Computational analysis of mitochondria in *Caenorhabditis elegans* using a Fully Convolutional Neural Network and common feature detectors

Dissertation zur Erlangung des Doktorgrades der Naturwissenschaften
Doctor rerum naturalium (Dr. rer. nat) an der Fakultät für Biologie der
Ludwig-Maximilians-Universität München

Vorgelegt von Christian Fischer

München, 18ter März 2020

1.Gutachter:	Prof. Dr. Barbara Conradt
2.Gutachter:	Prof. Dr. Christian Leibold
Tag der Abgabe:	18.03.2020
Tag der mündlichen Prüfung:	25.11.2020

Eidesstattliche Erklärung

Ich versichere hiermit an Eides statt, dass die vorgelegte Dissertation von mir selbstständig und ohne unerlaubte Hilfe angefertigt ist.

Erklärung

Diese Dissertation im Hauptfach Biologie wurde von Prof. Dr. Barbara Conradt betreut. Hiermit erkläre ich, dass die Dissertation nicht ganz oder in wesentlichen Teilen einer anderen Prüfungskommission vorgelegt worden ist und dass ich mich nicht anderweitig einer Doktorprüfung ohne Erfolg unterzogen habe.

München, 18ter März 2020

_____

Christian Fischer

Table of Contents

“All you really need to know for the moment is that the universe is a lot more complicated than you might think, even if you start from a position of thinking it's pretty damn complicated in the first place.”

Douglas Adams

Abstract

It is known that the morphology of mitochondria has a profound impact on the organelles function and while the analysis of mitochondrial morphology has emerged as an important tool in the study of mitochondrial biology, quantification of microscopy images depicting mitochondria presents a challenging task. Here I present the development of three computational workflows to study the membrane potential and dynamics of mitochondria in *C. elegans*. Specifically, the main project involved the generation of a pretrained deep learning segmentation model called MitoSegNet, which enables researchers to perform a high-accuracy segmentation of mitochondria. The segmentation performance of MitoSegNet was tested against three feature detectors and the machine learning segmentation tool ilastik. The MitoSegNet outperformed all other methods in both pixelwise and morphological segmentation accuracy. The MitoSegNet was successfully applied to unseen fluorescence microscopy images depicting mitoGFP labelled mitochondria of wild type and *catp-6* mutant *C. elegans* adults. Additionally, MitoSegNet was capable of accurately segmenting mitochondria in HeLa cells treated with fragmentation inducing reagents. The MitoSegNet is provided in combination with the easy-to-use segmentation tool MitoS and the statistical analysis and visualisation tool MitoA. Both tools have been tested extensively, can be run without any prerequisite installations and are distributed as standalone executable files for Windows and Linux operating systems. The MitoSegNet was successfully applied to examine oxidative stress and membrane potential in *C. elegans* mitochondria in an effort to study the ATP13A2-mediated endo-lysosomal polyamine export (Vrijsen, Besora-Casals et al., unpublished). Furthermore, I developed two automated ImageJ-based segmentation workflows prior to the completion of MitoSegNet and used to quantify the mitochondrial membrane potential in *C. elegans*. The data generated with these workflows enabled significant discoveries that are included in two published manuscripts. Using the automated ImageJ

workflows, Rolland et al. (2019) found the membrane potential to be interdependent with the mitochondrial unfolded protein response (UPR^{mt}) and Haeussler et al. (2020) determined how an increase in autophagic flux increases mitochondrial membrane potential.

Abbreviations

°C	degrees Celsius
µm	micrometres
µM	micromolar
2D	2-Dimensional
3D	3-Dimensional
ADP	Adenosine Diphosphate
AGI	Artificial General Intelligence
AI	Artificial Intelligence
ANC-1	Abnormal Nuclear Anchorage
ANI	Artificial Narrow Intelligence
ANN	Artificial Neural Network
ANOVA	Analysis Of Variance
API	Application Programming Interface
ATP	Adenosine Triphosphate
bit	binary digit
catp-6	cation transporting ATPase
CCD	Charged Coupled Device
CED-1	Cell Death Abnormality
CMOS	Complementary Metal Oxide Semiconductor
CNN	Convolutional Neural Network
CPS1	Carbamoyl Phosphate Synthetase I
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
cuDNN	CUDA Deep Neural Network
dc	dice coefficient
DIC	Differential Interference Contrast
DNA	Deoxyribonucleic Acid
DoG	Difference of Gaussians

EGL-1	Egg Laying Defective
FCN	Fully Convolutional Network
GFP	Green Fluorescent Protein
GMM	Gaussian Mixture Models
GPU	Graphics Processing Unit
kDa	kilodalton
LDA	Latent Dirichlet Allocation
LoG	Laplacian of Gaussians
mm	millimetre
MRC	Mitochondrial Respiratory Chain
ms	milliseconds
mtDNA	mitochondrial DNA
NA	Numerical Aperture
NADH	Nicotinamide Adenine Dinucleotide
OS	Operating System
OTC	Ornithine Transcarbamylase
PCA	Principal Component Analysis
RAM	Random Access Memory
ReLU	Rectified Linear Unit
RFP	Red Fluorescent Protein
RGB	Red-Green-Blue
RNA	Ribonucleic Acid
SARSA	State-Action-Reward-Station-Action
SGD	Stochastic Gradient Descent
SVD	Singular Value Decomposition
SVM	Support Vector Machines
TMRE	Tetramethylrhodamine Ethyl Ester
UPR^{mt}	mitochondrial Unfolded Protein Response
WAH-1	Worm AIF Homologue
wt	wild type
NGM	Nematode Growth Medium

Introduction

The model organism *C. elegans*

Caenorhabditis elegans or more commonly referred to as *C. elegans* is a transparent nematode of about 1 mm length that lives on top of temperate soil. The name derives from the Greek words caeno (recent), rhabditis (rod-like) and the Latin elegans, which means elegant. The worm *C. elegans* was made famous by Sydney Brenner, who in 1963 first proposed research on neuronal development in the worm. In 1974, Brenner began to study the developmental biology of *C. elegans*, which later made the worm a widely used model organism (Brenner 1974). There are many factors that make this worm a favourable model organism. *C. elegans* development from egg to egg-laying adult takes only 3 days at 25°C (Fig. 1) and most worms exist as self-fertilizing hermaphrodites, while males only comprise around 0.2% or less of a typical *C. elegans* culture. This means that they can be grown inexpensively in large numbers and are ideally suited for the study of developmental processes due to their short life cycle. The number of somatic cells in adult *C. elegans* worms is invariant, which allowed to track the fate of every cell from fertilization to adulthood and thus create a complete cell lineage (Sulston and Horvitz 1977, Kimble and Hirsh 1979, Sulston, Schierenberg et al. 1983). Mutations that cause behavioural or developmental defects are easily detectable in genetic screens of *C. elegans* due to the invariant wild type cell lineage. Furthermore, roughly 38% or more *C. elegans* protein-coding genes have predicted orthologues in the human genome, 60-80% of human genes have an orthologue in the *C. elegans* genome and 40% of genes that are associated with human diseases have orthologues in the genome of *C. elegans* (Culetto and Sattelle 2000, Kaletta and Hengartner 2006, Shaye and Greenwald 2011). In 1998, *C. elegans* was the first multicellular organism to have its whole genome sequenced and reverse and forward genetics

have since led to the discovery of key genes in many cellular and developmental processes (*C. elegans* Sequencing Consortium (1998)). Perhaps one of the most important reasons why *C. elegans* was chosen by Sydney Brenner and was later established as model organism is the easy genetic manipulation. Because the hermaphrodites can self-fertilize, after being mutagenized, the mutant alleles can be preserved in the F1 and F2 generation by self-propagation without mating. This makes obtaining homozygous progeny easier than with other more complex model organisms. Due to their short life cycle, one can detect mutant homozygotes just one week after mutagenesis. Because *C. elegans* worms have transparent bodies, single cells can be studied within the multicellular organism, making it a highly favourable model organism among cell biologists. For example, the mitochondrial respiratory chain (MRC) of the nematode is very similar to the mammalian MRC and sequencing of the *C. elegans* mtDNA revealed it to be only slightly smaller than the human counterpart (Murfitt, Vogel et al. 1976, Okimoto, Macfarlane et al. 1992). Among eukaryotes the 12 mtDNA-encoded MRC polypeptides were found to be highly conserved and studies of the *C. elegans* nuclear genome also showed that most of the mammalian nuclear-encoded MRC genes are highly conserved (Tsang and Lemire 2003). For that reason, studying mitochondrial function in *C. elegans* has greatly contributed to our understanding of mitochondria in multicellular organisms.

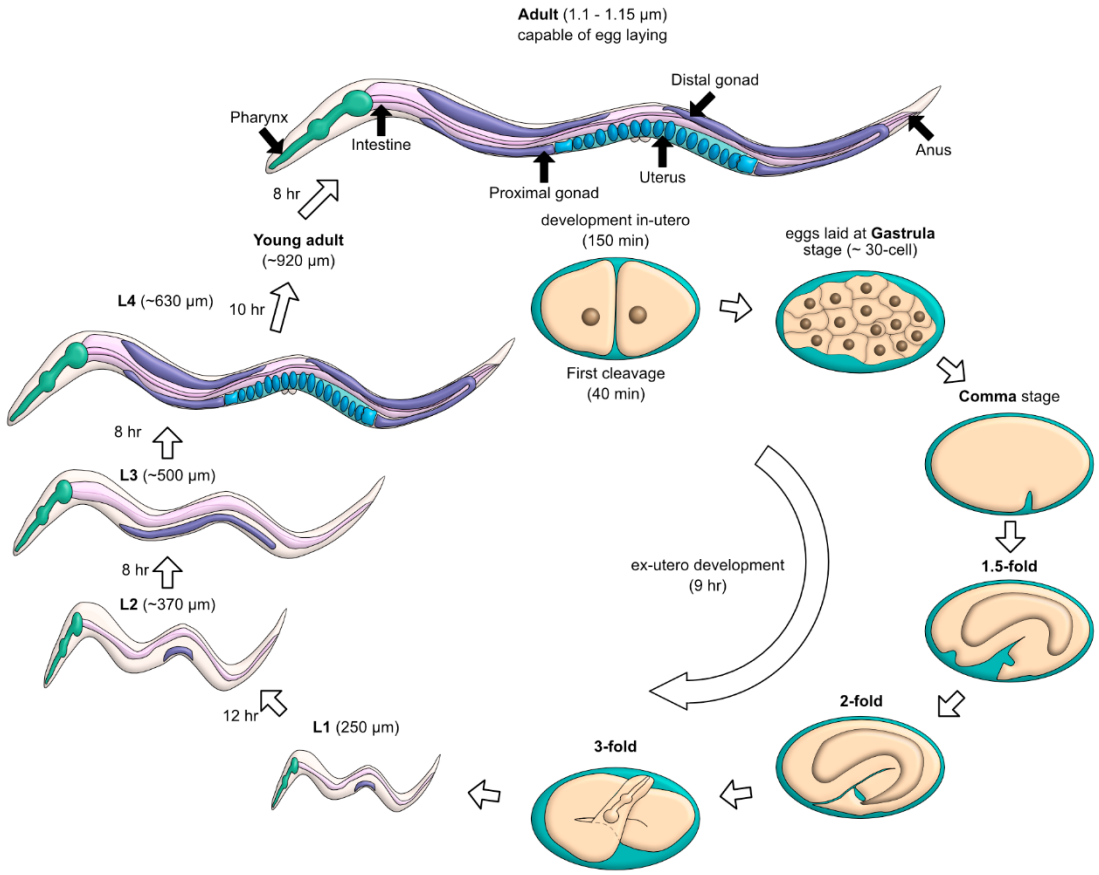


Figure 1. *C. elegans* life cycle at 22°C. *C. elegans* embryogenesis is divided into two stages, proliferation and organogenesis. Proliferation spans up to 350 minutes, of which the first 150 minutes take place within the uterus. The embryo is laid outside at gastrulation and continues proliferation ex-utero until 5.5 to 6 hours post-fertilisation, after which the organogenesis begins for 6 to 8 hours. Prior to hatching, pharyngeal pumping begins. The embryo hatches at around 14 hours post-fertilisation and the post-embryonic development begins (Schierenberg, Miwa et al. 1980, Sulston, Schierenberg et al. 1983). If food is present, the L1 larvae continue development, or else enter the L1 arrest stage and can survive for 6 to 10 days without food (Johnson, Mitchell et al. 1984). During the L2 stage germ cell divisions continue and the gonad begins to elongate, followed by an extension of the gonad arms in opposite directions from L3 until the mid-L3 stage. The gonadogenesis is completed in the L4 stage and between 45 to 50 hours after hatching, recently matured hermaphrodites lay their first eggs, completing the life cycle. Based on (Altun, Herndon et al.).

Mitochondria

Introduction

Mitochondria were most likely first observed in the mid-19th century and in 1890 they were described as cytoplasmic structures that are similar to bacteria (Altman 1890, Ernster and Schatz 1981). Today, most scientists believe that mitochondria originate from eubacteria that were engaged in an endosymbiotic relationship with a host cell (Gray, Burger et al. 1999). The word mitochondrion is derived from the Greek words mitos (thread) and chondrion (granule). It is a double-membrane organelle, consisting of an inner membrane that separates the mitochondrial matrix from the intermembrane space and the outer membrane forms the boundary between the organelle and the cellular cytoplasm. The inner membrane is a highly convoluted and folded structure (cristae), where the ATP synthase is located (Fig. 2A). The outer membrane gives structure to the mitochondria, is responsible for organelle biogenesis and is permeable to most molecules with a weight of less than 10 kDa.

Mitochondrial respiratory chain

The main function of mitochondria is to produce ATP through oxidative phosphorylation, which in turn is driven by the mitochondrial membrane potential ($\Delta\Psi_m$). Four electron transporting protein complexes, which all are located in the inner membrane, are responsible for the transfer of electrons from electron donors (NADH or succinate) to electron acceptors (O_2). This process generates an electrochemical proton gradient, which the ATP synthase uses to phosphorylate ADP. The transport of electrons between complexes I or II to complex III is facilitated by the lipophilic carrier coenzyme Q (ubiquinone). Electron transfer between complexes III and IV is regulated by the hydrophilic heme-containing protein cytochrome c

(Fig. 2B). A recent discovery questions the idea of a homogenous $\Delta\Psi_m$ along the inner mitochondrial membrane. This study found disparate membrane potentials between different cristae of the same mitochondrion, stating that cristae within the same mitochondrion behave as independent bioenergetic units (Wolf, Segawa et al. 2019).

Other roles of mitochondria

Besides oxidative phosphorylation, mitochondria carry out many other important cellular functions. Within mitochondria the coordination complex heme is synthesized, which is needed to give hemo- and myoglobin the ability to bind oxygen (Ajioka, Phillips et al. 2006). Furthermore, amino acids and lipids are synthesized in mitochondria. The important biochemical precursor to oxidative phosphorylation, the citric acid cycle, also takes place within the matrix of the mitochondrion. The cycle uses acetyl-CoA, the end product of pyruvate after dehydrogenation, thereby generating NADH, which is fed into the electron transport pathway (Mailloux, Bériault et al. 2007). Besides production of acetyl-CoA through glycolysis that takes place in the cytoplasm, mitochondria can also generate acetyl-CoA with beta-oxidation (Houten and Wanders 2010). The urea cycle converts toxic ammonia to urea for excretion and two enzymes, namely carbamoyl phosphate synthetase I (CPS1) and ornithine transcarbamylase (OTC) are involved in the urea cycle and are located within mitochondria (Diez-Fernandez and Häberle 2017). The intrinsic apoptosis pathway is also referred to as the mitochondrial apoptosis pathway due to the essential involvement of mitochondria. In *C. elegans*, CED-4 (cell death abnormality) is bound by CED-9 on the outer mitochondrial membrane, which blocks apoptosis. The BH3-only protein EGL-1 is regulated during development or in response to apoptotic stimuli. Once EGL-1 binds to CED-9, CED-9 undergoes a conformational change and CED-4 is released. CED-4 proteins translocate to the

perinuclear membrane and undergo oligomerization, which assists in bringing two CED-3 proenzymes to close proximity, thereby initiating autoproteolytic activation of CED-3 (Fig. 2C) (Conradt and Horvitz 1998, del Peso, Gonzalez et al. 1998, Parrish, Metters et al. 2000, Yan, Gu et al. 2004). Furthermore, WAH-1 (worm AIF homolog) is released from EGL-1 bound mitochondria, which can cooperate with CSP-6 (Ced-3 protease suppressor) to degrade DNA and induce cell killing when expressed together (Parrish, Li et al. 2001, Wang, Yang et al. 2002).

Mitochondrial dynamics

Unlike the often propagated view of mitochondria being static, bean-shaped organelles, they are in fact dynamic structures that most often form tubular networks (Bereiter-Hahn 1990). Mitochondria are known to interact with certain components of the cytoskeleton, and it is believed that such interactions may enable the localization of mitochondria to regions where a high energy demand is needed (Heggeness, Simon et al. 1978, Morris and Hollenbeck 1993). Positioning and distribution of mitochondria in *C. elegans* is regulated by multiple proteins. Mutations in the *anc-1* gene of *C. elegans* cause a defect in the anchoring of mitochondria in the hypodermis, causing the organelles to float unattached in the cytoplasm of syncytial cells (Hedgecock and Nichol Thomson 1982). The mitochondria of cells carrying a mutated *anc-1* gene were also found to be mostly spherical instead of tubular. As the *anc-1* gene encodes a protein consisting of an actin-binding domain and several large coiled regions, it is believed that ANC-1 may connect mitochondria with the actin cytoskeleton (Starr and Han 2002). Fusion and fission of mitochondria allows the organelles to respond to environmental or metabolic stresses, while maintaining homeostasis. Through fusion, contents of partially damaged mitochondria are mixed, which acts as a form of complementation (Yang, Long et al.

2015). The membrane anchored protein FZO-1 carries out the fusion of the outer mitochondrial membrane, while EAT-3 fuses the inner membrane (Rolland, Lu et al. 2009). Fission creates new mitochondria, enables the removal of damaged mitochondria and can also facilitate apoptosis during increased levels of cellular stress. This process is mediated by DRP-1, a cytosolic dynamin family member in worms and mammals. It is transported from the cytosol to form constricting coils around mitochondria that sever both inner and outer membranes (Fig. 2D) (Legesse-Miller, Massol et al. 2003, Jagasia, Grote et al. 2005). These different morphological states are associated with numerous pathophysiological conditions (Nunnari and Suomalainen 2012). Fragmentation of mitochondria is often linked to mitochondrial dysfunction as this morphological state is most commonly observed during high stress levels and cell death (Zemirli, Morel et al. 2018). It is believed that fused mitochondria allow the distribution of matrix components and thereby increase oxidative phosphorylation (Mishra and Chan 2016). Elongation of mitochondria enables protection against autophagy caused by nutrient starvation and is associated with cell survival (Rambold, Kostelecky et al. 2011).

Microscopy-based examination of fluorescently labelled mitochondria in different *C. elegans* loss-of-function mutants can greatly contribute to the understanding of mitochondrial dynamics. To make reliable conclusions, a large number of images have to be processed and subsequently analysed, which is best done in an automated fashion. The crucial step within an automated image analysis workflow prior to generating numerical data is image segmentation.

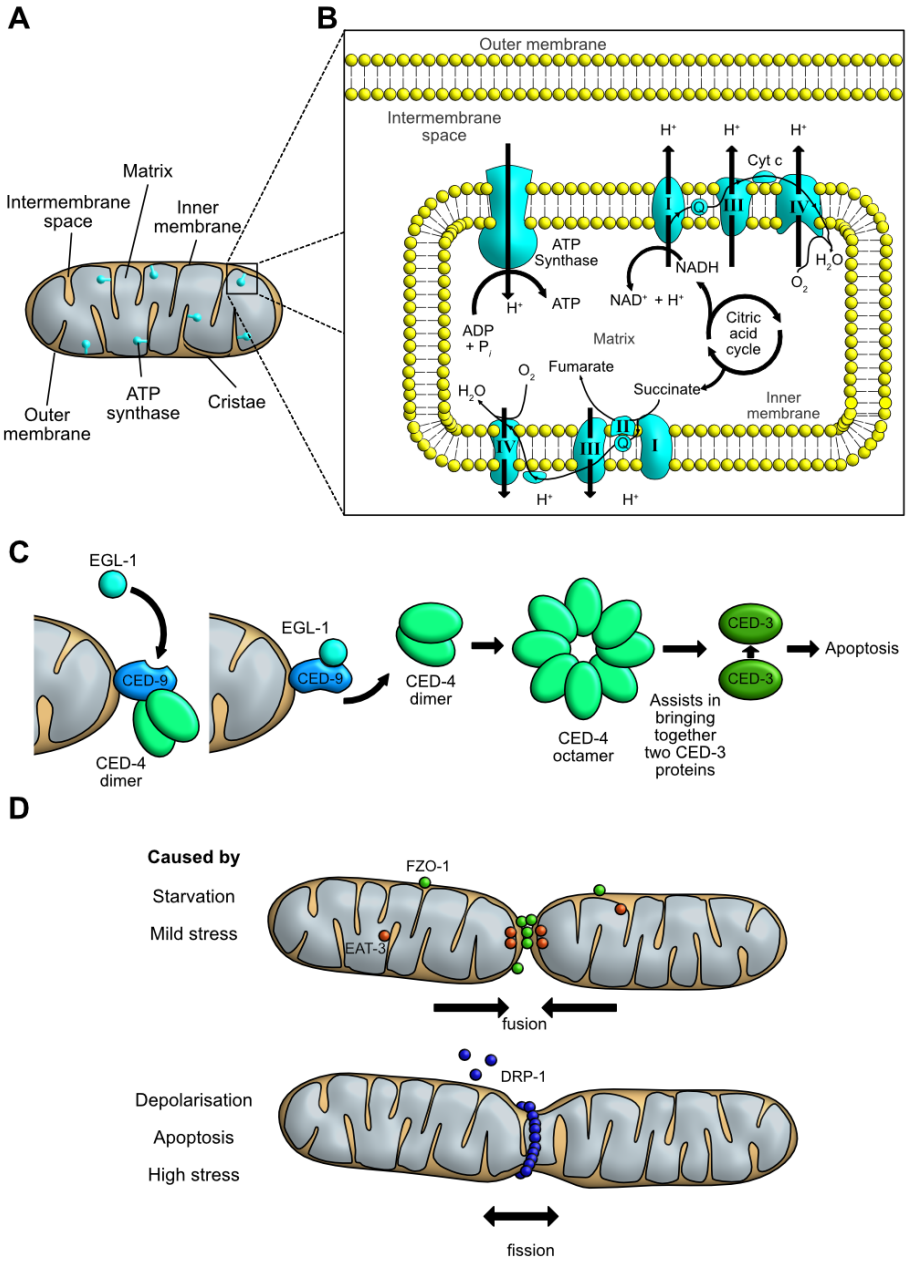


Figure 2. An overview of mitochondrial anatomy and function. (A) Mitochondria are double-membrane organelles that produce most of the cell's ATP. The outer membrane separates the organelle from the cytoplasm and the inner membrane separates the intermembrane space from the matrix. The inner mitochondrial membrane is compartmentalized into multiple cristae, which expands the surface area of the inner membrane, thereby enhancing ATP production via the ATP synthase proteins. **(B)** Mitochondrial respiratory / Electron transport chain. ATP synthesis is driven by a series of protein complexes (I-IV) that transfer electrons from electron donors (such as NADH or Succinate generated during the Citric acid cycle) to the electron acceptor oxygen which is reduced to water. The energy obtained through the transfer of electrons along the complexes is used to pump protons from the mitochondrial matrix to the intermembrane space, thus creating a proton gradient that drives the ATP synthase to phosphorylate ADP to ATP. Ubiquinone (Coenzyme Q) is a coenzyme that acts as an

electron carrier which receives electrons from complex I and complex II and passes them on to complex III. Cytochrome c is another electron carrier, which is located in the intermembrane space and transfers electrons from complex III to complex IV. Based on (Reece, Urry et al. 2014). **(C)** Intrinsic apoptotic pathway in *C. elegans*. EGL-1 binds to CED-9, which together with CED-4 is bound to the outer mitochondrial membrane. Upon EGL-1 binding to CED-9, a conformational change is induced which releases CED-4 from CED-9. CED-4 proteins undergo oligomerization and the resulting CED-4 octamer assists in bring two CED-3 proteins in close proximity to each other to induce autoproteolytic activation of CED-3, which then leads to apoptosis. Based on (Conradt and Xue 2005). **(D)** Mitochondrial dynamics. Three key proteins involved in mitochondrial shape change in *C. elegans* are FZO-1, EAT3 and DRP-1. FZO-1, located on the outer mitochondrial membrane and EAT-3, located at the inner membrane are necessary for mitochondrial fusion, which is induced by mild stress or starvation. DRP-1 is located in the cytosol and activated by high stress, apoptosis or depolarisation, after which it forms constricting coils around mitochondria and severs both inner and outer membranes. Based on (van der Bliek, Sedensky et al. 2017).

