its architecture are influencing the handling of the underlying structure of the data and the feature richness found by it. As a model gets deeper, more complex, abstract and distinctive features are able to be found. However, this behaviour is not infinite and overfitting tends to occur when simply adding parameters to the model by making it deeper.

An ANN as described above belongs to the DL models called *stacked autoencoders*. Other model types are *Recurrent Neural Networks (RNNs)* [34], or more specifically *Long Short-Term Memory (LSTM)* [35], which are used for processing sequential data; *Generative Adversarial Networks (GANs)* [36], the primary idea of which was to generate data; and finally *Convolutional Neural Networks (CNNs)*.

CNNs are popular for processing 2D array input data such as images [2], which we focus on in this review.

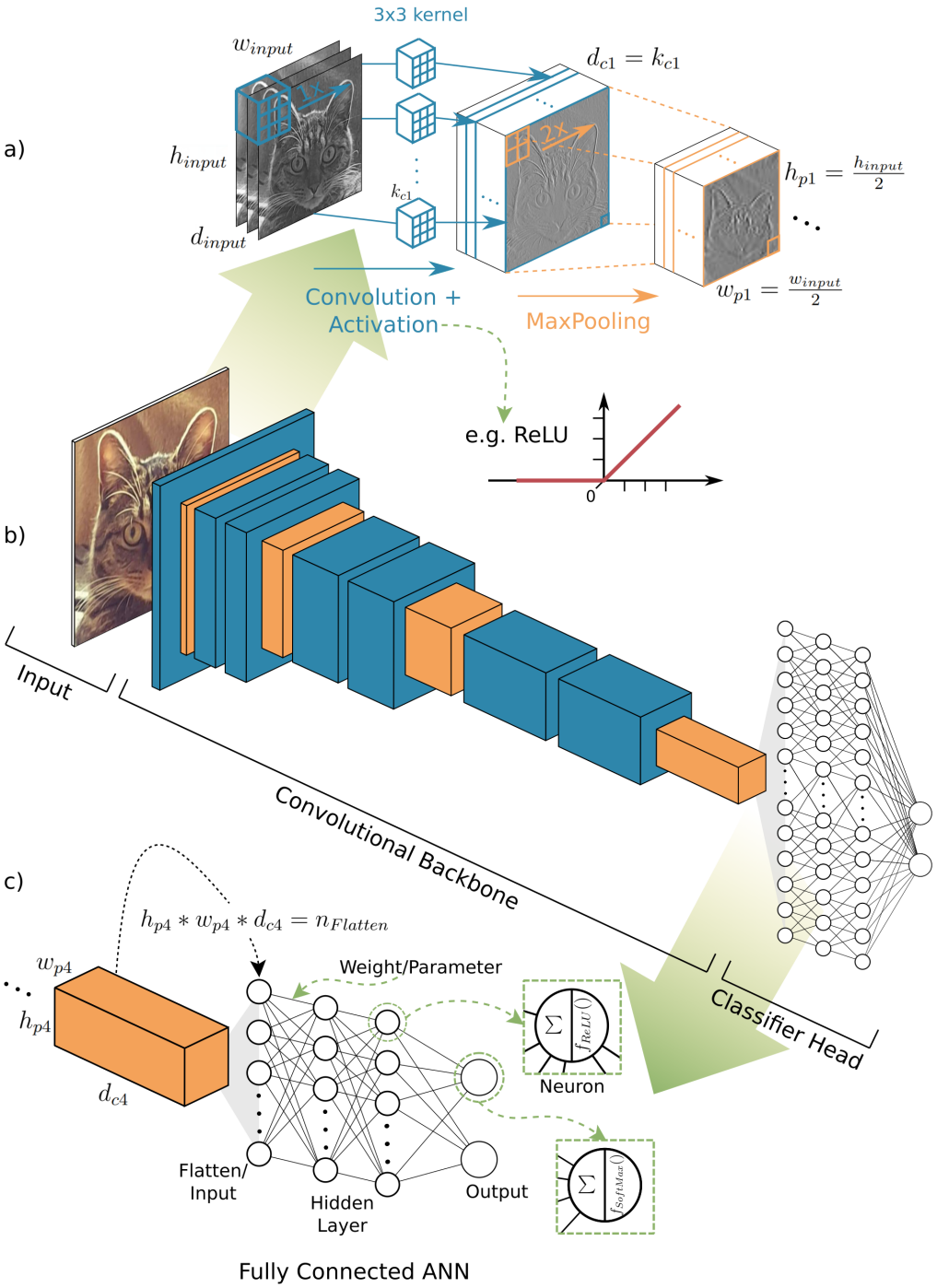

Figure 2. Overview and details of a convolutional neural network (CNN) architecture for image recognition. (**a**) Zoom in on a three-channel RGB input, convolution + activation function, e.g., Rectified Linear Unit (ReLU) (blue) and adjacent max pooling operations (orange). Those operations are used repeatedly in (**b**) the convolutional backbone, part of the overall structure of the architecture. From an input image, feature maps are created by convolution and resized by max pooling operations getting smaller in resolution but deeper in feature maps until they reach a classifier, the head of the architecture, here a fully connected artificial neural network (ANN). (**c**) Details on the transition between convolutional backbone and classifier head, as well as the structure of a multi-layer ANN performing classification.

Images are records of natural signals and therefore provide information as pixel values and their local connectivity. From this property, low level features such as edges can be combined to features of higher levels with a semantic meaning. Therefore, analysing images means exploring pixel values and their local connectivity to find strong, distinctive representations which can be used for image classification. Using a DL model such as an ANN would not fully take the local connectivity of a natural signal into account. In contrast, a CNN allows for the learning of those representational features of imagery data to an increasingly abstract degree, while also being aware of their values and local arrangement. The convolutional operations are thereby trainable kernel functions connecting the layers in a CNN as the linear connections are doing it in an ANN. Due to the local sensitivity of a kernel function, CNNs are able to take local connectivity of the input data into account when learning the features from them [2].

To further introduce the basic functionality of a CNN, a network architecture for image recognition is to be assumed, as shown in Figure 2b. The overall architecture can roughly be split into three modules: input, convolutional backbone and classifier head. The 2D array input is passed through a sequence of convolutions, activations and max pooling operations called the convolutional backbone in order to extract high level features. The adjacent classifier, located at the end of the backbone and therefore the head, is here, similar to an ANN, a sequence of stacked, fully connected layers. It uses the extracted features from the convolutional backbone to classify them into output classes and provide their probability.

Figure 2a shows how k_{c1} kernels of size 3×3 convolve the 2D input array over its entire depth, producing a stack of d_{c1} feature maps corresponding to the kernel functions. Since the kernel functions are linear operations, each feature map is activated with a non-linear function, e.g., ReLU $f(x) = \max(0, x)$ [31–33]. On each activated feature map, a pooling operation is applied such as max pooling with a 2×2 kernel, selecting the maximum value from a 2×2 neighbourhood. Pooling introduces translation invariance to the model [2,95] and reduces resolution by factor 2 when applied with a stride of 2. This is useful, when going deeper into the net where progressively semantically low level features with high local relations are combined to many semantically high level features with low spatial relevance [2,96]. To keep the computations reasonable as well as allowing deeper stacks of feature maps with larger feature variance, the resolution is decreased by max pooling while the number of feature maps is increased by convolutional layers using gradually more kernels (depicted as subsequently smaller but deeper blocks in Figure 2b). All convolutional operations, the kernel functions, are the weights in a CNN, which are adjusted during training. Therewith, the extracted features are not hand crafted but learned from training data [3].

To enter the classifier, the last stack of max pooled feature maps in the backbone is flattened. This is done by transferring each pixel value of the final feature maps to an input neuron for a shallow fully connected ANN, as depicted in Figure 2c. The final output layer holds as many neurons as there are classes and the activation of this layer is in this case the softmax function, which returns the probability for each class based on the values transported to the last layer. The output probability distribution finally tells how likely each possible class is predicted for the whole image. From this introduction, major characteristics of CNNs and differences to ANNs can be summarised:

- The convolutional backbone is a strong feature extractor for a natural signal, while it maintains the fundamental structure of that signal and is sensitive for local connectivity.
- Instead of pairwise connections of neurons, kernel functions are used to connect layers, in order to learn features from training data.
- By sequentially repeating convolution, activation and pooling, the idea of how natural signals are composed, of low combined to high level features, the artificial architectures of CNNs for extracting features follows the hierarchical structure of a natural signal and mimic the behaviour of the visual cortex of living mammals [96–99].
- The modular composition of both the convolutional backbone itself and the overall architecture makes the CNN approach highly adaptable for a variety of tasks and optimisations.

However, image recognition was chosen as an introductory example because of its relatively simple network architecture. By changing the head structure after the convolutional backbone, the network can be changed to perform a completely different task, such as image segmentation object detection or instance segmentation. In Figure 3, example applications are shown for all these tasks. The tasks main characteristics can be summarised as:

- Image recognition is understood as the prediction of a class label for a whole image.
- Image segmentation, semantic segmentation or pixel wise classification segments the whole image into semantic meaningful classes, where the smallest segment can be a single pixel.
- Object detection predicts locations of objects as bounding boxes and a class label.
- Instance segmentation is an object detection task on which an image segmentation task for the specific bounding box and class is applied additionally. This results in a segmentation mask of the specific object predictions. In this review, instance segmentation is discussed together with object detection, due to their evolutionary closeness.

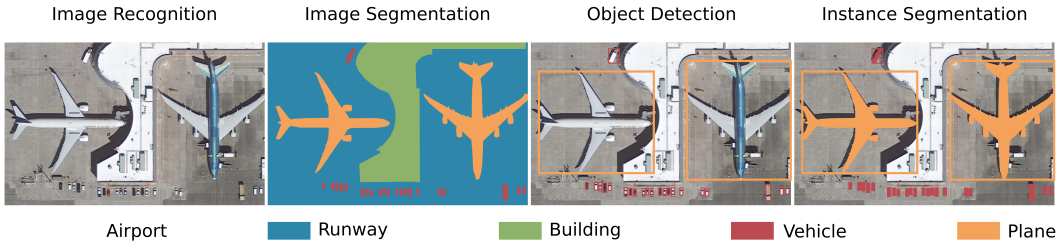

Figure 3. Examples for the tasks of: image recognition, assigns a single label to a whole image; image segmentation, densely classifies each pixel, object detection: locates and classifies specific objects in an image by providing a bounding box; and instance segmentation, provides a segmentation mask for detected objects within a bounding box. The example image is from the DOTA dataset, an object detection dataset of high resolution RGB aerial images [100].

3. Evolution of CNN Architectures in Computer Vision

The CNN family grew bigger when AlexNet was introduced during the ILSVRC in 2012 [3]. To provide an overview of the evolution of CNNs, AlexNet is used as the root of the highly ramified field, which reaches until late 2019 in this review. As mentioned in Section 2, next to the main trunk of architectures used for image recognition, the focus is also put on branches with architectures for image segmentation and object detection tasks. Despite this focus, the number of architectures and variations since AlexNet is still overwhelming. Hence, the evolutionary review focusses on the main successors, defined by their performance on well-established benchmark datasets of their specific tasks, as well as their legacy to the field. Thus, a thread through the evolution of CNNs is provided, which starts in 2012 and follows recent trends until late 2019 (see Table 1).

3.1. Image Recognition and Convolutional Backbones

As introduced in Section 2, the main part of a CNN is the convolutional backbone. Its design is of high relevance for optimising its performance. Since convolutional backbones are widely used in other DL tasks, such as image segmentation and object detection, achievements in image recognition can be seen as a main driver for the field. The review on architectures for image recognition in this section therefore discusses more sophisticated backbones and concepts about CNNs than the one already introduced in Section 2. For a better overview of the evolution, the different architectures are assigned to four groups of CNN families which we call: *Vintage Architectures*, containing AlexNet [3], ZFNet [38] and VGG variants [39]; the *Inception Family*; the *ResNet Family*; and finally architectures of the MobileNet family and such designed using neural architecture search with the goal of being efficient, which we refer to as the *Efficient designs*.

In addition to new architectures, training data is one significant component of the recent successes in DL [3–5]. The ImageNet dataset, with about 14 million labelled images for image recognition [3,37] in its latest version, is used during the ILSVRC. In 2012, Krizhevsky et al. [3] won the image recognition task with large margin using their CNN called AlexNet. Since then, the 2012 version of the ImageNet dataset is widely used as benchmark for CNNs on image recognition [21].

Concerning the benchmarking of architectures designed for image recognition, two measures are of importance when reviewing the evolution: their numbers of parameters and top five accuracy (*acc*@5):

$$acc@5 = 1 - \frac{1}{n}\sum_{i=1}^{n} \min_{j} d(\hat{y}_{ij}, y_i) \tag{1}$$

where n is the number of images, $\hat{y}_{ij}$ with $j = 1, \ldots, 5$ are the five predicted classes for an image with the highest probability, y_i the ground truth label of that image and for $d(\hat{y}_{ij}, y_i) = 0$ if $\hat{y}_{ij} = y_i$ and 1 if $\hat{y}_{ij} \neq y_i$. The top five accuracy is used since the images in the dataset might show more than one class but the ground truth labels y_i just label one class for each image in image recognition on ImageNet [21]. The higher is the *acc*@5, the better is the performance, where a smaller number of parameters enables more efficient processing and abstraction. Hence, the goal is to maximise *acc*@5 and minimise the number of parameters.

3.1.1. Vintage Architectures

Figure 4 shows the evolution of *acc*@5 performance for milestone architectures over time, where the size of a circle relates to the numbers of parameters in log scale. It becomes clear that, since AlexNet in 2012, the *acc*@5 first rapidly increased and until late 2015 saturates around 95% with a tendency of stable to smaller numbers of parameters. That leads to the questions of which major advances were introduced during the last seven years and which of them are still prominent in state of the art (sota) architectures in late 2019. **AlexNet** [3], named after Alex Krizhevsky, as well as its two follow up architectures **ZFNet** [38], named after the authors Zeiler and Fergus, and **VGG-19** [39], named after the Visual Geometry Group (see Figure 5), are here grouped together as *Vintage Architectures*. All three architectures are similar in their design: convolutions with non-linear activation and max pooling layers are repeated. With it, features are extracted from an input image by subsequently deeper feature maps with smaller resolution until a fully connected classifier head is reached. This classifier predicts on the extracted features and provides the probability for each possible class. Whereas AlexNet uses 11×11 and 5×5 kernel sizes for the first convolutional layers, also called the stem of a network, reaching 81.8% *acc*@5 at 62M parameters [3], ZFNet decreased the size to 7×7 and 5×5, improving *acc*@5 to 83.5% at 62M parameters. Zeiler and Fergus [38] argued that a smaller receptive field in the beginning of the network extracts more information. Due to smaller kernel sizes in the stem, they were able to extract more feature maps holding information and made their net even deeper by increasing the number of feature maps in each convolutional layer [3,38].

However, a significant leap in accuracy was possible with the VGG-19 architecture. Simonyan and Zisserman [39] introduced, as a relative to AlexNet and ZFNet, a deep network variant of 19 layers. The depth was reached by repeating building blocks, starting with a stack of convolutional layers up to four times and adjacent dimension reduction by max pooling. They also used 3×3 kernels exclusively in their convolutional layers, which is still common in recent architectures. The stacked 3×3 kernels are able to synthesise larger receptive fields by using fewer parameters. The main ideas of these *Vintage Architectures* can be summarised as:

- The convolutional backbone consists of repeated convolutions to increase the feature depth and some kind of resizing method such as pooling with stride 2 to decrease resolution.
- The ReLU activation after convolutional layers is used to speed up training with backpropagation with stochastic gradient descent [3,31–33,101].
- In VGG-19, the repeated building blocks with stacked convolutions of constant size enlarge the receptive field and deepen the network.

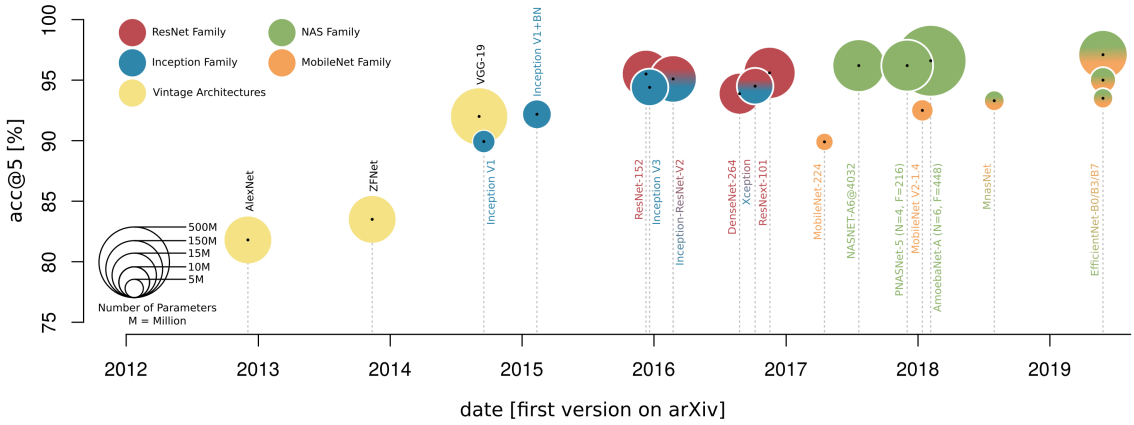

Figure 4. Evolution of milestone architectures for image recognition from 2012 to 2019, compared by their accuracy (*acc*@5) metric on the ImageNet 2012 dataset and showing their numbers of parameters in log scale as size of the circle. The different colours show relations between the architectures; mixed colours mean that concepts of two architecture families are combined in those architectures.

