university of groningen / faculty of science and engineering

Classifying Infection Stages in Patients with Ventricular Assist Devices

Master's Thesis

Noël Lüneburg
s1773135

Internal Supervisor:
Dr. Marco Wiering (Artificial Intelligence, University of Groningen)

External Supervisor:
MSc. Sybren Jansen (Target Holding B.V., Atoomweg 6B, Groningen)

Artificial Intelligence
University of Groningen, The Netherlands

Abstract

Deep learning has made large steps forward in the image processing field, including in highly specific medical cases. Skin infections are a common problem in patients with a heart assist device. The research in this thesis focuses on classifying skin infections by making use of recent deep learning advancements. A semantic segmentation convolutional neural network was trained to identify objects in the images unrelated to infection status. A small and imbalanced data set formed one of the challenges. In addition to affine transformations, generative adversarial networks (GANs) were used to generate training data as a form of data set augmentation, artificially increasing the size of the data set. Results show that applying segmentation masks of irrelevant objects to images during the classification step does not improve classification performance. Images similar to the training data were generated by a GAN. However, the available data set was likely too small for GANs to be used as data augmentation. Classification results show that infection class prediction based solely on external photos is a difficult problem, as confirmed by a validation experiment with medical experts.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Patients with severe heart conditions find themselves in life threatening conditions. A heart transplant is an extreme but possible treatment option. Not all patients qualify for a heart transplant due to the reason that there is only a limited number of donor hearts available at any given time. A left ventricular assist device (LVAD) is a common alternative for a heart transplant, either as a temporary solution or as a lifelong solution, termed destination therapy [39].

One of the major issues with LVAD implants is the driveline exit site, which is susceptible to skin infections [56]. Different stages of infections exist, from minor to severe. It is important to monitor infection status in LVAD patients as a severe infection can be fatal if left untreated for too long. Monitoring of driveline exit site infections is done in specialized clinics. Long travel time to clinics is one of the main disadvantages of the current monitoring process. Therefore, a significant increase in patient quality of life could be realized by a telemonitoring approach [42], in which patients would be monitored remotely for potentially developing infections.

1.1 Project scope

A concrete case for LVAD infection classification was created as part of the international ITEA Medolution project[1]. Within this project, colleagues from Target Holding[2] and myself cooperated with Schüchtermann-Schillersche Kliniken and Medical School Hannover's department of Cardiothoracic, Transplantation and Vascular Surgery. The LVAD case revolved around automatically classifying LVAD patient photos (taken by a mobile device for example) in order to provide decision support for the doctors. All data used in the experiments was provided by the heart clinics and an interactive setting was created between medical experts and machine learning engineers.

1.2 Deep learning applications

Before deep learning became popular, machine learning methods such as the support vector machine and multi-layer perceptron were widely used in medical applications supported by artificial intelligence [31]. Deep learning approaches started gaining popularity in the medical field around 2016 [26].

[1] http://medolution.org/
[2] https://www.target-holding.nl/?lang=en

1.2.1 Infection classification

One of the requirements for training very deep networks is a large labelled data set. In highly specific medical cases a large data set is often not available or the distribution of class labels is highly imbalanced. Two common methods for applying deep learning to smaller data sets is to use transfer learning [54] or data set augmentation [21, 38]. In transfer learning, a neural network which has been trained on a specific data set is used in combination with a different data set. This allows for applying a very deep network to a smaller data set if it has been *pretrained* on a larger data set. To overcome the difference in class labels or features of the data, the model may be *finetuned*, where a pretained model is further trained on the target data set.

No research was discovered with regard to deep learning applications in LVAD infection classification. An instance of related research focuses on infection classification of general wounds in a binary setting [51]. Cases related to skin infection classification often researched in the machine learning medical field include skin lesions such as melanomas and carcinomas [32]. This type of research focuses on dermoscopic image data, which entails a standardized image structure [5]. In skin lesion research, convolutional neural networks (CNNs) have been applied as feature extractors, separate from a classification model [16, 30]. Other research on skin lesion classification makes use of transfer learning applied to a pretrained CNN classifier [28]. In a three-class setting, a CNN model trained on a large data set of 129,450 images, has been shown to be capable of outperforming a human expert in skin cancer classification [9].

1.2.2 Data set augmentation

Data set augmentation is a method of artificially extending a training data set by using label-preserving techniques. Next to transfer learning, data augmentation is one of the main techniques used to overcome the scarcity of medical data sets [32]. Successfully applying image data augmentation requires some insight into the training data as to how the data can be manipulated without affecting the labels. If a classification problem includes determining whether an object on a table surface is upright or laying down then augmentation by rotating training images would be ill-advised.

A separate branch of deep learning models, called generative modeling, may be used as a form of data augmentation [34]. Generative adversarial networks (GANs) [11] can learn the underlying distribution of a data set and can therefore be used to generate unique data samples which are similar to the original training samples. GAN data augmentation has been successfully applied in skin lesion segmentation research [1]. The authors make use of a conditional GAN image-to-image translation network [13, 37], to generate training data conditioned on the ground-truth segmentation masks of the same training data.

Additional steps have to be taken to generate labelled generated samples using GANs, in order to be useful in classification applications. Alternatively, data augmentation can be realized by combining a GAN with a classifier in a semi-supervised setting [44].

1.2.3 Segmentation

In addition to classification, semantic segmentation is an important topic in the medical field, which can be used to localize areas of interest or, for example, to determine the shape of melanomas [46]. In semantic segmentation, each pixel in an image is assigned a specific class. This is one level of detail lower than instance segmentation, in which separate objects within a class are identified. Semantic deep learning CNN segmentation networks have been shown to be capable of accurate segmentation performance in highly specific tasks [43]. In the case of LVAD patients infection classification, segmentation can be applied in two approaches. The first approach attempts to separate skin from non-skin

objects, allowing the classification network to focus only on the relevant areas of an image. The second approach aims at predicting regions of interest, such that the majority of irrelevant background features can be ignored by a classifier. An example of the second approach is given in [51], which combines wound segmentation and infection classification. The wound segmentation CNN here consists of an encoder-decoder architecture. This allows for using the high-level features extracted during segmentation to be used for infection classification by a separate classifier. The high-level features of the segmentation network may not be suitable for infection classification, as the authors of [51] report a binary F1 score of 0.348.

1.3 Research questions

The main contribution of this thesis is an attempt at achieving human-level performance on infection type classification in LVAD patients by building on the state of the art of existing deep learning models. To support the main goal, this thesis attempts to answer the following research questions:

1. What increase in performance of the classifier is realized by augmenting the data with artificial images by using geometric transformations or a generative adversarial network?

2. What increase in performance of the classifier is realized by semantically segmenting the visible driveline tube from patient photos?

Throughout this thesis a subdivision is made into the topics classification, segmentation and augmentation. In Chapter 2 the theoretical background of relevant methods and models is described. Chapter 3 describes the properties of the image data set. Chapter 4 contains descriptions of experiments and configurations of models. Chapter 5 lists and analyzes the results of the experiments. Finally, Chapter 6 contains the conclusions drawn from the experiments and remaining future work.

Chapter 2

Theoretical background

2.1 Convolutional neural network

The individual features in highly dimensional data samples are often related in some way. In time series data, the sequence of features within a single data point has a temporal relation. In 2- or 3-dimensional image data the features are spatially related. A convolutional neural network (CNN) is suited for these types of data as a convolution kernel is a temporal or spatial representation of a local receptive field.

A CNN contains a number of convolution layers. Within a convolution layer a number of filter kernels of a particular size are convolved with the input to the layer. The output of a convolution layer in the discrete domain is computed by calculating the cross-correlation between each filter kernels and the layer's input [8]. A filter kernel is repeatedly applied to different positions in the input structure to compute local features across the full input structure. The method of repetition is determined by the kernel's stride. For each dimension of the input data, a stride value of 1 ensures that the filter kernel is centered on each element of the input data. Higher stride values result in fewer applications of the filter which could cause the kernel to skip over potentially important features in the input. When considering a multi-layered CNN a higher stride value in a particular convolution layer effectively increases the receptive field of the later layers of the network by reducing the output dimensions. The size of a filter kernel determines the receptive field of the current convolution layer; the area of the input data that can be captured inside a single kernel. The output dimensions of a convolution layer are determined by the filter kernel's stride and size. For any dimension of the input the following relation holds [8]:

$$o = \lfloor \frac{i-k}{s} \rfloor + 1 \tag{2.1}$$

where o is the output size, i is input size, k is the filter kernel size and s is the kernel stride. Given a two-dimensional input image the output dimensions are then given by $D = \{O_w, O_h, N\}$, where O_w and O_h are the output sizes obtained by applying (2.1) using the width and height dimensions of the input image. N is the number of output feature maps and is determined by the number of kernel filters present in the convolution layer. A visualization of a convolution for the case of $i = 4$, $k = 3$, $s = 1$ and $o = 2$ is given in Figure 2.1. In the image domain the term 2D convolution is used even though the computations are usually done on 3D input, where the third dimension represents either the color channels of the input image or the feature maps in higher layers. Each kernel applied in 2D convolution layers has an additional dimension which contains weights for each channel of the input. The 3-dimensional result of applying a single kernel to each input channel is then summed along the channel's dimension to create a single output channel. Convolution layers can be applied to data samples of three dimensions or higher following the same methods as described here. In this thesis only image-based 2D convolutions are discussed.

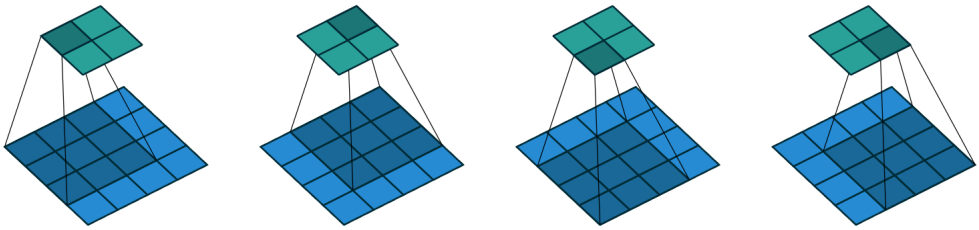


Figure 2.1: Abstract visualization of convolving a 4x4 single channel, non-padded image with a 3x3 kernel resulting in a 2x2x1 feature map output. Taken from [8].

2.1.1 Padding

When a convolution layer with a stride of 1 is applied to an input image, the size of the resulting feature map will be smaller than the input image, as indicated by Equation (2.1). To overcome this change in size padding can be applied to the input image. This is a method of adding additional pixels to the edges of an image to artificially increase its size.

In CNN architectures which try to preserve input size, a popular method is to use zero padding which pads an image with values of zero. This can create artefacts in the feature maps, especially when combined with a small kernel size and small input size. When a substantial area must be added to an image, reflection padding is likely more realistic than zero padding [55]. Reflection padding adds pixels by mirroring the pixels about each edge.

2.1.2 Pooling

Pooling operations can be used in sequence with convolution layers. A pooling operation functions as a sliding window method similar to a convolution kernel, although a pooling window has no internal weights. Instead, only the window size and stride are parameterized, in combination with an aggregation function that produces a single value, such as the maximum or average. An intuitive representation of a (max) pooling layer is that of translation invariance. By reducing the spatial features within a pooling window to its maximum, the information about the most important feature in the window is kept while the information pertaining to the exact location of the feature is discarded. For this reason, max pooling operations are often configured with stride s and window size k as $s = k$ to avoid overlap.

2.1.3 Global pooling

In a classification CNN the feature maps eventually have to be connected to a representation of the possible class outputs. This can be realized by connecting the last feature map to an output vector with a fully connected structure. Each element in the output vector represents the prediction value for a particular class. A fully connected layer has many learnable weights. In addition, a fixed input size is required for the fully connected layer which in turn means a fixed input size is required for the convolutional layers. A fully connected layer can be substituted for a global pooling layer. Instead of aggregating areas in a feature map using a sliding window, a global pooling operation aggregates each feature channel to a single value, reducing a 3D feature map to a 1D vector, where its length is equal to the number of channels in the feature map. Global pooling requires no parameters (except for the aggregation function) as opposed to a fully connected layer. Because the aggregation function converts an arbitrary number of values to a single element in the output vector the network's input size can be variable.

2.1.4 Transposed convolution

In a CNN which consists only of regular convolution layers with no padding, the width and height of the feature map output of each subsequent layer decreases in size, resulting in a high level dense representation of features at the output of the network. In some applications it may be desirable to apply convolution in the opposite direction, where high level representations are converted into formats that are more similar to the input of a CNN. This type of convolution is called *transposed convolution* (sometimes incorrectly referred to as *deconvolution* [8]), illustrated in Figure 2.2. In a transposed convolution layer the kernel is defined as usual, however, the forward pass and backward pass are swapped. For any given combination of input and convolution kernel, the convolution operation can be described as a matrix A which is multiplied with the input $\mathbf{x}$ to create output $\mathbf{y}$:

$$\mathbf{y} = A\mathbf{x} \tag{2.2}$$

During backpropagation the transpose of the matrix, A^T, can be used to compute the gradients of the kernel weights [8]. The forward pass of a transposed convolution layer can be described as $\mathbf{y} = A^T\mathbf{x}$. During backpropgation the transposed matrix then becomes $(A^T)^T = A$.

2.1.5 Dilated convolution

Dilated convolutions, also referred to as *atrous convolutions* consist of regular convolution kernels which contain empty spaces. Intuitively, the kernel is "stretched out" across a larger area, see Figure 2.3. The size of the gaps in the kernel is controlled by the dilation rate. By using a dilated convolution layer instead of regular convolution some information from the input is lost, however, as a single kernel can cover a larger area of the input, the receptive field is increased. A similar increase in receptive field can be realized by chaining multiple regular convolution layers, at the cost of a larger number of learnable kernel weights.

2.1.6 Activation functions

The usage of non-linear activation functions allows a neural network to learn non-linear functions. In addition, they may perform a type of regularization by squashing the output of a node in a network to a specific range.