Computer vision

Introduction

Computer vision originated in the 1960s as a by-product of pioneering research on artificial intelligence. The main goal of computer vision is to enable computers to gain human-level insights from digital images or videos (Huang 1996). Originally, artificial intelligence researchers believed this goal to be achievable over the course of a summer project (Papert 1966) and it was only later realised that creating an artificial visual system was a far more formidable challenge. Only recently, with the rise of computing power and improved deep learning algorithms is the field of computer vision moving closer to its original goal (LeCun, Bengio et al. 2015). All computer vision related methods begin with image acquisition, which describes the generation of digital images through detection of photons (or other type of particles) by one or multiple image sensors. The two most widely used image sensors are the charged-coupled-device (CCD) and the active-pixel sensor (CMOS), which are both based on metal-oxide-semiconductor technology. Image sensors are made up of photodiodes that

convert photons into electron charges. The smallest area on an image sensor capable of detecting photons corresponds to a single pixel (picture element) and the pixel's intensity value directly correlates with the photon count of the photodiode. The image bit-depth determines the value range of each pixel and based on the fundamental computational storage unit byte (8 bit) the range is given as 2 to the power of the bit-depth. For an 8-bit image there are $2^8 = 256$ possible intensity values per pixel. The pixel intensity information can then be used for pre-processing, feature extraction, object detection and high-level processing, such as registration (Meijering 2012). One of the most prominent fields of computer vision is image segmentation.

Image segmentation

Image segmentation is the process of dividing an image into multiple segments or biologically meaningful parts, thus simplifying the representation of an image and making it easier to analyse (Shapiro and Stockmann 2001). The simplest form of image segmentation is intensity thresholding, in which each pixel value is compared to a fixed constant and set to 0 if the value is lower than the fixed constant or set to 255 (for 8-bit images) if the value is greater (Fig. 3). Automatic thresholding is based on statistical analysis of global or local image intensities and can be further categorised into six groups, which are histogram shape-, clustering-, entropy, object attribute-, spatial- and local-based methods (Sezgin and Sankur 2004). Thresholding is widely used for identification of biological features prior to analysis but is only successful if such features are well separated and their intensities vary considerably from the background (Torborg and Feller 2004). These requirements are seldom met in live cell imaging due to autofluorescence, noise or fluctuating intensities (Fig. 3).

Instead of relying on absolute intensities for segmentation, it is also possible to instead use intensity derived features that can be segmented with the aid of feature detectors. Three commonly used feature detectors are the Difference of Gaussians (DoG), the Laplacian of Gaussian (LoG) and the Determinant of Hessian (DoH). The DoG feature enhancement can for example be used to improve intensity separation between cells and background. Two copies of the original image *f(x,y)* are blurred with Gaussian kernels of different variances ($G_{\sigma 1}$ and $G_{\sigma 2}$) and the DoG enhancement is achieved by subtracting the blurred image g_2 from the less blurred version of the same image g_1.

$$G_{\sigma 1}(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$g_1(x,y) = G_{\sigma 1}(x,y) * f(x,y)$$

$$g_2(x,y) = G_{\sigma 2}(x,y) * f(x,y)$$

$$DoG * f(x,y) = g_1(x,y) - g_2(x,y)$$

The DoG operator is defined as

$$DoG \triangleq G_{\sigma 1}(x,y) - G_{\sigma 2}(x,y)$$

DoG is used to enhance the visibility of edges by removing high frequency information but at the cost of reducing the overall image contrast (Fig. 3). For the LoG feature enhancement, the input image is also convolved with a Gaussian kernel G_σ and the additional application of the Laplace operator Δ.

$$LoG * f(x,y) = \Delta[G_\sigma(x,y) * f(x,y)]$$

The LoG operator is defined as

$$LoG \triangleq \Delta G_\sigma(x,y) = \frac{\partial^2 G_\sigma(x,y)}{\partial x^2} + \frac{\partial^2 G_\sigma(x,y)}{\partial y^2}$$

LoG is useful for detecting edges that appear at different image scales or degrees of focus (Fig. 3) (D. Marr 1980, Lindeberg 1994). The DoH method is based on the Hessian matrix *H*, which is a square matrix of second-order partial derivatives of a scalar-values function or field and describes the local curvature of a multivariate function.

$$H = \begin{bmatrix} \frac{\partial^2 f(x,y)}{\partial x_1^2} & \frac{\partial^2 f(x,y)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x,y)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x,y)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x,y)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x,y)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x,y)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x,y)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x,y)}{\partial x_n^2} \end{bmatrix}$$

By calculating the eigenvalues λ of the Hessian matrix one can calculate the object curvature, which enables the detection of curvilinear structures such as nerve fibres or blood vessel (Lindeberg 1998). Eigenvalues are a special set of scalars associated with a linear system of equations. An eigenvector does not change direction, only its magnitude when multiplied with a matrix. So eigenvectors are only scaled and its scaling factor is the eigenvalue. The eigenvalues can be calculated by solving the following equation

$$|H - \lambda I| = 0$$

where *H* is the Hessian matrix, λ represents the eigenvalues and *I* is the identity matrix of *H*. This method has been used to segment retinal blood vessels and was implemented in the ImageJ Tubeness plugin (Fig. 3) (Sato, Nakajima et al. 1998, Li, Gong et al. 2015).

If linear image filtering does not achieve the desired segmentation accuracy, morphological filtering and its use of non-linear operations can offer a valid alternative. Morphological filtering enables segmentation based on geometrical and topological properties of objects in images. The most prominent morphological operations are closing, opening, dilation and

erosion. The erosion applied to a binary image *f* by a structuring element *s* generates a new image *g* defined as

$$g = f \ominus s$$

Erosion removes a layer of pixels from the outer and inner boundaries of a region, widening holes and gaps between objects and filtering out small details. Image dilation is defined as

$$g = f \oplus s$$

and has the opposite effect of erosion as it adds pixel layers to the inner and outer boundaries of objects. Closing and opening are compound operations which represent a combination of erosion and dilation. Opening is erosion followed by dilation and defined as

$$f \circ s = (f \ominus s) \oplus s$$

As the name suggests, opening opens up gaps between objects that are connected by a narrow segment of pixels. Closing describes a dilation followed by an erosion and is defined as

$$f \bullet s = (f \oplus s) \ominus s$$

and is often used to fill holes in regions without changing the object size. While the aforementioned binary morphological operations are used as a postprocessing step, grayscale morphological operations are a pre-processing step to enhance certain image structures. Important grayscale morphology operations include top-hat transform, morphological gradients and the watershed algorithm (Vincent and Soille 1991, Meijering 2012).

With the rapid developments in the field of artificial intelligence in the last few years, new image segmentation approaches have started to incorporate methods from machine and deep learning, which have largely outperformed previous segmentation methods (O. Ronneberger 2015, Chen, Dai et al. 2017, Sadanandan, Ranefall et al. 2017, Kreshuk and Zhang 2019).

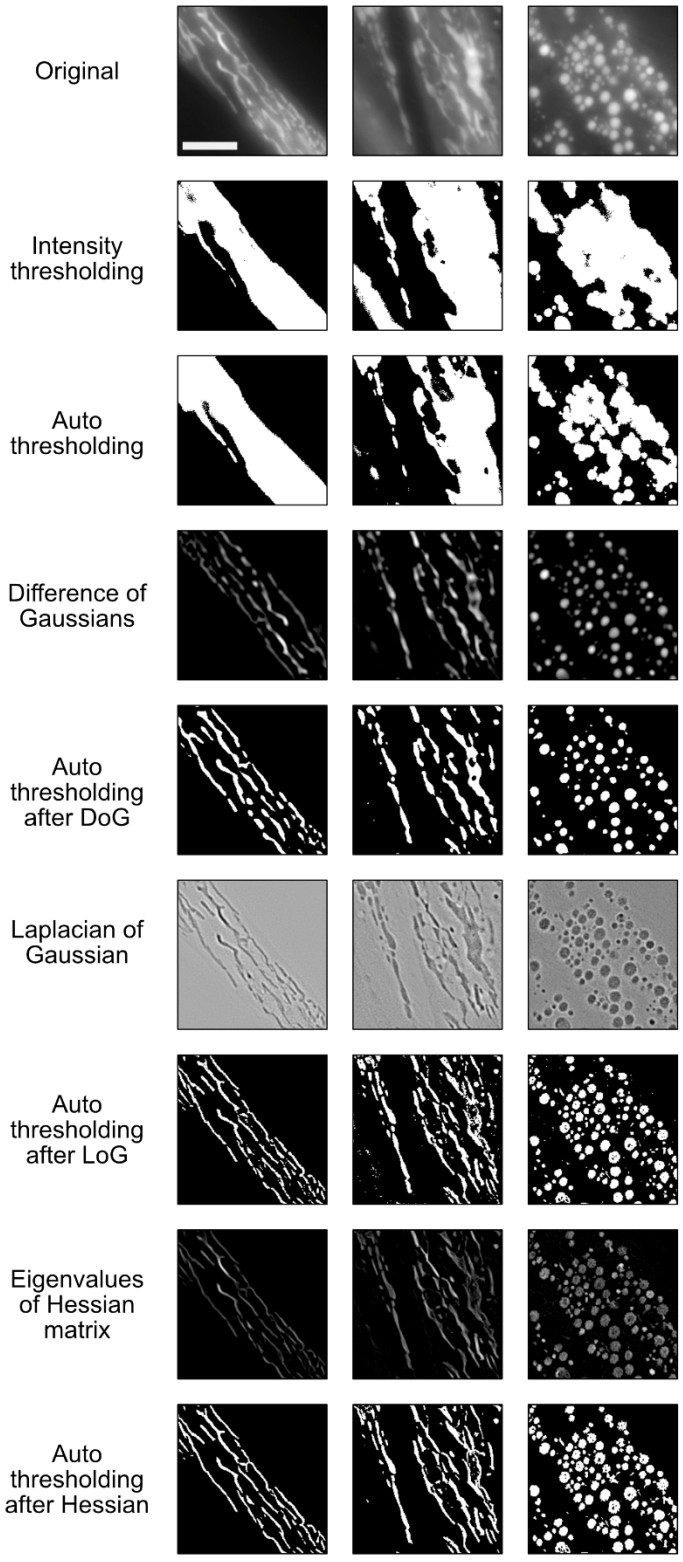


Figure 3. Thresholding without and with prior linear image filtering on images depicting mitochondria in adult *C. elegans*. Intensity thresholding yields poor segmentation results as the intensity difference between background and mitochondria is too low. The Auto-thresholding method (which is based on the IsoData algorithm) performs only marginally better than the prior intensity threshold. Difference of Gaussians (DoG) filtering outputs a non-binary image in which due to the subtraction of two Gaussian-blurred images the distinction between mitochondria and background appears much sharper. Auto-thresholding after DoG performs much better than auto-thresholding without DoG, due to the edge contrast enhancement. Similar results are obtained from application of the Laplacian of Gaussian method, which is better at detecting smaller objects than the DoG algorithm. While this might be useful to detect smaller objects this also increases the possibility of noise. Extracting the eigenvalues of a Hessian matrix using the ImageJ Tubeness plugin results in an image that contains the score for how "tube-like" each point in the image is. This works well for curvilinear structures such as tubular mitochondria and while this method is also capable of detecting elongated and fragmented mitochondria, more segmentation errors are introduced when mitochondria are broader or more circular shaped. The scale bar represents 5 µm.

Artificial Intelligence

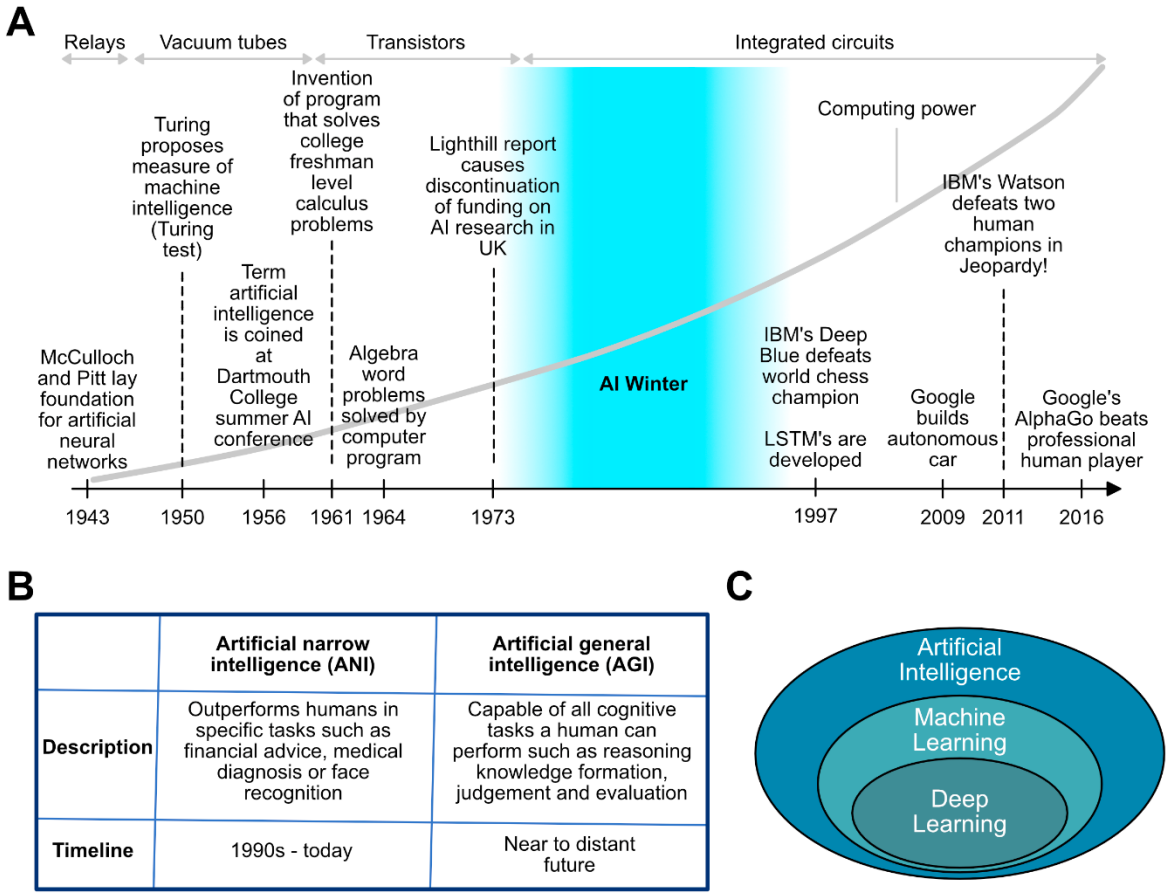
Introduction

Artificial intelligence (AI) as a field can be understood as the science of developing intelligent machines and AI as a concept is defined as the ability of machines to demonstrate intelligence. The exact definition of AI is still a topic of discussion as there is still no clear definition of the concept of intelligence. Some of the earliest ideas of AI can be found in old Greek myths such as of the god Hephaestus, who created the first human woman, or the belief that sacred mechanical statues built in ancient Egypt and Greece were believed to be capable of wisdom and emotion (McCorduck 2004). A more computational concept of AI arose in the 20th century with Alan Turing's development of computation theory and the idea that a machine capable of shuffling zeros and ones, could simulate any process of reasoning (Church-Turing thesis) (Copeland and Shagrir 2018). The earliest work that today is considered as AI was the design for Turing-complete artificial neurons in 1943 by McCulloch and Pitts (Fig. 4A) (Russell and Norvig 2009). The term Turing-complete refers to a system of data manipulation rules that can be used to simulate any Turing machine (hypothetical machine that can simulate any computer algorithm, regardless of its complexity). The field of AI research was initiated by Allen Newell, Herbert Simon, John McCarthy, Marvin Minsky and Arthur Samuel at a workshop at Dartmouth College in 1956 (Fig. 4A) (Russell and Norvig 2009). During what later was described as the golden years of AI, computers capable of learning checkers strategies, speaking English or proving logical theorems were developed. The optimism of this time led leading scientists in the field of AI to predict that within a few decades, machines would be able to do any work humans can do. As progress began to slow down in the 1960s and 70s, government funding for exploratory research was cut off and a period referred to as AI winter began, during which it was difficult to obtain funding for AI research (Fig. 4A) (Lighthill

1973). AI research was revived in the 1980s after the commercial success of expert systems, which simulate knowledge and analytical skills of human experts. However, this revival was not permanent and after the collapse of the Lisp machine market, AI research again came to a standstill. The latest revival of AI research began in the late 1990s and was benefitted by the increasing computational power available, development of new or improved algorithms and the growing collaboration with other scientific fields (Kurzweil 2006). Since IBM's Deep Blue computer beat world chess champion Garry Kasparov in 1997, the field of AI research has steadily progressed until the present (Fig. 4A). Recent events of great significance for the development of AI include the development of the long short-term memory recurrent neural network (used for speech recognition and robot control) (Hochreiter and Schmidhuber 1997), generation of large annotated databases such as MNIST or ImageNet, IMB's Watson winning in the "Jeopardy!" quiz show and the more recent success of AlphaGo becoming the first Go-playing computer to beat a professional Go player (Silver, Schrittwieser et al. 2017) (Fig.4 A).

Despite the significant developments in the last two decades, the AI field continues to face complex challenges such as computationally replicating reasoning, social or general intelligence. Reasoning is seen as a characteristically human practice and thus far no efficient systems for artificial reasoning have been created. Previous algorithms proved insufficient to solve larger reasoning problems due to a combinatorial explosion, causing the systems to exponentially slow down as problems grow larger (Russell and Norvig 2009). The incorporation of social intelligence into machines and computer programs is called affective computing and aims to detect, interpret and simulate human emotion (Tao and Tan 2005). There have been moderate successes in textual sentiment analysis or multimodal affect analysis (AI classification of affect displayed by a videotaped subject) but due to the complexity of human emotion the progress of developing socially intelligent AI moves at slower pace as other fields such as natural language processing or AI learning (Poria, Cambria et al. 2017). The final

goal of AI research is the development of artificial general intelligence (AGI) or full AI, which describes a machine that can learn any task a human can do (Fig. 4B). Thus far, most AI applications are considered to be narrow AI applications such as skin cancer detection, autonomous driving or recommender systems (Fig. 4B). There are currently only few research projects or companies explicitly aiming to develop AGI, the three largest being DeepMind, the Human Brain Project and OpenAI (Baum 2017). One of the largest AI fields in both research and industry is machine learning, which is the study of computer algorithms that improve through experience (Fig. 4C).



	Artificial narrow intelligence (ANI)	Artificial general intelligence (AGI)
Description	Outperforms humans in specific tasks such as financial advice, medical diagnosis or face recognition	Capable of all cognitive tasks a human can perform such as reasoning knowledge formation, judgement and evaluation
Timeline	1990s - today	Near to distant future

Figure 4. History of AI, distinction between ANI and AGI and the relationship of AI with machine and deep learning. (A) An overview over some of the milestones in AI research history. The foundations for artificial neural networks were laid by McCulloch and Pitts in 1943. Alan Turing proposed the measure of machine intelligence (known as Turing test). The Turing test states that a machine can be considered intelligent if a human interacting with two terminals, one operated by a human, the other by a computer (and all are physically separated) cannot correctly distinguish which respondent was human. In 1956 the term AI is officially coined by Allen Newell, Herbert Simon, John McCarthy, Marvin Minsky and Arthur Samuel at the Dartmouth College summer AI conference. In the following years, new developments in AI research led to computer programs able to solve complex calculus or algebra word problems. After Minsky and Papert demonstrated previously unrecognized limits of the current AI systems in 1969 (Minsky and Papert 1969) and with the release of the Lighthill report in 1973, global funding for AI research decreased, marking the beginning of the so called AI winter. During this period which lasted until the early 1990s AI research continued to make progress but at a slower pace and generally received little public attention. In the 1990s AI research did not only profit from the continued gain of computational power but also from the increasing usage of the still emerging internet, which offered AI algorithms the much needed data to learn new tasks. In 1997 IBM's Deep Blue computer defeated the world chess champion Gary Kasparov and long short-term memory recurrent neural networks were developed, which greatly improved both efficiency and practicality of recurrent neural networks. Many

databases were created in the late 90s and early 2000s, such as MNIST or ImageNet, which are considered to have contributed to the still ongoing AI boom. In 2009 Google began developing self-driving technology and two years later IBM's Watson defeated two human champions in a Jeopardy! competition by combining machine learning, natural language processing and information retrieval techniques. Another milestone was reached in 2016, when Google's AlphaGo became the first computer program to defeat an unhandicapped human player. Content partially based on (Press 2016). **(B)** There are two types AI, artificial narrow intelligence (ANI) and artificial general intelligence. ANI is now commonly applied in many different sectors such as medical diagnosis or face recognition, while AGI still remains a future goal of AI research. **(C)** Artificial intelligence encompasses machine learning, which encompasses deep learning.

Machine learning

Machine learning algorithms train mathematical models with sample data (known as training data) to make predictions or decisions without being specifically programmed to perform such a task. The approaches of machine learning are classified in three major types of algorithms, which are supervised, unsupervised and reinforcement learning (Fig. 5). The process of learning is largely based on optimisation, which is the minimisation of a specified loss function for a machine learning model. The loss function describes the discrepancy between the predicted and real output and through optimisation algorithms the model parameters are updated to minimise the loss function (Sra, Nowozin et al. 2011).

Supervised learning, learns a function *f* from labelled training *X* data that maps any given input to predict the output *Y*.

$$Y = f(X)$$

There are two main types of supervised machine learning methods, namely classification and regression. Classification involves the prediction of categorical labels, while regression is used for predicting continuous numerical values. Important concepts that need to be taken into consideration before applying supervised machine learning are output noise, function

complexity and training data size, bias-variance trade off and input dimensionality. Noise in the output values can partially result from overfitting, which means the function matches the training data too well, making it incapable of predicting output values based on input not used for training. This can be prevented by removing noisy training examples or to implement regularisation methods such as early stopping (Smith and Martinez 2011). The modelled function complexity often correlates with the amount of training data. If the learned function is simple, then an inflexible learning algorithm with a high bias and low variance (such as linear or logistic regression) can learn from smaller sets of training data. The bias arises from simplifying assumptions made by a specified model and describes the difference between prediction and ground truth. The variance is the variability of model prediction and characterizes the change in prediction based on using different training data. To model a highly complex function that involves multiple different input features, a flexible algorithm with a low bias and high variance (such as decision trees or support vector machines) trained on large sets of training data has to be used. The bias-variance trade off describes the impossibility of predictive models to generalise well beyond the training data while simultaneously minimizing bias and variance (Geman, Bienenstock et al. 1992). Due to the bias-variance trade off, a decrease in bias can lead to overfitting, while a major decrease in variance can cause underfitting. Input space dimensionality is important as high-dimensional input vectors can cause a high variance in the learning algorithm, which is why most often models are tuned to have a high bias and low variance when training data has many features. Additionally, dimensionality reduction methods such as feature selection (removing irrelevant features) or feature projection (e.g. principal component analysis) can be employed (Roweis and Saul 2000).

Unsupervised learning does not require any labelled training data but instead seeks to find previously unknown patterns in unlabelled data (Fig. 5) (Hinton and Sejnowski 1999).

Common algorithms applied in unsupervised learning are clustering (k-means or hierarchical clustering), neural networks (autoencoders or generative adversarial networks) and principal component analysis (Kramer 1991, Hastie, Tibshirani et al. 2009, Goodfellow, Pouget-Abadie et al. 2014).

The third type of machine learning is reinforcement learning, which aims to create programs that learn from experience (Fig. 5). Unlike supervised learning it does not require any labelled input or output and any action taken by reinforcement learning does not need to be explicitly correct (Kaelbling, Littman et al. 1996).

Neural networks are a class of machine learning algorithms that can be configured for supervised, unsupervised or reinforcement learning and have gained considerable popularity due to their superior performance over many other machine learning algorithms.

Figure 5. The three main types of machine learning approaches. Supervised machine learning algorithms are divided in classification and regression. Classification algorithms output a limited set of values (such as categorical or binary values), while regression outputs numerical values within a range. Examples for classification include Decision Trees, Support Vector Machines (SVM) and Convolutional Neural Networks (CNN). The most common regression algorithms include Linear, Logistic and Multivariate Regression (used when there are multiple independent variables). The two main methods of unsupervised machine learning are dimensionality reduction and clustering. The most well-known dimensionality reduction approach is Principal Component Analysis (PCA), which is used to reduce the number of variables. Other methods of dimensionality reduction include Singular Value Decomposition (SVD) and Latent Dirichlet allocation (LDA). Clustering methods such as hierarchical-, k-means-clustering or Gaussian mixture models (GMM) are used to group a set of unlabelled objects that are more similar to each other than to other groups. Reinforcement learning is based on programs that take actions in an environment to maximise their reward. Examples of

reinforcement learning algorithms are the Q-algorithm, Monte Carlo methods and the State-action-reward-station-action (SARSA) algorithm (Wiering and Otterlo 2014).

