Generative Image Deblurring for Human Protein Cell Classification

Rafael Fernandes Gonçalves
DETI - Universidade de Aveiro
Aveiro, Portugal
rfg@ua.pt
Student ID: 102534

Daniel Jorge Bernardo Ferreira
DETI - Universidade de Aveiro
Viseu, Portugal
djbf@ua.pt
Student ID: 102885

***Abstract*—Microscope images often suffer from motion blur, due to long exposure times or camera shake, compromising the success of computer vision tasks, such as image segmentation and classification. In this work, we address the problem of image deblurring by leveraging a dataset from the Human Protein Atlas (HPA), a project that aims to find protein organelles in human cells. We first simulate motion blur in the spatial domain through kernel convolution and then implement three Deep Learning methods: a Deep Convolutional AutoEncoder (DCAE) and a Generative Adversarial Network (GAN) - both from scratch, as well as the fine-tuning of MAXIM, a pre-trained deblurring model from Google Research. A simple Wiener filter is also applied as a baseline. We then evaluate and compare the performance of the proposed solutions based on two well-known image quality metrics: Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM). Finally, we validate our models by feeding a publicly available protein image classifier with our predicted images. The results show that the developed methods can effectively deblur the images and improve the classifier's F1 score by at most 226.5%.**

***Index Terms*—Deep Learning, Computer Vision, Image Deblurring, Convolutional AutoEncoder, Generative Adversarial Network, Fine-tuning, Human Protein Cell Classification**

I. Introduction

Biological imaging data is inherently complex and extracting meaningful information from it requires a deep understanding of the underlying biological processes, so it is often manually annotated by domain experts. On top of being time-consuming and error-prone, the process has scalability issues. As a consequence of the advances in high-throughput microscopy, the amount of biological imaging data has grown exponentially in recent years [1]. In this sense, there must be computer-assisted tools capable of segmenting and classifying microscope images by automatically interpreting cellular structures. Modern slide-scanning systems enable the acquisition of high-resolution images of tissue samples. Nevertheless, the sharpness of the image is usually preserved at the expense of a long acquisition time [2]. Speeding up the capture may lead to motion blur, a phenomenon that is caused by the relative motion between the camera and the objects being imaged. Aware of the importance of image quality in computer vision tasks, in this work, we implement Deep Learning solutions to restore their sharpness. For this purpose, we take advantage of a competition dataset hosted by Human Protein Atlas (HPA) project on Kaggle containing thousands of fluorescence microscopy images of human cells. The goal of the challenge, named "HPA Single Cell Classification" [1], is to segment and classify each individual cell in the image. Conversely, in the scope of the first assignment of "Complements of Machine Learning" (Complementos de Aprendizagem Automática) course, we focus on the image deblurring problem (Fig. 1).

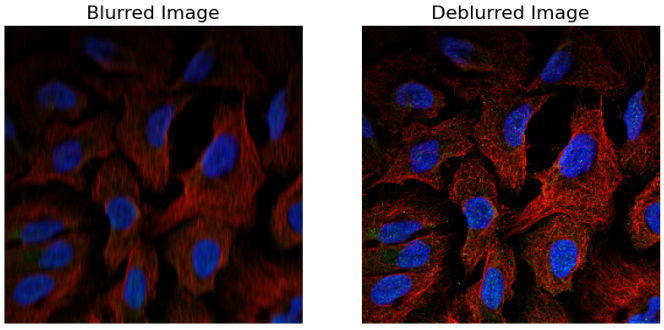

Fig. 1: Expected outcome of the image deblurring process on a sample HPA image.

Since all HPA images are quite sharp, we simulate motion blur by convolving them with a kernel, before applying the proposed models and a traditional Weiner filter, which served as a starting point for our comparative study. The following sections describe the methods used, the experiments conducted, and the results proving an effective restoration of a decent level of detail.

II. State-of-the-art

Image deblurring is a well-known problem in Computer Vision and has been addressed by several methods. Traditional approaches are based on the estimation of the blur kernel and the restoration of the image using inverse filtering or Wiener filtering. However, these methods are sensitive to noise and may not be effective in real-world scenarios. In recent years, Deep Learning has emerged as a powerful tool for image enhancement, which relies on impressive generative capabilities.

[1] https://www.kaggle.com/competitions/hpa-single-cell-image-classification

A survey on deep image deblurring methods [3] shows how they are more robust than traditional deconvolution methods, which typically define the task as an inverse filtering problem, where the blurred image is modelled as the outcome of a convolution with a given blur filter that is assumed to be known. In contrast, deep learning methods are able to recover both the sharp image and the blur kernel itself, even when the latter is unknown (blind deblurring). However, the authors recognise that there are scenarios where simpler methods can outperform complex ones, so it is worth considering them as a decent starting point. Most of the survey content draws our attention to the latest deep learning methods, discussing the most used image quality assessment metrics, network architectures and loss functions (Tab. I).

