Towards robust fetus segmentation from MRI imaging: Accelerating annotations and diving into high semantic features

Jarod Lévy*
Decision and Bayesian Computation, USR 3756 (C3BI/DBC) & Neuroscience department CNRS UMR 3751, Institut Pasteur, Université de Paris, CNRS, Paris, France and Ecole des Mines de Paris - Promotion P20

Charlotte Godard and Jean-Baptiste Masson†
Decision and Bayesian Computation, USR 3756 (C3BI/DBC) & Neuroscience department CNRS UMR 3751, Institut Pasteur, Université de Paris, CNRS, Paris, France

Assembling and analyzing a full new database is a time consuming process. Often several months to few years can separate the constitution of the database, its annotation and the automated analysis. We present a pipeline dedicated to accelerating this process. We developed two different tools: a tablet application to accelerate data annotation and a meaningful features extraction pipeline through a self-supervised learning model using inpainting. These two tools were developed and tested on the LUMIERE Platform database composed of pregnancy MRI. Using MRI instead of ultrasound images for fetus analysis is a new approach and developing the two tools on this database could allow us on the one hand to understand more aspects of fetal development and on the other hand to constitute a reference atlas of fetus MRI. The work done for the LUMIERE database could be then adapted to any medical database.

I. INTRODUCTION

A. Fetus and MRI

Antenatal screening is used in order to identify abnormalities and to improve perinatal care. Congenital disorders are one of the leading causes of neonatal mortality worldwide. Thus, detecting and preventing those disorders are of the utmost importance.

Prenatal ultrasound (US) is usually carried out in two dimensions to check organs' presence, shape and global morphology. Nonetheless, there are several sources of complexities when using US such as the lack of standardization of the analysis, difficult technical conditions, mother health variability, "geometry" and differences in the medical doctor's skills.

Recently, fetal magnetic resonance imaging (MRI) emerged as an important tool for providing information about the developing fetus [1–3]. In situations where the ultrasound is unclear or if we need more accurate diagnosis of certain observations, magnetic resonance imaging could strongly reinforce robustness. MRI can be instrumental in fetus and mother care as it allows to obtain a full view of the fetus, have a higher sensitivity than ultrasound images and less user-dependency. The two imaging modalities, US and MRI are complementary [4], they could clarify diagnosis and optimize perinatal management. Nonetheless, MRI's benefits come with some challenges. Users have to deal with complex imaging and motion artefacts. Above all, as we are in the early stages of fetal research using MRI, we do not yet have an atlas to serve as reference point for research work.

Among the major initiatives leveraging MRI for fetus diagnosis, the LUMIERE Platform [5] located at the Necker Children's Hospital is one of the first hospital labs with an MRI dedicated to fetal research.
Here is a link to consult the full segmentation manually made by one of the LUMIERE's expert.
In order to accelerate the process of analyzing their data they decided to collaborate with the *Institut Pasteur Paris* to solve the following problematic : How to automate fetal MRI analysis ? We aim to analyze the data provided by the LUMIERE Platform using techniques such as segmentation to further increase the quality of medical diagnosis and to build a deeper understanding of the fetal development. By going to a sufficient degree of precision, these analyses could allow us to help the surgery decision in case of an intervention during pregnancy, perform virtopsia that could give the same amount of information as a real autopsy or even constitute a fetus MRI atlas.

B. The Project

With a new medical database, one can follow this general pipeline to entirely segment it:

- Separate the database in two groups, a small and a large one
- Extract meaningful features and annotate the data from the small group
- Feed a segmentation model with your extracted features, the annotated data and the large group