A standard approach for setting up a multi-class classification network is to include one output node for every possible class label. Common objective functions such as the cross-entropy / negative log likelihood require output probabilities per class instead of values in an unbounded range. To satisfy this requirement, a popular choice is to apply the softmax normalization function to the output node activations $\mathbf{y}$:

$$f(y_i) = \frac{e^{y_i}}{\sum e^{y_j}} \tag{2.3}$$

Rectified linear unit

In CNN architectures the rectified linear unit (ReLU) [33] is a popular activation function. It is a piecewise linear function that is described as $f(x) = max(0,x)$. Its derivative is easy to compute and it does not squash extreme input values to a very narrow range unlike the sigmoid function for example. When $x <= 0$, the derivative of the ReLU is 0, which could prevent useful weight updates during network optimization. A variant of the ReLU that does not have this property is the leaky ReLU [53], which defines a slope in the negative region:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{otherwise} \end{cases} \tag{2.4}$$

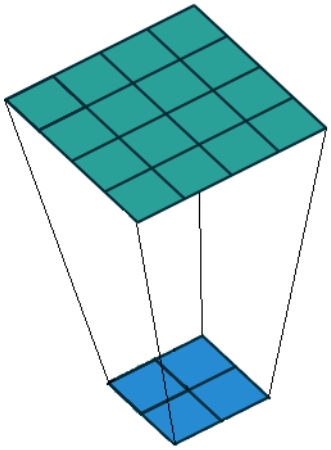


Figure 2.2: Transposed convolution. Note that the bottom part represents the input and the upper part the output. The image size is effectively increased by a transposed convolution. Adapted from [8].

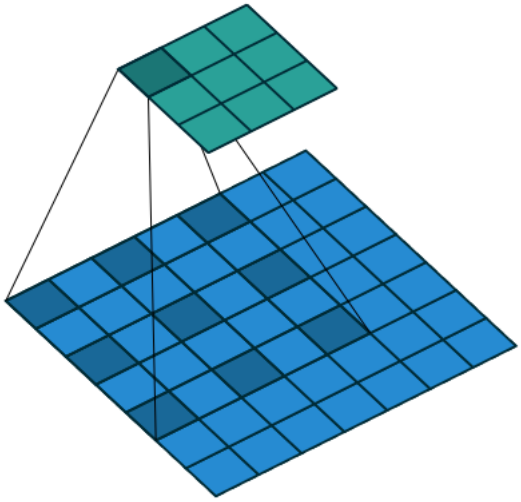


Figure 2.3: Dilated convolution. The dark blue pixels indicate a 3x3 kernel with a dilation factor of 2. Taken from [8].

where $a > 0$ is a constant that controls the steepness of the slope.

2.1.7 Inception v3

A popular CNN architecture in the deep learning image processing field is the Inception v3 network [50]. A schematic of the architecture is display in figure 2.4. The network started as GoogLeNet [49] which was a computationally and memory efficient implementation of a deep CNN compared to earlier deep CNNs. The Inception architecture makes use of *Inception modules* which allows convolutions of multiple kernel sizes to be applied to a feature map in parallel, in addition to pooling. The results of each *branch* in this parallel structure are then concatenated to form the output of an Inception module. 1x1 convolutions are used to support depth-wise relations, as opposed to spatial relations which are learned using larger convolution kernels. Improvements to the Inception module include factorization of convolutions for computational efficiency [50]. In addition to Inception modules, the Inception architecture contains an auxiliary output, used to prevent vanishing gradients during training.

2.2 Segmentation

Segmentation in image processing refers to the method of grouping the pixels of an image into one or more components which each contain a visually similar region of the image. The pixels within each region do not have to form a single connected component.

Image segmentation can be done using unsupervised or supervised methods. Unsupervised segmentation is a form of clustering since the data samples (pixels) are grouped together based on relations in some color space. Examples of unsupervised segmentation algorithms include k-means segmentation [29], Felzenszwalb graph-based segmentation [10] and active contour models [15].

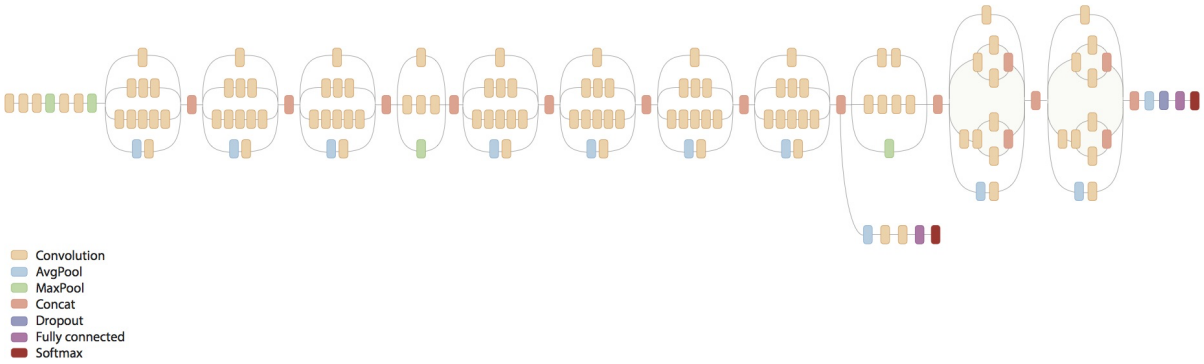


Figure 2.4: Inception v3 architecture[a]. Inception modules are identified by sections of parallel branches.
[a] Source: https://github.com/tensorflow/models/tree/master/research/inception

Supervised segmentation methods consist of training a model with ground-truth segmentation examples. The term semantic segmentation is used when a model outputs a class label for each pixel in the input image. Neural network approaches are popular for semantic segmentation. Example methods include fully convolutional neural networks (FCNN) [27], optionally in combination with conditional random fields [4] or visualized as a U-shaped network containing cross-connections between distant layers in the network [43].

2.2.1 Felzenszwalb segmentation

The Felzenszwalb algorithm [10] is a greedy graph-based segmentation method. It attempts to make use of local and global properties in the image to decide appropriate segmentation boundaries.

An image can be represented as a graph containing vertices connected via weighted edges. Each vertex is a pixel and the weight of each edge is a similarity measure between two connected pixels. The vertices of the graph can be grouped into components, such that each component is a subset of the initial graph. A pair of separate components, C_1 and C_2 can be compared to decide whether they should be merged based on inter-component and intra-component differences. The internal difference of a single component C is defined as:

$$Int(C) = \max_i w(E_i) \tag{2.5}$$

where E is the set of edges within the minimum spanning tree of component C and $w(E_i)$ is the weight of edge i in C. In regular terms, the strongest edge in the component graph determines the internal difference. The difference between two components is defined as:

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2} w((v_i, v_j)) \tag{2.6}$$

where $w((v_i, v_j))$ represents the weight of a given edge that connects the two components. The inter-difference formulation is less intuitive than the intra-difference equation, however, the authors of [10] remark that the minimum operator works well here. The decision function M for merging two components C_1 and C_2 can now be defined as:

$$M(C_1, C_2) = \begin{cases} \text{false} & \text{if } Dif(C_1, C_2) > S(C_1, C_2) \\ \text{true} & \text{otherwise} \end{cases} \tag{2.7}$$

Figure 2.5: Example image and output segmentation mask of the Felzenszwalb algorithm. Each separate component is given its own unique color. Note that the colors are semantically irrelevant here as there are no class labels. Taken from [10].

where S is the minimum internal difference:

$$S(C_1, C_2) = min(Int(C_1) + \tau(C_1),\ Int(C_2) + \tau(C_2)) \tag{2.8}$$

where τ is a threshold function based on the size of the component, such that τ results in a higher value for larger components. Without the threshold function a component containing a single element (pixel) will have no internal edges and therefore $Int(C) = 0$. The strength of τ can be altered via a scaling parameter, which represents a bias towards larger or smaller components in the segmentation.

The Felzenszwalb algorithm is initialized by grouping each pixel into its own component and then iteratively merging components using the merge decision function described in Equation (2.7). Edges are initialized by using a fixed number of nearest neighbors in the feature space ((r, g, b) in the case of color images). An example image and its computed segmentation mask is shown in Figure 2.5.

2.2.2 Kuwahara filter

Unsupervised segmentation methods may be sensitive to strong variations in contrast not consistent across multiple images. To prevent these local peaks in contrast, the Kuwahara edge-preserving smoothing filter [23] can be applied to each image. The Kuwahara filter applies smoothing to visually coherent regions, yet prevents smoothing edges.

2.2.3 U-net semantic segmentation

U-net [43] is an extension of a deep FCNN. As such, its architecture makes a distinction between an expanding path and a contracting path. Beyond the standard FCNN architecture, U-net introduces cross-connections which are direct paths between levels of the expanding and contracting sections of the network. An illustration of the architecture is given in Figure 2.6.

The contracting part of U-net is similar to a regular CNN in which an image with few channels is fed through a series of pooling and convolution layers resulting in a small spatial image containing many channels. In a regular CNN, the output of a series of convolutions and pooling operations is usually connected to a classification output structure via a fully connected layer or a global pooling layer (see also Section 2.1.3). In U-net a mirrored version of a contracting network is concatenated, which features either transposed convolutions (Section 2.1.4) or upsampling operations (refer to Section 2.3.1 for a comparison of the two methods). These upsampling blocks allow the network to learn to convert a dense feature map into an image-like object.

The cross-connections connecting the contracting path and expanding path are a concatenation operation in which the output from lower level convolution layers is directly concatenated to the input of one of the upsampling convolution layers. When regular non-padded convolutions are used in the contracting path then the dimensions of the feature maps on either side of a cross-connection are not

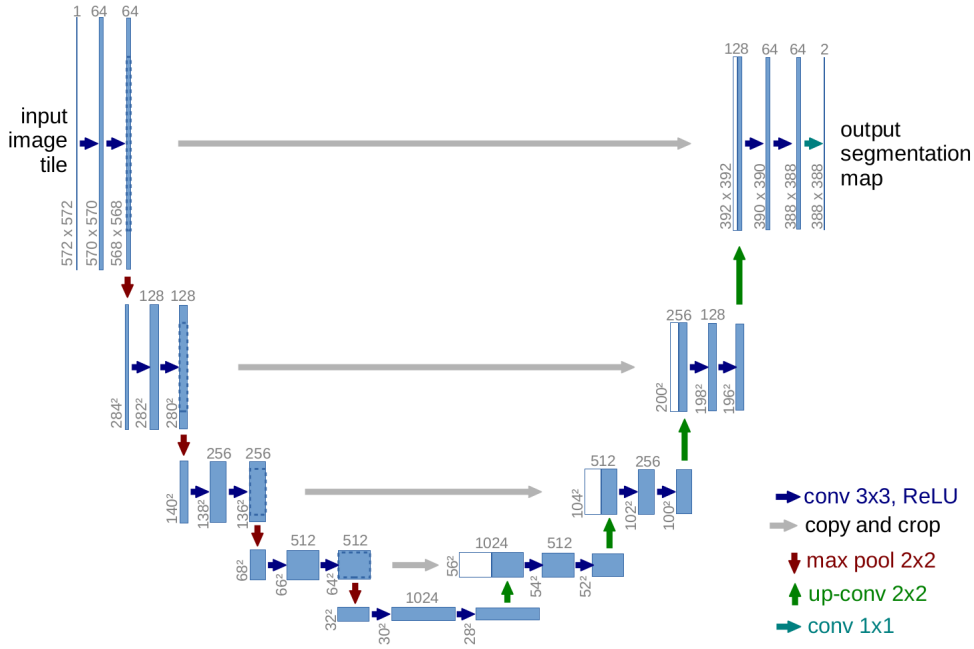


Figure 2.6: U-net architecture of five levels deep designed to accept single-channel input of 572x572 resolution and outputs a 388x388x2 segmentation mask. The left side of the figure consists of the contracting path and the right side consists of the expanding path. The architecture features four cross-connections connecting the convolutional blocks of both paths at multiple levels. Taken from [43].

equal, requiring cropping before concatenation as displayed in Figure 2.6. When the input of each convolution layer is padded to ensure identical output size, no cropping is necessary.

2.2.4 Dice score coefficient

To measure the quality of overlap between two binary sets a number of different metrics are available. This section will describe the Dice score coefficient [7] (DSC) with regard to image segmentation quality assessment.

The binary DSC for two sets A and B can be formalized as:

$$DSC(A,B) = \frac{2\,|A \cap B|}{|A| + |B|} \tag{2.9}$$

which is identical to the F1 score and very similar to the Jaccard index (J), also known as the intersection over union (IoU). The two metrics both result in a value of 1 for a perfect match and a value of 0 when the two sets are completely disjoint. In-between the extremes they are related via $DSC = \frac{2J}{(J+1)}$ [45].

The Dice score coefficient can be used to assess image segmentation quality in terms of sets of pixels. Intuitively, the DSC measures the amount of overlap between ground truth and a segmentation mask. The DSC can be applied to multi-class segmentation settings, termed DSC_+, by combining the Dice score for each class independently:

$$DSC_+ = \sum_{n=1}^{N} w_n DSC_n \tag{2.10}$$

where DSC_n is the Dice score coefficient of class n and w_n is the class weight. This is a form of the generalized Dice score coefficient [48]. To make sure a value in the range $[0,1]$ is obtained, the class weights must sum up to 1. For equal class weights, $w_n = 1/N$. Adjusting class weights can provide a more balanced performance measure when the class labels are imbalanced, as is often the case in semantic segmentation problems.

The DSC can also be used as an objective function. As objective functions are often minimized the complement, $1-DSC_+$, can be used. This is called the *soft* Dice loss, as the output of a model is used directly in the calculation of the DSC instead of first applying the $argmax$ operation to determine the predicted class for each pixel.

2.3 Generative adversarial network

A generative adversarial network (GAN) [11] is a network structure designed to capture the distribution of a data set and to allow sampling from the learned distribution in order to generate unique samples that are similar to the samples in the data set.

A GAN consists of two specialized networks which are connected, a *generator* and a *discriminator*, see Figure 2.7.

The generator network is designed to output a data sample. The network itself is deterministic (assuming no application of techniques such as dropout [47]). The properties of a generated sample are determined by the input of the generator, which is a noise vector sampled from what is called a *latent space*. The latent space can be implicitly leveraged by the generator to group high-level features together. A generator G thus maps a variable sampled from a latent probability distribution, $\mathbf{z} \sim p_{\mathbf{z}}$, to a learned target distribution via $G(\mathbf{z})$. Another variant of a generative model which uses a latent space is a variational autoencoder (VAE) [18]. In a VAE sampling from the latent space is done with parameters of a distribution whereas in GANs the latent space is discrete. VAEs often lead to more blurry generated samples when compared to GAN generated samples. A combination of the two model types is also possible, where the generator of a GAN is replaced with a VAE encoder-decoder structure [24], in order to avoid optimizing using a pixel-wise reconstruction error metric.

The discriminator network is designed to determine whether a given data sample is real or fake, i.e. generated, by outputting a value between 1 and 0, corresponding to real and fake respectively. This entails that the discriminator extracts features to form a high level representation of its input in order to distinguish between real/fake. If the target application of the GAN is data generation, then the discriminator is only used during training.