TABLE I: Summary of some of the most popular Deep Learning Techniques for Image Deblurring

Aspect	Options	Considerations
Metrics	PSNR, SSIM	They are the most popular image quality assessment metrics.
Network Architectures	U-Net-based AutoEncoders, GANs, Cascaded Networks	U-Net was designed for image segmentation, but can be adapted for deblurring. For GANs, the authors emphasise that human visual perception are not always consistent with image quality metrics. Cascaded networks have much more parameters to train (computationally expensive).
Loss Functions	Pixel-wise (L1 and L2), Perceptual Loss	L2 tends to produce better PSNR values than L1, but it ignores long-range dependencies in the image Perceptual loss functions are a solid alternative, as they are based on the high-level features extracted by a pre-trained network, being more close to human perception.

PSNR: Peak Signal-to-Noise Ratio
SSIM: Structural Similarity Index
GAN: Generative Adversarial Network
L1 loss: Mean Absolute Error (MAE)
L2 loss: Mean Squared Error (MSE)

A recent study [4] introduces a deblurring Masked AutoEncoder (MAE) tailored for ultrasound image recognition. This method builds upon MAE for self-supervised learning, but incorporates an additional deblurring task during the pre-training stage. It uses a Vision Transformers (ViT) encoder-decoder architecture to reconstruct the original image from the blurred and masked input. This choice of architecture capitalises on the recent success of Vision Transformers in image processing tasks, potentially allowing the deblurring MAE to capture more intricate spatial relationships within ultrasound images compared to convolutional approaches. Their findings demonstrate that the deblurring MAE outperforms vanilla MAE and achieves state-of-the-art performance in ultrasound image recognition, suggesting that deblurring could be a valuable pre-processing step to improve performance in image recognition and classification tasks.

As mentioned, GAN is a increasingly popular solution for deblurring problems. The DeblurGAN [5] is a conditional GAN that reaches state-of-the-art performance in terms of structural similarity and visual appearance. The article corroborates a strategy that we have followed in this work: validating the deblurring models with a downstream task, such as image classification. In this case, among other experiments, the authors conducted an object detection benchmark on a pre-trained 'You Only Look Once' (YOLO). The results are very insightful, as they show that precision tend to be higher on blurry images given the lack of sharp edges, which are often misclassified as objects. Nevertheless, DeblurGAN significantly improves the performance of the object detection model in terms of recall and consequently the F1 score. More recently, in another project, the restoration power of GANs was harnessed to accelerate the acquisition of microscope images without loosing definition [2]. The validation was based on PSNR and SSIM metrics, but classification and segmentation tasks were not applied, which opens a new research opportunity.

III. Dataset Analysis

A. Data Description

"HPA Single Classification" annotated dataset consists of 21806 microscope images, each containing multiple cells. The images have a resolution of 2048x2048 pixels and 4 channels (RGBY) separated in different PNG files. Besides the traditional RGB channels, each image has an additional yellow channel. Here, colours have meaning: microtubule channels are red, nuclei channels are blue, endoplasmic reticulum channels are yellow and the protein of interest is green. There are 19 labels in total, 18 of which represent the presence of a protein in the cell and one label represents a negative or unspecific signal. As the purpose of the competition was multi-label classification, we were also provided with a CSV file containing a list of labels for each image.

B. Exploratory Data Analysis (EDA)

Even though our main goal was to deblur the images, as we intended to perform a classification task for validation, EDA was useful to understand the data we were dealing with. First, we checked the distribution of the labels in the dataset. We found that the dataset is highly imbalanced, with some labels being more frequent than others. Label 0 (nucleoplasm) appears many times, while labels 11 (mitotic spindle) and 18 (negative/unspecific signal) are far less frequent. This is a common issue in multi-label classification tasks, as the model may be biased towards the most frequent labels. Possible solutions are downsampling the majority classes and upsampling the minority classes. For this work, we decided to keep the raw dataset, since we were not interested in the absolute values of classification metrics, but in the relative improvement of the model after deblurring the images. We were also interested

in the combination of classes, i.e, how many classes appear together in a single image. We found that most images have only one or two classes, but there are also images with up to 5 classes. Lastly, EDA confirmed a conclusion that organisers and top competitors from HPA challenge have already drawn: microtubles (red channel) and endoplasmic reticulum (yellow channel) are typically in the same location, so it is safe to discard the yellow channel and keep only the RGB format. This was particularly useful for us, as we intended to apply perceptual loss functions based on pre-trained models that only accept 3-channel images.

