Deep Learning model to estimate the defocus in Cryogenic Electron Microscopy

Daniel Marchán Torres
UNIVERSITY OF NAVARRA

Index

Abstract

The research line of this project is related to advanced image processing oriented to the single particle analysis (SPA) in Electron Microscopy under cryogenic conditions (cryoEM). Electron microscopy (EM) is one of the best-established techniques for obtaining structural information from biological macromolecules. In this domain, the correct estimation of the Contrast Transfer Function (CTF) allows the characterization of the system and allows the subsequent correction of the images obtained. Consequently, the objective of this Master thesis is first to obtain, curate and preprocess a set of micrographs that will serve as input data for the second objective, which is to evaluate, improve and implement an existing model that uses a Convolutional Neural Network (CNN) to estimate the defocus values of the CTF. The main corrections are focus on the CNN model's architecture. Results confirmed improvements in forms of decreasing the number of parameters needed and decreasing the mean absolute error of the initial model. As well, they exhibit the possibility of integrating the model to the Scipion framework in form of a protocol that enables a previous defocus calculation for a better CTF estimation. This supportive preliminary evidence presents this method as a possible algorithm to predict the defocus values. Nevertheless, further tuning and refinement of the architecture needs to be study and the error needs to be lowered to be on the verge of a breakthrough.

1. Introduction

1.1 Project introduction

In the last decades, imaging techniques have allowed to address new domains and contribute with new solutions to different fields, such as tumor diagnostic in medicine, pattern recognition in computer vision or image enhancing in microscopy, offering a powerful source of detailed information. Among all these applications, this Master thesis is focused in medical imaging, more specially in the field of Structural Biology, which objective is the explanation and analysis of the 3D structure of macromolecular complexes. The 3D structures can help us to understand their action mechanisms and functions inside the cell which at the same time can help us study their biological behavior, drugs developments, or the design of new structural complexes, among others [1].

Different imaging techniques such as, nuclear magnetic resonance (NMR), X-ray crystallography, or electron microscopy (EM) are used in this filed. This last technique, considers the use of electrons as lighting source to get the image projections of the specimen. It is applied for the reconstructions of biological complexes via image processing techniques. Electron microscopy is one of the most disruptive techniques and can push further the resolution limit in comparison to others techniques available. This increment of the maximum resolution allows the study of new structures and complexes unable to be analyzed before [2].

This Master Thesis has been developed at the Biocomputing Unit of the National Center for Biotechnology (CNB) CSIC. This investigation group is dedicated to advanced image processing oriented to the analysis of individual

particles in electron microscopy under cryogenic conditions (cryo-EM). The project developed has a direct implication in the post processing of the raw image obtained by the microscope, more in detail, in the estimation of the defocus an essential parameter in the microscope´s contrast transfer function (CTF).

1.2 Project Description

Transmission electron microscopy (TEM), as most imaging devices, introduces aberrations that are usually modeled in Fourier space by the above-mentioned CTF. The CTF, mathematically characterizes the microscope transfer function that convoluted with the object results in the measured image. Aberrations and defocus are considered in the CTF; therefore, it describes how aberrations and defocus are introduced by the microscopy and affect the image. A correct Contrast Transfer Function (CTF) estimation allows the characterization of the system and allows the subsequent correction of the images obtained [3].

The objective of this project is to create and implement a deep learning model that better estimates the defocus values of the CTF. The precision of the model is defined as the mean-absolute error (MAE) between the estimated defocus and the one estimated with a CTF estimation algorithm [3]. The idea is to obtain a robust and generalized model and then, implement it to an integration software called Scipion [4]. This can be decomposed into three major modules:

- Obtaining, cleaning and preprocessing the data.
- Creation, training and evaluation of the deep learning model.
- Implementation of the trained model to the Scipion framework.

1.3 Methodology justification

Different studies agree that a correct defocus estimation is of great relevance for a correct CTF estimation and to what it implies later, a better CTF correction and therefore, a higher resolution achieved in the reconstruction [3], [5], [6]. For this purpose, a deep learning algorithm has been proposed. Deep learning and artificial intelligence techniques are useful tools for complex problem solving. Neural networks are universal learners and can learn features directly from data with which they are trained. Its architecture is design to recognize patterns and can produce a prediction output from unstructured input data as is the case of images. From DNN primarily one particular algorithm CNN is now the go-to model on every image related problem

Its power over the others Machine Learning algorithms comes from their ability to capture the Spatial and Temporal dependencies in an image through the application of relevant filters. In other words, the network can be trained to understand the sophistication of the image better [7]. A convolutional neural network model fits perfectly with the objective of this project, dealing with non-structural data and from all the information contained in an image a real value (the defocus) must be estimated.

1.4 Material description

The presented methods and algorithms are included in the software framework of Scipion, developed by the Biocomputing Unit at the CNB. The source code is open and free access. It is a software framework for integrating several 3DEM software packages through a workflow-based approach [4].

The post processing in cryo-EM starts with the TEM acquired images named micrographs or movies. Movies are video acquisitions; an alignment of its frames is required to convert it into a micrograph. In this project, alignment was done previously and we were dotted with a set of micrographs. A micrograph is nothing more than a motion corrected image. It is 2D and a greyscale image where the structural information of the macromolecule is given by the contrast between the ice (background) and the complex (signal). The higher contrast the better structural details can be observed [8].

For the deep learning model, the software package used for an intuited way of designing the network was TensorFlow in conjunction with the python library Keras, which gave us a more comfortable interface to interact with these design tools. The main lines of the design are easy to follow since the constructing process consist mainly in the consecutive addition of several layers. The first one added is an input layer, which accepts the images into the network, and, after the propagation of the input data through the network, the defocus values are predicted at the output.

2. State of the Art

The objective of this chapter is to provide the overall concepts and elements which describe the transmission electron microscope (TEM) and single particle analysis (SPA). As well as, to understand the importance of the defocus value in the CTF estimation.

2.1 Fundamentals of electron microscopy

One of the imaging techniques to study proteins and macromolecular complexes is electron microscopy, which allows high magnification images to be obtained. In this way, by means of large sets of images it is possible to reconstruct 3D structure of complexes with resolutions close to an atomic one. The use of electrons as illuminating source represents the main differences against optical microscopes. The use of electrons will extend the imaging capability for measuring smaller details due to its associated shorter wavelength. This allows changing the diffraction limit and consequently solving smaller structures [8].

The electron microscope is a device that works in vacuum, which imposes a serious restriction on the measurement of biological samples. In transmission microscopy, it is necessary to fix the sample making it watertight and solid. Also, the sample is required to be thin enough to allow electron transmission, as well as has to be as homogeneous as possible. The sample is placed on a grid, which will be measured by the microscope and there are two different kinds of grid preparation:

- **Negative Stain** (negative stain): the diluted sample is placed on the grid and then a heavy metal is deposited with a double intention, drying and fixing the sample. This method offers a high contrast but a low resolution.
- **Vitrification** (cryo): this was the true revolution in microscopy and the one studied in this Master thesis. As in negative stain, the solution is placed on the grid and it is vitrified, that is, the sample is frozen by means of a thermal shock. In this way, the molecules are arranged in their natural state.

Transmission Electron Microscope (TEM) consider the optical axis aligned in the same column, so that the electrons pass through the sample and form an image in a detector. In this way the sample is located between the collimator and the projection lens. By passing the electrons through the sample, it is possible to obtain information from the interior, as well as from the surface of the sample. In Figure 1, a TEM schematic is pictured:

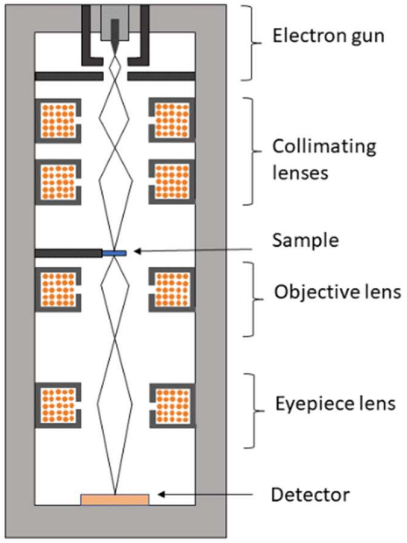

Figure 1. TEM schematic. Every electron microscope is composed of the following elements: **1. Electron gun**: filament of a heavy metal subjected to high temperature; **2. Condensing lenses**: its purpose is to illuminate the sample; **3. Sample holder**: where the grid with the studied macromolecules is located; **4. Objective lenses**: its purpose is to image the sample on a sensor and **5. Detector**: collects the signal [8].

