# En ny æra for uthenting av informasjon fra satellittbilder ved hjelp av maskinlæring

Mathilde Ørstavik og Terje Midtbø

*Mathilde Ørstavik and Terje Midtbø, A New Era for Feature Extraction in Remotely Sensed Images by The Use of Machine Learning*

KART OG PLAN, Vol. 77, pp. 93–107, POB 5003, NO-1432 Ås, ISSN 0047-3278

There has been a revolution within the field of machine learning that has given rise to new and improved methods for visual recognition the last years. The leading technique is the convolutional neural network (CNN), and this paper covers implementation of these networks and their potential. Through looking at previous work, the paper shows that leading methods today are networks based on previously proposed techniques, and they are usually fine-tuned networks. Existing methods give classification accuracy up to 99,47% (Nogueira, Penatti, & dos Santos, 2016) and segmentation accuracy up to 88.5% (Marmanis et al., 2016). Both methods are proposed in papers released this year, which indicates that the methods will keep on improving. The paper also provides an example of a possible problem that could be solved with the existing technology – detection of buildings in satellite images. This could be done by building a CNN which takes a combination of multispectral images and a digital surface model as input.

*Key words:* Machine Learning, Convolutional Neural Networks, Remote Sensing, Satellite images

*Mathilde Ørstavik and Terje Midtbø*, Norwegian University of Science and Technology, NO-7491 Trondheim. E-mail: mathildo@stud.ntnu.no

## 1 Introduction

The field of machine learning has increased in popularity recent years, and its techniques help us solve complex problems. Computers don't have to be explicitly programmed, but can instead change and improve their algorithms, and thereby learn from the given data. This enables us to make use of the enormous amounts of data that is being, and has been, collected over the years.

Artificial intelligence and machine learning have been around since the 1950's. Some years earlier McCulloch & Pitts (1943) presented their paper about a computational model for neural networks. It was not feasible to realize their ideas until computing capacity was adequate in the 1950's. In the 1960's and 1970's methods within neural networks evolved slowly, and there was a campaign to discredit neural networks. However, a few researchers continued the work on problems as pattern recognition (Macukow, 2016). Still, important foundations for later research were established in this period. It is given that the continuously improvement of computing hardware has played a role in the development of neural networks, and in the 1980's research within the field got a new boost. In the 1990's significant advances were made in all areas of artificial intelligence. Scientists began creating programs for computers to analyse large amounts of data and draw conclusions from the results (Marr, 2016; Schmidhuber, 2015).

In 2012, a new revolution within the use of machine learning for visual recognition tasks began. The idea of *deep learning* was introduced trough a new composition of a network called a Convolutional Neural Network (CNN). Contrary to popular belief, CNN was not invented in 2012 but already in the 70's (Nielsen, 2015). However, it wasn't until 2012 that CNN showed its massive capacity within visual recognition. Again, one major factor was the improvement in hardware. Finally, computers were good enough to train a CNN within reasonable time. Another reason was available datasets, which made it possible to properly train the networks (Russakovsky et al., 2015). Ever since,

CNNs have been the leading technique within visual recognition.

Still, there seems to be a general belief that the technology is not good enough, and probably never will be. This is an assumption that might not be correct, and may be contradicted through thorough research of the state-of-the-art techniques.

In recent years there has also been a significant increase in the number of different satellite sensors, which deliver large volumes of very high resolution (VHR) remotely sensed images. This opens for new ways to retrieve and process geographical information. Even though some software exists that supports semi-automated visual recognition (GISGeography, 2016), in practice most images are still classified, labelled, and drawn manually (Marmanis et al., 2016). However, the rapid development within machine learning over the last years have given rise to new research, where neural networks are used in the extraction of information from remotely sensed images. Examples of such research can be found in He et al. (2015), Long et al. (2017) and Castelluccio et al. (2015).

Since methods for machine learning might be unfamiliar for many in the remote sensing community, this paper will give a thorough introduction to fundamental techniques within artificial intelligence and neural networks, before the focus shifts towards the recent development within the field.

The objective of this paper is to:

1. Introduce convolutional neural networks (CNNs).
2. Look at state-of-the-art techniques for visual recognition within machine learning over the past years.
3. Assess how current techniques may be employed to extract geographical information in remote sensing images.

## 2 The Basics of Machine Learning for Visual Recognition

Visual recognition is one of the fastest growing fields of artificial intelligence. Even though the amount of visual data available today is enormous, it is still the most difficult data to harness (F.-F. Li & Karpathy, 2015). We have a hard time grasping the content of an image using machines. Take for example the task of determining if an image is of a cat. There are so many possible images (Figure 1), and a machine must know what the common denominators are for all of them.
Visual recognition is split into different tasks (Figure 2). Among them are:

- **Classification:** Determining which of specified classes an image belongs to. Such classes may, for example, be “Cat”, “Dog” and “Rabbit”.
- C**lassification + localization:** As well as classifying an image, a bounding box describing where in the image the object exists is determined.
- **Object Detection:** What objects exist in the image is determined, including the bounding box for each object.
- **Instance Segmentation:** The shape of the objects in the image is determined by returning all pixels that belong to a specific object.

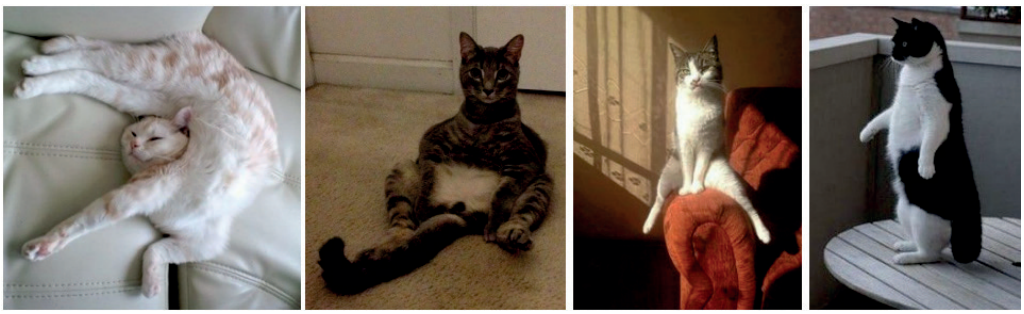

*Figure 1: Examples of cats in different poses, making it difficult to determine the typical shape of a cat. Source: (F.-F. Li & Karpathy, 2015).*

