M.Sc. Thesis
Master of Science in Engineering

DTU Compute
Department of Applied Mathematics and Computer Science

Using Convolutional Neural Networks and Data Science to Explore Patterns of Religiosity over Centuries

Thorsteinn Gunnar Jonsson (s171945)

Kongens Lyngby 2020

DTU Compute
Department of Applied Mathematics and Computer Science
Technical University of Denmark

Matematiktorvet
Building 303B
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary

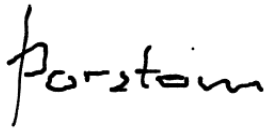
Religion is an important part of human life all over the globe. The importance of religiosity in social life is, however, different. In some areas, religion is the main social value to follow, while in other areas religion, albeit being a part of daily life there, isn't the only social value of importance to human beings. By using recent advancements in object detection and semantic segmentations in the field of machine learning, a neural network has been taught to classify a few commonly appearing symbols. Those religious symbols are; a cross, the Bible, praying hands, doves, angels and the Star of David.

Using a neural network to detect symbol greatly speeds up a process which has to be done manually. Hundreds of thousands of memorials can now be scanned in a matter of hour and the information output from that can help shine a light on the correlation between religiosity and lifespan in different parts of the world. It also indicates a relationship being between religious symbolism on gravestones and religiosity of that person in real life.

Preface

This Master's thesis was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfillment of the requirements for acquiring a Master's degree in Mathematical Modelling and Computation.

Kongens Lyngby, January 1, 2020



Thorsteinn Gunnar Jonsson (s171945)

Acknowledgements

Firstly, I want to thank my supervisor Sune Lehmann Jørgensen for his advice over the course of the thesis. For help with segmentation of images my greatest gratitudes to the work done by Jochen Gebauer, Tobias Ebert and their team of student assistants. For support, help with technical material over the course of the project and proofreading, I want to thank my friends Hallgrimur Hrafn Einarsson, Ingvar Bjarki Einarsson and Enrique Yegros Miller.

Contents

Nomenclature

Abbreviations

ANN	Artifical neural network
CNN	Convolutional Neural Network
CPU	Central processing unit
FPN	Feature pyramid network
GPU	Graphics processing unit
HOG	Histogram of oriented gradients
IoU	Intersection over Union
mAP	mean average precision
R-CNN	Regions with Convolutional Neural Network features
ReLU	Rectified Linear Unit function
ResNet	Residual network
RoI	Region of interest
RPN	Region proposal network
SIFT	Scale invariant feature transformation

List of symbols

α	Momentum
δ	Errors
η	Learning rate
λ	Weight decay
μ	Mean

ρ Correlation

σ Standard deviation

σ_x Standard error of the mean

b Batch size

t Targets

w Weights

a Activation or feature map

d Depth

f Function

h Activation function

$\mathrm{h}_k, h_a, h_{out}, h_I$ Height of kernel, input feature map, output feature map and image.

I Input image

k Kernel filter

L Cost function

N Number of samples

P Padding

S Stride

$\mathrm{w}_k, w_a, w_{out}, w_I$ Width of kernel, input feature map, output feature map and image.

X Data set

x_i Pixel value

y Output set

y_i Pixel value

z Transformed activation

CHAPTER 1

Introduction

Whether personal religiosity has any psychological benefits has been studied extensively[1]. Religion has been found to have a positive effect on health[2]. But those effects are not the same everywhere. The effects of religiosity as a social has been hypothesized in [1] to explain cross-cultural differences in benefits it offers. Religious countries are said to have religiosity as the defining social value, while in non-religious countries religiosity, albeit being a social value, is not as important.

[1] uses results from three studies to examine the relationship between religiosity and self-esteem. They use covariates such as GDP per capita to exclude alternative explanations, and their results found relations between religiosity and self-esteem to be higher in religious countries than secular countries.

Other methods have been proposed to examine the positive psychological benefits of religiosity, and the differences of benefits between religious and secular areas. [3] proposes using imagery data of memorials to investigate this connection. Data on 6400 memorials from `https://www.findagrave.com/` is used in that investigation. [3], [4] and [4] suggest using symbols engraved on gravestones to classify them as either religious or non-religious. Symbols on gravestones and their connections with religion are described in [4]. [3] uses the the lifespan of each memorial, the symbol types and the religious adherency rate of each memorials corresponding county[5] and finds correlation between the symbolism and reached age, depending on county.

Manually classifying the symbolism engraved on gravestones is very time consuming. Is it possible to automate that process?

This project trains a powerful CNN to locally segment symbols on gravestones from `https://www.findagrave.com/` automatically. Convolutional Neural Networks have proved very capable at image classification[6]. In image classification competitions, CNNs have reportedly beaten the error rate of humans[7], [8]. For instance segmentation and local object detection, CNNs have had increased popularity since [9] was published. [10] reports promising success in these areas and has since being published been used to segment nuclei in microscopic images[11] and reconstruct 3 dimensional building from aerial footage[12].

This project aims to utilize [10] to segment religious symbols on gravestones from `https://www.findagrave.com/` to classify them as religious or non-religious, like done in [3]. The religious symbols considered will be crosses, Bibles, angels, doves, praying hands and the Star of David.

A theoretical discussion on how neural networks and convolutional neural networks are built will be presented in chapter 2. Chapter two will also further articulate the progress which has been made on CNN models, specifically those which do local object detection.

Chapter 3 will go over technical details of the project. The programming work done in this project will be explained. How the data in this project was gathered from the internet will be explained, as well as how data was used and modified. The characteristics and hyperparameters of the specific neural network used in this project will be outlisted as well.

The performance of the neural network will be reviewed in the fourth chapter. The network will then be used to segment a large amount of data scraped from `https://www.findagrave.com/`. Results from that coupled with data from memorial pages will be used to explore the relationship between symbol engravings, geographical religiosity score and length of life. That will be done both for data from the United States and for other countries, but will be done in seperate sections.

A conclusion and possible future work will follow in chapter 5, looking back on the results and performance of the network, and possible changes which could be made.

All work done in this project can be found on `https://github.com/IceTharu/masters_thesis/`.

CHAPTER 2

Theory

This chapter will briefly introduce the concept of religiosity as a social subject and previous studies made on the relationship between religiosity and wellbeing. Following that, the basic operations of neural networks and the different layers in convolutional neural networks will be introduced, as well as a more focused explanation about the usage and advancements made in using convolutional neural networks in the field of object detection and semantic segmentation.

2.1 Religiosity as a Social Value

The effects of personal religiosity and self-esteem had previously been examined in [1], [2]. It has been found to have a general positive influence on health. In [1] the hypothesis of religiosity as a social value is introduced to explain the difference of the importance of religiosity depending on location. It describes that religiosity is the defining social value in religious cultures and fulfilling religious requirements is beneficial for individuals living there. It also states that in secular cultures, where other social values are more important than religiosity, the connection between religiosity and wellbeing is less apparent. [1] uses online data gathered through a survey between 2001 and 2009 in 65 countries. The paper reports a correlation between religiosity and self-esteem.

Other types of data to examine the relationship between religiosity and wellbeing has been used in [3], where imagery data of gravestones is used. The images are classified as a religious sort if symbols of religious nature appear on them, or as a non-religious sort in case religious symbols are absent. Symbols can be classified as religious or not accordin to [4]. Manually classifying symbols such as crosses, Bibles, angels, lambs and doves on 6400 images of gravestones found on `https://www.findagrave.com/` from 64 counties in the United States shows a correlation between the longevity of life and whether a religious symbol appears on the gravestones. The counties are classified by their religious adherence rate from the U.S. religion census[5].

2.2 Neural networks

This section will introduce artificial neural network, starting with a overview of how they are built up. Following that, a few of the processes of ANNs will be explained. Forward pass, activation functions, optimization, backpropagation and regularization of neural networks will be given a look.

Neural networks were introduced as the Perceptron by Rosenblatt[13] back in the 1960's, but they didn't reach the widespread popularity they have nowadays until in recent times.

Neural networks are loosely inspired by the way information is processed in biological brains[14]. Neural networks have been successfully used in a lot of different tasks. Simple artificial neural networks can be used to classify images of handwritten digits, and more complex neural networks can be trained to recognize a variety of objects, such as segmenting nuclei in microscopic images[11] and reconstructing 3D buildings from aerial images[12].

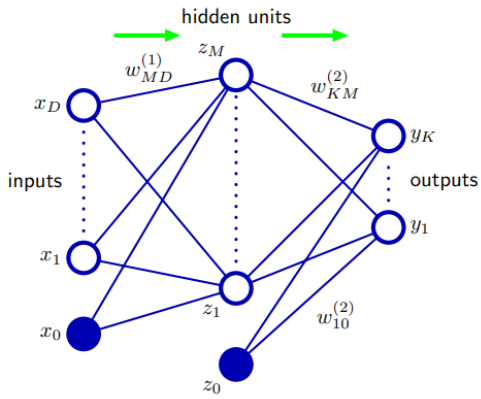


Figure 2.1: A simple diagram of a two layer neural network.[14] (p. 228)

Neural networks are built up of nodes and layers. The number of layers within a neural network denote its depth, and the number of nodes within a layer denotes its width. Artificial neural network require three different types of layers; an input layer, hidden layers and an output layer. An artifical neural network only has one single input and output layer, but of hidden layer can be several. Nodes within each layer are connected to nodes in the adjacent layers and the connection between them are called weights w. An extra bias node is included in the input and hidden layers, and allows better flexibility of a neural network. A general architecture of an artificial neural network composed of three layers can be seen in figure 2.3

The nodes in the hidden layer need to be more than the nodes of either the input or output layer. In the hidden layers, nonlinearities are introduced to make neural networks capable of representing nonlinear functions. These nonlinear functions in-

troduced at the hidden layer are generally required to be nonlinear and differentiable, but some of the most commonly used nonlinear functions such as the rectified linear functions are not differentiable at every point. That is permitted due to the extremely slim chances that gradients will reach a point of local minimum during training.[14], [15]

The number of nodes in the last layer depends on the type of the application intended for the artificial neural network. Neural networks can both be applied to regression and classification problems. For regression, the output layer will be composed of a single node, but for classification, the number of nodes will usually represent the numbers of different classes required.

Neural networks are function approximators which in theory can map any function f(X). They take as an input a vector X, and using a nonlinear function f, map those inputs as an output vector y.

$$y = f(X) \tag{2.1}$$

2.2.1 Forward pass

Forward pass, also called forward propagation, is an integral part of neural networks and has been the building block of the networks since they were first introduced in 1958.[13] During forward propagation, dot products of the nodes within that layer with the weights connecting them to the next layer are computed, resulting in the activations of nodes in the next layer. Those activations are transformed with a nonlinear activation function to ensure that the solution will be nonconvex. At the output layer, depending on the number of nodes in it, an output activation function such as the sigmoid or softmax funtions are used.

For the first layer of 2.3 with D dimensions, the activations a of nodes in the next layer with M dimensions are computed as a dot product of the weights w and input values x, as shown in equation 2.2. Those activations are then transformed using a nonlinear activation function h. After transforming the activations in the hidden layer, the activations of the output layer with dimensions K is computed as shown in equation 2.4 These computations can be repeated if more than one hidden layer is included within an artificial neural network.

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{2.2}$$

$$z_j = h(a_j) \tag{2.3}$$

$$a_k = \sum_{i=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{2.4}$$

Equations 2.2, 2.3 and 2.4 show how the activations are computed.

$$y_k = \frac{1}{1 + e^{-a_k}} \tag{2.5}$$

For artificial neural networks used for regression, the activation of the output node can be output directly as y, since there will only be one. For binary classification problems the sigmoid function can be used to transform the activation of the output node to a probabilistic value.

For multiclass problems, the softmax function in equation 2.6 is used to convert the output activations to probabilistic values. The softmax function is advantageous for neural networks classifying multiple classes due to the fact that all elements y will sum together to 1 and represents a probability distribution.[15]

$$y_k = \frac{e^{a_k}}{\sum_{i=0}^{n} e^{a_i}} \tag{2.6}$$

2.2.2 Activation functions

Several nonlinear functions can be used in neural networks. As mentioned previously, they are generally required to be nonlinear and differentiable.

Amongst commonly used nonlinear functions are the sigmoid function (shown in eq. 2.5), hyperbolic tangent and the rectifier function. Both the sigmoid function and hyperbolic tangent suffer from the problem that their gradients are zero for both very large and very large activations. When backpropagating with these nonlinear functions there is a risk that the gradient of the weights will vanish and thus no updates will be done in reality.

The rectifier function is preferred for deep neural networks[15].