The lenses of electron microscopes are magnetic lenses. A lens is defined as any element capable of deflecting a wave and focusing it on a point. If the incident light beam is collimated and therefore parallel to the optical axis of the lens, then at the output, if the lens is convergent, the beam will be directed towards the point called focus. The way that lenses are achieved in electron microscopy is through the use of magnetic fields capable of confining the beam and redirecting it to a

point. They are usually made up of several poles or coils that confine the beam as a magnetic mirror [9], as it is illustrated in Figure 2:

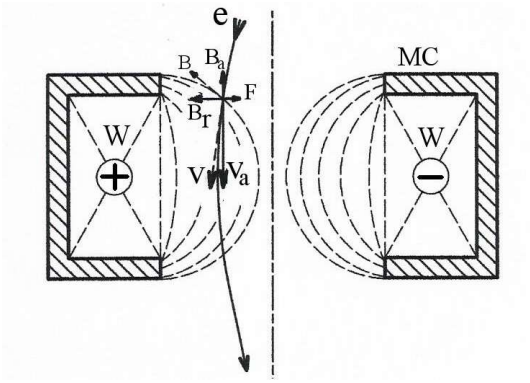

Figure 2. Schematic of a magnetic lens interacting with an electron.

In TEM the illumination beam passes through the object, as already noted above, this involves the use of thin samples. The high energy of the electrons together with the small thickness of the sample means a minimal electron-matter interaction. Therefore, the samples for practical purposes are 'transparent', so how it is possible to obtain an image of a transparent object. Most of the electrons pass through the sample without any interaction, while others undergo an interaction that we consider elastic due to the high energy of the beam. Although the first beam maintains its invariant phase, the second presents a small variation of the same as a result of a small interaction. The result is two-waves interference at the sample output. Which means, that they are waves of the same amplitude and wavelength, but out of phase by a certain amount [8]. The higher this amount (contrast), the better the specimen will be appreciated and there are two common ways to increase contrast:

- Deposition of heavy atoms (Negative Stain)
- Defocus images, as the phase also depends on the defocus and therefore, it is possible to move the detector from its stigmatic position by slightly

moving away from focus the image but thus achieving an observable phase difference. This will be comment in further detail in next sections.

2.2 Fundamentals of single particle analysis

Although different structural approaches can be followed to analyze the structures of macromolecules, this project focuses on cryo-EM single particle analysis (SPA). It is a structure determination technique (image processing) that allows to get the 3D density map from a set of cryo-EM images of a particular macromolecule. Images obtained by an electron microscope are called micrographs. These are very noisy images with projections of the particles contained in the sample. The fundamental hypothesis for this processing technique is to consider that all particles contained in the micrograph are projections of the same macromolecule, logically, since particles in the sample are randomly arranged, each projection will present a different orientation [8].

The set of successive tasks aimed to get the 3D density map is known as image processing workflow. The main steps of the general workflow are detailed from top (movies) to bottom (refined volume) in Figure 3:

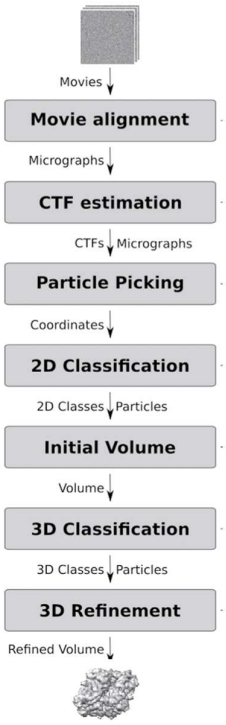

Figure 3. Single Particle Analysis workflow [4].

The workflow considers as input the movie acquired by the microscope and works as follow:

1. **Movie alignment:** new detectors called direct detectors (DDD) allow the acquisition of multiple frames. Thus, movies of the sample are acquired by the microscope. The electron beam induces movement in the sample (beam induces movement), as a consequence the particles are displaced from their initial positions in each frame. In order to correct BIM-induced image blurring and restore important high-resolution information, the stack of individual frames contained in each movie needs to be aligned. Only one image will be generated, and this image is called micrograph.
2. **CTF estimation:** the CTF characterizes the formation of the image in an electron microscope. Thus, the images obtained must be corrected using

the CTF in order to compensate for aberrations and obtain higher quality, more will be discussed in the next section.

3. **Particle Picking:** Since the reconstruction of the 3D density map of the macromolecule is based on images of individual single particles, the next step is essential in the image processing workflow. Picking is the process of selecting particles in the micrograph for extracting them later. It can be done manually, which requires to selected manually particle by particle; automatically, where a set of parameters is provided and the method determines particles position or semi automatically, an automatic method supervised by the user. The extraction will consider the CTF estimation in order to correct them.
4. **2D Classification:** Once all the particles have been extracted and CTF corrected, they are classified into groups. The idea is to be able to group all the particles that have the same orientation and thus achieve an average particle, which will be cleaner (less noise) than the previous ones. In order to classify the particles and group them according to their similarity, they must be aligned and with the same orientation. Consequently, classification step also requires an alignment of the particles.
5. **Initial Volume:** with the 2D classes, it is necessary to construct a first approximation of the macromolecule. The problem lies in the orientation of the classes. The representative of each class is a very clean particle, but we do not know from what point of view this representative would correspond to the projection of the particle. Right angles of each projection image are needed to reconstruct the 3D map. However, these angles are

unknown and we have to estimate them. The most popular way of estimating them is comparing the projections of a volume similar to ours, known as initial volume, with the images obtained from the microscope.

6. **3D Classification:** once the initial volume is known, it is possible to re-classify and determine classes or groups of particles that have the same orientation. In this way, a second reconstruction with two angles will give rise to a refined volume of much higher quality.
7. **3D Refinement:** finally, with the classes obtained and the new orientations, a higher quality volume is rebuilt. This new rebuilding process follows an iterative process in order to refine the initial 3D map.

2.3 CTF and defocus estimation

Since close to focus images of biological specimens embedded in vitreous ice generate very little contrast, we take our movie-frames out of focus, and retrieve systematically distorted images of our specimens. The alteration observed is due to the different transference of contrast for each spatial frequency. In an ideal microscope all the frequencies are transferred with total contrast (+1). In a normal one, some frequencies are transferred with contrast 0 or even -1. The CTF (Contrast Transfer Function) indicates how much contrast is transferred to the image as a function of the spatial frequency. The estimation of the CTF is the first step to correct it, repair its negative effect and retrieve our specimens undistorted.

Since part of the Fourier components are lost, attenuated or inverted, images of the specimens taken in a non-ideal microscope will appear blurry. We define this blurry effect with the PSF (Point Spread Function). The effect of the PSF makes that a discrete point in the specimen is reproduced in the image as a broad

point with a complex shape. The PSF can be directly estimated from the micrographs, since the PSF and the CTF are related through the Fourier Transform, and by computing the Fourier Transform of the PSF, we can directly estimate the CTF.

The acquired image is the convolution of the PSF with the object measured and due to the Fourier's properties (convolution theorem), the convolution in the reciprocal space is nothing more than the multiplication of both Fourier Transforms. Therefore, the Fourier Transform of the acquired image is the Fourier transform of the object times the CTF. Once both elements have been defined, the most common approach to estimate the CTF begins with the power spectrum density (PSD) calculation of the acquired image. In this PSD some concentric fringes, known as Thon rings, can be observed. These fringes establish the base for the CTF estimation since the zeros present in the PSD coincide with the zeros present in the CTF, allowing the feature extraction which is focused in the sinusoidal behavior of the signal and hence the CTF. An example is depicted in Figure 4:

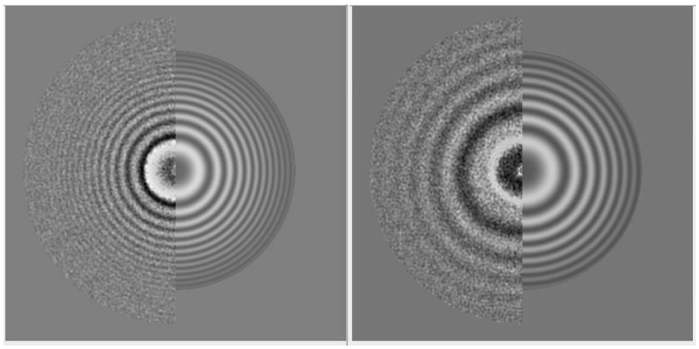

Figure 4. Power Density Spectrum of a micrograph (left half-plane) versus CTF estimation (right half-plane) by xmipp protocol. On the left is a correct CTF estimation and on the right an incorrect CTF estimation.