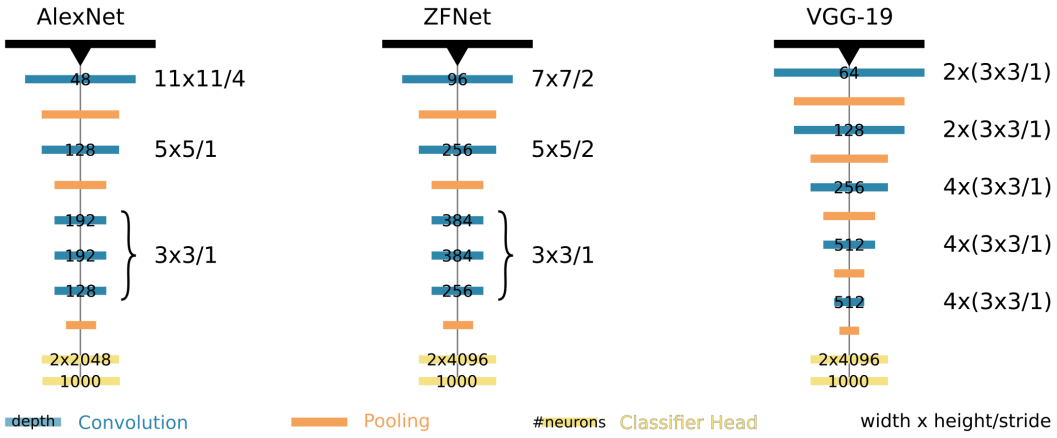

Figure 5. Conceptual overview of the *Vintage Architectures*. The architecture designs of AlexNet [3] and ZFNet [38] are similar despite smaller kernel in the first convolutional layers and deeper feature maps in ZFNet, whereas VGG-19 [39] is considerably deeper overall and it uses a uniform kernel size.

3.1.2. Inception Family

In the same year that VGG-19 was presented, the GoogLeNet variation with Inception modules [40] was introduced, a late successor to the initial work of LeCun et al. (1989) [30]. Since this should be the starting point of the *Inception Family*, the network is called **Inception V1** further on. The main idea of the network was that, after the stem of first convolutions, the novel Inception modules are repeated as building blocks with sporadic max pooling in between for dimension reduction (see Figure 6, right). An Inception module itself is made up of parallel convolutional layers of different kernel sizes and max pooling. As a result, an increased variety of feature representation is reached which is processed from the same input (see Figure 6, left). To avoid an explosion of parameters, so-called bottleneck layers were introduced in the beginning of the Inception module. These are 1×1 convolutional layers, which intermediately reduce the depth of an input tensor before it enters one of the next parallelised convolutions. Due to this depth reduction of feature maps, bottleneck layers lead to fewer parameters needed for each parallel operation but gain richer features when concatenating the results later on.

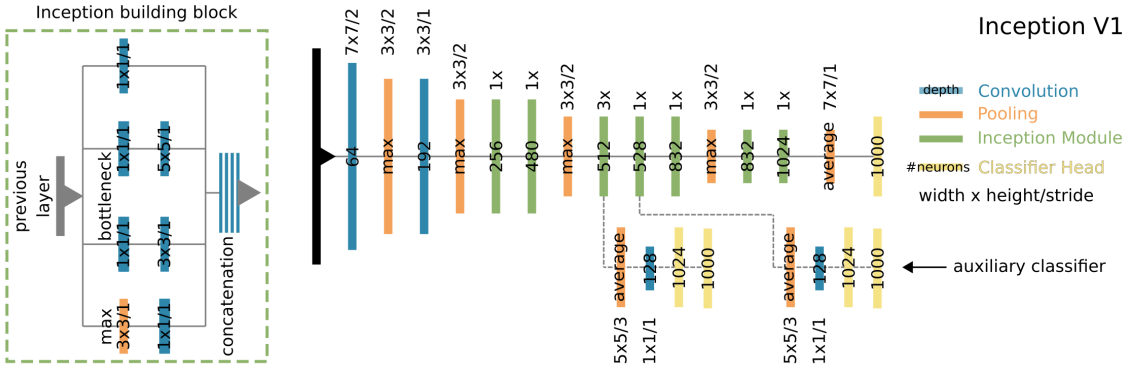

Figure 6. Conceptual overview of the Inception module and Inception V1 architecture [40]: (**Left**) the Inception module shows the 1×1 convolutional bottleneck layers, which reduce depth before the 3×3 and 5×5 convolutional operations; and (**Right**) the overall architecture of Inception V1 that consists of stacked Inception modules with increasing output depth and sparse max pool operations in between.

Therewith, Szegedy et al. [40] were able to build a 21-layer deep network, when counting the convolutional layers. Because of this deep design, backpropagating the gradients while training became increasingly difficult. To avoid or reduce any training effects on the early layers in the network due to vanishing gradients [5,40,41], two additional classifiers in the middle of the network were integrated. The task of the so-called auxiliary classifiers is to provide additive gradients to early layers, so that they have an additional training effect. The so-derived gradients are able to adjust even early layers in the network during backpropagation. During inference, the auxiliary classifier branches are not used and cut off as they are only helpful during training the weights. The resulting Inception V1 reached a *acc*@5 of 89.9% with only 6.8M parameters. This is much more efficient than the VGG-19 design with a slightly higher *acc*@5 score of 92% but many more parameters, at 144M.

Five months later, Ioffe and Szegedy [41] proposed an adaptation of Inception-V1 with a stem of stacked 3×3 conv layers such as VGG-19. However, what was more important for this novel implementation and even the whole DL field was the introduction of batch normalisation after convolution and before ReLU activation. Batch normalisation [41] together with appropriate parameter initialisation [18,32,102–104] and suitable activation functions [31–33,101] are part of the solution of solving the problems of vanishing and exploding gradients [32,105]. These concepts became highly important due to increasing network depth and the ability of those deep networks to converge during training. Enhancing the **Inception V1 with batch normalisation**, the updated variant was able to exceed VGG-19 in *acc*@5 performance, with 92.18% at 11.5M parameters being more efficient.

The idea of the Inception modules was further improved in **Inception V2 and V3**, where Szegedy et al. [42] applied factorisation on convolutions. Like VGG-19 they used stacked 3×3 convolutions to increase the receptive field. Factorisation can furthermore be used on a $m \times n$ kernel to split it into a stack of $1 \times n$ and $m \times 1$ kernels. Therefore, instead of $m \times n \times d$ parameters, $1 \times n \times d + m \times 1 \times d$ parameters are sufficient.

By applying factorisation on the original Inception module, three modified modules were created to increase the representational capacity of the model by using parameters more efficiently. The different modules are plugged into the network according to their ability to represent features in specific depths of the network, see Section 6 in [42]. The resulting Inception V3 network leaped up to an *acc*@5 of 94.4% using 23.6M parameters.

The idea of factorised convolutional filter banks can also be translated to a depth-wise factorisation, where convolutions are applied to each input channel separately. Chollet [45] assumed in his work "[...] that cross-channel correlations and spatial correlations can be mapped completely separately" (p. 1801). With this extreme idea of factorisation, Chollet [45] presented a network called **Xception**. Beside the stem, it exclusively uses depth-wise separable convolutions as feature extractors. Those are

1×1 or pointwise convolutions with adjacent 3×3 depth-wise convolutions for each output channel, without a nonlinear activation in between. Benchmark results of Xception are similar to Inception V3 with 94.4% *acc*@5 but slightly less parameters 22.8M. The *Inception Family* proposed and established the following cornerstones of sota CNNs:

- Bottleneck designs and complex building block structures
- Batch normalisation to make deep networks trainable faster via stochastic gradient descent
- Factorisation of convolutions in space and depth

3.1.3. ResNet Family

Beside the Inception modules, Xception also uses so-called residual connections, which lead to another stage of development, the *ResNet Family*. The first **ResNet** [43] was introduced in late 2015, at the time when Inception V3 was proposed. What gives this family its name is the above-mentioned residual connection used in the ResNet building block (see Figure 7). Technically, within the block's main trunk, the input depth is reduced by a 1×1 convolution, then features are extracted by a 3×3 convolutional layer and finally depth is increase again using 1×1 convolution. This bottleneck design is closely related to the slightly older *Inception Family*, focussing on parameter efficiency. What is new is what goes around the main trunk. Before the first convolution, the input is branched away from the trunk and merged back after the second 1×1 convolution by simply adding the values to the output of the trunk. This branch is called a shortcut or residual connection. In case that the input is not manipulated on the shortcut, it is called an identity shortcut. As the depth and resolution have to be the same of residual branch and main trunk for adding them back together, but still allow the depth to increase and resolution to decrease within the main trunk, the shortcut can also perform operations such as convolution [43] (see version 2 of the ResNet building block in Figure 7).

With the two ResNet building block designs, which are less complex than the Inception building blocks, it was possible to stack the components to very deep networks. However, increasing depth led to the counterintuitive degradation problem, which can be described as saturation, and rapidly decreased accuracy when using deeper architectures. Inception V1 [40] used the auxiliary classifiers to cope with this problem, but they made the network overly complex with additional branches. To avoid the degradation of deeper networks and support the optimisation with backpropagation, the above-described ResNet design reformulates the function $H(x, W_i)$, which describes the convolutional operations of the main trunk. In ResNet, it is now a residual function $H(x, W_i) := F(x, W_i) + x$, where $F(x, W_i)$ is the residue to approximate and x the input before the convolutional operation. Since the x is known due to the residual connection, approximating the residue is considered easier [43]. Furthermore, when letting the residue of convolutional operations in the final part of the network be zero, an identity function is learned from x. From the backpropagation point of view, it is now possible to transport larger gradients to convolution operations early in the network and enable optimisation of those layers. Hence, residual connections enable deeper networks to converge during training and perform better than shallow networks [43,44,46].

ResNet [43] was presented with multiple depths, where the ResNet-152 model (152 layers, see Figure 7), performed with an *acc*@5 of 95.5% at 60.3M parameters. Similar to the *Inception Family*, the ResNet idea evolved over time. **ResNeXt** [44] changed the ResNet building blocks by introducing so-called cardinality, which can be imagined as parallel convolutional operations with fewer feature maps in each parallel branch. This design widens the block to the next dimension, hence the name.

Technically, this was implemented by redesigning the 3×3 convolution operation in the main trunk of the block to be grouped convolution, introduced in **ShuffleNet** [106]. The use of 32 groups on a 128 deep input feature map leads to four feature maps per group, which are convolved together. ResNeXt-101 was able to improve *acc*@5 slightly up to 95.6% at 83.6M parameters.

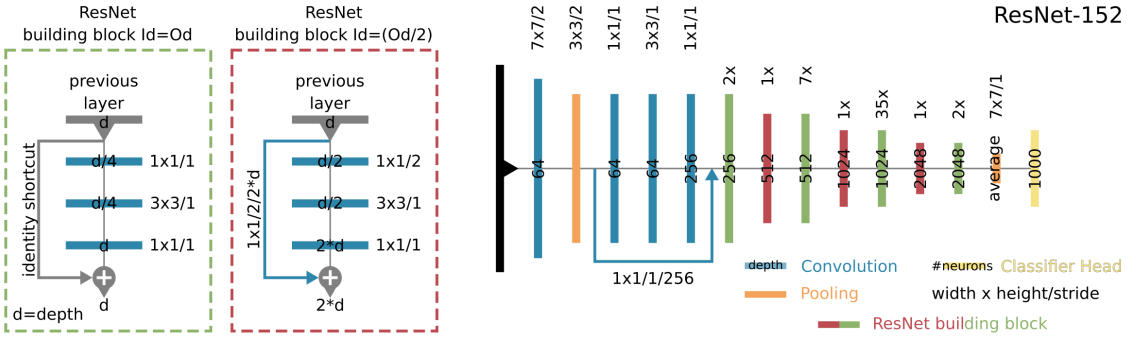

Figure 7. Conceptual overview of the ResNet building block and the ResNet-152 architecture [43]: (**Left**) The two ResNet building blocks show the convolutional operations in the main trunk, extracting features and the residual connections surrounding these operations to enable a more efficient transport of gradients during training. The difference between the two variants is that the first transports the input value of the block without changing it (identity shortcut) where the second uses convolution with stride 2, to match the output resolution and depth of the main trunk. (**Right**) Overall architecture of ResNet-152 consists of 152 layers due to stacked ResNet building blocks.

Another ResNet variant is **DenseNet** [46] which uses residual connections intensively. As the name suggests, in a DenseNet building block,. each convolutional layer takes as input the result of the previous convolution as well as all previous inputs within the block via multiple residual connections, forming a densely-connected building block. Instead of addition, DenseNet uses concatenation to merge the layers resulting in a very deep feature map as the output of a block. For this reason, before entering the next block, depth is reduced in a transition block using a bottle neck design. For a detailed description, we refer to the work of Huang et al. [46]. DenseNet is trailing ResNet and ResNeXt designs with an *acc*@5 of 93.9% for DenseNet-264, which uses significantly fewer parameters, just 34M.

Considering the network depth of the *Vintage Architectures* or the *Inception Family*, the *ResNet Family* led to very deep architectures, which were trainable without auxiliary classifiers by using the novel residual connection and the now established batch normalisation. Furthermore, the residual connection introduced by the *ResNet Family* is still a widely used component in CNN architectures and contributes to more sophisticated flow control in network architectures. Together, both families, *Inception* and *ResNet* contributed to the evolution by demonstrating that CNNs are highly modular models. Xception as well as the older **Inception-ResNet-V2** [107] are exemplary for merging the corner stones of each family into new architectures. The results are Inception-like modules with bottleneck designs and batch normalisation, enhanced by residual connections. Those networks are deeper than their ancestors without auxiliary classifiers and perform better or equal to those with fewer parameters.

Besides complete new architectures, plug in modules such as the Squeeze and Excitation (SE) module [47] demonstrated that the modification of existing model layouts can enhance even the *Vintage Architectures* in an efficient way. SE modules used in **SENet** are small fully connected neural networks which weight the feature map outputs of a convolutional operation. They support the hypothesis that not all features in a feature map are equally responsible for the final prediction. The usage of such weighted convolutional operations improves *Vintage Architectures* such as VGG as well as networks from the *ResNet* and *Inception Family* by adding few parameters to the network [47].

3.1.4. Efficient Designs

The modular concept of CNNs and their building blocks is crucial for the next group of architectures. In 2017, two major trends led to today's state of the art architectures in image recognition: highly parameter efficient networks and, instead of hand crafted designs, architectures drafted by other neural networks in a so-called Neural Architecture Search (NAS) [48,49].

To use DL to solve every day problems, the architectures had to run on mobile devices. The designs making this possible are grouped as *Efficient designs* in this review. Motivated by the restriction of computations for mobile devices, the lightweight **MobileNet** family was founded. The first MobileNet-224 [50] had only 4.2M parameters; nevertheless, it performed with an *acc*@5 of 89.9%. The network mainly consists of depth-wise separable convolutions, which use a highly parameter efficient stack of 3×3 convolution on each input feature map separately, with an adjacent 1×1 pointwise convolution across the entire depth. The next version, **MobileNet-V2** [108], further improves this idea by using mobile inverted depth-wise convolution with residual connections, pictured in Figure 8 (left). This describes a building block, which first performs an 1×1 pointwise convolution that expands the depth of feature maps for the adjacent 3×3 depth-wise convolution. Afterwards, another 1×1 pointwise convolution defines the output depth, which is normally smaller than the intermediate expansion depth. A surrounding residual connection adds the input to the output maps by connecting the bottleneck layer. MobileNet-V2 performed with an *acc*@5 of 92.5 at 6M parameters [108,109].

In parallel, a few months after the first MobileNet was introduced, **NASNet** [49] and therewith a new way to define architectures was added to the development of CNNs. Neural architecture search (NAS) follows the idea "[. . .] learning beats programming" of Krizhevsky et al. [110] (p. 84) not only for features but also whole architectures. In NAS, a defined search space of CNN building blocks was used by a controller, such as a recurrent neural network (RNN) [34], to find the best so-called child network architecture. The controller does this by using reinforcement learning which maximises the accuracy of prediction on an underlying dataset reached by the child during every iteration. Thereby, the RNN architecture in combination with reinforcement learning allows subsequently adapting the design of the child network. The reward signal, the accuracy performed by the resulting child network, is used to update the controller in order for it to produce a new child, which performs better in its defined task [48,49].