Artificial Neural Networks

Artificial neural networks (ANNs) are computing systems that are based on biological neural networks and just like other machine learning algorithms perform tasks based on given examples without having been programmed how to complete such tasks (Bishop 1995). ANNs have been successfully applied in many different fields, including machine translation, medical diagnosis, computer vision or speech recognition (LeCun, Bengio et al. 2015). There are multiple different types of ANNs but the most commonly used are feedforward and recurrent neural networks (Bebis and Georgiopoulos 1994, Hochreiter and Schmidhuber 1997). An ANN consist of perceptrons (equivalent to biological neurons), connections (equivalent to axons) between the perceptrons and weights, which describe the connection strength. The ANN takes an input and passes it through multiple layers of perceptrons and outputs a prediction (Fig. 6A). The input layer accepts input features and passes them on to the hidden layer, which describes any layer between the input and output layer that performs the bulk of computational work (the term deep learning refers to neural networks with multiple hidden layers). The predicted output is generated by the output layer. During the training process the data is usually split into training and validation data (James, Witten et al. 2014). Training data is used to update the weights and find the optimal objective function and after the ANN has iterated through all training samples, the objective function is applied to the validation data, which was excluded from training the ANN. The model's performance on the validation data gives insights into whether over- or underfitting is occurring. Each perceptron takes one or more weighted inputs with an additional constant value (bias) and applies an activation function to the weighted sum of inputs (Fig. 6B). The bias is an additional parameter that allows to shift the activation function to better map the

true output value. The ability of neural networks to model complex non-linear relationships is largely based on the usage of non-linear activations functions (Fig. 6C). Activation functions such as the sigmoid function are most often used in the output layer of binary classifications when the result is either 0 or 1. Alternatively, if multiple classes need to be predicted the softmax function can be used as activation function in the output layer (Haykin 1998). The tanh (tangent hyperbolic) function was often used as activation function in hidden layers and performed better than the sigmoid function due to the stronger gradient (Goodfellow, Bengio et al. 2016). However, recently tanh has been replaced as the main activation function in hidden layers by the ReLU (rectified linear unit) function, because it solves the vanishing gradient problem of the sigmoid and tanh activation function. Both the sigmoid and tanh function have segments in which the gradient is very small, which decreases the size of the weight updates, thus slowing down the training process. Additionally, ReLU is less computationally expensive than tanh or sigmoid due to the simpler mathematical operations involved (Nair and Hinton 2010). For each predicted output a loss function is used to quantify the prediction error and through a process called backpropagation the derivatives of the prediction error are used to update the randomly initialised weights and thus minimise the loss (Fig. 6D). The choice of loss functions usually depends on the output layer. For regression problems a mean squared error loss function can be chosen and for binary classification problems a binary cross-entropy loss function can be used (Goodfellow, Bengio et al. 2016). The minimisation of loss is performed through application of an optimisation algorithm, which is used to find the direction of steepest descent of the loss function (Le Roux, Bengio et al. 2011). Through iteratively updating the weights during training, the minimum of the loss function should eventually be reached. The process can be regulated through the learning rate, which determines the step size in which the weights are updated (Murphy 2012). A low learning rate is usually more precise but the gradient calculation is also more time consuming, which means the training process

will take longer. Most optimisation algorithms are based on gradient descent, which is a first-order iterative method to find the local minimum of a function (Bertsekas 1999). Gradient descent works best with convex loss functions that only have one local minimum but struggles with non-convex functions that have multiple minima (Fig. 6D). Furthermore, because the technique requires the entire dataset for optimisation, larger data makes this method computationally very expensive. The above mentioned problems can be solved by using stochastic gradient descent (SGD), which is more efficient at finding the global minimum of non-convex functions and uses only a batch of randomly selected samples instead of the entire dataset (Bottou and Bousquet 2007). Commonly used optimisation algorithms include different variants of stochastic gradient descent (SGD) such as Momentum or Adam (Rumelhart, Hinton et al. 1986, Diederik P. Kingma 2014). To ensure that no overfitting occurs while training an ANN, different methods of regularisation are applied. Regularisation controls the model complexity by adding penalty terms to the objective function, thereby preventing it to exactly fit the training data.

Regularisation techniques include data augmentation, dropout or early stopping (Fig. 6E). Data augmentation is used to increase the size of available training data by augmenting existing training data (for images this could be rotation, horizontal flips, shearing, etc.), thus increasing the variability of the training data (Mikołajczyk and Grochowski 2018). Dropout layers can be included to randomly drop certain outputs generated by the previous layer, thereby adding noise to the training process, which forces the model to not rely on any one feature and instead spread out the weights (Srivastava, Hinton et al. 2014). If an ANN model is trained too little, the model might be underfit but if the model is trained too long this can lead to overfitting. Early stopping can be used to prevent this problem by automatically stopping the training process if the validation error is increasing (Girosi, Jones et al. 1995). Another more recently

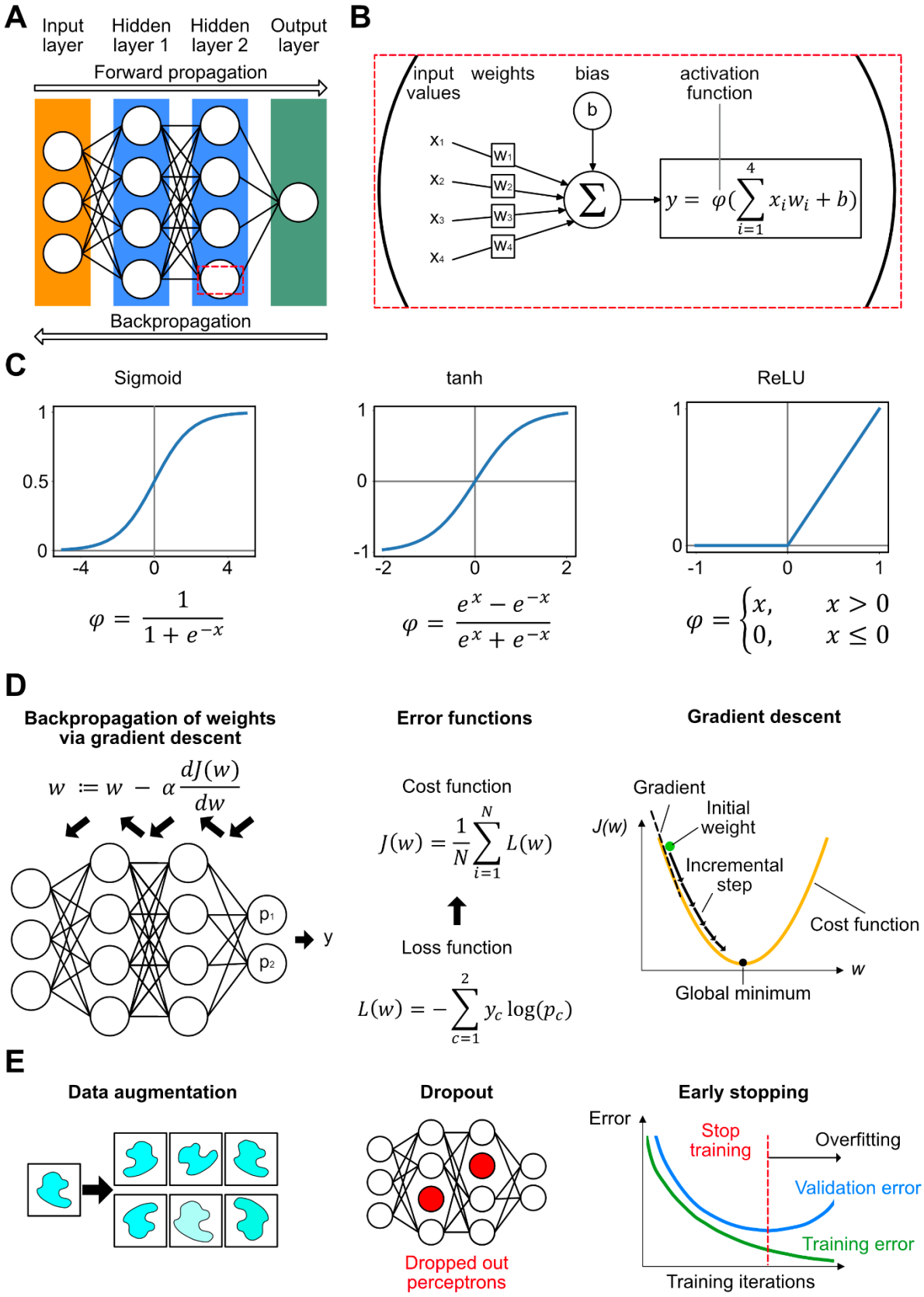


Figure 6. The basic principles of how Artificial Neural Networks (ANNs) work. (A) The architecture of a feedforward neural network consists of one input layer (orange), one or more hidden layers (blue) and the output layer (green), all of which are made up of interconnected perceptrons. Input

values are manipulated within the hidden layers and propagated towards the output layer, while weights are updated through backpropagation. **(B)** The inner workings of a perceptron. Incoming values from previous perceptrons are multiplied with different weights, summed together and a bias is added. An activation function is applied to introduce non-linearity, thus giving the ANN the ability to model complex functions. **(C)** Three commonly used activation functions include the sigmoid, tanh and ReLU function. The sigmoid function is commonly used as binary classifier in the output layer but due to the vanishing gradient problem it is not often applied in hidden layers. The tanh function although providing a stronger gradient than the sigmoid function, which improves the convergence towards the minimum loss, also suffers from the vanishing gradient problem. The ReLU function effectively avoids the vanishing gradient problem and due to its less complex mathematical nature is faster to compute. **(D)** How ANNs learn. In this example, a simple neural network with a binary output is shown. The predicted output value $\hat{y}$, the true output y and the probabilities p_1 and p_2 are used in a binary cross-entropy loss function to determine the loss. For all samples the losses calculated for each sample are summed together and divided by the number of samples to get the cost function. The chosen optimisation algorithm in this example is gradient descent, which works well for convex functions with only one local minimum. The purpose of gradient descent is to minimise the cost function $J(w)$ by updating the weights through backpropagation. The weights are updated by subtracting the current weight by the product of the learning rate α and the derivative of $J(w)$. To keep this example simple, the update of the bias through backpropagation was neglected. The learning rate determines the step size the weight updates take to reach the global minimum of the cost function. **(E)** Regularisation methods to prevent overfitting. A highly effective regularisation method is data augmentation, which involves the random manipulation (augmentation) of existing data to increase the variance of the training data and thus improve the models ability to generalise. Another popular method is dropout, which involves randomly dropping out a certain percentage of perceptrons (here indicated in red) in a dropout layer, which introduces noise and forces the network to learn from a sparse representation, which has been shown to reduce overfitting. Another simple method to prevent overfitting is early stopping in which the validation error is monitored and training is stopped when the validation error is increasing.

developed regularisation technique is batch normalisation, which has been shown to be highly effective when training very deep neural networks (Amodei, Ananthanarayanan et al. 2016, He, Zhang et al. 2016). Due to the changing parameters of prior layers, the input distribution of subsequent layers changes, which slows down the training by requiring reduced learning rates. This change in distribution of input values within a neural network is labelled as internal covariate shift. Batch normalisation aims to coordinate the parameter update of multiple layers

by standardising the activations of each input variable (Ioffe and Szegedy 2015). It furthermore has been found to simplify the optimisation function, which ensures more predictive gradients, usage of higher learning rates and faster loss convergence.

ANNs can also be used for image classification but have the limitation of only being applicable to very small images. Images are two- or multidimensional arrays of numbers and an RGB image of 200x200 pixels in size would yield 120,000 input features for the input layer. A typical hidden layer consisting of 1024 perceptrons would mean 122,880,000 weights for the first layer alone, which would take a considerable amount of time to train even on powerful GPUs. What regular ANNs neglect is the fact that pixels are most useful in the context of their neighbours. Furthermore since every pixel is assigned to a single perceptron the detection of certain features is heavily dependent on the objects position within the image, which might prove problematic if the position of the object changes. These problems have been addressed through the development of Convolutional Neural Networks (CNN) (LeCun, Haffner et al. 1999).

Convolutional Neural Networks

The main difference between CNNs and ANNs is the repeated application of convolution and pooling prior to feeding the data into a fully connected neural network (Fig. 7A). CNNs were inspired by the research of Hubel and Wiesel on the primary visual cortex of cats (Wiesel and Hubel 1963). They found that some neurons fired rapidly by watching lines at specific angles, while other neurons responded to light and dark patterns or detect motion in different directions (Wiesel and Hubel 1963, Hubel and Wiesel 1970). Although CNNs were invented in the 1980s (Fukushima 1980), their widespread usage began only in the 2000s after the performance of graphics processing units (GPUs) had sufficiently increased. Today CNNs are often used in image recognition systems such as facial recognition due to their low error rate (Lawrence,

Giles et al. 1997). They have also been successfully used for natural language processing, predicting interactions between drugs and proteins and were also implemented in AlphaGo (Grefenstette, Blunsom et al. 2014, Wallach, Dzamba et al. 2015, Silver, Schrittwieser et al. 2017). Multidimensional arrays fed into a CNN are usually first passed to a convolutional layer, which performs a convolutional operation on the input. Convolution in the context of CNNs describes the application of small filters (usually 3x3 pixels) to enhance specific features within the array (Fig. 7B). A Gaussian blur filter (or Gaussian kernel) as mentioned in the previous chapter or a vertical line detector are two examples for convolutional operations. The output images generated by convolving the input images with different filters are called feature maps and just like in a regular ANN, each value within a convolutional filter is assigned a weight and bias. The size and number of feature maps is controlled by the filters size, depth, stride and zero-padding (Dumoulin and Visin 2016). Depth describes the number of filters that should be applied and determines the number of generated feature maps. The stride is the number of pixels a convolutional filter is moved across the input image, which means that a greater stride will result in smaller feature maps (Fig. 7B). Zero-padding allows to directly control the feature map size by adding zeros around the input image border. After the convolutional operation (which is linear) an activation function, such as ReLU is applied to introduce non-linearity. The next step is pooling, which reduces the dimensionality of each feature map by applying different pooling operations such as max, average or sum (Goodfellow, Bengio et al. 2016). By sliding a 2x2 pixel pooling window, such as max pooling, across the feature map, only the highest value within the 2x2 pooling window will be retained, thus decreasing the size of the pooled feature map (Fig. 7B). Because pooling reduces the number of network parameters and makes the network invariant to minor transformations, it effectively prevents overfitting and furthermore decreases the amount of necessary computations, thus improving training speed (Goodfellow, Bengio et al. 2016). Multiple segments of convolution, non-linear activation and

pooling can be applied in CNNs to improve the models ability to classify more complex images. In a classical CNN, this is followed by a fully connected layer, which is a traditional ANN just like described in the previous chapter. Classical CNNs only output a predicted label but no information about the exact location of the label within the image. Semantic segmentation, which is a pixel-based classification, can be achieved through application of fully convolutional networks (FCNs) (Long, Shelhamer et al. 2014). Unlike regular CNNs, FCNs transform height and width of the spatially reduced feature map back to the size of the original input image and assigns a label to each pixel. This is possible through applying transposed convolution, which is an upsampling method combined with convolution to increase the size of the input feature map (Fig. 7C) (Dumoulin and Visin 2016). The resulting enlarged feature map contains the necessary feature information for classification but lacks spatial information due to repeated application of convolution and pooling. Through element-wise addition of the up-sampled feature map with a previously generated feature map of identical size with more spatial information, both feature and location information can be enhanced (Fig. 7C).

There are many different existing architectures of convolutional neural networks, one of the oldest being the LeNet, a 7-level network that classifies digits (Lecun, Bottou et al. 1998). The more recent CNN, called AlexNet won the ImageNet Large Scale Visual Recognition Challenge in 2012 and has since been cited more than 50,000 times (Krizhevsky, Sutskever et al. 2012). Other successful architectures include the GoogleLeNet, the ResNet and the U-Net, a modified FCN designed for segmentation of biomedical images (Szegedy, Liu et al. 2014, Ronneberger, Fischer et al. 2015, He, Zhang et al. 2016).

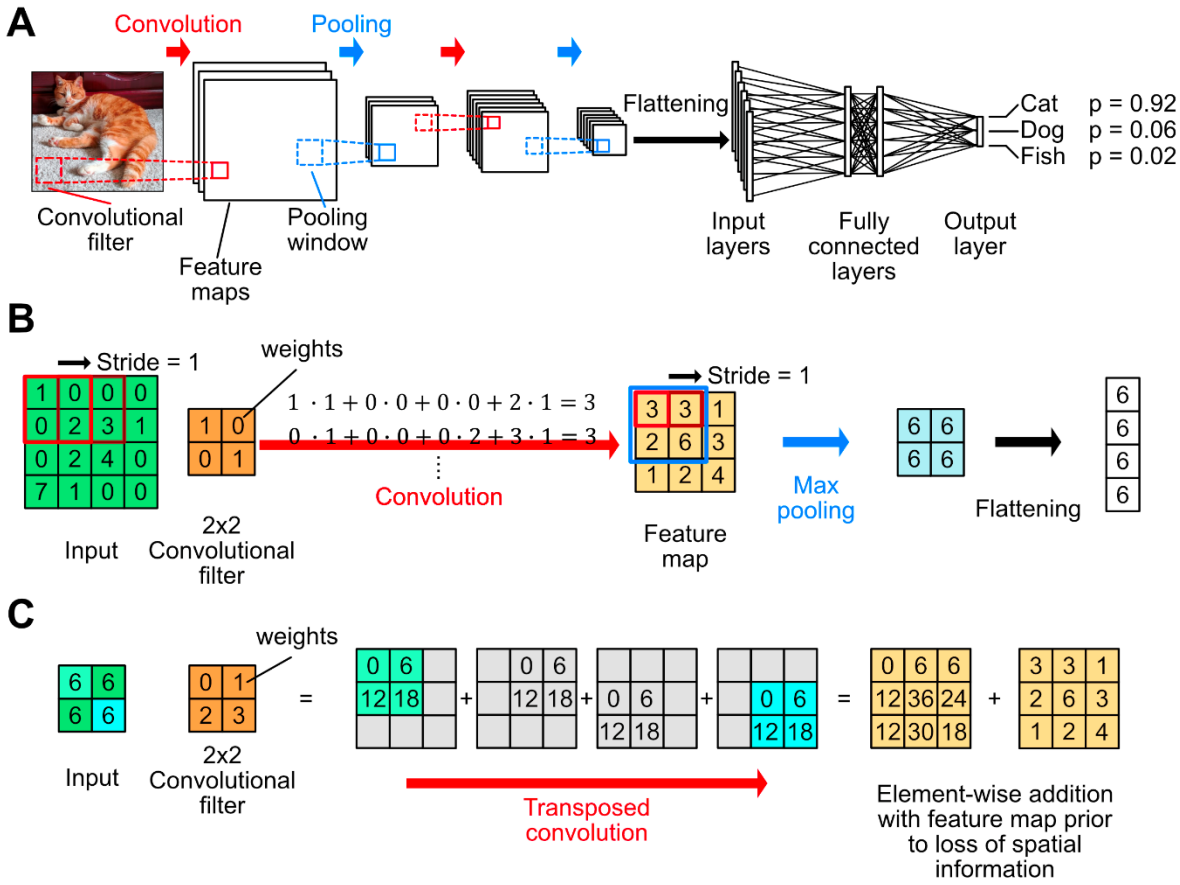


Figure 7. Architecture of Convolutional Neural Networks and basics of convolution arithmetic. **(A)** Multiple different convolutional filters are moved across the input image to generate feature maps which are then subjected to pooling in which a pooling window summarizes spatial information and thereby reduces the size of the feature map. This figure does not show the application of non-linear activation functions to the feature maps generated after convolution. The process of convolution, non-linear activation and pooling can be repeated multiple times before flattening reduces the 2D array to a 1D array, ready to be used as input layers of the ANN. The hidden layers (or fully connected layers) function in the same way as described in the previous chapter and the output layer generates probabilities to which class the input image belongs to. **(B)** A 4x4 array is convolved with a 2x2 convolutional filter in which the values of the filter represent weights than are learned during the training process. The convolutional filter is moved across the input array with a stride of 1 and generated a 3x3 feature map. Because no zero-padding was added in this example, the feature map is smaller than the input. A 2x2 max pooling window with a stride of 1 is used to retain only the maximum value of the feature map, leaving a 2x2 array of identical values. The last step prior to classification in an ANN is flattening. **(C)** The process of transposed convolution is important for semantic segmentation. Here the input represents the spatially reduced 2x2 array from (B) and is subjected to a 2x2 convolutional filter. Each value of the input is multiplied element-wise with the convolutional filter, resulting in four separate arrays, which when summed up result in the up-sampled feature map. To combine the feature information, which is gained through repeated convolution and pooling operations, with the spatial

information, the resulting feature map can be fused with an identical shaped feature map prior to pooling. Figure concept based on (Dumoulin and Visin 2016).

The U-Net architecture

The U-Net addresses a common problem when training CNNs, which is the unavailability of large amounts of training data. This holds especially true for life science research where the acquisition and annotation of millions of images if often not feasible. In 2015, (Ronneberger, Fischer et al. 2015)) developed the U-Net based on the FCN architecture outlined in Long et al. (2014) to improve the task of biomedical image segmentation previously attempted by Cireşan et al. (2012), which used a regular CNN. The U-Net was shown to work well with just 30 (512x512 pixel) 2D images used as training data and was shown to outperform the network of Cireşan, et al. (2012). Its architecture is based on a contracting and expansive path illustrated in Figure 8. A weight map is used to compensate for the imbalanced frequency of pixel classes in the training data and to force the network to learn separation borders between touching objects (formula defined in methods section).

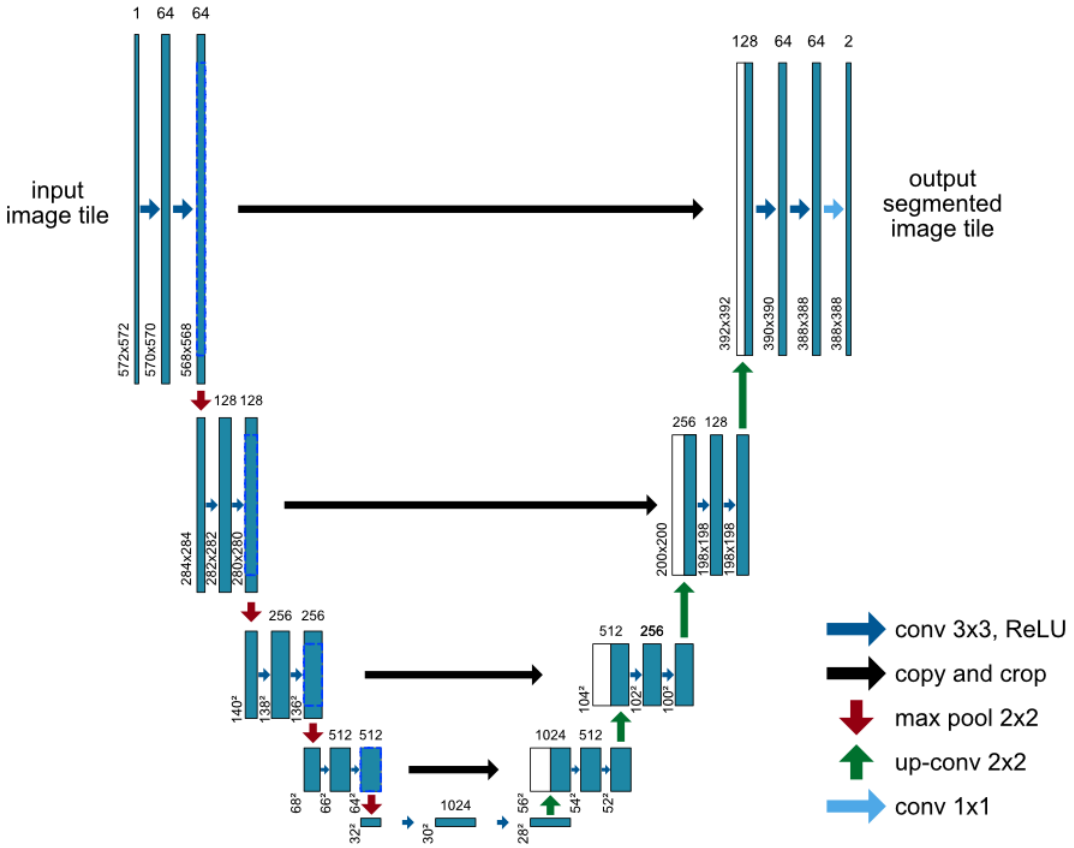


Figure 8. U-Net architecture. Each blue line represents a feature map, the thickness corresponds to the number of feature maps (indicated at the top of each blue box) and the length of each line corresponds to the feature map size (shown below). The contracting pathway represents a classical CNN (convolution > non-linear activation > max pooling), which aims to classify the input image. The expansive pathway uses transposed convolutions (green arrows) to restore the original input image size and fuses the up-sampled feature maps with feature maps from the contracting pathway that contain more spatial information (indicated by grey arrows). The original U-Net applies unpadded convolutions, which is why the feature map size gradually decreases and the output image is smaller than the input image. Based on (Ronneberger, Fischer et al. 2015).

Aim of this study

Commonly used segmentation methods often fail to provide reliable and consistent segmentation results, thus preventing accurate quantification. Although deep learning segmentation is now well established, its usage often requires profound programming skills and knowledge of artificial neural networks. The few deep learning tools available to biologists

that come with a graphical user interface require installation of various frameworks, additional software and changing system configurations via the terminal, thereby making these tools unnecessarily difficult to use for biologists with little expertise in computer science. By using a modified version of the fully convolutional neural network U-Net, I developed a model specifically trained to accurately segment mitochondria in adult *C. elegans*. The MitoSegNet segmentation performance was tested against three feature detection methods and the machine learning software ilastik. The segmentation model was generated in combination with two custom-built computational tools, the MitoS segmentation and MitoA analysis tool that enable high-accuracy segmentation of 2D microscopy images together with morphological analysis. The graphical user interface tools require no prerequisite installation of additional software or frameworks as all dependencies are contained within the executable files. The main work presented in this PhD thesis aims to address the lack of easy-to-use tools enabling biologists without computational expertise to use deep learning segmentation. Furthermore, I present two feature detection segmentation methods implemented in ImageJ that were used for mitochondrial membrane potential measurements in two other *C. elegans* studies. Additionally, the completed MitoSegNet was used in a third study to quantify mitochondrial membrane potential and mitochondrial oxidative stress levels in *C. elegans*.