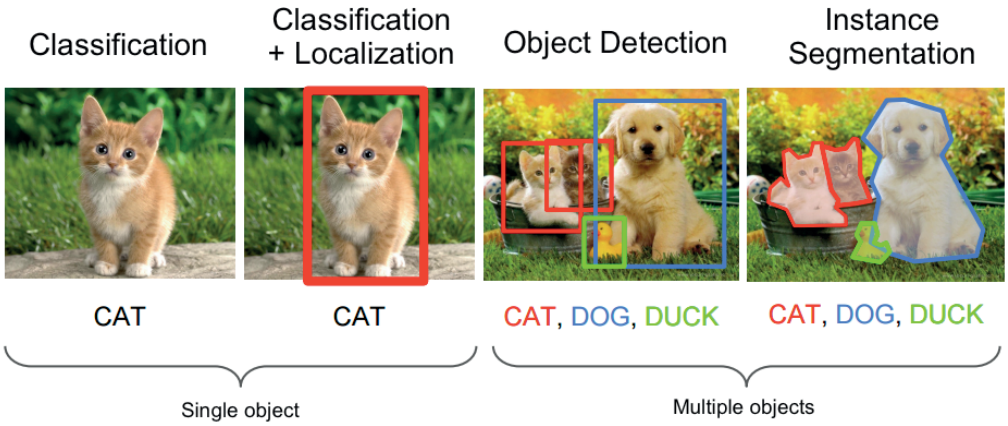


*Figure 2: Examples of visual recognition tasks. Source: (F.-F. Li & Karpathy, 2015).*

### *2.1 Neural networks*

To understand convolutional neural networks, it is first important to understand the concept behind a neural network. Neural Networks are modelled as collections of nodes (neurons) which are connected in a directed acyclic graph (Figure 3).

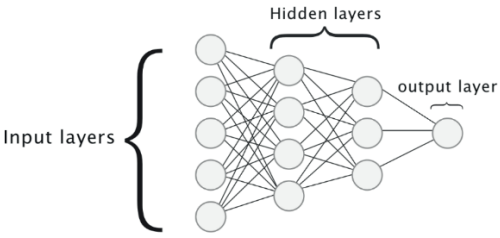


*Figure 3: A neural network consist of an input layer, an output layer, and depending on the model, a number of hidden layers in-between. This example has two hidden layers, and the layers are fully connected. Source: (Nielsen, 2015).*

Each connection between nodes has a weight, $w$, that represents how "important" the specific connection is. Each node takes the weighted sum[1] of the input and process it through an activation function, $f(x)$. The output of the activation function gives the output of the node. There are different activation functions, but the one proven to work best is the ReLU (Rectified Linear Unit) function (Figure 4). It is used in almost all of the state-of-the-art networks today – as will be shown in Section 3.

ReLU computes the function $f(x) = max(0,x)$, and it was found to accelerate the convergence of stochastic gradient descent[2]. However, an undesired property of the ReLU function, is that the units are fragile during training, and can "die." If a unit "dies" it means that the weights are updated in such a way that the node never activates again on any data point. The gradient flowing through the unit will forever be zero.

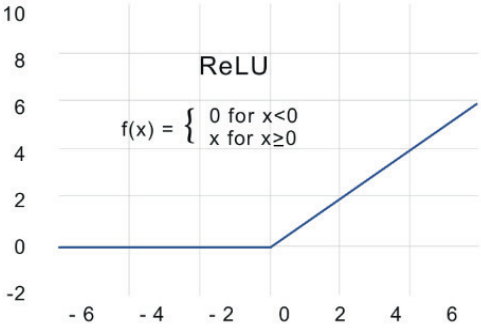


*Figure 4: The rectified linear unit.*

1. The weighted sum is the sum of all inputs times its weight
2. Stochastic gradient descent is a method for finding the minimums or maximums by iterations.

A bias node is a node that does not take an input, but instead has a constant value. The bias nodes are added to provide flexibility to the model. Take for instance a small network with two input nodes, and an output node (Figure 5). If the input nodes ($x1$ and $x2$) have the value zero, the weighted sum of the inputs would also be zero, no matter the value of their weights ($w1$ and $w2$). The network would lose its ability to change its output, and thereby it's ability to learn. If we, however, add an extra node with a constant value – the bias node – the network would be able to change the weight for the bias node, and thereby keep its ability to change the output of the network.

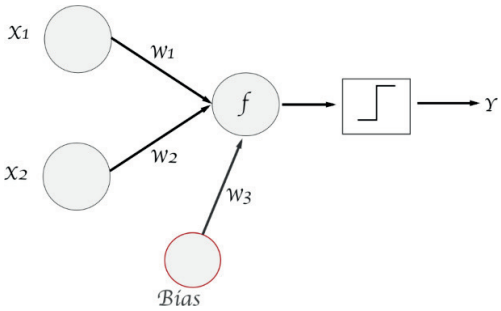


*Figure 5: An example of a small network where a bias node is added to increase the models' flexibility. If $x_1 = x_2 = 0$ the output y would be the same no matter how the values for $w_1$ and $w_2$ changed. However, when the bias node is added, weight w3 can be changed – and thereby the output.*

### *2.2 Convolutional neural networks*

CNNs are types of neural networks, and are as well made up of nodes that have learnable weights and biases. However, CNNs make the explicit assumption that the inputs are images. This allows encoding of certain properties into the architecture that cause a vast reduction in the number of parameters in the network. This is an important reason why CNNs are fast, despite their depth.

Since CNNs assume an image as input, it arranges its nodes in three dimensions – width, height, and depth. The width and height corresponds to the image size, and the depth represents the three channels of an image; red, green and blue. Most modern CNNs have three important layers – convolution layer, pooling layer and a fully connected layer (Karpathy, 2015).

**Convolution layer**

The convolution layer is the core building block of CNNs, and contains filters. A popular terminology for the convolution operation is to imagine a flashlight shining on an image. The area the flashlight shines on, represents the size of the filter. The flashlights then slide across the image, looking at small areas, peace by peace. As the filter slides across, it multiplies the values in the filter with the pixel values of the image, and sums them up (computing dot products). Every unique location in the input space therefore produces one number, and all these numbers are combined into a matrix called an activation map (Figure 6).

Each of the filters look for certain features in the image. Such features may, for example, be a curve, an edge or a feature of a specific color. Higher level filters (filters deeper into the network) look for combinations of these simpler features. The deeper into the network, the more complex the features become (Figure 7). The numbers in the activation maps therefore give an indication of, to what degree, the feature exists in the image and in which parts.