One drawback is that the briefly described search algorithm needs to train each child it produces. Since ImageNet is a relatively large dataset, it is common praxis to use a smaller dataset such as the CIFAR-10 dataset [111] during NAS. After the new architecture is defined, it is scaled up to match the larger variance of ImageNet without being trained on it each time [49,51,52,112–114]. Scaling can be done by simply repeating the NAS defined building blocks to build a deep CNN [49,112], or by defining a more complex scaling rule [52], which is mentioned below. The NASNet variant NASNet-A (6@4032), introduced by Zoph et al. [49], performed with an *acc*@5 of 96.2% at 88.9M parameters.

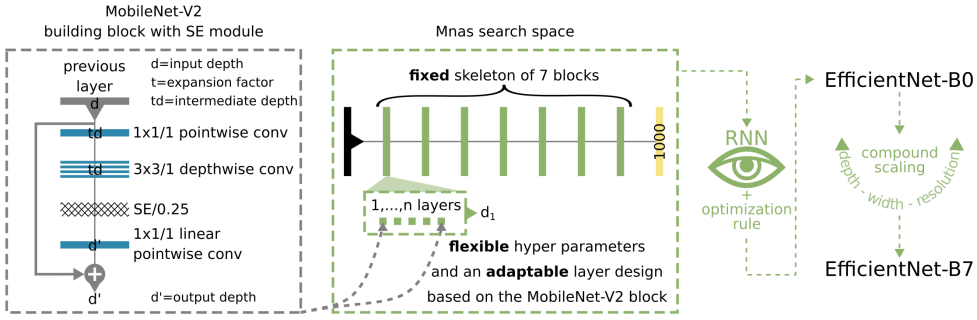

Figure 8. Conceptual overview of the *Efficient designs*: (**Left**) The MobileNetV2 building block [108], here with an additional Squeeze-and-Excitation (SE) module [47]. In comparison with a ResNet building block, the bottleneck design is inverted, so that first the expansion factor (t) is larger than 1, which leads to intermediate deeper feature maps (td) as the final output depth of the building block (d'). (**Middle**) The Mnas search space with a fixed overall architecture of the network, the skeleton, but fully optional layer designs, based on the MobileNetV2 building block [51]. (**Right**) A recurrent neural network (RNN) [34] controller searches the search space for the best performing combination of layer designs by maximising an optimisation rule [48]. The resulting architecture is scaled in depth, width and resolution to become the EfficientNet-B7 architecture, the sota design in late 2019 [52].

Further adaptions of the NAS approach, concerning how the children are handled and which search space is used, **PNASNet-5** (*acc*@5 = 96.2%, parameters = 86.1M) [112] and models of **AmoebaNet** (*acc*@5 = 96.3%, parameters = 155.3M, model = C) [113] outperformed hand crafted networks and established NAS.

Due to this new way of designing networks, the next generation of the MobileNet family used the NAS approach to generate the **MobileNASNet (MnasNet)**. The Mnas search space, pictured in Figure 8, defines a fixed skeleton of seven repeated blocks, where each block has n layers with $d_{1,\dots,7}$ output depth, where n and $d_{1,\dots,7}$ are searched. Furthermore, the layer model of each block is a SE module enhanced MobileNet-V2 building block, but each component of this block is optional and the layer design is searched by the RNN [51]. By increasingly more complex layer designs found during NAS, layer diversity was increased, which was found to be important for richer feature representations [112,115]. With *acc*@5 of 93.3% at only 5.2M parameters, the resulting MnasNet performed better than the hand crafted MobileNet-V2, on which it is based, with fewer parameters needed [51].

The same search space was used in late 2019 by Tan and Le [52] to create the last family of CNNs for image recognition reviewed in this paper, the **EfficientNets**. Since the search space remained unchanged and the optimisation rule was also similar to MnasNet, the resulting baseline model EfficientNet-B0 (*acc*@5 = 93.5%, parameters = 5.3M) performed similar to MnasNet. However, what makes EfficientNet successful is the scaling approach. Tan and Le [52] proposed compound scaling, which balances scaling in depth, width and resolution of the network (see Figure 8). The resulting EfficientNet-B7 architecture reached 97.1% *acc*@5 at 66M parameters, which means an increase in *acc*@5 of 15.3% at 4M more parameters used, compared to AlexNet seven years before, see Table 2.

EfficientNet [52] and its closest relatives, the MobileNets [50,51,108,114] and other NAS induced architectures [49,112,113], are based on the findings of their ancestors. The most relevant features and concepts from *Vintage Architectures* and the *Inception* and *ResNet Family*, which are bottleneck designs, the factorisation of convolutional operations and residual connections are the main components which are characteristic for the sequential evolution of CNNs. All of these components are finally included in the MobileNet building blocks and NAS search spaces (see Table 2). Furthermore, using NAS and scaling successfully demonstrates how flexible CNNs can be designed to fit specific tasks [52]. However, what has not changed over the whole evolution is the meta structure of CNNs. For image recognition, it still remains: input, convolutional backbone and a final classifier head.

Table 2. Summary of the evolution of convolutional neural networks (CNNs) for image recognition. Bottleneck and Factorisation refer to the design concepts mainly developed in the *Inception Family*, whereas Residual describes the use of residual connections introduced in the *ResNet Family*. NAS is neural architecture search, the most recent development for efficient CNN architectures. The number of parameters is given in millions (M). *acc*@5 describes the performance on the ImageNet dataset.

Architecture	Year	Bottleneck	Factorisation	Residual	NAS	M Parameters	*acc*@5 [%]
AlexNet [3]	2012					62	81.8
ZFNet [38]	2013					62	83.5
VGG-19 [39]	2014					144	91.9
Inception-V1 + BN [41]	2015	✓				11	92.2
ResNet-152 [43]	2015	✓		✓		60	95.5
Inception-V3 [42]	2015	✓	✓			24	94.4
DenseNet-264 [46]	2016	✓		✓		34	93.9
Xception [45]	2016	✓	✓	✓		23	94.5
ResNeXt-101 [44]	2016	✓		✓		84	95.6
MobileNet-224 [50]	2017	✓	✓	✓		4.2	89.9
NasNet [49]	2017	✓	✓	✓	✓	89	96.2
MobileNet V2 [108]	2018	✓	✓	✓		6.1	92.5
MnasNet [51]	2018	✓	✓	✓	✓	5.2	93.3
EfficientNet-B7 [52]	2019	✓	✓	✓	✓	66	97.1

Applying CNNs for image recognition on Earth observation data is normally done by classifying a small patch of a remote sensing image with one or multiple labels of land cover classes. An example dataset is the BigEarthNet dataset which contains 152 Sentinel-2 tiles divided into 590,326 patches labelled with 43 land cover classes [116]. Sumbul et al. [117] used a VGG-19 and ResNet-50 to predict labels on BigEarthNet and reached an average precision of 64.87% and 74.78%, respectively. Therewith, they demonstrated the increase in performance between this architectures even for Earth observation data.

3.2. Image Segmentation

As pointed out in the previous section, the features extracted by a convolutional backbone hold high level semantic information and are used for predicting classes of whole images. Image segmentation also uses this high level features extracted by a convolutional backbone, but predicts classes on pixel level, which leads to the following problem: within a convolutional backbone, the feature maps are subsequently resized to a lower resolution with increasing semantic information. Hence, they provide less accurate location information, which is necessary for accurate pixel wise prediction. As a result, image segmentation is confronted with a high resolution-feature depth trade-off.

Additionally, to predict the class of a single pixel, contextual relationship is important. This information can be found in a range of long and small distances around the pixel. The contextual information depends on the size and continuity of the semantic uniform segment the pixel belongs to, as well as the amount and density of neighbouring segments of other classes and background. Therefore, image segmentation can be seen as a multi scale context problem, even when prediction is made on single pixels. The advances in image segmentation are focussing strongly on solving this problem by exploiting features of different stages in the network or preserving or reconstructing high resolution during feature extraction and prediction.

Since image segmentation makes predictions on pixel level, the underlying benchmark dataset and metric to compare architectures has to be different from ImageNet and *acc*@5. In this review, the *Segmentation on PASCAL-VOC 2012* `test` *dataset* [53,54] is used and is further referred to as PASCAL-VOC, if not stated otherwise in this section. This specific dataset was chosen to give an overview due to its long lasting tradition over other more recently popular and challenging datasets such as the Cityscape dataset [118,119] to overlook the evolution since 2014. The most frequently reported metric for this dataset is the mean Intersection over Union (mIoU) over all classes [53,78].

$$\text{mIoU} = \frac{1}{C} \sum_{c=1}^{C} \frac{|y_c \cap \hat{y}_c|}{|y_c \cup \hat{y}_c|} \tag{2}$$

where C is the number of classes, $|y_c \cap \hat{y}_c|$ is the intersection between the ground truth y and predicted segmentation $\hat{y}$ per class and $|y_c \cup \hat{y}_c|$ is the union of ground truth y and predicted segmentation $\hat{y}$ per class. Its range is between 0% and 100% with the goal to maximise mIoU. Over the years, higher accuracies were achieved, as shown in Figure 9, similar to the evolution of image recognition.

In the following overview about image segmentation, focus is put on Fully Convolutional Networks (FCNs) inspired architectures. For a wider review of segmentation approaches which use LSTM or GANs we refer to Minaee et al. [78], Garcia-Garcia et al. [120]. Image segmentation with CNNs was strongly influenced by the work of Long et al. [55] who introduced FCNs in 2014. To further discuss the evolution of FCN architectures, they are separated into two groups: *naïve decoder* and *encoder–decoder* architectures. By using the term naïve decoder models, we follow, e.g., Chen et al. [60], who used this term to describe feature map upsampling mainly by bilinear interpolation and without using additional information from the encoder path. The overall idea of naïve decoder models (Figure 10, left) is to use a convolutional backbone to extract feature maps, upsample the feature maps to recover the input resolution by using bilinear interpolation and finally make pixel-wise prediction and optional post processing to gain segmentation masks.

In encoder–decoder models (Figure 10, right), the encoder part of the network can be seen as the convolutional backbone which extracts feature maps from input data. The decoder uses the final feature map in combination with spatially more accurate information from earlier stages in the encoder path. Within the decoder, the feature map is upsampled or deconvolved and additionally combined with spatially more accurate information transported from the corresponding layer in the encoder to the decoder via skip connections. After the input resolution is recovered, pixel wise predictions are made to produce the segmentation mask. The following overview of image segmentation architectures will first look at the original FCN, then focus on naïve decoder, characterised by the DeepLab family, and finally discuss encoder–decoder models.

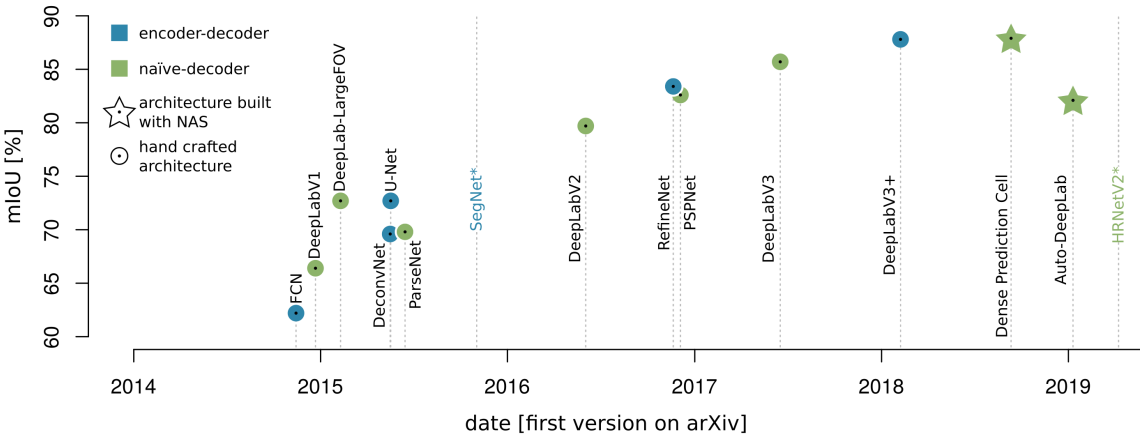

Figure 9. Evolution of milestone architectures for image segmentation from 2014 to 2019, compared by their mean intersection over union (mIoU) metric on the PASCAL-VOC 2012 `test` set for segmentation. Architectures marked with an * are tested on different datasets and no metric on PASCAL-VOC 2012 is reported.

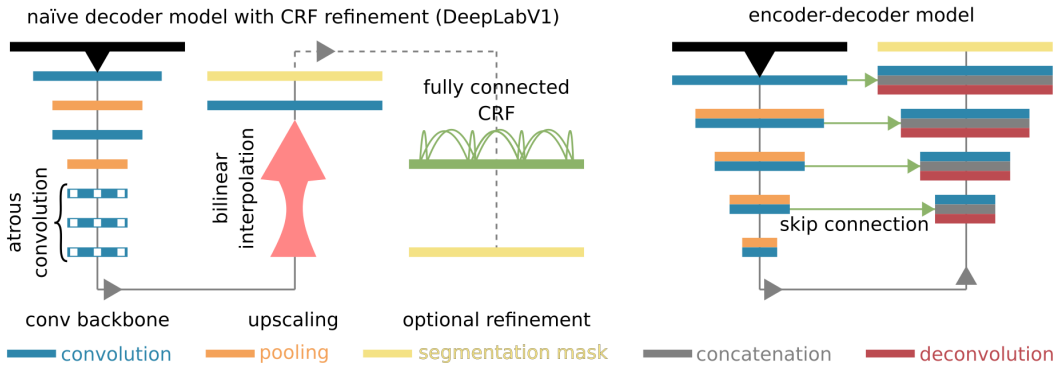

Figure 10. Conceptual overview of naïve decoder (**left**) and encoder–decoder design (**right**). As an example for the naïve decoder design, DeepLabV1 was chosen, which uses atrous convolution as the last layers in the backbone, bilinear interpolation as upscaling method and a fully connected conditional random field (CRF) module for refinement [56]. As example for the encoder–decoder models, an U-Net inspired design shows the skip connections, transporting feature maps from the encoder path to the corresponding decoder path, providing increasingly precise feature localisation during upscaling [67].

3.2.1. Naïve Decoder

In 2014, Long et al. [55] introduced **Fully Convolutional Networks (FCN)** for semantic segmentation. They used VGG-16 as backbone to create feature maps and trained deconvolutional layers to upsample them to input resolution. To increase the precision of fine grained features in their best performing FCN-8s variant (mIoU = 62.2%), they used feature maps from earlier layers in the VGG-16 backbone via skip connections during upsampling. With that, they provided a combination of

spatially more accurate and higher level semantic features for the pixel wise classification. Inspired by the FCN design, Chen et al. [56] introduced **DeepLabV1**. The major differences are their use of atrous convolutions, explained below and a naïve decoder performing upsampling by bilinear interpolation. Atrous or dilated convolutions, as depicted in Figure 11 (left), which are intensively used in the DeepLab family, are inspired by the algorithme á trous [121]. In atrous convolutions, "holes" are inserted in a convolutional kernel to increase the receptive field and at the same time maintain resolution in order to gain dense feature maps with high resolution [56,58]. Because of these dense feature maps, upsampling can be done with the more computational efficient bilinear interpolation, compared to trainable layers used 2014 Long et al. [55] in FCN. However, despite the dense feature maps, after pixel wise prediction, in DeepLabV1 [56], the segmentation map is further refined by a fully connected conditional random field (CRF) [61]. When published, DeepLabV1 performed with a mIoU of 66.4%. The disadvantage of this combination of CNN and CRF is that DeepLabV1 is not end-to-end trainable in one stage.

The later proposed **DeepLab-LargeFOV** [57] variation uses only 3×3 kernels and a larger dilation rate of 12 for the atrous convolution in order to generate a large receptive field. After CRF refinement, they reached 72.7% mIoU [57,58]. The successor of DeepLabV1, **DeepLabV2** [58], uses instead of an VGG-16 as backbone a ResNet-101 and the novel Atrous Spatial Pyramid Pooling (ASPP) module in order to introduce multiscale feature exploitation to the DeepLab family.

ASPP is motivated by SPPNet [74], which will be discussed further in Section 3.3. The core idea of ASPP can be described as a parallel multiscale exploitation of feature maps processed by atrous convolution of different rates, see Figure 11 on the right. Therefore, multiscale information can be extracted using an efficient network design, instead of computational expensive processing of multiscale images, so-called image pyramids. DeepLabV2 consists of an atrous convolution enhanced ResNet-101 backbone and an ASPP module of convolutional layers with four different dilation rates, 6, 12, 18 and 24. The output feature maps from ASPP are upsampled to input resolution by bilinear interpolation on which pixel wise prediction is performed. After an adjacent fully connected CRF refinement, DeepLabV2 reached a mIoU of 79.7% [58].

Even when the multiscale exploitation due to the ASPP module was successful, some problems were noted later on. By using the largest dilation rate of 24 within ASPP, one can imagine that the outer cells of the kernel can lie outside the input feature map. Hence, this atrous convolution is degraded to a 1×1 convolution of the centre, and the goal of extracting long distance context information by using large dilation rates is turned upside down [59].

Another contributing factor of long distance context on pixel-wise classification in segmentation architectures was investigated in **ParseNet** [65], which can be seen as an alternative to large dilation rates. Global image context was exploited by using global average pooling and fusing this branch of the network with the standard features extracted by convolution. ParseNet was able to reach 69.8% mIoU without CRF refinement and is an end to end trainable CNN for image segmentation.