Materials and Methods

Main project: MitoSegNet

Generating the MitoSegNet model

The MitoSegNet model was trained for 20 epochs with 12 fluorescence microscopy images, depicting adult *C. elegans* mitochondria in body wall muscle cells. Each image is 1300 x 1030 pixels (px) in size and mitochondria were labelled with GFP ($P_{myo\text{-}3}$*::mitoGFP*) (Rolland, Motori et al. 2013). The images were recorded on a fluorescence microscope equipped with a 63x 1.4 NA oil lens (Axioskop 2; Carl Zeiss Inc.) and a charge-coupled device camera (1300; Micromax). Two annotators manually segmented a set of 6 microscopy images. Raw 3D microscopy images were converted to 2D images through maximum intensity projection. Based on the annotation of four mitochondrial phenotypes (elongated, fragmented, mixed and tubular), 3 images of each phenotype were used for training. Prior to augmentation each image was divided into four overlapping 656 x 656 px images due to GPU memory constraints. The input tile size was chosen to allow for the 2x2 max-pooling operations during training. Each image (and the corresponding ground truth) underwent random augmentation using the Keras library, producing 80 differently augmented images per input image, which involved random shearing (in a range of 30%), rotations (in a range of 180°), change of zoom (in a range of 30%), brightness change (in a range of 20%), horizontal and vertical flipping, x- and y-shifts (in a range of 20%) and mirroring. This increased the total number of images used for training from 48 to 3,840. I used a batch size of 1 and included the weight map as implemented in the U-Net publication and set the class balance factor w_{bal} to 0.042 (1 divided by the number of background pixels per object pixel to achieve a 1:1 ratio between object and background

pixels). The MitoSegNet model was trained at a learning rate of $7 \cdot 10^{-05}$ for 20 epochs and reached its minimum validation loss at epoch 18 (Fig. 9A).

MitoSegNet model prediction

To evaluate the segmentation performance of the MitoSegNet model on all 12 images, a cross validation was performed, and 12 separately trained models were generated (in the same manner as described in the previous chapter). The dice coefficient, a statistic metric used for comparing the similarity of two binary datasets (a more detailed explanation can be found on page 49), and the binary cross entropy loss (see page 48) were chosen to quantify the segmentation performance. Each model was trained with 11 images for 15 epochs and used to predict the segmentation of the test image that was excluded from training (Fig. 9B). Training and prediction were performed using the Python-based MitoS tool on a Devuan GNU/Linux server with a Nvidia TITAN X (12 GB GDDR5 RAM), using CUDA 10.0 and cuDNN 7.6.1.

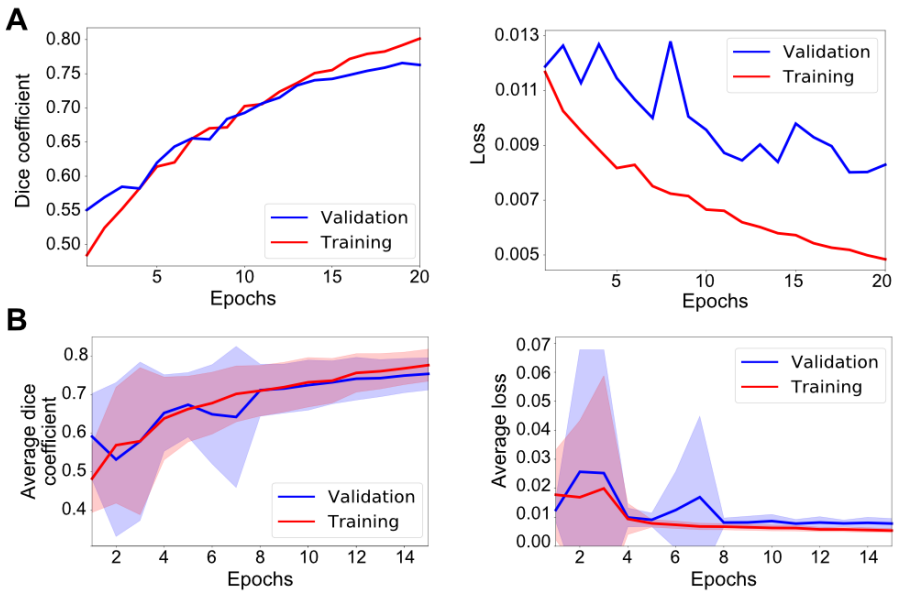


Figure 9. Training performance of MitoSegNet training on 12 images. (A) Training (red) and validation (blue) dice coefficient and loss of final MitoSegNet model trained on 12 images. After epoch 15, discrepancy between training and validation dice coefficient begins to increase. Training and validation loss decrease and reaches its minimum of 0.008 at 18 epochs. **(B)** Average training performance of 12 MitoSegNet models trained on 11 images for cross validation. Average validation (blue) and training (red) dice coefficient steadily increases over 15 epochs of training. Average training

loss decreases as expected, while validation loss enters a plateau after 8 epochs. Blue and red areas indicate the standard deviation added (upper border) or subtracted (lower border) from the average of the validation and training metric, respectively.

MitoS segmentation and MitoA analyser tool

The MitoS and MitoA tools were written with Python 3.7.3 and converted to standalone executable files using the PyInstaller 3.5 application. Because PyInstaller is not a cross-compiler, the same Python scripts were bundled separately on a Windows 10 platform and on a Devuan GNU/Linux system. Below are tables containing the names of all Python scripts that were used to create the MitoS (CPU/GPU) and MitoA tools.

The following tables contain the name and purpose of all Python scripts and libraries used for the MitoS and MitoA tools.

Table 1. CPU-specific Python scripts for the MitoS tool

Python script	Purpose
MitoS_Main_CPU.py	Graphical user interface (CPU-only)
MitoS_Train_Predict_CPU.py	MitoSegNet model training and prediction

Table 2. GPU-specific Python scripts for the MitoS tool

Python script	Purpose
MitoS_Main_GPU.py	Graphical user interface (CPU / GPU)
MitoS_Train_Predict_GPU.py	MitoSegNet model training and prediction

Table 3. CPU/GPU independent Python scripts for the MitoS tool

Python script	Purpose
MitoS_Create_Project.py	Create project folder in advanced mode
MitoS_Training_DataGenerator.py	Generate weight maps and augment data
MitoS_Train_Val_Analyser.py	Plot training and validation accuracy

Table 4. Python scripts for the MitoA tool

Python script	Purpose
MitoA_Main.py	Graphical user interface and module functions
Plot_Significance.py	Plot statistical significance if detected

Table 5. Python libraries used for the MitoS tool

Python library	Purpose
copy	Shallow and deep copy operations
cv2	Computer vision and machine learning
glob	Finding and matching pathnames based on Unix shell rules
keras	High-level neural network API
math	Mathematical functions defined by the C standard
matplotlib	2D plotting
numpy	High-level mathematical functions on multi-dimensional arrays and matrices
os	Operating system dependent functionality
pandas	Data manipulation and analysis
pathlib	OS-appropriate filesystem paths
re	Regular expression matching operations

scipy	Scientific and technical computing
seaborn	2D plotting based on matplotlib
shutil	High-level file operations
scikit-image	Collection of image processing algorithms
sys	Access to variables used by interpreter
tensorflow / tensorflow-gpu	High- and low-level machine learning API with or without GPU support
time	Time-related functions
tkinter	Graphical user interface package
warnings	Warning control
webbrowser	High-level interface to display web-based documents

Table 6. Python libraries used for the MitoA tool

Python library	Purpose
collections	Specialised container datatypes
copy	Shallow and deep copy operations
itertools	Iterator building blocks
matplotlib	2D plotting
numpy	High-level mathematical functions on multi-dimensional arrays and matrices
os	Operating system dependent functionality
scipy	Scientific and technical computing
seaborn	2D plotting based on matplotlib
skan	Analyse object skeletons in images

scikit-image	Collection of image processing algorithms
tkinter	Graphical user interface package
webbrowser	High-level interface to display web-based documents

The MitoS tool is can be used for deep learning-based segmentation of 2D microscopy images. The tool can be executed in two modes, the "basic mode" and the "advanced mode". The "basic mode" is designed for researchers with no prior deep learning experience (Fig. 10A). The MitoS basic mode uses the pretrained mitochondria specific MitoSegNet model and images of mitochondria similar to those used for training can be segmented with the MitoS tool (Fig. 10A). If the MitoSegNet model segmentation does not yield satisfactory results, self-generated training data can be used to finetune the existing segmentation model (Fig. 11A). The "advanced mode" is intended for researchers with deep learning experience and allows users to customise training parameters (Fig. 12). For those users that do not have a CUDA-capable GPU, a CPU-only version of the MitoS segmentation tool is available. Because the MitoSegNet model was generated using 656 x 656 px image tiles, any images intended for segmentation are fitted to this pre-set tile size. Images larger than the pre-set tile size are split into overlapping tiles with a mirrored border to avoid prediction in border regions. Smaller images are enlarged by adding a mirrored border to match the pre-set tile size. After prediction, the mirrored parts of the tiles are removed, and the tiles are stitched back together. A final threshold is applied to the fully stitched image to convert it to an 8-bit binary mask ($p_i < 128$ to 0 and $p_i >= 128$ to 255). Furthermore, any objects below 10 px in size are considered to be noise and are thus removed from the final segmentation image. The finetuning module automatically generates weight maps and augments the novel training data using the following augmentation operations: shearing (in a range of 30%), rotation (in range of 180°), zoom (in a range of 30%) and brightness change (in a range of 20%), horizontal and vertical flip, width and height shift

(in a range of 20%). Once augmentation is completed, model training begins with a pre-set learning rate of 0.0001, a batch size of 1 and a class balance factor of 1/(foreground to background pixel ratio). I tested the MitoS finetuning module on images depicting adult *C. elegans* mitochondria visualised with a non-integrated $P_{myo\text{-}3}$*::mitoGFP* reporter (Rolland, Motori et al. 2013). The intensity of the fluorescently labelled mitochondria appeared weaker and hence the MitoSegNet model failed to accurately segment these images (Fig. 11B). One image of weak fluorescent intensity mitochondria was therefore segmented by hand and used to finetune the pretrained model for 10 epochs, generating a finetuned MitoSegNet model that generated satisfactory segmentation results (Fig. 11B). Alternatively, it is possible to create a custom segmentation model using the advanced mode of the MitoS tool (Fig. 12). The advanced mode consists of four modules: Module 1 automatically generates a folder structure that is later used for model training. Module 2 enables data augmentation, which allows the user to specify the range of the above listed augmentation operations. Module 3 carries out the model training for which the parameters can be freely chosen. Module 4 predicts new segmentations using the fully trained model (Fig. 12).

The MitoA analyser tool is a separate Python-based application that can be run after successful segmentation for quantification and visualisation of potential morphological differences (Fig. 13). It measures 13 different features using both raw and segmented images for each object: area, minor and major axis length, eccentricity, perimeter, solidity, mean, max and min intensity, number of branches, branch length, total branch length and curvature index. The minor or major axis lengths are defined as the lengths of the line segment connecting the two co-vertices or vertices of an ellipse fitted around an object. The eccentricity of a conic section is a non-negative real number that characterizes its shape and the eccentricity for a circle, ellipse and parabola are 0, $0 > x > 1$ or 1 respectively. The solidity is defined as the object area

divided by the convex object area. The curvature index indicates if the branches of the mitochondria are straight or curved and is defined as

$$c_i = \frac{branch\ length - euclidean\ distance}{euclidean\ distance}$$

The Euclidean distance is the shortest path between the start and end point of one branch. The single object features are summarised for each image as average, median, standard deviation, standard error, minimum and maximum and saved in an Excel table. One table is generated for a group of images, which can subsequently be compared. Both the two- and multi-comparison of samples allows to generate a table containing statistically relevant information and a visualisation using boxplots. Each statistical table contains the descriptors p-value of the D'Agostino's K-squared test to determine if the data is normally distributed (D'Agostino 1971, D'Agostino and Pearson 1973). Based on this result and the Levene's test p-value to test for equality of variances an appropriate hypothesis test is chosen and the name of the test, including its p-value are displayed in the table (Levene 1960). For comparison of two normally distributed datasets with equal variance, a Student's t-test is used. If either of the two criteria are not met, the Mann–Whitney U test is used. For comparison of multiple samples with equal variance and normal distributions, a one-way ANOVA is used or alternatively, the Kruskal-Wallis-Test is applied (Kruskal and Wallis 1952). For a two-sample comparison the effect size is also calculated, and the boxplot visualisation indicates statistically significant differences. Furthermore, the two-sample comparison also enables the user to perform a correlation analysis. The correlation analysis visualises correlation by letting the user select up to four different descriptors to correlate against each other in both samples and then have them displayed as scatterplots. The distributions are subjected to the D'Agostino's K-squared test to determine the usage of the Pearson or Spearman rank-order correlation. Correlation coefficients and p-values are displayed in the MitoA terminal or can be saved to an Excel file.

The MitoSegNet Analyser was used for statistical testing and visualisation of morphological differences for Figure 16 and 17.

The MitoSegNet architecture

The MitoSegNet architecture is based on the previously published U-Net (Fig. 8) (Ronneberger, Fischer et al. 2015) and was implemented in Keras (Python 3.7.3). It consists of a contracting path, which follows the standard architecture of convolutional neural networks, consisting of repeated application of 3x3 convolutions (padded convolutions), each followed by a batch normalization layer, a rectified linear unit (ReLU) and a 2x2 max pooling operation with a stride of 2. Unlike the original U-Net, the MitoSegNet does not utilize any drop-out layers at the end of the contracting path as I have found the batch normalization layer to reduce the training time. After 1024 feature channels have been generated, the expanding pathway uses 2x2 up-convolutions to halve the number of channels, followed by a concatenation with the corresponding feature map from the contracting path and subsequent 3x3 convolutions, followed by a ReLU. The final convolutional layer (1x1) is followed by a sigmoid function. In total, the MitoSegNet consists of 24 convolutional layers. The optimization algorithm chosen for the training process is Adam (adaptive moment estimation), which is an extension to stochastic gradient descent (Diederik P. Kingma 2014). During training, 20% of the data is excluded from the process and instead used for model validation after each training epoch. The energy function is computed by a pixel-wise sigmoid function over the final feature map combined with a binary cross entropy loss function

$$p_k(x) = \frac{1}{1 + e^{-a_k(x)}}$$

where $a_k(x)$ denotes the activation in feature channel k at the pixel position x and $p_k(x)$ is the approximated maximum-function for that feature channel. The binary cross entropy is then used to calculate the loss at each pixel

$$E = -\sum_{i=1}^{2} w \cdot y_i' \cdot \log(y_i)$$

where y_i is the predicted probability of the class i, y_i' is the true probability for that class and w is the weight map. The separation weight map w_{sep} is implemented as described in (Ronneberger, Fischer et al. 2015) and prevents touching objects to be segmented as one object by increasing the weights in border regions. The MitoSegNet also includes a class balancing weight map, which decreases the weight of background pixels by a factor of w_{bal} so the final weight map is defined as

$$w = w_{bal} + w_{sep}$$

Just as in the original U-Net, initial weights are drawn from a Gaussian distribution with a standard deviation of $\sqrt{2/N}$, where N is the number of incoming nodes of one neuron.

Other segmentation methods

All segmentation methods include the removal of uninformative image slices from the stack and the remaining stack is reduced to a single image via maximum intensity projection. Prior to all ImageJ-based segmentation methods a background subtraction (rolling ball algorithm) is applied (Schneider, Rasband et al. 2012). After applying the three different blob detector-based segmentation approaches with ImageJ, they are all followed by a final filter step in which all particles smaller than 10 px in size are removed from the final mask.

The method denoted as **Laplacian** is based on the de Boer lab workflow on quantitative analysis of mitochondrial morphology in *C. elegans* (de Boer, Smith et al. 2015). It is based on the application of a local contrast enhancement and subsequent object enhancement using a multi-scale Laplacian operator (De Vos, Van Neste et al. 2010). The images are then binarized according to a Yen autothresholding procedure (Yen, Chang et al. 1995).

The **Hessian** method calculates eigenvalues of a Hessian matrix using the Tubeness plugin in Fiji (Sato, Nakajima et al. 1998). To generate a binary image the IsoData autothresholding is used (Huang and Wang 1995).

The **Gaussian** method calculates the difference of Gaussians and the binary mask is generated using default autothresholding.

To also include a machine learning segmentation approach, I used the open-source software **ilastik**. This software learns from labels provided by the user, using a random forest classifier in the learning step, in which each pixel's neighbourhood is characterized by a set of nonlinear features (Kreshuk and Zhang 2019). To train the classifier, 12 images were used for training. Each appendant label was created with ilastik by partially marking objects of interest and background.

Segmentation performance on test set

The segmentation performance of the five approaches was evaluated with different measures, each focusing on particular aspects of the segmentation result with specific limitations. The Dice coefficient (dc, also known as F1 score) (Dice 1945, Sørensen 1948) is a statistic value used for comparing the similarity of two binary datasets.

$$dc = \frac{2 \cdot |ground\ truth\ \cap\ prediction|}{|ground\ truth| + |prediction|}$$

On each image, the dc is evaluated on all pixels. The distributions of dice coefficients from all images were tested for normality using the D'Agostino's K-squared test and statistical difference was determined using the Kruskal-Wallis test followed by Dunn's multiple comparisons test. Upon comparing the dc's with the visual segmentation results, I realised that the dc values do not fully reflect the morphological segmentation accuracy (see Fig. 15B).

To obtain information about morphological segmentation accuracy, I compared single object shapes. The single object shape comparison uses the following shape descriptors for comparison: area, eccentricity, aspect ratio (dividing the major axis length by the minor axis length), perimeter and solidity. The difference between predicted and ground truth shape descriptor is calculated as fold deviation for each object and defined as

$$sd_{dev} = \frac{|sd_p - sd_{gt}|}{sd_p}$$

where sd_p is the predicted and sd_{gt} the ground truth shape descriptor. Object correspondence was assumed if at least one identical pixel coordinate was found in both the ground truth and the predicted object. Only objects that were predicted to correspond with a single object in the ground truth or vice versa were included in this analysis. For single correspondence, a fold deviation (the predicted value multiplied or divided by that fold deviation would yield the ground truth value) for all five morphological descriptors from the ground truth is then calculated for each object (Fig. 15C). The single object fold deviation values were tested for normality using the D'Agostino's K-squared test and statistical differences between methods were determined using the Kruskal-Wallis test followed by Dunn's multiple comparisons test. Because the single object shape comparison neglects false positive predictions, I compared the predicted shape distribution of all objects per image against the ground truth. The shape descriptors are the same as used for the single object shape comparison. For segmented objects,

distributions of these five descriptors were obtained and compared to the distributions of the ground truth images by calculating the energy distance (Fig. 15E).

The energy distance between two distributions d1 and d2 is defined as

$$D(d1, d2) = (2E\,|X - Y| - E|X - X'| - E|Y - Y'|)^{1/2}$$

where X and X' (resp. Y and Y') are independent random variables with a probability distribution of *d1* (resp. *d2*) and E is the expected value (Szekely 2002). Energy distance values were normalised prior to being tested for normality using the D'Agostino's K-squared test, followed by applying a Kruskal-Wallis test and a subsequent Dunn's multiple comparisons test.

All methods to test the performance of each segmentation approach were implemented in Python 3.7.3 using the following libraries: NumPy, OpenCV, scikit-image, scikit-learn, scikit-posthocs, Matplotlib, Seaborn, Pandas and SciPy.

Comparison of mitochondrial morphology between *catp-6* mutant and wild type

Images segmented with the pretrained segmentation model were recorded with a fluorescent microscope, using a 63x 1.4 NA oil lens (Axioskop 2; Carl Zeiss Inc.) and a charge-coupled device camera (1300; Micromax). The *C. elegans* mutant *catp-6(ok3473)* was compared to wild type and mitochondria were made visible with the usage of the $P_{myo\text{-}3}$*::mitoGFP* reporter. Quantitative analysis was performed using the MitoA tool.

Application of the MitoSegNet model segmentation to mitochondria in HeLa cells

Images segmented with the pretrained MitoSegNet were recorded with an inverted Zeiss LSM 880 system equipped with a DPSS 561-nm laser, using a Plan-Apochromat 63x / 1.4 oil DIC objective and a GaAsP detector. Images were collected with the ZEN 2 software at 1024 x

1024 pixels resolution. To visualise mitochondria, HeLa cells (ATCC) were transfected with mitoRFP using Lipofectamine 2000. After 24 hours of transfection, the HeLa cells were either left untreated or treated with 2.5 µM oligomycin (Calbiochem) and 1 µM antimycin A (Sigma) in fresh growth medium for 3 hours. Segmented and raw images were subjected to quantitative analysis using the MitoA tool.

Side project 1: Compromised Mitochondrial Protein Import Acts as a Signal for UPR^{mt}

L2/L3 larvae were inoculated on NGM plates (Stiernagle, 2006) supplemented with 0.1µM Tetramethylrhodamine, Ethyl Ester (TMRE) with a small inoculum of OP50 E. coli bacteria. After an incubation over-night at 20°C, L4 larvae were analysed by fluorescence microscopy using a microscope equipped with a 63× 1.4 NA oil lens (Axioskop 2; Carl Zeiss, Inc.) and a charge-coupled device camera (1300; Micromax). The acquisition was performed with 100 ms exposure using the software Metamorph (Molecular Devices). For the RNAi experiments, L4 larvae were inoculated onto 6mM IPTG RNAi plates as indicated above. After 2 days, L2/L3 larvae of the F1 generation were inoculated on TMRE plates and analysed the next day as indicated above (Rolland, Schneid et al. 2019).

Image segmentation was implemented in ImageJ using the IJ1 macro language. First, a background subtraction was performed using the "rolling ball" algorithm with a radius of 15 pixels to remove continuous background signal from the image (Sternberg, 1983). This was followed by the application of the Tubeness plugin, which generates a score of how tube-like each point in the image is (Sato et al., 1998). The resulting 32-bit image was converted to 8-bit and an automatic threshold (using the IsoData algorithm) was used to generate a binary mask. The final step involved the removal of any particles that are smaller than 10 pixels in size for they are assumed to be noise. After manually removing any remaining unwanted objects,

another IJ1 macro was used to measure the mean fluorescence intensity. All objects in the binary mask segmented with the first macro were selected and this selection was restored on the original image, allowing to measure the mean fluorescence intensity within regions that correspond to the binary mask. This binary mask corresponds to the TMRE labelled mitochondria in the image. The mean fluorescence intensity outside the mask was defined as the background and subtracted from the signal. Since the mean fluorescence intensity corresponds to the sum of the grey values of all the pixels in the selection divided by the number of pixels in the selection, the values correspond to fluorescence intensities per area.

Side project 2: Autophagy compensates for defects in mitochondrial dynamics

TMRE stainings were performed with the F1 generation of respective RNAi treatments. L2 larvae were inoculated onto plates containing 0.1 µM TMRE (Thermo Life Sciences T669) and imaged during the L4 stage using a 63x objective on the Zeiss Axioskop 2 and MetaMorph software (Molecular Devices).

The image was first converted to an 8-bit image, after which the continuous background signal was removed through background subtraction using the rolling ball algorithm with a ball radius of 15 pixels. To remove remaining noise, two filters were applied. The first being a minimum filter with a value of 1, therefore replacing each pixel in the image with the smallest pixel value in a particular pixel‘s neighbourhood. This was followed by a mean filter with a radius of 2, which replaces each pixel with the neighbourhood mean. Next, the Tubeness plugin was run with a sigma value of 1.0. The resulting 32-bit image was converted back to 8-bit and an automatic threshold (using the IsoData algorithm) generates a binary mask. The final step involved the removal of any particles that are smaller than 10 pixels in size for they are assumed to be noise. Raw image files were opened in parallel to their appendant binary masks (generated

by the segmentation macro) and a mask-based selection was created in the raw image. Within this selection measurements were obtained in the raw image and collected in a table for subsequent analysis.

Both ImageJ segmentation methods were developed and implemented prior to the completion of the MitoSegNet and can hence in future be replaced with the superior segmentation method offered by MitoSegNet.

Side project 3: ATP13A2-mediated endo-lysosomal polyamine export provides a mitochondrial antioxidant response

Mitochondrial oxidative stress in *C. elegans* was visualised by adding MitoSOX Red dye to a final concentration of 20 µM in NGM plates. After overnight incubation at 20°C in the dark, animals during the L3 stage were imaged using 63x objective on the Zeiss Axioskop 2 and MetaMorph software (Molecular Devices). Images depicting the stained mitochondria were segmented using the MitoSegNet and quantified with a custom ImageJ macro that automatically opens segmented and raw images and uses the mask to measure the intensity in the raw images. Mitochondrial membrane potential in *C. elegans* was made visible by addition of TMRE up to a final concentration of 0.1 µM in NGM plates. L3 worms were visualised after overnight incubation at 20°C in the dark with an 63x objective on the Zeiss Axioskop 2 and MetaMorph software (Molecular Devices). Images depicting the stained mitochondria were segmented using the MitoSegNet and quantified using the same custom ImageJ macro as for the MitoSOX Red intensity measurements.