Since each filter has different nodes that look at different parts of the image, the network becomes locally connected. This means that only some of the neurons in one layer are connected to a neuron in the neighboring layer.

A convolution layer takes four hyper parameters:

- K = Number of filters
- F = The filters spatial size
- S = Stride, how much the filter is moved each step of the convolving
- P = Amount of zero padding

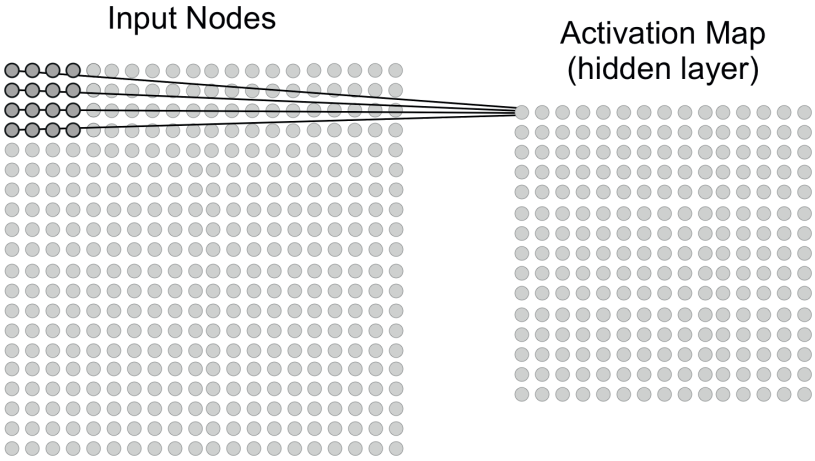


*Figure 6: Visualization of a 4 x 4 filter convolving around an input and producing an activation map. Source: (Deshpande, 2016).*

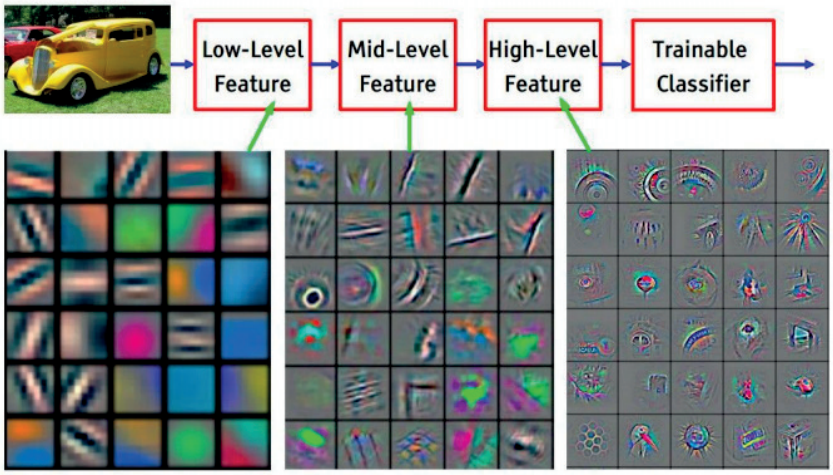


*Figure 7: Example of features that the filters in a convolution layer look for at different levels in a network. The deeper into the network (higher level), the more complex the features are. Source: (F.-F. Li & Karpathy, 2015).*

**Padding** means that you expand the spatial area by adding borders of zeros (Figure 8). Padding prevents the spatial area from decreasing when convolving the layers, and the necessary amount depends on the size of the filter. If you have a filter of size $F$ x $F$, you should use zero padding with $(F - 1)/2$ borders.

**Pooling layer**

In a CNN there is often pooling layers in between the convolution layers. The pooling layers are used to reduce the amount of parameters and computational complexity of the network by reducing the spatial size of every depth of the input. The pooling layer that has shown to perform best is the **MAXPOOL** (Karpathy, 2015). It traverses the matrix with a filter of size $F$ x $F$ and selects the largest element in the submatrix at each step (Figure 9). In other words, it saves the most significant part of the picture.

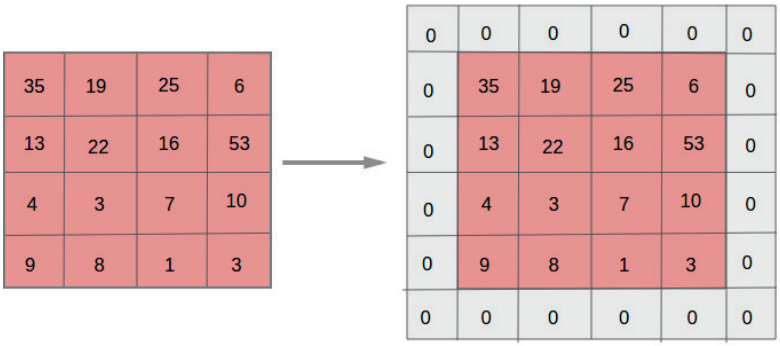


*Figure 8: Visualization of a matrix that is zero padded with one border.*

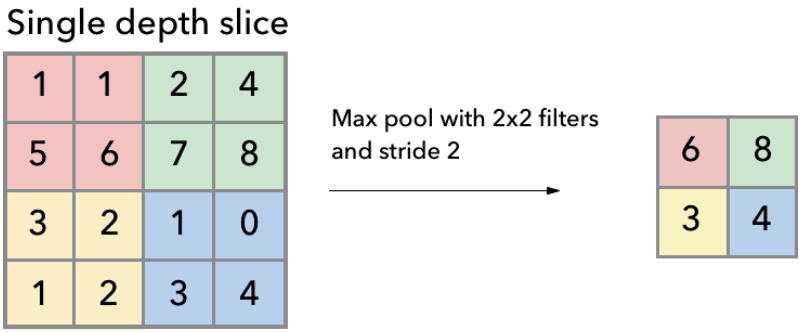


*Figure 9: Example of the maxpool operation, where the largest element in each submatrix is chosen.*

### *2.3 Training and testing*

A network's ability to learn is achieved through a training process, where the network is given a set of inputs with corresponding known outputs. By adjusting the weights that control the signal between two nodes, the network tries to map the input with the desired output. If the network generates a "good" output (output similar to desired output), there is no need to adjust the weights. If the network produces a poor output, the system adapts by altering the weights. This adjustment is done through a process called backpropagation.

**Backpropagation**

The training algorithm backpropagation can be split into 4 parts; the forward pass, the loss function, the backward pass and the weight update.