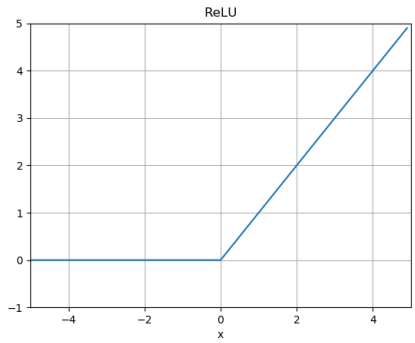


Figure 2.2: ReLU function

It is the most widely used activation function in deep convolutional neural networks, often abbreviated as *ReLU*, standing for rectified linear unit. The reason why the *ReLU* function is preferred over other activation functions such as the sigmoid

and hyperbolic tangent function is that for very high activations we do still have a value for the gradient so updates will be made to the weights. For activations of lower than zero, the *ReLU* function will increase sparsity in the computation.

$$f(x) = max(0, x) \tag{2.7}$$

The mathematical representation of the *ReLU* function is shown in equation 2.7 and visualized in figure 2.2. It is a differentiable function at every point except at $x = 0$, where $f'(x_-) = 0$ and $f'(x_+) = 1$. The derivative of the function at that points is generally either configured to 0 or 1.[15].

2.2.3 Optimization

A lot of different functions can be used to compute the loss of a neural networks, such as a multiclass SVM loss, log loss and cross-entropy loss. The cost function for neural networks will be nonconvex due to the nonlinearities introduced in the hidden layers[15].

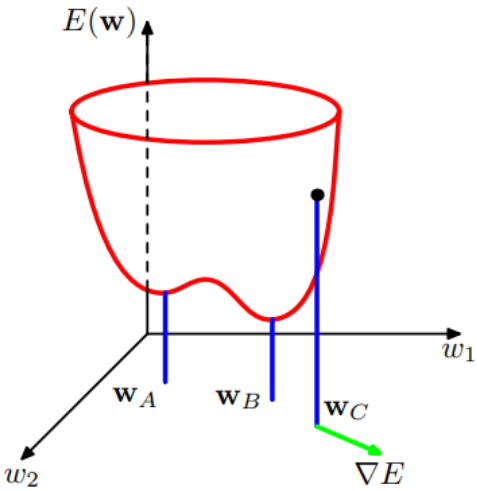


Figure 2.3: A 3-dimensional representation of an error function.[14] (p. 236)

The cross-entropy function can be used to compute the error of a classification neural network for many classes, and is defined in equation 2.8

$$L(w) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\} \tag{2.8}$$

where y is the value computed by the neural network and t is the true value. The loss computed will be zero when $y_n = t_n$. In other cases, it will take a value higher than zero.

To minimize the loss function and move closer the predicted values y closer to true values t, steps can be taken in the negative direction of the gradient of the loss function, or $\nabla L(w) \to 0$. The gradient calculates the slope of the loss function, and taking it in the negative direction causes us to move in the direction of a local minima.

The loss is minimized by using stochastic gradient descent, which computes for one point at a time the weight gradient given a learning rate η. The momentum of the gradient can be used to speed up converence. It uses the updates Δw at each timestep t, accompanied with a momentum constant α to accelerate the learning[15].

$$w^{t+1} = w^t - \eta \nabla L_n(w^t) + \alpha(w^t + \Delta w^t)[15] \tag{2.9}$$

2.2.4 Backpropagation

The learning algorithm vital to neural networks today, backpropagation, was introduced back in 1986 by utilizing the Chain Rule of Calculus to calculate the gradients of the loss through neural networks[16]. That addition allows evaluating the gradient between the weights in a computationally easy manner and, most importantly, allows neural networks to learn.

The gradients of the error function (f.x. equation 2.8) used in a neural network are fed backwards, or backpropagated, through the network after an iteration of forward pass, updating the weights between nodes and the activations of nodes in the process.

Backpropagation can be broken up into two separate parts. In one part the derivatives of the loss with respect to the weights is derived. In the other part do we use those derivatives to adjust the weights.

The derivative of the loss with respect to the weights can be split into two parts, the derivative of the loss L with respect to the activations a and the derivative of the activation with respect to weights.

$$\frac{\partial L}{\partial w_{ij}^{(l)}} = \frac{\partial L}{\partial a_i^{(l)}} \frac{\partial a_i^{(l)}}{\partial w_{ij}^{(l)}}. \tag{2.10}$$

The second derivative equals the transformed activation z from the nodes in the layer before that, or the inputs in the first layer x as can be derived from equations 2.2 and 2.4.

The derivative of the loss L with respect to the activations a at the output layer, will be different depending on which error function is used, but for the cross entropy functions will be found to be simply the difference of the predicted value and true value $\delta_i = y_i - t_i$ for the output layer, where δ is called the error[14]. The derivative of the activations with respect to weights can be found be differentiation function 2.4.

To compute the derivative of the loss with respect to activations in internal layers, we need to use the chain rule to break the derivative into sums, where M is the number of nodes.

$$\frac{\partial L}{\partial a_i^{(l)}} = \sum_M \frac{\partial L}{\partial a_M^{(l+1)}} \frac{\partial a_M^{(l+1)}}{\partial a_i^{(l)}}. \tag{2.11}$$

Here M are the number of activations in a layer. The errors in the hidden layer will then be found to be $\delta_i = h'(a_i) \sum_M w_{kj}\delta_k$ by differentiating equation 2.4.

So equation 2.10 can be written on the form $\frac{\partial L}{\partial w_{ij}^{(l)}} = \delta_j z_i$[15].

Using these derivatives, the updates of weights between layers in the neural network can then be performed.

When a neural network is trained, these processes of forward pass and backward pass will be done in alternation. One performance of forward pass and backward pass through a whole network is called an epoch.

2.2.5 Regularization

Regularization is used in deep learning to make models better at recognizing new inputs. Most regularization error sacrifice training error to try to lower the validation error [15].

If a neural network is able to model perfectly the relationship between the input variables and outputs within a training sampla, there is a chance of it simply being overfitting to the training data and that isn't certain to generalize as well to data which the neural network hasn't seen before.

Regularization methods attempt to sacrifice training error for improved generalization throuh a variety of different methods.

Ridge regression is one of those methods. In ridge regression a weight decay is added to the error function with a hyperparameter λ

$$\hat{L}(w) = L(w) + \frac{\lambda}{2}||w||_2^2 \tag{2.12}$$

The purpose of this is to force the predictions not to have to high of a variance and become simpler.

The hyperparameter gives control over how much we want to force the predicted model to be simple.

Another possible regularization method, especially applicable to image classification[15], is dataset augmentation. Increasing the amount of training data will improve the generalizability of a neural network, and augmentation is a simple way of creating fake data for training.

Dataset augmentation on images simply alters images in the training dataset in selected ways.

2.3 Weight initialization

In [15], weights are recommended to be initialized as small random values. Weights can be initialized from pretrained weights from another neural network[17]. Initializing weights in that manner is called transfer learning, and if done correctly can reduce convergence times of deep neural networks. Transfer learning can be described as a way of taking an educated guess on what values we expect the weights to have, instead of randomly initializing them. Convolutional neural networks who are trained to detect similar images can benefit well from that. Pretrained neural networks for image classification over a whole range of different classes such as the one in the COCO challenge[18], and transferring weights from those to different tasks but of a similar nature is likely to help. Transferring weights from inappropriate CNNs on the other hand would be counterproductive[17].

2.4 Convolutional Neural Networks

This section will start with a brief historical introduction on convolutional neural networks. A more detailed explanation about layers which can be included in a convolutional neural network architecture will follow.

Convolutional neural networks are a specific type of neural networks, generally used for data which can be represented as a grid. They can be used for 1 dimensional data such as time-series data. In this project they are used for images which form a 2 dimensional grid of pixels[15].

Nowadays, convolutional neural networks can be used effectively on image classification[6]. They can also learn to detect and classify local objects within images[19], as well as segmentation of local objects within images [10]. They are, for example, being used in the training of self-driving cars by Tesla[20].

One of the first attempt to try to map the activities of neurons in the visual part of the brain was done by David Hubel and Torsten Wiesel in a series of studies, beginning in 1959. The studies were made on the visual stimuli of cats by showing them different types of edges, shapes and forms, while measuring the response of the neurons in their brain. In following studies they made similar studies on the visual stimuli on other mammals. They discovered, amongst other things, that neurons in the visual cortex operate in a hierarchical order. There are neural cells in the brain which respond to light, other neural cells which respond to the orientation of light, and even other ones which respond to movement of light. For their research they were awarded the Nobel Prize in Physiology and Medicine in 1981[21].

Inspired by the discoveries made by Hubel and Wiesel, the Neocognitron was introduced by Kunihiko Fukushima in 1979, being the first model which included alternating layers of simple cells with modifiable parameters and complex pooling cells. However, it did not include a supervised learning algorithm[22].

Backpropagation and learning based on gradients were introduced to convolutional neural networks in 1998, when a convolutional neural network was trained to recognize digits of zip codes successfully[23].

It was only as recently as 2012 that the first monumental results for image recognition using convolutional neural networks were achieved by Krizhevsky's AlexNet, which performed better than other methods used in computer vision on the ImageNet classification challenge[6]. The AlexNet was designed by Alex Krizhevsky, then only a student at the University of Toronto. It was able to utilize both the abundance of data available on a digital format in the current era, as well as the computational capabilities offered by GPU's, which was only invented in 1999.

For a network to be considered a convolutional neural network, it is required to include at least a single convolutional layer[15]. Other layers frequently a part of convolutional neural networks are, a fully connected layer, nonlinear layer and a pooling layer. Generally, convolutional layers and nonlinear layers are used in alternation, with a pooling layer shrinking the image at a few intervals. To compute the output, a fully connected layer which maps all the weights into a long vector is used. An exemplary image of the effects these layers have on an image is shown in figure 2.8.

2.4.1 Convolutional Layer

Convolution, typically represented as shown in equation 2.13 is a mathematical function which outputs a function a representing how function I is changed by function k.

$$a(t) = (I * k)(t). \tag{2.13}$$

In the terms of convolutional neural networks, the functions will be an image I and a kernel filter k, which when convolved will create a feature map a[15].

Strictly speaking, convolutional neural networks use a function called cross-correlation, not convolution. Cross-correlation uses a flipped kernel filter relative to convolution.

For an image I and kernel filter k of two dimensions the convolutional equation will be of form

$$a[x_I, y_I] = (I * k)[x_I, y_I] = \sum_{w_k} \sum_{h_k} I(x_I + w_k, y_I + h_k) k(w_k, h_k). \tag{2.14}$$

where a is the output feature map, I is the image, k is the kernel filter, x_I and y_I are pixel locations within the image, and w_k and h_k represent the width and height of the kernel.

The convolutional layer is the first layer of a CNN, and shares it's name with this type of network. In it, filters are convoluted, or slid, over an image spatially, while dot products are computed at each spatial location. An input image I has a shape of

(w_I, h_I, d_I), where w_I, h_I and d_I represent the height width and depth of an image respectively. The kernel filter k has shape (w_k, h_k, d_k), where w_k is the width of the filter, h_k is it's height and the depth of d_k it is the same as the depth of the input image. The width and height of the kernel filter will be a lot smaller than the width and height of the image.

Each convolution with a filter over an image will produce a two dimensional feature map of shape $(w_I - w_k + 1, h_I - h_k + 1, 1)$. N different filters can be convoluted over the image, producing a feature map of shape $(w_I - w_k + 1, h_I - h_k + 1, N)$. For computational reasons, generally a number of filters which is a power of 2 is used.

The receptive field of each pixel location with a feature map will be the pixels in the respective location of the previous feature map of where the dot product has been taken.

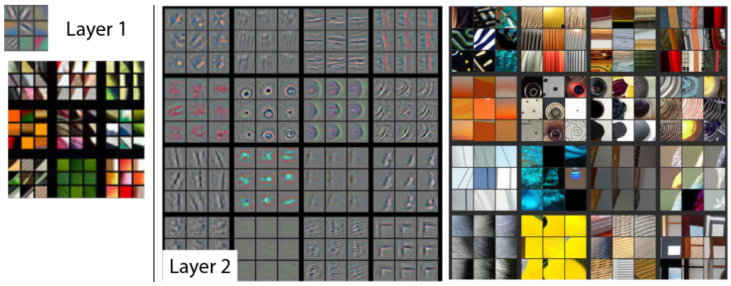


Figure 2.4: Visualization of filters and their effects during the first two layers of a convolutional neural network.[24]

A deep CNN will consist of sequences of these convolutional layers, with nonlinear layers and pooling layers in between. The deeper in a neural network architecture we reach, the receptive field of the activation map will cover a larger part of the original input image and more complex features of images will be detectable as shown in figure 2.4. At the first layer only simple features such as edges will be detected, but the at greater depths features will start to look more like complicated forms. That resonates well with the discoveries Hubel and Wiesel had made previously on the visual stimuli of cats[21].