Training procedure

Unlike many deep networks which optimize a single error metric to convergence, a GAN is trained in an adversarial setting, in which the generator (G) and discriminator (D) networks try to outperform each other, essentially playing a *minimax* game. There is no general convergence of the model as both G and D are optimized simultaneously. Formally, G is optimized by the gradient *descent* equation:

$$\nabla_G = \frac{1}{m}\sum_{i=1}^{m}\log(1-D(G(\mathbf{z_i}))) \tag{2.11}$$

where $\mathbf{z_i}$ is a noise sample input in G. The generator is penalized if the discriminator correctly identifies a generated sample $G(z)$. D is optimized by the gradient *ascent* equation:

$$\nabla_D = \frac{1}{m}\sum_{i=1}^{m}[\log D(\mathbf{x_i}) + \log(1-D(G(\mathbf{z_i})))] \tag{2.12}$$

where $\mathbf{x_i}$ is a real data sample. The discriminator is penalized if the discriminator cannot determine the validity of a real data sample x, or misclassifies a generated sample $G(z)$ as real.

Mode collapse

Under ideal circumstances, if the generator and discriminator networks have enough capacity to capture the training data distribution, a GAN will settle in a Nash equilibrium [41] where the discriminator cannot distinguish between real and fake samples, outputting a value of 0.5 for each sample. However, the convergence to the equilibrium is not clearly observable from the errors of G and D.

When the generator and discriminator fall out of balance with each other, such as when G converges faster than D, the *Helvetica scenario* [11, 34], or mode collapse, may occur, in which G has found a set of very similar generated data samples that are always misclassified by D. In practice, this is often a point where the learning halts and the generator will only output a single unique data sample.

Adaptations in the training procedure can make it less likely for mode collapse to occur. If G converges to fooling D with almost no diversity in the generated examples then updating D more often relative to the update rate of G can prevent the performance of G and D from getting out of balance with each other. A more substantial adaption is to make use of Wasserstein training [2], in which the *critic* (analogous to a discriminator) can be trained till optimality which avoids mode collapse. Changes to the network architecture can also help to prevent mode collapse, as noted in [40], where batch normalization [12] is used to stabilize learning, especially at the start of training.

Latent space interpolation

A properly converged GAN should be able to map a random point in latent space to a unique image. This entails that interpolation between two points in latent space can be converted to a smooth transition between the images corresponding to these two points. Instead of linear interpolation, spherical interpolation inspired by animation physics is a more suitable technique as it leads to improved results when sampling generative networks [52]. The equation for spherical interpolation is defined as:

$$Slerp(q_1, q_2, \mu) = \frac{sin\ (1-\mu)\theta}{sin\ \theta} q_1 + \frac{sin\ \mu\theta}{sin\ \theta} q_2 \tag{2.13}$$

where q_1 and q_2 are two vectors between which interpolation is done and μ is the linear distance between q_1 and q_2. $cos\ \theta = q_1 \cdot q_2$.

2.3.1 Deep convolutional GAN

The application of GANs is most prevalent in the image domain where they are used to generate realistic looking images. A deep convolutional generative adversarial network (DCGAN) [40] is a GAN which instantiates the generator and discriminator as convolutional architectures. Specifically, the discriminator is a regular CNN classification structure which classifies a natural image as real or fake. The generator almost mirrors this structure, converting a high level representation into a natural image by transposed convolutions or upsampling layers (Section 2.1.4). The authors of [40] define a number of hyperparameters that they have found led to stable training:

- Replace pooling layers in the generator by strided convolutions and use transposed convolutions in the generator,
- use batch normalization [12] between convolution layers,
- remove any extra fully connected layers,

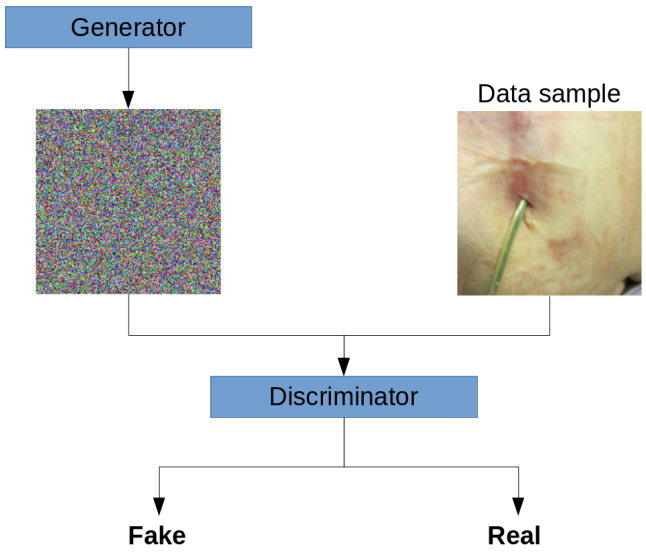


Figure 2.7: Simplification of a generative adversarial network. The generator network outputs generated images. The discriminator network assesses whether an input image originated from the generator network or from real data.

- in the generator, use rectified linear units (ReLU, Section 2.1.6) and the hyperbolic tangent function in the last layer,
- in the discriminator, use leaky ReLU.

Transposed convolution layers in the generator introduce overlap in the output if the kernel size of a transposed convolution is not divisible by the stride, which is often true. These overlaps can lead to artefacts in generated images, which is called the *checkerboard effect* [35]. It can occur at multiple scales in an image if transposed convolutions are applied in multiple layers. A network should be able to learn to avoid producing the checkerboard effect during training, however, this does not always happen in practice. The effect can also be avoided by replacing transposed convolutions by a simple upsampling operation followed directly by a 1x1 convolution which allows the network to learn a smooth upsampling, in addition to introducing non-linearity via a non-linear activation function applied to the output of the convolution.

2.3.2 Fast conditional GAN

Regular GANs can be used to generate data without labels. However, to generate labelled data an extension to the architecture is necessary. A GAN architecture that is designed to generate labelled data is the auxiliary classifier generative adversarial network (AC-GAN) [36]. The generator of an AC-GAN trained on data labelled with a set of classes $K = \{k_1, ..., k_n\}$ can generate an image sample with accompanying class label $k \in K$. The properties of an AC-GAN are similar to a regular GAN, however, the definition of the generator, $G(z)$, is adapted to also include the input class label, $G(z, k)$. An auxiliary output variable is included in the discriminator, which contains the predicted class label associated with the input of D. The objective function is extended to include the cross-entropy term calculated by the class label output of D. It has been demonstrated that the organization of the latent space $\mathbf{z}$ is independent of the class label k by visualizing the interpolation within a 2D latent space while keeping k fixed [19].

The AC-GAN model can be modified slightly to a fast conditional GAN (FC-GAN) [25]. During training, an additional fake class label $k_{n+1} \notin K$ is added which is associated with samples generated by $G(z, k_{n+1})$. The authors of FC-GAN demonstrate that general convergence and class differentiation is accelerated in comparison to an AC-GAN.

2.3.3 Inception score

It is not trivial to determine a metric that quantifies the quality of samples obtained from a generative model as there exists no ground truth for generated data. Inspired by the way that humans assess the quality of generated images, the authors of [44] propose the Inception score (IS) for generative models, which is defined as:[1]

$$IS(G) = \mathbb{E}_{\mathbf{x} \sim p_G} D_{KL}[p(y|\mathbf{x}) \parallel p(y)] \tag{2.14}$$

where $\mathbf{x} \sim p_G$ is a sampling of $\mathbf{x}$ from p_G, $p(y|\mathbf{x})$ is the conditional class distribution, $p(y)$ is the marginal class distribution and $D_{KL}(P \parallel Q)$ is the Kullback-Leibler divergence [22] between distributions P and Q.

The statistics required for evaluating Eq. (2.14) are obtained by applying the Inception v3 network [50] pretrained on ImageNet [6] on samples generated by the generative model. Samples from a well performing generative model result in a high Inception score if the samples respect two properties. The first is that each generated sample should contain a single clear entity, resulting in a high probability value for a single class in the output of the Inception network, which results in a low entropy for $p(y|\mathbf{x})$. The second property is that the generated samples should be highly diverse, resulting in a high entropy for $p(y)$. If the two properties are met, the two class distributions are sufficiently different, resulting in a high Inception score.

Two main issues can be identified with regard to the Inception score [3]. The score definition relies on class probabilities obtained by an optimized Inception v3 network, however, trained models are often categorized only by classification accuracy, which is different from the class probabilities. A pair of separately trained Inception v3 networks with an identical architecture may yield identical classification accuracy while producing different class probabilities. Second, since the behavior of the IS is closely related with the features and classes of ImageNet data, it can give skewed results if a generative model is trained on data different from ImageNet. For example, the most common class predictions of Inception v3 on CIFAR-10 [20] images do not align with the actual CIFAR-10 classes [3].

[1] The original formulation specifies the Inception score as an exponential function, e^{IS}, which is only relevant when comparing multiple Inception scores.

Chapter 3

Data set

The image data used in this research consisted of 1134 photos of in total 84 patients ($\mu = 13.5$, $\sigma = 13.2$), provided by the Schüchtermann heart clinic and the heart clinic at Hannover Medical School. The number of available photos varies highly between patients. Each photo was taken in one of the clinics during patient treatment or check-up. Factors such as surgical revisions and infection status determine the frequency of patient visits to the clinic, and therefore the number of photos taken. Each photo is roughly centered at the patient's driveline exit site. Photographing the wound area was not standardized, as the photos show high variability in zoom, sharpness, lighting and camera position/angles. The size of the photos ranged between 3-16 megapixels, with aspect ratio 4:3. In some cases parts of dressing or hands of the nurse are visible near or above the wound. Some patients have had surgical revisions on their wound, creating suture marks.

More patient photos became available over the course of the project, therefore not all experiments have been conducted with all of the 1134 images. Section 4 includes the number of data samples used in each experiment.

Class	**Name**	**Samples**	**Portion (%)**
0	no infection	561	50.0
1	minor infection	308	27.5
2	mild infection	215	19.2
3	severe infection	37	3.3
Total		1121	100

Table 3.1: Distribution of labelled image samples.

3.1 Labelling method

Out of the 1134 images, 1121 of these were assigned an infection label in the form of a natural number between 0 and 3. Label 0 indicates no infection; label 1 is a minor infection with symptoms such as a wet exit area and redness; label 2 is a mild infection, indicated by redness, fluid secretion and driveline movement; label 3 is the most severe infection label that presents itself with the symptoms redness in a large area, fluid secretion, a swollen wound area and driveline movement with breathing. See Table 3.1 for the class distribution. Figure 3.1 shows example images of each of the infection types. After consulting with the clinics we agreed to limit the number of infection classes to three in order to reduce classification complexity. This is realized by combining each sample labelled as either 0 or 1 into a single class (*no/minor infection*). These labels were difficult to distinguish from each other

Figure 3.1: Examples of manually cropped photos for each type of infection. The amount of redness and swelling tends to correlate with the infection severity.

and the distinction between labels 0 and 1 was not important enough to support in the classification algorithm, as an infection belonging to one of these classes did not warrant immediate action.

The infection label for each photo has been determined by a combination of two methods. A physician has made an assessment of the infection severity based on the features visible on a single photo. In addition, clinical data, such as presence of bacteria and sub-skin temperature, were available from tests during the patient's stay in the clinic. Both methods factor into a photo's final infection label, however, the weighting of each method in determining the final infection score is unknown. The clinical data has not been provided in a structural manner and was not used in any of the experiments described in Section 4. In addition, the Medolution project focused on a telemonitoring solution in which no clinical data would be available in a production setting.

3.2 Region of interest

To assist the feature extraction network in ignoring irrelevant features, a region of interest (ROI) was manually created by us for each training sample. A ROI is a rectilinear region centered around the relevant area of the image. Each ROI was created such that the driveline exit site is in the center of the ROI while making sure the region contains an area of healthy skin at least as large as the driveline wound area. See Figure 3.2 for an illustrated example of a ROI.

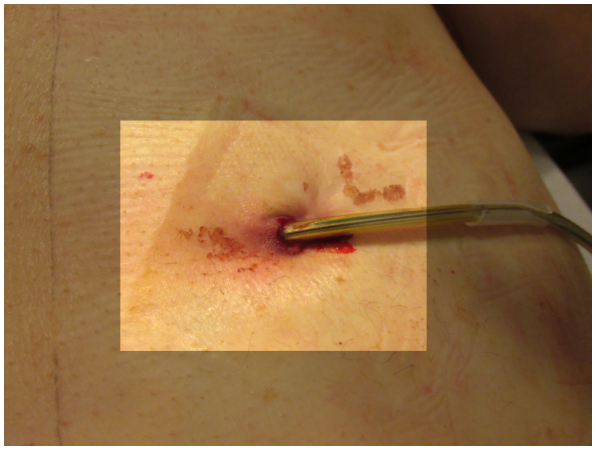


Figure 3.2: Example of a region of interest visualization.

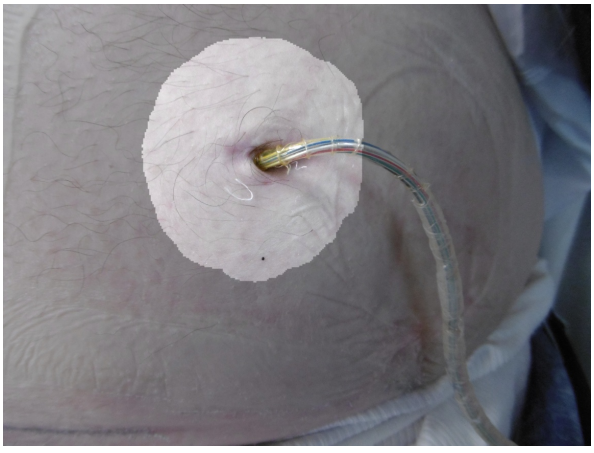


Figure 3.3: Example of a region of interest segmentation output mask.

3.3 Segmentation masks

Semantic segmentation, as described in Section 2.2, requires annotated segmentation masks. An open source annotation tool was adapted and made available to both of the heart clinics. It allowed the clinics to provide detailed geometrical descriptions of the driveline tube and other objects that obstruct the view of the wound/skin within each photograph. Manually annotating images is a time intensive process, therefore only a total of 144 images were annotated. The annotations consisted of convex or concave polygon representations of the visible part of the driveline tube and other occlusive objects. The classes in the annotated segmentation masks are imbalanced and not every image features every class, as described in Table 3.2.