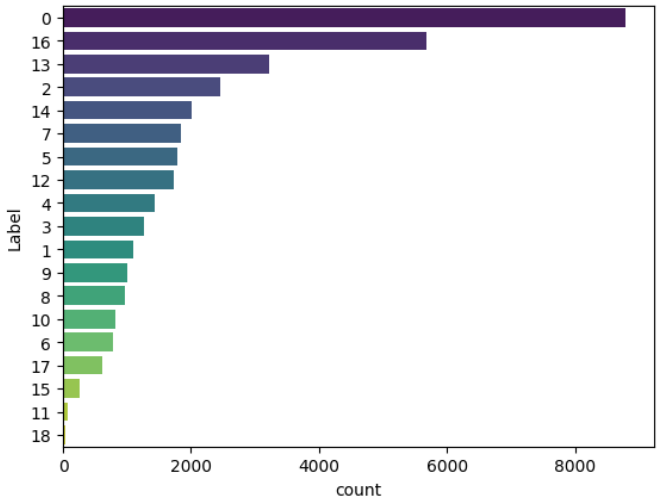

Fig. 2: Distribution of classes

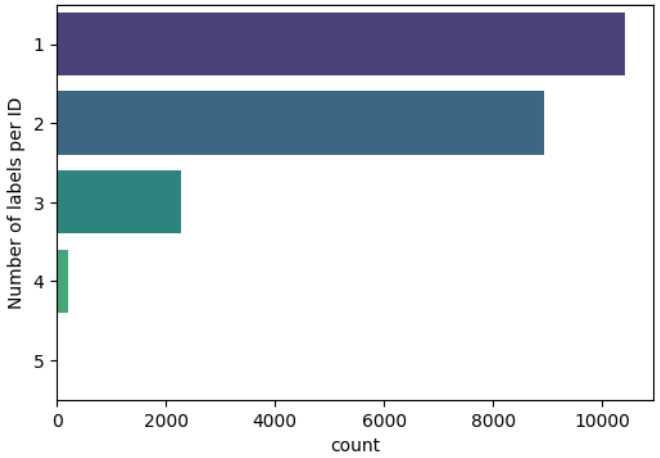

Fig. 3: Absolute frequency of each class list size.

C. Data Preprocessing

As previously mentioned, we simulate motion blur by convolving the images with a kernel. Using OpenCV, we generate a kernel with a random degree (the greater the degree, the higher the blur) and a random angle (orientation of the blur). The kernel is generated by applying a rotation matrix to a diagonal matrix of ones. The resulting blur images are stored in a folder 'blur', while the original ones are kept in a folder 'sharp'. Both types of images are resized to 256x256 pixels, as all developed models require this size as input to reduce the computational burden. The dataset is huge, so we only keep 5000 images for further splitting into training and validation sets and another 1000 images to test the models on the protein classification task. Split ratios are further mentioned in each model subsection. When loading both folders, a pre-processing pipeline is applied to the images. Images come in various scales and pixel values might range from 0 to 255. Pixel normalisation is important to avoid having larger values dominating the learning process. Otherwise, we may lead the model to a slow training or to converge to a sub-optimal solution. Therefore, we normalise the images to the range [-1, 1] for GAN and [0, 1] for AutoEncoder and MAXIM. Initially, we were applying [0, 1] range for all of them, but as an attempt to improve the GAN results, we decided to change the range to [-1, 1], since many models lean on the assumption of zero-centred inputs. Additionally, we apply data augmentation techniques, because we want to increase the diversity of the training data despite the dataset size constraints. We rather have a smaller and more diverse dataset than a larger dataset without any transformation. Hence, depending on the model, we apply random flips, rotations and zooms with a given probability (Fig. 4).

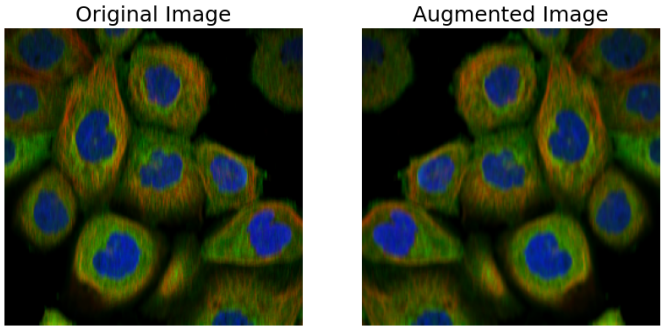

Fig. 4: Example of a random zoom and flip augmentation.