Two hyperparameters are introduced in the convolutional layer. Firstly we can adjust the stride, S, of steps which the kernel filter takes between locations. If the stride is set as $S = 1$, a convolutional operation will be done at every single possible pixel. If the stride is assigned of a higher value, for example $S = 2$, we will skip every other possible pixel. Selecting a stride larger than 1 will cause the image to shrink and make the height and width of the feature map smaller. The mathematical relation between the output image and stride is as follows:

$$\frac{w_I - w_k}{S} + 1 = w_{out}, \tag{2.15}$$

where w_I is the width of the input image, w_k the width of the kernel filter, S is the stride and w_{out} is the width of the stride. The relationship is identical for the height of the output feature map. The dimensions of the output image must be an integer, so not all values of stride are acceptable.

The second hyperparameter introduced in the convolutional layer is padding P. Padding is used to avoid the shrinking of the feature map in the convolutional layer. A filter kernel is only convoluted over areas of the image where it fits, so if no padding is introduced the image will shrink even if a stride of 1 is used. And if shrinking is too rapid, it limits the number of convolutional layers the original image can be run through. By padding an input image, the dot product of the kernel filter and input image can be computed with a centre over all pixel locations of the image, excluding obviously the padded areas. Several versions of padding pixels can be used, such as zero-padding[15], reflection padding and replication padding, but zero-padding is the most commonly used[25].

$$\frac{w_I + 2 * P - w_k}{S} + 1 = w_{out}. \tag{2.16}$$

Equation 2.16 shows the relationship between input and output width when padding is involved, where P is the number of pixels the image is padded one direction. The relationship is identical for the height of the output feature map.

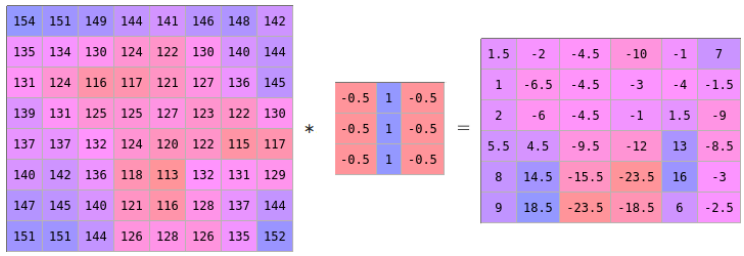


Figure 2.5: An example of the convolution operation, with pixel values from image **??** and a filter, done with a stride of 1.

2.4.2 Nonlinear layer

The nonlinear layer in CNNs has the same functionality as the hidden layers do in ANNs, introducing nonlinearities to ensure that the neural network won't simply be a linear classifier.

The *ReLU* function is most applicable for deep neural networks[15].

1.5	-2	-4.5	-10	-1	7
1	-6.5	-4.5	-3	-4	-1.5
2	-6	-4.5	-1	1.5	-9
5.5	4.5	-9.5	-12	13	-8.5
8	14.5	-15.5	-23.5	16	-3
9	18.5	-23.5	-18.5	6	-2.5

Figure 2.6: Example of what effects the *ReLU* function has on the convolved output from figure 2.5.

2.4.3 Pooling layer

A pooling layer takes as input an feature map of size (w_a, h_a, d_a), where w_a, h_a and d_a are the width, height and depth of the feature map respectively. Two hyperparameters are needed in this layer, the size of the filter k and the stride S. Usually, overlap will be avoided by configuring the values of filter and stride in that manner. They will be connected in a way such that the whole picture is pooled without overlap. The output dimensions from this will be a feature map of size $(w_{out}, h_{out}, d_{out})$, where $w_{out} = \frac{w_a - w_k}{S} + 1$, $h_{out} = \frac{h_a - h_k}{S} + 1$ and $d_{out} = d_a$.

Max-pooling is a frequently used version of pooling, where simply the maximum value of the are which the filter covers at each step will represent that area of the feature map. Other versions such as average pooling are also usable, but max-pooling better represent the extreme signals within a feature map[15].

1.5	0	7
5.5	0	13
18.5	0	16

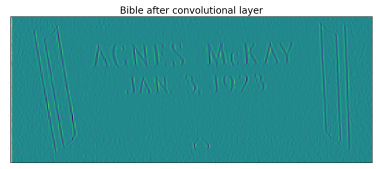
Figure 2.7: An example of what effects the max-pooling the feature map from figure 2.6 has, with a kernel filter of size 2 and stride of 2.

Reverse pooling layers also do exist, and are used, for example, in a Fully Connected Network, which Mask R-CNN utilizes to generate masks. The output dimensions from this will be a feature map of size $(w_{out}, h_{out}, d_{out})$, where $w_{out} = w_a * w_k$, $h_{out} = h_a * h_k$ and $d_{out} = d_a$. Reverse pooling can be done in several different ways. Nearest neighbor reverse pooling will fill areas of size $w_a * w_k$ and $h_a * h_k$ in the output feature map with the respective value in the original feature map. Bed of Nails reverse pooling will only keep the values of the original feature map, but fill around them with zeros[24][26].

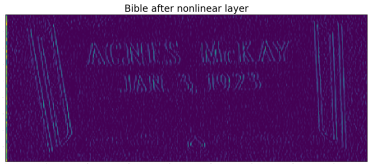
2.4.4 Fully Connected Layer

The fully connected layer in a CNN typically comes after all other layers. In a fully connected layer, the grid-like shape of the feature map will not be kept. The preceding layers will have gathered knowledge about features within an image, and the fully connected layer stretches the feature map of the connectin layer to a vector of shape shape $(w_a * h_a * d_a)$. This vector functions like an ANN, where all pixels in the grid form are now nodes, connected to output nodes via weights. The dot product of those weights with the vector will compute output values, which can be transformed to probabilistic values using equation 2.6.

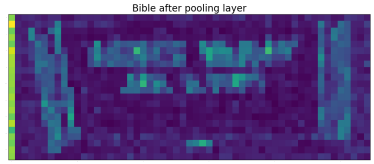
(a) Single channel image of a Bible with zero padding of 2 pixels in each direction.



(b) Same Bible convoluted with a filter of size three and stride 1.



(c) Same Bible after going through the non-linear layer using the $ReLU$ function.



(d) The Bible shown after max-pooling with filter size of 8.

Figure 2.8: Example of how an image of a Bible is altered while processing through the layers of a CNN. The filter which the image is convoluted with is shown in 2.5.

2.5 Region-based Convolutional Neural Networks

This section will go through the advancements made in convolutional neural networks with regards to object detection and semantic segmentation, touching briefly on relevant network models and architectures which have led to the development of the model used in this project, Mask R-CNN.

2.5.1 R-CNN

Before R-CNN was introduced, convolutional neural networks had already shown considerable capabilities in the field of image classification as demonstrated by the *AlexNet*[6].

An R-CNN, short for regions with convolutional neural network features, was introduced in 2014 as a CNN capable of semantic segmentation and object detection, a different challenge than image classification. It requires accurate localization of objects within an image to be able to classify them. It successfully managed to outperform more conventional computer vision methods which had previously been successful within the competition, such as various different implementations of SIFT and HOG. On the PASCAL VOC dataset from 2012, it achieved a mAP (mean average precision) of 53.3%, an increase of about 15% compared to what had been achieved before[9].

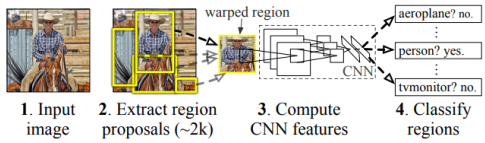


Figure 2.9: Model of R-CNN[9].

The proposed object detection system in R-CNN is composed of three separate parts. The first part of those generates region proposals within the image, such as the man and the horse seen in figure 2.9, which define a set of possible objects detectable. In [9], it is done by using a method called *selective search*, which returns 2000 regions of interest within the image. Due to possible regions of interest potentially being of vastly different shapes and sizes, all object proposals found are warped to a rectangle with a height and width of 227 pixels[9]. The second part is a convolutional neural network run over all of those object proposals and outputs a feature vector of 4096 dimensions. The third and last part of R-CNN utilized support vector machines trained for each class to calculate a score for each feature vector. To avoid classifying the same object many times, non-maximum suppression is used to accept or reject regions, based on their IoU overlap score. The IoU score simply calculates how much of the object is within the bounding box.

Although being a groundbreaking combination of deep learning and computer vision methods, this neural network is still very computationally expensive due to needing to run forward propagate for each object proposal[27].

2.5.2 Fast R-CNN

Fast R-CNN was built on the original R-CNN and published in 2015 by Ross Girshick, who had also been a part of the group which published the the prior. The training

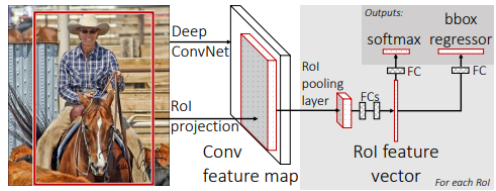


Figure 2.10: Model of Fast R-CNN[27].

model is changed so that the CNN forward pass is done on the whole input image in one step, instead of small parts of it.

The model takes an input image as well as object proposals, and outputs a feature map. Areas of the feature map which have object proposals are run through a RoI max-pooling layer, generated object proposal feature maps of fixed sizes. A RoI max-pooling layer works identically as a max-pooling layer described in chapter 2.4.3, but only over the area where the object proposal, or the RoI, is located.

Those RoI-pooled feature maps are then mapped to a feature vector, from where classification scores are computed using the softmax function (eq. 2.6) and bounding box regression scores are computed using a L1 loss[27].

Slight improvements in the mAP is noted compared to the R-CNN. A nine fold speedup of the training time was experienced compared to the forementioned R-CNN[27]. However, the training speed can still be sped up, since the bottleneck is now the region proposal[19].

2.5.3 Faster R-CNN

The paper describing Faster R-CNN was published in January 2016, by a team including Ross Girshick and Kaiming He.

Faster R-CNN, as the name implies, is a faster version of the previously mentioned Fast R-CNN. The bottleneck of the Fast R-CNN model was the time spent on object proposal, which Faster R-CNN improved on.

It does so by replacing the object proposal step with a Region Proposal Network (RPN). That RPN is merged with the Fast R-CNN model to create Faster R-CNN, and eliminating the bottleneck that region proposals had been [19].

Region Proposal Network

The RPN takes an input image, and predicts rectangular objects with an objectness score.

To the Region Proposal Network an image is input, and the network outputs rectangular object proposals as well as an objectness score. It shares convolutional layers with a Fast R-CNN network.

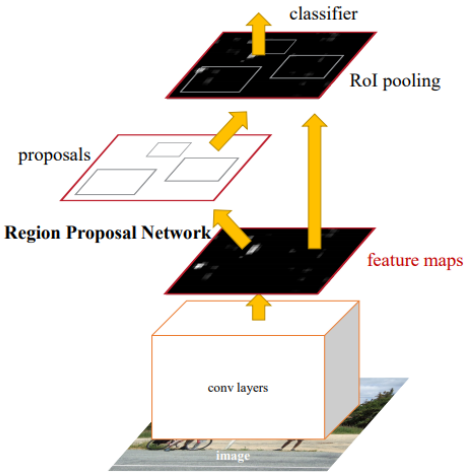


Figure 2.11: Model of Faster R-CNN[9].

On the feature map output from the last shared convolutional layer,a small sliding window is run. The feature map from every window is mapped to a feature vector, which is then run throuh two branched fully convolutional layers, one for the bounding box regression and another for the classification[19].

Within each sliding window, a lot of different regions are proposed. They are all rectangular regions with different scales and shapes. At most, N regions can be proposed. That means, that for each region we can get at most $4N$ bounding box outputs, for the 4 corners of each bounding box. The classification fully connected layer outputs at most $2N$ different proposals, with the probabilities of a proposal being an object, and another probability of it not being an object.

To train the Region Proposal Network, each of the k anchor boxes gets a binary class label of being an object or not being an object. Positive labels are given to anchors with either the highest IoU overlap with a ground-truth box, or if an anchor has an IoU overlap with a ground-truth box of more than 0.7. A negative label is given to an anchor if it has an IoU overlap with all ground-truth boxes of less than 0.3[19].

Training of Faster R-CNN alternates between region proposal and object detection[19].

The speedup achieved by this architecture is almost 10-fold compared to the previous Fast R-CNN[19].

2.5.4 Residual Network

Residual blocks were introduced in 2015 by a team includin Kaming He, to address a degradation problem on training accuracy that deep convolutional neural networks have. This problem is not caused by overfitting[7].

The residual blocks include a shortcut connection, a previous feature map is simply added via a shortcut to an output feature map computed by a series of layers.