Class	**Samples**	**Pixel portion (%)**
Background	144	93.7
Driveline tube	143	3.2
Occlusive objects	25	14.3

Table 3.2: Image statistics of annotated samples. The pixel portion column indicates the percentage of pixels that were assigned a specific class if it occurred in the image.

3.3.1 Train and test partitioning

The 144 annotated image samples were divided in a training set (115 images) and a validation set (29 images). This validation set was kept static throughout experiments to allow for comparing performance of multiple methods. Cross-validation was not used in driveline segmentation experiments to allow for comparison between multiple configurations, as it would lead to a significant increase in training time for supervised approaches.

Chapter 4

Methods

In this chapter experiments and model configurations are described, describing multiple methods to support infection type classification. Sections 4.1 - 4.2 focus on unsupervised and supervised segmentation methods for driveline tube identification and region of interest prediction. Two approaches of data augmentation to extend a small data set are explored. These are generative adversarial networks (Section 4.3) and geometric transformations (Section 4.4). Lastly, research on infection classification is described in Section 4.5, with a focus on the Inception v3 architecture.

4.1 Unsupervised segmentation

Unsupervised segmentation is used as an approach for driveline tube identification in patient photos. A successful identification of the driveline tube is achieved when there is a single segment which separates the part of the tube visible in the image from any other visible objects.

4.1.1 Felzenszwalb segmentation

When using the Felzenszwalb algorithm [10] (Section 2.2.1) it is not possible to specify the number of components in the resulting segmentation mask. In this application, a successful segmentation consists of two or more components which can be separated into background and tube components by postprocessing.

Before ground truth annotations for image segmentation were available, segmentation masks produced by the Felzenszwalb algorithm were visually inspected for quality. Based on a sample of segmentation masks, medical experts categorized each result as "good", "reasonable" or "bad", depending on how much of the driveline tube was correctly masked and how much of the wound area was erroneously included in the tube mask.

Preliminary experiments showed that applying the Kuwahara filter (Section 2.2.2) led to better segmentation results on average. After Kuwahara smoothing, a mild two-dimensional Gaussian filter is applied ($\sigma = 1.0$) for further smoothing, which compensates for digitization effects [10]. The scale parameter used in the threshold function of the algorithm, denoted as k, was varied in preliminary experiments to find a value that led to the best segmentation results on average across all input samples. The scale value that generalized best across all samples was 700.

The components in the output mask of the Felzenszwalb algorithm are unordered and have no direct meaning. Any components smaller than 1% of the total image area are merged until there are no small components left. The classification of background and tube components is done by a heuristic. The component that contains the most pixels at the border of the image is classified as the background component. All remaining components are then merged and classified as the

driveline tube component. This entails that there is no restriction on the connectedness of the driveline components, meaning that gaps in the tube region can exist.

The metric to determine the segmentation quality was the Dice coefficient score (Section 2.2.4), measured on the annotated images validation set, as described in Section 3.3.1. When using Felzenszwalb there are no probability values present in the output. Instead, after the postprocessing step of determining tube/background components, the two components were separated into two channels, each containing probability values of 1.0 for the pixels that were assigned the class of the respective channel. This allowed computing the Dice score coefficient between segmentation masks and ground truth.

4.2 Supervised segmentation

This section describes two supervised approaches to segmentation. The first experiment is driveline tube segmentation using the U-net network (Section 4.2.2). There were 144 annotated images available for supervised driveline segmentation. Due to the small number of annotated images and the need for accurate segmentation, an appropriate candidate CNN was the U-net architecture due to its cross-connections leading to more detailed segmentation masks [43].

The second segmentation experiment is ROI segmentation (Section 4.2.3). It builds on the driveline segmentation methods, using the same general U-net architecture.

4.2.1 U-net

In all U-net experiments zero padding was used at the image borders before each convolutional layer to preserve the dimensions of the image. Since 3x3 convolution kernels were used, reflection padding would not make sense for a 1-pixel border padding. This allowed concatenation of features via cross-connections without having to crop the image features. As an additional benefit, the output mask of the U-net network has a size identical to the input image.

In the binary segmentation setting it is possible to use a single channel for all classes in the output mask, in which values are normalized to the $[0,1]$ range with the sigmoid activation function for example. However, part of the segmentation experiments involved three classes in the output mask and therefore all experiments were set up with multiple output channels for a fair comparison, as opposed to using different activation functions for the binary and three-class setting. In the multi-class output configuration, each channel contained the probabilities for a particular object class. This allowed for the use of a pixel-wise softmax function so that the output probabilities of all classes within each element of the output mask sum up to 1.

4.2.2 Driveline tube segmentation

Two class configurations were used in the driveline segmentation experiments. The first configuration was a binary segmentation objective, in which each pixel is classified either as *driveline tube* or *other*. In the second configuration three classes were considered, in which the first class contained *background*, the second class contained *driveline tube* and the third class contained any additional *occlusive objects* (such as hands or partially removed wound dressing) which were not present in all annotated images. The three-class setting was aimed at discovering whether the trained U-net is capable of distinguishing different shapes/materials of objects, instead of classifying anything that is not tube-shaped as background.

To discover if the capacity of a U-net using the author's convolutional parameters [43] exceeds the capacity required to generalize beyond our data set, we trained a U-net in which the number of filters in each convolutional layer was halved with regard to the original values. The standard U-net could be

seen as unnecessarily complex for this data set if the network retains close to identical segmentation performance with half the number of filters. A smaller network is also less susceptible to overfitting and, in turn, less susceptible to poor generalization on test images.

Dilated convolutions (Section 2.1.5) were used to increase the perceptive field of the network without introducing additional convolution layers. If the perceptive field of the network is too narrow for larger segments, the network may not have access to contextual features around the segment, such as object edges. Larger segments occur in the class of *occlusive objects*, suggesting that dilated convolutions may be valuable when occlusive objects must be segmented as foreground objects.

As the Felzenszwalb algorithm benefits from smoothing the input image, various parameter settings for the Kuwahara filter were applied to images as the first step in U-net preprocessing to investigate the effect of edge-preserving smoothing in combination with a deep convolutional network.

The same validation set used in the unsupervised segmentation experiments (Section 3.3.1) was used for evaluating driveline segmentation quality.

Baseline configuration

The baseline U-net model has an architecture similar to that of the model proposed in [43], except for the addition of zero padding. The baseline model failed to converge on a small training set, which is why initial experiments were aimed at finding a stable baseline. A stable binary setting was achieved with an affine transformation augmentation factor of 8, 32 starting filters and class weights set to 0.1 and 0.9 for background and foreground respectively. By reducing the number of filters from 64 to 32 in the first convolution block, the number of filters in each subsequent block is also halved, as the number of filters in each block is doubled. The lower number of starting filters allowed for a larger batch size (16) and increased the training speed. The hyperparameters for the stable baseline are displayed in Table 4.1

Parameter	Value	Notes
Input size	512x512x3	
Pixel preprocessing	Standard scaling	Mean-centering and scaling to unit standard deviation.
Augmentation factor	8x	Number of augmented images as a factor of number of training samples.
Starting filters	32	Number of feature filters in the first layer.
Dilation factors	[1, 1, 1, 1, 1]	Dilation factor for each convolutional block in the contracting path.
Class weights	0.1, 0.9	Weight assigned to the background and foreground class, respectively, during optimization.
Optimization method	Adam [17]	
Learning rate	0.0001	
Epochs	75	
Batch size	16	Limited by GPU memory (NVIDIA Titan X 12 GB).

Table 4.1: Baseline hyperparameters for U-net semantic segmentation experiments. The configuration is binary, in which background is separated from foreground (driveline tube).

4.2.3 ROI segmentation

Region of interest segmentation is the process of predicting the ROI of the driveline exit site given an input image. We used a U-net architecture very similar to that used in the driveline tube segmen-

tation experiments, trained on manually labelled ROIs[1]. Since a ROI is a larger part of the image compared to the driveline tube, no cropping was done and the aspect ratio of the input image was retained. After preliminary experiments, the resolution was set at 384x288. Class weights were set uniform, as the pixel distribution between ROI and background was not extremely unbalanced. Since there were many more ROI-annotated images available compared to driveline-annotated images, the augmentation factor was set to 2.

In classification experiments, ROIs were required to be rectangular in shape. Figure 3.3 shows an example of a ROI output mask. In order to transform the "blobs" output by U-net to a rectangle, a postprocessing step was implemented in which a rectangle was fit on the U-net output by maximizing the overlap of the rectangle area and the output mask, constrained by a minimum rectangle size of 10% of the total image size.

The ROI segmentation performance was evaluated by computing the unweighted Dice coefficient score in a 5-fold cross-validation setting on 683 annotated images.

4.2.4 Dice score coefficient

When training a segmentation model, we are interested in the overlap of the predicted output mask with the corresponding label mask. The Dice score coefficient (see Section 2.2.4) was used as a measure of overlap during both training and evaluation. In multi-class segmentation the Dice score of each individual class mask is computed separately. A weighted average was used to aggregate the Dice scores of each class into a single value. The weights allowed us to assign importance to particular classes, as the classes in the labelled segmentation masks are unbalanced (Table 3.2).

4.3 Generative adversarial networks

In this section experiments are described in which generative networks were used to generate data samples. Samples generated from a learned data distribution can be used as data augmentation, as one can modify the latent variables to vary high-level feature representations. The training set consisted of 745 images at the time, of which 732 were labelled.

4.3.1 DCGAN

Since convolutional networks are suitable for image processing, a deep convolutional generative adversarial network (DCGAN, Section 2.3.1) is used for generating image data, in which we use upsampling layers in combination with convolutional layers in the generator and regular convolutional layers in the discriminator.

The DCGAN architecture by Radford and Metz [40] was used as a baseline. Their research provides guidelines for image generation based on empirical experiments. As training GANs is often unstable, these guidelines were used to confine the large space of hyperparameters to a narrower range.

If there are too many learnable parameters in the generator convergence may be affected. To investigate this, a 'compact' version of the generator was tested, which has two layers instead of four, with 128 and 64 filters in the convolution layers.

In the DCGAN experiments, label smoothing was explored. This technique has been shown to improve the quality of generated samples [44]. Instead of using values of 0 and 1 to represent fake or real, the labels are smoothed to lie in the interval between 0 and 1.

Learning in a GAN is realized by alternating generator and discriminator updates. In a default configuration, the number of updates for the generator and discriminator is equal for each epoch. However, one of the two networks could be converging at an accelerated rate compared to the other,

[1] My colleague, Michiel van de Steeg, was responsible for the experiments described here.

potentially leading to mode collapse (Section 2.3). For this reason, we experiment with a varying update ratio between generator and discriminator during training.

Ideally generated images are identical in resolution to the real images. However, a resolution of 128x128 is already considered high resolution when generated by a DCGAN architecture [36]. Experiments were set up to discover what the maximum achievable resolution was, which is defined as the highest resolution for which the network converged to a stable (local) optimum. As image resolution increases, the number of parameters in the network increases non-linearly, requiring more GPU memory during training and an increased difficulty for the model to converge. A significant increase in achievable resolution was not expected, as the DCGAN architecture was designed for images up to 128x128 resolution. Significantly higher resolutions likely requires more data and a different architecture and/or training approach, such as progressively growing GANs [14].

In addition to what is listed above, a number of other parameters are varied in the experiments. These parameters include standard scaling pixel preprocessing, dropout probability and the update ratio between generator and discriminator.

4.3.2 Latent space interpolation

Radford and Metz [40] demonstrate that the latent space of a DCGAN is semantically structured, allowing arithmetic on latent vectors. In the DCGAN experiments, some insight into the latent structure was achieved by interpolating between samples which were generated with random latent vectors. The hypothesis here was that if two random latent vectors lead to samples of sufficient quality, then the space between the samples must represent a smooth transition between the samples. Spherical interpolation was used to explore the latent space between two random points of the space.

4.3.3 Baseline configuration

After initial exploration with images of size 64x64 it appeared that the DCGAN architecture was able to converge with images of size 128x128. Therefore, this image resolution was set in the baseline parameters. Table 4.2 lists the baseline hyperparameters which were partially copied from [40].

Each convolution layer in the generator consists of an upsampling operation, a 3x3 convolution, batch normalization and a non-linear activation function. Each layer in the discriminator consists of a 3x3 convolution with stride 2 (except the last layer which has stride 1), batch normalization, a non-linear activation function and a dropout operation. The final feature map output is flattened and connected to a single validity output node via a fully connected layer. The validity output is binary, therefore its activation is fed through the sigmoid function.

4.3.4 FC-GAN

For augmentation purposes in a classification problem, generated data must have the correct labels. In order to generate labelled data using a DCGAN one can use a fast conditional GAN as explained in Section 2.3.2. During training, the infection labels of the real data are added as an auxiliary input to the generator. An artificial class for generated samples is added to the labels. In addition to a single real/fake output for the discriminator, the predicted class label is produced in an auxiliary output of the discriminator.

The DCGAN model baseline was used in the FC-GAN setting. In the generator an additional embedding layer was added which encoded the class label after which the embedded label was multiplied by the noise vector. Auxiliary multi-class output nodes were added to the discriminator and their values were constrained to the [0, 1] range by the softmax activation function.

Parameter	Value	Notes
Discriminator input size	128x128x3	
Augmentation factor	3x	Number of augmented images as a factor of number of training samples.
Pixel preprocessing	min-max scaling	Normalize values to the range [-1, 1].
Latent dimensionality	100	Size of the latent noise vector.
Generator upsampling	upsampling	Nearest interpolation + 1x1 convolution + ReLU.
Generator activations	ReLU	Activation function used in the generator.
Discriminator activations	Leaky ReLU	Activation function used in the discriminator.
Dropout	25%	Dropout probability after each layer in the discriminator.
Generator filters	[1024, 512, 256, 128]	Number of filters in each convolution layer of the generator.
Discriminator filters	[64, 128, 256, 512]	Number of filters in each convolution layer of the discriminator.
Optimization method	Adam [17]	
Learning rate	0.0002	
G:D ratio	2:1	Update ratio between generator and discriminator.
Iterations	12000	Number of batches trained on.
Batch size	32	

Table 4.2: Baseline hyperparameters for the DCGAN architecture.

An FC-GAN allows for generating multiple classes using a single generator so that features that are independent from the infection features can be learned independently of the features that correlate with the label encoding.

4.3.5 Quality of generated samples

Due to the disadvantages of the Inception Score metric mentioned in Section 2.3.3, the choice was made to assess generated image quality manually by inspecting the output of the generator. An indirect performance measure of generated samples is given by applying labelled generated samples as augmentation to training data in an infection classification experiment, as described below.