Results

Main project: MitoSegNet

Generating the MitoSegNet model

The MitoSegNet model was generated by training a modified U-Net with a training set of 12 1300 x 1030 px fluorescent microscopy, maximum-intensity projection images, depicting mitochondria in body wall muscle cells of adult *C. elegans* worms (Fig. 10A). The U-Net modification entails the removal of dropout layers at the end of the contracting pathway and instead placing batch normalisation layers after every convolutional layer prior to ReLU activation in the contracting pathway. It was found that this modification decreased the amount of necessary training time. The $P_{myo\text{-}3}$*::mitoGFP* reporter was used to visualise mitochondria and each image was split into 4 overlapping tiles, of which each 80 augmented copies were generated for training the model. Further information can be found in the methods section. A cross validation was performed to estimate the test performance of the MitoSegNet model and compare it against other segmentation methods (Fig. 10B).

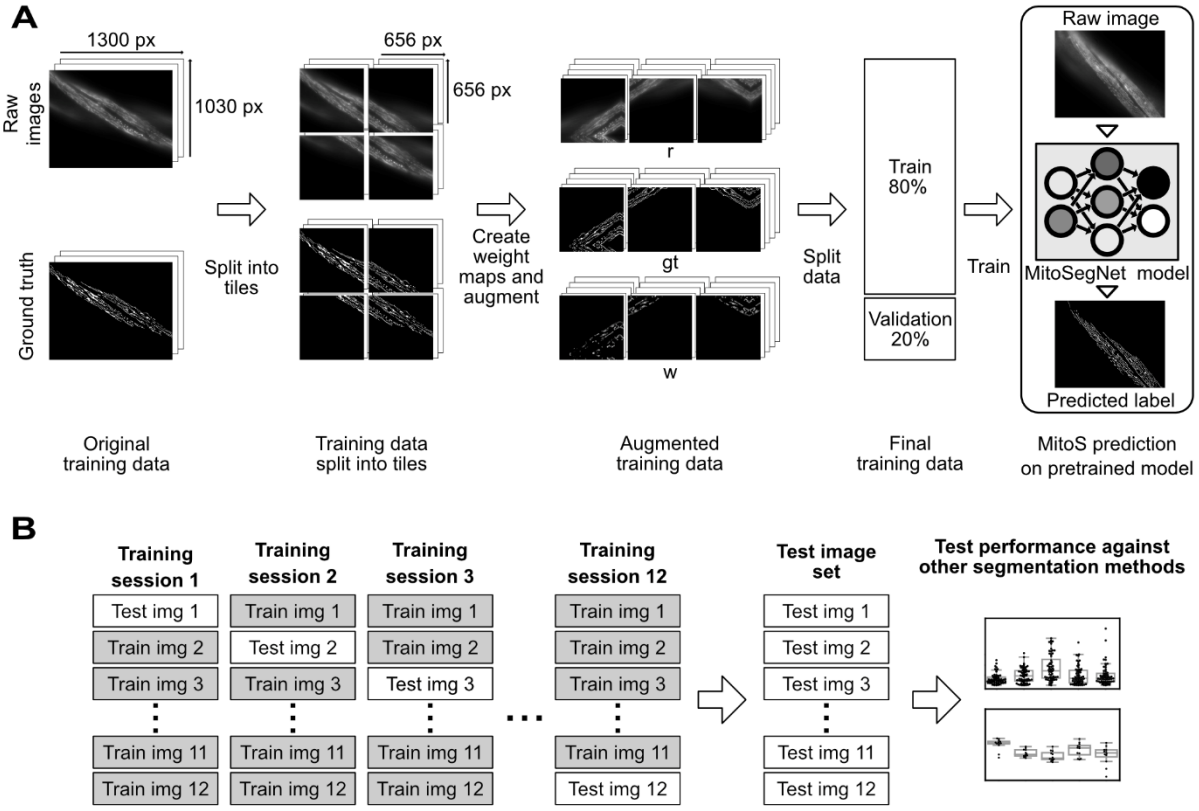


Figure 10. Training the MitoSegNet model and using it with the MitoS tool. (A) The original training data is comprised of 12 raw images and the appendant hand-generated ground truth images. Each image is split into 4 overlapping tiles of equal tile length. For each tile, a weight map is generated and subsequently all three set of tiles (r: raw images, gt: ground truth, w: weight maps) are augmented 80 times, increasing the size of the training data to 3,840 image tiles. Prior to training the MitoSegNet model, the augmented training data is split into training (80%) and validation data (20%). With the MitoS tool, the pretrained MitoSegNet model can be easily used to segment new images of mitochondria. **(B)** To also include test data, I performed a cross validation for which 12 separate MitoSegNet models were trained each with 11 images, excluding one image that was later used to test the prediction accuracy against other segmentation methods.

The MitoS and MitoA tool

To enable non-experts, I implemented the MitoSegNet in an easy-to-use tool, the MitoS segmentation tool, a Python-based, standalone executable. The tool can be executed in a basic mode, which utilizes the pre-trained MitoSegNet intended for segmentation of mitochondria and allows to easily apply the model without prior deep learning experience (Fig. 10A).

I applied the MitoS GPU and CPU image segmentation using the pretrained MitoSegNet on two different systems. For all cases, 10 images of each 1300 x 1030 px size (8-bit) were segmented. The MitoS GPU segmentation was run using an NVIDIA GeForce GTX 960M and NVIDIA TITAN X and segmentation took 65 and 15 seconds, respectively. Segmentation using the MitoS CPU version was performed on an Intel(R) Core(TM) i7-6700HQ CPU and a system using four Intel(R) Xeon(R) CPU E5-2680 v4 processors and lasted 7.5 minutes and 65 seconds, respectively. It is therefore strongly recommended to use MitoS GPU when possible. The basic mode furthermore includes a finetuning module, which allows to finetune the pre-trained MitoSegNet to optimise its segmentation accuracy on new images (Fig. 11A). The finetuning functionality was also tested by applying the pretrained segmentation model on images depicting *C. elegans* mitochondria using a non-integrated reporter. Due to the usage of a non-integrated $P_{myo\text{-}3}$*::mitoGFP* reporter, the intensity of fluorescently labelled mitochondria appeared weaker and hence the MitoSegNet failed to accurately segment these images (Fig. 11B). One image was therefore segmented by hand and used to train (or finetune) the pretrained model for 10 epochs, generating a finetuned model that generated satisfactory segmentation results. The finetuned model was able to more accurately segment the weaker mitochondria and the generated masks were visually better compared to the non-finetuned segmentation (Fig. 11B). The advanced mode can be used if other structures besides mitochondria should be segmented and if the user wishes to build a self-configured deep learning segmentation model (Fig. 12).

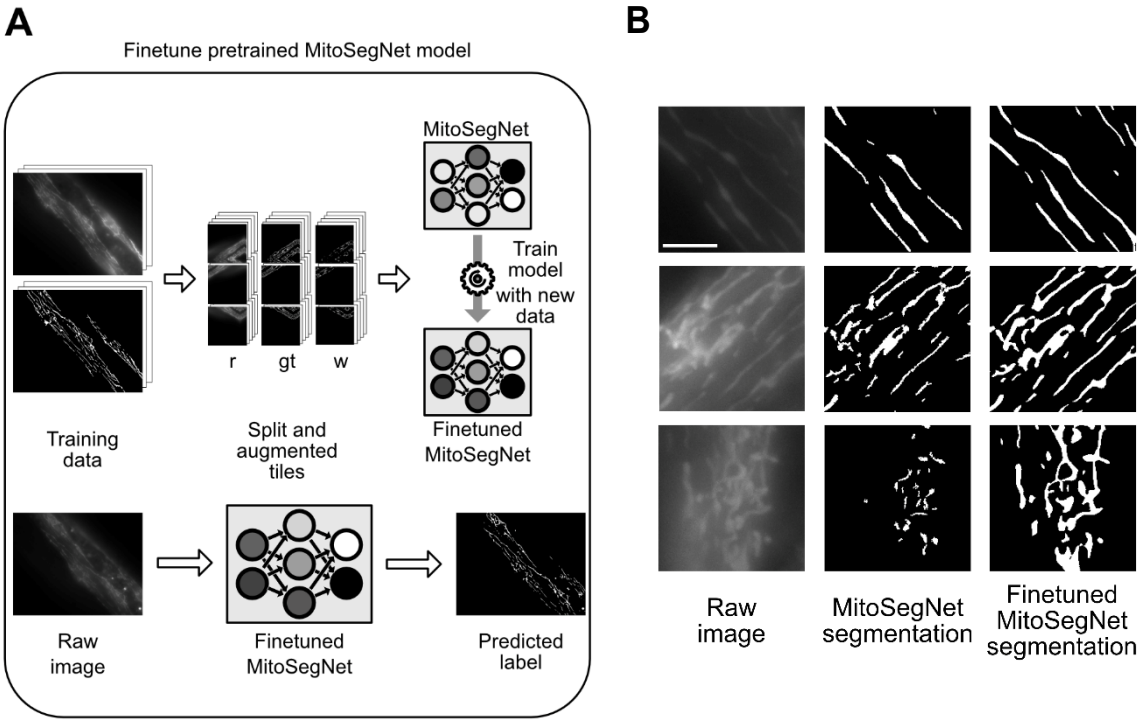


Figure 11. MitoS segmentation tool finetuning module workflow and application. (A) The finetuning module enables the user to add new training data to finetune the existing MitoSegNet model. This function automatically splits the new training data into tiles, performs augmentation (r: raw images, gt: ground truth images, w: weight map images) and the augmented data is then used to train the pretrained MitoSegNet. After completion of training, the finetuned MitoSegNet can be used for segmentation of new images using the prediction function. **(B)** Due to the usage of a non-integrated $P_{myo\text{-}3}$*::mitoGFP* reporter, the intensity of fluorescently labelled mitochondria appeared weaker and hence the MitoSegNet model failed to accurately segment these images. The input images are shown on the left, in the middle are the binary masks before finetuning and to the right the same masks after finetuning. By segmenting one image by hand and finetuning the pretrained segmentation model for 10 epochs, the visual segmentation results could be largely improved. The scale bar is 5µm.

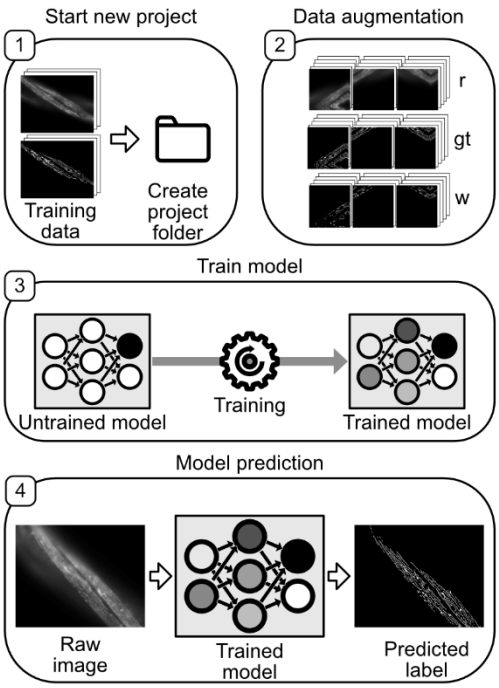


Figure 12. The advanced mode allows the user to create a new deep learning segmentation model by adjusting the network parameters. [1] Generation of a new project folder, in which all subsequent advanced functions will be carried out. [2] Data augmentation parameters can be specified and users can decide whether weight maps should be generated. (r: raw, gt: ground truth, w: weight map) [3] Model training allows the user to set the learning rate, batch size, class balance weight factor and if weight maps should be used. [4] Once the model has been trained the prediction function of the MitoS tool can be used to predict the segmentation on previously unseen images.

The MitoA analyser tool is a separate Python-based, standalone executable tool that can be run after successful segmentation for quantification and visualisation of potential morphological differences (Fig. 13). It measures ten different morphological and three intensity-based features for each object and summary statistics for all object features per image are generated. The tables of two or multiple samples containing these summary statistics can then be further subjected to hypothesis testing, visualisation and correlation analysis. The MitoS and MitoA tools require no installation and no prerequisite installations (such as frameworks) and they are available for both Windows and Linux.

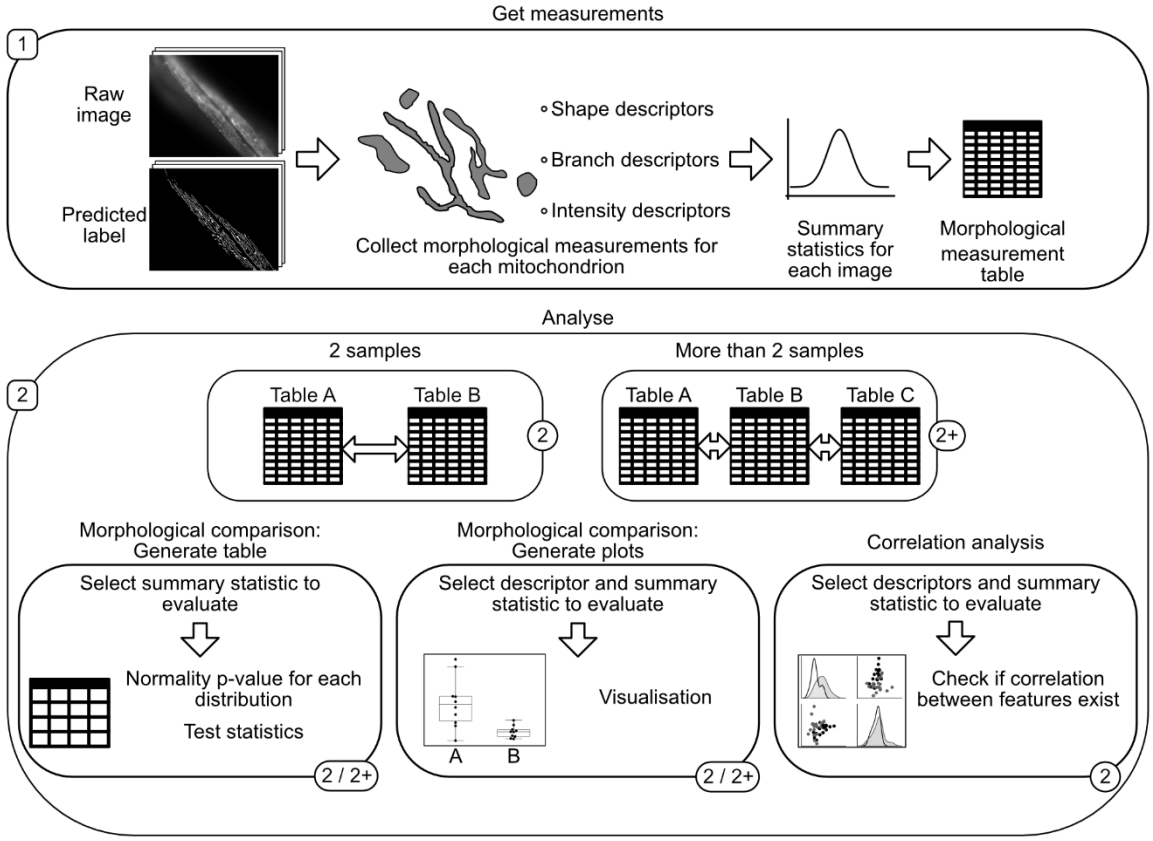


Figure 13. MitoA analyser tool workflow. The two main functions of the MitoA tool are to get measurements or to analyse. To analyse images of interest, one has to first get the relevant measurements. [1] The "Get measurements" function uses images and their predicted labels to measure 6 different shape descriptors, 4 branch descriptors and the mean, maximum and minimum fluorescence intensity for each segmented object. The summary statistics are then saved to a measurements table. [2] Analysis can either be performed on only two samples (2) or more than two samples (2+). Subsequent statistical analysis and visualisation can be used in both cases, but the correlation analysis is currently only implemented for the two-sample comparison. The generate table function allows the user to select a summary statistic to evaluate (such as average, median or standard deviation) after which a table is generated in which sample distribution normality is tested and an appropriate statistical test is selected. The generate plots function lets the user select a feature descriptor and summary statistic to display as a boxplot. With the correlation analysis, up to 4 different feature descriptors from two samples can be visualised as scatterplots, including the display of the correlation coefficients.

Visual comparison of segmentation performance

To qualitatively evaluate the performance of the MitoSegNet, I compared the predicted segmentations against manually segmented ground truth. The same procedure was repeated for four other segmentation methods to compare each segmentation visually. Three are classical feature enhancement methods (feature detection based segmentation) followed by different thresholding algorithms, all implemented in ImageJ (Gaussian, Hessian and Laplacian) and the fourth method is the machine learning segmentation tool ilastik (Kreshuk and Zhang 2019). The Gaussian, Hessian, Laplacian and ilastik methods failed to consistently prevent false positive and/or false negative segmentation on all phenotypes (Fig. 14). The Gaussian segmentation produced large sections of false positive predictions in the mixed and tubular phenotype. The Hessian and Laplacian segmentation largely avoided false positive predictions but instead often failed to recognize signal of interest, resulting in false negative segmentations in the elongated, mixed and tubular phenotype (and fragmented for the Laplacian segmentation) (Fig. 14). The ilastik-based segmentation produced only very little false negative predictions but like the Gaussian segmentation it predicted large amounts of false positive segmentation in all but the fragmented phenotype. The MitoSegNet segmentation drastically reduces the amount of false negative or false positive segmentation when compared to the other methods and yielded consistent results across all different phenotypes (Fig. 14).

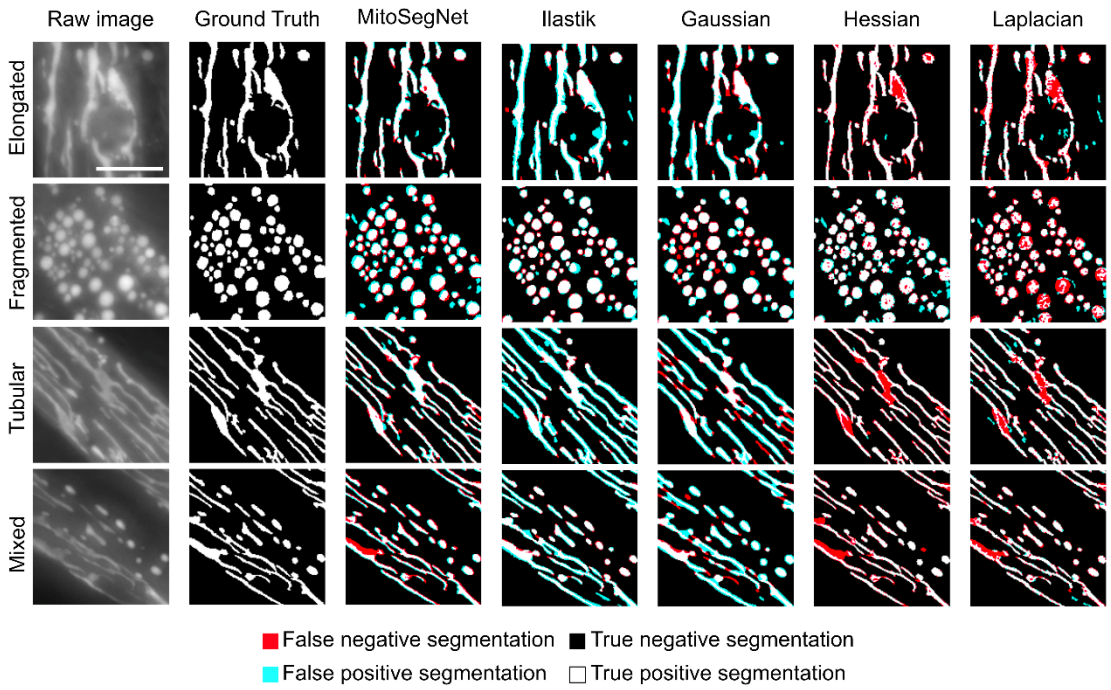


Figure 14. MitoSegNet model segmentation visually outperforms feature detection and ilastik-based segmentation approaches. On the left side four images of elongated, fragmented, tubular and mixed *C. elegans* mitochondria and their respective ground truth are shown. The masks next to each image show the results of the five different segmentation methods applied to each image, displaying the false negative segmentation in red, the false positive segmentation in turquoise, the true negative segmentation as black and the true positive segmentation as white. The scale bar is 5µm.

Quantitative comparison of segmentation performance

To compare the segmentation performance more quantitatively, I evaluated the pixelwise segmentation accuracy of the prediction using the dice coefficient (Taha and Hanbury 2015). The MitoSegNet segmentation significantly outperforms the feature-based and ilastik generated segmentations (Fig. 15A) with a median dice coefficient of 0.89 and a lower and upper 95% confidence interval of 0.87 and 0.91 (n = 12) ($p = 5.11 \cdot 10^{-5}$, Kruskal-Wallis test). However, pixelwise accuracy as measured by the dice coefficient does not necessarily guarantee correct prediction of morphology if the dc value is less than 1.0 (Fig. 15B). As biologists commonly use segmented images for morphological quantification, I assessed the

morphological accuracy with two other approaches. The single object shape deviation per object was measured for five different shape descriptors (area, eccentricity, aspect ratio, perimeter and solidity) and averaged over 12 images (Fig. 15C). The MitoSegNet with a median average fold deviation of 1.086 and a lower and upper 95% confidence interval of 1.067 and 1.116 (n = 60) outperforms all other methods in the accurate prediction of single object morphology ($p = 7.41 \cdot 10^{-10}$, Kruskal-Wallis test) (Fig. 15D). Because the single object shape deviation method does not consider false negative predictions, I also compared all segmented objects in the ground truth and prediction. For each image and each of the five object descriptors, the energy distance between the ground truth and predicted distributions was calculated (Fig. 15E). Due to the different value ranges among the descriptors the values were normalised prior to statistical analysis. The MitoSegNet segmentation achieves a median normalised energy distance of 0.197 with a lower and upper 95% confidence interval of 0.16 and 0.23 (n = 60) and again statistically outperforms all other non-deep-learning segmentation methods ($p = 3.32 \cdot 10^{-18}$, Kruskal-Wallis test) (Fig. 15F).