The algorithm compares the calculated output of the network with the desired output. It calculates the difference (error) between the two values, and this computed error is fed backward through the network, and used to adjust the weights so that the overall error of the network decreases. The goal of the process is to minimize the amount of loss, and the process can be seen as an optimization problem (Figure 10). The weights are adjusted according to the function:

$$W = Wi - \eta \cdot \frac{dL}{dW}$$

where $W$ is the weight, $L$ is the value, and $\eta$ is the learning rate. The learning rate, η, is a chosen parameter. A high learning rate means that bigger steps are taken during updating of the weights, and it may take the

model less time to reach the optimal set of weights. However, if the learning rate is too high it may lead to jumps that are too large to obtain a convergence of the error.

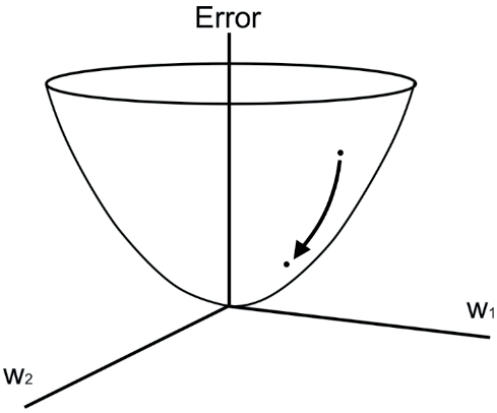


*Figure 10: The task of minimizing the loss can be visualized through a 3D parabola, where the weights of the neural net are the independent variables, and the dependent variable is the loss. The goal is to adjust the weights so that the loss decreases. In visual terms we want to get the lowest point in our 3D object.*

**Overfitting**
During training, a problem called overfitting may occur. Sometimes your model fit your training data perfectly, but it is still completely useless. The network has memorized instead of generalized the training data and does not know how to handle new data.

**Dropout layer**
One approach for decreasing the chance of overfitting is to include dropout layers. The dropout layer simply drops random sets of activations by setting them to zero in the forward pass. This prevents the network from becoming too "fitted" to the training data, since it has to learn how to provide the correct output even after losing some activations.

**Fine-tuned network**
A curious property of modern deep neural networks is that the networks tend to learn filters such as Gabor filters[3], edge and color blob detectors in their first layers – independent of the training set (Nogueira et al., 2016). These filters are useful for many different tasks, and doesn't have to be relearned for every new model. Training can therefore be resumed with a new dataset on a previously trained model, and this is called fine-tuning a network.

## 3 Development within visual recognition

A fair amount of publications on the topic of visual recognition refers to the *ImageNet* ILSVRC (*Large Scale Visual Recognition Challenge*)(Karpathy, 2015; Krizhevsky, Sutskever, & Hinton, 2012; J. Long, Shelhamer, & Darrell, 2015; Springenberg, Dosovitskiy, Brox, & Riedmiller, 2015). *ImageNet* is an image database that was designed for use in software research within visual recognition. As of 2017, 14 million URLs of images have been hand-annotated to indicate what objects are pictured. In 2010 *ILSVRC* was founded, and since then, it has acted as an annual software contest where research teams submit software that competes to correctly classify and detect objects and scenes in the images from the *ImageNet* database.

The challenge has attracted participants from more than fifty institutions, and among them teams from Microsoft and Google (Russakovsky et al., 2015). The participants are also allowed to submit closed work, and commercial companies do not need to reveal their code to be able to participate. It is therefore a fair assumption that the submitted work represents some of the best methods within visual recognition since 2010. Several of the publications described in this paper are from the ImageNet challenge, as the winners of the contest is a good indicator for progress within the field.

**AlexNet** (Krizhevsky, Sutskever, & Geoffrey E., 2012)
2012 was a turning point within classification and localisation due to the submission of

3. Gabor filter is a linear filter and is often used for edge detection.

a network called AlexNet. It was the first large-scale CNN, and it significantly outperformed previously implemented networks (Deng et al., 2009). The network stacked multiple convolutional layers on top of each other, which at the time was uncommon, but soon became the new norm.

**ZF net** (Zeiler & Fergus, 2014)
Zeiler and Fergus focused on increasing the understanding of CNNs, and stated that without a deeper understanding of the networks, future work would be based solely on trial and error. They therefore proposed a visualisation technique called deconvolution. A *deconvnet*, was attached to every layer and gave a path back to the image pixel. This made it possible to examine what type of structure had generated the specific activation map (Figure 11).

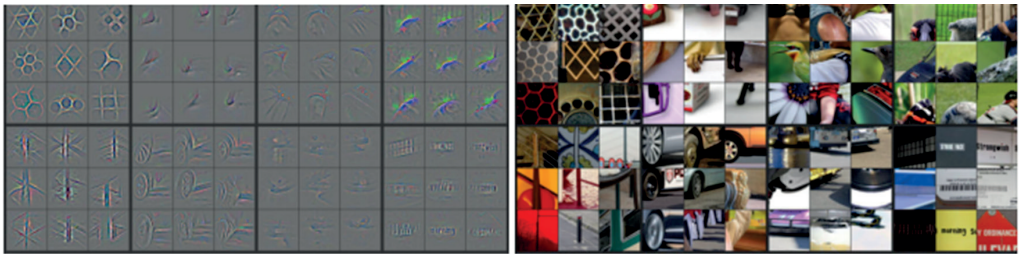

*Figure 11: An example of activation maps and the actual structure (Zeiler & Fergus, 2014).*

**GoogLeNet** (Szegedy et al., 2014)
A research team from Google presented a more efficient network called GoogLeNet. Its main contribution was the development of an *Inception Model*. By inserting 1 x 1 convolution blocks before the expensive parallel blocks (Figure 12), they reduced the number of features drastically, which made the network faster.

**VGGNet** (Simonyan & Zisserman, 2015)
Simonyan and Zisserman introduced a deeper network, with up to 19 weight layers. In order to reduce the number of parameters, they used small filters of size 3 x 3, and a stride of one. The design of the network increased performance significantly.