The layers thus instead learn residuals functions with reference to layer inputs, $H(x) = F(x) + x$, instead of only $F(x)$. These modifications resulted in a decrease in both training and validation error when CNNs are made deeper, instead of an error increase, which had been apparent in previous structures[7].

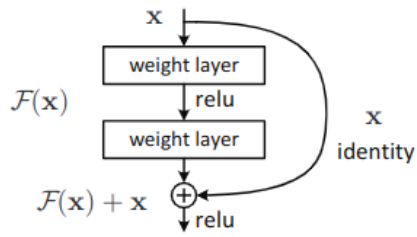


Figure 2.12: A residual block as introduced in [7].

2.5.5 Feature Pyramid Network

Feature pyramids networks were published in [28] in 2017 by a team including Piotr Dollár, Ross Girshick and Kaiming He.

Feature pyramid networks are used in deep neural networks, where outputs from every stage of the network are fed into the FPN. The outputs are taken from the last output of the last convolutional layer in a convolutional architecture before the feature map is downsampled.

At the deepest stages, features with high resolution will be detected, while at the shallowest stages, better information about localization of features will be available.

The feature maps are upsampled to the same dimensions of the shallowest feature map by using nearest neighbour unpooling previously described in 2.4.3.

These feature maps are then connected by skip connection with features in shallower layers. In [28] FPN is used with a Residual Network architecture. upsampled

2.5.6 Fully Convolutional Network

A fully convolutional network is a CNN which performs semantic segmentation mask generation. An input image is run through a series of convolutional, nonlinear and

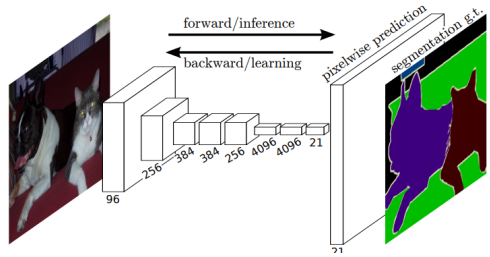


Figure 2.13: Model of a fully convolutional network.[26].

pooling layers to generate a fully convolutional layer, which, instead of being used for classification, which is upsampled again to form the shape of the original image.

Upsampling is not done by unpooling in this network, but by deconvolution. Upsampling by deconvolution is simply doing convolution backwards[26].

2.5.7 Mask R-CNN

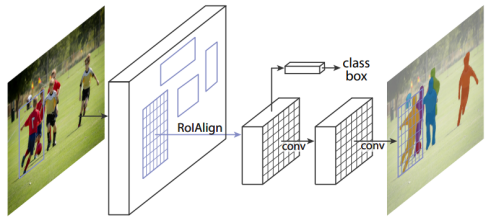


Figure 2.14: Architecture of Mask R-CNN[10].

Mask R-CNN was introduced in [10], published by Kaiming He, Georgia Gkioxari, Piotr Dollár and Ross Girshick in January 2018.

It builds on Faster R-CNN by adding a layers capable of instance segmentation. Instance segmentation combines both the task of object detection and semantic segmentation. Object detection classifies objects to different classes, while semantic segmentation classifies all pixels in an image to a belonging class.

In [19] the final branch fully convolutional layers predicted classes and bounding boxes on the proposed RoI, but now [10] adds a third segmentation branch. The segmentation mask output is a binary mask for proposed RoI.

To predict the segmentation mask, a Fully Convolutional Network is used n every RoI predicted by the RPN.

RoI-pooling, as used in [27] is replaced with a RoIAlign layer, which preserves object locations accurately. Quantization involved in RoI-pooling causes a misalign-

ment between the RoI used and features extracted. RoIAlign avoids quantization and greatly improves mask accuracy[10].

The Mask R-CNN is built up of two parts, a *backbone* and a *head*. The backbone takes care of the convolutional computations, while the heads perform classification, bounding-box regression and mask prediction[10].

In [10], a backbone architecture using both 50, and 101 layered ResNets is used, as well as an architecture combining a FPN and ResNet, with the FPN using feature maps from several different stages of the architecture.

The head architecture takes seperate input feature maps from the last layer of the convolutional network. One input feature map gets stretched to fully convolutional layers and used for classification and bounding box regression. The other input feature map has its grid like features kept, and is run throuh a series of convolutional and deconvolutional layers to predict a segmentation mask.

During training, the loss of the network is composed of a combination of the classification loss, bounding box loss and segmentation mask loss,

$$L = L_{cls} + L_{box} + L_{mask}[10], \tag{2.17}$$

where L is the loss.

CHAPTER 3

Methods

This chapter will introduce the configurations of the neural network used in this project. Computational and technical information will first be discussed. Following that, the process of scraping for the data used in the project will be detailed. Regularization methods used in the project will follow, and lastly the backbone of the network used will be explained.

Mask R-CNN is available implemented on Python 3, using Keras and Tensorflow on GitHub [29]. The model used in this project is unchanged from the version presented there, with a few adjustments made in the configurations. It was chosen as the optimal neural network for this project due to its recorded success at outperforms other existing models on all COCO challenges[10].

All work done in this project was done by using Python. When jobs were run on DTU's high performance computers, Python 3.6.2 was used. Python is currently ranked as the 4th most commonly used programming language according to Stack Overflow's user survey [30]. It's an open source programmed language, supported by various different modules which can be created by anyone.

Some highly used modules in this project are, Tensorflow and Keras through Mask R-CNN, BeautifulSoup was used for scraping webpages, Pandas was used for data manipulation and visualization. Imgaug was utilized for augmentations. Matplotlib and Seaborn were also used for visualization.

- **Tensorflow** is an open-source library written in Python, C++ and CUDA usable for various machine learning applications. It was developed by an internal team at Google and originally released in 2015[31]. Initially, tensorflow-1.5 was used, but great speedup was seen when switching to tensorflow-gpu-1.5, which helped training the neural network in a much shorter amount of time.

- **Keras** is an open-source library written in Python, used mostly on neural networks. It is run on top of Tensorflow, and is also capable of running on top of other well known data science libraries. Version 2.0.7 of Keras was used for this project[32].

- **Pandas** is a software library written in Python, Cython and C which allows user friendly data manipulation[33].

- **BeautifulSoup** is a module written in Python, which allows parsing `html` and `xml` documents[34]. That makes extracting HTML data, which has been done extensively throughout this project, manageable.

- **imgaug** is an image augmentation library which also augments segmented masks accordingly[35].

The Bash[36] shell in Linux was used to run programs straight to the terminal on remote high performance computers using Bash cripts. The Mask R-CNN to detect religious symbols was trained in DTU's high performance computational cluster. It was done on a GPU called Tesla V100 32 GB, callable with queuename `gpuv100` through a bash script. For scraping, 4 scripts were run simultaneously on a CPU for 72 hours at a time, callable with queuename `hpc` through a bash file, which allowed gathering of data from approximately 130 thousand memorials in 24 hours. Addresses for images were stored and downloaded seperately.

3.1 Data gathering

This section will go through the processes involved in webscraping and data gathering of the project.

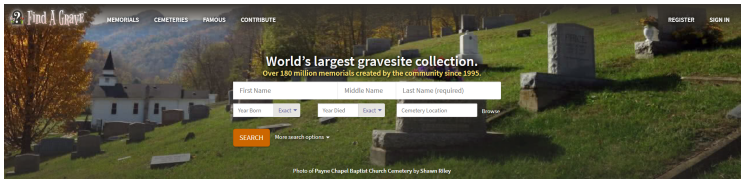


Figure 3.1: The opening header on `https://www.findagrave.com/`.

All data gathered and used in this project can be found on `https://www.findagrave.com/`. It is an online gravestone collection, with claims of being the biggest online gravestone memorial site online. The website includes memorials in remote locations such as North Korea and Samoa. Alternative memorial websites do exist, such as `http://findengrav.dk/` in Danish and `https://www.gardur.is/kirkjugardar.php` in Icelandic, but were not considered for this project.

The website claims to have over 180 memorials stored. The quality of memorial pages vary greatly, some memorials have been removed, some memorials do not have an image, and some lack information on age. Around 40% of memorials scraped were found to be missing images, and were excluded. Figure 3.2 shows two image examples from the website. Memorials on the webstite can be accessed through the address `https://www.findagrave.com/memorial/ID`, where `ID` simply needs to be configured to be an integer between 1 and approximately 200 million.

Memorial pages have a variety of interesting information. The header includes information about the location of the memorial, information about the date of birth and death, as well as geographical locations of birth and death. Memorials are assigned an identification number (`ID`), which is also included within the header. Below the header, further information is often included. Information about family relations and commemorative text can sometimes be seen, as well as more images. The images shown below the were not scrapeable by using only BeautifulSoup, because they are not included in the raw `HTML` code, but included in the page using `AJAX`, a JavaScript technique. Due to the complications proposed by that, it was decided to only scrape header images.

Even though the website has memorials from all over the world, randomly scraping returned by far the most data from the English speaking world. It does have memorials from all across the globe, but random scraping found that memorials outside of the English speaking world were very sparse.

(a) Memorial plaque, covered in parts by dirt.

(b) Memorial headstone showing an image of a person and a bald eagle.

Figure 3.2: Exemplary images from `https://www.findagrave.com/`.

Webscraping was done in two different parts, with each part having two steps. First, a set of 6025 images, which had previously been manually labelled, was scraped, returning all necessary statistics from the website in a spreadsheet. Then, the links to images, included in the spreadsheet, were all downloaded. These images amount to 6.7GB's of data. Images from that subset were used for network training.

The next part was a random webscraping of a large amount of data. A total of 693953 images were scraped over a course of several days, amounting to 716.0GB's of data. Webscraping proved to be a bottleneck during the project, and the highest speed which scraping was able to be done at was scraping information from 400

thousand memorials in 72 hours. Information was gained from 2918 of the 3142 counties within the United States, as well as information from 129 other countries or territories. 93% of data scraped is from the United States. Images in this subset were used for inference.

The spreadsheets storing the data had columns for: Memorial ID, image link, birthdate, deathdate, birthplace, deathplace, cemetery and the city, county, state and country of the cemetery.

3.2 Data handling

This section will go briefly over what modifications were made on the data.

A lot of randomly scraped data was found to be lacking exact information about birth and death. Some memorials had either the birthyear or the deathyear missing, and those were removed. Other memorials had the birthyear and deathyear scraped, but did not have more exact information on the birthdate. If birth- or deathmonths were found missing, a month of January was assigned. If birth- or deathdays were found missing, a day of 1 was assigned.

The lifespan of each memorial had to be computed. That was done by using theinformation on birthdate and deathdate. Not all information on the website was accurate, and to handle some inaccuracies, memorials with a recorded lifespan of more than 123 years were excluded[1].

Memorials from the United States were assigned a FIPS[2] county code according to the cemetery county they were located in. The religious adherency index used for the project listed counties by their FIPS code[5], and it aided with geographical representation of the data as well.

Memorials scraped from the countries of England, Scotland, Northern Ireland and Wales were assigned a unified country of the United Kingdom.

3.3 Image segmentation

This section will detail how images were segmentated.

To train a neural network such as this, a lot of training data is required. It was decided to train the neural network to be able to detect 6 different classes of religious symbols, as well as an extra class for the cases where an image of a person has been scraped as well as the background class. Those classes were; angels, bibles, crosses, praying hands, doves, David's star and a person. Examples of those can be seen in figure A.1 in the Appendix.

CNNs used for localized segmentation need when training knowledge about the true objects located within an image. The symbols in the images were segmented

[1] The oldest person ever according to Wikipedia reached an age of 122 years and 164 days.
[2] Federal Information Processing Standards

using VGG Image Annotator[37], an online image annotator which supports manual segmentation. The image annotator outputs a JSON file with locations of the segmentations as well as the name of the class they belong to.

Figure 3.3: Example of an annotated image, with two annotated Bibles and two annotated praying hands.

Figure 3.3 shows an image with segmented and annotated symbols. Engravings which the neural network is not being trained to recognize, such as the rings in figure 3.3, are not segmented. In addition to that, symbols on gravestones in the background are not segmented.

In total, 1240 images were segmented for training purposes, of which 972 were assigned to a training set and 268 to a validation set. Initially, a much lower number of images had been segmented.

3.4 Configurations

Here follows a description on the configurations of the network during training and inference. Following that the backbone network and heads of the Mask R-CNN will be articulated. Regularization methods used will be shown.

The training steps and validation steps in one epoch were set to be equal to the number of images within the training data or validation data, multiplied by the number of augmentations made. Training was done on a GPU with a memory of 32GB. When training all layers of the network, it was possible to assign 4 images at each training step, thus resulting in 1458 training steps per epoch. Images are not augmentated during validation. Validation steps were 68.