Pyramid Scene Parsing Network (PSPNet) [66] goes even further with its Pyramid Pooling Module. This module pools from the input feature map on different scales, and applies 1×1 convolution and upsampling in order to concatenate context features from different scales with the original input feature map of the module. After a last convolutional operation on those fused features, the final pixel-wise prediction is performed. Therewith, local context of different scales is exploited for better classification. PSPNet with a ResNet backbone and without any CRF refinement performed at 82.6% mIoU. The described advances, such as the extraction of features from multiple scales, the usage of contextual information from global and local scales and relying completely on a CNN model without combining it with graphical models for refinement, were aggregated in **DeepLabV3** [59] (see Figure 11, right). Its structure is the same as in DeepLabV2 [58], but changes were made in the ASPP module. Chen et al. [59] removed the largest dilation rate, due to the problem described above and instead introduced image level feature extraction by global average pooling. Therewith, long range information on image level, which was used successfully in ParseNet [65] and

PSPNet [66], was incorporated into the ASPP module in DeepLabV3. Upsampling is still done with bilinear interpolation until input image resolution is reached, but no CRF refinement is applied after prediction of the last convolutional layer that produces the segmentation mask. DeepLabV3 is therefore the first end to end trainable DeepLab variant, and, when published, it performed with state-of-the-art results of 85.7% mIoU [59].

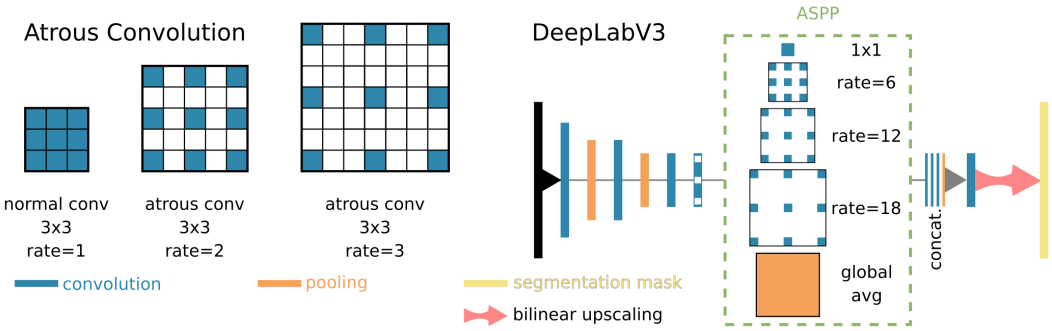

Figure 11. Conceptual overview of atrous convolution (**left**) and the Atrous Spatial Pyramid Pooling module (ASPP) within the DeepLabV3 [59] architecture (**right**). In atrous convolution, "holes" are inserted into the kernel. They provide a larger receptive field and maintain resolution at the same time. The ASPP module combines atrous convolutions of different rates and global image context via global average pooling in order to exploit feature maps on different scales efficiently [56,58,59].

The last naïve decoder model reviewed here is **HRNetV2** [71], published in 2019. Since no metric is reported on PASCAL-VOC but on the Cityscape dataset, HRNetV2 cannot directly be compared with the other architectures discussed before. To narrow this gap, we compare it to the reported performance of DeepLabV3 on the Cityscape test set with 81.3% mIoU. Both HRNetV1 [70] and its successor HRNetV2 [71] use four stages of five convolutional blocks. From the second stage, parallel convolutions on different resolutions are performed and finally merged in the fifth convolution by upsampling. With it, in Stages 2–4, features of different resolutions are combined with every other resolution in the so-called multi-resolution block. The difference between HRNetV1 and HRNetV2, is that V1 only uses the block of highest resolution for final prediction, whereas V2 uses all blocks of all resolutions by combining them using concatenation and stronger upsampling of low resolution feature maps. The final input resolution for prediction is reached by further upsampling with bilinear interpolation. The performance of HRNetV2 on the Cityscape test set is 81.6% mIoU, slightly better than DeepLabV3 [70,71].

3.2.2. Encoder–Decoder Models

All of the image segmentation architectures reviewed until now, besides FCN [55], share relatively naïve decoders. This means that mostly bilinear interpolation was used to upsample the feature maps back to input image resolution. The encoder–decoder models are different from that by introducing more complex decoder (see Figure 10, right). The main idea is to use shortcuts or skip connections to transport information from the encoder branch to the decoder. Therewith, high level feature from the later layer in the encoder are subsequently fused with more locally precise information from early layers in the encoder during upscaling.

In 2015, Noh et al. [64] introduced **DeconvNet**, which uses a VGG-16 as encoder and a mirrored VGG-16 as decoder. Within the decoder, instead of pooling, unpooling layers use the spatial localisation recorded during pooling, called pooling indices. With this information, a sparse feature map of higher resolution is restored and an adjacent deconvolution layer densifies them. Therewith, the decoder reaches the input resolution where pixel wise prediction is performed to generate the segmentation mask. DeconvNet reached a mIoU of 69.6%, matching the DeepLab version at this time without using additional refinement with a CRF.

In **SegNet**, Badrinarayanan et al. [122] used a similar approach and added batch normalisation with ReLU activation for each convolutional layer in the encoder and decoder. SegNet outperformed DeconvNet slightly; those results were reported not on PASCAL-VOC but the CamVid dataset [123].

In 2015, Ronneberger et al. [67] proposed **U-Net**, an encoder–decoder architecture whose structure has been applied widely in many domains since it was introduced. U-Net, originally built for medical image analysis for cell tracking, is similar to SegNet [55] and ConvNet [64]. Its encoder is made up of five building blocks where each consists of two adjacent 3×3 convolutions which double the amount of feature maps subsequently from 64 up to 1024. Between the blocks, the feature maps are downscaled with 2×2 max pooling of stride 2. The decoder uses deconvolution layers for upscaling within five blocks, where each block receives the whole feature maps from the encoder path of the same resolution, not only the pooling indices. The feature maps are transported via skip connections from the encoder to the corresponding building block in the decoder. Feature maps coming "up" within the decoder are concatenated with the feature maps from the encoder. Thereafter, high level semantic information is combined with more precise local information but lower semantic meaning during upscaling. When input resolution is finally restored, a last 1×1 convolution predicts the segmentation map [67]. Since U-Net was developed for a specific cell tracking task, no results on PASCAL-VOC are reported in the original publication. However, Zhang et al. [124] reported a 72.7% mIoU for a vanilla U-Net on PASCAL-VOC. The U-Net design was modified, for instance in **Tiramisu** [68], which combines U-Net structure and DenseNet [46] building blocks, and in **U-Net++** [125], where nested dense building blocks perform convolution on the skip connections.

With **RefineNet** [69], an encoder–decoder design was proposed, which focusses on residual connections. The encoder is built on a ResNet-152, the decoder of RefineNet blocks, which are using residual connections with sparse nonlinear activation. With this combination in RefineNet, a direct flow of features within the decoder was emphasised. RefineNet reached a mIoU of 83.4%.

Due to the convincing results achieved by encoder–decoder architectures, the DeepLab family also modified their naïve decoder model to an encoder–decoder design in **DeepLabV3+** [60]. DeepLabV3+ uses a modified Xception backbone in the encoder. The modifications were inspired by the depth-wise separable convolutions [45] and the last convolutional layers became separable atrous convolutional layers. Due to this factorisation in depth, the model gained computational efficiency. With an overall stride of 16 for the encoder, the decoder first bilinearly upsamples the encoded feature maps by a factor of 4. After that, the corresponding feature maps from the encoder are concatenated via skip connections and after additional convolutional operations an adjacent bilinear upsampling of factor 4 restores input resolution. On these feature maps, the final prediction is performed to produce the segmentation mask. This effective decoder model in combination with the modified DeepLabV3 model reached a mIoU of 87.8%.

Similar to image recognition, after architectures were designed and optimised heavily by hand, NAS was used to search for high performance and computationally efficient image segmentation architectures. **Dense Prediction Cell (DPC)** [62], which belongs to the DeepLab family, and Auto-DeepLab [63] are two examples based on NAS [48]. DPC focusses on the optimisation of the ASPP module by searching for, e.g., dilation rates and the grid size of average spatial pyramid pooling [62]. DPC performs with 87.9% mIoU on PASCAL-VOC using a modified Xception backbone [62].

In contrast, **Auto-DeepLab** [63] focusses on searching for an image segmentation optimised backbone instead of optimising a single module. That means that in addition to the operations in a building block, the overall network structure is searched too. Hence, no fixed skeleton is defined such as in Mnas searchspace [51] or the DPC. For further details on the searchspace, we refer to the work of Liu et al. [63]. The Auto-DeepLab-L variant reached a mIoU of 85.6% but performed comparatively better on the more challenging Cityscape dataset, where it reached the same mIoU of 82.1% as DeepLabV3+ [63].

Reviewing the advances in CNNs for image segmentation, it was pointed out that multiscale feature and image context exploitation by maintaining high resolution and fusing features from

different stages of the model are the cornerstones of sota architectures. Atrous convolution which are mostly used in naïve decoder models and skip connections which are typical for encoder–decoder models, are major concepts that were established during the evolution of image segmentation architectures. Table 3 summarises this evolution and demonstrates the decreasing use of CRFs for refinement and increasing use of atrous convolution and multiscale feature extraction.

Table 3. Summary of the evolution of convolutional neural networks (CNNs) for image segmentation and the above discussed cornerstones. The Backbone is the reported CNN used for feature extraction; CRF describes the use of a conditional random field for refinement; Atrous shows the application of atrous convolution; Multiscale shows if a dedicated module that handles multiscale feature extraction, e.g. ASPP, is used; NAS means that parts of the architecture are drafted by using neural architecture search; and mIoU describes the performance on the PASCAL-VOC 2012 dataset for image segmentation.

Architecture	Year	Backbone	Type	CRF	Atrous	Multiscale	NAS	mIoU [%]
FCN-8s [55]	2014	VGG-16	encoder–decoder					62.2
DeepLabV1 [56]	2014	VGG-16	naïve decoder	✓	✓			66.4
DeconvNet [64]	2015	VGG-16	encoder–decoder					69.6
U-Net [67]	2015	Own	encoder–decoder					72.7
ParseNet [65]	2015	VGG-16	naïve decoder			✓		69.8
DeepLabV2 [58]	2016	ResNet-101	naïve decoder	✓	✓	✓		79.7
RefineNet [69]	2016	ResNet-152	encoder–decoder					83.4
PSPNet [66]	2016	ResNet-101	naïve decoder			✓		82.6
DeepLabV3 [59]	2017	ResNet-101	naïve decoder		✓	✓		85.7
DeepLabV3+ [60]	2018	Xception	encoder–decoder		✓	✓		87.8
DensePredictionCell [62]	2018	Xception	naïve decoder		✓	✓	✓	87.9
Auto-DeepLab [63]	2019	Own	naïve decoder		✓	✓	✓	82.1

When CNNs for image segmentation are applied to Earth observation data, encoder–decoder models are a popular choice (see Section 4.2). The ISPRS 2D labelling dataset [126] is a widely used benchmark dataset consisting of a digital surface model and multispectral aerial images with very high spatial resolution. One example investigation of this dataset by Wang et al. [127] shows the application of the above-discussed DeepLabV3+ based on a ResNet-101 backbone. They modified this combination with an auxiliary loss directly after the ASPP module and a final CRF refinement and therewith demonstrated the applicability of a sota CNN coming from computer vision and its adaptation to the needs of Earth observation data.

3.3. Object Detection

While image recognition and image segmentation can be modelled as classification problems, object detection is a multi-task problem. Predicting the object class remains a classification problem, whereas predicting the location, which is a bounding box around each predicted object, is a regression problem. Therefore, the benchmark dataset has to contain additional bounding box labels and architectures have to handle both classification and regression at the same time. The used benchmark dataset is the object detection on Microsoft Common Objects in Context (MS-COCO) test-dev set [72]. The measure of interest is the mean Average Precision (mAP) or in the case of MS-COCO the AP, which is considered the same (see [128]).

MS-COCO's AP, also called $AP_{[0.5:0.05:0.95]}$, is an average over all classes for 10 different IoU (Intersection over Union) levels, between ground truth and predicted bounding boxes. The IoU threshold defines if a prediction is a true positive. By taking different IoU levels into account, models with more accurate localisation characteristics are favoured. The calculation of AP for a single IoU is the average of interpolated precision values p_{interp} at 101 equally spaced recall values $r = \in 0.0, \ldots, 1.0$ [128,129]. Precision and recall are thereby:

$$Precision = \frac{TP}{TP + FP} = p \tag{3}$$

$$Recall = \frac{TP}{TP + FN} = r \tag{4}$$

where TP are true positives, FP false positives and FN false negatives. The empirical precision recall curve $p(r)$ is interpolated with

$$p_{interp}(r) = \max_{\tilde{r}:\, \tilde{r} \geq r} p(\tilde{r}) \tag{5}$$

in order to generate $p_{interp}(r)$. Finally AP is calculated with [53,129]:

$$AP = \frac{1}{101} \sum_{r \in \{0.0, \ldots, 1.0\}} p_{interp}(r) \tag{6}$$

Other jointly reported metrics beside AP on MS-COCO are AP_{75} and AP_{50} with a single fixed IoU of 75% and 50%, respectively. Since MS-COCO was introduced in 2014, the first architectures reviewed use the PASCAL-VOC 2007 test set of the object detection task [54]. For these architectures, the mAP is reported, which is calculated similarly to MS-COCO's AP but with a fixed IoU and 11 instead of 101 bins of the precision–recall curve (see Equation (6)).

Architectures for object detection can be divided into two groups: two-stage detectors and one-stage detectors. In general, two-stage detectors perform with higher AP, which is shown in the progress made from late 2013 to 2019 in Figure 12. One-stage detectors are lightweight in parameters and complexity and thereby faster, measured in processed frames per second (fps) [130]. On a closer look, two-stage detectors are distinguished by a first stage, which processes class agnostic region proposals. The object class prediction of those potential regions and the final bounding box regression are then performed in the second stage. On the other hand, one-stage architectures perform class prediction and bounding box regression in a single shot from the input image. One-stage detectors play an important role in the evolution of CNN based object detection and are also widely used in applications and research [130–132]. Therefore, some popular designs are reviewed below. However, focus is put on two-stage detectors, and, among those, especially on the Region based CNN (R-CNN) family. For a wider overview on object detection, we refer to the works of Liu et al. [130], Wu et al. [132], Zhao et al. [133] and Jiao et al. [134].

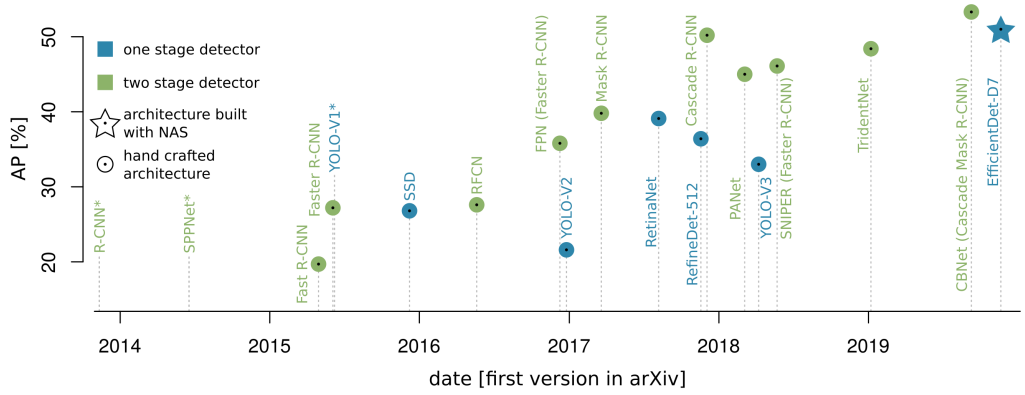

Figure 12. Evolution of milestone architectures for object detection from late 2013 to 2019, compared by their average precision (AP) metric on the MS-COCO test-dev set for object detection. Architectures marked with an * are tested on different datasets and no metric on MS-COCO test-dev set is reported.

3.3.1. Two-Stage Detectors

In 2013, Girshick et al. [73] introduced **Region based CNN (R-CNN)**, which incorporated a CNN into a pipeline of established graphical models and classifiers in order to predict objects in images. From the input image, class-agnostic region proposals are generated with the selective search algorithm of Uijlings et al. [135]. For each proposal produced this way, AlexNet [3] or VGG-16 [39] is used to

extract meaningful features for each proposal. Those features are then forwarded to binary class specific support vector machine (SVM) classifiers predicting the object class. To improve localisation, a class specific bounding box regression is applied using the initial region, coming from the selective search algorithm as a starting point. R-CNN with the bounding box regression scored a mAP of 66% on PASCAL-VOC 2007. However, R-CNN has some drawbacks, due to the property of performing tasks in a repetitive manner, e.g., feature extraction with a CNN for each region proposal or class specific SVM classifiers. Furthermore, the model is more of a model composition and therefore no end to end training is possible [73,76].