* jarod.levy@mines-paristech.fr
† jbmasson@pasteur.fr

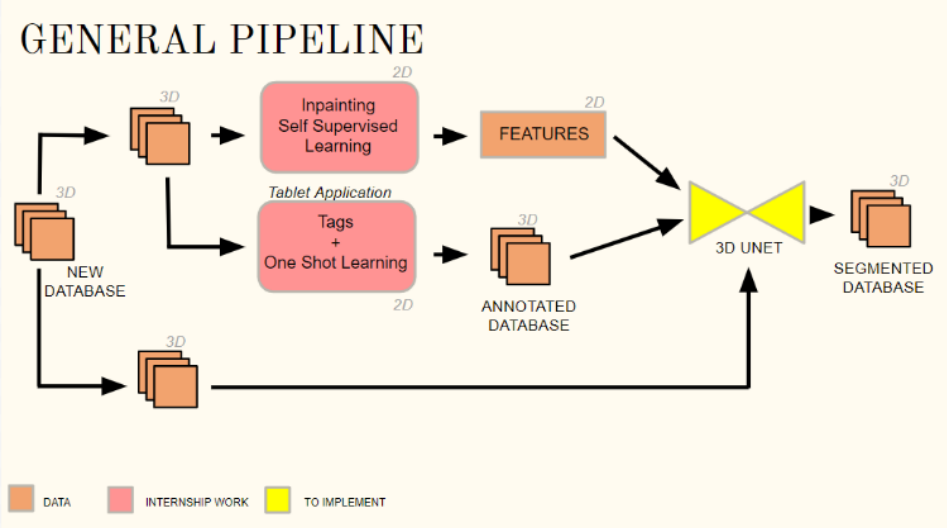


FIG. 1: **Representation of the general pipeline of a segmented database assembling from a raw MRI database.**

This pipeline is summarized in Figure 1. You end with a database fully segmented thanks to a limited amount of data.

The aim of this work is to go beyond this general pipeline by developing two different tools in order to obtain more easily the final segmentation. First, by accelerating the constitution of the annotated database thanks to a software tool that will ease the annotation work of the medical doctor. Secondly, by finding deep semantic features thanks to a self supervised learning task.

Creating a software product such as a tablet application that facilitates the process of annotating data could be beneficial for the medical doctor who would save a huge amount of time. The new software DIVA-cloud [6] developed by the *Institut Pasteur Paris* allows data annotations and learning in virtual reality [7]. It leverages recent initiatives and software developed to analyze data in VR [4, 8, 9]. Thanks to this tool, an inexperienced user can annotate complex data and get an initial segmentation of a structure of interest in a dozen seconds thanks to a one-shot learning implementation. Yet, we need our medical doctor partner to "proof" the annotation prior to beginning the deep learning phase. Due to the medical context, having medical doctors annotate data is challenging. It would thus be ideal to develop the app running on a digital tablet and allowing experts to quickly correct annotation using a stylus. This tool could be simply built on top of an already existing medical 3D images reader such as Napari [10], Ilastik [11] or Weka [12].

Regarding fetus image analysis, large databases won't be accessible for long. A few hundred MRIs are the maximum we can expect in the next few months. Hence, to devise a robust segmentation algorithm we also need to use self-supervised learning to provide improved features. We will focus our work on 2D inpainting pretext tasks because the latter is largely described in the litterature as a very efficient way to learn complex semantic features [13–15]. Using convolutional autoencoders, the latent representation learned by the inpainting task contains strong information for training a robust segmentation algorithm [16]. It is more convenient to start with 2D pretext tasks in order not to have memory issues. Indeed, the data from the LUMIERE Platform contain tens of 2D 512x512 pixels slides. Later on, we could simply reconstruct a 3D feature map from all the 2D maps.

II. AN ANNOTATION SOFTWARE

A. Aims of the tablet application

We aim to create a software able to accelerate the assembly of any medical database. Ideally, this software can run locally, on any computer and especially on a tablet. Implementation of one shot segmentation have been numerous in the literature [17–20] and have been extremely useful for our purpose. A similar software implementation has already been put in place in virtual reality (VR) [7]. Although our application does not use VR, we were largely inspired by this implementation. Here one of the main objectives is to appeal to basic reflexes while using our app : the expert simply draws with a stylus some annotations and can end with the segmentation of an object of interest.

Our application is composed of a reader that is able to load DICOM, NIFTI and TIF/TIFF files. Note that medical images are most often stored in the Digital Imaging and Communications in Medicine (DICOM). A radiologist-looking interface allows the user to switch