4.3.6 Training a classifier with GAN augmentation

Generated image samples from a DCGAN network are used as augmentation method by adding these samples to the infection classifier training data (Section 4.5.1). FC-GAN was not able to learn the data distribution at a reasonable image resolution for classification (192x192). Therefore, DCGAN was trained to generate image samples of only the largest class (no/minor infection). The other classes were augmented with images modified by affine transformations (Section 4.4) in a stratified fashion. This prevents a full comparison of GAN augmentation versus affine transformations, however, it allows for a comparison of the performance of only the no/minor infection predictions. The severe infection class would have been the best candidate to augment with realistic samples, however, due to the class imbalance there was not enough data available in this class to train a DCGAN with.

4.4 Data augmentation

For each of the experiments described in this section data augmentation was applied. The variance within the image training data was artificially increased by applying affine transformations. It was an attempt to distort images in a way that does not compromise relevant visual features. The amount of variance added depends on the number of affine operators applied to each image and to what extent the transformations modified the original image. Applied transformations consisted of: flipping (horizontal and vertical), rotation, translation and zoom. After applying translation, zoom or rotation values that are not a factor of 90 degrees, undefined pixels are introduced. These were filled by reflection padding (Section 2.1.1). See Figure 5.9 for a visual example of reflection padding.

An infinite number of generated images can be created by applying affine transformations. However, each generated sample remains a distorted version of the original, thus making it unlikely that the correlation between the number of generated samples and the network's ability to generalize exists for extreme application of data augmentation.

In balanced augmentation, more affine transformations of the classes that were least represented in the training data were created. This method results in an augmented data set with an equal class distribution. In the classification experiments, this entails that many affine transformations would be generated from the samples labelled as *severe infection*, as their representation in the original training data was very low (Chapter 3).

4.5 Infection type classification

Infection type classification is the process in which a patient photo is assigned one out of the three[2] infection classes. To accurately predict an infection class for unseen images a trained classifier is required. This section describes the details of the classifier and the approach used for training and validation.

4.5.1 Inception v3 pretrained

The available data set at the time of experiments consisted of 732 labelled images and each image has a resolution of approximately 16 megapixels, which means it may be difficult for a deep neural network to learn representations of the infection features embedded in the images. Therefore, the Inception v3 network (Section 2.1.7) was used for feature extraction.

Here, a version of the Inception v3 network pretrained on ImageNet [6] was used for feature extraction. The ImageNet data set contains over 14 million images divided across 1000 classes of common objects. Therefore, the pretrained Inception v3 network will have learned to extract high level features from a wide variety of images.

To use the pretrained network for infection classification the final fully connected layer of the network was removed, which is responsible for converting the activations from the last global pooling layer to probabilities for each of the 1000 ImageNet classes. The output of the global pooling layer is a one-dimensional vector of 2048 features. A logistic regression trained using a one-versus-rest approach was used to convert the extracted features into an infection class prediction. Because the classes in the training data are imbalanced the class weights in the logistic regression were set to be inversely proportional to the class distribution.

The default input size for the Inception v3 network for ImageNet data is 299x299. After preliminary experiments this was changed to 500x500 in the infection classification baseline model because of improved performance. Figure 4.1 demonstrates that images smaller than 299x299 and larger than 800x800 had a negative effect on classification accuracy. For the classification baseline an input size

[2]The two lowest infection classes were combined, see Chapter 3.

of 500x500 was chosen as a middle ground between prediction accuracy and memory efficiency. Pixel values in the RGB channels were normalized to be in the range $[-1, 1]$ to match the ImageNet input value range.

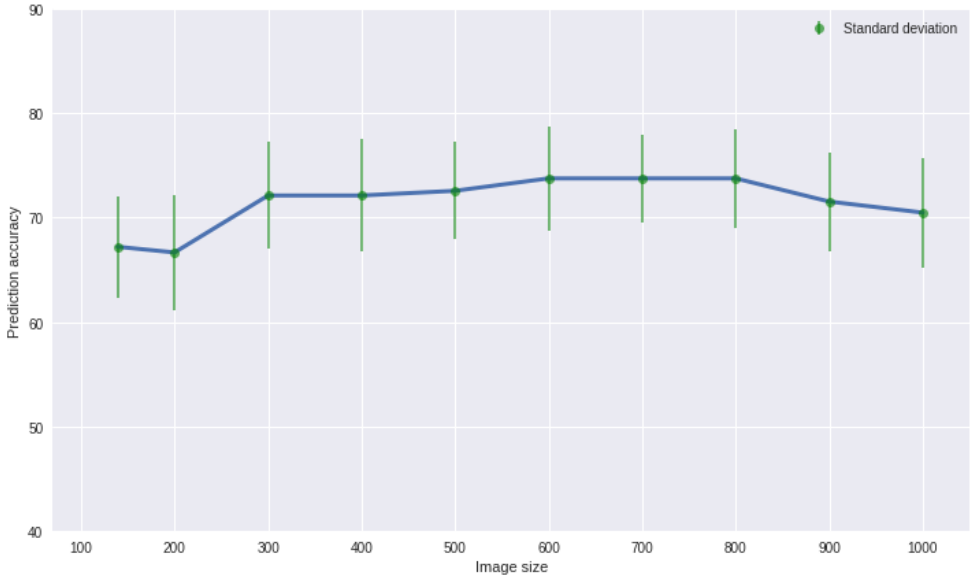


Figure 4.1: Relation between image size and infection classification accuracy in cross-validation using a pretrained Inception v3 feature extraction network.

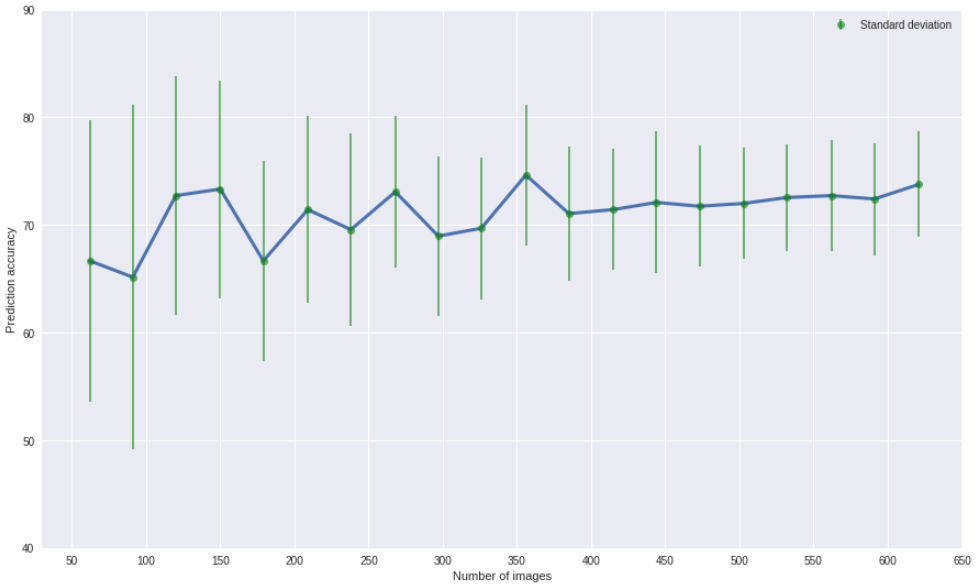


Figure 4.2: Relation between number of images in the training set and infection classification accuracy in cross-validation using a pretrained Inception v3 feature extraction network.

It is more fair to create a separate test set when evaluating different model configurations, however, the training set was small for a deep learning application, especially at the beginning of the project (Chapter 3). Preliminary experiments showed that adding more training samples slightly improved the classification accuracy when using a pretrained feature extraction network (Figure 4.2). More training data also led to a lower accuracy standard deviation between folds. The relation between number of images and classification performance supports the use of cross-validation for model evaluation.

Since the pretrained Inception v3 network was only used for feature extraction in the pretrained experiments, no further training of the network was done. Instead, the logistic regression weights were optimized using the feature vector output by Inception as input data. This process is much quicker than training a CNN which allowed us to apply repeated 10-fold stratified cross-validation (CV). It is identical to regular stratified CV, however the total collection of 10 folds is repeated, each time with random folds. The reason for repeated CV is to average out folds causing performance outliers, as with 10-fold CV each validation fold contained approximately 73 images.

With an imbalanced evaluation set, the average accuracy or F1 score across classes can be a deceiving metric. If the model is optimized towards correctly predicting the most frequent classes yet misclassifies the less frequent classes, then the accuracy will automatically be high. That is why the performance metrics that were recorded for each fold were the average accuracy over all classes, in addition to macro-average (unweighted) F1 score, which is an average metric that is not weighted proportional to the class distribution. The metrics per fold were aggregated by taking the median and calculating the standard deviation of the macro-average F1 scores.

4.5.2 Applying driveline segmentation masks

Binary driveline segmentation masks as described in Section 4.2 were used to mask the driveline tube in the input image. By masking irrelevant features in the classification input data, the classifier would not be 'distracted' by irrelevant features. The masking was done by making all pixels that were classified as belonging to the driveline tube black, before pixel value normalization. An alternative approach, in which the driveline pixels were set to a value of 0 after pixel normalization, led to a lower classification performance.

4.5.3 Inception v3 finetuning

The experiments described in this section explore transfer learning, where a pretrained Inception v3 network was finetuned on the LVAD training photos[3]. This allowed the network to learn to extract features relevant to LVAD data. Early layers in the pretrained network, which contain global features independent of the data set [54], were frozen while the later layers containing high level features were retrained to specialize in extracting the high level features relevant to the LVAD data set.

The transfer learning experiment was done in two stages. In the first stage, all layers of the Inception network were frozen and the connections from the final average pooling layer to the ImageNet classes layer were replaced by a fully connected output layer containing 3 nodes, representing one output for each infection class. Only the output layer was trained.

In the second stage of the experiment, the last two Inception modules (Section 2.1.7) were unfrozen to allow finetuning of the high-level feature extraction part of the network, along with the classification layer.

Experiments were done to find the optimal hyperparameters for the transfer learning experiments. These experiments fall outside the scope of this thesis, therefore, only the optimal configuration is described. Unless otherwise specified, the training configuration is the same as the pretrained Inception v3 baseline configuration (Section 4.5). Initial results showed that an input resolution of 500x500 did not work well in the finetuning experiments. Instead, a resolution of 512x384 was found to be optimal for classification performance. This resolution removed the need for center cropping, as the aspect ratio was the same for the majority of the training images. The region of interest crop was used during training and testing, as in the pretrained Inception v3 baseline.

Training of the top layer was done with a learning rate of 1×10^{-4} for 60 epochs. The finetuning of the inception modules was done with a lower learning rate (1×10^{-5}) for 70 epochs. In both stages

[3] My colleague, Pim van der Meulen, was responsible for the experiments described here.

the batch size was set to 60. Just as in the original Inception architecture, dropout was added to the final average pooling layer to prevent overfitting, with a probability of 0.2.

A stratified set of 139 images was used for evaluation, while the rest of the labelled images were augmented and used in training. The same metrics as in the pretrained experiments were measured; the macro-average F1 score and average accuracy. More detailed results are given by confusion matrices.

Chapter 5

Results

This chapter presents and discusses the results from the experiments described in Chapter 4. Section 5.1 contains the results on driveline tube and ROI segmentation, measured by the overlap between predicted and ground-truth segmentation masks. Section 5.2 presents the GAN experiments results, largely supported by illustrations. Section 5.3 lists the results of infection classification experiments with the Inception v3 network.

5.1 Segmentation

5.1.1 Felzenszwalb driveline segmentation

Figure 5.1a shows examples of driveline tube segmentation masks. The scale parameter value of 700 and a minimum component size of 1% of the total image area determined in preliminary experiments worked well for regular data samples. For data samples that deviated from the norm, such as patient photos displaying suture marks and stitches, the scale value of 700 sometimes resulted in a bad segmentation. However, no obvious pattern was found that gave insight into what scale value was suitable for data samples containing surgical revisions or other irregularities.

The median Dice score coefficient on the segmentation validation set was 0.49.

5.1.2 U-net driveline segmentation

An overview of the results of the driveline segmentation experiments on the validation set (containing 29 images) is given in Table 5.1. The median Dice score was used as the performance measure, since the Dice score distribution on the validation set contained outliers which would strongly affect the mean Dice score.

Generating additional training images using affine transformations increased the segmentation performance, to an extent. At an augmentation factor of 8x, the number of images in the training set is 1035. Adding even more data via augmentation had a negative effect on the validation set performance.

Adding dilated convolutions increased the performance compared to the baseline in most configurations. This may be due to the increased receptive field created by the dilated convolutions.

The larger network which starts at 64 filters in the first convolution layer achieved a slightly higher Dice coefficient score than the network starting with 32 filters, at a difference of 0.001.

Applying Kuwahara to the images did not improve the Dice score from the baseline score. A possible explanation for this result is that the U-net network attempts to find skin and plastic features to distinguish background from foreground, and the application of a Kuwahara filter results in washed out details which may represent important descriptors of the features.

The Dice score coefficient on a three-class segmentation setting was 0.6284. The three classes were *background*, *driveline tube* and *occlusive objects*. The Dice score coefficient was significantly lower than the binary baseline in the multi-class segmentation setting. This was expected, since the multi-class problem is more complex and only a portion of the training and evaluation images contained objects belonging to the *occlusive objects* class. Adding dilated convolutions improved the score to 0.6330, an increase of 0.0046 on the median. However, the binary baseline increased by 0.0085 after adding dilated convolutions and the binary baseline score was closer to the the maximum possible Dice score.