Once the micrograph PSD has been computed, CTF parameters corresponding to the experimental PSD are estimated. This is usually done by minimizing some measure of dissimilarity between the experimental PSD and a theoretical PSD determined from the CTF parameters [3]. The general form of the CTF can be modeled with the following mathematical expression,

$$CTF = \sin\left[-\pi\,\Delta z\,\lambda\;k^2 + \frac{C_s\,\lambda^3\,k^4}{2}\right]E(k)$$

where Δz is the defocus (the difference to focus), λ is the radiation wavelength, k is the spatial frequency, C_s is the coefficient of spherical aberrations of the objective lens and *E(k)* is the attenuation envelope. The defocus is our parameter to estimate, and its effects on the CTF can be explained with Figure 5:

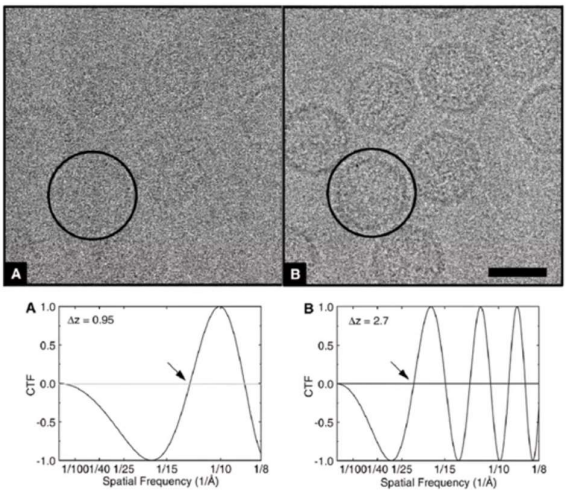

Figure 5. CTF estimation with two different defocus.

For instance, as you can see in Figure 5, if you defocus your sample, the CTF will suffer a shift, changing which spatial frequencies are strongly present in the image and which are lost. When you change the focus of the microscope, you will shift which wave function is being imaged on the detector, explaining its crucial importance on the CTF formulation [5]. In a perfect system, the defocus

would be the same for all the micrographs of the same acquisition and it could be set by the microscope controller. However, we are dealing with very high spatial resolutions and the error introduced by the system makes the defocus an unknown parameter between the ranges of the error.

There exist many different methods to estimate the defocus. The main disadvantages of these methods are that they try to bound the defocus within a predefined parametric function. This limits a lot its flexibility to learn complex patterns found in the PSD. For this reason, in an initial study conducted by a coworker at the CNB, a Convolutional Neural Network was proposed to estimate the mean defocus [6]. It is and was thought that the dimensionality of the network could give more flexibility to adapt and learn from the images, as the universal functions approximates they are. The model achieved in this previous study was not accurate enough and did not estimate defocus in both directions (U and V). However, it throwed some insights on how to use CNN to directly predict from the image the defocus value. Taking this in consideration, this Master Thesis is a continuation to improve and continue with this research line.

3. Materials and Methods

Once all the concepts were clarified in the past chapter, materials and methods used in this Master thesis are detailed in this chapter. The presented method is to be implemented in the software framework of Scipion, developed by the Biocomputing Unit of the CNB. The source code is open and free access. Scipion is a framework for integrating several 3DEM software packages through a workflow-based approach. It is mainly written in python programming language, as well as, having some bindings to execute code in C, C++, Matlab or Java. The

main characteristics that make this tool suitable for this study is that Scipion allows the execution of reusable, standardized, traceable and reproducible image-processing protocols. As well, these protocols incorporate tools from different programs while providing full interoperability among them and among protocols [4].

A protocol is a way to abstract algorithms as objects that perform a specific task and have well-defined inputs and outputs. Protocols serve as wrapper scripts that internally call one or more programs, allowing to store data (and their relationships) as an abstract representation instead of dealing with files, formats and to handle all data and metadata exchange between algorithms. Examples of data objects are Movies, Volumes, Masks, SetOfMicrographs, SetOfCTF, among others. The protocol where we want to implement our model is called deep micrograph defocus and accepts a set of micrographs as input and will connect as input of the CTF estimation protocol (xmipp3-CTF estimation) for a more robust defocus estimation.

Before entering in detail, it is important to remark that the python scripts mentioned in this chapter, form part of the CryoEMTools GitHub repository in the folder deepDefocusCTF [10]. Some of these scripts were created in a previous study conducted by a colleague [6]. New improvements and scripts were created by the former writer of this Master thesis. Also, due to the heavy computation needed, the preprocessing and storage of the Set of Micrographs and the training of the model were done remotely in the computation center (processors and storage) of the Biocomputing Unit at CNB.

3.1 Data structure and acquisition

One of the typical drawbacks in the usage of convolution neural networks, if not the most frequent, is the construction of a database with enough information able to properly train and generalized the developed network. To meet the criteria for a good design, good quality and huge quantity, were taking in consideration. First, the source of information is going to be limited to a specific kind of electron microscope working at a specific voltage acceleration, precisely at 200kV. The reason for this, is that the defocus is related with the acceleration voltage of the electrons and its wavelength.

Then, approximately 8 thousand micrographs have been analyzed from two different projects (5,595 and 2,470 respectively) acquired in the CryoEM Facility CSIC at CNB. The set of micrographs contained images of size 4096x4096 pixels with a pixel size of 0.85 Å per pixel. The quality of the input data was based on the xmipp3-CTF estimation protocol criteria and visually it could be observed based on the similarity between the estimated CTF and the micrograph Power Density Spectrum (PSD) as you can see in Figure 4. After estimating their CTFs and properly discarding the ones that did not fulfil the quality criteria, 7,171 micrographs were left.

Once enough data had been collected, we had to recover the micrographs PSDs with the defocus values (in both directions U and V) associated to them and store them somehow, they are suitable for training and testing our model. The data wanted to recover is stored in the xmipp3-CTF estimation protocol database (.SQLITE). Each protocol in Scipion framework contains a database

where it stores their results. Therefore, our first step was to connect to this database and copy the PSD and its defocus estimated. For this task, the ***prepareTrainingDataset.py*** script was used. For clarity this is an example of usage: *python3 prepareTrainingDataset.py <dirIn> <dirOut> <subsetNumber> <importDataFlag(0/1)>.* Where python3 is the interpreter and you have to specified where does the protocol .sqlite is and where you want to copy the data. As well, you have to specified the subset number and if you want to extract and store the defocus values associated to the PSD. This will be explained in the preprocessing section of this chapter. After this script is executed, the PSD images are copied to dirOut and a metadata file (.csv) is generated using pandas python package. The data frame contains as columns: the micrograph ID, the path of each PSD, its subset number (down sample value), the defocus in U and V and the voltage acceleration (200kV).

3.2 Data Preprocessing

Now that the information is ordered and stored in a .csv, we had to prepare the data to feed the Convolutional Neural Network. The model was trained looking for some homogeneity on the data in terms of resolution. To achieve this, a down sampling was introduced to obtain the same sampling rate in the set of micrographs. Down sampling is a useful tool for the CTF estimation and, specially, for the maximum index detection in the defocus calculation process. The reason for this is that, depending on the micrograph, a different disposition of the fringes can lead to a better detection of the maxima in the Fourier transform of the radial average function respect to the square of the frequency [6]. Due to

this, several possible down sampling factors can be used until the estimation is correctly performed.

It must be understood the implications of down sampling the PSD. A reduction in the sampling rate suppose that the maximum frequencies that can be sampled are also reduced, given by the Nyquist theorem. A reduction in the sampling rate suppose and expansion of the signal in the Fourier space, losing high-frequency information. Graphically, this can be explained with bigger rings shown when the sampling rate decreases, as you can appreciate in Figure 6:

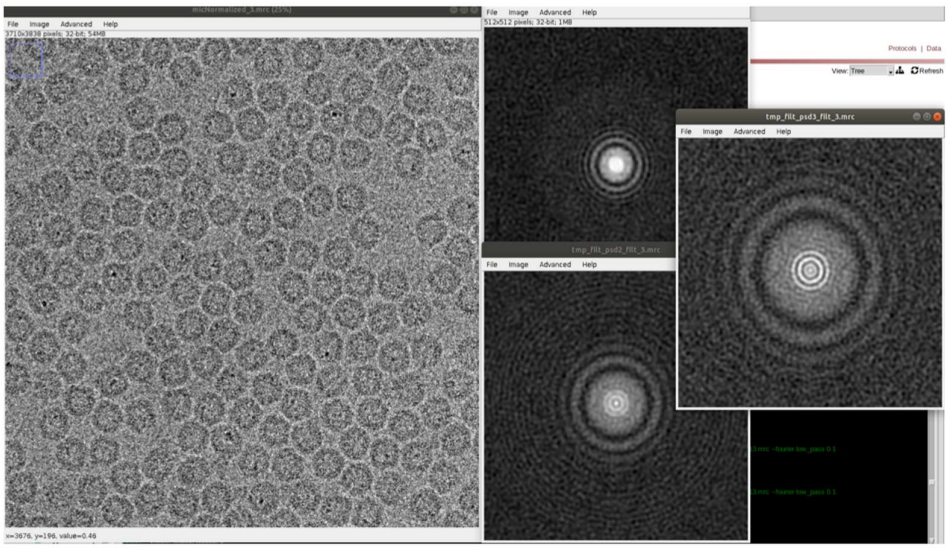

Figure 6. Micrograph and PSD with a different sampling rate. Micrograph is on the left and its respective PSD on the right at different sampling rate (1 Å/px (up), 2 Å/px (down), and 3 Å/px (right)).