IV. Model Implementation

The following section describe the architecture and training process of the three proposed solutions. Regarding hyperparameter tuning, we highlight that a grid search would not be feasible given the high training time of each model. Nevertheless, research in optimisation problems have shown that carrying out a random search for the best parameters can be more efficient than a grid search specially when the configuration space is very large [6]. In this sense, we explored several random hyperparameters and selected the ones that provided the best results. The experiments were executed on a Kaggle environment with a GPU accelerator.

A. Deep Convolutional AutoEncoder (DCAE)

An autoencoder architecture was implemented to learn a latent representation of the sharp images from their corre-

sponding blurred versions. The encoder part of the autoencoder consisted of three convolutional layers with increasing filters (96, 128, and 256) designed to extract progressively more complex features from the blurred images. Each convolutional layer used a 3x3 kernel size for capturing local spatial information, ReLU activation for introducing non-linearity, and 'same' padding to preserve the spatial dimensions throughout the encoding process. A 2x2 stride max pooling operation was applied after each convolution to reduce the spatial dimensionality and introduce invariance to small shifts in the input. The latent vector dimension was set to 256, acting as a compressed representation capturing the essential features of the sharp image.

The decoder part then mirrored the encoder architecture using transposed convolution layers to reconstruct the sharp image. Transposed convolutions essentially learn to upsample the feature maps and recover the spatial dimensions lost during the encoding stage. All convolutional layers, including those in the decoder, used He normal initialisation [7].

The model was compiled with the Adam optimiser [8], a popular optimisation algorithm known for its efficiency and effectiveness in various deep learning tasks. The perceptual loss function served as the guiding principle for the training process. This loss function was calculated by feeding the blurred and reconstructed images through a pre-trained VGG16 model [9], excluding the top classification layers. The VGG16 model acts as a pre-trained feature extractor, having already learned meaningful representations for a large image dataset (ImageNet). By measuring the mean squared error between the resulting feature maps from the VGG16 model, the perceptual loss function encourages the autoencoder to not only reconstruct the pixel values of the sharp image but also aims to preserve the higher-level features learned by the VGG16 model. These higher-level features often encode semantic information about the image content, promoting reconstructions that are not only visually similar to the sharp images but also semantically meaningful.

During training, 80% of the dataset was used for training the model, and the remaining 20% was used for validation. The model was trained for a maximum of 100 epochs with a batch size of 32, allowing sufficient time for the autoencoder to learn effective latent representations for image deblurring.

The specific hyperparameters used for training the autoencoder are summarised in Table II.

TABLE II: DCAE Training Hyperparameters.

Batch size	32
Optimiser	Adam
Learning rate	1e-3
Number of epochs	100

Despite the implemented architecture, the model appears to be suffering from overfitting, as evidenced by Fig. 5. While the model achieves decent performance on the training data, it fails to generalise well to unseen examples. The generated image exhibits artefacts, indicating the model is not effectively capturing the underlying features necessary for successful deblurring.

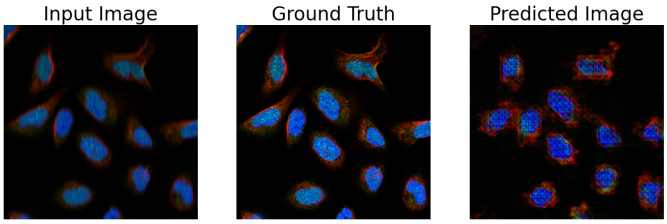

Fig. 5: Example of a generated image by the DCAE model.

We attempted various strategies to mitigate overfitting during training. These included employing early stopping to halt training when validation loss plateaus, dynamically reducing the learning rate to prevent it from surpassing optimal values, incorporating dropout layers to promote feature independence, applying L2 regularisation to penalise the model for large weights, and even experimenting with fewer filters and different network architectures to reduce model complexity. Disappointingly, none of these techniques yielded significant improvement. The model achieved a PSNR of 18.7227 dB and SSIM of 0.2650, indicating poor image reconstruction quality.