All images presented to the network were scaled and zero-padded to a square image with height and width between 800 and 1024 pixels. Only images with 3 channels were able to be used, so a very few number of images were thrown out due to having an invalid depth. All images are adjusted in a way that the mean pixel value of each channel is 123.7 for the first channel, 116.8 for the second channel and 103.9 for the third channel.

The anchors scales used within the RPN were all powers of two between 32 and 512. There were three anchor ratios used, 0.5, 1 and 2. All anchors were rectangular. To avoid multiple detections of the same object, a non-maximum suppression threshold of 0.7 IoU was used. Each image had 256 anchors trained within it.

Stochastic gradient descent with momentum is used to optimize the weights, with a learning rate η 0.001 and momentum α of 0.9, as is done in [29].

Weights were initialized with weights from a Mask R-CNN pre-trained to detect between the 82 different classes of the COCO dataset[18]. Initializing weights in that manner utilizes a method called transfer learning, where weights pre-trained for a task in a similar manner to our own already performs at an adequate level[38].

Initial attempts were made to only train the heads of the networks, but results from that were not promising. For example, detecting the difference between a Bible and a rectangle seemed to be almost impossible when only training the heads. That is due to how similar the symbols look after going through max-pooling layers. When training all the layers of the network, those problems evaporated. A minimum confidence for a detection was set to be 0.8 during training and validation. The final training system was to, start training only the heads of the network for 20 epochs, follwing that the whole network was trained for 40 epochs, and then an additional 20 epochs were trained on the heads again.

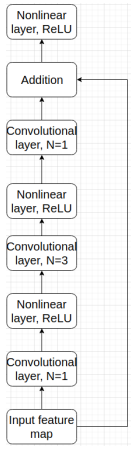
3.4.1 Backbone network

As mentioned in 2.5.7, the Mask R-CNN model is split into two parts, a backbone and heads.

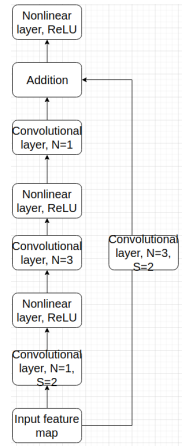
The backbone of the network is a 101 ResNet of 101 layers, composed of five stages. Two types of residual blocks are used to compose it. One is an identity block, where the skip connection doesn't have any layers on it, and the initial input feature map is simply added to the output feature map after going through a series of convolutional and nonlinear layers. The other is a convolutional block. Its difference compared to the identity block is that the skip connection includes a convolutional layer. These two blocks are shown in figure 3.4.

The first stage of the ResNet architecture takes an image as an input, zero-pads it 3 pixels in, and is then run through a convolutional layer with 64 different filters of shape (7×7) and with a stride of 2. After the convolutional layer it goes through a nonlinear layer using the ReLU activation function introduced in chapter 2. The first stage is concluded with a max-pooling layer with a filter size of 3 and stride 2. The output feature map is assigned as $C1$. The next stage continues on feature map $C1$, and runs it through one convolutional block, with stride of 1, not 2 like shown in figure 3.4b. That feature map is then run through two identity blocks. That concludes the second stage and returns an output feature map as $C2$.

The third stage continues on feature map $C2$, and runs it again through one convolutional block, now exactly like shown in figure 3.4b. Then it is run through three identity blocks, and returning an output feature map of $C3$.



(a) Identity block used in Mask R-CNN.



(b) Convolutional block used in Mask R-CNN.

Figure 3.4: Flowcharts showing the layers included within the two residual blocks used in Mask R-CNN.

The fourth stage continues on feature map $C3$, and runs it again through a single convolutional block like in stage three. After that, it is run through 22 identity blocks, returning an output map of $C4$.

The fifth and final stage continues on feature map $C4$. It is identical to stage two, except the convolutional block is like shown in figure 3.4b. Then we return an output map of $C5$, now with a dimensionality of 2048.

The feature pyramid network takes as input pooled regions of interests, as well as the feature maps output of the ResNet at stages 2, 3, 4 and 5, $C2$, $C3$, $C4$ and $C5$, which will now be labelled as labelled as $P2$, $P3$, $P4$ and $P5$. Those feature maps will be upsampled to match the shape of $C2$. It also takes inputs of a pooling size and class numbers, returning class logits, probabilities, bounding boxes and masks. The functions for the feature pyramid network are split into two parts, a classifier and a mask builder.

The region proposal network is input with a feature map, as well as anchor scales, ratios and strides. There are 5 anchor scales used, 3 anchor ratios and 1 anchor stride, creating a total of 15 different anchors at each location. It outputs anchor class logits, class probabilities and bounding boxes. A total of 256 anchors are trained for each image.

So in full, during training of the layers, feature maps are generated by using a residual network, and then build a feature pyramid networks. Along with that FPN are generated anchors, which are used on region proposals in the RPN. Regions of interests are proposed using RoIAlign[10] on levels of the feature map.

3.4.2 Regularization

Two regularization methods are used in this project. L_2 regularization is used with a weight decay of $\lambda = 0.0001$ as initialized in [29], and augmentations are used to increase the dataset available.

Augmenting images for training neural network is a commonly used regularization method to improve the generalization[15]. Although the training set is composed of almost 1000 images, getting more data for training is extremely time consuming, due to the hours needed to segment the data. Some classes of the religious symbols were also very hard to find, and even from a subsample of 15000 images less than 200 gravestones with a dove were successfully found.

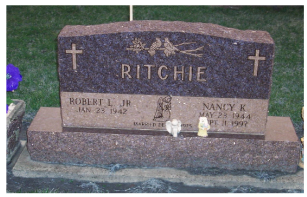
While training, several different augmentations were made available for each image. Adding to the original representation of the image, 5 additional images were generated by augmenting in one manner each time, with seven different methods available. Augmentations were dune using the `imgaug` library[35].

The augmentational methods available during training were:

Method:	Parameters
Horizontal flip	
Vertical flip	
Piecewise affine transformation	$s \in (0.01, 0.05)$
Gaussian blur	$\sigma \in (0, 0.5)$
Dropout	$p \in (0.05, 0.1)$
Grayscale	$\alpha = 0.5$
Canny	$\alpha \in (0.1, 0.3)$

Table 3.1

Horizontal and vertical flipping is done to replicate images on the website shown incorrectly. Piecewise affine transformation skews a representation of an image. Gaussian blur is used to replicate images with lower pixel quality. It artificially blurs images using a normal distribution, where s is the standard deviation. Dropout drops a percentage of pixels p out randomly. Canny edge detector slightly highlights edges in the image, is an edge detection algorithm used in computer vision. The image is also made grayer to replicate grayscale images which there might be some of on the website. All augmentation types are shown in figure 3.5



(a) Original image.

(b) Horizontally flipped.

(c) Vertically flipped.

(d) Piecewise affine transformation.

(e) Grayed out image.

(f) Slightly blurred image.

(g) Pixels dropped out.

(h) Canny edge detection.

Figure 3.5: All augmentations available during training of the network. Figure 3.5a is the original image. Figures 3.5b and 3.5c are horizontal and vertical flipped versions of it. Figure 3.5d shows the image piecewise affine transformed. Figure 3.5e shows the image with grayer colours. Figure 3.5f shows the image with a slight Gaussian blur. Figure 3.5g shows the image with pixels dropped out. Figure 3.5h shows the image with Canny edge detection effects.

CHAPTER 4

Results

Results of the project will be presented in three sections. First, a general overview of the performance of the network will be shown. Following that, results from inference over randomly sampled data will be presented in two sections. The prior will solely focus on data sampled from the United States. The latter will focus on data sampled from the rest of the World.

4.1 Network performance

This section will present losses of the network, while in training. Performance on the validation set will be shown, and a comparison with manually labelled data will be shown.

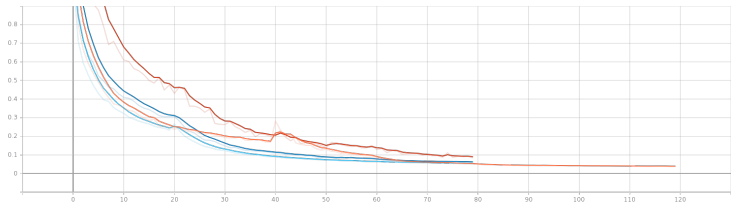
The network was trained several times in different ways over the course of the project, always initialized with the same weights. The lowest loss on the validation set was achieved by using the augmentations described before, and was trained in December 2019, after a set of segmented images was received. The validation loss at the lowest point was 0.7077.

The training and validation sets did not include equal amounts of segmented symbols from each class. The network, which was trained to detect 8 different symbols, had the most amount of crosses and Bibles segmented. An attempt was made to have the distribution of segmented classes equal, but that was done by selecting equal numbers of pictures which had each symbol engraved. Multiple engravings of crosses and Bibles on gravestones were frequent, and that led to them having a greater representation in the training and validation sets.

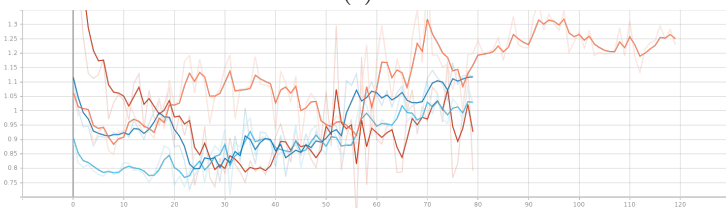
The training set included 147 instances of angels, 340 instances of Bibles, 394 instances of crosses, 112 instances of David's star, 171 instances of doves, 42 instances of people and 199 instances of praying hands.

The validation set included 29 instances of angels, 81 instances of Bibles, 100 instances of crosses, 20 instances of David's star, 22 instances of doves, 15 instances of persons and 61 instances of praying hands.

The worst performance on object detection on images in the validation set was for the class of doves, where almost a third of detections were false positives. The only perfectly performing class is the one with persons. That is due to how the weights are initialized, with weights from a well trained Mask R-CNN which detects classes of the COCO dataset[18]. In the COCO classification dataset, there are a



(a)



(b)

Figure 4.1: Figures 4.1a and 4.1b show the training and validation losses of 4 runs. Orange represents losses for a run only done when training heads. Dark blue represents losses when done with a third of the total training data. Maroon represents the losses when no augmentation is used. Light blue represent the losses with augmentations and using all training and validation data. Weights after 32 epochs from the ligth blue training were used for inference.

total of 82 classes, of which one is a class for persons. So, even when training the network on a different dataset, it still generalizes really well over a class it is already trained for. The difficulties in detecting doves stems from two reasons. Firstly, the variety of different birds appearing on gravestones, such as colibri birds, finches and the bald eagle, is a challenge. An example of this can be seen in figure 4.2. Floral depictions engraved on the gravestones also pose a challenge, especially for the angel and dove classes due to the variety of appearances those symbols can take. The floral engravings are thus often misclassified as a dove or an angel.

The mean average precision, or the average of correctly classified symbols in images, over the whole validation dataset was found to be be 0.825. Further numbers of performance of the model on the validation set can be seen in table 4.1. Exemplary detections can be seen in figure A.2.

To further evaluate how the model was performing, a comparison was made with 5092 manually labelled images. Those images are also collected from the same website. A slight caveat is that the classifications do not always have a chance of accurately matching due to the inability of webscraping all images on each memorial on `https://www.findagrave.com/`. The manual classification is made on the most appropriate image on each memorial site, while webscraping was only able to gather the header image. That image is, in most cases, an image usable for the project. However, some of the memorials have an image of a person and those memorials were excluded from

(a) (b)

Figure 4.2: Figures 4.2a and 4.2b show the challenges posed to the network. A bird, possibly a finch, has been classified as a dove with a 99% certainty. So has a sitting person around bushes been classified, albeit with a lower certainty.

	True detection:	False detection:	Undetected:
Angels	27	4	2
Bibles	78	4	3
Crosses	95	7	5
David's star	17	1	3
Doves	21	10	1
Persons	15	0	0
Praying hands	59	6	2

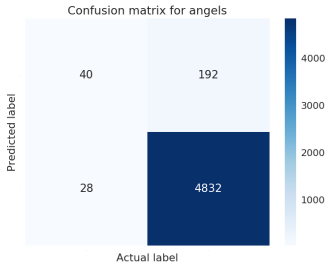
Table 4.1: Detection stats for the 7 symbol classes. The highest percentage of false detections is for doves, with 32% of detections of doves being false. Persons are detected perfectly. Bibles have the highest percentage of accurate symbols out of the religious symbol classes, with 95% of detections for Bibles being true.

classification. Some images also had an image with religious symbols within a part of the site which was not scraped, while the header either showed no such symbol or a different one.