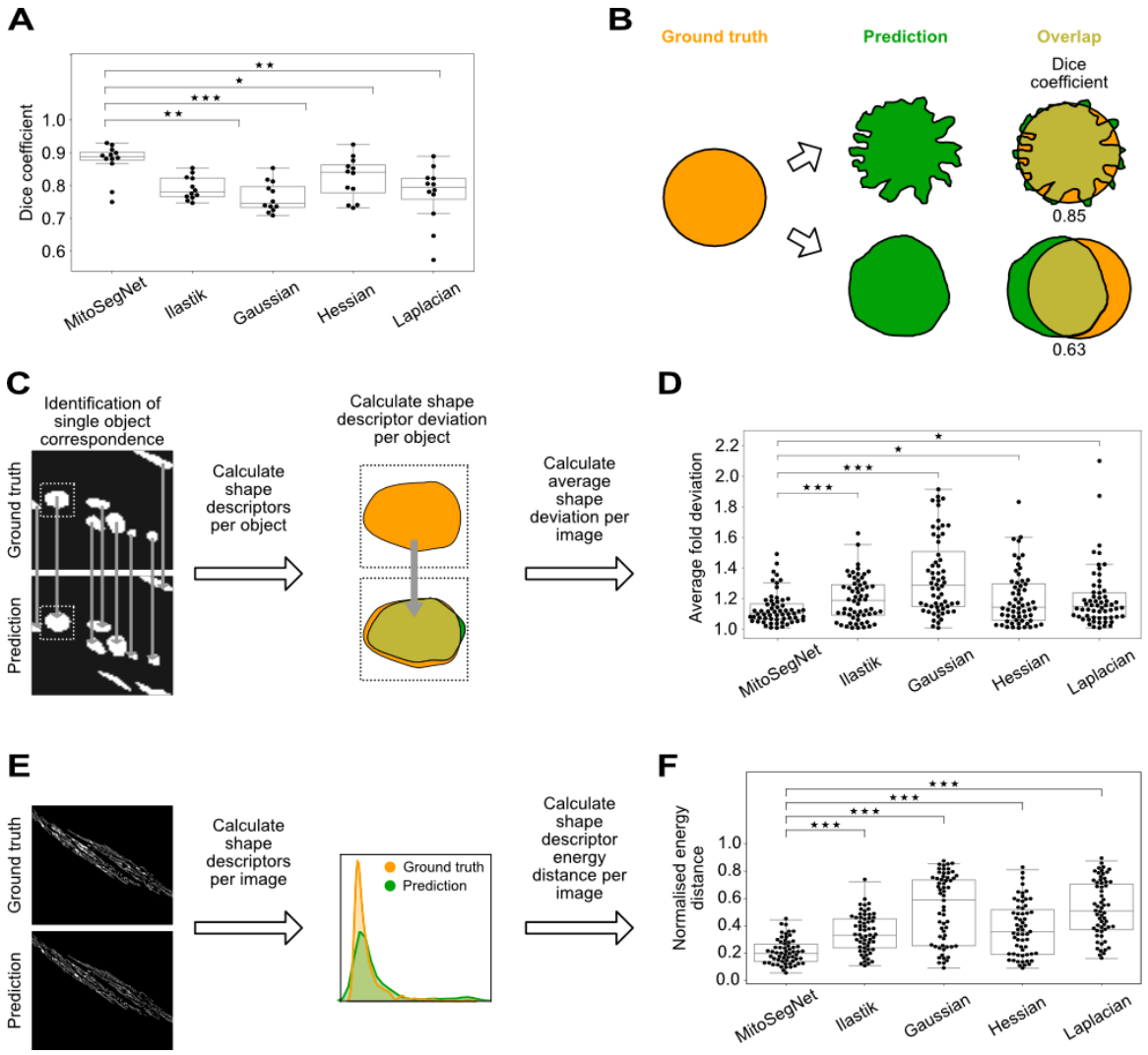


Figure 15. MitoSegNet model segmentation pixelwise accuracy outperforms non-deep learning segmentation methods and achieves the highest morphological segmentation accuracy. (A) The dice coefficient comparison reveals that the deep learning approach outperforms all non-deep-learning segmentation methods. The data was statistically evaluated by using the Kruskal-Wallis test followed by a Dunn's multiple comparisons test. **(B)** Illustrating the limitations of the dice coefficient as predictor of morphological segmentation performance. A dice coefficient close to 1.0 does not guarantee correct prediction of shape as illustrated in the example above. Contrary, a low dice coefficient does not rule out a more accurate shape prediction. Ground truth segmentation is shown in yellow, the predicted segmentation in green. Values used in this example are fictional and for representational purposes only. **(C)** To gain insight into how accurately the shape (area, eccentricity, aspect ratio, perimeter and solidity) of ground truth objects is predicted, corresponding object shape descriptors are compared by calculating the fold deviation. Predicted objects that correspond to more than one ground truth object (or vice versa) are excluded from this analysis. **(D)** The MitoSegNet model segmentation shows the lowest average fold deviation between predicted and ground truth object shape descriptors. Average fold deviation of predicted object shapes from uniquely corresponding ground truth object shapes. The data was

statistically evaluated by testing for normality using D'Agostino's K-squared test and then using the Kruskal-Wallis test followed by a Dunn's multiple comparisons test. N=60. **(E)** To determine the total morphological prediction accuracy of each segmentation method, the same shape descriptors as in C were measured. For each image, the descriptor distributions in the ground truth and predicted images were statistically evaluated for differences by calculating the energy distances between predicted and ground truth distribution. The collected energy distances for each shape descriptor and image were normalised prior to statistical analysis. **(F)** The MitoSegNet model segmentations shows the lowest median normalised energy distance compared to all other methods, statistically outperforming all other segmentation approaches. The data was first tested for normality using the D'Agostino's K-squared. After determining that all distributions were non-parametric, a Kruskal-Wallis test was used followed by a Dunn's multiple comparisons test. N = 60. *$p<0.05$, **$0.001<p<0.01$, ***$p<0.001$ for (A), (D) and (F).

Comparison of mitochondrial morphology between *catp-6* mutant and wild type

To evaluate the applicability of the MitoSegNet on a different, unseen set of images, I used the MitoS and MitoA tools to quantify morphological differences in mitochondria of wild type and mutant *C. elegans*. The *C. elegans catp-6* gene encodes a member of the family of P-type ATPases, a large family of membrane proteins that transport different compounds across membranes using ATP hydrolysis as energy source (Moller, Juul et al. 1996). The human orthologue of *catp-6* is *ATP13A2*, which if mutated leads to juvenile onset of Parkinson disease (Ramirez, Heimbach et al. 2006, Di Fonzo, Chien et al. 2007). So far, no differences between wild type and *catp-6(ok3473)* loss of function mutants in mitochondrial morphology have been reported. Indeed, upon brief visual inspection, no obvious differences in mitochondrial morphology are noticeable (Fig. 16A). To compare the morphology of mutant and wild type mitochondria, I applied the MitoSegNet to 19 fluorescence microscopy images of each phenotype and subsequently analysed the data with the MitoA tool. Segmentation masks (Fig. 16A) visually matched the raw images closely and subsequent quantification revealed a statistically significant morphological difference between wild type and mutant mitochondria.

Compared to the wild type, mitochondria are thinner and longer on average in the mutant, which was determined by comparing the average minor ($p = 0.047$, independent two-sample t-test) and major axis length ($p = 0.029$, independent two-sample t-test) (Fig. 16B). The average mitochondrial area ($p = 0.00039$, independent two-sample t-test) and perimeter ($p = 0.043$, independent two-sample t-test) are also larger in the wild type compared to the *catp-6(ok3473)* mutant (Fig. 15B). Fragmentation as cause for these observations can be excluded since the number of mitochondria in wild type and mutant images were similar ($p = 0.56$, independent two-sample t-test) (Fig. 16B). Differences were also found in mitochondrial branch morphology (Fig. 16C). While the average branch length of the mutant mitochondria was larger than in wild type ($p = 0.01$, independent two-sample t-test), the average number of mitochondrial branches was found to be significantly smaller in the mutant ($p = 0.009$, Mann-Whitney U test).

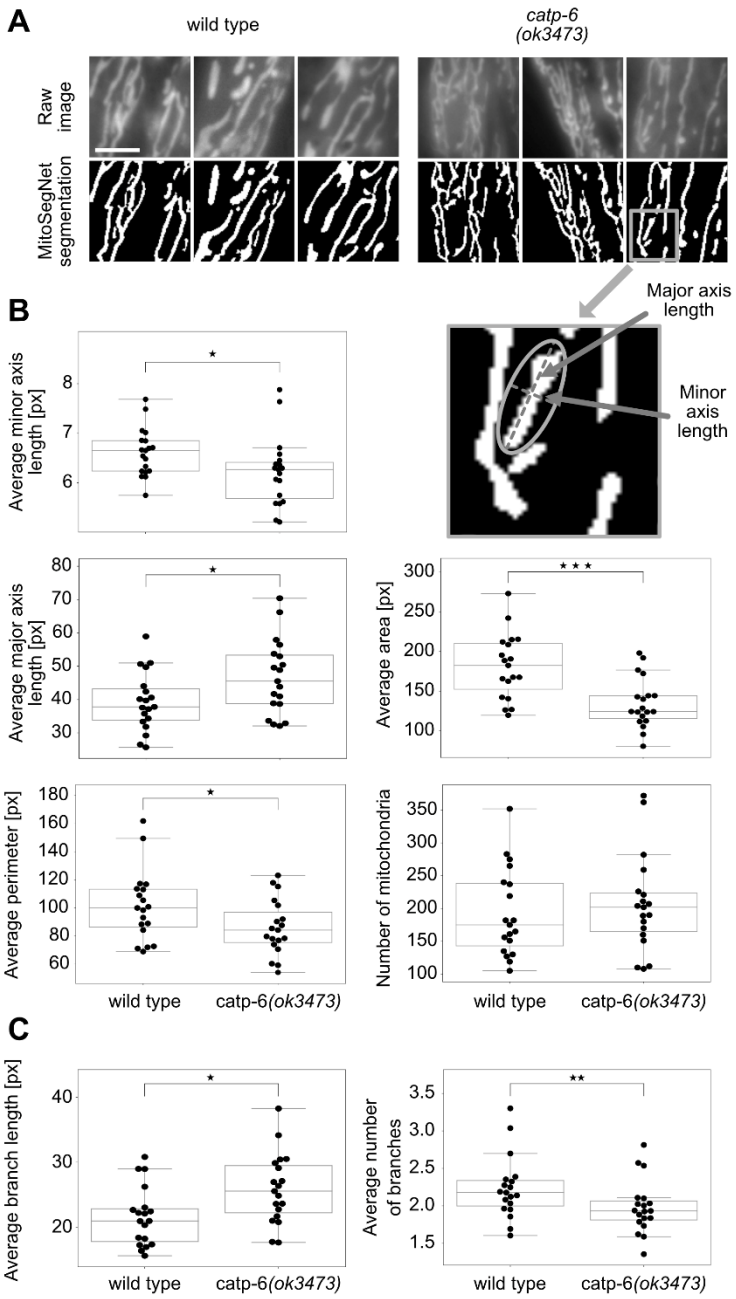


Figure 16. Wild type and *catp-6(ok3473)* mutant mitochondria show significant morphological differences. (A) Visual comparison of *catp-6(ok3473)* mutant and wild type mitochondrial morphology. Raw images are at the top and MitoSegNet model segmentations at the bottom. The scale bar is 3 µm. **(B)** Mitochondrial shape descriptor comparison. **(C)** Mitochondrial branch descriptor comparison. *p<0.05, **0.001<p<0.01, ***p<0.001 using the Mann-Whitney U test. N=19.

Application of the MitoSegNet model segmentation to mitochondria in HeLa cells

To test the generalisability of the MitoSegNet even further, I used the MitoSegNet to segment eight confocal microscopy images depicting mitochondria in HeLa cells (Fig. 17A). The fragmentation of mitochondria in HeLa cells treated with oligomycin and antimycin was captured in the segmentation both visually and quantitatively. Oligomycin and antimycin inhibit cellular respiration by disrupting oxidative phosphorylation. As expected, the average mitochondrial area is significantly larger in untreated cells compared to treated HeLa cells ($p = 0.0068$, independent two-sample t-test) (Fig. 17B). The average eccentricity is lower for the fragmented mitochondria compared to the untreated mitochondria, indicating a more circular shape ($p = 1.32 \cdot 10^{-8}$, independent two-sample t-test) (Fig. 17B). The average perimeter distribution reflects a similar pattern as found for the area, showing the fragmented mitochondria to have a smaller perimeter on average ($p = 0.00037$, independent two-sample t-test) (Fig. 17B). The average branch length is also significantly smaller in the treated mitochondria compared to the untreated mitochondria ($p = 1.30 \cdot 10^{-5}$, independent two-sample t-test) (Fig. 17B).

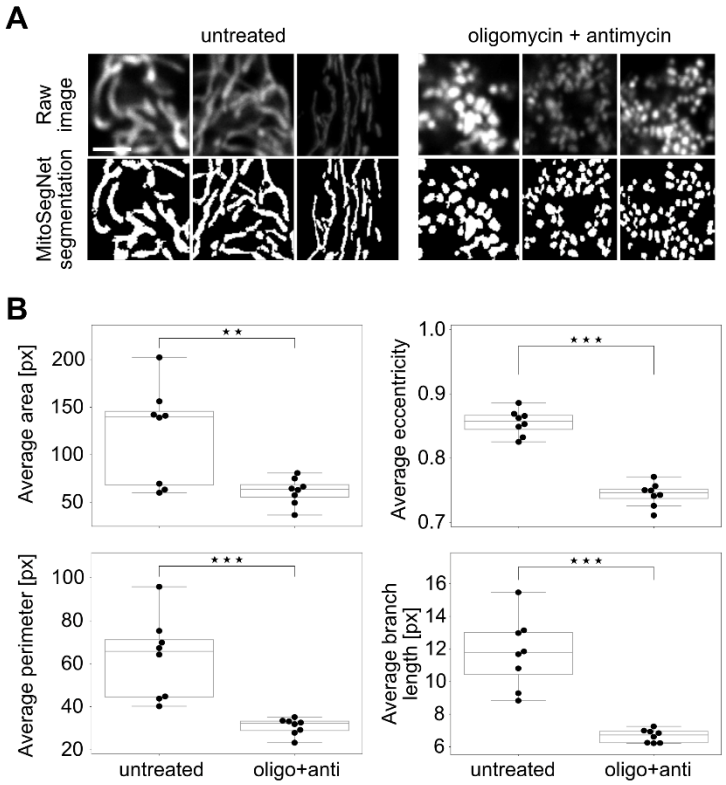


Figure 17. MitoSegNet model segmentation and morphological quantification can be applied to mitochondria of mammalian cells. Comparing untreated HeLa cells and HeLa cells treated with oligomycin or antimycin. **(A)** Visual comparison of untreated and treated mitochondrial morphology. Raw images are at the top and MitoSegNet model segmentations at the bottom. The scale bar is 2.5 µm. **(B)** Average area, eccentricity, perimeter and branch length of mitochondria were measured in segmented images of treated and untreated mitochondria. **0.001<p<0.01, ***p<0.001 using an independent two-sample t-test. N=8.

Side project 1: Compromised Mitochondrial Protein Import Acts as a Signal for UPRmt

To test if genes that induce the mitochondrial unfolded protein response (UPR^{mt}) are also necessary for maintaining the mitochondrial membrane potential, L2/3 larvae were stained with the mitochondrial potential-sensitive dye TMRE. The mitochondrial unfolded protein response (UPR^{mt}) is a conserved transcriptional response to mitochondrial dysfunction and regulated by mitochondrial-to-nuclear communication (Zhao, Wang et al. 2002). Decline in mitochondrial function activates the UPR^{mt} to promote mitochondrial repair and recovery and to maintain

cellular function (Lin and Haynes 2016). To measure the fluorescence intensity of the TMRE labelled mitochondria, I generated a custom-built image segmentation workflow, using the ImageJ IJ1 macro language. All images were subjected to background subtraction, followed by the application of the Tubeness plugin. The resulting 32-bit images were converted to 8-bit and the binary masks were generated using the IsoData autothresholding method. The binary images were used as masks to measure the mean fluorescence intensity of TMRE labelled mitochondria. A knockdown of UPR^{mt}-inducing genes was found to result in a reduced TMRE fluorescence intensity and thus a lower mitochondrial membrane potential (Fig. 18B, D). No such observation was made for loss-of-function mutations in genes that do not induce UPR^{mt} (*pdr-1(lg103)*, *pink-1(tm1779)* or *mcu-1(ju1154)*) (Fig. 18A, C) (Rolland, Schneid et al. 2019).

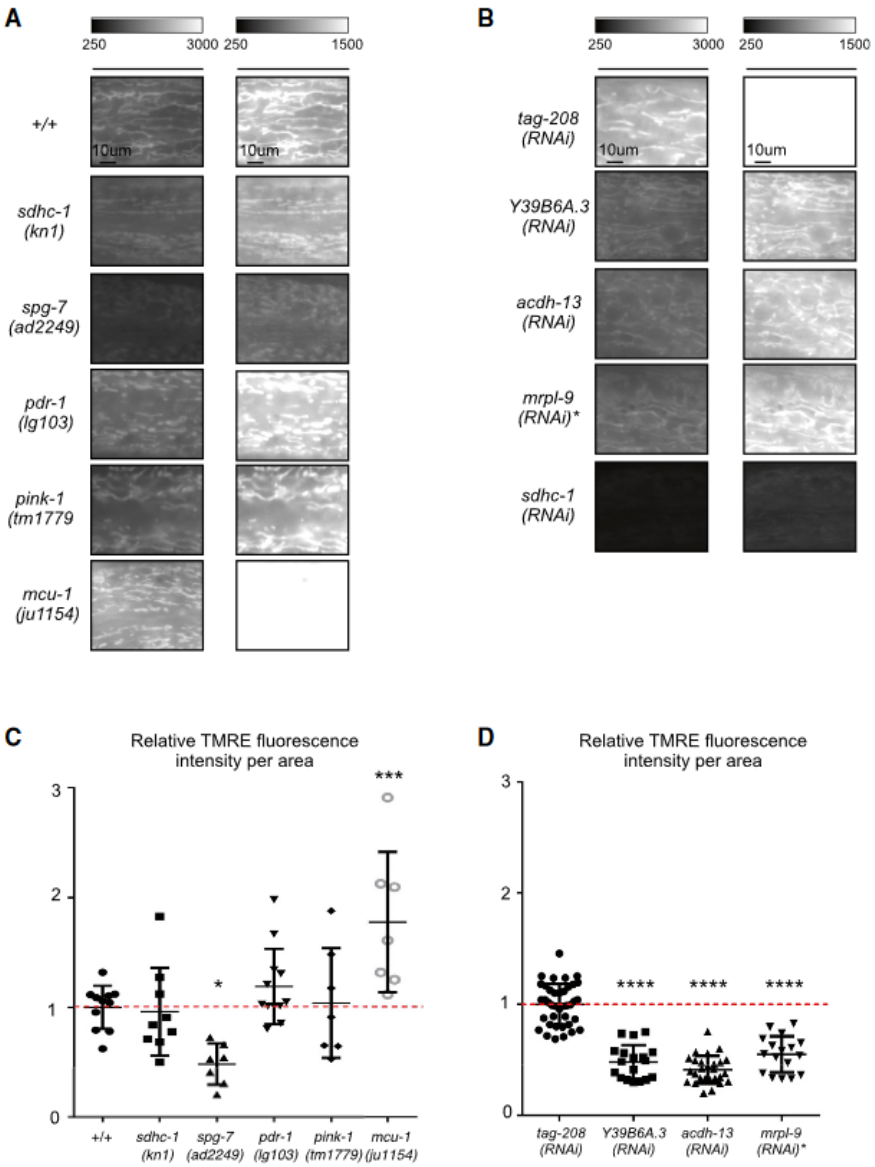


Figure 18. Genes that induce UPR^{mt} when knocked down are required to maintain mitochondrial membrane potential. (A) Wild type (+/+) or animals carrying different loss-of-function mutations were stained with TMRE. Representative images of mitochondria in hypodermal cells are shown with two intensity scales. **(B)** Wild type subjected to different RNAi (*mrpl-9(RNAi)* was diluted with *tag-208(RNAi)*) were stained with TMRE. Representative images of mitochondria in hypodermal cells are shown with two intensity scales. **(C and D)** Quantification of TMRE fluorescence intensity per area (C, n ≥ 7; D, n ≥ 17; n is the number of animals analysed; mean and standard deviation are shown; *p < 0.05, ***p < 0.001, and ****p < 0.0001 by one-way ANOVA with Bonferroni's multiple-comparison test to *tag-208(RNAi)* or +/+; *sdhc-1(RNAi)* with data was not quantified, as the signal was too low to be detected by the segmentation macro; for *mcu-1(ju1154)* and *tag-208(RNAi)* the 250–1,500 intensity images are white because of saturation. Taken from (Rolland, Schneid et al. 2019).

Side project 2: Autophagy compensates for defects in mitochondrial dynamics

Larvae were stained with TMRE to test whether induction of autophagy through *ESCRT(RNAi)* in *fzo-1(tm1133)* mutant affects mitochondrial membrane potential. The Endosomal Sorting Complex Required for Transport (ESCRT) plays an important role in endocytosis and is composed of five different subcomplexes. In *C. elegans*, the depletion of ESCRT components results in the induction of autophagy, which describes the cellular removal of dysfunctional or unnecessary components (Djeddi, Michelet et al. 2012). I developed an ImageJ IJ1 macro to automate the measurement of TMRE intensity to elucidate the effect of autophagy on mitochondrial membrane potential. All images underwent removal of continuous background signal, followed by the application of two filters to remove unwanted noise. The Tubeness plugin was applied and the binary image was created through IsoData-based thresholding. The binary images were used as masks to measure the mean fluorescence intensity of TMRE labelled mitochondria. It was found that TMRE intensity was reduced by 63% in *fzo-1(tm1133)* mutants compared to wild type (Fig. 19A). Interestingly, upon induction of autophagy, the TMRE fluorescence intensity increased in *fzo-1(tm1133)* mutants compared to the control group (Fig. 19B, C). In contrast, autophagy induction in wild type leads to a decrease in TMRE fluorescence intensity compared to the *control(RNAi)* (Fig. 19D, F). The mitochondrial TMRE fluorescence intensity is proportional to the mitochondrial membrane potential (Loew, Tuft et al. 1993). Therefore, induction of autophagy results in an increase in mitochondrial membrane potential in *fzo-1(tm1133)* mutants (Haeussler, Kohler et al. 2020).

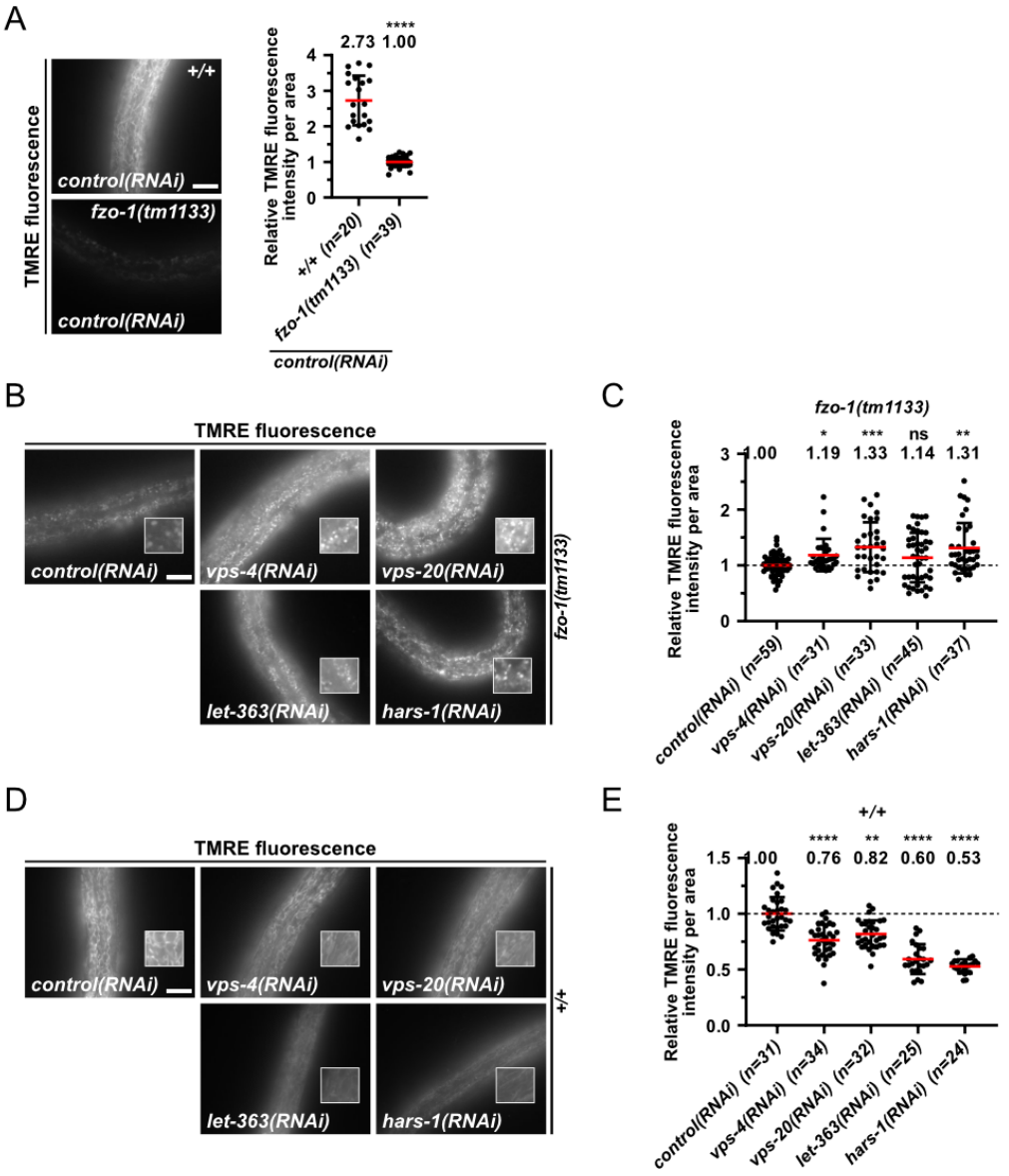


Figure 19. Induction of autophagy increases mitochondrial membrane potential and suppresses fzo-1(tm1133)-induced UPR^{mt}. (A) Fluorescence images and quantifications of L4 larvae stained with TMRE in wild type (+/+) or *fzo-1(tm1133)*. L4 larvae were subjected to *control(RNAi)* and the F1 generation was stained with TMRE overnight and imaged. Scale bar: 7.5 µm. Values indicate means ± standard deviation of 3 independent experiments in duplicates. ****p<0.0001 using unpaired two-tailed t-test with Welch's correction. **(B)** Fluorescence images of L4 larvae stained with TMRE in *fzo-1(tm1133)*. L4 larvae were subjected to *control(RNAi)*, *vps-4(RNAi)*, *vps-20(RNAi)*, *let-363(RNAi)* or *hars-1(RNAi)* and the F1 generation was stained with TMRE overnight and imaged. Scale bar: 10 µm. **(C)** Quantifications of fluorescence images from panel B. The values were normalized to *fzo-1(tm1133)* on *control(RNAi)* and each dot represents the quantification of fluorescence intensity per area from one L4 larvae. Values indicate means ± standard deviation of 3 independent experiments in duplicates. ns: not significant, *p<0.05, **p<0.01 using Kruskal-Wallis test with Dunn's multiple comparison test to *control(RNAi)*. **(D)** Fluorescence images of L4 larvae stained with TMRE in wild type. L4 larvae

were subjected to *control(RNAi)*, *vps-4(RNAi)*, *vps-20(RNAi)*, *let-363(RNAi)* or *hars-1(RNAi)* and the F1 generation was stained with TMRE overnight and imaged. Scale bar: 10 μm. **(E)** Quantifications of fluorescence images from panel D. The values were normalized to wild type on *control(RNAi)* and each dot represents the quantification of fluorescence intensity per area from one L4 larvae. Values indicate means ± standard deviation of 3 independent experiments in duplicates. ns: not significant, $**p<0.01$, ****p<0.0001 using Kruskal-Wallis test with Dunn's multiple comparison test to *control(RNAi)*. Modified from (Haeussler, Kohler et al. 2020).