B. Generative Adversarial Network (GAN)

A GAN consists of two networks: a generator and a discriminator. The generator is responsible for generating new data samples, while the discriminator tries to distinguish between real and fake samples. Starting with the generator, the input image is passed through 3 blocks of layers that double the depth of the input volume and try to capture higher-level features. Each block consists of a convolutional layer, a batch normalisation layer that contributes to fast convergence, and a ReLU activation function, except the first block that also includes a reflection padding layer before the convolution. This type of layer came up as an alternative to zero padding, which creates a hard boundary between the image and the padding. This may originate undesired artefacts in the output image, so reflection padding is used in image generation tasks to provide more realistic data at the edges and to lead to smoother gradients [10]. After the aforementioned blocks, the input volume goes through 9 residual blocks - with 256-filter convolutional layers and a dropout layer disabling pixels with a given probability. The output of the residual blocks is passed through two upsampling layers that double the height and width of the input volume, followed by convolutional layers that halve the depth of the input volume. An alternative to an upsampling layer would be a transposed convolution. Both are forms of deconvolution, but while the first one is a simple interpolation for scaling up, the second one is an operation whose kernel is learned during training. Even though transposed convolutions are more flexible, they increase a lot the number of parameters to be trained. Given the amount of layers already present in the model, the limited dataset size and the computational constraints, we opted for an upsampling

layer. Finally, the last convolutional layer is followed by a hiperbolic tangent activation function. There is also a skip connection between this stage and the input image to tackle the vanishing gradient problem.

Regarding the discriminator, we use a "PatchGAN" classifier [11] that aims to classify if each NxN patch of an image is real or fake. The input image is concatenated with the target image and passed through 3 downsample blocks (less height and width, more depth), each consisting of a convolutional layer, a batch normalisation layer and a Leaky ReLU activation function. The output of the last downsample block is zero-padded and passed through a convolutional layer with 512 filters, a kernel size of 4x4 and stride of 1 to maintain the spatial dimensions. The volume resultant from this layer is once again batch normalised and passed through a Leaky ReLU activation function. Then, the volume is zero-padded again and passed through a convolutional layer with a single filter and a kernel size of 4x4. The final output of the model is a probability map that contains a single value between 0 and 1 for each position, indicating the likelihood of the corresponding patch being real.

As there are two networks, we need two loss functions. The generator loss is composed of two terms (1): the gan_loss and a l2_loss whose importance is regularised by a hyperparameter λ. The GAN loss is the negative mean of the discriminator output (disc_generated_output), while the L2 loss is the mean of the squared difference between the high-level features extracted by VGG16 (method vgg16_features) from the target image (target) and the generated image (gen_output). Even though it is called L2, note that it not a trivial pixel-wise operation.

$$\text{total\_gen\_loss} = \text{gan\_loss} + \lambda \cdot \text{l2\_loss} \tag{1}$$

$$\text{gan\_loss} = -reduce\_mean(\text{disc\_generated\_output}) \tag{2}$$

$$\text{l2\_loss} = reduce\_mean(vgg16(\text{target}) - vgg16(\text{gen\_output})) \tag{3}$$

The discriminator loss takes into account another two Wasserstein losses 6. A Wasserstein loss is calculated by multiplying the expected outcome by the predicted outcome. The real loss is the mean of the Wasserstein loss between the discriminator output for the target image (disc_real_output) and a tensor of ones (representing the real class), while the generated loss is the mean of the Wasserstein loss between the discriminator output for the generated image (disc_generated_output) and a tensor of zeros (representing the fake class). As Wasserstein loss is a product, when one of the tensors is zero, the loss is always zero, so we are basically saying to the model that the generated image is fake. Basically, we are trying to push the model predictions towards the target image and away from the generated image.

$$\text{real\_loss} = \text{reduce\_mean}(\text{disc\_real\_output} \cdot \text{ones}) \tag{4}$$

$$\text{generated\_loss} = \text{reduce\_mean}(\text{disc\_generated\_output} \cdot \text{zeros}) \tag{5}$$

$$\text{total\_disc\_loss} = reduce\_mean(\text{real\_loss} + \text{generated\_loss}) \tag{6}$$

The model was implemented in native Tensorflow with some high-level Keras layers. The generator and discriminator are trained jointly with two simultaneous Adam optimisers and for experimentation purposes, the gradients are computed for each sample - as the original Stochastic Gradient Descent (SGD) algorithm. This is an approach that causes updates to the parameters more frequently. We tested other batch sizes, but ended up with size 1. Regarding dataset splitting, we use 70% of the images for training, 15% for validation and 15% for a provisory test set. We emphasise that the actual test set was built in the preprocessing stage.

Table III shows the hyperparameters used in the model.

TABLE III: GAN Training Hyperparameters.