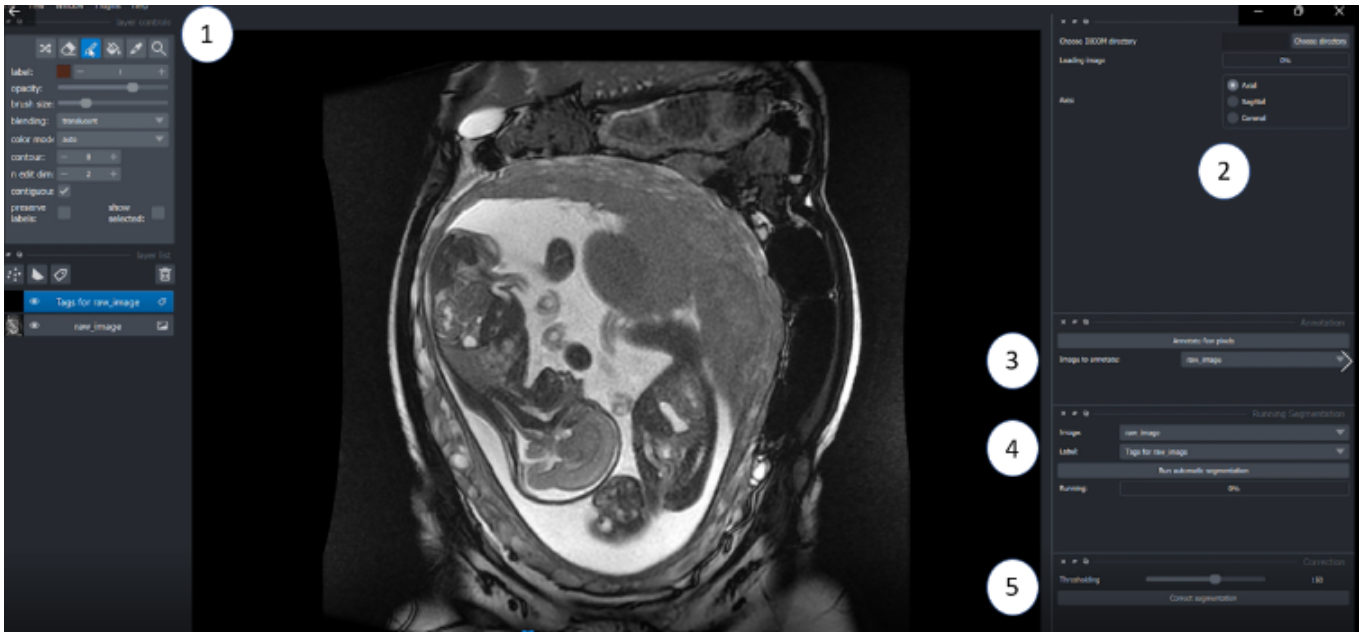
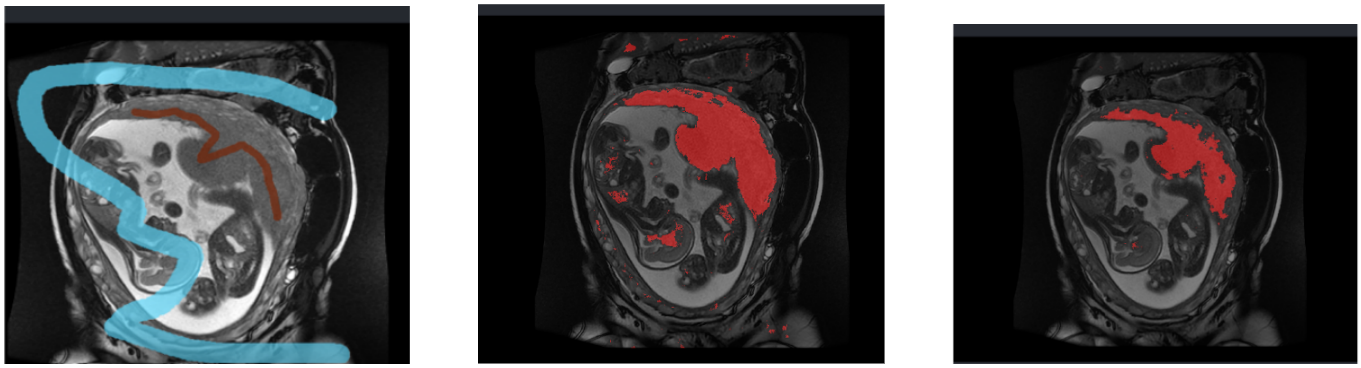


FIG. 2: **Main screen of the application.**
1- Load your file, 2- Change views, 3- Annotate, 4- Run segmentation, 5- Correct Segmentation



(a) Example of annotations on an MRI image from LUMIERE Platform

(b) Placenta segmentation result with threshold = 17.

(c) Placenta segmentation result with threshold = 180.