Spatial Pyramid Pooling (SPP) used in **SPPNet** by He et al. [74] and a redesigned R-CNN inspired architecture fixed some of the above mentioned issues. SPP is not a novel idea, but using it in a CNN context was a novel application. SPP pools features from arbitrary input sizes on different scales resulting in fixed length outputs. This enables processing on multiple scales which makes the object detection task more robust since objects are naturally appearing at different scales in images. Within the convolutional backbone, the SPP layer replaces the last pooling operation [74].

Another change in the overall design of SPPNet made it up to over one hundred times faster than the original R-CNN. This speed up was achieved by applying the convolutional backbone with adjacent SPP to extracted features only once on the whole input image instead of processing each region proposal coming from selective search separately. Therefore, the region proposals derived from the same input image with selective search are downscaled in order to match the resolution of the shared feature map to extract the corresponding features from there. SPPNet with bounding box regression performed with a mAP of 59.2% on PASCAL-VOC 2007 using a fast variant of ZFNet as convolutional backbone. This is less, measured by mAP, but 38 times faster than R-CNN [74].

Fast R-CNN was introduced by Girshick [75]. Similar to SPPNet, it also exploits shared feature maps for the regions proposed by selective search and established the use of shared feature maps as the standard approach for two stage object detection. An other similarity to SPPNet is the RoI (Region of Interest) pooling layer in Fast R-CNN, which connects convolutional backbone with a RoI wise classifier and bounding box regression. It performs max pooling by dividing the RoI provided by selective search into a feature map of fixed resolution. A clear difference from SPPNet is that Fast R-CNN introduced a multitask head that performs classification and regression within fully connected layers. Besides the region proposal, feature extraction and object detection are within one model, and no further SVMs are necessary. This was possible due to the definition of a multi task loss which sums the loss of regression and classifier head. Due to this combined design, training became much more efficient. Nevertheless, since selective search is still used for proposing regions the model is not end-to-end trainable. Using a VGG-16 backbone, Fast R-CNN performed with a mAP of 66.9% on PSCAL-VOC 2007. A first performance on MS-COCO is also reported with an AP of 19.7% [75].

The next successor from the R-CNN family is **Faster R-CNN** by Ren et al. [76], developed in 2015. With it, the Region Proposal Network (RPN) module was introduced and finally two stage object detection became end to end trainable models, unified in a single network performing all tasks needed for competitive object detection results. After the convolutional backbone the RPN, a small fully convolutional network is inserted. At each position of its sliding window, k translation invariant anchor boxes are investigated, where k is 9 in the proposed method, three scales and three aspect ratios. Since multiple anchors are predicted at the same sliding window position, heavy overlapping and overly noisy object proposals would be the result. To mark a region proposal as valid, it must be the region with the highest IoU score or a score greater than 0.7, whereas a proposal with an IoU less than 0.3 is marked as a negative example. Each positive anchor is then regressed to the proposed object boundary and is finally used as an RoI. Beside the RPN module, the architecture is the same as for Fast R-CNN: RoI pooling with adjacent multitask object class predictor and final bounding box regression. Based on a VGG-16 backbone, Faster R-CNN performs with 21.9% AP [76], changing the backbone to ResNet-101 AP goes up to 27.2% [101].

Further advances in gaining higher AP on MS-COCO address the following dilemma: for classification, high level semantic features are optimal but they lack localisation. At the same time, object detection needs precise local information to find objects of different scales and potentially densely packed in the image. These issues are comparable to the mentioned problems in image segmentation, in Section 3.2, and thus are the solutions.

Feature Pyramid Network (FPN) (see Figure 13), introduced in 2016 by Lin et al. [77], enables the inherent bottom up feature pyramid of the convolutional backbone with an adjacent top down path. The novel top down path is upsampled by factor of two beginning with the last layer of the convenient bottom up path by using nearest neighbour interpolation. In addition, each upsampled level is connected via lateral connections to the corresponding bottom up feature maps. In that way, fine grained low level features with high location information are passed to upscaled coarse high level semantic features. This idea is comparable with encoder–decoder models but as part of the convolutional backbone for object detection. Therefore, the feature pyramid defined by the levels of the novel top down path holds features in different scales and localisation precision, all enhanced by high level features. Each of the feature pyramid levels is connected to a RPN with three aspect ratios but no scale ratio for the anchors, since the different scales are now provided by the FPN levels. The FPN enhancement of a Faster R-CNN with ResNet-101 backbone reached 35.8% AP, an increase of 8.6%.

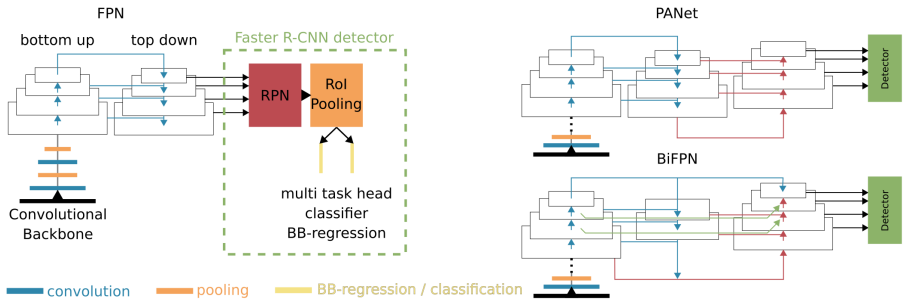

Figure 13. Conceptual overview of Feature Pyramid Network (FPN) and its variants. In the original FPN [77] (left), the last layers of the convolutional backbone (bottom up path) are paired with a top down path which uses the bottom up feature layers to enhance the semantic high level information with precise localisation. Prediction of region proposals is then done on the different layers of the top down feature pyramid. The two variants Path Aggregation Network (PANet) [82] and bidirectional FPN (BiFPN) [93] enhance the structure with an additional bottom up path (red arrows) and intra-scale skip connections (green arrows), respectively.

In 2017, with the advances in CNN architectures, made in object detection and image segmentation, He et al. [80] presented an end to end trainable DL model for instance segmentation **Mask R-CNN**. Besides the two heads for classification and bounding box regression, a third head which performs instance mask segmentation was added to the architecture. By using the features within a RoI, binary class specific masks are predicted with the FCN for image segmentation introduced by Long et al. [55]. Due to the higher spatial precision needed for image segmentation, RoI pooling was adapted to be RoI align. Therefore, the RoI coordinates were represented as floating point numbers instead of quantising them to discrete granularity. To extract the feature values for one RoI bin, values are also sampled at four equally spaced points, bilinearly interpolated and aggregated by using max or average pooling to finally represent the feature value in the RoI bin. Since standard object detection is still possible with instance segmentation architectures, performance is reported on MS-COCO object detection task with 39.8% AP [80].

Another optimisation of Faster R-CNN was introduced by Cai and Vasconcelos [79] who proposed **Cascade R-CNN**. Looking back at RPN, which, since it was introduced, is the state-of-the-art region proposal method for two-stage detectors, a positive anchor is mainly defined by an IoU between 0.5 and 0.7 during training. The trade-off between a high and small IoU is that a small IoU proposes

more regions of interest as positive. This leads to noisy object proposals, whereas a high IoU declines the hypothetical bounding box coming from RPN as too weak, resulting in false negatives. Therefore, a cascading IoU with increasing bounding box refinement, which starts at a low IoU, was introduced to tackle this problem. Starting with the proposals from RPN, RoI align extracts the features for anchors with an IoU of 0.5 and a first classification and bounding box regression is performed. The bounding box results are then used as if they were produced by RPN but now an IoU of 0.6 is applied. Again, RoI align extracts the features for the new, more accurate region proposals from the same feature map as before and the next classifier and bounding box regression use them. This is repeated for a third time with an IoU of 0.7 which produces the final results of the object detection. Using ResNet-101 as backbone, Cascade R-CNN performed with an AP of 42.8% on MS-COCO [79]. A later implementation for Cascade Mask R-CNN which uses a ResNeXt-152 backbone reached 50.2% [136]. Rethinking the feature pyramid of FPN, high semantic features are passed within the top down path to finer grained feature maps, with weaker semantic but better localised features, and thereby enhancing those. However, no fine grained information is directly passed to the first top down pyramid level, which holds coarse but high level features. **Path Aggregation Network (PANet)** [82] adds an additional augmented bottom up path with access to low level, highly localised features via lateral connections directly after the FPN inspired top down path. The feature pyramid of this second bottom up path therefore holds both high semantic feature and low semantic highly localised feature enhanced activation maps. The following architecture, which uses the Mask R-CNN detector head and a PANet enhanced ResNeXt-101 backbone, reached an AP of 45% [82].

FPN became a popular method to handle the important scale variation in object detection; however, different approaches do exist such as SNIP [83] and its successor SNIPER (Scale Normalisation for Image Pyramids with Efficient Resampling) [84] focussing on multi scale training. Another example for the handling of scale by using atrous convolution is the scale-aware **TridentNet** [85]. The last convolutional blocks of a backbone are replaced with Trident blocks. Those, as the name suggests, consist of three branches of atrous convolution with three different dilation rates. The branches share weights, to prevent overfitting due to potentially tripling the amount of parameters. Their detection is aggregated by none maximum suppression (NMS): for small objects, the branch with the smallest dilation rate finds the strongest activation and suppresses weaker activations from the other two branches and vice versa. After applying a scale aware training scheme, the TridentNet-enhanced ResNet-101 Backbone, which ends in a common Faster R-CNN detector, reaches an AP of 48.4% [85].

With FPN and TridentNet, the deeper layers in convolutional backbones were successfully enhanced in order to produce richer features for the subsequent detector. Nevertheless, the backbones remain architectures designed for image recognition tasks. However, object detection specific backbones have to be trained from scratch. That means, that no pre-trained parameters can be utilised for refinement, but all initialised values have to be solely trained on a given object detection dataset, which is a time and processing power consuming task [81]. Therefore, further enhancements of image recognition backbones pre-trained on ImageNet were investigated. Instead of changing some layers in a single backbone, a couple of the same backbones, differentiated into assistant backbones and one lead backbone, are connected to a **Composite Backbone Network (CBNet)** [81]. The novel composite connections of the neighbouring networks are lateral connections in each stage of the neighbouring backbones, following the so-called Adjacent Higher-Level Composition scheme. After each convolutional block in a network, the feature maps are passed to the neighbouring network as input to the same stage. Therefore, Adjacent Higher-Level Composition upsamples the feature map and performs a 1×1 convolution in a bottle neck style, to reduce feature map depth. Following that approach, features are passed from assistant to assistant until the lead backbone is reached, which then passes the feature maps from different stages and therefore in different scales to the detector. By using three connected backbones as CBNet and the most recent member of the R-CNN family, the Triple-ResNeXt-152 Cascade Mask R-CNN model reaches state-of-the-art AP of 53.3% on MS-COCO in 2019.

With RPN, FPN, RoI pooling the later RoI align, cascading classifier and regression heads and finally composite backbones, the most important modules and insights for object detection with two-stage detectors were developed in close relation to the R-CNN family. However, those modules are not exclusively made for the R-CNN design and most of them not even for two-stage detectors. This means that the introduced models and insights for object detection with CNNs are flexible and a foundation for highly task specific architectures. However, this flexibility comes at a price of highly complex models. One-stage detectors, on the other hand, tend to be less complex, by using predefined anchors to extract features after the convolutional backbone and passing them directly to a detector head, resulting in computational efficient and more streamlined architectures. To complete the review on object detection models, a brief overview about a selection of one-stage detectors is now provided.

3.3.2. One-Stage Detectors

In 2015, Redmon et al. [86] introduced the first member of the YOLO (You Only Look Once) family, called **YOLO-V1** in this review. The motivation for the whole YOLO family is representative for one-stage detectors: one-stage detectors are lightweight models that perform accurate real time detection of more than ~20 fps to bring object detection to mobile platforms, for example. The YOLO-V1 detector separates the feature map which comes from the backbone into $S * S$ cells. In each cell, YOLO-V1 looks for object centres and predicts object class and bounding box simultaneously. This approach results in many boxes which are then sorted out by NMS of a class agnostic objectness score. On PASCAL-VOC 2007, YOLO-V1 performed with mAP of 36.4%, trailing the Faster R-CNN at the time, but with 45 fps, where the Faster R-CNN processed at 7–18 fps, depending on the backbone [86]. **YOLO-V2** [87] introduced multi scale training, a custom and fast backbone called DarkNet-19, batch normalisation after convolutional layers, a fully convolutional design and more important the use of anchor boxes. The dimensions of the anchors are chosen by clustering the training dataset using k-means with five cluster centres. This approach is crucial, since well-chosen anchors have a high impact on the detection of objects. Thereafter, YOLO-V2 predicts five bounding boxes at each of the pixel of the last feature map after the backbone. The performance of the largest model with 17×17 pixels in the last feature map, YOLO-V2 performed on PASCAL-VOC 2007 with 40 fps and a mAP of 78.6%. Notably, it matched sota performance on this dataset and 21.6% AP on MS-COCO trailing sota models at the time of its publication.

Finally, **YOLO-V3** [88] uses ResNet-inspired building blocks for its DarkNet-53 backbone and a FPN-inspired detector that passes features from high semantic to lower semantic layers. Due to this enhancement in scale invariance, the issue of its ancestors, which perform poorly on smaller objects, was tackled [86,88]. The model detects on three different scaled feature maps using nine ratios for the anchors, defined by k-means similar to in YOLO-V2, resulting in three anchors per scale. YOLO-V3 performs with 33% on MS-COCO now trailing sota networks of two and one stage designs, but outperforms them in speed by a factor of 2.32 compared to RetinaNet [88,90]. The YOLO-V3 design is therefore partly similar to the earlier proposed **Single Shot MultiBox Detector (SSD)** [89], which also performs on different scaled feature maps, but uses manually predefined anchors. With a VGG-16 backbone, its performance was reported with an AP of 26.8%.

The above-mentioned **RetinaNet** [90] mainly focusses on an adaptive loss function, called focal loss, to tackle the problem of unbalanced foreground-background classes, explained here. Due to the lack of a RPN module, one-stage detectors sample many predefined anchors over the whole feature map in a dense manner. This leads to many boxes which classify background, easy negatives, and just a few boxes which classify objects. This property of one-stage detectors is used to provide small gradients for harder to classify foreground examples and an overwhelmingly large loss contribution, and therefore large gradients for easier to classify background examples. This unbalanced loss leads to minimal training effect considering the real objects. To overcome this problem, the focal loss down-weights the loss contribution which comes from easy negatives, in order to focus more on the harder to classify foreground examples. Since the advances of RetinaNet focus on the loss, its architecture consists

mostly of established modules such as a FPN enhanced ResNet-101 backbone connecting to a fully convolutional classifier and bounding box regression. Nevertheless, due to the focal loss, RetinaNet matches sota performance of 39.1% AP at 5 fps.

In late 2019, Tan et al. [93] introduced **EfficientDet**, a one-stage detector which further filled the gap towards two-stage detector performance. Based on the novel EfficientNet family for image recognition, which they use as backbone, they further used its compound scaling approach [52]. Instead of a FPN at the end of the EfficentNet, they redesigned it to BiFPN. In it, they use the improvements of PANet, an additional bottom up path, and orthogonal to the common top down and bottom up paths, cross scale lateral connections, first proposed in NAS-FPN [92] (see Figure 13, right). Feature connectivity across the FPN module is therefore enhanced even on a consistent scale level within the feature pyramid by a bidirectional flow, hence BiFPN. The best performance on MS-COCO was reached with the EfficientDet-D7 variant using auto-augmentation [137] with 52.2% AP [93].

As in image segmentation, object detection architectures focus heavily on multiscale feature exploitation, but instead of atrous convolution, the FPN is widely used. Two-stage detectors dominate the object detection designs, when it comes to AP performance. Nevertheless, with the EfficientDet family, one-stage detectors improved recently. What remains an advantage of two-stage detectors is their modularity, which makes them easily adaptable for new modules, concepts and to other types of imagery data. Furthermore, the initialisation of the ratio of anchor boxes and the selection of the IoU which defines a true positive should always be considered when designing the architecture to match the dataset at hand. This was pointed out by discussing the anchor initialisation, e.g., of the YOLO architectures and cascading IoU in Cascade R-CNN. Table 4 summarises those cornerstones and provides an overview of the evolution and the onset and use of specific cornerstones in both two- and one-stage detectors.

Table 4. Summary of the evolution of convolutional neural networks (CNNs) for object detection and the above-discussed cornerstones, divided into two- and one-stage detectors. The Backbone is the reported CNN used for feature extraction; RPN shows if a region proposal network is used; RoI describes which kind of pooling is used for the regions of interest; Anchors describes if prior anchor boxes are utilised; NAS describes if neural architecture search was used; Multiscale Feature describes the approach of multiscale feature extraction; and AP describes the performance on the MS-COCO dataset.