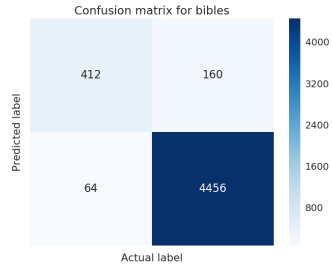
Comparisons of the performance of the network versus the performance of manual classification was also made for the all religious symbols the neural network was taught to recognize, excluding the Star of David, which did not have a comparable manual classification. Confusion matrices can be seen in figure 4.3.

The accuracy for all classes as well as the absolute class of a religious symbol

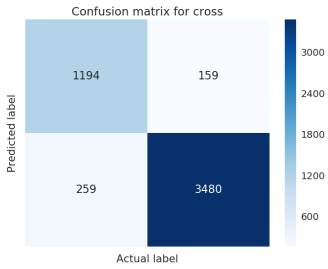
being present is higher than 84%. The accuracies of predicting each separate symbol is above 90% for each class. However, when detecting angels and doves, a fairly high number of false positives are obtained, dragging the precision down to around 17% and 20% for those classes respectively. However, those symbols appear lot less frequently than other classes.



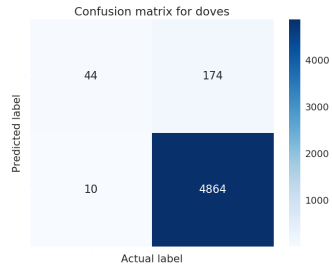
(a) Confusion matrix of angel detections.



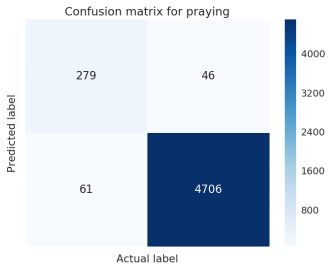
(b) Confusion matrix of Bible detections.



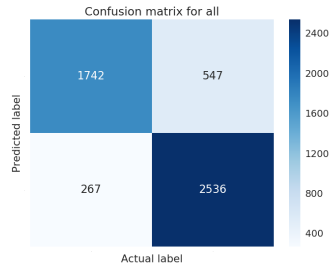
(c) Confusion matrix of cross detections.



(d) Confusion matrix of dove detections.



(e) Confusion matrix of praying hands detections.



(f) Confusion matrix of religious detections.

Figure 4.3: Confusion matrices for comparisons of manually classified symbols and symbols classified by a neural network. The accuracy for all classes as well as the absolute class of a religious symbol being present is higher than 84%. The accuracies of predicting each separate symbol is above 90% for each class. However, when detecting angels and doves, a fairly high number of false positives are obtained, dragging the precision down to around 17% and 20% for those classes respectively.

4.2 Statistics from sampled data

In this chapter, results from inference over randomly sampled test images using the weights with the lowest validation loss will be presented. First, a small overview over how the data is grouped together according to the religious adherency index [5] will be explained, as well as a visualization of where the sampled data is from.

Distributions of ages recorded for religiously and non-religiously sampled memorials for each religious adherency percentage group will be visualized in a boxplot. The whole distribution of samples from two of the religious adherency groups will be shown. The difference between the mean lifespan of the religiously classified memorials and the non-religiously classified memorials will also be visualized.

Lastly, a table with the amount of classified religious symbols within each religious adherency group will be shown.

4.2.1 United States

The network was run over memorials from the United States separately from other memorials and results from that will be shown here.

The network was run over randomly sampled data from `https://www.findagrave.com/`. Scraping the website by random sampling for a few days returned in the end a total of just over 587 thousand usable samples from the United States. Random sampling from the United States mostly returned samples in correlation with population, with 6223 samples coming from Los Angeles County in California, 5176 samples from Cook County, where Chicago is, and 4321 samples coming from Allegheny County in Pennsylvania, where the city of Pittsburgh is located. Los Angeles County and Cook County are the two most populous counties in the United States. Four counties only have 1 samples, Southeast Fairbanks Census Area in Alaska, Harding County in New Mexico, Reagan County in Texas and Ziebach County in South Dakota. All those counties have a current population of less than 10 thousand people. The distribution of counties with religious adherency now goes up to 146%, with the City of Fredericksburg in Virginia being the most religious county and Sanpete County in Utah with a religious adherency rate of less than 1%.

The religious adherency rate numbers are gathered from [5]. It might seem odd that religious adherency rate can go over 100%, but the reason for that is simply that people might go to church in a county which they do not reside in. The counties in the United States were grouped together by their religious adherency rate. Counties with an adherency rate between 0 and 10 percent were assigned to group 0, counties with an adherency rate between 10 and 20 were assigned to group 10, and so on.

The number of religiously classified memorials in the United States were 193624, while memorials classified as not religious were 393386.

The median age of all memorials which the neural network has attempted to detect religious symbols on is 70.0. For memorials which have had a religious symbol detected, the median age is 71.55 and for memorials which have not had a religious

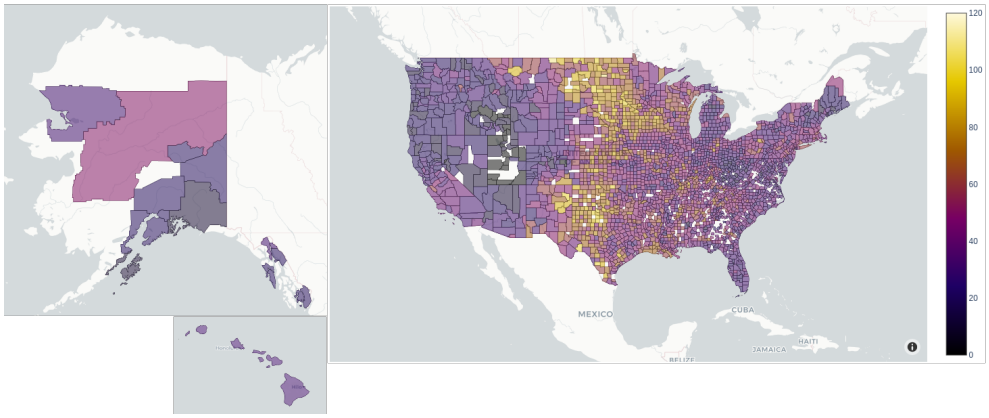


Figure 4.4: Religious adherency grouping of counties in the United States which had samples retrieve during scraping.

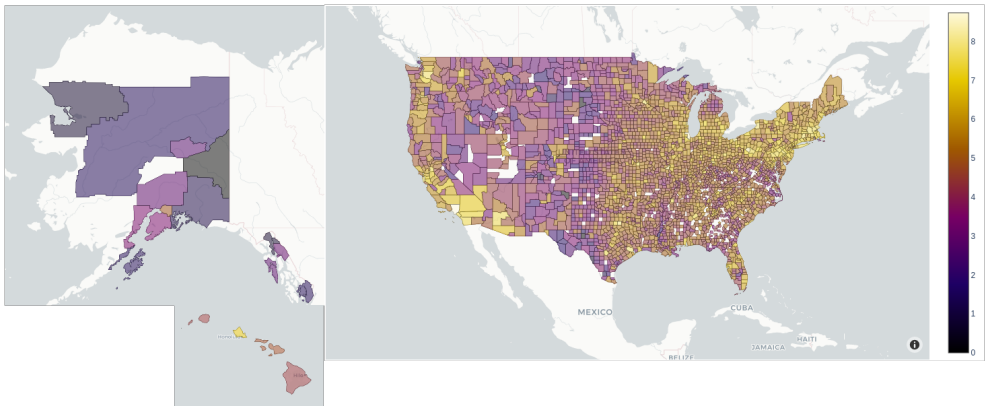
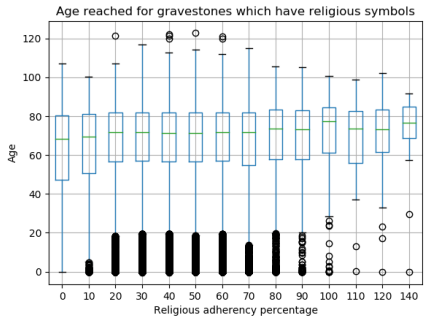
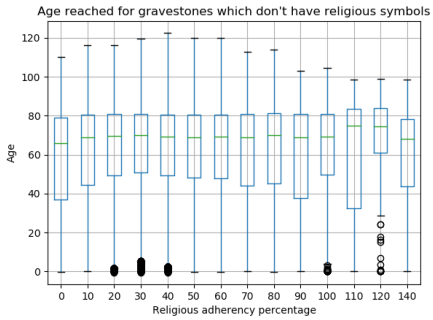


Figure 4.5: Number of samples scrapes within each geographical area in the United States, visualized logarithmically.

symbol detected, the median age is 69.44. The year of death for memorials in this dataset take a value between 1637 and 2019, with a median of 1967.



(a) Age distribution of religiously classified people in the 100 group (Blue) and 10 group (Orange).



(b) Age distribution of non-religiously classified people in the 100 group (Blue) and 10 group (Orange).

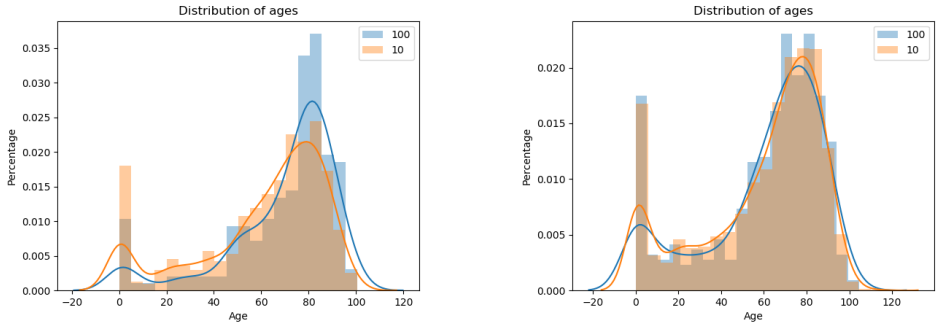
Figure 4.6: Figure 4.6a shows the age distribution for all religious adherency deciles which have sampled data classified as religious. A slight increase in the median age can be seen for counties within the 90 decile and 100 decile. Figure 4.6b shows the age distribution for all religious adherency deciles which have sampled data as not religious. A slight increase in the median age can be seen within the 100th and 110th decile, but otherwise the age distribution seems pretty uniform.

Figure 4.6 shows the distribution of religiously classified memorials (fig. 4.6a) and memorials not religiously classified (fig. 4.6b). The boxes in the plot cover between the lower quartile of $Q1 = 25\%$ and $Q3 = 75\%$ and the green line represents the median of the distributions. Outliers, shown with a black point, are points outside of the inquartile range defined as $IQR = 1.5 * (Q3 - Q1)$ [33]. The whole distributions shown in a histogram for all religious adherency groups can be seen in figures A.3 and A.4.

Figure 4.7 shows the full distributions of lifespans of memorials classified as religious (fig. 4.7a) and not religious (fig. 4.7b) of memorials sampled from counties within the 10 group and 100 groups. The distributions are bimodal. That might be due to no considerations being taken for the year of death of the memorials. Infant mortality has reduced by 93% since 1919 [39].

$$\mu(\text{age}_{\text{diff}}) = \mu(\text{age}_r) - \mu(\text{age}_{nr}) \tag{4.1}$$

Figure 4.8 shows the difference of the mean lifespan of memorials of religious gravestones and non religious gravestones, computed using equation 4.1. The red whiskers shown represent the standard error of the mean of the whole distribution, defined as:



(a) (b)

Figure 4.7: Figure 4.7a shows the distribution of samples within the 100th and 10th deciles of religiously classified samples. A difference is distinguishable for the distributions. Figure 4.7bshows the distribution of samples within the 100th and 10th deciles of non-religiously classified samples. A difference is hard to distinguish for the distributions.

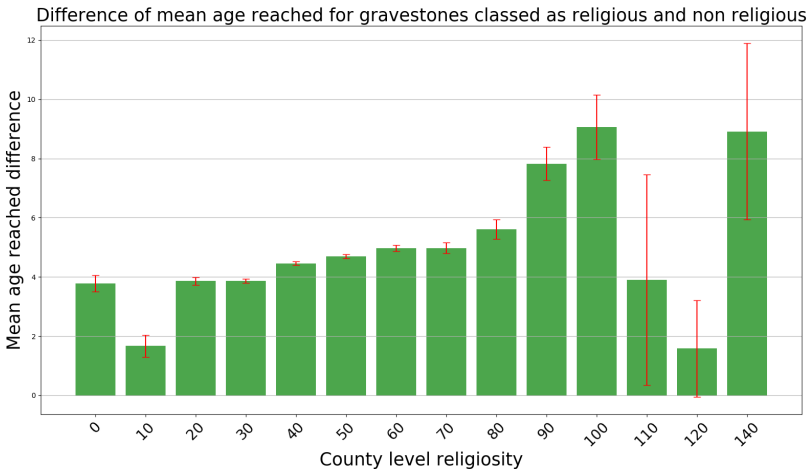


Figure 4.8: Difference of the mean age reached of memorials within each religious adherency group. A clear increase can be seen from religious adherency groups 0 to 100.

$$\sigma_x = \frac{\sigma}{\sqrt{N}}, \tag{4.2}$$

where σ is the standard deviation of the sample and N are the number of samples.