FIG. 3: **Output examples from the annotation application**
Different steps for placenta segmentation. In blue, annotation for the background pixels, in brown, annotation for placenta pixels.

from one medical view to another (coronal, axial, sagittal) and a simple contrast-enhancing method improves the visualization (Figure 2).

The user can overlay the data with an annotation layer of the same dimension as the MRI. On this layer, one can draw the two different classes he is interested in segmenting. Typically, the user will annotate with two colors: one for your object of interest and one for the background. In Figure 3.a, we can see this situation with the placenta as the object of interest.

After having annotated a few slides, we can run the main functionality of the application which is the one shot segmentation. Using the annotations, the model will infer on the other pixels and will output a segmentation of the object of interest (Figures 3.b and 3.c). Then, we can correct this first try by removing or adding annotations and re-iterate until we end with a satisfying segmentation. We can finally save the annotation and segmentation mask in TIFF format.

B. Napari

We used an existing medical reader Napari [10]. Napari is a multi-dimensional image viewer for Python. It is designed for browsing, annotating and analyzing large multi-dimensional images. It is built on top of Qt (for the GUI), vispy (for performant GPU-based rendering)

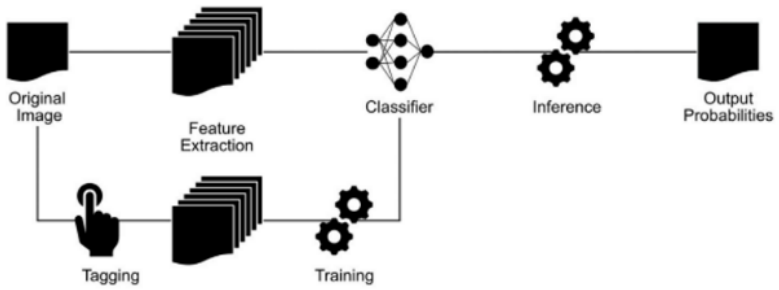


FIG. 4: **General pipeline of the one-shot segmentation procedure.**

FEATURES	KERNEL VALUE
Entropy	1 - 3
Standard Deviation	1 - 5
Gaussian Blur	3 - 5
Gradient	3 - 5
Laplacian	2 - 32
Maximum	1- 5 - 9
Minimum	1 - 5 - 9
Mean	1 - 5

(a) Features used

PARAMETERS	VALUE
Number of trees	50
Depth	50
Criterion	Gini
Max Features	Sqrt
Oob Score	True
Bootstrap	True
Resampling to have same amount of class in and out	True

(b) Random Forest Classifier parameters

TABLE I: **The one-shot segmentation model**

and the scientific Python stack (numpy & scipy). It is particularly convenient for our task as the option to overlay the raw data with annotations is already available in the raw version. In addition, Napari has implemented a slider that allows navigation through large 3D images which is ideal for the MRI from LUMIERE Platform. The user could quickly annotate a few stacks, run segmentation and see appearing fully segmented 3D images. On top of that, Napari is open-source, it can run everywhere and we can easily add functionalities to the raw version by simply adding python code to a notebook or even creating our own plugin.

C. One Shot Learning

Managing to perform quick segmentation to decrease the time spent on labeling datas is challenging. In our case, one-shot learning is especially appealing because it only requires one annotated example during the whole training stage and then could segment the unlabeled images. For our application, we need pixel-wise few-shot segmentation, that is to say : using the quick annotations of the medical doctor as labels and use them to identify to which class belongs the other pixels. Our objective is to exploit new data without prior information or a pre-trained model. This functionality was largely inspired from the DIVA Cloud implementation [7] where the procedure consists of rapid 3D tagging in VR, simple classifier training and inference on the entire dataset with iterative corrections performed within VR. In the updated version of the DIVA software, they implement a VR tagging functionality which allows voxel annotation. We try to adapt the pipeline created to pixel and 2D MRI slides. The complete procedure of the one shot learning functionality is described schematically in Figure 4.

Our application consists in accelerating data annotation using a simple one-shot learning procedure. We follow the same principles as those used in Ilastik [11], Weka [12] or behavior detection in larva [21], by tagging limited sets of data and stacking simple learners in order to train a collectively stronger classifier. We combine this model with a feature extraction procedure. Specifically, features are associated with tagged pixels, and learning is performed using robust classifiers in limited amounts of data. Furthermore, data tagging iterations allow the correction of anomalies in the learning to process the data. Whereas the DIVA software developed the quick annotation tool in 3D, here we focus on 2D features, we are working on pixels on each slide of the MRI. Each pixel contains a small subset of features [12]. It includes a wide variety of spatial filters (Gaussian, median, mean, etc.) with different kernel sizes for guaranteeing multiscale features. The kernel sizes used have to be roughly the scale of a typical medical object of interest, that is to say between 1 (typical tumor size) and 10 (typical organ size). Table I describes the different features and parameters used in the application model.