Two-Stage Detector Architecture	**Year**	**Backbone**	**RPN**	**RoI**	**Multiscale Feature**	**AP [%]**
R-CNN [73]	2013	AlexNet				-
Fast R-CNN [75]	2015	VGG-16		pooling		19.7
Faster R-CNN [76]	2015	VGG-16	✓	pooling		27.2
Faster R-CNN + FPN [77]	2016	ResNet-101	✓	pooling	FPN	35.8
Mask R-CNN [80]	2017	ResNeXt-101	✓	align	FPN	39.8
Cascade R-CNN [79]	2017	ResNet-101	✓	align	FPN	50.2
PANet [82]	2018	ResNeXt-101	✓	align	FPN	45
TridentNet [85]	2019	ResNet-101-Deformable	✓	pooling	3xAtrous	48.4
Cascade Mask R-RCNN [81]	2019	CBNet (3xResNeXt-152)	✓	align	FPN	53.3
One-Stage Detector Architecture	**Year**	**Backbone**	**Anchors**	**NAS**	**Multiscale Feature**	**AP [%]**
YOLO-V1 [86]	2015	custom				-
SSD [89]	2015	VGG-16	✓		✓	26.8
YOLO-V2 [87]	2016	DarkNet-19	✓			21.6
RetinaNet [90]	2017	ResNet-101	✓		FPN	39.1
YOLO-V3 [88]	2018	DarkNet-53	✓		FPN-like	33
EfficientDet-D7 [93]	2019	EfficientNet-B6	✓	✓	BiFPN	52.2

In a recent example in Earth observation, Ding et al. [138] presented the application of an established Faster R-CNN with a FPN enhanced ResNet-101 backbone for object detection on the DOTA dataset [100] containing 2806 optical, aerial images of very high resolution with 15 object classes. During their application they demonstrated how to adapt the region proposal step to the

needs of densely cluttered and arbitrary rotated objects in Earth observation data. Their approach takes rotation of objects and their specific height width ratio into account by adding an angle offset for rotation and spatial transformation of the RoI to fit the objects height width ratio. The adjacent RoI align does extract much more object specific features from this rotated RoIs resulting in state of the art performance on the DOTA dataset [138]. This study is one example on how to leverage the modularity of two-stage detectors to adapt the model to the needs of Earth observation data.

Independent of the task, image recognition, image segmentation or object detection, we want to mention that careful feature selection, an efficient training process and the choice, combination and possibly weighting of loss functions might have huge impacts on the model performance. The decision for an appropriate architecture is the first step but in DL more has to be considered to make the model perform well for the problem and data at hand. However, the next steps in training and fine tuning the network are more dependent on the combination of the task and data as well as used hardware. Due to this, we do not discuss them in this introduction to the cornerstones of DL architectures but want to highlight their importance and refer to open question number 9 "How to train and optimise the DL system?" (p. 32) by Ball et al. [8] for more details.

4. Popular Deep Learning Frameworks and Earth Observation Datasets

4.1. Deep Learning Frameworks

DL specific frame works are important tools for investigating and using the architectures introduced in Section 3. Among others, such frameworks should provide cornerstones to create architectures, input data pipelines to feed labelled data efficiently during the training process and training schemes themselves to fit a model to the training data, to name some important issues. Over the years, when architectures were proposed and optimised, DL frameworks were also introduced. They opened the method for a wider community with increasingly easier applications of DL. Here, the evolution of the two most popular DL frameworks TensorFlow [12] and Pytorch [13] and their closest relatives is outlined.

In 2007, Theano developed as Python package for ML eventually became a DL framework. One strength of Theano was the calculation of gradients by automatic differentiation of designed networks as well as CPU and GPU processing inspiring today's leading DL frameworks [139,140]. Its officially supported development has ended with version 1.0 in November 2017 [141]. For an easier access to Theano and later TensorFlow and other frameworks, Keras was introduced in May 2015 [142]. It became popular mainly because of its user friendly handling for a wider research community interested in applying DL. The main author is associated with Google, hence Keras is closely related to the TensorFlow library published later in November 2015, developed by GoogleBrain [12]. TensorFlow's handling was criticised for time intensive prototyping mainly due to its session initialisation. However, in September 2019, this major drawback was fixed by introducing eager mode in TensorFlow2 which also includes Keras as high level API. With this union, Keras provides easy access to TensorFlow in a native way but at the same time it is now possible to have full access to core TensorFlow functionality if necessary without leaving the framework [12,143,144].

The evolution of Pytorch, which is supported by Facebook AI Research (FAIR), starts with Caffe, which was originally developed by Jia et al. [145] in 2013. Caffe is a DL framework specialising in CNNs and image processing. In April 2017, Facebook released Caffe2 as an open source derivative and finally merged it with Pytorch one year later in May 2018 [146]. Pytorch [13] is mainly based on the Torch library and as the name suggests developed for Python, same as for Keras and TensorFlow. Both frameworks, Pytorch and TensorFlow, are open source and today's most popular and widely used in research and practice [140].

From an Earth observation perspective, data, features and labels are commonly processed and visualised in a geo information system (GIS). Commercial tools such as ArcGIS as well as open source solutions such as QGIS provide accessibility to the above mentioned frameworks and therewith

support the use of DL in a spatial data environment. For ArcGIS, the tools are implemented in the Image Analyst extension [147] supporting TensorFlow, Keras, Pytorch and CNTK. For QGIS the established Orfeo ToolBox for ML [148] supports DL via the remote OTBTF (Orfeo ToolBox meets TensorFlow) module [149], which uses TensorFlow as backend. Another open source project which leverages QGIS is rastervision [150] which supports TensorFlow, Keras and Pytorch.

4.2. Earth Observation Datasets

Given the tools as architectures and frameworks, the application on Earth observation data is the next step to bridge the gap between computer vision and applied Earth observation. Since overhead imaging data differ from so-called natural images, it is questionable if the architectures developed for the datasets presented above also perform well on Earth observation data. The main differences between Earth observation data and natural images as they were used in the datasets mentioned are:

- The position of the sensor in Earth observation data has mostly an overhead perspective relative to the scene, whereas a natural image is captured from a side looking perspective, hence the same object classes appear differently.
- Data intensively used in computer vision are often three channel RGB images, whereas Earth observation data often consist of a multichannel image stack with more than three channels, which has to be considered, especially when transferring models from computer vision to Earth observation applications.
- Computer vision model input data are often from the same sensor and platform, whereas in Earth observation both can change and data fusion has to be incorporated into the model.
- Objects which appear in overhead images do not have a general orientation. That means that objects of the same class commonly appear at 360° rotation, which has to be considered in training data, architecture or both. Whereas in natural images bottom and top of the image and therewith also of the pictured objects are often defined more specifically which results in a general orientation of objects which can be expected for natural images.
- In natural images, objects of interest tend to be in the centre of the image and in high resolution, whereas in Earth observation data the objects can lie off nadir or at boarders with coarse resolution.
- In Earth observation data, objects or classes tend to be more densely packed and heterogeneous than in natural images [6,134,151].

Due to these properties of Earth observation data, the tasks of image recognition, image segmentation and object detection can be considered more challenging. To cope with this problem, DL models which were fitted to computer vision datasets are fine-tuned on Earth observation datasets. Those Earth observation datasets are often smaller and therefore it is common practice to refine models which have already learned how to hierarchically extract features from imagery data on larger computer vision datasets. This refinement of models, originally trained on other datasets, is known as transfer learning. Already optimised parameters for a computer vision task are refined to an Earth observation task which can be seen as transferring learned skills to apply them in a different context [152–154]. However, even for transfer learning, special Earth observation datasets have to be created. In Table 5, popular Earth observation datasets are presented and also well performing architectures, best practice examples or the baseline models of the datasets are associated in the last column.

Table 5. Summary of DL datasets, with abbreviations for longer names, grouped by their tasks image recognition (IR), image segmentation (IS) and object detection (OD). In the column Topic, the abbreviation LULC means Land Use Land Cover classification. All datasets are freely available, with the need to contact the author for a few of them. The last column shows example applications by briefly describing the architectures.

Dataset	Task	Topic	Platform	Sensor	Resolution	Example Application by Architectures
NWPU RESISC45 [155]	IR	LULC	multiple platforms	optical	high	VGG-16 [155]
EuroSAT [156]	IR	LULC	Sentinel 2	multispectral	medium	Inception-V1 and ResNet-50 [156]
BigEarthNet [116,117]	IR	LULC	Sentinel 2	multispectral	medium	ResNet-50 [117]
So2Sat LCZ42 [157]	IR	local climate zones	Sentinel 1+2	mltspectr+SAR	medium	ResNeXt-29 + CBAM [157]
SpaceNet1 [158]	IS	building footprints	-	multispectral	low	VGG-16 + MNC [158,159]
SpaceNet2 [160]	IS	building footprints	WorldView3	multispectral	high	U-Net (modified: inputdepth = 13) [160]
SpaceNet3 [161]	IS	road network	WorldView3	multispectral	high	ResNet-34 + U-Net [161]
SpaceNet4 [162]	IS	building footprints	WorldView2	multispectral	high	SE-ResNeXt-50/101 + U-Net [162]
SpaceNet5 [163]	IS	road network	WorldView3	multispectral	high	ResNet-50 + U-Net [164], SE-ResNeXt-50 + U-Net [163]
SpaceNet6 [165,166]	IS	building footprints	WordView2 + Capella36	mltspectr + SAR	high	VGG-16 + U-Net [166]
ISPRS 2D Sem. Lab. [126]	IS	multiple classes	plane	multispectral	very high	U-Net, DeepLabV3+, PSPNet, LANet (patch attention module) [167], MobileNetV2(with atrous conv) + Dual path encoder + SE modules [168]
DeepGlobe-Road [169]	IS	road network	WorldView3	multispectral	high	D-LinkNet (ResNet-34 + U-Net with atrous decoder) [170], ResNet-34 + U-Net [171]
DeepGlobe-Building [169]	IS	building footprints	WorldView3	multispectral	high	ResNet-18 + Multitask U-Net [172], WideResNet-38 + U-Net [173]
DeepGlobe-LCC [169]	IS	LULC	WorldView3	multispectral	high	Dense Fusion Classmate Network (DenseNet + FCN varaint) [174], Deep Aggregation Net (ResNet + DeepLabV3 + variant) [175]
WHU Building [176]	IS	building footprints	multiple platforms	optical	high	VGG-16 + ASPP + FCN [177]
INRIA [178]	IS	building footprints	multiple platforms	multispectral	very high	ResNet-50 + SegNet variant [179], U-Net variant [180]
DLR-SkyScapes [181]	IS	multiple classes	helicopter	optical	very high	SkyScapesNet (custom design [181])
NWPU VHR-10 [182]	OD	multiple classes	airborne platforms	optical	very high	DarkNet + YOLO (modified: VaryBlock) [183], ResNet-101 + FPN (modified: Densely connected top-down path) + fully convolutional detector head [184]
COWC [185]	OD	vehicle detection	airborne platforms	optical	very high	VGG16 + SSD + correlation alignment domain adaptation [186]
CARPK [187]	OD	vehicle detection	drone	optical	very high	VGG16 + LPN (Layout Proposal Net) [187]
DLR 3K Munich [188]	OD	vehicle detection	airborne platform	optical	very high	ShuffleDet (ShuffleNet + modified SSD) [189]
DOTA [100]	OD	multiple classes	airborne platforms	optical	very high to high	ResNet-50+improved Cascade R2CNN see leader board of [100], ResNet-101/FPN + Fater R-CNN OBB + RoI transformer [138]
DIOR [24]	OD	multiple classes	multiple platforms	optical	heigh to medium	ResNet-101 + PAnet and ResNet-101 + RetinaNet [24]

When looking at the tasks in combination with platforms, sensors and resolution, a pattern can be observed that spaceborne platforms with multispectral sensors which provide lower resolution are used for image recognition of whole image chips, so-called scene labelling. On the other hand, for image segmentation and object detection sensors with a higher resolution are used. Examples are spaceborne systems, mainly WorldView, and airborne platforms with sensors of very high spatial resolution. This underlines the necessity of richer feature information for image segmentation and object detection tasks than for image recognition. Since sensors with higher resolution provide this information depth, optical and multispectral sensors are dominating the selection for image segmentation and object detection tasks presented in the overview.

To cope with the special properties of Earth observation data, the best performing architectures are mostly modified versions of the architectures discussed in Section 3 or completely new designs made especially for the requirements of Earth observation data. For the selected studies here, the *ResNet Family* is the most widely used backbone architecture, within which the layer depth is rather shallow between 18 and 101 compared to the 152 layers commonly chosen in the computer vision related designs.

In image segmentation, encoder–decoder models dominate, often due to their complex but modular design which can be adapted to the properties of Earth observation data. The same is true for object detectors. More complex two-stage detector models are used more often than one-stage detectors. However, what is more obvious when reviewing the evolution of Earth observation datasets and models for object detection is their ability to deal with rotated bounding boxes. This becomes necessary due to the rotational invariance of objects in Earth observation data, which when not considered can easily lead to a large amount of false negatives when objects are densely accumulated [100,138].

5. Future Research

The advances in DL models for image recognition, image segmentation and object detection since 2012 brought a better general understanding of hierarchical feature extraction from imaging data. The major findings are: the use of feature pyramid networks, atrous convolutions, image context exploitation and using features from different stages in the network. Overall, techniques focus on extraction and combination of multiscale features. With this increasing understanding, features can be used more effectively and models became better in general. However, with the recent emergence of NAS-induced network designs, architectures are now able to reach better or equal performance by using fewer parameters. This is partly due to the fact that they are now specifically trained to handle a specific task and even more important, on a specific type of dataset. Since NAS uses the reward signal of how designs perform on tasks and datasets to optimise the architecture, it can be argued that those architectures are no longer that easily transferable to other domains. Reflecting this development from an Earth observation perspective, the new advances in leading architectures which come from the computer vision domain might not directly match the properties of Earth observation data due to their now more specific design for an underlying computer vision dataset. To take advantage of NAS, it should be used for optimising Earth observation specific architectures or modules, too. Since NAS is highly computationally expensive, it remains questionable how fast and widely this technique will enter the Earth observation community compared to the established hand crafted designs.

Beside architectures, datasets are highly important for pushing DL itself and also in specific domains. The great effort to create a DL dataset, and therefore the lack of datasets for specific tasks, remains a major concern in Earth observation research. Most importantly, the diversity of sensors, labelled classes and topics, as well as the size of datasets should be increased. With increasing datasets and dataset size insights about the properties of Earth observation data with respect to DL models will be made. Using these insights, DL models can be better optimised for the properties of the data of interest [8] instead of solely relying on the findings made in computer vision, which uses different kinds of data. The example of U-Net [67], proposed in medical imagery analysis shows the impact that customised architectures can have when designed for specific data. This was also presented recently in

Earth observation by Azimi et al. [181] introducing the SkyScapes dataset and the custom designed SkyScapesNet which combines cornerstones of CNN architectures to create a model that matches the specific properties of Earth observation data. On the other hand, focus should also be lain on training and designing DL models with small datasets [8], for example with the established use of transfer learning and data augmentation, as well as weakly supervised learning [57,190–192].

The observation of shallower *ResNet Family* backbone networks in well performing deep learning models for Earth observation compared to computer vision tasks, points to the question of the optimal depth of network designs balancing accuracy and overfitting. Simply using deeper models to gain higher scores will eventually lead to overfitted models [6]. Hence, the relative shallow depth of the well performing architectures presented in Table 5 encourages to have a closer look at optimal network depth. By building architectures with as few parameters as possible, more generalised models are produced. That would contribute to better transferability of models, which is another open issue for DL.

Because of the trends in DL architectures and growing dataset diversity, we argue that a thorough knowledge of the cornerstones of DL is important to assess the vast amount of models in order to find the designs that match restrictions in data, hardware and time by balancing performance trade-offs [131]. Furthermore, a well-founded understanding of DL as Earth observation researcher is highly necessary when it comes to incorporating knowledge of physical and ecological relationships into DL models [7,9]—an ability that will make the difference between purely data driven Earth observation with a tendency to remain a black box and more understandable models, combining data science and geoscientific expert knowledge.

6. Conclusions

The emergence of deep learning (DL) led to the adaptation of models developed in computer vision for applications in Earth observation and provided novel possibilities to analyse remotely-sensed data. Nevertheless, the entry barriers remain high for Earth observation scientists who want to use DL models. To lower them, this review provided a fundamental introduction to the most popular DL model for image processing: the convolutional neural networks (CNNs). By discussing the evolution of CNNs in computer vision, we assigned main characteristics to specific tasks. For image recognition, they are:

- A so-called convolutional backbone extracts features from input data in a hierarchical manner by stacked convolutional operations. The repeated convolutions with non-linear activation increase the semantic meaning of features while going deeper into the model [2,3].
- A stack of fully connected artificial neurons uses the extracted features to predict the probability of the class.
- Such deep models need specific normalisation schemes such as batch normalisation to make supervised training of deep networks possible and faster [41].
- Residual connections further alleviates the training of increasingly deep architectures [43].
- To emphasise more complex networks, design elements such as bottleneck layers reduce intermediate feature depth [40] and factorisation of convolutional operations reduce the number of parameters [42].
- The recent findings in neural architecture search (NAS) bring together complex network structures and efficient usage of parameters. With NAS, architectures are searched for by an artificial controller. This controller tries to maximise specific metrics of the networks it creates iteratively and therewith finds highly efficient architectures [48,52].