**ResNet** (He et al., 2015)
Since the VGGNet showed how deeper networks improve accuracy, He et al., (2015) wondered: *"Is learning better networks as easy as stacking more layers?"*. They therefo-

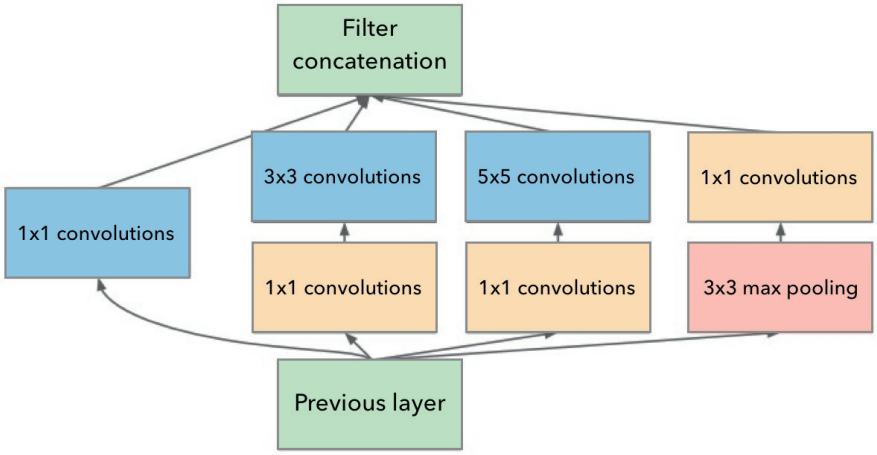


*Figure 12: GoogLeNet's inception model, where a 1 x 1 convolution block is added to reduce the number of parameters.*

re created a network that had a total of 152 layers – eight times the number of layers that the VGGNet had. They dealt with the increased depth by adding *shortcut connections* that made lower layers available to nodes in a higher layer. They only wanted to add a layer if it improved the performance, so you may say that each layer was responsible for fine-tuning the output from the previous layer. Because of these shortcut connections, ResNet actually had a lower complexity than VGGNet – despite the networks increased depth.

**Trimps-soushen** (Russakovsky et al., 2015) The team called Trimps-Soushen submitted the winning work within both the task of classification and localization in 2016. The network used several pre-trained models, including ResNet, as start parameters.

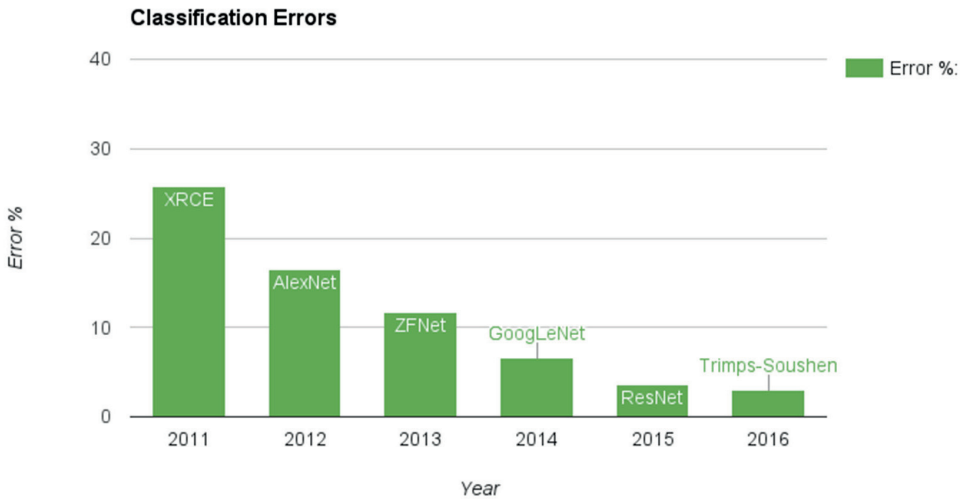


*Figure 13: Results from the classification task in ILSVRC from 2011–2016.*

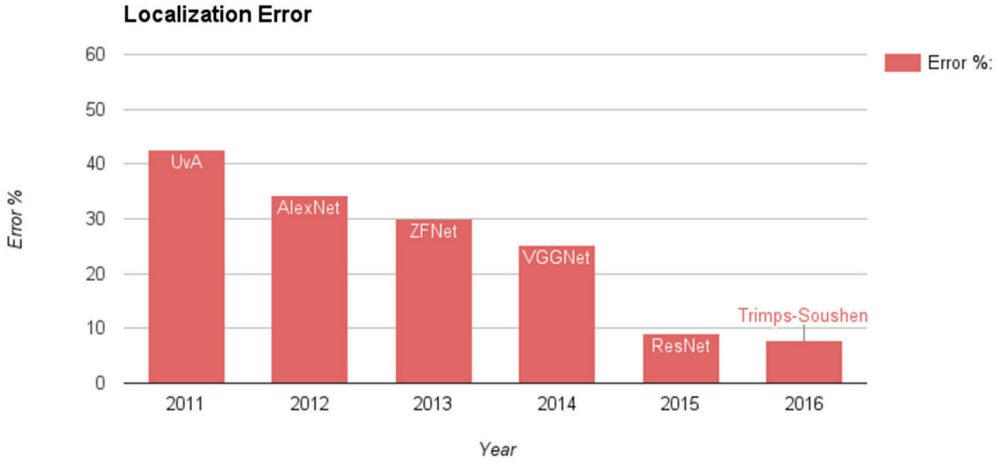


*Figure 14: Results from the localization task in ILSVRC from 2011–2016.*

### *3.1 Object detection*

When CNN was introduced, new methods for object detection followed. The first methods consisted of using fast CNN classifiers for detection. You simply tested every possible bounding box in the image, and the one with the highest classification rate was kept. The methods then shifted towards giving region proposals for the bounding boxes, instead of trying them all.

**R-CNN** (Girshick, Donahue, Darrell, & Malik, 2014)
*Region based CNN* (R-CNN) was proposed in 2014, and found region proposals for the input image, lowering the number of possible regions to around two thousand. Each region (box) in the image would be cropped and warped to some fixed size, and then run through a CNN classifier. The CNN then had a regression head and a classification head, that would correct the boxes that were a little off, for example shifted or of wrong size.

**Faster R-CNN** (Ren, He, Girshick, & Sun, 2015)
Even though R-CNN improved object detection a lot, it was really slow at test time. Faster R-CNN solved this problem by sharing computation of convolutional layers across different region proposals.

**HyperNet** (Kong, Yao, Chen, & Sun, 2016)
A problem with the methods combining CNN with region proposals is that they still test several thousand different positions, and struggle with small-size object detection and precise localisation. The network called HyperNet was therefore proposed. HyperNet handles region proposals and object detection jointly, by designing hyperfeatures which aggregate hierarchical activation maps first, and then compress them into a uniform space. Their method produces small numbers of object proposals while guaranteeing high recalls.