Basic classification approaches were used in the DIVA cloud implementation [7] since they are known to pro-

vide robust classification on small datasets, such as Random Forest Classification (RFC), Multi-Layer Perceptron (MLP), Gradient Boosting (XGB), Support Vector Machine (SVM), and Naive Bayes (NBC) as implemented in the Scikit-learn Python package [22]. Results show that the fastest and most efficient model is the RFC, so we build our model following this approach. The output of the inference is a probability segmentation map. We can set up a threshold value in order to be more or less severe in the discrimination of the pixels that belong to our class of interest. Figure 3.b and 3.c show the segmentation output with two different threshold values. The purpose of this segmentation is not to avoid any mistake but to be fast. Tags can be updated or erased if necessary. On the first run we will often end with a poor segmentation. Note that in this application, the usual problem of overfitting [23] is less present, as the goal is to annotate the data being explored and not to find a general learning scheme.

D. Results

Our analysis was performed on a Windows 10 based Ã64 system with an Intel i7-7700 CPU clocked at 3.60 GhZ, with 32 GB of RAM and an NVIDIA GeForce RTX 2080 Ti graphics card. Analysis scripts were coded in Python 3.7.

We share two videos to get a better understanding of the application functionalities. The first one is a presentation of the application : a file is loaded and then some annotations are drawn to perform placenta segmentation. The second one is an illustration of the segmentation functionality output.

Performance depends on the nature of the features extracted from the dataset. As features were designed to cover a large variety of patterns and scales, our method can be used in many additional applications and data types.

To evaluate the performance of our functionality, we compare the output of the segmentation with the same image fully tagged by an expert. We computed the Dice coefficient between our inferred probabilities and the expert segmentation. The Dice coefficient is computed on every threshold value and we kept only the best one. The goal is to end with a final segmentation much faster than the expert full tagging but with a Dice coefficient which remains acceptable. As we can notice on Figure 3.b when the threshold is low, there are many mistakes after the first run so there is a real need of iterating several times to end with the desired level of correct segmentation.

The average value of the dice coefficient on 10 different images is **0.56**. This value would not be very high in the case of a classical segmentation experiment but considering that we obtain this value by annotating only one 2D slide and without correction, we can easily approach a better value using iterating runs. The main result is the time saved for the annotation expert. The typical time spent for an expert from the LUMIERE Platform to segment entirely an image by using 3D Slicer (REFERENCES) is 7 minutes. With our application, we obtain a segmentation with the same precision (multiple runs to end with a Dice coefficient above 0.8) in only **3 minutes**.

E. Perspectives

We achieved to reach the goal of the application which was to accelerate the process of labeling data. The method developed in this application can be used for any type of medical images [7]. The number and nature of the features can be extended to capture specific properties of image stacks.

III. FEATURES EXTRACTION USING INPAINTING

After simplifying the medical doctor's work with our new software tool, we want to dive into high semantic features that characterize fetuses to reach our final purpose: segment an entirely new database with a little amount of data and robustness. In order to do so we will develop a self-supervised learning method using inpainting.

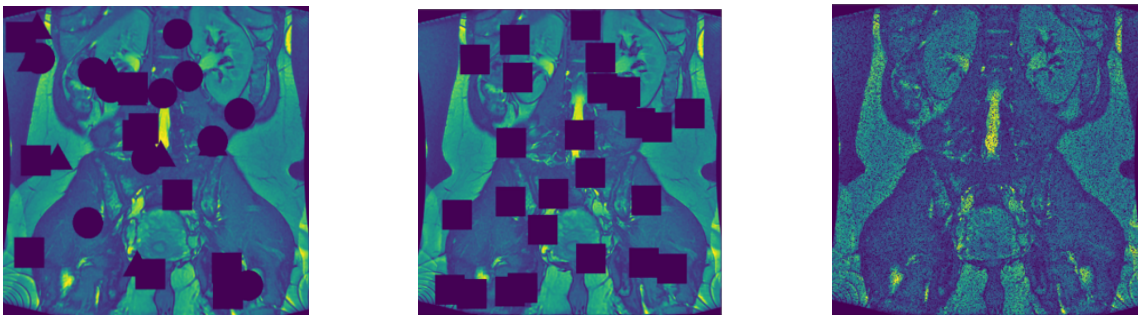
A. Self-supervised learning

Recently, there has been a particular interest in learning meaningful representations using weakly-supervised or self-supervised learning (SSL). SSL is learning from unlabeled data. It allows *the prediction of any unobserved or hidden part of the input from any observed or unhidden part of the input* [24].