Some preliminary results, showed that the number of CTFs properly estimated depends on the down sampling factor introduced to the micrographs [6]. Therefore, for a more robust implementation, three different CTF estimations with different down sampling factors were executed in order to obtain the PSD at

three different sampling rates: 1 Å/px, 2 Å/px, and 3 Å/px. In principle, each PSD image is associated with the resulting defocus of the CTF estimation. However, the most logical approach was to check the down sampled set that contained the biggest number of CTF properly estimated and use these defocus values for the rest as it should be the same.

This was done with the abovementioned ***prepareTrainingDataset.py*** script. In the protocol database, we were able to see with a tag (ENABLED) which micrographs were discarded and which were accepted based on the quality CTF estimation of the xmipp protocol. Therefore, we only took the defocus values from the enabled set that contained the biggest number of accepted CTF estimations. At the end we had 3 PSDs at a different down sample of the same image but the same defocus value. It is important to remark that the xmipp algorithm crops the PSDs in a 512x512 size window at the origin. This is done to capture the most important region of the signal, since at higher frequencies the captured is usually noise.

To feed the data to the model, we needed to stack the image in a NumPy array form. For this, the **prepareTrainingStack.py** script was used. This stacks in a 3D data structure which consisted in several blocks, each block of a 3D element with the size of the PSD (512x512 pixels) and 3 overlapping layers for each resolution provided of the image. It basically accesses to the each of the PSD files and extract all their information in a NumPy array that is sliced into a *(N x Height x Width x Subsets)* dimension array. Being *N* the number of PSDs in our database. At the same time, the defocus NumPy array is created with

dimensions (n, 2) with the target values to predict. It has 2 dimensions since we are predicting the defocus values in the two directions U and V. The usage of this script is very simple since you only have to specify where (dirOut) does your .csv and your images were stored from the previous step: *python3 prepareTrainingStack.py <dirOut>.*

Once the PSD were stacked in NumPy arrays, the rule of thumbs in Convolutional Neural Networks says, that images need to be normalized with mean 0 and standard deviation 1 (standardization). When the data is not normalized, the shared weights of the network have different calibrations for different features, which can make the cost function to converge very slowly and ineffectively [11]. Also, we believed that 8,000 images were not enough to train and test the model, so it was decided to double that number using data augmentation by rotating the images 90°. This operation does not affect the defocus values and therefore we could reassign the same to the rotated images. The image preprocessing used to transform a micrograph into the input of our model can be resume in Figure 7 steps:

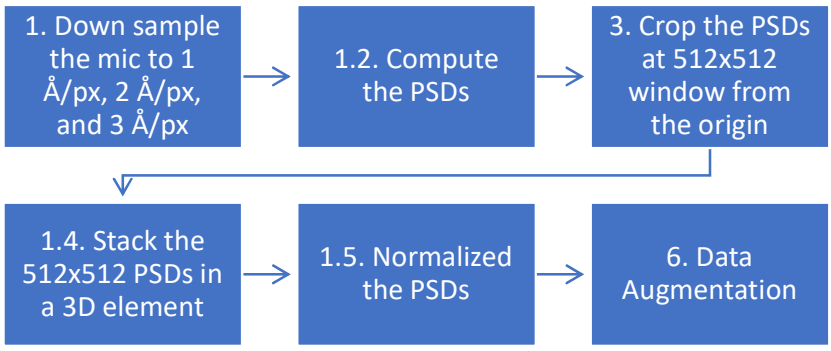

Figure 7. Preprocessing workflow

3.3 Statistical analysis

Now that our input data was explained and preprocess, it is important to understand the target values we are trying to estimate in our regression problem. For this some important statistical information are calculated. First, two histograms of the defocus values estimated with *xmipp3 – ctf estimation* are pictured in Figure 8:

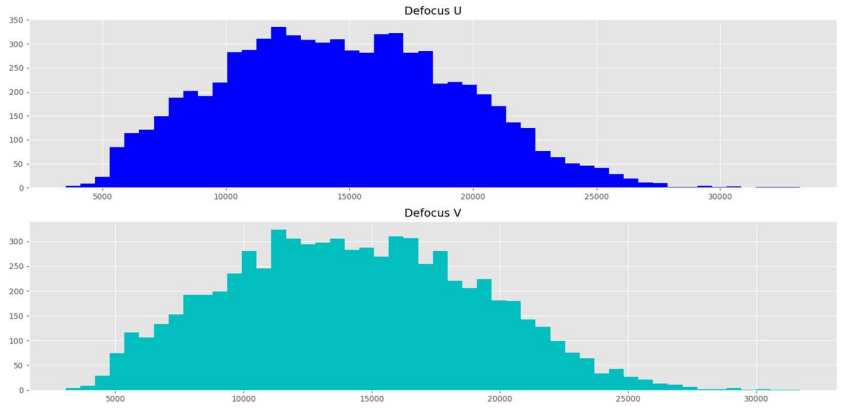

Figure 8. Histogram of defocus estimation in U direction (upper) and in V direction (lower).

The histograms give us an idea of the range in where our estimation should operate and the shape of their distribution. The defocus in both directions have a very similar gaussian shape distribution, symmetric around the mean. Then, to make a further statistical analysis was computed and presented in Table 1:

	Defocus U	**Defocus V**
Mean	14,651.2	14,283.4
Standard deviation	4,767.6	4,798.7
Median	14,481.2	14,124.4
Maximum	33,234.1	31,695.8
Minimum	3,515.3	3,061.5

Table 1. Statistical Analysis

Statistics confirmed what we saw in the histograms, both values have very similar statistics, the range for both defocus is between 3,000 to 33,000 with the mean and median almost perfectly located in the middle with an equal standard deviation. Also, an important measure to consider is the correlation between the two variables which is represented in Figure 9:

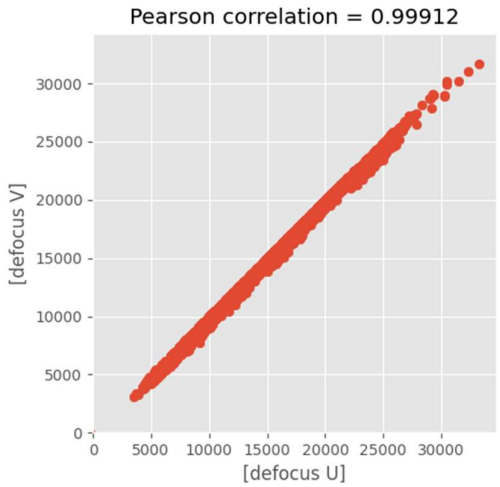

Figure 9. Pearson correlation of the defocus value in directions U and V

Statistics and distribution could give some hints regarding what kind of values our CNN could predict. With the symmetric distributions, we believe that the network is going to predict values close to the mean and may have some problems to predict some outliers located in the 30,000 ends. We have to be aware that failing to predict those outliers can increase our regression error. For the moment, outliers are not going to receive any special treatment. Also, an almost a perfect positive Pearson correlation can be considered as a quality measurement, since very different defocus values in both directions means that the acquired images is astigmatic.

The last source of errors for the data used to train the model is the criteria for the goodness of a CTF estimation. These criteria are composed by several features of the PSD and the CTF estimation that must be fulfilled in order to classify the estimation performed by the algorithm as a good estimation. Under this approach, if the PSD did not fulfill these criteria then the processed micrograph would be immediately discarded. It is important to understand that a bad established criterion could spoil a good estimation algorithm. Luckily protocol xmipp3 CTF estimation does the filtering by quality criterions when executed. The thresholds used for each of these criterions followed the quality standards in cryo-EM [3], [6].

3.4 Convolutional Neural Network

Once the data is collected, clean, preprocess and storage in the database, we can focus in the architecture and design of the CNN. For an easier way of designing the network, the software package TensorFlow has been used in conjunction with the library Keras. The project objective consisted in the development of a deep learning convolutional neural network able to predict the defocus (U and V) of a micrograph from its PSD and then, fully integrate this model to the Scipion framework. Before presenting the details of the architecture, few theoretical concepts are introduced to help with the explanation.