Configuration	Median Dice	Mean Dice	Std. Dice
Augmentation factors			
augmentation 1x	0.9248	0.8932	0.0921
augmentation 2x	0.9406	0.9085	0.0810
augmentation 3x	0.9364	0.9050	0.0882
augmentation 5x	0.9404	**0.9176**	0.0769
augmentation 8x	**0.9473**	0.9142	0.0871
augmentation 12x	0.9470	0.9160	0.0879
Dilation factors			
[1, 1, 1, 2, 3]	**0.9497**	0.9140	0.0899
[1, 1, 2, 2, 3]	0.9407	0.9109	0.0866
[1, 1, 1, 2, 4]	0.9450	**0.9203**	0.0812
[1, 2, 2, 2, 3]	0.9396	0.9120	0.0853
Starting filters			
64	**0.9504**	0.9088	0.1044
32	0.9494	**0.9236**	0.0799
3-class output: background, driveline, objects			
no dilation	0.6284	0.6292	0.0546
dilated; [1, 1, 1, 2, 3]	**0.6330**	**0.6368**	0.0573

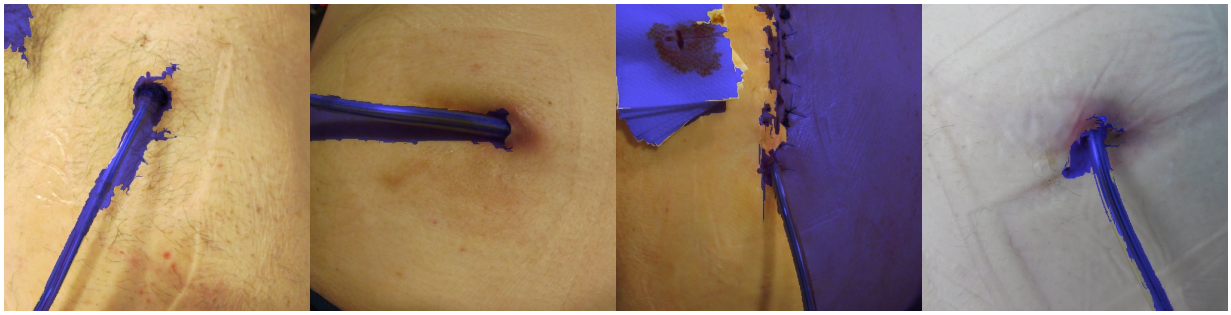
Table 5.1: Results of U-net configurations on the segmentation validation set which contains 29 images. Each of the settings was tested using the baseline model (Section 4.2.2, note that the *augmentation x8* setting is identical to the baseline), with the exception of the *starting filters* configurations. These used class weights of 0.03 and 0.97 for background and foreground, respectively.

The authors of [43] claim U-net is able to produce precise segmentation masks, which is shown in Figure 5.1b. The segmentation masks match the borders of the tube object well. The driveline tube is identified correctly in most cases with very few false positive pixels.

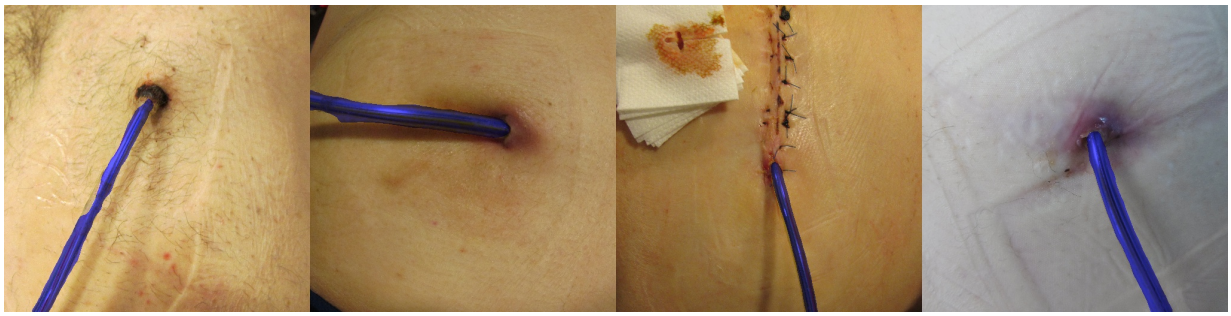
Figure 5.2 shows examples of U-net segmentation trained in a 3-class setting, where annotations of foreground objects other than the driveline tube were assigned a separate class. The tube segmentation in the multi-class setting is in some cases negatively affected by the added complexity, however, if we compare the third example from the left in Figure 5.1b with Figure 5.2b it is clear that the network is able to correctly identify the visible dressing as a separate entity in this case.

5.1.3 U-net ROI segmentation

The Dice score coefficients of each sample in each of the five CV folds were combined. The mean Dice score of all test samples in ROI segmentation was 0.8402 (median 0.8579) with a standard deviation of 0.090.



(a) Felzenszwalb segmentation masks, at $k = 700$ and $\sigma = 1.0$. The algorithm sometimes produces far from optimal results for complex images.



(b) U-net binary segmentation masks. The examples show that U-net is able to produce accurate segmentation masks for images that are visually more complex.

Figure 5.1: Examples of driveline segmentation masks from (a) Felzenszwalb and (b) U-net. The blue overlays represent the driveline tube component, the rest of the image is classified as background.

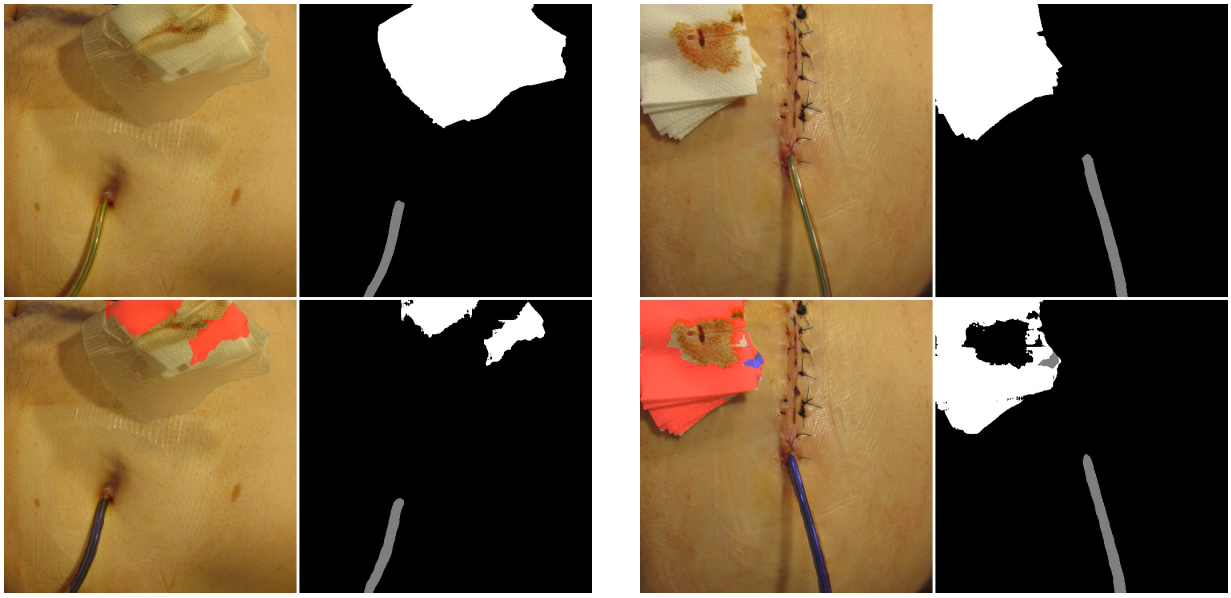
Comparing the Dice score coefficients of the ROI segmentation experiments to the driveline tube segmentation Dice scores is not reasonable, because there is a difference in ground-truth mask detail between the two segmentation experiments. The ground-truth masks for driveline segmentation contain the exact shapes of the objects to be segmented, however, the ground-truth masks for ROI segmentation were less strictly defined.

5.2 Generative adversarial networks

5.2.1 DCGAN experiments

To evaluate the convergence of generative adversarial models during training, every 75 iterations nine samples were generated. Figure 5.3 shows generated samples from the baseline model. After 600 iterations it was clear that the network had learned to generate skin-colored images with a distinct blob near the center. After 6000 iterations samples were generated that resemble photos of exit sites, while driveline tubes were missing. By 12000 iterations generated samples generally included at least part of a driveline tube. The network was further trained beyond 12000 iterations to see if the quality of generated samples kept improving. It was more difficult to judge the difference in quality beyond 12000 iterations. At 60000 iterations it seemed that the visual features in the images were more clearly defined, however, not in all cases. As is visible in Figure 5.3d, some generated samples still lacked a realistic structure. After convergence it appeared that the baseline model was capable of learning that the background of the image should consist mainly of skin features and that the image should contain a driveline tube which exits from the skin at a wound area near the center of the image.

The ‘compact’ version of the generator which contains two layers instead of four was still able



(a) (b)

Figure 5.2: Examples of U-net segmentation masks in a multi-class setting (three classes). Top left: Original image (center square cropped). Top right: ground truth mask. Bottom right: U-net output mask. Bottom left: U-net mask overlayed on original image, where the blue area is the driveline tube and red areas include other objects that obstruct the view of the skin.

to generate samples of similar quality as samples generated by the baseline model (Figure 5.4a). A difference was noted early in training, where the model was not able to generate skin-like features in the background unlike the baseline model. This observation suggests that fewer parameters in the generator may lead to slower convergence with regard to sample quality.

Label smoothing during DCGAN training was found to have an adverse affect, contrary to the claims in [44]. Label smoothing was applied in two configurations with probabilities of 5% and 15% for smoothing to occur, which modified the validity labels up to 0.4 in the direction of the opposite label. In both configurations the generator did not converge to produce realistic samples.

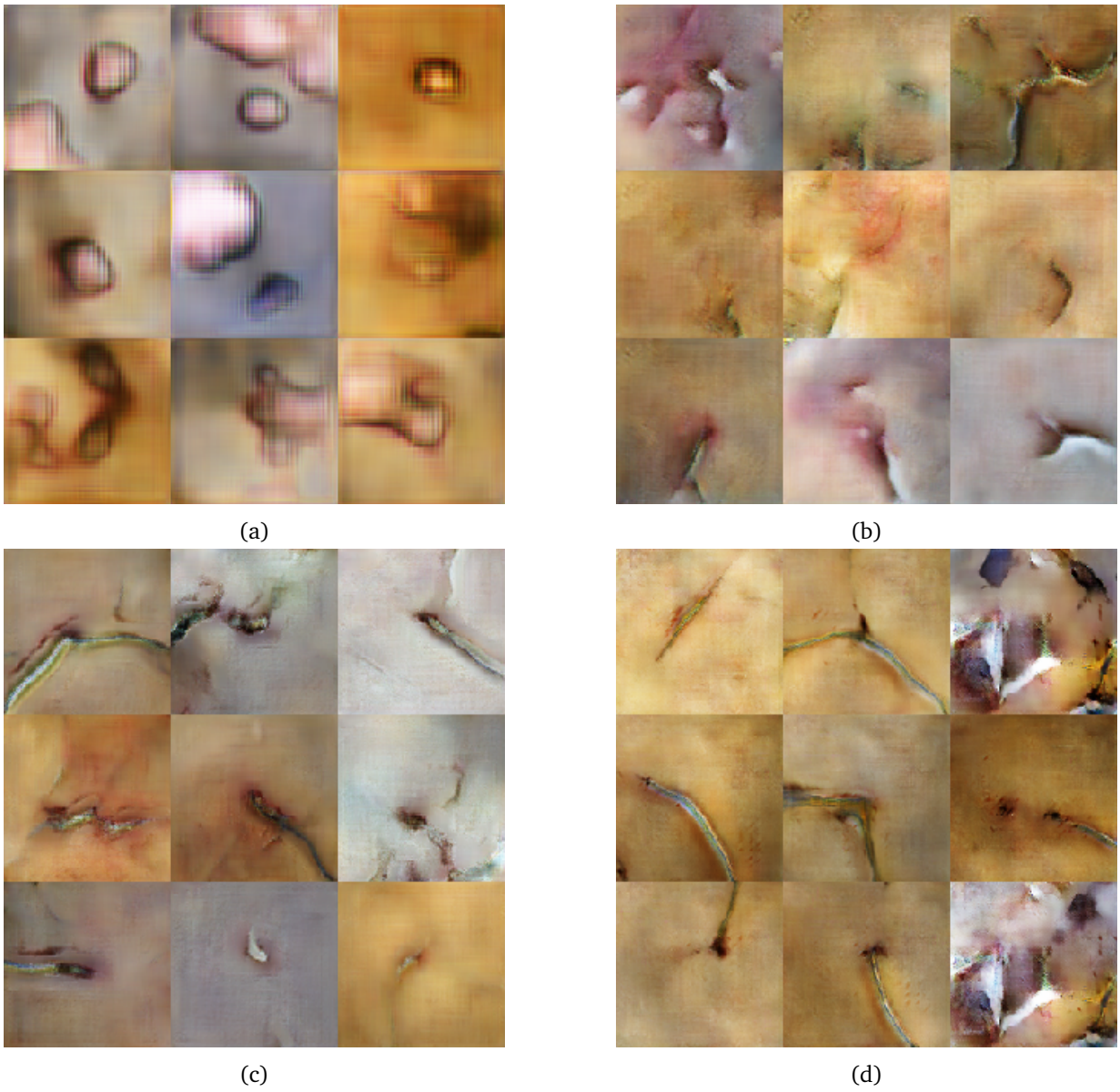
The baseline DCGAN used a generator to discriminator update ratio of 2:1. The network still converged at a ratio of 3:1, however the sample quality appeared to be slightly lower than the baseline.

A varying amount of dropout after each layer in the discriminator was tested, up to a maximum of 50%. The baseline had 25% dropout. The DCGAN was still able to converge at 50% dropout (Figure 5.4b), however the objects in the generated images were less defined. At a dropout probability of 10% the train run resulted in mode collapse, suggesting that dropout is important to prevent the discriminator from outperforming the generator.

The unstable nature of a DCGAN became clear when experimenting with minor changes to the configuration. When the activation functions within the generator were changed from ReLU (baseline) to leaky ReLU the generated samples after 12000 iterations were significantly worse than the baseline as shown in Figure 5.4c. Similarly, the network failed to converge when the training data was normalized via mean centering and standard deviation unit scaling as opposed to normalizing each pixel to the [-1, 1] range.

Augmenting the training data by using affine transformations helps to artificially enlarge the size of the training set, however, the DCGAN may be harmed by the unnatural features created in certain transformations. Some examples of affine transformations are given in Figure 5.9. Using reflection

padding makes sure that the distribution of artificially created pixels is similar to that of real pixels. However, the reflected area may contain unnatural features, such as multiple driveline exit sites. If these features are learned by the DCGAN, the generator may generate images with multiple driveline tubes which a human might consider to be flawed examples, while in reality the distribution of the generated images still matches the augmented training data.



(a) (b) (c) (d)

Figure 5.3: Generated samples from the DCGAN baseline, after (a) 600 iterations, (b) 6000 iterations, (c) 12000 iterations, (d) 60000 iterations. Each image is 128x128 in resolution.

For non-adversarial neural networks, analyzing the error over time during training is often a good way to track convergence. Figure 5.5 shows that interpreting the error metrics of an adversarial model is not trivial. The errors of the discriminator and generator are at their lowest values after approximately 15000 iterations. After this point, the generator error increases, however, this could indicate that both the generator and discriminator improve, even though the discriminator might be improving at a faster rate than the generator.