Dropout (Residual block)	0.5
Batch size	1
Optimiser	Adam
Learning rate	1e-4
$\beta 1$	0.9
$\beta 2$	0.999
ϵ	1e-8
Number of weight updates	5 · DATASET_SIZE

$\beta 1$: exponential decay rate for the 1st moment estimates in Adam optimiser.
$\beta 2$: exponential decay rate for the 2nd moment estimates in Adam optimiser.
ϵ: smallest difference between two numbers that is considered significant (avoid divisions by zero when the gradient is almost null).

GAN achieved a PSNR of 21.8703 dB and a SSIM of 0.4340 on the final test set. These metrics are discussed in section V.

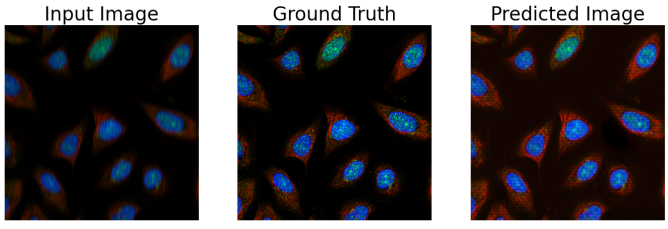

Fig. 6: Example of a generated image by the GAN model.

In Fig. 6, GAN prediction show a decent improvement in sharpness, as the organelle boundaries are more defined. However, there is a slight loss of detail inside the cells, when comparing with the reference image (in the middle).

C. MAXIM

Building a Deep Learning model for image deblurring from scratch is a challenging task. It requires designing a complex architecture with numerous layers capable of not only understanding image features but also discerning the specific patterns of blur. Then, we would need to train it on a massive dataset of blurred and deblurred image pairs, which demands significant computational resources and time. Fortunately, there are pre-trained models available that have been trained on large and diverse datasets and can be fine-tuned for our specific task of deblurring human protein images.

Fine-tuning is an approach where we take a pre-trained model with already computed weights and train it on a downstream task. Sometimes we only want to fine-tune the last layers, so we can freeze the first ones. This increases the training speed but may lead to a worse performance.

In our work, we chose to fine-tune the *maxim-s3-deblurring-reds* variant of the MAXIM architecture [12], trained on the REalistic and Diverse Scenes (REDS) dataset [13].

This model relies on an encoder-decoder backbone as its foundation (Fig. 7), following the design principles that originated with U-Net [14].

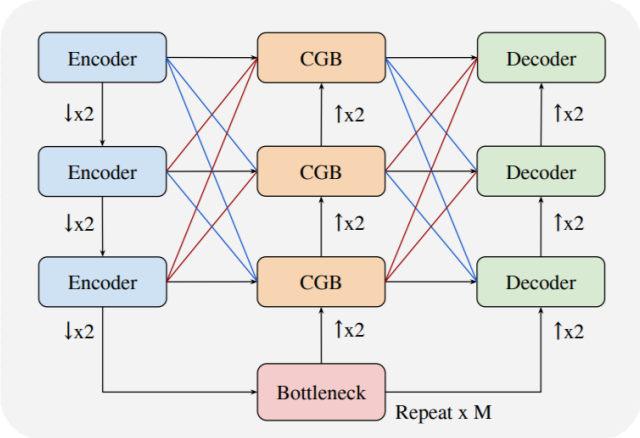

Fig. 7: Encoder-decoder backbone: This is the foundation of the MAXIM architecture. It processes the image through an encoder (captures features) and a decoder (reconstructs the image). [12]

Within this backbone, each component – the encoders, decoders, and bottleneck – benefits from two key building blocks: the multi-axis gated Multi-Layer Perceptron (MLP) block and the residual channel attention block (Fig. 8).

The multi-axis gated MLP block allows the model to efficiently capture information across different spatial dimensions within an image. This extracted information is then refined by the residual channel attention block, which focuses on the most relevant channels for the task at hand.

MAXIM further enhances its capabilities through the incorporation of a cross-gating block (Fig. 9). This block serves as a more computationally efficient alternative to the commonly used cross-attention module found in Transformers. The key

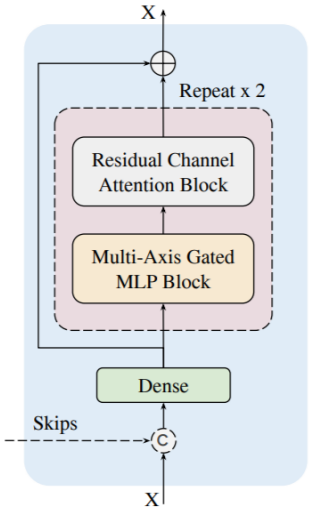

Fig. 8: Multi-Axis Gated MLP Block: This block is a key component within the encoder, decoder, and bottleneck of the MAXIM architecture. It efficiently mixes local and global visual information. [12]