As well as the HyperNet they also tweaked their architecture to test a version called HyperNet-SP where they speed up the network by allowing a small decrease in accuracy. Table 1 and Table 2 shows the results of their methods.

*Table 1: Results from comparing different detection methods on the PASCAL VOC 2012 dataset for detection of nineteen different classes. Bold number represent the highest accuracy number for the specific class. As can be seen, HyperNet is the network that has the most classes with the highest accuracy. However, the difference between HyperNet and HyperNet-SP is very small. Source: (Kong et al., 2016)*

| Approach | mAP | Aero | Bike | Bird | Boat | Bottle | Bus | Car | Cat | Chair | Dog | Horse | Mbike | Person | plant |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Fast R-CNN** | 68.4 | 82.3 | 78.4 | 70.8 | 52.3 | 38.7 | 77.8 | 71.6 | **89.3** | 44.2 | **87.5** | 80.5 | 80.8 | 72.0 | 35.1 |
| **Faster R-CNN** | 70.4 | **84.9** | 79.8 | 74.3 | 53.9 | 49.8 | 77.5 | 75.9 | 88.5 | 45.6 | 86.9 | **81.7** | 80.9 | 79.6 | 40.1 |
| **HyperNet** | **71.4** | 84.2 | 78.5 | 73.6 | **55.6** | **53.7** | **78.7** | **79.8** | 87.7 | **49.6** | 86.0 | **81.7** | **83.3** | **81.8** | **48.6** |
| **HyperNet-SP** | 71.3 | 84.1 | 78.3 | 73.3 | 55.5 | 53.6 | 78.6 | 79.6 | 87.5 | 49.5 | 85.6 | 81.6 | 83.2 | 81.6 | 48.4 |

*Table 2: Overview of running time for object detection methods on the PASCAL VOC 2012 dataset. The times are given in milli seconds, and shows that the time needed for calculating proposals and detection are much less for the HyperNet-SP. Source: (Kong et al., 2016).*

| Approach | Conv (shared) | Proposal | Detection | Total |
|---|---|---|---|---|
| **Fast R-CNN** | 140 | 2260 | 170 | 2570 |
| **HyperNet** | **150** | 810 | 180 | 1140 |
| **HyperNet-SP** | 150 | 20 | 30 | **200** |

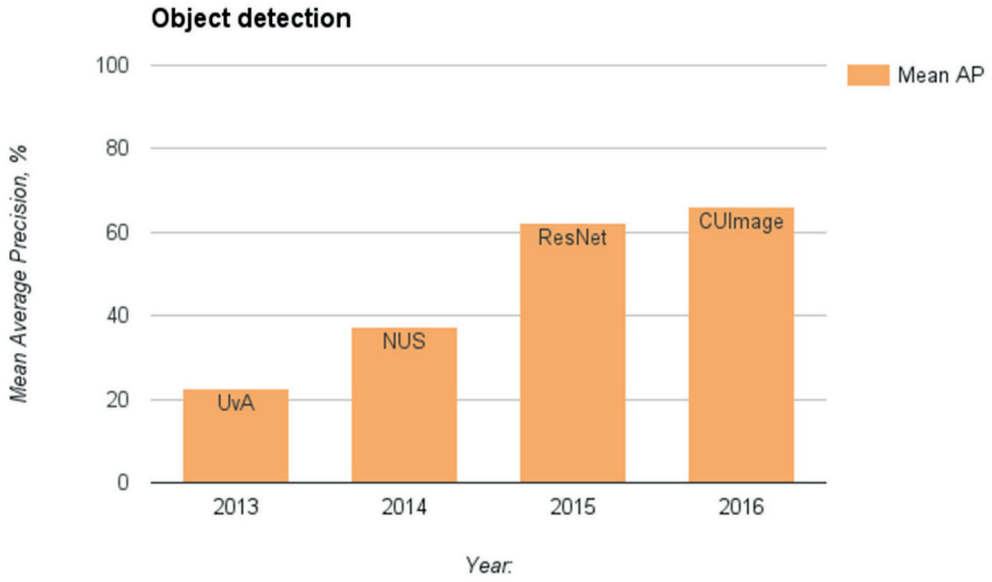


*Figure 15: Results from the object recognition task in ILSVRC from 2013–2016.*

### *3.2 Image Segmentation*

Image segmentation has also improved considerably since the revolution of deep learning. The idea behind segmentation is to look at patches of the input image and classify each patch through a CNN. For each pixel in the image, a patch is created from its surrounding pixels. The patch is classified, but instead of assigning the class to all the pixels in the patch, it is only assigned to the center pixel – the pixel that created the patch (Figure 16). A problem with this approach is the down-sampling of the image through strides and pooling layers. The output image will be smaller (less pixels), thus less accurate than the input image. Another problem is that for *each pixel* a patch of the image is classified by a CNN, which is computationally expensive.

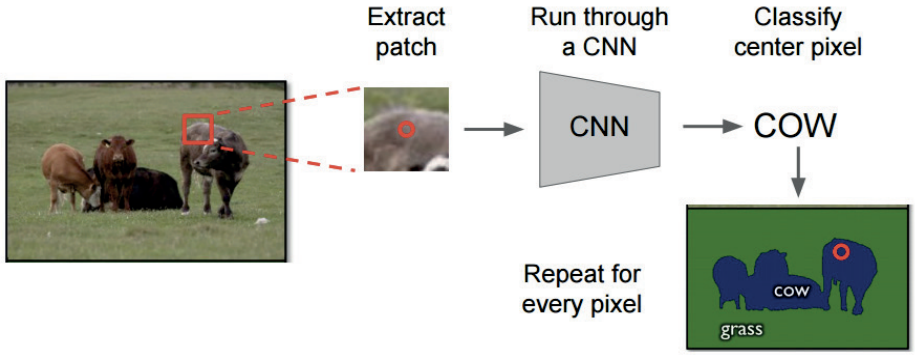


*Figure 16: Visualization of the basic idea behind image segmentation. Each pixel in the image is classified to the value that its surrounding patch is classified to, by the use of a CNN. Source: (F. Li & Karpathy, 2015).*

**Fully Convolutional Networks** (J. Long et al., 2015)
In 2015, a new fully connected network was proposed that used deconvolution to up sample the image after classification (Figure 18). They also added skip-connections similar to ResNet (He et al., 2015), that improved the borders of the segmentation (Figure 17). The network dramatically improved performance, while also speeding up the learning process (J. Long et al., 2015). The network got, among other results, a pixel accuracy [4] of 85,2%, which showed state-of-the-art performance.