A Convolutional Neural Network is a Deep Learning algorithm which can take as input an image, learn important features in the image and then is able to differentiate, segment or quantify them. A CNN have the ability, with enough training, to capture the spatial and temporal dependencies in an image through

the application of relevant filters [12]. These filters are applied through a convolution operation (main building block). A convolution is a mathematical operation to merge two sets of information. In this case the convolution is applied on the input image using a convolution filter to produce a feature map. It performs the convolution operation by sliding this filter over the input at every location, do element-wise matrix multiplication and sum the result [13]. This architecture performs a better fitting to the image dataset than a common DNN, as the CNN reduces the images into a form which is easier to process, without losing features which are critical for getting a good prediction [12].

Now that a general idea of CNN is explained, the main lines of the design have to be covered. With Keras software package, the network is constructed through the addition of several predefined layers that can be characterized with some feature parameters. This interface allows the design of very different networks using the same tools. Here, are the most important layers we used for the confection of our model:

- **Input** (shape, name): the input layer is the first one and the shape and name of the data is specified. The rest of the layers will do automatically shape inference, so this is the only one which needs explicit information about the shape of the input data. Our network is fed with a 3D data structure which consist in the three PSDs at three different sampling rates of the same image: 1 Å/px, 2 Å/px, and 3 Å/px. Each block is a 3D element with size (512 x 512 x 3).
- **Conv2D** (filters, kernelSize = (l, h, w), activation): in this layer a convolutional kernel is created. We perform multiple convolutions on the

input, each using a different filter and resulting in a distinct feature map. All these feature maps are stack together and that becomes the final output of the convolution layer. In other words, this layer is going to convolve the input images with a set of filters from which some features are captured. CNN can have more than one convolutional layer, in fact the first layer will capture low-level features such as edges, gradient orientation, etc. With added layers, the architecture adapts to higher-level features giving us a deeper understanding of the image in the dataset. The kernel size would determine the perceptive field captured by the filter, therefore is one of the important parameters to tune.

- **Batch Normalization**: it is used to normalize the batches of data through a mathematical transformation of the input image by adjusting and scaling the activations. The reason behind applying this layer is to improve the speed, performance and stability of the network by reducing the covariance shift. The covariance shift is defined as the change from the input distribution to the internal layers of the network due to small changes from the input that get amplified down the network. Reducing the covariance shift implies a faster convergence of the network [14].
- **MaxPooling2D** (pool_size=(x,x)): this layer applies the maximum pooling operation for spatial data. Pooling layer is responsible for reducing the spatial size of the convolved feature. This is used to decrease the computational power required to process the data through dimensionality reduction. This operation is a wise-behavioral down sampling procedure in which, inside the filter specified, the algorithm will return the maximum value from the portion of the image covered by the Kernel [12].

- **Dropout** (rate): the dropout procedure consists in randomly setting a fraction of the input units to 0 during the training process. This application is done in order to avoid the overfitting of the network to the input data.
- **Flatten**: the procedure followed in this step is very simple. The shape of the data is prepared in order to fit as input of a dense layer of the neural network. With this data undergo a flattening process allowing a sequential feeding of the network.
- **Dense** (units, activation): this layer consists in a regular densely-connected neural network layer. All the previous steps were performed in other to a proper preparation of the data and feature extraction, but once they are completed, the data generated is ready to feed the dense layer of the deep neural network. This layer includes the deep learning network and the output generator.

Adding a Fully-Connected layer is a good way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and predict the defocus values using a linear activation function for the output layer, as every regression problem would do.

It is also necessary to introduce the concept of network hyper parameters and which are their implications in the functioning of the network. These network hyper parameters are going to characterize the model training. Specific changes on each of them lead to a different behavior of the network and the model

obtained as a whole. The first hyper parameter is the batch size, a batch is a fixed size group of training elements, the network is fed with every batch one following the next. Once each batch has been propagated through the whole network, the weights are recalculated, and the next batch is introduced. Batch size are used to speed up computationally, however too large batch size can produce poor generalization problems and too small batch size can take very long to train, for this project a batch size of 64 was used.

The training process is also divided in epochs. An epoch consists in an iteration of the training process in which the network has been fed with the whole training dataset once. This process can be iterated until the network converges or the number or set epoch for the training process is completed. Keep in mind that if training process last too many epochs, the size of the data and the network architecture is not compensated; the network can suffer from overfitting. Overfitting, is a phenomenon in which due to an intensive training process, the output model is too closely fitted to the specific data with which it has been trained. In this project, the number of epochs used by the default is 100, however to avoid overfitting different techniques where applied to avoid overfitting.

Although at the end of the training process the next followed step is the validating process, some inner testing during the training is needed to check how the network is behaving meanwhile. To accomplish this, TensorFlow has the functionality of before starting the training process some amount of the training data is removed before the construction of the batches and is used as a validation group at the end of the epoch. The network uses this data to test how the training

process is progressing. The measure of this error set during the training was the mean absolute error (MAE).

As an extra functionality, Keras allows to include specific behavioral features to the developed models, known as callbacks. These callbacks allow diverse and different functionalities, from a more detailed output of the developed training process, including graphical and analytical information, to changes in the behavior of the network. These are the ones used in this study:

- **TensorBoard**: visualization tool for machine learning processes. With the application of this callback, a log file is generated while the training process is taking place. Then, this file can be visualized with the TensorBoard Tool, offering a very comprehensive view of the process performed.
- **ReduceLROnPlateau**: this callback will decrease the learning rate of the network optimizer when the metric has stopped improving from a determined number of epochs. The learning rate gives a measure of the magnitude at which the network is going to change its weights when the input data is propagated through the network. Models usually benefits from a decrement of the learning rate when they get stagnated. In our case, we started with a learning rate of 0.001.
- **EarlyStopping:** this callback stop training when a monitored metric has stopped improving. This can be used to avoid overfitting if the metric monitored is taken from the validation set. As main argument you can choose the number of epochs with no improvement after which training will be stopped (patience), in our model we used a patience of 10.

The default structure for our convolutional layers is based on three Conv2D layers with a ReLU activation, followed by a BatchNormalization, MaxPooling and then finally a Flatten, Dropout and a Dense layer. Each of these layers is then followed by the final output Dense layer. Justification of the network architecture choices (kernel size of the convolutional layers, pooling size, number of dense units, etc) are to be commented in the next section.

4. Results

After the exposition of the materials and methods employed in the development of this project, results obtained are presented. The chapter is divided in two different sections. First, presenting the results obtained from the model search to find the optimal parameters and architecture for our CNN for the defocus estimation. Second, presenting the integration of the model to the Scipion framework in form of a functional protocol capable of interacting with different algorithms in the image processing workflow.

4.1 Model analysis

In the first section of this chapter, results obtained in the process of finding the best architecture for our deep learning model are presented. Only does who obtained the better results and are the most relevant are exposed. The original dataset contained 7,171 elements, after data augmentation it contained 14,342 elements, from which 10,362 were used for training and 1,828 for validating (85%), and other 2,152 for testing (15%). The split of the data was done randomly with function train_test_split and with the argument validation split.

The initial model architecture was taken from a previous research project done by a fellow coworker at CNB [6]. It was used to calculate the mean defocus value for micrographs acquired at 300kV. This architecture was the best out of three tested in that investigation and therefore it was a good starting point. In Table 2 the architecture is exposed and in Figure 9 the training is plotted:

Layer type	Output shape	Number of params	Features
Input Layer	515x512x3	0	(512, 512, 3)
Conv2D	498x498x16	10,816	16, (15, 15), "relu"
BatchNormalization	498x498x16	64	
MaxPooling2D	166x166x16	0	(3, 3)
Conv2D	158x158x16	20,752	16, (9, 9), "relu"
BatchNormalization	158x158x16	64	
MaxPooling2D	79x79x16	0	(2,2)
ConV2D	75x75x16	6,416	16, (5, 5), "relu"
BatchNormalization	75x75x16	64	
MaxPooling2D	37x37x16	0	(2, 2)
Flatten	21,904	0	
Dense	256	5,607,680	"relu activation"
Dropout	256	0	0.2
Dense	2	257	"linear activation"

Table 2. Initial model architecture

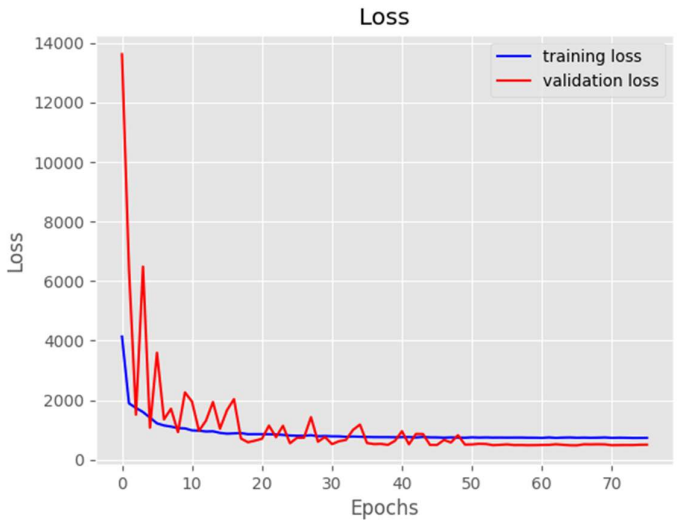

Figure 10. Training and validation curve for the initial model

For an initial evaluation of the model it is necessary to attend to the training and validation curve, meaning to check the error measurement of the estimation performed over the training dataset. From iteration 50, it presented a stable profile

around a MAE of 500 and we can see that the **EarlyStopping** callback stopped the training before we arrived to the 100 epochs set at the beginning. The lowest MAE value that the trained model presented once abandoned the transitory regime was 500.4, at iteration 70. To evaluate the performance of the model, testing set was fed to the model with a predicted error of 560.03. This is the ultimate evaluation step since this dataset has not been seen by the model and can be used to see how good generalizing our model is.