In natural language processing, Collobert and Weston [25] were able to build a contextual language solver : a model able to retrieve a word between two known words in a sentence. They use a discriminative approach. The algorithm Word2Vec [26] solve the same kind of problem but formulate it as word prediction.

In computer vision, one of the first applications of SSL was to use the temporal information contained in videos. The consistency across temporal frames has been used to train models able to predict a frame between two known frames thanks to the learned embedding representation [27, 28]. Many efforts were also performed on images. Doersch et al. [29] use image patches and their relative positions to train an unsupervised deep feature representations. They seek to solve the discriminative task : is patch A above patch B or below ? If the model is able to answer this question, it has learned an embedding representation of the image context.

We will try to go further using inpainting by solving a pure prediction problem which is : what pixel intensities should go in the hole we created ? This approach



(a) Corrupted MRI (random shapes corruption).

(b) Corrupted MRI (classic hole corruption).

(c) Corrupted MRI (biased toss corruption).

FIG. 5: **Corrupted images from the LUMIERE Platform database.**

CORRUPTION TYPE	TRAINING LOSS	VALIDATION LOSS
Classic Hole	0.0021	0.0019
Biased Toss	0.0020	0.0020
Random Shapes	0.0025	0.0020

TABLE II: **Effects of corruption type on the model performance**

is challenging because we need to predict many real values per training example compared to NLP applications [25]. Moreover image pixel prediction is also harder to cheat since low-level image features do not provide any meaningful information.

There are many different SSL pretext tasks [30–34], our first step was to choose the most relevant one for our features extraction task. Although Jigsaw Puzzle techniques show interesting results in learning visual representations [30], we decided to focus on the inpainting task thanks to the studies already available in the literature [13, 16, 35] that shows substantial results when dealing with latent features representations. It seems that the inpainting pretext task allows a model to learn global, semantic meaningful features which seem ideal for our purpose to get a better understanding of the fetus body organization.

B. Inpainting

Image inpainting consists in rebuilding missing or damaged patches of an image. Typical applications of inpainting are used for old photos, painting restoration or image editing. For example, Photoshop has a powerful completion tool. An interesting point to notice is that convolutional neural networks (CNNs) perform better than humans on a classification task [36] but for inpainting results, those are visibly worse than human predictions. It is easy for us to mentally reconstruct a missing section of the image, we just compare the context with our knowledge of the world.

Several authors explored a large variety of techniques trying to compete with humans. Pathak et al. [13] build an hybrid L2-adversarial loss and emphasize the importance of Leaky ReLU in their results. A team of researchers from Google DeepMind propose a model [35] that is using a product of conditional probabilities. These probability functions are modeled by a convolutional neural network on neighboring pixels (Pixel CNN). In their paper, this method was thought to generate images but it can be applied to our task. Another interesting approach is [37]. Yang et al. propose a method to tackle inpainting of large sections on large images. They train two networks: a content and a texture network that minimizes local texture loss. They use neural matches to ensure that the texture inside the constructed crop is similar to the texture of the local context.

C. our Model

The efficiency of an inpainting model is dependent on numerous hyperparameters. Our aim was to devise a task that would capture the global characteristic of the foetus.