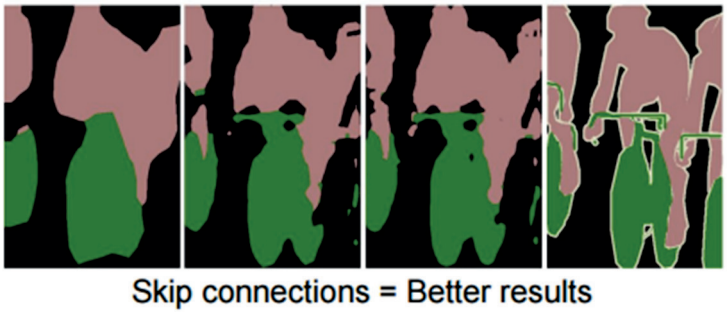


*Figure 17: Adding skip-connections to the network improved, especially, the borders of the segmentation. Source: (J. Long et al., 2015)*

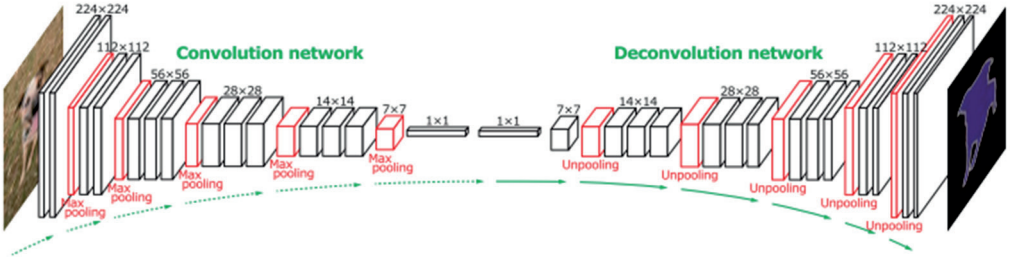


*Figure 18: Noh, Hong, & Han, (2015) used deconvolution to upsample the image after classification.*

## 4 Visual recognition in remote sensing images

Remote sensing methods measure the amount of electromagnetic energy reflected from objects, and mathematical and statistical algorithms are applied to the results to extract valuable information. The remotely sensed data may be obtained systematically through very large geographical areas, and is now critical to the successful modelling of numerous natural and cultural processes (Jensen, 2014).

The previous section showed work on visual recognition, but the datasets used in the training and testing were never aerial or remotely sensed images. A prominent question will be how we can use existing neural networks for the remote sensing domain in the best way.

**Towards better exploiting CNNs** (Nogueira et al., 2016)
Nogueira et al. (2016) stated that it is not always feasible to fully design and train a new CNN. They wanted to both test how different networks performed, but also what strategy best benefitted existing CNNs. The three strategies they tested were: (1) fully trained CNNs,

4. Pixel accuracy is the number of pixels correctly classified divided by the amount of pixels.

(2) fine-tuned CNNs and (3) pre-trained CNNs. By using three remote sensing datasets they tested six popular CNNs – among them: *AlexNet*, *VGGNet* and *GoogLeNet*.

Their results showed that fine-tuning tends to be the best strategy. They achieved classification accuracy up to 99.47% when fine-tuning *GoogLeNet*, which is higher accuracy than any other networks at present. However, the most important discovery was that fine-tuning works really well, even when the images used for fine-tuning are different from the images that the network was originally trained for.

**Semantic segmentation of Aerial Imagery** (Sherrah, 2016)
The conclusion from Nogueira et al. (2016) also coincides with the results from Sherrah (2016). They proposed a fine-tuned fully convolutional network that also gave state-of-the-art results for semantic segmentation . The fine-tuned network showed superior results compared to the network trained from scratch. This shows that fine-tuning existing networks doesn't only work for the task of classification, but also for the task of segmentation within the remote sensing domain.

**Ensemble of CNNs** (Marmanis et al., 2016)
In a publication on semantic segmentation, Marmanis et al. (2016) also used semantic segmentation. They used deconvolution to undo the spatial down-sampling, and fully convolutional networks to save the explicit location of the classified pixels. They used very high resolution aerial images, with less than 10 cm ground sampling distance[5] in their work. Even though the images used was not satellite images, the method still applies. The only difference is that satellite images has lower resolution, and would therefore have lower accuracy. As well as the aerial images, they also made use of a DEM[6]. They set up two separate paths in the network for the two types of input. They assumed that height- and pixel-values have statistical differences that would need different features for recognition, and chose to merge the two paths at a very high level. Their results showed segmentation accuracy up to 88.5%, which is state-of-the-art results (Figure 19).

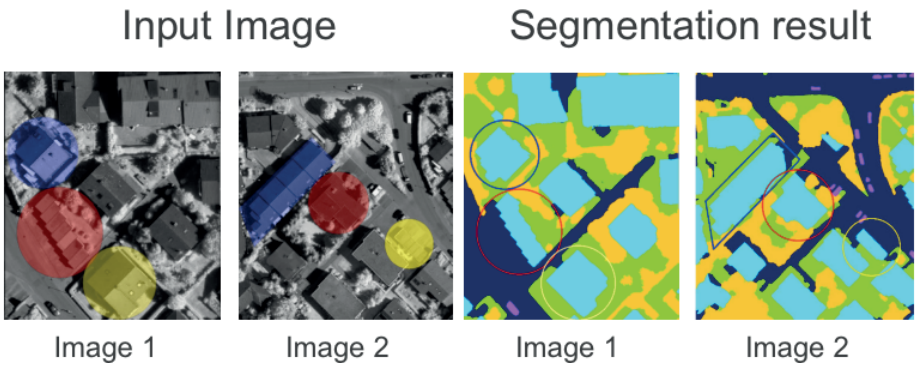


*Figure 19: Example of results from the segmentation performed by Marmanis et al. (2016).*

**5 Conclusion**
Based on the theory and techniques covered in this paper it is likely that software can help automation of visual recognition in remotely sensed images. As was shown in Section 4, existing methods have achieved classification accuracy up to 99.47% and segmentation accuracy up to 88.5%. Taking into consideration how much faster a computer can process images than humans, this high

5. Ground sample distance (GSD) is the distance of the square one pixel in an image covers in the terrain
6. A digital elevation model (DEM) gives height measurements for the terrain. Depending on the model, it includes objects (e.g. houses, trees).

accuracy indicates unused potential. In many ways, it is only your imagination that sets the limit for how this technology can be used. The most obvious within remote sensing might be to help digitize images for creating maps, by segmenting and classifying objects and areas. Aside from this, it could also be used in change detection, monitoring of animals, invasive plant ranges, etc.