At this point, we realized that we were not taking advantage of the fact that we had 3 down samples of the same image, since we were using a sequential architecture for the CNN. This means that with this architecture we used the same Kernel for the triad of images. Conceptually, those images have a different representation of the Thon rings but the same defocus values, being smaller for those with the smaller sampling rate and bigger for the other ones. This gave us, that we could learn the same value in three different ways with three different types of kernels. The solution we took was to split the input into three independent Convolutional branches that will merge together once the most important features were collected and will enter concatenated in a flatten layer to the next Dense layers to estimate the defocus values.

Once we were on this path, we questioned the number of parameters used in the previous study, 5,646,917 trainable parameters. We considered that to learn such a “simple” task (sinusoidal attenuation of the fringes from the center to the edges), we did not need to capture such higher-level features. Therefore, we reduced the number of trainable parameters by reducing the number of filters and

dense units from the convolutional and dense layers respectively. In Figure 11, the new architecture with 119,096 trainable parameters is presented:

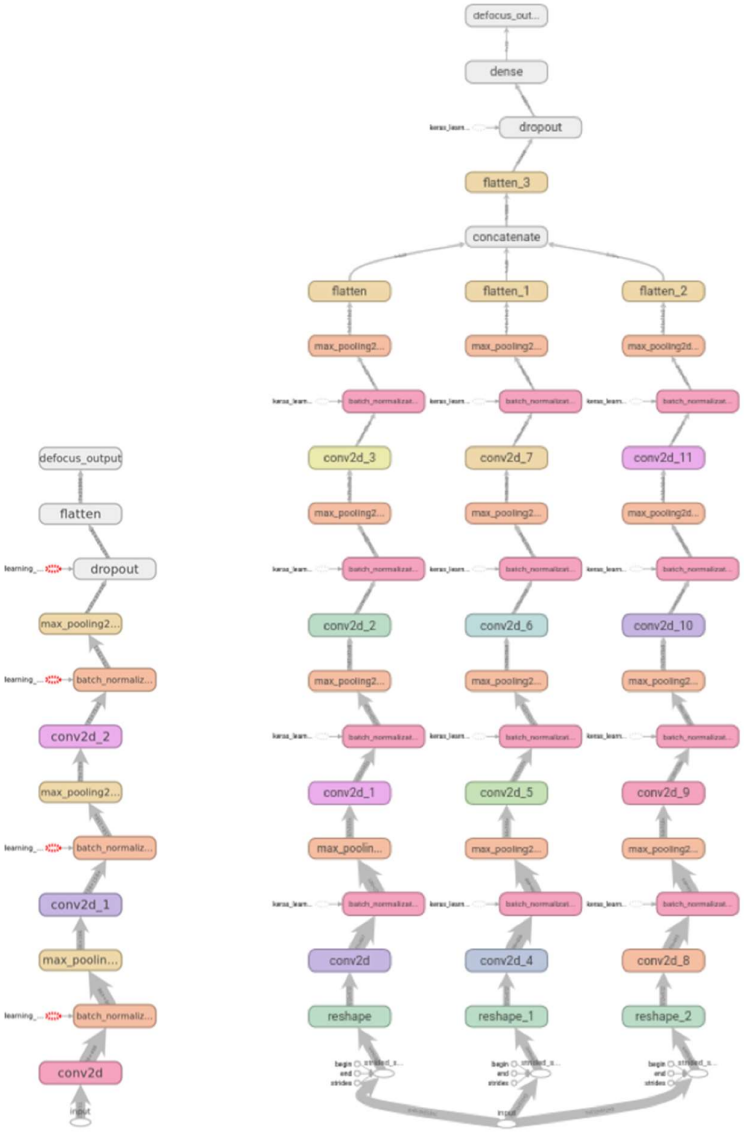

Figure 11. Picture of both architectures: the sequential model (left) and the parallel one (right)

As you can see, the input is split into 3 different images and each down sample goes in a different branch with different kernels. The kernel size used in the three branches were different, since the size of the fringes and the patterns we wanted to captured varied depending on the sampling rate. Therefore, we had

to use bigger kernels for bigger sampling rates and vice versa. We tried several filters size for each of the branches and the ones that worked the best are presented in Table 3:

Layer type	Output shape	Number of params	Features
Input Layer	515x512x3	0	(512, 512, 3)
Reshape	(1) 512x512x1	0	
	(2) 512x512x1	0	
	(3) 512x512x1	0	
Conv2D	(1) 501x501x16	2,320	16, (12, 12), "relu"
	(2) 498x498x16	3,616	16, (15, 15), "relu"
	(3) 493x493x16	6,416	16, (20, 20), "relu"
BatchNormalization	(1) 501x501x16	64	
	(2) 498x498x16	64	
	(3) 493x493x16	64	
MaxPooling2D	(1) 167x167x16	0	(3, 3)
	(2) 166x166x16	0	(3, 3)
	(3) 164x164x16	0	(3, 3)
Conv2D	(1) 162x162x8	4,616	8, (6, 6), "relu"
	(2) 157x157x8	12,808	8, (10, 10), "relu"
	(3) 150x150x8	28,808	8, (15, 15), "relu"
BatchNormalization	(1) 162x162x8	32	
	(2) 157x157x8	32	
	(3) 150x150x8	32	
MaxPooling2D	(1) 81x81x8	0	(2,2)
	(2) 78x78x8	0	(2, 2)
	(3) 75x75x8	0	(2, 2)
ConV2D	(1) 79x79x4	292	4, (3, 3), "relu"
	(2) 76x76x4	292	4, (3, 3), "relu"
	(3) 69x69x4	1572	4, (7, 7), "relu"
BatchNormalization	(1) 79x79x4	16	
	(2) 76x76x4	16	
	(3) 69x69x4	16	
MaxPooling2D	(1) 39x39x4	0	(2, 2)
	(2) 38x38x4	0	(2,2)
	(3) 34x34x4	0	(2,2)
ConV2D	(1) 37x37x2	74	2, (3, 3), "relu"
	(2) 36x36x2	74	2, (3, 3), "relu"
	(3) 34x34x2	74	2, (3, 3), "relu"
BatchNormalization	(1) 37x37x2	8	
	(2) 36x36x2	8	
	(3) 34x34x2	8	
MaxPooling2D	(1) 18x18x2	0	(2, 2)
	(2) 18x18x2	0	(2, 2)
	(3) 16x16x2	0	(2, 2)
Flatten	648	0	
	648	0	
	512	0	
Concatenate	1,808	0	
Flatten	1808	0	
Dropout	1808	0	0.3
Dense	32	57,888	"relu"
Dense	2	66	"linear activation"

Table 3. Final Model architecture. The numbers 1,2 and 3 corresponds to the down sampled branches.