Improvements in image segmentation and object detection mainly focus on the so-called heads of the architectures, which use the backbones developed in image recognition. The output of image segmentation, a segmentation mask where each pixel is assigned to one class, has the same resolution as the input image. Since the feature extraction first downscales the input image, image segmentation models have to recover input resolution. Those opposed operations lead to a high resolution-feature

depth trade-off. The following properties of CNNs for image segmentation were found to effectively handle this problem:

- Encoder–decoder models, which first encode information by extracting features and use the downscaled features maps from different stages of the encoder to recover input resolution in the decoder, are the most popular designs, namely U-Net [67] and DeepLabV3+ [60].
- So-called atrous convolutions [58], which maintain resolution while extracting features, are widely used to cope with the high resolution-feature depth trade-off.
- The combination of feature maps of different scales with context information from image level was found to contribute effectively to pixel wise classification [59].

In object detection, an additional bounding box has to be provided which presents precise localisation information. Thereby, objects of different size and density should be equally well detected. This leads to problems which relate to multiscale feature extraction. The most effective modules invented in respect to object detection are:

- Two stage object detectors show both good performance and adaptability. The most popular detectors are the Faster R-CNN [76] design and its successors. In the first stage, they propose class agnostic regions of interest (RoIs) for objects. During the second stage those RoIs are classified and the bounding box is regressed to tight object boundaries.
- For multiscale processing, the feature pyramid network (FPN) [77] enhances the convolutional backbone by merging high semantic features with precise localisation information.
- Cascading classifiers and bounding box regression suppress noisy detections by iteratively refining RoIs [79].

Atrous convolution, image context aware designs and FPN show how important multiscale feature exploitation is for image segmentation and object detection. However, these findings can be applied regardless of the task for which they were originally developed. This leads to fruitful development but is also the reason for the high entry barriers, since interrelations of different approaches have to do be considered. To narrow the gap between theoretical concepts and the application of DL to Earth observation research, an overview of Earth observation related datasets is given and recent trends are discussed from an Earth observation perspective. The major findings are:

- Building models with NAS might lead to overly optimised architectures for specific tasks and datasets. Therefore, it is questionable if such models which are recently successful in computer vision tasks perform equally well in Earth observation, as was the case with hand crafted designs. However, NAS can also be used to find Earth observation specific designs.
- The number of DL datasets for Earth observation applications is still small in relation to possible applications and sensor diversity. Since datasets are highly important to push the understanding of the interaction between DL models and specific types of data, an increase in datasets has huge potential for further advances for DL in Earth observation.
- Beside more datasets, weakly supervised learning provides encouraging results as an alternative to expensive dataset creation. It is especially important for proof of concepts studies and experimental research.

While reflecting on the differences and commonalities of data and used architectures in computer vision and Earth observation, we state that the advances in computer vision have to be adapted to match Earth observation applications. Therefore, a thorough understanding of DL concepts is crucial for assessing and adapting models. With the provided extensive introduction to CNNs, we created a foundation to closely review the application of DL in the field of Earth observation research. In Part II of this survey, we will use this basis to further discuss the recent trends of CNNs doing image segmentation and object detection by reviewing published findings in leading Earth observation journals.

Author Contributions: Conceptualisation, T.H. and C.K.; writing—original draft preparation, T.H.; writing—review and editing, T.H. and C.K.; visualisation, T.H.; and supervision, C.K. All authors have read and agree to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: We would such as to thank David Marshall for final proofreading.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bengio, Y. Deep Learning of Representations: Looking Forward. In *Statistical Language and Speech Processing*; Dediu, A.H., Martin-Vide, C., Mitkov, R., Truthe, B., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 1–37.
2. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
3. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*; Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2012; Volume 25, pp. 1097–1105.
4. Voulodimos, A.; Doulamis, N.; Doulamis, A.; Protopapadakis, E. Deep learning for computer vision: A brief review. *Comput. Intell. Neurosci.* **2018**, *2018*, 7068349. [CrossRef]
5. Shrestha, A.; Mahmood, A. Review of Deep Learning Algorithms and Architectures. *IEEE Access* **2019**, *7*, 53040–53065. [CrossRef]
6. Zhang, L.; Zhang, L.; Du, B. Deep Learning for Remote Sensing Data: A Technical Tutorial on the State of the Art. *IEEE Geosci. Remote Sens. Mag.* **2016**, *4*, 22–40. [CrossRef]
7. Zhu, X.X.; Tuia, D.; Mou, L.; Xia, G.; Zhang, L.; Xu, F.; Fraundorfer, F. Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources. *IEEE Geosci. Remote Sens. Mag.* **2017**, *5*, 8–36. [CrossRef]
8. Ball, J.E.; Anderson, D.T.; Chan, C.S., Sr. Comprehensive survey of deep learning in remote sensing: Theories, tools, and challenges for the community. *J. Appl. Remote Sens.* **2017**, *11*, 1–54. [CrossRef]
9. Reichstein, M.; Camps-Valls, G.; Stevens, B.; Jung, M.; Denzler, J.; Carvalhais, N.; Prabhat. Deep learning and process understanding for data-driven Earth system science. *Nature* **2019**, *566*, 195–204. [CrossRef]
10. Google Scholar. Top Publication. Available online: https://scholar.google.com/citations?view_op=top_venues&hl=en (accessed on 1 April 2020).
11. Acemap. NeurIPS Affiliation Statistics. Available online: https://archive.acemap.info/conference-statistics/affiliation-rank?name=NIPS&year=2018&type=affiliation#table-1 (accessed on 1 April 2020).
12. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Available online: https://www.tensorflow.org/ (accessed on 1 April 2020).
13. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*; Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; Volume 32, pp. 8024–8035.
14. Bengio, Y.; Courville, A.; Vincent, P. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1798–1828. [CrossRef]
15. Dahl, G.E.; Ranzato, M.; Mohamed, A.R.; Hinton, G. Phone Recognition with the Mean-Covariance Restricted Boltzmann Machine. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems—Volume 1*; Curran Associates Inc.: Red Hook, NY, USA, 2010; pp. 469–477.
16. Dahl, G.E.; Yu, D.; Deng, L.; Acero, A. Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. *Trans. Audio Speech and Lang. Proc.* **2012**, *20*, 30–42. [CrossRef]
17. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.E.; Mohamed, A.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.N.; et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Process. Mag.* **2012**, *29*, 82–97. [CrossRef]
18. Hinton, G.E.; Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science* **2006**, *313*, 504–507. [CrossRef] [PubMed]

19. Bengio, Y.; Lamblin, P.; Popovici, D.; Larochelle, H. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*; Schölkopf, B., Platt, J., Hoffman, T., Eds.; MIT Press: Cambridge, MA, USA, 2007; Volume 19, pp. 153–160.
20. Ciresan, D.; Meier, U.; Schmidhuber, J. Multi-column deep neural networks for image classification. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, 16–21 June 2012; pp. 3642–3649.
21. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vision* **2015**, *115*, 211–252. [CrossRef]
22. Tsagkatakis, G.; Aidini, A.; Fotiadou, K.; Giannopoulos, M.; Pentari, A.; Tsakalides, P. Survey of Deep-Learning Approaches for Remote Sensing Observation Enhancement. *Sensors* **2019**, *19*, 3929. [CrossRef]
23. Ma, L.; Liu, Y.; Zhang, X.; Ye, Y.; Yin, G.; Johnson, B.A. Deep learning in remote sensing applications: A meta-analysis and review. *ISPRS J. Photogramm. Remote Sens.* **2019**, *152*, 166–177. [CrossRef]
24. Li, K.; Wan, G.; Cheng, G.; Meng, L.; Han, J. Object detection in optical remote sensing images: A survey and a new benchmark. *ISPRS J. Photogramm. Remote Sens.* **2020**, *159*, 296–307. [CrossRef]
25. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef]
26. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
27. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
28. Cho, K.; Raiko, T.; Ilin, A. Enhanced Gradient for Training Restricted Boltzmann Machines. *Neural Comput.* **2013**, *25*, 805–831. [CrossRef]
29. Cho, K. Foundations and Advances in Deep Learning. Ph.D. Thesis, Aalto University, Espoo, Finland, 2014.
30. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Comput.* **1989**, *1*, 541–551. [CrossRef]
31. Nair, V.; Hinton, G.E. Rectified Linear Units Improve Restricted Boltzmann Machines. In *ICML*; Fürnkranz, J., Joachims, T., Eds.; Omnipress: Madison, WI, USA, 2010; pp. 807–814.
32. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 249–256.
33. Glorot, X.; Bordes, A.; Bengio, Y. Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 11–13 April 2011; pp. 315–323.
34. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*; Rumelhart, D.E., Mcclelland, J.L., Eds.; MIT Press: Cambridge, MA, USA, 1986; pp. 318–362.
35. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
36. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2014; Volume 27, pp. 2672–2680.
37. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
38. Zeiler, M.D.; Fergus, R. Visualizing and Understanding Convolutional Networks. In *Computer Vision–ECCV 2014*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 818–833.
39. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
40. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1–9. [CrossRef]

41. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proc. Mach. Learn. Res.* **2015**, *37*, 448–456.
42. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826. [CrossRef]
43. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
44. Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; He, K. Aggregated Residual Transformations for Deep Neural Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 5987–5995. [CrossRef]
45. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1800–1807. [CrossRef]
46. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2261–2269.
47. Hu, J.; Shen, L.; Sun, G. Squeeze-and-Excitation Networks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7132–7141. [CrossRef]
48. Zoph, B.; Le, Q.V. Neural Architecture Search with Reinforcement Learning. *arXiv* **2016**, arXiv:1611.01578.
49. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning Transferable Architectures for Scalable Image Recognition. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8697–8710. [CrossRef]
50. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
51. Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Le, Q.V. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 2815–2823.
52. Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *Proc. Mach. Learn. Res.* **2019**, *97*, 6105–6114.
53. Everingham, M.; Van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vision* **2010**, *88*, 303–338. [CrossRef]
54. Everingham, M.; Eslami, S.M.; Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes Challenge: A Retrospective. *Int. J. Comput. Vision* **2015**, *111*, 98–136. [CrossRef]
55. Long, J.; Shelhamer, E.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *39*, 640–651.
56. Chen, L.C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. *arXiv* **2014**, arXiv:1412.7062.
57. Papandreou, G.; Chen, L.C.; Murphy, K.; Yuille, A.L. Weakly- and Semi-Supervised Learning of a DCNN for Semantic Image Segmentation. *arXiv* **2015**, arXiv:1502.02734.
58. Chen, L.C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *40*, 834–848. [CrossRef]
59. Chen, L.C.; Papandreou, G.; Schroff, F.; Adam, H. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv* **2017**, arXiv:1706.05587.
60. Chen, L.C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *Computer Vision–ECCV 2018*; Ferrari, V., Hebert, M., Sminchisescu, C.; Weiss, Y., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 833–851.
61. Krähenbühl, P.; Koltun, V. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. In *Advances in Neural Information Processing Systems*; Shawe-Taylor, J., Zemel, R.S., Bartlett, P.L., Pereira, F., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2011; Volume 24, pp. 109–117.

62. Chen, L.C.; Collins, M.; Zhu, Y.; Papandreou, G.; Zoph, B.; Schroff, F.; Adam, H.; Shlens, J. Searching for Efficient Multi-Scale Architectures for Dense Image Prediction. In *Advances in Neural Information Processing Systems*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2018; Volume 31, pp. 8699–8710.
63. Liu, C.; Chen, L.; Schroff, F.; Adam, H.; Hua, W.; Yuille, A.L.; Fei-Fei, L. Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 82–92.
64. Noh, H.; Hong, S.; Han, B. Learning Deconvolution Network for Semantic Segmentation. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1520–1528. [CrossRef]
65. Liu, W.; Rabinovich, A.; Berg, A.C. ParseNet: Looking Wider to See Better. *arXiv* **2015**, arXiv:1506.04579.
66. Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid Scene Parsing Network. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6230–6239.
67. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*; Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 234–241.
68. Jégou, S.; Drozdzal, M.; Vázquez, D.; Romero, A.; Bengio, Y. The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1175–1183.
69. Lin, G.; Milan, A.; Shen, C.; Reid, I.D. RefineNet: Multi-path Refinement Networks for High-Resolution Semantic Segmentation. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 5168–5177.
70. Sun, K.; Xiao, B.; Liu, D.; Wang, J. Deep High-Resolution Representation Learning for Human Pose Estimation. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 5686–5696.
71. Sun, K.; Zhao, Y.; Jiang, B.; Cheng, T.; Xiao, B.; Liu, D.; Mu, Y.; Wang, X.; Liu, W.; Wang, J. High-Resolution Representations for Labeling Pixels and Regions. *arXiv* **2019**, arXiv:1904.04514.
72. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In *Computer Vision–ECCV 2014*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 740–755.
73. Girshick, R.B.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
74. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *37*, 1904–1916. [CrossRef]
75. Girshick, R.B. Fast R-CNN. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
76. Ren, S.; He, K.; Girshick, R.B.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *39*, 1137–1149. [CrossRef]
77. Lin, T.Y.; Dollár, P.; Girshick, R.B.; He, K.; Hariharan, B.; Belongie, S.J. Feature Pyramid Networks for Object Detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 936–944.
78. Minaee, S.; Boykov, Y.; Porikli, F.; Plaza, A.; Kehtarnavaz, N.; Terzopoulos, D. Image Segmentation Using Deep Learning: A Survey. *arXiv* **2020**, arXiv:2001.05566.
79. Cai, Z.; Vasconcelos, N. Cascade R-CNN: Delving Into High Quality Object Detection. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6154–6162.
80. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R.B. Mask R-CNN. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venicnisee, Italy, 22–29 October 2017; pp. 2980–2988.
81. Liu, Y.; Wang, Y.; Wang, S.; Liang, T.; Zhao, Q.; Tang, Z.; Ling, H. CBNet: A Novel Composite Backbone Network Architecture for Object Detection. *arXiv* **2019**, arXiv:1909.03625.

82. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path Aggregation Network for Instance Segmentation. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8759–8768.
83. Singh, B.; Davis, L.S. An Analysis of Scale Invariance in Object Detection—SNIP. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 3578–3587.
84. Singh, B.; Najibi, M.; Davis, L.S. SNIPER: Efficient multi-scale training. In *Advances in Neural Information Processing Systems*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2018; Volume 31, pp. 9310–9320.
85. Li, Y.; Chen, Y.; Wang, N.; Zhang, Z. Scale-Aware Trident Networks for Object Detection. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 6053–6062.
86. Redmon, J.; Divvala, S.K.; Girshick, R.B.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
87. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525.
88. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
89. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.E.; Fu, C.Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Computer Vision – ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Switzerland, 2016; pp. 21–37.
90. Lin, T.Y.; Goyal, P.; Girshick, R.B.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2999–3007.
91. Zhang, S.; Wen, L.; Bian, X.; Lei, Z.; Li, S.Z. Single-Shot Refinement Neural Network for Object Detection. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4203–4212.
92. Ghiasi, G.; Lin, T.; Le, Q.V. NAS-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 7029–7038.
93. Tan, M.; Pang, R.; Le, Q.V. EfficientDet: Scalable and Efficient Object Detection. *arXiv* **2019**, arXiv:1911.09070.
94. LeCun, Y.; Boser, B.; Denker, J.S.; Howard, R.E.; Habbard, W.; Jackel, L.D.; Henderson, D. Handwritten Digit Recognition with a Back-Propagation Network. In *Advances in Neural Information Processing Systems*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1990; Volume 2, pp. 396–404.
95. Ranzato, M.; Huang, F.J.; Boureau, Y.; LeCun, Y. Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition. In Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, 17–22 June 2007; pp. 1–8.
96. Cadieu, C.F.; Hong, H.; Yamins, D.L.K.; Pinto, N.; Ardila, D.; Solomon, E.A.; Majaj, N.J.; DiCarlo, J.J. Deep Neural Networks Rival the Representation of Primate IT Cortex for Core Visual Object Recognition. *PLOS Comput. Biol.* **2014**, *10*, 1–18. [CrossRef]
97. Hubel, D.H.; Wiesel, T.N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J. Physiol.* **1962**, *160*, 106–154. [CrossRef]
98. Fukushima, K.; Miyake, S. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognit.* **1982**, *15*, 455–469. [CrossRef]
99. Felleman, D.J.; Van Essen, D.C. Distributed Hierarchical Processing in the Primate Cerebral Cortex. *Cerebral Cortex* **1991**, *1*, 1–47. [CrossRef]
100. Xia, G.S.; Bai, X.; Ding, J.; Zhu, Z.; Belongie, S.; Luo, J.; Datcu, M.; Pelillo, M.; Zhang, L. DOTA: A Large-Scale Dataset for Object Detection in Aerial Images. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 3974–3983.
101. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1026–1034.