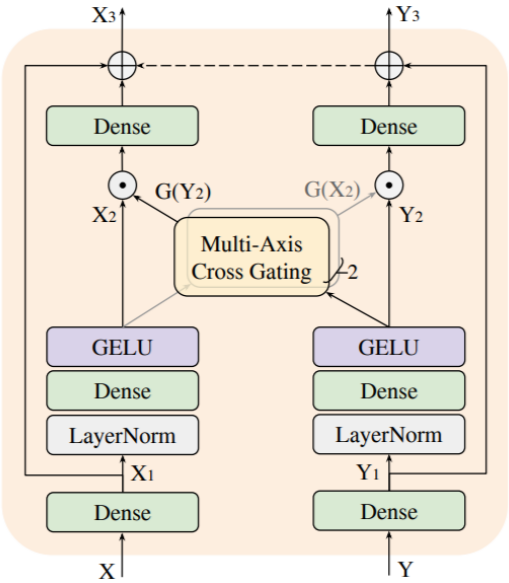

Fig. 9: Cross-Gating Block: This block refines the MAXIM architecture by allowing global features to control the flow of information through skip connections, enhancing the final output. [12]

advantage of the cross-gating block is that it relies solely on gated MLPs. This not only reduces the computational burden but also proves to be particularly beneficial for low-level image tasks like deblurring, where processing large amounts of data efficiently is critical.

The authors of the MAXIM paper compared their model to existing methods on a variety of benchmarks, including tasks like denoising, deraining, and deblurring (like the one we're interested in for protein images). It achieved state-of-the-art

performance across multiple different benchmarks, while still maintaining a low computational cost.

Since the model produces multi-stage outputs, we created a custom wrapper layer to encapsulate the pre-trained model and ensure only the final image output is used for training. During training, the weights of the first 3670 layers were frozen, essentially treating them as feature extractors. Only the final 50 layers were left trainable, allowing them to learn and adapt to our specific dataset. Generally, it is a good idea to leave only a small portion of the model trainable to improve training efficiency and reduce the risk of overfitting. To further guide the training process, we used 80% of the data for training and 20% for validation. For the training data, we employed perceptual loss calculated using a pre-trained VGG16 model in conjunction with the Adam optimiser and early stopping, mirroring the approach adopted for the autoencoder described in section IV-A.

The specific hyperparameters used are detailed in Table IV.

TABLE IV: MAXIM Training Hyperparameters.

Batch size	32
Optimiser	Adam
Learning rate	1e-3
Number of epochs	18

The results obtained by fine-tuning the MAXIM model are promising. In many images, the model successfully removes blur and generates clear and detailed protein structures, as shown in Fig. 10. However, for certain images, such as those containing large cytosol regions, the deblurring process may introduce artefacts or leave residual blur.

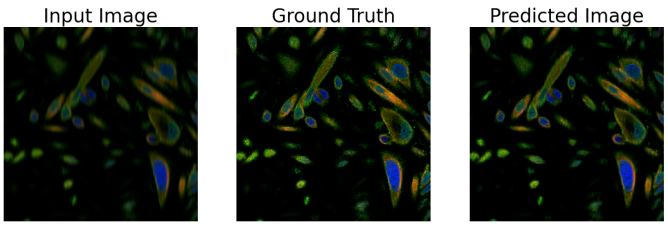

Fig. 10: Example of a generated image by the fine-tuned MAXIM model.

The model achieves a PSNR of 24.9807 dB and a SSIM of 0.6839 on the test dataset. This suggests that the model performs reasonably well in terms of deblurring protein structures and maintaining image fidelity.

V. Image quality comparison

Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) are two well-known image quality metrics that are widely used in the literature. PSNR is a measure of the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the quality of its representation. SSIM is a metric that quantifies the similarity between two images. The higher the PSNR and SSIM values, the better the quality of the image, in theory. The table below shows the PSNR and SSIM values for the images generated by a traditional 5x5 Wiener filter and the proposed alternatives.

TABLE V: Model comparison results.

Novelty And Contributions		
Model	**PSNR (dB)**	**SSIM**
Wiener (baseline)	5.6402	0.0361
DCAE	18.7227	0.2650
GAN	21.8703	0.4340
MAXIM	24.9807	0.6839

From the results, we conclude that all the implemented models were able to improve the quality of the images, comparing with the baseline method. The best model was MAXIM, which achieved simultaneously the highest PSNR and SSIM values. We see that the model complexity increase is consistent with the performance gains.