The search was not exhaustive due to computational time, it would have been impossible to test a lot of different kernel size for each of the convolutional layers at each of the branches (training the model takes around 3-4 hours in GPU). The training of this new model can be appreciated in Figure 12:

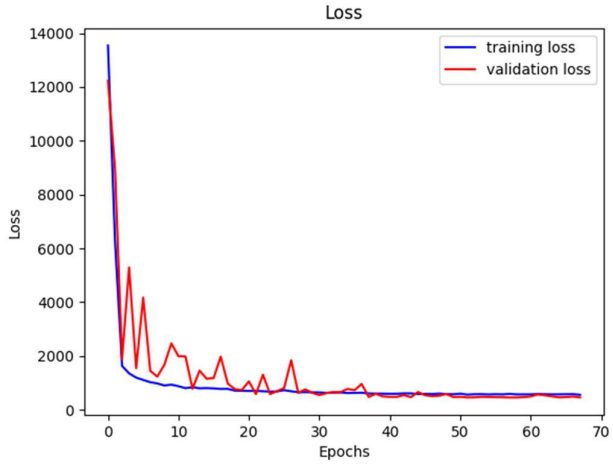

Figure 12. Final model training

In an initial evaluation of the model attending to the training and validation curve. From iteration 50 both the validation and the training curve present a stable profile around a MAE of 500. We can see that the **EarlyStopping** callback stopped the training process before we arrived to the 100 epochs set at the beginning of the training. The lowest MAE value presented once it abandoned the transitory regime was 458 at iteration 58. To evaluate the performance of the model, testing set was fed to the model with a predicted error of 470. Regarding overfitting, there is no any hint of this behavior. Both curves are close enough and the testing is very similar to both the training and validation MAE. This is important to remark since we used different techniques such as l1 and l2 regularization and dropouts in some of our layers.

In the Figure 13, we appreciate the **ReduceLROnPlateau** callback working in the training process. Every time the validation loss was stuck, the learning rate reduced to half. This was important as we on purpose chose a high learning rate of 0.001 to try to speed up the training.

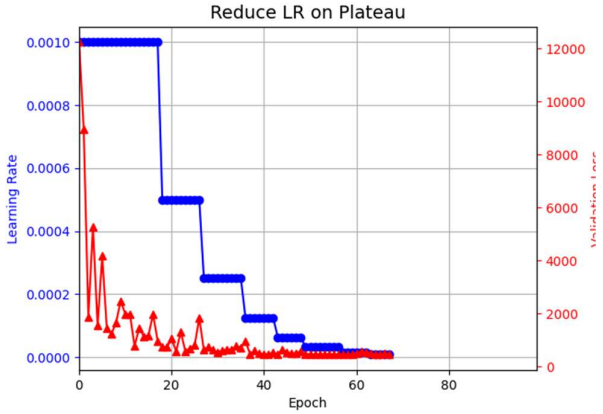

Figure 13. Learning rate decrease vs Validation Loss

When reduced the learning rate, the validation loss continuous decreasing, which means that if this was not done, our model would had probably fell in a local minimum without reaching its true potential. Regarding the error, no abnormalities have been seen. The model does not introduce big errors, neither tends to overpass or underpass the predictions. The error has a symmetric distribution around 0 as it can be appreciated in Figure 14:

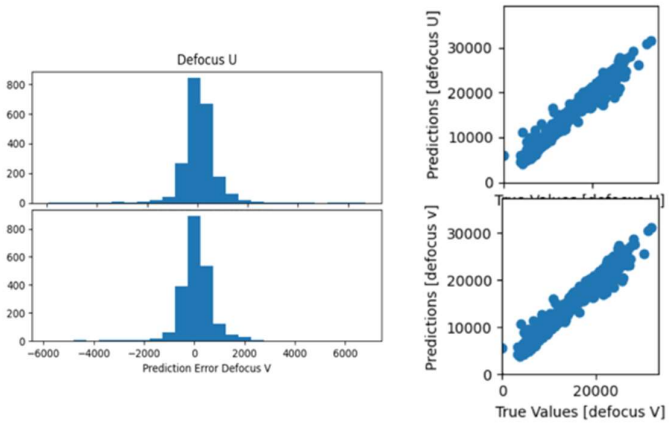

Figure 14. Defocus error (left) and Prediction vs True defocus values (right)

We can appreciate that for both defocus (V and U), predictions tend to be very similar to the true values. The visible diagonal is drawn between the predicted values and the true values, meaning that the error is mostly small values.

4.2 Program description

For the integration of our new method to the Scipion Framework, the creation of a protocol was needed. For this project, the main input value is a set of micrographs (Scipion object) containing the metadata of each micrographs and the output will be the same as the input but each micrograph has attached two extra parameter that is the defocus values estimated in U and V directions. The protocol form is displayed in Figure 15:

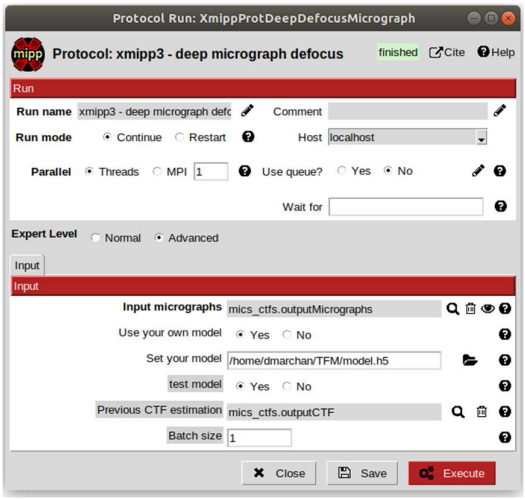

Figure 15. Deep micrograph defocus protocol form (advance options are visible)

All the protocols normal and advanced mode functionalities are displayed in Figure 15 and are the following:

- **Input interoperability** (normal mode): the input can be received from a different protocol that produces set of micrograph objects as output (i.e: pwem-import micrographs, any–movie alignment protocol, etc).
- **Model options** (normal mode): depending on the acquisition voltage of the samples the defocus varies significantly due the proportional relation with the EM lens convergence. Due to this, the same model used to predict micrographs at 200kV cannot be used to predict defocus in those acquired

at 300kV. As a result, with this protocol you are able to choose the model you want to use to predict the defocus values by selecting the path to it.

- **Testing option** (advance mode): by providing a set of previous CTF estimations, the protocol is in charge of comparing the defocus values predicted with the ones that were previously estimated by the means of other algorithms. To compare the prediction error, the mean absolute error is computed and displayed in the summary window.
- **Batch size** (advance mode): this is important when the protocol is running in streaming mode due to the fact that it waits until *n* number of micrographs are ready as input and feds them to the model prediction. Computationally speaking, this process is less expensive since you are calling the model only once per batch size rather than calling the model each time a new micrograph arrives.

It is intended that the defocus predictions will serve as input for the xmipp3 - CTF estimation protocol, where you can choose to pass previous estimated defocus values, as seen in Figure 16:

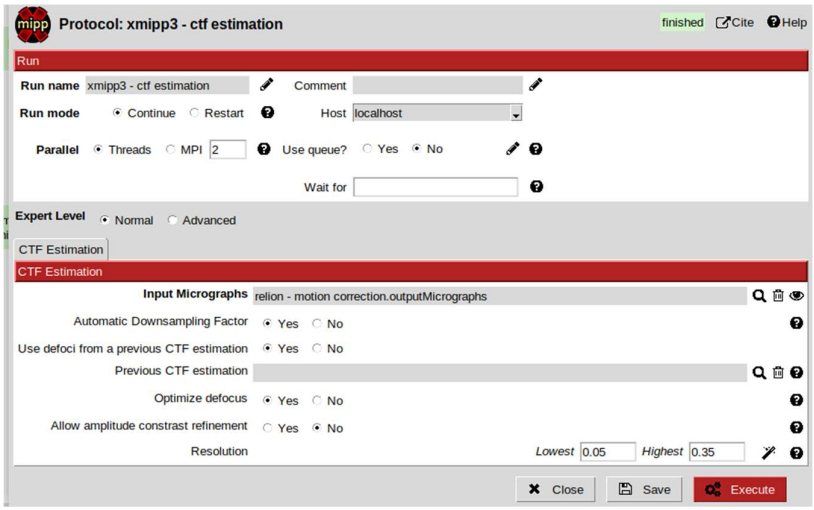

Figure 16. Xmipp3 – CTF estimation protocol (Use defoci from a previous CTF estimation).

At this point it is important to remark, that a set of micrographs can be quite heavy to download or process. As a consequence, the field of cryoEM points to speed up the computational time. This protocol was dotted with three different ways of speeding up the process:

- The first way was the use of **parallelizes computing**. Thanks to Scipion framework this feature can be easily programed in the protocol. Scipion has a simple mechanism to achieve parallelism by running independent steps simultaneously. For instance, the protocol parallelizes the processing of each micrographs using threads.
- The second consists of considering the processing of next steps of the SPA workflow before the end of the previous ones, **streaming**.
- **Batch size**

These approaches are attempts to speed up the whole processing workflow. An example of usage of deep micrograph defocus in the image processing workflow can be observed in Figure 17, where it receives micrographs from a movie alignment algorithm and predict the defocus values for the CTF estimation algorithm:

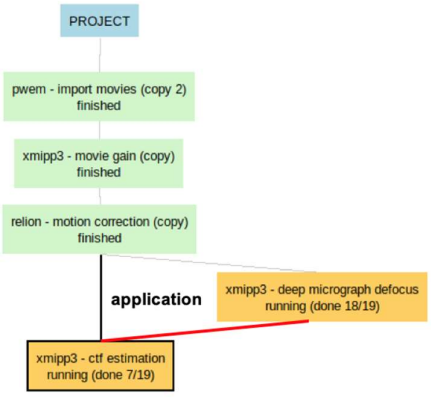

Figure 17. Example of streaming mode with deep micrograph defocus.