102. LeCun, Y.; Bottou, L.; Orr, G.; Müller, K. Efficient BackProp. In *Neural Networks: Tricks of the Trade*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1998; Chapter 2, p. 546. [CrossRef]
103. Sutskever, I.; Martens, J.; Dahl, G.; Hinton, G. On the Importance of Initialization and Momentum in Deep Learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning—Volume 28*; Microtome Publishing: Brookline, MA, USA, 2013; pp. III-1139–III-1147.
104. Saxe, A.M.; McClelland, J.L.; Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv* **2013**, arXiv:1312.6120.
105. Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* **1994**, *5*, 157–166. [CrossRef]
106. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6848–6856.
107. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A.A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; pp. 4278–4284.
108. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520. [CrossRef]
109. Sandler, M. MobileNet V2 ImageNet Checkpoints. Available online: https://github.com/tensorflow/models/blob/master/research/slim/nets/mobilenet/README.md (accessed on 1 April 2020).
110. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
111. Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. Available online: https://www.cs.toronto.edu/kriz/learning-features-2009-TR.pdf (accessed on 1 April 2020).
112. Liu, C.; Zoph, B.; Neumann, M.; Shlens, J.; Hua, W.; Li, L.J.; Fei-Fei, L.; Yuille, A.; Huang, J.; Murphy, K. Progressive Neural Architecture Search. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 18–14 September 2018; pp. 19–35.
113. Real, E.; Aggarwal, A.; Huang, Y.; Le, Q.V. Regularized evolution for image classifier architecture search. In Proceedings of the AAAI Conference on Artificial Intelligence, Honululu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 4780–4789.
114. Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for MobileNetV3. *arXiv* **2019**, arXiv:1905.02244.
115. Zhang, X.; Li, Z.; Loy, C.C.; Lin, D. PolyNet: A Pursuit of Structural Diversity in Very Deep Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 3900–3908. [CrossRef]
116. Sumbul, G.; Charfuelan, M.; Demir, B.; Markl, V. BigEarthNet: A Large-Scale Benchmark Archive For Remote Sensing Image Understanding. In Proceedings of the IEEE International Geoscience and Remote Sensing Symposium, Yokohama, Japan, 28 July–2 August 2019; pp. 5901–5904, doi:10.1109/IGARSS.2019.8900532. [CrossRef]
117. Sumbul, G.; Kang, J.; Kreuziger, T.; Marcelino, F.; Costa, H.; Benevides, P.; Caetano, M.; Demir, B. BigEarthNet Dataset with A New Class-Nomenclature for Remote Sensing Image Understanding. *arXiv* **2020**, arXiv:2001.06372.
118. Cordts, M.; Omran, M.; Ramos, S.; Scharwächter, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; Schiele, B. The Cityscapes Dataset. In Proceedings of the CVPR Workshop on the Future of Datasets in Vision, Boston, MA, USA, 7–12 June 2015; Volume 2.
119. Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; Schiele, B. The Cityscapes Dataset for Semantic Urban Scene Understanding. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 3213–3223.
120. Garcia-Garcia, A.; Orts-Escolano, S.; Oprea, S.; Villena-Martinez, V.; Martinez-Gonzalez, P.; Garcia-Rodriguez, J. A survey on deep learning techniques for image and video semantic segmentation. *Appl. Soft Comput.* **2018**, *70*, 41–65. [CrossRef]

121. Holschneider, M.; Kronland-Martinet, R.; Morlet, J.; Tchamitchian, P. A Real-Time Algorithm for Signal Analysis with the Help of the Wavelet Transform. In *Wavelets*; Combes, J.M., Grossmann, A., Tchamitchian, P., Eds.; Springer: Berlin/Heidelberg, Germany, 1990; pp. 286–297.
122. Badrinarayanan, V.; Kendall, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [CrossRef]
123. Brostow, G.J.; Fauqueur, J.; Cipolla, R. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognit. Lett.* **2009**, *30*, 88–97. [CrossRef]
124. Zhang, Z.; Zhang, X.; Peng, C.; Xue, X.; Sun, J. Exfuse: Enhancing feature fusion for semantic segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 18–14 September 2018; pp. 269–284.
125. Zhou, Z.; Rahman Siddiquee, M.; Tajbakhsh, N.; Liang, J. Unet++: A nested u-net architecture for medical image segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support—4th International Workshop, DLMIA 2018 and 8th International Workshop, ML-CDS 2018 Held in Conjunction with MICCAI 2018*; Maier-Hein, L., Syeda-Mahmood, T., Taylor, Z., Lu, Z., Stoyanov, D., Madabhushi, A., Tavares, J., Nascimento, J., Moradi, M., Martel, A., et al., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2018; pp. 3–11. [CrossRef]
126. ISPRS. 2D Semantic Labeling Challenge. Available online: http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html (accessed on 1 April 2020).
127. Wang, Y.; Liang, B.; Ding, M.; Li, J. Dense Semantic Labeling with Atrous Spatial Pyramid Pooling and Decoder for High-Resolution Remote Sensing Imagery. *Remote Sens.* **2019**, *11*, 20. [CrossRef]
128. Common Objects in COntext. Detection Evaluation. Available online: http://cocodataset.org/#detection-eval (accessed on 1 April 2020).
129. Common Objects in COntext. MS-COCO Github Repository: Cocoapi. Available online: https://github.com/cocodataset/cocoapi (accessed on 1 April 2020).
130. Liu, L.; Ouyang, W.; Wang, X.; Fieguth, P.W.; Chen, J.; Liu, X.; Pietikäinen, M. Deep Learning for Generic Object Detection: A Survey. *Int. J. Comput. Vis.* **2018**, *128*, 261–318. [CrossRef]
131. Huang, J.; Rathod, V.; Sun, C.; Zhu, M.; Balan, A.K.; Fathi, A.; Fischer, I.C.; Wojna, Z.; Song, Y.; Guadarrama, S.; et al. Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 3296–3297.
132. Wu, X.; Sahoo, D.; Hoi, S.C. Recent advances in deep learning for object detection. *Neurocomputing* **2020**. [CrossRef]
133. Zhao, Z.; Zheng, P.; Xu, S.; Wu, X. Object Detection With Deep Learning: A Review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [CrossRef]
134. Jiao, L.; Zhang, F.; Liu, F.; Yang, S.; Li, L.; Feng, Z.; Qu, R. A Survey of Deep Learning-Based Object Detection. *IEEE Access* **2019**, *7*, 128837–128868. [CrossRef]
135. Uijlings, J.; van de Sande, K.; Gevers, T.; Smeulders, A. Selective Search for Object Recognition. *Int. J. Comput. Vis.* **2013**, *104*, 154–171. [CrossRef]
136. Girshick, R.; Radosavovic, I.; Gkioxari, G.; Dollár, P.; He, K. Detectron. Available online: https://github.com/facebookresearch/detectron (accessed on 1 April 2020).
137. Cubuk, E.D.; Zoph, B.; Mané, D.; Vasudevan, V.; Le, Q.V. AutoAugment: Learning Augmentation Strategies From Data. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 113–123.
138. Ding, J.; Xue, N.; Long, Y.; Xia, G.; Lu, Q. Learning RoI Transformer for Oriented Object Detection in Aerial Images. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 2844–2853.
139. Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv* **2016**, arXiv:1605.02688.
140. Nguyen, G.; Dlugolinsky, S.; Bobák, M.; Tran, V.; López García, A.; Heredia, I.; Malík, P.; Hluch, L. Machine Learning and Deep Learning Frameworks and Libraries for Large-Scale Data Mining: A Survey. *Artif. Intell. Rev.* **2019**, *52*, 77–124. [CrossRef]
141. Theano Development Team. Theano: News. Available online: http://deeplearning.net/software/theano/ (accessed on 1 April 2020).

142. Chollet, F. Keras. Available online: https://keras.io (accessed on 1 April 2020).
143. TensorFlow Development Team. TensorFlow Github Repository. Available online: https://github.com/tensorflow/tensorflow (accessed on 1 April 2020).
144. TensorFlow Development Team. TensorFlow Keras API. Available online: https://www.tensorflow.org/guide/keras (accessed on 1 April 2020).
145. Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; Darrell, T. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22nd ACM International Conference on Multimedia, Orlando, FL, USA, 3–7 November 2014; pp. 675–678.
146. Caffe2 Development Team. Caffe2: News. Available online: https://caffe2.ai/blog/2018/05/02/Caffe2_PyTorch_1_0.html (accessed on 1 April 2020).
147. ESRI. Image Analyst—Deep Learning in ArcGIS Pro. Available online: https://pro.arcgis.com/de/pro-app/help/analysis/image-analyst/deep-learning-in-arcgis-pro.htm (accessed on 15 May 2020).
148. OTB Development Team. Orfeo ToolBox—Documentation. Available online: https://www.orfeo-toolbox.org/CookBook/ (accessed on 15 May 2020).
149. Cresson, R. A framework for remote sensing images processing using deep learning technique. *arXiv* **2018**, arXiv:1807.06535.
150. azavea. Rastervision Documentation. Available online: https://docs.rastervision.io/en/0.10/index.html (accessed on 15 May 2020).
151. Zhang, C.; Wei, S.; Ji, S.; Lu, M. Detecting Large-Scale Urban Land Cover Changes from Very High Resolution Remote Sensing Images Using CNN-Based Classification. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 189. [CrossRef]
152. Pan, S.J.; Yang, Q. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359. [CrossRef]
153. Bengio, Y. Deep Learning of Representations for Unsupervised and Transfer Learning. In *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop—Volume 27*; Microtome Publishing: Brookline, MA, USA, 2011; pp. 17–37. [CrossRef]
154. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 9. [CrossRef]
155. Cheng, G.; Han, J.; Lu, X. Remote Sensing Image Scene Classification: Benchmark and State of the Art. *Proc. IEEE* **2017**, *105*, 1865–1883. [CrossRef]
156. Helber, P.; Bischke, B.; Dengel, A.; Borth, D. Introducing Eurosat: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification. In Proceedings of the IGARSS 2018—2018 IEEE International Geoscience and Remote Sensing Symposium, Valencia, Spain, 22–27 June 2018; pp. 204–207.
157. Zhu, X.; Hu, J.; Qiu, C.; Shi, Y.; Kang, J.; Mou, L.; Bagheri, H.; Haberle, M.; Hua, Y.; Huang, R.; et al. So2Sat LCZ42: A Benchmark Dataset for Global Local Climate Zones Classification. *arXiv* **2019**, arXiv:1912.12171.
158. SpaceNet. SpaceNet 1: Building Detection v1. Available online: https://github.com/SpaceNetChallenge/BuildingDetectors (accessed on 1 April 2020).
159. Dai, J.; He, K.; Sun, J. Instance-Aware Semantic Segmentation via Multi-task Network Cascades. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 3150–3158.
160. SpaceNet. SpaceNet 2: Building Detection v2. Available online: https://github.com/SpaceNetChallenge/BuildingDetectors_Round2 (accessed on 1 April 2020).
161. SpaceNet. SpaceNet 3: Road Network Detection. Available online: https://github.com/SpaceNetChallenge/RoadDetector (accessed on 1 April 2020).
162. SpaceNet. SpaceNet 4: Off-Nadir Buildings. Available online: https://github.com/SpaceNetChallenge/SpaceNet_Optimized_Routing_Solutions (accessed on 1 April 2020).
163. Etten, A.V. City-Scale Road Extraction from Satellite Imagery v2: Road Speeds and Travel Times. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), Snowmass Village, CO, USA, 1–5 March 2020; pp. 1786–1795.
164. Etten, A.V. City-scale Road Extraction from Satellite Imagery. *arXiv* **2019**, arXiv:1904.09901.
165. SpaceNet. SpaceNet6: Multi Sensor—All Weather. Available online: https://spacenet.ai/sn6-challenge/ (accessed on 1 April 2020).
166. Shermeyer, J.; Hogan, D.; Brown, J.; Etten, A.V.; Weir, N.; Pacifici, F.; Haensch, R.; Bastidas, A.; Soenen, S.; Bacastow, T.; et al. SpaceNet 6: Multi-Sensor All Weather Mapping Dataset. *arXiv* **2020**, arXiv:2004.06500.

167. Ding, L.; Tang, H.; Bruzzone, L. Improving Semantic Segmentation of Aerial Images Using Patch-based Attention. *arXiv* **2019**, arXiv:1911.08877.
168. Zhang, G.; Lei, T.; Cui, Y.; Jiang, P. A Dual-Path and Lightweight Convolutional Neural Network for High-Resolution Aerial Image Segmentation. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 582. [CrossRef]
169. Demir, I.; Koperski, K.; Lindenbaum, D.; Pang, G.; Huang, J.; Basu, S.; Hughes, F.; Tuia, D.; Raskar, R. DeepGlobe 2018: A Challenge to Parse the Earth Through Satellite Images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Salt Lake City, UT, USA, 18–23 June 2018; pp. 172–17209.
170. Zhou, L.; Zhang, C.; Wu, M. D-LinkNet: LinkNet with Pretrained Encoder and Dilated Convolution for High Resolution Satellite Imagery Road Extraction. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 192–1924.
171. Buslaev, A.; Seferbekov, S.S.; Iglovikov, V.; Shvets, A. Fully Convolutional Network for Automatic Road Extraction From Satellite Imagery. In Proceedings of the CVPR Workshops, Salt Lake City, UT, USA, 18–23 June 2018; pp. 207–210.
172. Hamaguchi, R.; Hikosaka, S. Building Detection from Satellite Imagery using Ensemble of Size-Specific Detectors. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 223–2234.
173. Iglovikov, V.; Seferbekov, S.; Buslaev, A.; Shvets, A. TernausNetV2: Fully Convolutional Network for Instance Segmentation. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 228–2284.
174. Tian, C.; Li, C.; Shi, J. Dense Fusion Classmate Network for Land Cover Classification. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 262–2624.
175. Kuo, T.; Tseng, K.; Yan, J.; Liu, Y.; Wang, Y.F. Deep Aggregation Net for Land Cover Classification. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 247–2474.
176. Ji, S.; Wei, S.; Lu, M. Fully Convolutional Networks for Multisource Building Extraction from an Open Aerial and Satellite Imagery Data Set. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 574–586. [CrossRef]
177. Ji, S.; Wei, S.; Lu, M. A scale robust convolutional neural network for automatic building extraction from aerial and satellite imagery. *Int. J. Remote Sens.* **2019**, *40*, 3308–3322. [CrossRef]
178. Maggiori, E.; Tarabalka, Y.; Charpiat, G.; Alliez, P. Can Semantic Labeling Methods Generalize to Any City? The Inria Aerial Image Labeling Benchmark. In Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Fort Worth, TX, USA, 23–28 July 2017; pp. 3226–3229.
179. Audebert, N.; Boulch, A.; Le Saux, B.; Lefèvre, S. Distance transform regression for spatially-aware deep semantic segmentation. *Comput. Vision Image Underst.* **2019**, *189*, 102809. [CrossRef]
180. Duke Applied Machine Learning Lab. DukeAMLL Repository of Winning INRIA Building Labeling. Available online: https://github.com/dukeamll/inria_building_labeling_2017 (accessed on 1 April 2020).
181. Azimi, S.M.; Henry, C.; Sommer, L.; Schumann, A.; Vig, E. SkyScapes Fine-Grained Semantic Understanding of Aerial Scenes. In Proceedings of the IEEE International Conference on Computer Vision, Seoul, Korea, 27–28 October 2019; pp. 7393–7403.
182. Cheng, G.; Han, J.; Zhou, P.; Guo, L. Multi-class geospatial object detection and geographic image classification based on collection of part detectors. *ISPRS J. Photogramm. Remote Sens.* **2014**, *98*, 119–132. [CrossRef]
183. Zhang, H.; Wu, J.; Liu, Y.; Yu, J. VaryBlock: A Novel Approach for Object Detection in Remote Sensed Images. *Sensors* **2019**, *19*, 5284. [CrossRef]
184. Tayara, H.; Chong, K.T. Object Detection in Very High-Resolution Aerial Images Using One-Stage Densely Connected Feature Pyramid Network. *Sensors* **2018**, *18*, 3341. [CrossRef]
185. Mundhenk, T.N.; Konjevod, G.; Sakla, W.A.; Boakye, K. A Large Contextual Dataset for Classification, Detection and Counting of Cars with Deep Learning. In *Computer Vision–ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 785–800.
186. Koga, Y.; Miyazaki, H.; Shibasaki, R. A Method for Vehicle Detection in High-Resolution Satellite Images that Uses a Region-Based Object Detector and Unsupervised Domain Adaptation. *Remote Sens.* **2020**, *12*, 575. [CrossRef]

187. Hsieh, M.; Lin, Y.; Hsu, W.H. Drone-Based Object Counting by Spatially Regularized Regional Proposal Network. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 4165–4173.
188. Liu, K.; Mattyus, G. Fast Multiclass Vehicle Detection on Aerial Images. *IEEE Geosci. Remote Sens. Lett.* **2015**, *12*, 1938–1942.
189. Azimi, S.M. ShuffleDet: Real-Time Vehicle Detection Network in On-Board Embedded UAV Imagery. In *Computer Vision–ECCV 2018 Workshops*; Leal-Taixé, L., Roth, S., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 88–99.
190. Zhou, Z.H. A brief introduction to weakly supervised learning. *Natl. Sci. Rev.* **2017**, *5*, 44–53. [CrossRef]
191. Shi, Z.; Yang, Y.; Hospedales, T.M.; Xiang, T. Weakly-Supervised Image Annotation and Segmentation with Objects and Attributes. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2525–2538. [CrossRef]
192. Diba, A.; Sharma, V.; Pazandeh, A.; Pirsiavash, H.; Van Gool, L. Weakly Supervised Cascaded Convolutional Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 5131–5139.