One specific case that might be solved is the mapping of rooftops from satellite images by performing semantic segmentation. As was shown in Section 4, Marmanis et al. (2016) proposed a network that used both digital elevation model and images as input, that gave state-of-the-art results. This is a technique that could work well for recognizing buildings as well, since buildings have a distinct increase in elevation, compared to its surroundings.

Instead of only using RGB (red, green, blue) matrices as input, it would likely give higher performance to add a non-visible specter as well (Bollinger, 2017). A good band for distinguishing buildings from its surroundings is the infrared band. However, having five input matrices would increase the complexity of the network drastically, so removing one of the RGB matrices might be necessary to lower the training time. Which of the colors that would cause the least decrease in accuracy requires further research.

The papers described in Section 3 and 4 shows that there has been a huge improvement within the field of visual recognition the last four years, and that the best methods are from papers released in 2016 and 2017. This indicates that we are at the beginning of the revolution within the use of deep learning for visual recognition, and the methods will most likely keep on improving.

## 6 Bibliography


Bollinger, D. (2017). Open Source Machine Learning – Development Seed. Retrieved January 24, 2017, from https://www.developmentseed.org/blog/2017/01/30/machine-learning-learnings/

Castelluccio, M., Poggi, G., Sansone, C., & Verdoliva, L. (2015). Land Use Classification in Remote Sensing Images by Convolutional Neural Networks. *arXiv Preprint arXiv:1508.00092*, 1–11. Retrieved from http://arxiv.org/abs/1508.00092

Deng, J., Dong, W., Socher, R., Li, L., Li, K., & Fei-Fei, L. (2009). ImageNet. Retrieved November 24, 2016, from http://image-net.org/

Deshpande, A. (2016). A Beginner's Guide To Understanding Convolutional Neural Networks Part 2. Retrieved November 29, 2016, from https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 580–587. http://doi.org/10.1109/CVPR.2014.81

GISGeography. (2016). Spectral Signature Cheatsheet – Spectral Bands in Remote Sensing – GIS Geography. Retrieved November 25, 2016, from http://gisgeography.com/spectral-signature/

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *Arxiv.Org*, *7*(3), 171–180. http://doi.org/10.3389/fpsyg.2013.00124

Jensen, J. R. (2014). *Remote Sensing of the Enviroment: An Earth Resource Perspective* (Second Edi). Pearson Education.

Karpathy, A. (2015). CS231n? : Convolutional Neural Networks for Visual Recognition. Retrieved September 26, 2016, from http://cs231n.github.io/convolutional-networks/

Kong, T., Yao, A., Chen, Y., & Sun, F. (2016). HyperNet: Towards Accurate Region Proposal Generation and Joint Object Detection. *Cvpr*, 845–853. http://doi.org/10.1109/CVPR.2016.98

Krizhevsky, A., Sutskever, I., & Geoffrey E., H. (2012). Imagenet. *Advances in Neural Information Processing Systems 25 (NIPS2012)*, 1–9. http://doi.org/10.1109/5.726791

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, 1–9. http://doi.org/http://dx.doi.org/10.1016/j.protcy.2014.09.007

Li, F.-F., & Karpathy, A. (2015). Convolutional Neural Networks (Lecture 7). Retrieved November 28, 2016, from http://cs231n.stanford.edu/slides/winter1516_lecture7.pdf

Li, F., & Karpathy, A. (2015). Lecture 2: Image Classification pipeline. Retrieved November 26, 2016, from http://cs231n.stanford.edu/slides/winter1516_lecture2.pdf

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07–12–June*, 3431–3440. http://doi.org/10.1109/CVPR.2015.7298965

Long, Y., Gong, Y., Xiao, Z., & Liu, Q. (2017). Accurate Object Localization in Remote Sensing Images Based on Convolutional Neural Networks. *IEEE Transactions on Geoscience and Remote Sensing*, 1–13. http://doi.org/10.1109/TGRS.2016.2645610

Macukow, B. (2016). *Neural Networks – State of Art, Brief History, Basic Models and Architecture*. (K. Saeed & H. Wladyslaw, Eds.) (Vol. 8104). Springer International Publishing. http://doi.org/10.1007/978-3-642-40925-7

Marmanis, D., Wegner, J. D., Galliani, S., Schindler, K., Datcu, M., Stilla, U., & Sensing, R. (2016). Semantic segmentation of aerial images with an ensemble of cnns, *III*(July), 12–19. http://doi.org/10.5194/isprsannals-III-3-473-2016

Marr, B. (2016). A Short History of Machine Learning. Retrieved November 7, 2016, from https://tanjo.ai/contents/1084258

McCulloch, W. S., & Pitts, W. (1943). A Logical Calculus of the Idea Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, *5*, 115–133. http://doi.org/10.1007/BF02478259

Nielsen, M. A. (2015). *Nural Networks and Deep Learning*. Determination Press.

Nogueira, K., Penatti, O. A. B., & dos Santos, J. A. (2016). Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognition*. http://doi.org/10.1016/j.patcog.2016.07.001

Noh, H., Hong, S., & Han, B. (2015). Learning Deconvolution Network for Semantic Segmentation, *1*. http://doi.org/10.1109/ICCV.2015.178

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Nips*, 1–10. http://doi.org/10.1016/j.nima.2015.05.028

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., … Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, *115*(3), 211–252. http://doi.org/10.1007/s11263-015-0816-y

Schmidhuber, J. (2015). Deep Learning in neural networks: An overview. *Neural Networks*, *61*, 85–117. http://doi.org/10.1016/j.neunet.2014.09.003

Sherrah, J. (2016). Fully Convolutional Networks for Dense Semantic Labelling of High-Resolution Aerial Imagery. *arXiv*, 1–22. Retrieved from http://arxiv.org/abs/1606.02585

Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations*, 1–14. http://doi.org/10.1016/j.infsof.2008.09.005

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2015). Striving for Simplicity: The All Convolutional Net. *Iclr*, 1–14. Retrieved from http://arxiv.org/abs/1412.6806

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., … Arbor, A. (2014). Going Deeper with Convolutions. http://doi.org/10.1109/CVPR.2015.7298594

Zeiler, M. D., & Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. *Computer Vision–ECCV 2014*, *8689*, 818–833. http://doi.org/10.1007/978-3-319-10590-1_53