5. Discussion

Once the results are presented it is necessary to achieve some discussion in order to put them in context and understand its relevance. For this purpose, this chapter is divided into two sections. First, for the analysis of the deep learning model for defocus estimation and second, its potential application in the Scipion Framework and in SPA.

5.1 Results discussion

In this second section, results obtained from the developed deep learning network for defocus estimation are analyzed. As it can be seen in the results sections, different networks have been developed adding some changes from one to the next in order to achieve the better performance of the output model. Searching for the correct architecture of the Convolutional Neural Network was a tough and time-consuming task. Around 20 different models where trained and tested during this investigation, each of the models took around 4 hours to train and test the data. The difficulty of this task was to choose from a vast pool of options. A huge number of possible combinations of hyperparameters, parameters of each of the layers and different disposition of the layers. Due to this fact, every change of parameters, change of architecture, was carefully plan and thought before launching the training of a new model.

The initial model error of around 560 Armstrong compared to the range of our defocus values (3,000-30,000) seems to be small, approximately the 2% of the maximum value. Results in the training, validation and testing were correct, no sign of overfitting, good generalization for the test set and a stable condition were found as you can see in Figure 10. However, when dealing with resolution at level

of Armstrong, you need to push this error to the minimum as possible. In SPA, the magnitude of this type of errors will determine the resolution you will achieve in your final reconstruction, the lower the error, the higher resolutions you could achieve. Taking this into account, we decided to improve the model and try to decrease the prediction error.

When dealing with this difficult task of improving a working model, we had to look back to the foundation of the problem. It was known that for different down samples of the micrographs, a different rate of good CTF estimation was achieve. Therefore, three down samples of the same image as input to a neural network gave as much information as possible to the CNN, or that is what we thought. We realized that with the sequential architecture seen in Figure 11, we were not taking full advantage of this approach. We were feeding the input images to kernels with the same sizes and we know as matter of fact, that the Thon rings have different spatial dimension depending on the down samples used.

Having this in mind, the new approach raised. Continue feeding the network with 3 down samples, but extracting their features individually in different convolutional branches to capture the spatial difference found between down sampled images. Once these features were extracted separately, we hypothesized that since they all share the same information to predict the defocus values, the ending features flattened at each of the branches would be complementary. Then, by concatenating the three-end features we were able to fed these to a small feed-forward neural network that could be able to relate these extracted features with the defocus values. At the same time, we tried to reduce the number of parameters used in the initial model (from 5,646,917 to 119,096

trainable parameters). We considered that to learn such a “simple” task it was not necessary to capture such higher-level features.

Results were surprisingly better, the part of dividing into three different branches the convolutional part was hypothesized to work better, but the fact that reducing the number of parameters in such dimensions would make our model learn more and better was a very good result. Results showed that the new model had a validation error of 458 and a testing error of 470, decreasing by 50-100 Armstrong the mean absolute error in both cases. As well, the new model converged 10 epochs faster and had 2% of the trainable parameters that the initial model had. With all this, it is safely to say that we had improved and simplified the initial model by some levels of high considerations.

5.2 Application

Now that results of the model have been analyzed, its potential application in the Scipion Framework and in SPA has to be discussed. The software architecture (Scipion Protocol) to implement the model in Scipion processing workflow is ready to use. It loads a trained model, accepts as input a set of micrographs, preprocess the micrographs to be fed to the trained model and each of the micrograph is assigned with the two predicted defocus values at the output of the protocol. Then, these defocus values can be used as input of the CTF estimation protocol, as a previous estimated parameter. This was thought to improve the CTF estimation and therefore, the final resolution that you could achieve at the reconstruction following this image processing workflow.

Unfortunately, the prediction error of our model is not yet small enough to be implemented in Scipion workflow. This does not mean that results obtained in this Master Thesis were no good, actually on the contrary, with this study we are sure that following this new approach we are going to be able to reduce the error. Of course, this is not something trivial to do, and a lot of testing with different filters size and variations in our architecture had to be done in order to improve in this matter.

6. Conclusion and Future work

In this study, our findings provide supportive preliminary evidence that the defocus values can be estimated using Convolutional Neural Networks. We observed that with a relatively small Neural Network (119,096 trainable parameters), we were able to provide with defocus values prediction with an error of 470 Armstrong which shows an improvement in relation with the initial model. The new architecture suggests that the features to extract the defocus values, can be obtained from different down sampled images, meaning that using three different down samples in a different convolutional branch would increase the probability of extracting does features. Moreover, the possible implementation of a trained model to predict the defocus value to the Scipion framework was proven and a protocol able to interact with others and specially with the CTF estimation protocol was created. Further studies to improve the prediction error will be needed for a complete result confirmation and a final Scipion implementation.

Lack of time was one of the most challenging problems faced in this project. Luckily, this investigation is to be continue and improvements have already been thought and can be resumed as the followings:

- Increase the amount of data by means of more data augmentation or new data available. For an initial approach, 14,342 elements are suitable but for a more robust and generalized model more data is needed.
- A most robust “ground truth” for the defocus values. The defocus is an unknown parameter that has to be estimated. However, to improve in this matter, the defocus values can be obtained from a consensus of different algorithms that estimate the CTF. Fortunately, Scipion contains those algorithms and the consensus protocol to follow this approach. This will not only increase the reliability of the defocus values, but will help to improve the training of the model by introducing less experimental error. The aggregate solution of different algorithms with different mathematical approaches will reduce the bias and variance of the estimation [15].
- Different unexplored compositions are to be discovered and small changes in the model can make huge improvements in the output variable. Therefore, a more thorough search for the optimal hyper/parameters of the model is needed.

If future results confirmed our expectations, a brand-new method for the defocus estimation in Cryogenic Electron Microscopy would be discovered, helping to push the limit to reach higher resolutions.

Bibliography

[1] A. V. Vlasov *et al.*, “Raman scattering: From structural biology to medical applications,” *Crystals*, vol. 10, no. 1. 2020.

[2] T. Nakane *et al.*, “Single-particle cryo-EM at atomic resolution,” *Nature*, vol. 587, no. 7832, 2020.

[3] C. O. S. Sorzano, S. Jonic, R. Núñez-Ramírez, N. Boisset, and J. M. Carazo, “Fast, robust, and accurate determination of transmission electron microscopy contrast transfer function,” *J. Struct. Biol.*, vol. 160, no. 2, pp. 249–262, 2007.

[4] J. M. de la Rosa-Trevín *et al.*, “Scipion: A software framework toward integration, reproducibility and validation in 3D electron microscopy,” *J. Struct. Biol.*, vol. 195, no. 1, pp. 93–99, 2016.

[5] J. A. Mindell and N. Grigorieff, “Accurate determination of local defocus and specimen tilt in electron microscopy,” *J. Struct. Biol.*, vol. 142, no. 3, 2003.

[6] F. De Isidro, “Image processing algorithms for the determination of the optical aberrations o an electron microscope,” 2019.

[7] T. R. Cook, “Neural Networks,” *Adv. Stud. Theor. Appl. Econom.*, vol. 52, pp. 161–189, 2020.

[8] J. Luis, V. Prieto, C. N. De Biotecnología, F. De Ciencias, and U. Autónoma, “Local quality assessment of cryoEM reconstructions and its apllications,” 2019.

[9] C. F. E. Sierra, “Fundamentals of transmission electron microscopy, the technique with the best resolution in the world,” *Bogota*, no. February, 2019.

[10] F. De Isidro, “CryoEMTools.” [Online]. Available: https://github.com/fede-pe/cryoEMTools.

[11] A. Dertat, “Applied Deep Learning - Part 4: Convolutional Neural Networks,” *Medium*, 2017. [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2#7d8a.

[12] T. Christofer, “An introduction to Convolutional Neural Networks,” *Medium*, 2019. [Online]. Available: https://towardsdatascience.com/an-introduction-to-convolutional-neural-networks-eb0b60b58fd7.

[13] S. Saha, “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way,” *Mediium*, 2018. .

[14] M. Riva, “Batch Normalization in Convolutional Neural Networks,” *Baeldung*, 2021. [Online]. Available: https://www.baeldung.com/cs/batch-normalization-cnn.

[15] C. O. S. Sorzano, “Why is a single execution of a single algorithm not enough in CryoEM?,” *SME*, 2021.