Correlation between lifespan and religious adherency group of memorials were calculated using Pearson's correlation coefficient, defined as

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}, \tag{4.3}$$

where ρ is the correlation of variables X and Y and cov is the covariance and σ is the standard deviation. The correlation was calculated separately for religiously classified memorials and not religiously classified memorials.

The correlation of the mean lifespan of religiously classified memorials with the religious adherency group is 0.77. The correlation of the mean lifespan of non-religiously classified memorials with the religious adherency group is 0.36.

Decile	**Cross**	**Bible**	**David**	**Praying**	**Angels**	**Doves**	**Religious (%)**	**N**
0	11.82	16.22	0.99	5.04	3.75	3.93	29.22	11135
10	12.62	10.40	1.22	1.86	5.15	3.48	26.61	5636
20	19.98	9.19	1.52	2.68	3.08	2.78	30.17	38735
30	22.97	9.08	1.82	3.43	3.20	2.77	32.80	127911
40	23.37	9.91	1.73	3.65	2.94	2.74	32.87	174632
50	23.24	11.41	1.59	3.62	2.83	2.71	33.11	133878
60	26.20	12.23	1.66	3.99	3.05	3.09	35.62	61782
70	26.68	13.20	1.01	3.86	2.71	3.03	34.82	22875
80	26.06	13.00	0.80	3.25	2.75	2.36	34.20	6992
90	26.00	14.08	1.25	3.22	2.66	2.62	34.21	2485
100	19.08	15.46	1.15	5.10	2.14	2.47	31.74	608
110	14.49	8.70	1.45	1.45	2.90	0.00	27.54	69
120	18.63	6.37	0.49	2.94	1.47	2.45	24.02	204
140	14.71	16.18	0.00	2.94	0.00	0.00	22.06	68

Table 4.2: Number of appearances of each symbol per 100 gravestones in each religious adherency group. The distribution of symbols within each area are not uniform, with a difference of 25 between the religious adherency group with the fewest crosses and most crosses.

The number of appearances of each symbol is not uniform. The least religious counties in the US have had a lot of Bibles and praying hands detected, while crosses have been detected most in the mid to high range of religious counties. The highest percentage of religiously classified are from the US counties in the 60th decile of religious adherency. The full distribution can be seen in table 4.2.

4.2.2 Rest of the World

Data gathered from other countries than the United States will be presented here.

Randomly scraping `https://www.findagrave.com/` returned 47359 samples from countries other than the United States. A total of 129 different geographical entities had samples returned, but a lot of them had extremely few memorials. Countries which had fewer than 500 memorials were excluded from calculations.

Only 7 countries had more than 500 memorials webscraped. A total of 42.472 memorials were sampled from these countries. Table 4.3 shows the total number of samples from every country with more than 500 memorials as well as their religiosity level.

Country-level religiosity used in [1] was used as a metric on the religiosity of the respective countries. Philippines have by far the highest religiosity, with a value of 3.27, while other countries have a much lower religiosity value.

The study which these values are based on were done by collecting data online in a study between 2001 and 2009. The study involved scales ranging from 1 to 5, with 1 labelled as strongly disagree and 5 as strongly agree. The country-level religiosity is then calculated as the average personal religiosity within each country [1].

The number of religiously classified memorials in these seven countries were 10397, while memorials classified as not religious 32075

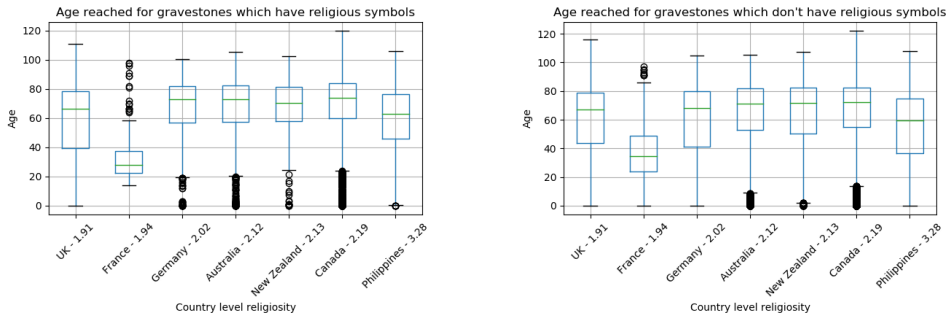
Country	**N**	**Religiosity**
Australia	6012	2.12
Canada	23346	2.19
France	792	1.94
Germany	2389	2.02
New Zealand	1246	2.13
Philippines	892	3.27
United Kingdom	7795	1.91

Table 4.3: Countries with more than 500 samples returned, showing the number of samples N as well as their country-level religiosity.

The median age of all memorials which the neural network has attempted to detect religious symbols on is 70.58. For memorials which have had a religious symbol detected, the median age is 72.57 and for memorials which have not had a religious symbol detected, the median age is 70.24. The year of death for memorials in this dataset take a value between 1135 and 2019, with a median of 1900.

The whole age distributions shown in figure 4.9 do seem to be really hard to decipher. Data from France had a lot of memorials with deaths from the first and second worlds war, located in specific war cemeteries. Memorials from such locations are expected to have a low lifespan. Histograms of the whole distributions from these 7 countries can be seen in figures A.5 and A.6.

Figure 4.10 shows the difference of the mean ages of religious and non religious



(a) (b)

Figure 4.9: Figure 4.9a shows the age distribution for religiously classified memorials per country. Figure 4.9b shows the age distribution for non-religiously classified memorials per country. The age distribution from France really stands out, with a median age of 32.8 for all samples.

data, computed using equation 4.1. It does seem to show a higher difference for countries with a higher country-level religiosity value, barring Germany. The red whiskers are the standard error of the means, computed using equation 4.2.

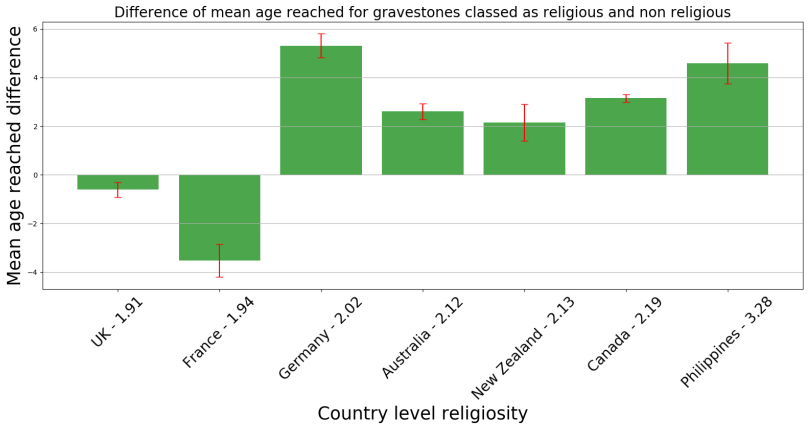


Figure 4.10: Difference of the mean age reached of memorials within each religious adherency group.

The correlation, computed using equation 4.3, of the mean lifespan of religiously classified memorials with the country-level religiosity is 0.13. The correlation of the

mean lifespan of non-religiously classified memorials with the country-level religiosity is -0.00.

Even though the correlation computed here is lower than reported from data from the United States, there is a higher correlation for religiously classified memorials than non-religiously classified memorials.

Country	**Cross**	**Bible**	**David**	**Praying**	**Angels**	**Doves**	**Religious (%)**	**N**
United Kingdom	9.51	0.64	4.27	0.26	0.74	1.68	14.61	7674
France	23.80	0.25	2.03	0.13	0.38	3.16	16.96	790
Germany	41.90	0.47	3.47	0.85	1.23	1.35	39.07	2365
Australia	19.62	1.99	1.74	0.34	2.01	1.74	22.17	5927
New Zealand	9.50	3.00	1.46	0.08	0.89	1.46	15.02	1232
Canada	20.81	6.69	1.68	2.54	3.95	4.43	28.06	22869
Philippines	29.52	2.24	1.01	2.47	1.80	6.96	33.89	891

Table 4.4: Number of appearances of each symbol per 100 gravestones in each country. A big difference in commonly appearing symbols can be noted.

Table 4.4 shows how many times each symbol appears per 100 gravestones in each countries. The difference of commonly appearing symbols are stark. Gravestones in the dataset from Germany have almost 42 crosses per 100 gravestones, while for the United Kingdom that rate is less than 10. The Bible and praying hands do not seem particularly popular in Europe.

CHAPTER 5

Conclusion and Future Work

The target of this project was to be able to utilize Convolutional Neural Networks to explore patterns of religiosity. That was done by training a Mask R-CNN to detect 7 different symbols, of which 6 are religious. Weights from the trained network were used to classify gravestones from a large amount of webscraped data.

The thesis is split into three main parts; theoretical explanation, outlisting of programming methods used, and results. The first part, theoretical explanations, walked through the functionality and buildup of artifical neural networks and convolutional neural networks. Various Deep convolutional models were also introduced. Those models have all been developed for the field of localized object detection and had success.

The second part, programming methods, explains how data was gathered in this project. It also mentions some of the configurations in the code as well as providing an explanation of how the network is built up.

The third and last main part, results, was split into two section. The prior section detailed the performance of the neural network during training and on the validation set. It also compared performance of the neural network to manual classification. The latter section showed relations between regional religiosity and lifespan of memorials located within them, based on if they were classified as religious or not. It was split into a part focusing solely on the United States and a part focused on data scraped from other regions of the world.

5.1 Future Work

A few challenges encountered will need to be addressed in the future.

The lack of an accurate test set was noted. Manually classified gravestones received from Jochen Gebauer and Tobias Ebert was used to evaluate the performance of the network. However, that classification was not always done on the same image the neural network was shown. Creating an accurate test set of a few hundred images could better highlight the performance of the network. That could also highlight

differences on generalizability for gravestones from each country. A difference in appearence of symbols in the United States and other European countries was noted in tables 4.2 and 4.4. It might well be that other symbols not considered in this project appear more frequently in other countries than the United States.

The results were that even without perfectly classifying all symbols of the neural network, a correlation between religiosity, lifespan and symbolism on gravestones was apparent. Involving other variables like Gross Domestic Product or Human Development Index could be interesting. It is easy to imagine such factors having an effect on wellbeing and length of life. Comparing that with the effect of religious symbolism on gravestones could tell a story.

To gather more data from other countries than the United States, websites such as `http://findengrav.dk/` could also be used. Religious classifications on subdivisions on other countries than the United States could also highlight some interesting traits within them. Another possibility is to use religiosity adherency rates of the United States throughout time, as is available on [5]. That can be matched with the birthyears and deathyears of memorials.

A lot of other interesting data is to be found on `https://www.findagrave.com/` as well. Information about birthplaces and deathplaces could shine a light on common movements of people throughout their lives. Information on family relationship could be examined as well.

The rates at which neural network models have progressed in image classification and object detection has been very swift since [6] was introduced. [10] was published back in 2017, so there might well be more recent models which have not come to this projects attention.

APPENDIX A

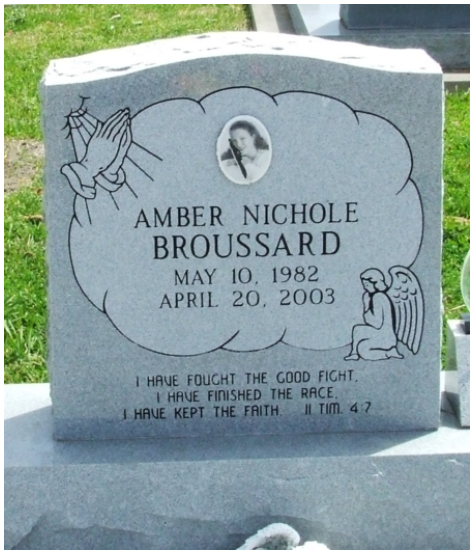
Appendix

The appendix will show a few images of the symbols in the project as well as some detections. The full distributions from figures 4.6 and 4.9 will also be shown.

A.1 Images

(a) Example of the symbols of praying hands and an angel.

(b) Example of the symbols of bibles and a cross.

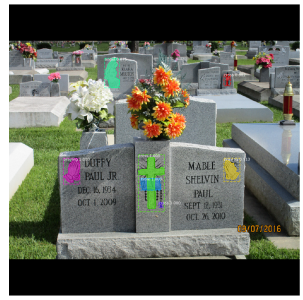


(c) Example of the symbols of a dove.