(a) Compact generator (2 layers) (b) 50% dropout in discriminator (c) Leaky ReLU in generator

Figure 5.4: Generated DCGAN samples from multiple configurations after 12000 iterations of training. Each image is 128x128 in resolution.

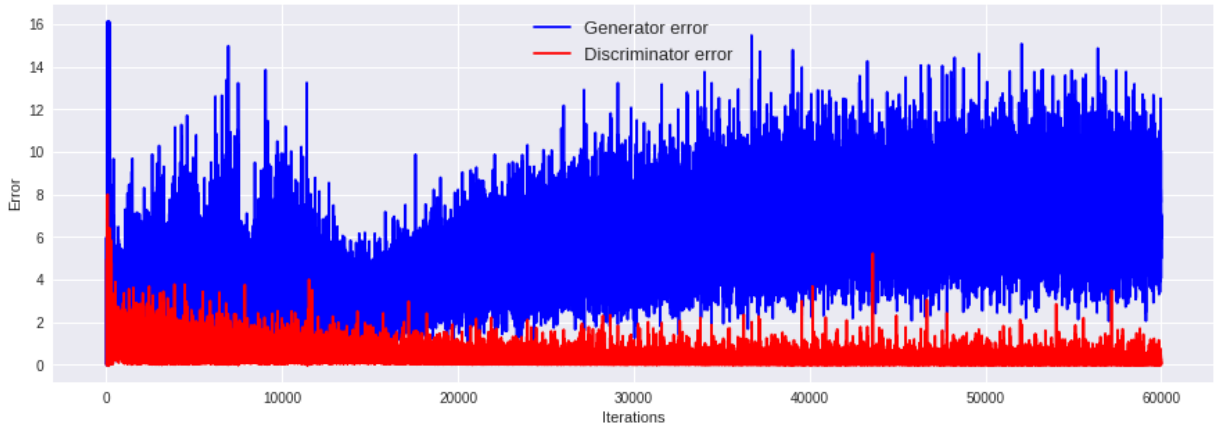


Figure 5.5: DCGAN baseline cross-entropy error during training.

High resolution

The maximum resolution that still allowed the DCGAN architecture to converge was 192x192, compared to the 128x128 baseline. This means the image dimensions in the first convolution layer of the generator was 12x12. Any resolution higher than 192x192 prevented the DCGAN from learning, even when adding an extra layer to the generator to reduce the image resolution in the first layer. The higher resolution required some changes to fit training batches in GPU memory. The batch size was reduced to 25 and the number of filters in the layers of the generator were set to [500, 250, 125, 62]. Figure 5.6 shows examples of generated images at 192x192 resolution. The image quality is arguably lower than the baseline generated samples at 128x128, so a trade-off exists between image quality/diversity and resolution.

Latent space interpolation

An example of latent space interpolation is given in Figure 5.7. A smooth transition between the two randomly generated samples is observed. The driveline tube first disappears and later appears in a different orientation. In addition, a change in skin color is observed.

Figure 5.6: Samples generated from high resolution DCGAN (192x192) after 46000 iterations.

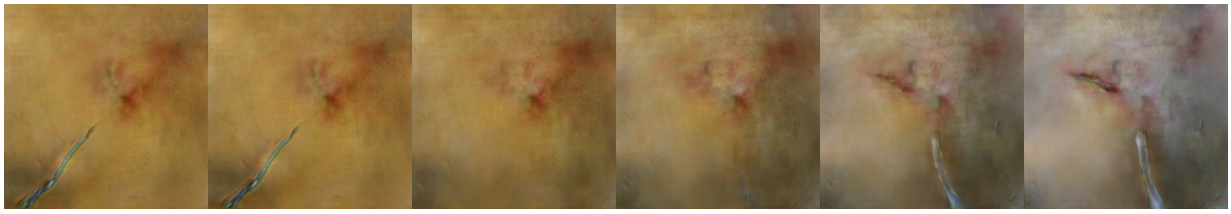


Figure 5.7: Latent space spherical interpolation between two generated samples from DCGAN at 192x192 resolution. The left-most and right-most images were generated using random latent vectors.

5.2.2 FC-GAN experiments

The FC-GAN model was not able to learn the distribution of the training data at a resolution of 128x128. At a resolution of 64x64 the model was able to converge and generate samples with specified class labels. Figure 5.8 shows a number of examples for each class. Mode collapse has occurred in the *severe infection* class which is likely due to the large imbalance in class labels in the training data. Only approximately 5% of the training images were labelled as severe infection, which is likely not enough for the FC-GAN model to learn to generate samples of sufficient quality while at the same time learning to separate severe infection features from the other two classes.

5.3 Infection classification

5.3.1 Inception v3 pretrained

Table 5.2 shows the cross-validation results on infection classification with a combination of Inception v3 and a logistic regression. The table contains both macro-F1 and accuracy metrics. A discussion of the results of each configuration is found below.

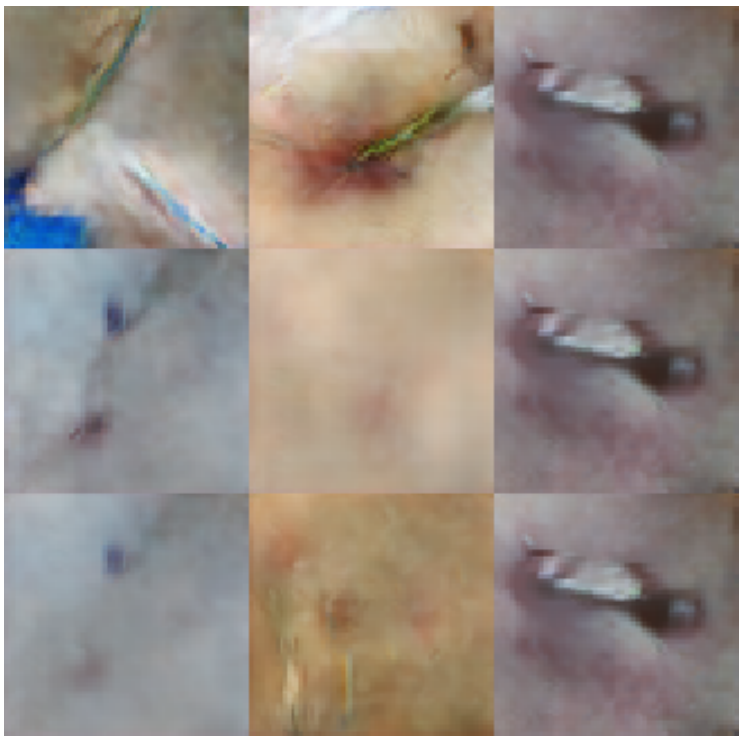


Figure 5.8: Samples generated by the FC-GAN architecture after 50000 iterations at a resolution of 64x64. *Left column*: no/minor infection class, *middle column*: mild infection class, *right column*: severe infection class in which mode collapse is observed.

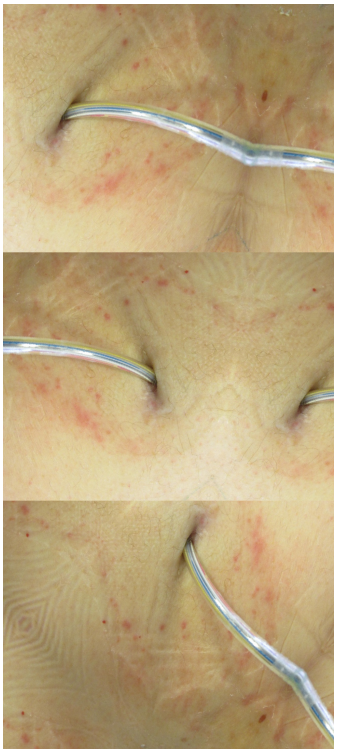


Figure 5.9: Examples of random affine transformations applied to a single image. Making use of reflection padding means that transformed images can contain several driveline exit sites kinks in the driveline tubes.

Augmentation

The standard method of training data augmentation was applying affine transformations to each image, where the augmentation factor determines how many random transformations were created for each image. Table 5.2 shows that an augmentation factor of 5 leads to the best F1 score on a data set containing 732 images. At an augmentation factor of 2, balanced augmentation performs better than standard affine augmentation. An increase in prediction performance on the severe infection validation samples was expected, as the balanced training set contained many more images of this class compared to baseline augmentation. The confusion matrices in Figure 5.10 show that indeed images were more often correctly classified as severe infection when using balanced augmentation, however, the classifier seems to have become more biased towards predicting severe infection, as the other classes were more often misclassified as severe infection compared to the baseline.

The baseline confusion matrix in Figure 5.10a indicates why there is a large difference in value between macro-average F1 and average accuracy metrics. The classes that occur more frequently in the training data were classified correctly more often proportional to the least frequent class. The fact that F1 scores across classes were unequal when using balanced augmentation does not prove that the class balance in training data is irrelevant. Augmentation by affine transformations is not the same as adding unique data samples. However, it is still possible that the less represented severe

Configuration	F1	σ_{F1}	Accuracy
Augmentation			
augmentation 1x	0.472	0.078	0.719
augmentation 2x **[b]**	0.481	0.078	0.716
augmentation 3x	0.481	0.079	**0.725**
augmentation 5x	**0.484**	0.088	0.716
augmentation 40x	0.476	0.080	0.712
balanced 2x	**0.484**	0.074	0.693
Region of interest			
ROI cropped **[b]**	**0.481**	0.078	**0.716**
Full image	0.445	0.075	0.676
GAN Augmentation (192x192)			
affine augmentation x2	0.431	0.080	**0.667**
GAN augmentation x2	0.434	0.079	0.642
affine augmentation x10	**0.451**	0.085	**0.667**
GAN augmentation x10	0.448	0.072	0.567
Driveline masking			
Driveline visible **[b]**	**0.481**	0.078	**0.716**
Driveline masked	0.466	0.069	0.712

Table 5.2: Results of multiple infection classification configurations on 10-fold repeated stratified cross-validation using a pretrained Inception v3 network in combination with a logistic regression classifier. The **[b]**-tag indicates the baseline configuration. The F1 score is the macro-average across all classes.

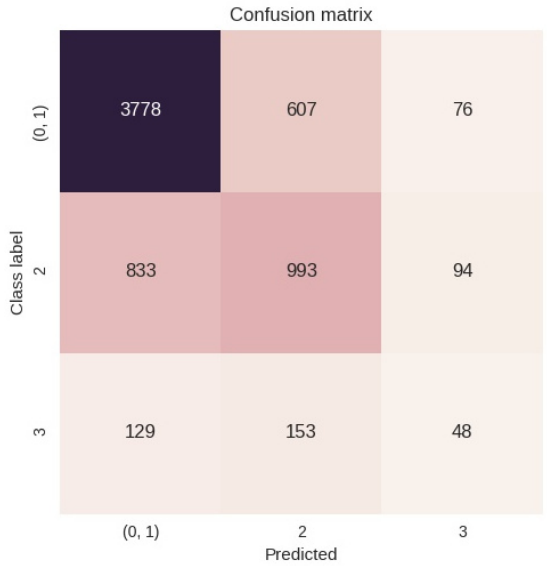
infection class is harder to distinguish from other classes because there simply are fewer visual clues in the image that define it as a severe infection.

Region of interest

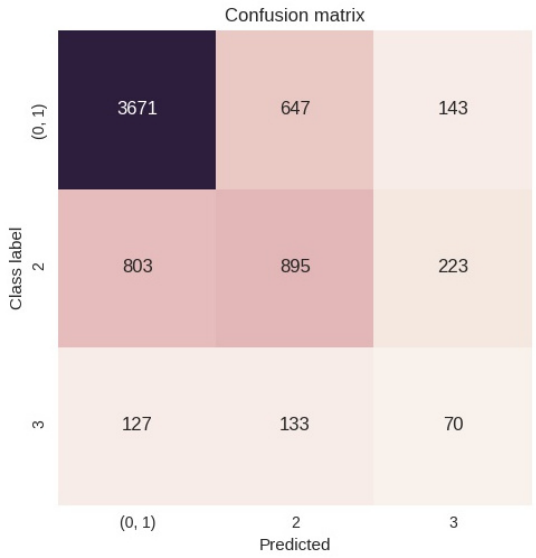
A comparison was made between the baseline classification model, which used the ROI crops as input images, and a configuration in which the full images were used. The comparison in Table 5.2 shows that cropping the ROI is beneficial for infection classification. An explanation for this difference is that the ROI crop contains fewer segments of background and more explicitly, fewer segments that are uncorrelated with the driveline wound. This allows the logistic regression classifier to ignore irrelevant background features as these should occur less frequently in the feature vectors generated by the pretrained Inception v3 network.

GAN data augmentation

The configuration with a GAN-augmented data set (Section 4.3.6) cannot be directly compared to the classification baseline as the resolution of GAN samples was 192x192. Therefore, the baseline was adapted to use an input size of 192x192. Table 5.2 includes a comparison between affine and GAN-based augmentation for two augmentation factors. From the reported metrics it appears that a data set augmented with GAN generated images yields similar performance metrics compared to the baseline, however the conclusion of the figures is not straightforward. As mentioned in Section 4.3.6, GAN images could only be generated for a single class (no/minor infection), all other classes were



(a) Each training sample is augmented twice, not affecting the class distribution



(b) Samples are augmented such that class labels are balanced

Figure 5.10: Confusion matrices of 10 times 10-fold CV on infection classification with an augmentation factor of 2. Class labels are *(0, 1)*: minor/no infection, *2*: mild infection, *3*: severe infection.

augmented with affine transformations. At an augmentation factor of 10x, 90.9% of all images in the no/minor infection class were generated samples, which means that the classifier may have learned to recognize whether an image is a GAN-generated example instead of belonging to the no/minor infection class. The effect on classification performance can only be determined accurately if all classes are augmented with GAN samples.

Driveline masking

The bottom part of Table 5.2 shows the comparison between the baseline model and the experiment in which driveline tubes were masked by black pixels. The baseline model performs better with regard to F1 score and accuracy. There are multiple possible explanations for this. The Inception v3 classification network was pretrained on ImageNet, which likely means it does not know how to handle sections of black pixels in natural images. The structure of the extracted feature vectors may suffer because of the black pixels, affecting the embedded features that correlate highly with the type of infection. Even if the network was finetuned on data containing masked sections, the sharp transitions between the masked sections and the rest of the image may have been difficult to learn to ignore by a CNN. Lastly, the generated segmentation masks are not always perfect, in other words, the errors from the segmentation network propagate to the classification model.