Side project 3: ATP13A2-mediated endo-lysosomal polyamine export provides a mitochondrial antioxidant response

To elucidate the antioxidant function of ATP13A2 and the subsequent stress response, the *C. elegans catp-6(ok3473)* mutant strain was treated with MitoSOX Red dye and TMRE to test how ATP13A2 affects mitochondrial oxidative stress and mitochondrial membrane potential. All fluorescent microscopy images were segmented using the MitoSegNet and quantified using a self-built ImageJ macro. *C. elegans catp-6(ok3473)* animals were hypersensitive to rotenone (inhibits the transfer of electrons from complex I to ubiquinone), demonstrated by an increase in lethality compared to control animals (Fig. 20A). Mitochondrial membrane potential (MMP) is reduced in *catp-6(ok3473)* mutants compared to wild type and upon rotenone addition, the MMP in mutants decreases even further (Fig. 20B). The addition of MitoTempo (mitochondria-targeted antioxidant) partially restores MMP in *catp-6(ok3473)* mutants (Fig. 20C). Furthermore, mitoROS levels were continuously increased in *catp-6(ok3473)* mutants compared to controls, which was further enhanced by rotenone addition and decreased after application of MitoTempo (Fig. 20D). The MitoTempo induced decrease in mitoROS also reduced the lethality in response to rotenone (Fig. 20E) (Vrijsen, Besora-Casals et al. unpublished).

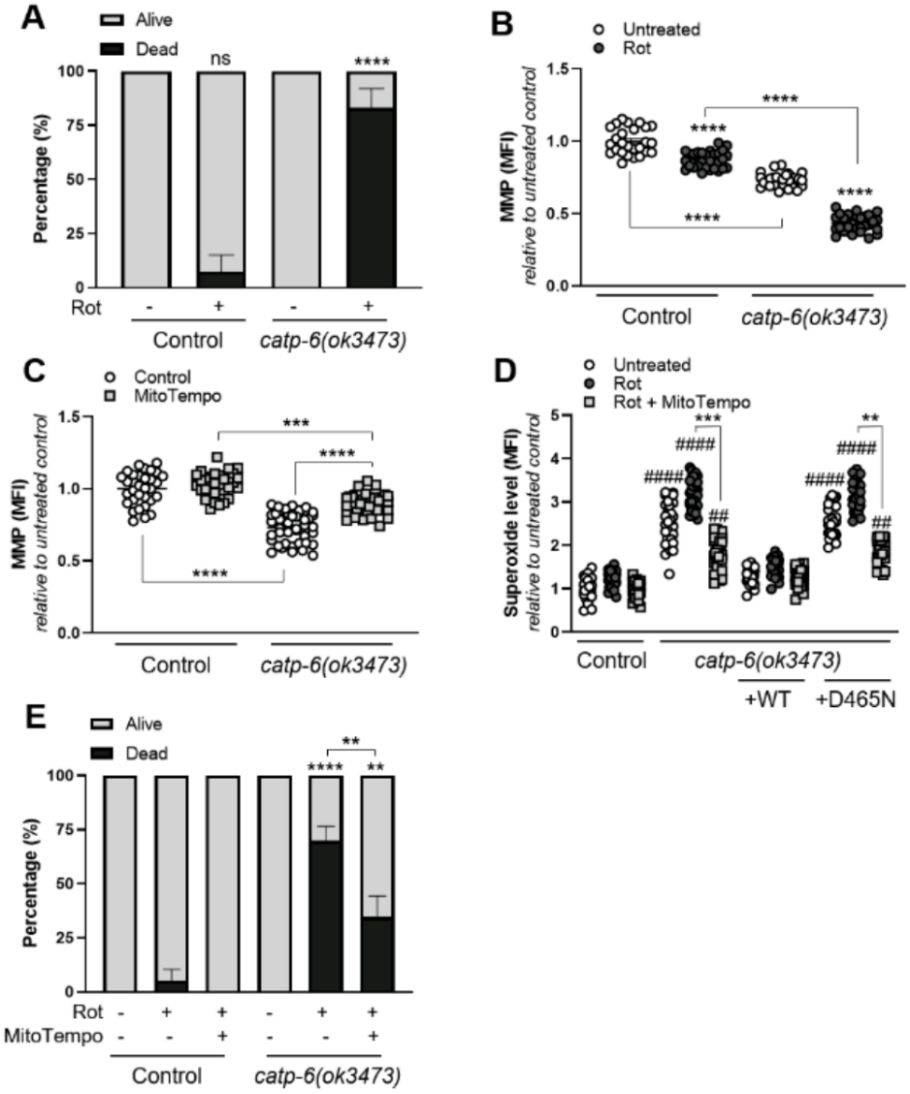


Figure 20. The ATP13A2 orthologue *catp-6* exerts a mitochondrial protective antioxidant function in vivo in *C. elegans*, thereby preventing the activation of a stress response. Wild type (control) *C. elegans* and strains carrying a loss of function mutation *(ok3473)* in the P5B-orthologue *catp-6*, either rescued or not by overexpression of wild type *catp-6* (WT) or a catalytically inactive mutant (D465N), were exposed to rotenone (Rot, 10 μM) (A, B, D, E) or analysed under basal conditions (C) in absence (A, B) or presence (C, D, E) of MitoTempo (10 mM). Subsequently, we measured (A, E) lethality (60 h and 72 h Rot exposure, respectively), (B, C) mitochondrial membrane potential (MMP) (16 h Rot exposure), (D) superoxide levels (16 h Rot exposure). Data are the mean of a minimum of three independent experiments ± SEM. MFI, Mean Fluorescence Intensity. ** $p<0.01$; *** $p<0.001$; **** $p<0.0001$ versus respective untreated unless otherwise indicated; ## $p<0.01$; #### $p<0.0001$ versus rotenone-treated control; ANOVA or Kruskal-Wallis with post-hoc Tukey's (A, B, E) or Dunn's (C, D) multiple comparison test, respectively. Modified from (Vrijsen, Besora-Casals et al. unpublished).

Discussion

MitoSegNet segmentation performance

While the superior visual and quantitative performance of the MitoSegNet model segmentation might not come as a surprise to researchers acquainted with the capabilities of deep learning-based segmentation, I believe these results to be interesting to those researchers who still use classical image segmentation or more sophisticated methods such as implemented in the machine learning tool ilastik. Unlike most other accuracy estimates of deep learning segmentation models, I did not rely purely on pixel-based accuracy as I found this to be an insufficient measure of morphological accuracy, which is what most biologists are more interested in when segmenting microscopy images. Our single object shape comparison as well as the calculation of energy distances for 10 different feature descriptors per image demonstrate that the MitoSegNet model segmentation yields the best morphological accuracy compared with the four other more commonly applied segmentation methods. An important note of caution has to be made when generating and using deep learning models, such as the MitoSegNet. The performance of the model is heavily dependent on the ground truth that is used for training, which in our case was generated by hand. Most ground truth data are created through manual annotation, which can lead to error-prone and biased deep learning models. Due to the limited availability of experienced annotators, 12 ground truth images were created by only two annotators. Ideally, the higher the number of annotators is, the more label variance is introduced and the less biased the deep learning model should become. Interestingly, a recent study found deep learning models to be robust against erroneous ground truth if the training data size is large enough (Rolnick, Veit et al. 2017). However, the results should be taken with cautions due to the limited noise types that were analysed for this study. To further improve

the segmentation performance and robustness of the MitoSegNet, more training data from different annotators should be added.

MitoS and MitoA tools

Most deep learning applications in the field of biological image segmentation were created for the purpose of 2D cell segmentation (Chen, Dai et al. 2017, Al-Kofahi, Zaltsman et al. 2018, Falk, Mai et al. 2019, Kusumoto and Yuasa 2019), while organelle-specific deep learning applications are scarce. Although most tools allow the user to retrain available 2D cell segmentation models to segment other biological structures of interest, this often requires computer science related skills, such as familiarity with programming languages, shell interaction or knowledge on how to install various deep learning frameworks. In 2018, the U-Net was made accessible through a downloadable plugin in ImageJ (Falk, Mai et al. 2019). The plugin requires an additional installation of the deep learning framework caffe and there is little online documentation available on how to generate a customised U-Net model with this plugin. Furthermore, the U-Net plugin does not allow to configure the class balance factor, which is pre-set at a value of 0.1. Since the ratio of foreground to background pixel can vary extensively, a pre-set class balance factor might not be sufficient to mitigate class imbalance effects. One of the main motivations behind the MitoSegNet and the MitoS tool was to make deep learning segmentation accessible to researchers that do not have an extensive background in computer science or deep learning. The MitoS tool can be run without installation and does not require the installation of any additional drivers, frameworks or software libraries. Extensive documentation is available on GitHub and MitoS automatically calculates the optimal class balance factor based on the training data. The simple graphical user interface allows users to quickly navigate the MitoS and MitoA tools. The MitoS basic mode also comes

with a finetuning module that allows researchers that would like to segment other organelles or images taken under different conditions than those used for training the MitoSegNet model. The past assumption in training deep learning models was that training data should be in the same feature space and have the same value distribution but for many practical applications this has been found to be incorrect (Weiss, Khoshgoftaar et al. 2016). Instead, training pre-trained models, also known as transfer learning, on different training data has been shown to improve the model's generalisation capabilities (A Pan and A Yang 2010, Kaya, Keceli et al. 2019). Since the subsequent step after segmentation is usually the analysis, I included the MitoA tool to save researchers the time to look up appropriate analysis tools and instead be able to quickly obtain potentially interesting insights.

Comparison of mitochondrial morphology between *catp-6* mutant and wild type

There is evidence pointing toward a possible link between a deficiency of ATP13A2 (of which *catp-6* is the *C. elegans* orthologue) and mitochondrial dysfunction (Ramirez, Heimbach et al. 2006, Vives-Bauza and Przedborski 2011). While there seems to be a connection between mitochondrial function and morphology, the underlying molecular mechanisms are still not well understood (Ferree and Shirihai 2012). The visual comparison of mitochondria in wild type and *catp-6(ok3473)* mutants did not permit any conclusions on differences in morphology as in both wild type and mutant, the mitochondria appeared to be largely tubular. However, the quantitative analysis revealed that average mitochondrial area, perimeter and minor axis length of the mutant are smaller than those in the wild type. Based on this analysis, it can be concluded that mitochondria in *catp-6(ok3473)* mutants are longer, thinner and have fewer branches than wild type mitochondria. These quantitative findings could not have been obtained by visual analysis, thus highlighting the sensitivity of our deep learning segmentation approach. No link between *catp-6(ok3473)* loss of function and change in mitochondrial morphology in *C.*

elegans has thus far been reported. Vrijsen et al. (unpublished) demonstrated a decrease in mitochondrial membrane potential and increased levels of reactive oxygen species in *catp6(ok3473)* mutant mitochondria. These results could suggest a link between changes in mitochondrial morphology and the reduction of mitochondrial membrane potential and increased levels of reactive oxygen species in *catp6(ok3473)* mutants.

Application of the MitoSegNet segmentation to mitochondria in HeLa cells

Although the pretrained model was generated with standard fluorescence microscopy images, depicting mitoGFP labelled mitochondria in *C. elegans*, the same pretrained model was able to generate visually accurate segmentations of mitoRFP labelled mitochondria in HeLa cells using a laser scanning confocal microscope. This demonstrates the high robustness and generalisation capabilities of the pretrained MitoSegNet. It also shows that it can be used for segmentation of mitochondria in organisms other than *C. elegans*. Furthermore, our MitoSegNet Analysis tool quantitatively confirmed the morphological differences of mitochondria between untreated HeLa cells and HeLa cells treated with oligomycin or antimycin.

MitoSegNet model

The MitoSegNet architecture is largely based on the U-Net but through testing various changes in the original architecture, I found that the validation dice coefficient as well as the validation loss improved upon removing the dropout layers. Instead, a batch normalisation layer (Ioffe and Szegedy 2015) was placed after every convolution layer in the contracting pathway. Interestingly, a recent study found that the combined usage of batch normalisation followed by dropout (forming an independent component layer) stabilised the training process, increased convergence speed and improved the convergence limit (Guangyong Chen, Pengfei Chen et al.

2019). Further testing would be required to find out if the usage of an independent component layer would improve the current MitoSegNet training performance. Another recent development in the field of deep learning is the usage of leaky ReLU as activation functions (Maas, Hannun et al. 2013). They avoid the "dying ReLU" problem that occurs when the values are always below 0, which causes a gradient of 0. Leaky ReLU's allow a small gradient when the unit is not active. The MitoSegNet was established using ReLU activation functions and testing the current MitoSegNet training and validation performance against a MitoSegNet trained with leaky ReLU might reveal the potential to further improve the segmentation performance. It is worth noting that the current architecture, augmentation methods and hyperparameters were chosen based on the 12 training images used to create the MitoSegNet. With an increasing amount of training data, the MitoSegNet configuration might have to be changed to further improve its segmentation accuracy.

Side projects

All three side projects were focused on the measurement of mitochondrial membrane potential and / or mitochondrial oxidative stress levels in *C. elegans*. These measurements require the precise segmentation of mitochondria to accurately quantify both parameters. Unfortunately, TMRE quantifications for Rolland et al. (2019) and Haeussler et al. (2020) were required prior to the completion of the MitoSegNet, which is why a Hessian-based segmentation workflow was used. Although the Hessian segmentation was shown to perform poorly on measuring morphology on fragmented or elongated mitochondria, membrane potential quantifications require no accurate morphological measurements, which is why the Hessian segmentation can still be seen as a viable method. While both feature based segmentation methods served only as a part of two wider studies, they both provided vital results for the conclusions of the two studies. Based on the results from TMRE measurements of different loss-of-function

mutations, Rolland et al. (2019) concluded that there is a correlation between reduced mitochondrial membrane potential and the UPR^{mt} induction. The TMRE measurements allowed Haeussler et al. (2020) to suggest that the rescue of the decreased mitochondrial membrane potential suppresses *fzo-1(tm1133)*-induced UPR^{mt} after ESCRT depletion and not the fragmented mitochondrial phenotype. After its completion, the MitoSegNet was successfully applied to segment mitochondria and measure TMRE and mitoSOX intensities. Vrijsen et al. (unpublished) showed that ATP13A2 provides protection against mitochondrial toxins such as rotenone in SH-SY5Y cells (human neuroblastoma cell line). The MitoSegNet segmented images of *catp6(ok3473)* mutants stained with TMRE and mitoSOX enabled Vrijsen et al. (unpublished) to conclude that the ATP13A2 mediated antioxidant effect seems to be highly conserved and plays a vital role.

Conclusion

For the conclusion, I would like to briefly recapitulate the historical developments during the course of this PhD that led to the establishment of the MitoSegNet and the computational tools to use it. The original aim of this PhD thesis began with the task of finding an efficient and automated way to segment mitochondria in *C. elegans* and subsequently perform morphological and intensity-based measurements. At first, the feature detection methods proved to be the most reliable, in particular the Laplacian or Hessian method. However, comparing the generated segmentations with ground truth images revealed the inaccuracies of the feature detection methods, especially for morphologically fragmented, mixed and elongated mitochondria. This was the point at which I turned to deep learning and eventually found the U-Net. The model architecture and hyperparameters were configured to meet our needs and the resulting model was named MitoSegNet. This method outperformed all feature detection methods in both pixelwise and morphological segmentation accuracy. To further test the MitoSegNet, I also compared its output with that of the machine learning tool ilastik and the MitoSegNet proved again to be superior. At this point, the workflow was established and I began to apply the MitoSegNet to unseen images of mitochondria in *C. elegans* and HeLa cells to test its robustness. In most but not all cases the MitoSegNet delivered visually convincing segmentation results. For those few cases in which segmentation failed, a finetuning functionality was established, which allowed the pretrained MitoSegNet to undergo further training on new images to improve its segmentation accuracy. To make this workflow broadly applicable, the decision was made to develop a computational tool (MitoS) with a graphical user interface to enable other mitochondria researchers with no deep learning background to make use of the MitoSegNet. The MitoA analysis tool was added subsequently to enable a fast transition from raw images to quantitative data. The MitoSegNet model, the MitoS and MitoA

tools should not be viewed as static entities as they can be further developed. The code for the MitoS and MitoA tools is publicly available on GitHub (see page 85) and can be readily modified by anybody. The field of deep learning is advancing fast and existing FCNN model architectures are being extended or new architectures developed every year (Hesamian, Jia et al. 2019, Ibtehaz and Rahman 2020). To my current knowledge, the U-Net is still one of the most prominent and successful FCNN architectures in the field of biological image segmentation. However, it is only a matter of time until this architecture will be replaced by a superior method. It is also worth noting that the power of a deep learning model lies not only in its underlying algorithm but especially in the data used to train it. The MitoSegNet was trained with just 12 images, which in the world of deep learning is considered a very small data size. The accuracy and robustness of the MitoSegNet is not fixed and can be increased by adding more training data. Furthermore, the MitoS and MitoA tools will remain maintained for the foreseeable future and will be upgraded accordingly if deemed necessary. In addition to the development of the MitoSegNet and the computational tools to use it, I also worked on three other projects, which all involved the segmentation of mitochondria and subsequent quantification of different mitochondrial parameters in *C. elegans*. The two feature detection method workflows used to quantify TMRE intensity in Rolland et al. (2019) and Haeussler et al. (2020) remain reliable methods despite the existence of the MitoSegNet but would not be recommended to use when quantifying mitochondrial morphology. In this case, the MitoSegNet would provide far more accurate results as demonstrated in this thesis. After its completion, the MitoSegNet was successfully applied to segment unseen images of TMRE and mitoSOX labelled mitochondria in *C. elegans* by Vrijsen et al. (unpublished). In conclusion, this thesis demonstrated the broad applicability and high accuracy of the MitoSegNet compared against four other commonly used segmentation approaches. Together with the MitoS and Mito

A tools, the MitoSegNet offers researchers a complete AI-based computational toolkit to accurately segment and analyse mitochondria in *C. elegans* and other model organisms.

Software availability

The software documentation for the MitoS and MitoA tool can be found under https://github.com/mitosegnet. The MitoSegNet segmentation model, the MitoA analysis and MitoS segmentation tool (GPU / CPU) for Linux and Windows are available under https://zenodo.org/search?page=1&size=20&q=mitosegnet.

References

(1998). "Genome sequence of the nematode C. elegans: a platform for investigating biology." Science 282(5396): 2012-2018.

A Pan, S. J. and Q. A Yang (2010). "A Survey on Transfer Learning." IEEE Transactions on Knowledge and Data Engineering 22: 1345-1359.

Ajioka, R. S., J. D. Phillips and J. P. Kushner (2006). "Biosynthesis of heme in mammals." Biochimica et Biophysica Acta (BBA) - Molecular Cell Research 1763(7): 723-736.

Al-Kofahi, Y., A. Zaltsman, R. Graves, W. Marshall and M. Rusu (2018). "A deep learning-based algorithm for 2-D cell segmentation in microscopy images." BMC Bioinformatics 19(1): 365.

Altman, R. (1890). "Die Elementarorganismen und ihre Beziehung zu den Zellen." Viet & Comp, Leipzig.

Altun, Z. F., L. A. Herndon, C. A. Wolkow, C. Crocker, R. Lints and D. H. Hall. (2002-2020). "WormAtlas." from http://www.wormatlas.org

Amodei, D., S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, J. Chen, J. Chen, Z. Chen, M. Chrzanowski, A. Coates, G. Diamos, K. Ding, N. Du, E. Elsen, J. Engel, W. Fang, L. Fan, C. Fougner, L. Gao, C. Gong, A. Hannun, T. Han, L. Johannes, B. Jiang, C. Ju, B. Jun, P. LeGresley, L. Lin, J. Liu, Y. Liu, W. Li, X. Li, D. Ma, S. Narang, A. Ng, S. Ozair, Y. Peng, R. Prenger, S. Qian, Z. Quan, J. Raiman, V. Rao, S. Satheesh, D. Seetapun, S. Sengupta, K. Srinet, A. Sriram, H. Tang, L. Tang, C. Wang, J. Wang, K. Wang, Y. Wang, Z. Wang, Z. Wang, S. Wu, L. Wei, B. Xiao, W. Xie, Y. Xie, D. Yogatama, B. Yuan, J. Zhan and Z. Zhu (2016). Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin. Proceedings of The 33rd International Conference on Machine Learning. B. Maria Florina and Q. W. Kilian. Proceedings of Machine Learning Research, PMLR. 48: 173--182.

Baum, S. (2017). "A Survey of Artificial General Intelligence Projects for Ethics, Risk, and Policy." Global Catastrophic Risk Institute Working Paper

Bebis, G. and M. Georgiopoulos (1994). "Feed-forward neural networks." IEEE Potentials 13(4): 27-31.

Bereiter-Hahn, J. (1990). "Behavior of mitochondria in the living cell." Int Rev Cytol 122: 1-63.

Bertsekas, D. P. (1999). Nonlinear programming.

Bishop, C. M. (1995). Neural networks for pattern recognition. Oxford; New York, Clarendon Press ; Oxford University Press.

Bottou, L. and O. Bousquet (2007). The tradeoffs of large scale learning. Proceedings of the 20th International Conference on Neural Information Processing Systems. Vancouver, British Columbia, Canada, Curran Associates Inc.: 161-168.

Brenner, S. (1974). "The genetics of Caenorhabditis elegans." Genetics 77(1): 71-94.

Chen, M., W. Dai, S. Y. Sun, D. Jonasch, C. Y. He, M. F. Schmid, W. Chiu and S. J. Ludtke (2017). "Convolutional neural networks for automated annotation of cellular cryo-electron tomograms." Nat Methods 14(10): 983-985.

Cireşan, D., U. Meier, J. Masci and J. Schmidhuber (2012). "Multi-column deep neural network for traffic sign classification." Neural Networks 32: 333-338.

Conradt, B. and H. R. Horvitz (1998). "The C. elegans protein EGL-1 is required for programmed cell death and interacts with the Bcl-2-like protein CED-9." Cell 93(4): 519-529.

Conradt, B. and D. Xue (2005). Programmed cell death. WormBook. T. C. e. R. Community, WormBook.

Copeland, B. J. and O. Shagrir (2018). "The Church-Turing thesis: logical limit or breachable barrier?" Commun. ACM 62(1): 66-74.

Culetto, E. and D. B. Sattelle (2000). "A role for Caenorhabditis elegans in understanding the function and interactions of human disease genes." Hum Mol Genet 9(6): 869-877.

D'Agostino, R. B. (1971). "An omnibus test of normality for moderate and large sample size." Biometrika 58: 341-348.

D'Agostino, R. B. and E. S. Pearson (1973). "Tests for departure from normality." Biometrika 60: 613-622.

D. Marr, E. H. (1980). "Theory of edge detection." Proceedings of the Royal Society of London. Series B. Biological Sciences 207: 187-217.

de Boer, R., R. L. Smith, W. H. De Vos, E. M. Manders, S. Brul and H. van der Spek (2015). "Caenorhabditis elegans as a Model System for Studying Drug Induced Mitochondrial Toxicity." PLoS One 10(5): e0126220.

De Vos, W. H., L. Van Neste, B. Dieriks, G. H. Joss and P. Van Oostveldt (2010). "High content image cytometry in the context of subnuclear organization." Cytometry Part A 77A(1): 64-75.

del Peso, L., V. M. Gonzalez and G. Nunez (1998). "Caenorhabditis elegans EGL-1 disrupts the interaction of CED-9 with CED-4 and promotes CED-3 activation." J Biol Chem 273(50): 33495-33500.

Di Fonzo, A., H. F. Chien, M. Socal, S. Giraudo, C. Tassorelli, G. Iliceto, G. Fabbrini, R. Marconi, E. Fincati, G. Abbruzzese, P. Marini, F. Squitieri, M. W. Horstink, P. Montagna, A. D. Libera, F. Stocchi, S. Goldwurm, J. J. Ferreira, G. Meco, E. Martignoni, L. Lopiano, L. B.

Jardim, B. A. Oostra, E. R. Barbosa and V. Bonifati (2007). "ATP13A2 missense mutations in juvenile parkinsonism and young onset Parkinson disease." Neurology 68(19): 1557-1562.

Dice, L. R. (1945). "Measures of the Amount of Ecologic Association Between Species." Ecology 26(3): 297-302.

Diederik P. Kingma, J. B. (2014). "Adam: A Method for Stochastic Optimization." CoRR abs/1412.6980.

Diez-Fernandez, C. and J. Häberle (2017). "Targeting CPS1 in the treatment of Carbamoyl phosphate synthetase 1 (CPS1) deficiency, a urea cycle disorder." Expert Opinion on Therapeutic Targets 21(4): 391-399.

Djeddi, A., X. Michelet, E. Culetto, A. Alberti, N. Barois and R. Legouis (2012). "Induction of autophagy in ESCRT mutants is an adaptive response for cell survival in C. elegans." J Cell Sci 125(Pt 3): 685-694.

Dumoulin, V. and F. Visin (2016). A guide to convolution arithmetic for deep learning.

Ernster, L. and G. Schatz (1981). "Mitochondria: a historical review." J Cell Biol 91(3 Pt 2): 227s-255s.