VI. Benchmark with Image Classification

To validate the deblurring models, we feed the predictions of the generated images into a XceptionNet model that was trained by a HPA competitor on the entire original sharp dataset [2]. For measuring the performance, we selected precision, recall and F1 score, because true negatives are not relevant for the problem, hence accuracy may be misleading. On the sharp version of our test set, the classifier achieved a precision of 72.74 %, a recall of 49.25 % and an F1 score of 58.75 %. These metrics represent an upper bound for the performance of the classifier and suggest that the prediction of 19 labels is a challenging task, On the blurred version, the classifier achieved a precision of 22.36 %, a recall of 14.40 % and an F1 score of 17.55 %. In blurry images, as the smooth edges are more difficult to detect (less false positives), the classifier's precision sometimes is higher than expected. Nonetheless, our main goal was to develop at least one solution that could outperform the input images (blurred) and a Wiener filter, which is still very used in state-of-the-art methods.

TABLE VI: Model comparison results on the classification task (metrics are indicated in %.)

Model	**Final Precision**	**Max. Precision (single class)**	**Recall**	**F1 Score**
Wiener (baseline)	25.55	39.57	24.97	25.21
DCAE	8.71	60.00	8.22	8.44
GAN	22.47	100.00	11.68	15.33
MAXIM	64.45	100.00	51.55	57.30

As Tab. VI illustrates, the MAXIM model was the most successful in terms of precision, recall and F1 score. With GAN, many labels were not correctly predicted, while a few were predicted with maximum or near-maximum precision. This result was quite surprising, due to the visual quality of the generated images, but can be explained by a loss of detail

[2]https://www.kaggle.com/code/bhamin/human-protein-atlas-xceptionnet-3-chs

inside the cells. Our first attempt, DCAE model, performed poorly in all metrics, except in specific labels. In summary, with our best model, we were able to improve the classifier's F1 score by 127.29 % comparing with the Wiener filter and by 226.50 % comparing with the blurry input images.

VII. Final Conclusions

This work highlights the difficult of developing from scratch models with generative capabilities that could possibly outperform approaches based on fine-tuning of pre-trained models. In fact, MAXIM was the most successful model in terms of PSNR and SSIM, but also in a downstream task. The GAN model, despite presenting visually accurate results from a human eye perspective, was not able to achieve the same level of detail as MAXIM. Lack of sample images for a model that is not so deep as a MAXIM and imperceptible changes of tone in the generated output may be possible reasons for GAN underperformance. Overall, there are some limitations that should also be addressed. First, the range of tested intensities and angles of motion blur could be increased. Second, the classifier used for benchmarking was implemented and trained by a third party, so we could not guarantee that the model was not overfitting. Third, the data provided to the deblurring models is synthesised. Real motion blur images could have different characteristics that may affect the performance of the models. Finally, regarding future applications, a continuous motion deblurring approach based on MAXIM could be proposed for slide-scanning systems, but the inference time and costs should be studied in future work.

VIII. Work load per student

Rafael Fernandes Gonçalves: 50%
Daniel Jorge Bernardo Ferreira: 50%

References

[1] W. Ouyang, C. F. Winsnes, M. Hjelmare, *et al.*, "Analysis of the human protein atlas image classification competition," *Nature Methods*, vol. 16, pp. 1254–1261, Dec 2019.

[2] M. J. Fanous and G. Popescu, "Ganscan: continuous scanning microscopy using deep learning deblurring," *Light: Science Applications*, vol. 11, Sep 2022.

[3] K. Zhang, W. Ren, W. Luo, W.-S. Lai, B. Stenger, M.-H. Yang, and H. Li, "Deep image deblurring: A survey," 2022.

[4] Q. Kang, J. Gao, K. Li, and Q. Lao, "Deblurring masked autoencoder is better recipe for ultrasound image recognition," 2023.

[5] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. Matas, "Deblurgan: Blind motion deblurring using conditional adversarial networks," 2018.

[6] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, p. 281–305, feb 2012.

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

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[10] A.-D. Nguyen, S. Choi, W. Kim, S. Ahn, J. Kim, and S. Lee, "Distribution padding in convolutional neural networks," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 4275–4279, 2019.

[11] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," 2018.

[12] Z. Tu, H. Talebi, H. Zhang, F. Yang, P. Milanfar, A. Bovik, and Y. Li, "Maxim: Multi-axis mlp for image processing," *CVPR*, 2022.

[13] S. Nah, S. Baik, S. Hong, G. Moon, S. Son, R. Timofte, and K. M. Lee, "Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1996–2005, 2019.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.