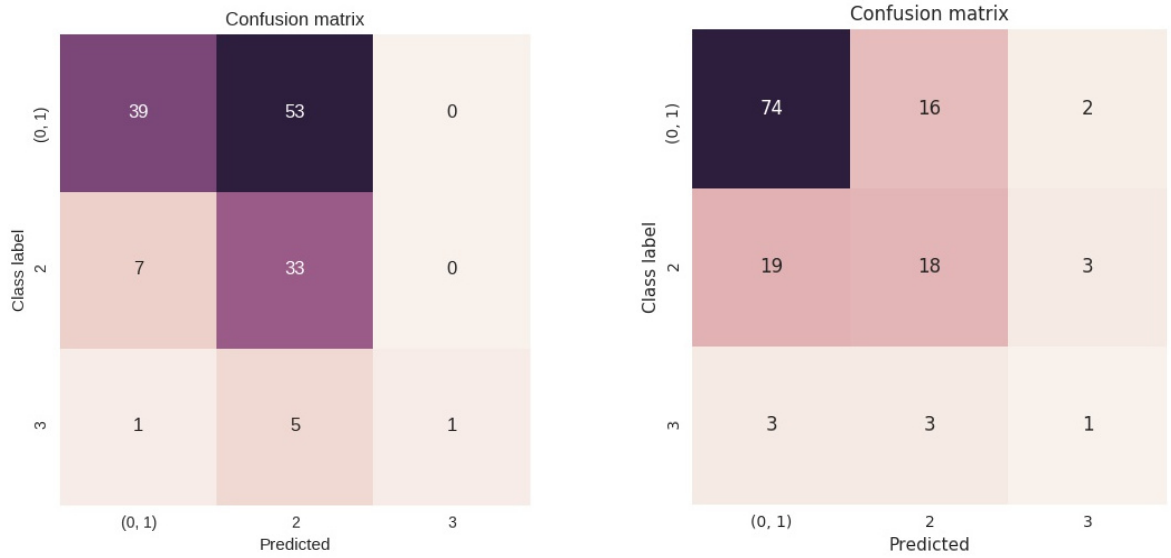
5.3.2 Inception v3 finetuned

The results of the two stages of the Inception v3 finetuning experiments are displayed as confusion matrices in Figure 5.11. Some differences can be observed between the two stages. After training only the top classification layer, the model seems biased towards predicting samples as class 2, whereas after finetuning the Inception layers, the model seems more biased towards predicting samples as class 1. Additionally, after finetuning the Inception modules, the model is more likely to incorrectly

predict images as class 3, leading to a lower precision on this class. The recall on class 3 is identical in both stages.

The increase in performance from stage 1 to stage 2 may be solely accredited to the classification layer, which is trained in both stages. However, training only the classification layer for the same number of epochs as the total epochs of both stages resulted in lower performance measures, compared to finetuning the Inception modules. This entails that the classification layer was trained to convergence in stage 1.

From these results it cannot be said whether the pretrained Inception v3 plus logistic regression or the finetuned version of the Inception network is more suitable for infection classification. The macro F1 score of 0.470 obtained in stage 2 finetuning is lower than the macro F1 of the baseline of the pretrained experiments (Table 5.2), however, it is a comparison of a single train-validation fold result to a 10 times 10-fold cross-validation result. Given that the standard deviation on the pretrained baseline result is 0.078, the difference between the F1 scores of the pretrained and finetuning configurations is not significant.



(a) Stage 1: Only the top classification layer is trained from scratch. Macro F1: 0.438; average accuracy: 0.525.

(b) Stage 2: The final two Inception modules, in addition to the classification layer, are finetuned. Macro F1: 0.470; average accuracy: 0.669.

Figure 5.11: Confusion matrices of the evaluation set of 139 images for two stages of Inception v3 finetuning.

Chapter 6

Conclusion

In this thesis a deep learning approach for classification of skin infections in patients with heart assist devices (LVADs) was explored. A number of different methods for getting more value out of small data sets were investigated.

6.1 Segmentation

6.1.1 Unsupervised segmentation

Without ground truth data for segmentation, unsupervised segmentation methods such as the Felzenszwalb algorithm are suitable for coarse image segmentation. However, it proved difficult to generalize the method to obtain accurate segmentation masks for each image in the evaluation set. A successful unsupervised segmentation approach would likely involve tuning the parameters of the method based on the properties of any particular image. Additionally, postprocecssing was required to convert a generated segmentation mask into a semantic segmentation mask.

6.1.2 Supervised segmentation

Experiments with a U-net CNN confirmed that it is a powerful tool for accurate driveline segmentation without requiring a large data set, as claimed in [43]. A Dice score of 0.956 was achieved on a binary segmentation task (driveline tube versus background). It was demonstrated that U-net was partially capable of multi-class semantic segmentation, even though most training images only contained two classes.

In addition to driveline segmentation, it was shown that the U-net architecture and hyperparameters were transferable to a significantly different semantic segmentation problem: region of interest segmentation, where the ROI is the area around the driveline exit site.

As an alternative method to direct ROI segmentation, accurate driveline segmentation masks could be used to discover the region of interest in non-standardized LVAD photos. The only assumption that would need to be made is that a driveline tube enters the image from one of the borders and ends at the driveline exit site, which is where features are located which correlate with infection type. Whether this alternative to ROI segmentation is worth the effort is left for future research. It likely depends on which problem is easier to solve for a U-net; directly locating the driveline exit site, or creating an accurate segmentation mask of the driveline tube.

6.2 Data augmentation

The first part of research question 1, which asks what classification performance gains are discovered when applying affine transformation augmentation to the training data set, can now be answered. Experiments with supervised segmentation, generative adversarial networks and infection classification all indicated that training data augmentation by method of random affine transformations helps during training, as suggested by earlier research [38]. Transformed images could be seen as distorted copies of the original, however, they give a model opportunities to learn from identical features occurring in varying shapes, orientations and positions.

The number of augmented images added to the training set affects the performance of a trained model. An infinite number of random affine transformations exists, however, these suffer from diminishing returns as the variety in generated images does not increase linearly with the number of transformed training samples. No heuristic was found for determining the optimal augmentation factor. For infection classification a factor of 2 yielded the best prediction performance, while a factor of 8 worked best in the semantic segmentation experiments. The original number of training images was almost four times as large in the classification experiments compared to the segmentation training data, which could explain the difference in the best found augmentation factors.

6.3 Generative adversarial networks

Despite the fact that generative adversarial networks are generally difficult to train and display unstable convergence characteristics, the DCGAN architecture was applied to learn the data distribution of LVAD infection photos. A proper evaluation metric for generated images was not available, therefore it was hard to evaluate the quality and variation of DCGAN models. As affine transformations were used to augment the DCGAN training set, the DCGAN generator was essentially instructed to learn unnatural infection features, such as multiple driveline exit sites. However, without affine transformations to augment the training data the data set was too small to be learned by a DCGAN architecture.

In a standard DCGAN configuration an image resolution of 128x128 for generated samples is considered high resolution [36, 40]. The maximum resolution generated by the DCGAN architecture in this research was 192x192, which is still not optimal to be used as augmentation data for infection classification (Figure 4.1). The second part of research question 1 can therefore be answered. Without being able to generate realistic class-conditional images of sufficient resolution, the classification performance suffers when augmenting training data with GAN-generated images.

The FC-GAN architecture was not successfully applied to the data set available during this research. The data set was heavily imbalanced and the class-conditional aspect of an FC-GAN likely made convergence during training too complex.

6.4 Infection classification

A number of infection classification configurations which made use of the Inception v3 CNN, pretrained on ImageNet, were explored. Experiments demonstrated that a feature extraction network trained on a very different data set was still able to extract generalized feature vectors that are indicative of infection type, and can be used by a simple logistic regression to make predictions on infection class. The motive behind using a pretrained feature extraction network was the fact that the LVAD data set was likely too small and imbalanced for training a deep feature extraction CNN from scratch.

Finetuning the high-level feature extraction part of the Inception v3 network on infection classes resulted in similar classification performance as in the pretrained Inception classification experiments.

On a single validation set, the performance increased when finetuning high-level Inception modules when compared to only training the classification layer.

Applying segmentation masks to input images did not lead to improved classification performance, however, the method of masking the driveline tube was very basic. Improved results may be achieved by masking the tubes in a more natural manner, such as reflection filling along the axis of the tube. Research question 2, which asks what performance increase is realized by segmenting the driveline tube during classification, is therefore still open for future research.

Despite the fact that the segmentation masks could not be applied directly, the experiments showed that using a region of interest (ROI) led to a higher classification performance compared to using full images. Therefore, the ROI segmentation U-net used to infer the ROI within images would be a valuable component in a production setting.

The difficulty of the infection classification task became apparent in classification experiments, as a maximum macro-average F1 score of 0.484 was achieved. A likely explanation is the scarce representation of the severe infection class in the training set. However, it is also possible that assessing the infection type based only on a patient photo is a difficult problem even for humans. To gain insight into this hypothesis, a stratified set of 100 images was selected from the training set, unlinked from their class label and presented to two LVAD experts for visual evaluation. Both experts achieved a macro F1 score of 0.56 (average accuracy of 0.66 and 0.69). This implies that without additional information besides the patient photos, it is difficult even for medical experts to correctly predict the infection type, possibly impossible in some cases. A possibility for future research is to combine patient photos with thermographic scans which capture the infrared channel and can reveal sub-surface heat sources which are often associated with skin infections.

Acknowledgments

I would like to thank Dr. Ioannis Giotis for introducing me to the medical image processing field, Dr. Rolf Neubert for supporting scientific research within the Medolution project, Sybren Jansen for his valuable ideas and input on various algorithms and models, Dr. Marco Wiering for providing a clear direction to the experiments and a critical view on the scientific writing, and my colleagues at Target Holding for creating an enjoyable working environment.

Bibliography

[1] Kumar Abhishek and Ghassan Hamarneh. Mask2lesion: Mask-constrained adversarial skin lesion image synthesis. *arXiv preprint arXiv:1906.05845*, 2019.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

[3] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. *arXiv preprint arXiv:1412.7062*, 2014.

[5] Noel CF Codella, David Gutman, M Emre Celebi, Brian Helba, Michael A Marchetti, Stephen W Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, et al. Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (ISBI), hosted by the international skin imaging collaboration (ISIC). In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 168–172. IEEE, 2018.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

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

[8] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[9] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.

[10] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, September 2004.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[13] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[14] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[15] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[16] Jeremy Kawahara, Aicha BenTaieb, and Ghassan Hamarneh. Deep features to classify skin lesions. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pages 1397–1400. IEEE, 2016.

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[19] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.

[20] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[22] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[23] M Kuwahara, K Hachimura, S Eiho, and M Kinoshita. Processing of ri-angiocardiographic images. In *Digital processing of biomedical images*, pages 187–202. Springer, 1976.

[24] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

[25] Chengcheng Li, Zi Wang, and Hairong Qi. Fast-converging conditional generative adversarial networks for image synthesis. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2132–2136. IEEE, 2018.

[26] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[27] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[28] Adria Romero Lopez, Xavier Giro-i Nieto, Jack Burdick, and Oge Marques. Skin lesion classification from dermoscopic images using deep learning techniques. In *2017 13th IASTED International Conference on Biomedical Engineering (BioMed)*, pages 49–54. IEEE, 2017.

[29] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.

[30] Amirreza Mahbod, Gerald Schaefer, Chunliang Wang, Rupert Ecker, and Isabella Ellinge. Skin lesion classification using hybrid deep neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1229–1233. IEEE, 2019.

[31] Ammara Masood and Adel Ali Al-Jumaily. Computer aided diagnostic support system for skin cancer: a review of techniques and algorithms. *International journal of biomedical imaging*, 2013, 2013.

[32] Danilo Barros Mendes and Nilton Correia da Silva. Skin lesions classification using convolutional neural networks in clinical images. *arXiv preprint arXiv:1812.02316*, 2018.

[33] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[34] Jonas Natten. Generative Adversarial Networks for Improving Face Classification. Master's thesis, University of Agder, Norway, 2017.

[35] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 1(10):e3, 2016.

[36] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. *arXiv preprint arXiv:1610.09585*, 2016.

[37] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019.

[38] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[39] Sean P Pinney, Anelechi C Anyanwu, Anuradha Lala, Jeffrey J Teuteberg, Nir Uriel, and Mandeep R Mehra. Left ventricular assist devices for lifelong support. *Journal of the American College of Cardiology*, 69(23):2845–2861, 2017.

[40] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[41] Lillian J Ratliff, Samuel A Burden, and S Shankar Sastry. Characterization and computation of local Nash equilibria in continuous games. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 917–924. IEEE, 2013.

[42] Nils Reiss, Thomas Schmidt, Frerk Müller-von Aschwege, Wolfgang Thronicke, Jan-Dirk Hoffmann, Jenny Inge Röbesaat, Ezin Deniz, Andreas Hein, Heiko Krumm, Franz-Josef Stewing, et al. Telemonitoring and medical care of heart failure patients supported by left ventricular assist devices-the medolution project. In *eHealth*, pages 267–274, 2017.

[43] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[44] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training GANs. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc., 2016.

[45] David W Shattuck, Stephanie R Sandor-Leahy, Kirt A Schaper, David A Rottenberg, and Richard M Leahy. Magnetic resonance image tissue classification using a partial volume model. *NeuroImage*, 13(5):856–876, 2001.

[46] Margarida Silveira, Jacinto C Nascimento, Jorge S Marques, André RS Marçal, Teresa Mendonça, Syogo Yamauchi, Junji Maeda, and Jorge Rozeira. Comparison of segmentation methods for melanoma diagnosis in dermoscopy images. *IEEE Journal of Selected Topics in Signal Processing*, 3(1):35–45, 2009.

[47] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[48] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 240–248. Springer, 2017.

[49] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[50] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[51] Changhan Wang, Xinchen Yan, Max Smith, Kanika Kochhar, Marcie Rubin, Stephen M Warren, James Wrobel, and Honglak Lee. A unified framework for automatic wound segmentation and analysis with deep convolutional neural networks. In *2015 37th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, pages 2415–2418. IEEE, 2015.

[52] Tom White. Sampling generative networks. *arXiv preprint arXiv:1609.04468*, 2016.

[53] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[54] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[55] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122, 2015.

[56] Andreas Zierer, Spencer J Melby, Rochus K Voeller, Tracey J Guthrie, Gregory A Ewald, Kim Shelton, Michael K Pasque, Marc R Moon, Ralph J Damiano Jr, and Nader Moazami. Late-onset driveline infections: the achilles' heel of prolonged left ventricular assist device support. *The Annals of thoracic surgery*, 84(2):515–520, 2007.