Falk, T., D. Mai, R. Bensch, O. Cicek, A. Abdulkadir, Y. Marrakchi, A. Bohm, J. Deubner, Z. Jackel, K. Seiwald, A. Dovzhenko, O. Tietz, C. Dal Bosco, S. Walsh, D. Saltukoglu, T. L. Tay, M. Prinz, K. Palme, M. Simons, I. Diester, T. Brox and O. Ronneberger (2019). "U-Net: deep learning for cell counting, detection, and morphometry." Nat Methods 16(1): 67-70.

Ferree, A. and O. Shirihai (2012). "Mitochondrial dynamics: the intersection of form and function." Advances in experimental medicine and biology 748: 13-40.

Fukushima, K. (1980). "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position." Biological Cybernetics 36(4): 193-202.

Geman, S., E. Bienenstock, Ren\, \#233 and Doursat (1992). "Neural networks and the bias/variance dilemma." Neural Comput. 4(1): 1-58.

Girosi, F., M. Jones and T. Poggio (1995). "Regularization theory and neural networks architectures." Neural Comput. 7(2): 219-269.

Goodfellow, I., Y. Bengio and A. Courville (2016). Deep Learning, The MIT Press.

Goodfellow, I. J., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville and Y. Bengio (2014). Generative Adversarial Nets. NIPS.

Gray, M. W., G. Burger and B. F. Lang (1999). "Mitochondrial evolution." Science 283(5407): 1476-1481.

Grefenstette, E., P. Blunsom, N. d. Freitas and K. M. Hermann (2014). "A Deep Architecture for Semantic Parsing." ArXiv abs/1404.7296.

Guangyong Chen, Pengfei Chen, Yujun Shi, Chang-Yu Hsieh, Benben Liao and and Shengyu Zhang (2019). "Rethinking the Usage of Batch Normalization and Dropout in the Training of Deep Neural Networks." CoRR abs/1905.05928.

Haeussler, S., F. Kohler, M. Witting, M. F. Premm, S. G. Rolland, C. Fischer, L. Chauve, O. Casanueva and B. Conradt (2020). "Autophagy compensates for defects in mitochondrial dynamics." PLoS Genet 16(3): e1008638.

Hastie, T., R. Tibshirani and J. H. Friedman (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer.

Haykin, S. (1998). Neural Networks: A Comprehensive Foundation, Prentice Hall PTR.

He, K., X. Zhang, S. Ren and J. Sun (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Hedgecock, E. M. and J. Nichol Thomson (1982). "A gene required for nuclear and mitochondrial attachment in the nematode caenorhabditis elegans." Cell 30(1): 321-330.

Heggeness, M. H., M. Simon and S. J. Singer (1978). "Association of mitochondria with microtubules in cultured cells." Proceedings of the National Academy of Sciences of the United States of America 75(8): 3863-3866.

Hesamian, M. H., W. Jia, X. He and P. Kennedy (2019). "Deep Learning Techniques for Medical Image Segmentation: Achievements and Challenges." Journal of Digital Imaging 32(4): 582-596.

Hinton, G. and T. J. Sejnowski (1999). Unsupervised Learning: Foundations of Neural Computation, The MIT Press.

Hochreiter, S. and J. Schmidhuber (1997). "Long Short-Term Memory." Neural Comput. 9(8): 1735-1780.

Houten, S. M. and R. J. A. Wanders (2010). "A general introduction to the biochemistry of mitochondrial fatty acid β-oxidation." Journal of inherited metabolic disease 33(5): 469-477.

Huang, L. and M. Wang (1995). "Image thresholding by minimizing the measure of fuzziness." Pattern Recognition 28(1): 41-51.

Huang, T. (1996). "Computer Vision: Evolution and Promise." 19th CERN School of Computing: 21-25.

Hubel, D. H. and T. N. Wiesel (1970). "The period of susceptibility to the physiological effects of unilateral eye closure in kittens." J Physiol 206(2): 419-436.

Ibtehaz, N. and M. S. Rahman (2020). "MultiResUNet : Rethinking the U-Net architecture for multimodal biomedical image segmentation." Neural Networks 121: 74-87.

Ioffe, S. and C. Szegedy (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift. Proceedings of the 32nd International Conference on

International Conference on Machine Learning - Volume 37. Lille, France, JMLR.org: 448-456.

Jagasia, R., P. Grote, B. Westermann and B. Conradt (2005). "DRP-1-mediated mitochondrial fragmentation during EGL-1-induced cell death in C. elegans." Nature 433(7027): 754-760.

James, G., D. Witten, T. Hastie and R. Tibshirani (2014). An Introduction to Statistical Learning: with Applications in R, Springer Publishing Company, Incorporated.

Johnson, T. E., D. H. Mitchell, S. Kline, R. Kemal and J. Foy (1984). "Arresting development arrests aging in the nematode Caenorhabditis elegans." Mech Ageing Dev 28(1): 23-40.

Kaelbling, L. P., M. L. Littman and A. W. Moore (1996). "Reinforcement Learning: A Survey." ArXiv cs.AI/9605103.

Kaletta, T. and M. O. Hengartner (2006). "Finding function in novel targets: C. elegans as a model organism." Nat Rev Drug Discov 5(5): 387-398.

Kaya, A., A. S. Keceli, C. Catal, H. Y. Yalic, H. Temucin and B. Tekinerdogan (2019). "Analysis of transfer learning for deep neural network based plant classification models." Computers and Electronics in Agriculture 158: 20-29.

Kimble, J. and D. Hirsh (1979). "The postembryonic cell lineages of the hermaphrodite and male gonads in Caenorhabditis elegans." Dev Biol 70(2): 396-417.

Kramer, M. A. (1991). "Nonlinear principal component analysis using autoassociative neural networks." AIChE Journal 37(2): 233-243.

Kreshuk, A. and C. Zhang (2019). "Machine Learning: Advanced Image Segmentation Using ilastik." Methods Mol Biol 2040: 449-463.

Krizhevsky, A., I. Sutskever and G. E. Hinton (2012). ImageNet classification with deep convolutional neural networks. Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1. Lake Tahoe, Nevada, Curran Associates Inc.: 1097-1105.

Kruskal, W. H. and W. W. Wallis (1952). "Use of Ranks in One-Criterion Variance Analysis." Journal of the American Statistical Association 47(260): 583-621.

Kurzweil, R. (2006). The Singularity Is Near: When Humans Transcend Biology, Penguin (Non-Classics).

Kusumoto, D. and S. Yuasa (2019). "The application of convolutional neural network to stem cell biology." Inflammation and Regeneration 39(1): 14.

Lawrence, S., C. L. Giles, A. C. Tsoi and A. D. Back (1997). "Face recognition: a convolutional neural-network approach." Trans. Neur. Netw. 8(1): 98-113.

Le Roux, N., Y. Bengio and A. Fitzgibbon (2011). Improving First and Second-Order Methods by Modeling Uncertainty. Optimization for Machine Learning. M. I. T. Press, MIT Press.

LeCun, Y., Y. Bengio and G. Hinton (2015). "Deep learning." Nature 521(7553): 436-444.

Lecun, Y., L. Bottou, Y. Bengio and P. Haffner (1998). "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86(11): 2278-2324.

LeCun, Y., P. Haffner, #233, o. Bottou and Y. Bengio (1999). Object Recognition with Gradient-Based Learning. Shape, Contour and Grouping in Computer Vision, Springer-Verlag: 319.

Legesse-Miller, A., R. H. Massol and T. Kirchhausen (2003). "Constriction and Dnm1p recruitment are distinct processes in mitochondrial fission." Mol Biol Cell 14(5): 1953-1963.

Levene, H. (1960). "Robust tests for equality of variances. In Ingram Olkin; Harold Hotelling; et al. (eds.). Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling." Stanford University Press: 278-292.

Li, Y., H. Gong, W. Wu, G. Liu and G. Chen (2015). An automated method using hessian matrix and random walks for retinal blood vessel segmentation. 2015 8th International Congress on Image and Signal Processing (CISP).

Lighthill, J. (1973). " Artificial Intelligence: A General Survey." Artificial Intelligence: a paper symposium.

Lin, Y. F. and C. M. Haynes (2016). "Metabolism and the UPR(mt)." Mol Cell 61(5): 677-682.
Lindeberg, T. (1994). "Scale-space theory: A basic tool for analysing structures at different scales." Journal of Applied Statistics 21(2): 225-270.

Lindeberg, T. (1998). "Feature Detection with Automatic Scale Selection." International Journal of Computer Vision 30(2): 77-116.

Loew, L. M., R. A. Tuft, W. Carrington and F. S. Fay (1993). "Imaging in five dimensions: time-dependent membrane potentials in individual mitochondria." Biophys J 65(6): 2396-2407.
Long, J., E. Shelhamer and T. Darrell (2014). "Fully Convolutional Networks for Semantic Segmentation." CoRR abs/1411.4038.

Mailloux, R. J., R. Bériault, J. Lemire, R. Singh, D. R. Chénier, R. D. Hamel and V. D. Appanna (2007). "The Tricarboxylic Acid Cycle, an Ancient Metabolic Network with a Novel Twist." PLOS ONE 2(8): e690.

McCorduck, P. (2004). Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence, AK Peters Ltd.

Meijering, E. (2012). "Cell Segmentation: 50 Years Down the Road [Life Sciences]." IEEE Signal Processing Magazine 29(5): 140-145.

Mikołajczyk, A. and M. Grochowski (2018). Data augmentation for improving deep learning in image classification problem. 2018 International Interdisciplinary PhD Workshop (IIPhDW).

Minsky, M. and S. A. Papert (1969). Perceptrons: An Introduction to Computational Geometry, The MIT Press.

Mishra, P. and D. C. Chan (2016). "Metabolic regulation of mitochondrial dynamics." J Cell Biol 212(4): 379-387.

Moller, J. V., B. Juul and M. le Maire (1996). "Structural organization, ion transport, and energy transduction of P-type ATPases." Biochim Biophys Acta 1286(1): 1-51.

Morris, R. L. and P. J. Hollenbeck (1993). "The regulation of bidirectional mitochondrial transport is coordinated with axonal outgrowth." J Cell Sci 104 (Pt 3): 917-927.

Murfitt, R. R., K. Vogel and D. R. Sanadi (1976). "Characterization of the mitochondria of the free-living nematode, Caenorhabditis elegans." Comp Biochem Physiol B 53(4): 423-430.

Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective, The MIT Press.

Nair, V. and G. E. Hinton (2010). Rectified linear units improve restricted boltzmann machines. Proceedings of the 27th International Conference on International Conference on Machine Learning. Haifa, Israel, Omnipress: 807-814.

Nunnari, J. and A. Suomalainen (2012). "Mitochondria: in sickness and in health." Cell 148(6): 1145-1159.

O. Ronneberger, P. F., and T. Brox (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation." arXiv

Okimoto, R., J. L. Macfarlane, D. O. Clary and D. R. Wolstenholme (1992). "The mitochondrial genomes of two nematodes, Caenorhabditis elegans and Ascaris suum." Genetics 130(3): 471-498.

Papert, S. A. (1966). "The Summer Vision Project."

Parrish, J., L. Li, K. Klotz, D. Ledwich, X. Wang and D. Xue (2001). "Mitochondrial endonuclease G is important for apoptosis in C. elegans." Nature 412(6842): 90-94.

Parrish, J., H. Metters, L. Chen and D. Xue (2000). "Demonstration of the in vivo interaction of key cell death regulators by structure-based design of second-site suppressors." Proc Natl Acad Sci U S A 97(22): 11916-11921.

Poria, S., E. Cambria, R. Bajpai and A. Hussain (2017). "A review of affective computing: From unimodal analysis to multimodal fusion." Information Fusion 37: 98-125.

Press, G. (2016). A Very Short History Of Artificial Intelligence (AI). Forbes.

Rambold, A. S., B. Kostelecky, N. Elia and J. Lippincott-Schwartz (2011). "Tubular network formation protects mitochondria from autophagosomal degradation during nutrient starvation." Proc Natl Acad Sci U S A 108(25): 10190-10195.

Ramirez, A., A. Heimbach, J. Grundemann, B. Stiller, D. Hampshire, L. P. Cid, I. Goebel, A. F. Mubaidin, A. L. Wriekat, J. Roeper, A. Al-Din, A. M. Hillmer, M. Karsak, B. Liss, C. G. Woods, M. I. Behrens and C. Kubisch (2006). "Hereditary parkinsonism with dementia is caused by mutations in ATP13A2, encoding a lysosomal type 5 P-type ATPase." Nat Genet 38(10): 1184-1191.

Reece, J. B., L. A. Urry, M. L. Cain, S. A. Wasserman, P. V. Minorsky, R. B. Jackson and N. A. Campbell (2014). Campbell biology.

Rolland, S. G., Y. Lu, C. N. David and B. Conradt (2009). "The BCL-2-like protein CED-9 of C. elegans promotes FZO-1/Mfn1,2- and EAT-3/Opa1-dependent mitochondrial fusion." J Cell Biol 186(4): 525-540.

Rolland, S. G., E. Motori, N. Memar, J. Hench, S. Frank, K. F. Winklhofer and B. Conradt (2013). "Impaired complex IV activity in response to loss of LRPPRC function can be compensated by mitochondrial hyperfusion." Proc Natl Acad Sci U S A 110(32): E2967-2976.

Rolland, S. G., S. Schneid, M. Schwarz, E. Rackles, C. Fischer, S. Haeussler, S. G. Regmi, A. Yeroslaviz, B. Habermann, D. Mokranjac, E. Lambie and B. Conradt (2019). "Compromised Mitochondrial Protein Import Acts as a Signal for UPR(mt)." Cell Rep 28(7): 1659-1669.e1655.

Ronneberger, O., P. Fischer and T. Brox (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation, Cham, Springer International Publishing.

Roweis, S. T. and L. K. Saul (2000). "Nonlinear Dimensionality Reduction by Locally Linear Embedding." Science 290(5500): 2323-2326.

Rumelhart, D. E., G. E. Hinton and R. J. Williams (1986). "Learning representations by back-propagating errors." Nature 323: 533.

Russell, S. and P. Norvig (2009). Artificial Intelligence: A Modern Approach, Prentice Hall Press.

Sadanandan, S. K., P. Ranefall, S. Le Guyader and C. Wahlby (2017). "Automated Training of Deep Convolutional Neural Networks for Cell Segmentation." Sci Rep 7(1): 7860.

Sato, Y., S. Nakajima, N. Shiraga, H. Atsumi, S. Yoshida, T. Koller, G. Gerig and R. Kikinis (1998). "Three-dimensional multi-scale line filter for segmentation and visualization of curvilinear structures in medical images." Med Image Anal 2(2): 143-168.

Schierenberg, E., J. Miwa and G. von Ehrenstein (1980). "Cell lineages and developmental defects of temperature-sensitive embryonic arrest mutants in Caenorhabditis elegans." Dev Biol 76(1): 141-159.

Schneider, C. A., W. S. Rasband and K. W. Eliceiri (2012). "NIH Image to ImageJ: 25 years of image analysis." Nat Methods 9(7): 671-675.

Sezgin, M. and B. Sankur (2004). "Survey over image thresholding techniques and quantitative performance evaluation." Journal of Electronic Imaging 13(1): 146-165, 120.

Shapiro, L. G. and G. C. Stockmann (2001). "Computer Vision." Prentice Hall: 137-150.

Shaye, D. D. and I. Greenwald (2011). "OrthoList: a compendium of C. elegans genes with human orthologs." PLoS One 6(5): e20085.

Silver, D., J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel and D. Hassabis (2017). "Mastering the game of Go without human knowledge." Nature 550(7676): 354-359.

Smith, M. R. and T. Martinez (2011). Improving classification accuracy by identifying and removing instances that should be misclassified. The 2011 International Joint Conference on Neural Networks.

Sørensen, T. (1948). "A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons." Kongelige Danske Videnskabernes Selskab 5(4): 1-34.

Sra, S., S. Nowozin and S. J. Wright (2011). Optimization for Machine Learning, The MIT Press.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov (2014). "Dropout: a simple way to prevent neural networks from overfitting." J. Mach. Learn. Res. 15(1): 1929-1958.

Starr, D. A. and M. Han (2002). "Role of ANC-1 in tethering nuclei to the actin cytoskeleton." Science 298(5592): 406-409.

Sulston, J. E. and H. R. Horvitz (1977). "Post-embryonic cell lineages of the nematode, Caenorhabditis elegans." Dev Biol 56(1): 110-156.

Sulston, J. E., E. Schierenberg, J. G. White and J. N. Thomson (1983). "The embryonic cell lineage of the nematode Caenorhabditis elegans." Dev Biol 100(1): 64-119.

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich (2014). "Going Deeper with Convolutions." CoRR abs/1409.4842.

Szekely (2002). "E-statistics: The energy of statistical samples." Technical Report 02-16.

Taha, A. A. and A. Hanbury (2015). "Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool." BMC Med Imaging 15: 29.

Tao, J. and T. Tan (2005). Affective Computing: A Review, Berlin, Heidelberg, Springer Berlin Heidelberg.

Torborg, C. L. and M. B. Feller (2004). "Unbiased analysis of bulk axonal segregation patterns." J Neurosci Methods 135(1-2): 17-26.

Tsang, W. Y. and B. D. Lemire (2003). "The role of mitochondria in the life of the nematode, Caenorhabditis elegans." Biochim Biophys Acta 1638(2): 91-105.

van der Bliek, A. M., M. M. Sedensky and P. G. Morgan (2017). "Cell Biology of the Mitochondrion." Genetics 207(3): 843-871.

Vincent, L. and P. Soille (1991). "Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations." IEEE Trans. Pattern Anal. Mach. Intell. 13(6): 583-598.

Vives-Bauza, C. and S. Przedborski (2011). "Mitophagy: the latest problem for Parkinson's disease." Trends Mol Med 17(3): 158-165.

Wallach, I., M. Dzamba and A. Heifets (2015). "AtomNet: A Deep Convolutional Neural Network for Bioactivity Prediction in Structure-based Drug Discovery." ArXiv abs/1510.02855.

Wang, X., C. Yang, J. Chai, Y. Shi and D. Xue (2002). "Mechanisms of AIF-mediated apoptotic DNA degradation in Caenorhabditis elegans." Science 298(5598): 1587-1592.

Weiss, K., T. M. Khoshgoftaar and D. Wang (2016). "A survey of transfer learning." Journal of Big Data 3(1): 9.

Wiering, M. and M. v. Otterlo (2014). Reinforcement Learning: State-of-the-Art, Springer Publishing Company, Incorporated.

Wiesel, T. N. and D. H. Hubel (1963). "SINGLE-CELL RESPONSES IN STRIATE CORTEX OF KITTENS DEPRIVED OF VISION IN ONE EYE." J Neurophysiol 26: 1003-1017.

Wolf, D. M., M. Segawa, A. K. Kondadi, R. Anand, S. T. Bailey, A. S. Reichert, A. M. van der Bliek, D. B. Shackelford, M. Liesa and O. S. Shirihai (2019). "Individual cristae within the same mitochondrion display different membrane potentials and are functionally independent." Embo j 38(22): e101056.

Yan, N., L. Gu, D. Kokel, J. Chai, W. Li, A. Han, L. Chen, D. Xue and Y. Shi (2004). "Structural, biochemical, and functional analyses of CED-9 recognition by the proapoptotic proteins EGL-1 and CED-4." Mol Cell 15(6): 999-1006.

Yang, L., Q. Long, J. Liu, H. Tang, Y. Li, F. Bao, D. Qin, D. Pei and X. Liu (2015). "Mitochondrial fusion provides an 'initial metabolic complementation' controlled by mtDNA." Cell Mol Life Sci 72(13): 2585-2598.

Yen, J., F. Chang and S. Chang (1995). "A New Criterion for Automatic Multilevel Thresholding." IEEE Trans. on Image Processing 4(3): 370-378.

Zemirli, N., E. Morel and D. Molino (2018). "Mitochondrial Dynamics in Basal and Stressful Conditions." Int J Mol Sci 19(2).

Zhao, Q., J. Wang, I. V. Levichkin, S. Stasinopoulos, M. T. Ryan and N. J. Hoogenraad (2002). "A mitochondrial specific stress response in mammalian cells." The EMBO journal 21(17): 4411-4419.

Acknowledgements

I would like to express my sincere gratitude toward Prof. Dr. Barbara Conradt and Dr. Carsten Marr for being such understanding and supportive supervisors and mentors. Thank you for all the insightful discussions and for providing moral support throughout the more challenging periods of the PhD.

Special thanks to all members of the Conradt and Marr lab from whom I received a lot of constructive feedback that greatly helped me rethink and optimise my PhD project. In particular, I would like to thank Simon and Laura who provided the data that made the MitoSegNet possible. Furthermore, I want to thank Eric, Nikhil, Jeffrey and Stephane for the interesting and inspiring discussions.

I would also like to thank Dietfried Molter and Werner Schimmel who not only trained me in becoming well acquainted with the Linux environment but also provided incredible technical support without which this project would have not been possible.

Last but not least I would like to thank my fiancée and my family for their constant love and support and without whom I would not have been able to reach this pinnacle of my academic career.

Curriculum vitae

Personal Data

Name	Christian Alexander Fischer

Education

10/2016 – 12/2019

PhD in Computational Biology (Dr. rer. nat)
LMU Munich, Faculty of Biology and Helmholtz Zentrum München, Institute for Computational Biology

Computational analysis of mitochondria in Caenorhabditis elegans using a Fully Convolutional Neural Network and common feature detectors

10/2014 – 10/2016

Master of Biology (MSc) [final grade 1.19]
LMU Munich, Faculty of Biology

Master thesis

Development of a semi-automatic approach for segmentation and lineage analysis of neural progenitor cells in the zebrafish embryo

10/2010 – 07/2014

Bachelor of Bioengineering (BEng) [final grade 2.2]
Munich University of Applied Sciences, Department of Applied and Mechatronics

Bachelor thesis
University of Manchester (03/2014 – 07/2014)

Gene expression analysis of in silico selected BCR-ABL downstream Targets in the CML cell line K562

2007 – 2010 State College Bad Tölz

2003 – 2007 Heinrich-Campendonk Secondary School Penzberg

2001 – 2003 Secondary School Benediktbeuern

1997 – 2001 Primary School Benediktbeuern

Research Internships

11/2015 – 09/2016 LMU Munich, Faculty of Biology

Establishment of multiple online courses on usage of computational tools in biology for students

09/2015 – 11/2015 LMU Munich, Faculty of Biology

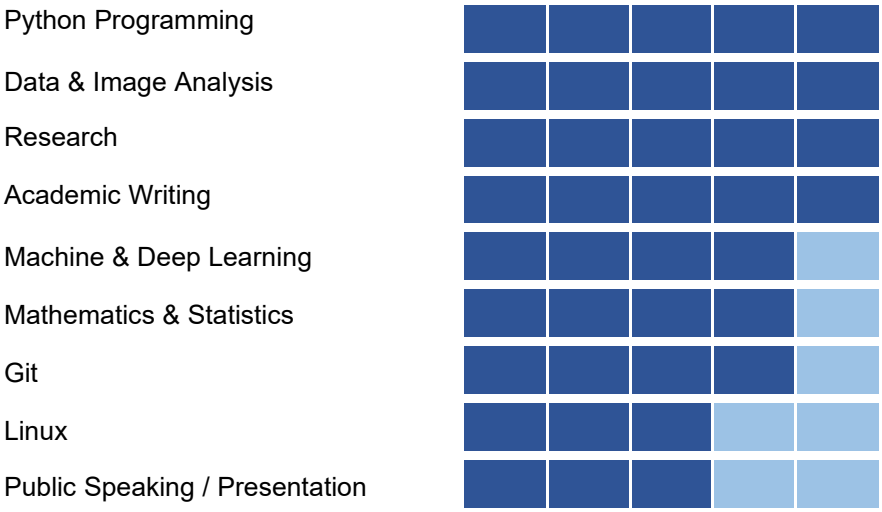
Development of automatic correction algorithm of chromatic aberration in 3D-SIM images

09/2013 – 02/2014 Helmholtz Zentrum München, Institute of Experimental Genetics

Evaluation of the effect of parameter variations in the computer model of dynamic expression patterns in somitogenesis

Skills

Technical



Personal

Language

Publications

As Author	Status
Rolland et al., 2018. 'Compromised Mitochondrial Protein Import Acts as a Signal for UPR^{mt}'	Published
Haeussler et al., 2020. 'Autophagy compensates for defects in mitochondria dynamics'	Published
Rackles et al., 2020. 'Impaired peroxisomal import triggers a peroxisomal retrograde signaling'	Under Review
Vrijsen et al., 2020. 'ATP13A2-mediated endo-lysosomal polyamine export provides a mitochondrial antioxidant response'	Published
Fischer et al., 2020. 'MitoSegNet: an easy to use deep learning segmentation model for morphological analysis of mitochondria'	Published

Work included

Tiedemann et al. 2014. 'Fast synchronization of ultradian oscillators controlled by delta-notch signaling with cis-inhibition', PLoS Comput Biol, 10: e1003843.	Accepted

Workshops

02/08/2018 – 11/08/2018 Light Sheet Microscopy Practical Course 2018 (EMBO)

Work experience abroad

03/2014 – 07/2014 University of Manchester
Manchester Institute of Biotechnology

Completion of Bachelor thesis