(d) Example of the symbol of a David's star.

Figure A.1: Examples of the religious symbols the neural network is taught to recognize.

(a) Three detected angels, one detected cross. One undetected cross



(b) Detected praying hands, crosses and Bible. Detected angel from another gravestone.

(c) Detected Bible, cross and angel.

(d) Detected Star of David and Bible.

(e) Detected crosses, Bibles and praying hands.

(f) Detected person.

(g) Detected angel. Incorrect Bible and cross detection.

(h) Detected crosses but undetected praying hands.

Figure A.2: Figures A.2a-A.2h show examples of correct and incorrect detections.

A.2 Full distributions

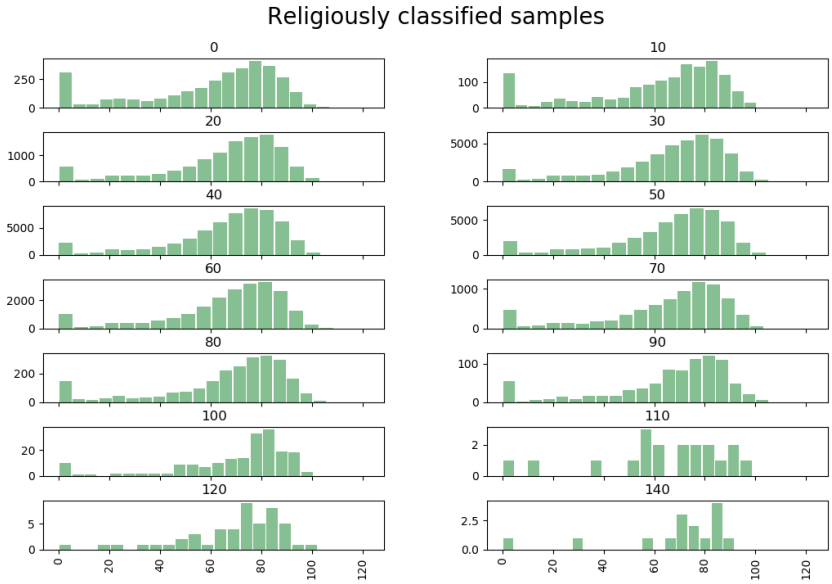


Figure A.3: Distributions of ages of memorials classified as religious within all religiosity deciles in the United States from a random sample of 587010 samples.

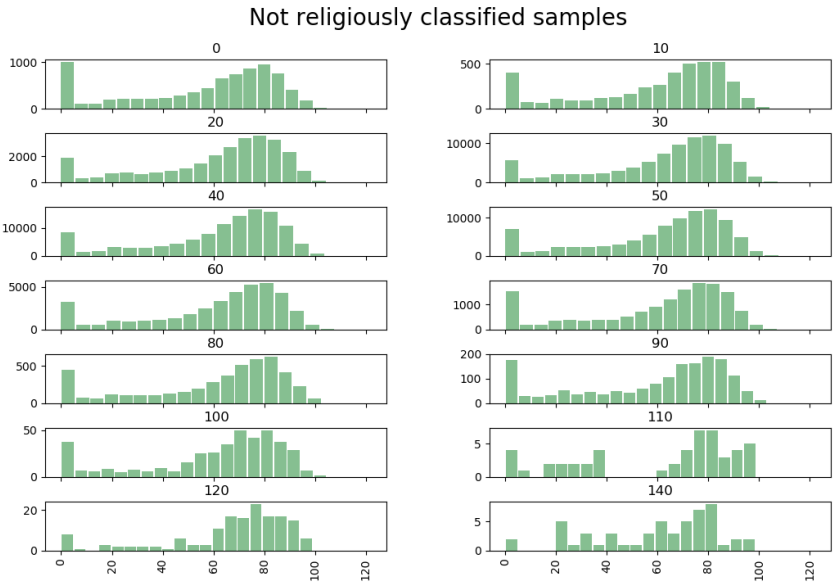


Figure A.4: Distributions of ages of memorials not classified as religious within all religiosity deciles in the United States from a random sample of 587010 samples.

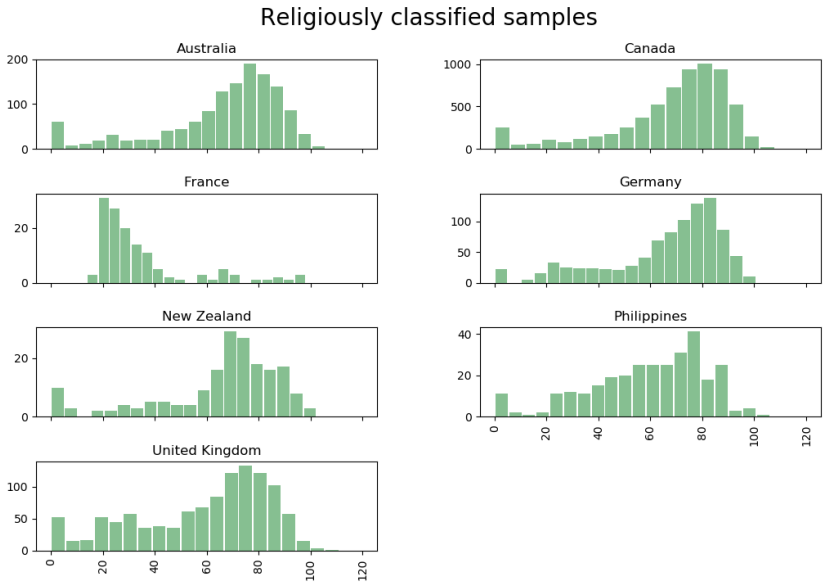


Figure A.5: Distributions of ages of memorials classified as religious from each country considered in section 4.2.2. Total of 42472 samples.

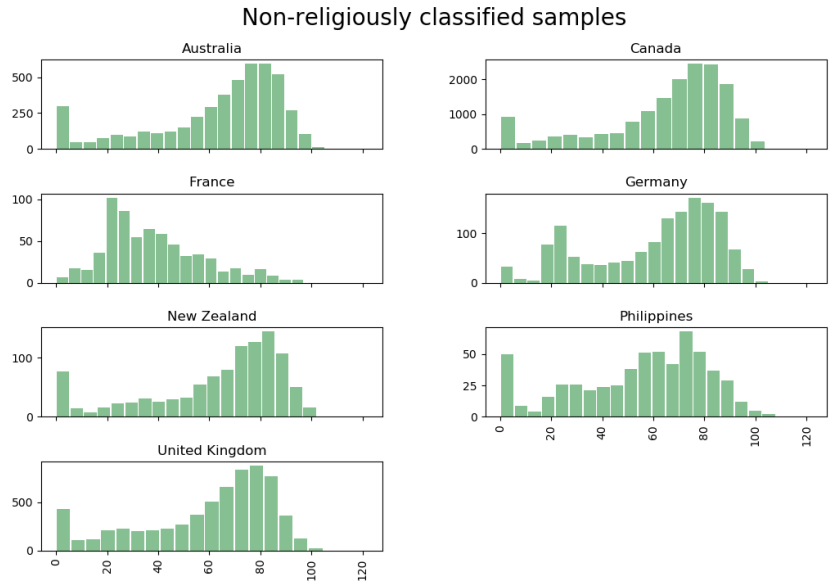


Figure A.6: Distributions of ages of memorials classified as non-religious from each country considered in section 4.2.2. Total of 42472 samples.

Bibliography

[1] J. Gebauer, F. Schönbrodt, P. Rentfrow, C. Sedikides, W. Bleidorn, and J. Potter, "The Religiosity as a Social Value Hypothesis: A multi-method replication and extension across 65 countries and three levels of spatial aggregation," *Journal of Personality and Social Psychology*, volume 113, number 3, pages 18–39, 2017.

[2] Y. Chida, A. Steptoe, and L. Powell, "Religiosity/Spirituality and mortality a systematic quantitative review," *Psychotherapy and psychosomatics*, volume 78, pages 81–90, March 2009.

[3] T. Ebert, J. Gebauer, J. Talman, and J. Rentfrow, "What the dead may tell us about the living: Religiosity and longevity based on gravestone inscriptions,"

[4] W. Zelinsky, "The gravestone index: Tracking personal religiosity across nations, regions, and periods," *Geographical Review*, volume 97, number 4, pages 441–466, 2007.

[5] *U.s. religion census | religious statistics demographics*, `http://www.usreligioncensus.org/index.php`, Accessed: 2019-12-28.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pages 1097–1105.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

[8] A. Karpathy, *What i learned from competing against a convnet on imagenet)*, `https://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/`, January 2020.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, series CVPR '14, Washington, DC, USA: IEEE Computer Society, 2014, pages 580–587.

[10] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, 2017. [Online]. Available: `http://arxiv.org/abs/1703.06870`.

[11] J. W. Johnson, "Adapting mask-rcnn for automatic nucleus segmentation," *CoRR*, volume abs/1805.00500, 2018.

[12] D. Kudinov and D. Hedges. (2018). Reconstructing 3d buildings from aerial lidar with ai: Details, [Online]. Available: `https://medium.com/geoai/reconstructing-3d-buildings-from-aerial-lidar-with-ai-details-6a81cb3079c0` (visited on November 13, 2019).

[13] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, pages 65–386, 1958.

[14] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006, ISBN: 0387310738.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, `http://www.deeplearningbook.org`.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1," in, D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds., Cambridge, MA, USA: MIT Press, 1986, ch. Learning Internal Representations by Error Propagation, pages 318–362.

[17] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," *CoRR*, volume abs/1808.01974, 2018. eprint: `1808.01974`.

[18] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, *Microsoft coco: Common objects in context*, 2014. [Online]. Available: `http://arxiv.org/abs/1405.0312`.

[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pages 91–99.

[20] A. Karpathy, *Predicting cut-ins (Andrej Karpathy)*, `https://www.youtube.com/watch?v=A44hbogdKwI`, November 2019.

[21] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurons in the cat's striate cortex," *Journal of Physiology*, volume 148, pages 574–591, 1959.

[22] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, volume 36, pages 193–202, 1980.

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, volume 86, 1998, pages 2278–2324.

[24] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks.," *CoRR*, volume abs/1311.2901, 2013. [Online]. Available: `http://dblp.uni-trier.de/db/journals/corr/corr1311.html#ZeilerF13`.

[25] G. Liu, K. J. Shih, T. Wang, F. A. Reda, K. Sapra, Z. Yu, A. Tao, and B. Catanzaro, "Partial convolution based padding," *CoRR*, 2018.

[26] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, volume 39, number 4, pages 640–651, April 2017.

[27] R. Girshick, "Fast r-cnn," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, series ICCV '15, Washington, DC, USA: IEEE Computer Society, 2015, pages 1440–1448.

[28] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *CoRR*, 2016.

[29] W. Abdulla, *Mask r-cnn for object detection and instance segmentation on keras and tensorflow*, `https://github.com/matterport/Mask_RCNN`, 2017.

[30] Stackoverflow. (2019). Stack Overflow developer survey results 2019, [Online]. Available: `https://insights.stackoverflow.com/survey/2019` (visited on December 14, 2019).

[31] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: `https://www.tensorflow.org/`.

[32] F. Chollet *et al.*, *Keras*, `https://keras.io`, 2015.

[33] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pages 51–56.

[34] L. Richardson, "Beautiful soup documentation," *April*, 2007.

[35] A. B. Jung, K. Wada, J. Crall, S. Tanaka, J. Graving, S. Yadav, J. Banerjee, G. Vecsei, A. Kraft, J. Borovec, C. Vallentin, S. Zhydenko, K. Pfeiffer, B. Cook, I. Fernández, W. Chi-Hung, A. Ayala-Acevedo, R. Meudec, M. Laporte, *et al.*, *imgaug*, `https://github.com/aleju/imgaug`, Online; accessed 25-Sept-2019, 2019.

[36] P. GNU, *Free software foundation. bash (3.2. 48)[unix shell program]*, 2007.

[37] A. Dutta and A. Zisserman, "The VIA annotation software for images, audio and video," in *Proceedings of the 27th ACM International Conference on Multimedia*, series MM '19, Nice, France: ACM, 2019, ISBN: 978-1-4503-6889-6/19/10. DOI: 10.1145/3343031.3350535. [Online]. Available: https://doi.org/10.1145/3343031.3350535.

[38] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. on Knowl. and Data Eng.*, volume 22, number 10, pages 1345–1359, October 2010, ISSN: 1041-4347. DOI: 10.1109/TKDE.2009.191. [Online]. Available: https://doi.org/10.1109/TKDE.2009.191.

[39] *Achievements in public health, 1900-1999: Healthier mothers and babies*, https://www.cdc.gov/mmwr/preview/mmwrhtml/mm4838a2.htm, Accessed: 2019-12-30.