First, we dealt with the type of corruption to apply to the MRI images. We designed 3 different corruptors which are presented in Figure 5. The random shapes corruptor (Figure 5.a) is a combination of circles, triangles and squares randomly located in the image. The classical hole corruptor (Figure 5.b) is using multiple squares randomly inserted in the image. The biased toss corruptor (Figure 5.c) is using a binomial test for each pixel value to decide whether it will be black or not, you can set up the binomial parameter according to the percentage of the image you want to hide. We took the most complex corruptor that is to say the one that gives the highest

PATCH SIZE	TRAINING LOSS	VALIDATION LOSS	INPUT ARRAY SIZE
8x8	0.0013	0.0012	8 GB
11x11	0.0015	0.0014	10 GB
32x32	0.0018	0.0017	25 GB
50x50	0.0020	0.0020	40 GB
64x64	0.0040	0.0040	60 GB

TABLE III: **Effects of patch size on the model performance**

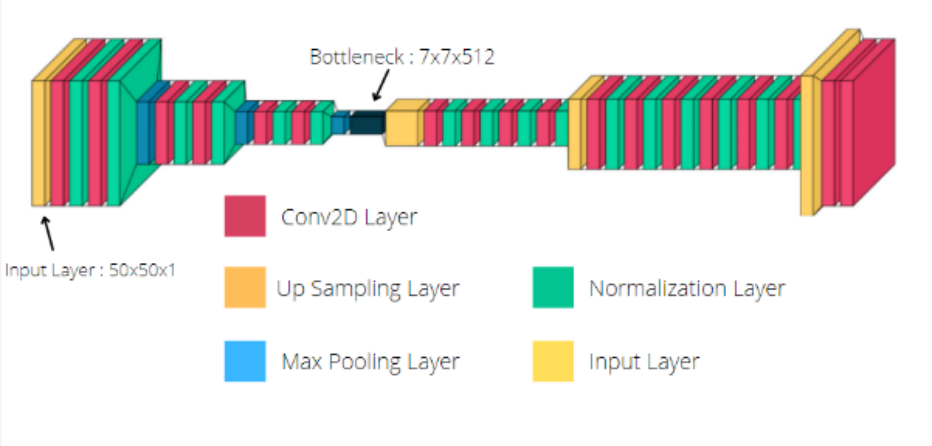


FIG. 6: **General architecture of the inpainting model.**

loss values (see Table II) when performing the training phase.

Another important parameter was the patch size. Here, we used 2D inpainting due to the size of the MRI stack that was incompatible with 3D analysis. We learn features representation on 2D slides of the MRI and we will then reconstruct the different slides in order to use the learnt representations in our final model. 2D slides were too large to feed efficiently a network. 2D slides from the LUMIERE Platform are 512x512 pixels. The patch size has to be sufficiently large to catch the fetus characteristics and sufficiently small to avoid memory issues. A good compromise was the 50x50 pixels patch (see Table III). We decided to patch the image and then corrupt the patches, not the other way around, we found that this decision does not influence the model performance.

The literature gives us the percentage of corruption needed [13, 16] to learn meaningful features. We have set this percentage at 20%. The limited amount of data prevented a larger percentage of corruption

We use a classical mean squared loss and the RMSProp optimizer from the Keras library.

Inpainting is part of a large set of image generation problems. Methods to solve those problems usually rely on autoencoders. Autoencoders are made of two networks: the encoder and the decoder. First, we encode the image into a lower-dimensional representation of an image (the latent space). Then, the decoder will work from this embedding to reconstruct the original image. This architecture forces the encoder to try to encode as much information as possible into the embedding. Since this is a small vector (a bottleneck), the encoder has to learn high-level, smarter features to compress the input with minimum loss. Those high-level features are hence more robust to noise and changes than raw pixels.

We started with a very simple model with a bottleneck of 16 filters using only convolutional layers and we modified the architecture iteratively after each performance improvement. For our final model, we took an image patch 50x50x1 (grayscale images) as input and a bottleneck of 7x7x512. Smaller one would not allow for enough space to encode the input information. Literature encourages us to use several small filters rather than bigger ones. Indeed, we can get the same receptive field with deeper networks. For the entire project, we only used filters of size 3. We tested size 5 and 7 and they always performed worse. We experimented with an encoder with five blocks composed of two 2D convolutional layers,a normalization layer after each of them and a pooling layer that doubles the number of filters (from 32 to 512). The decoder is a symmetric version of the encoder composed of 5 blocks with upsampling layers. Using batch normalization on each layer increases the performance with similar training times.

MODEL	NUMBER OF DATA	TRAINING LOSS	VALIDATION LOSS
Naive	80.000	0.0019	0.0024
Naive	300.000	0.0020	0.0020
Final	80.000	0.0015	0.0015
Final	300.000	0.0005	0.0005

TABLE IV: **Training and validation loss values according to number of data and the model used**
Naive is the model with only 16 filters and Final is our chosen architecture.

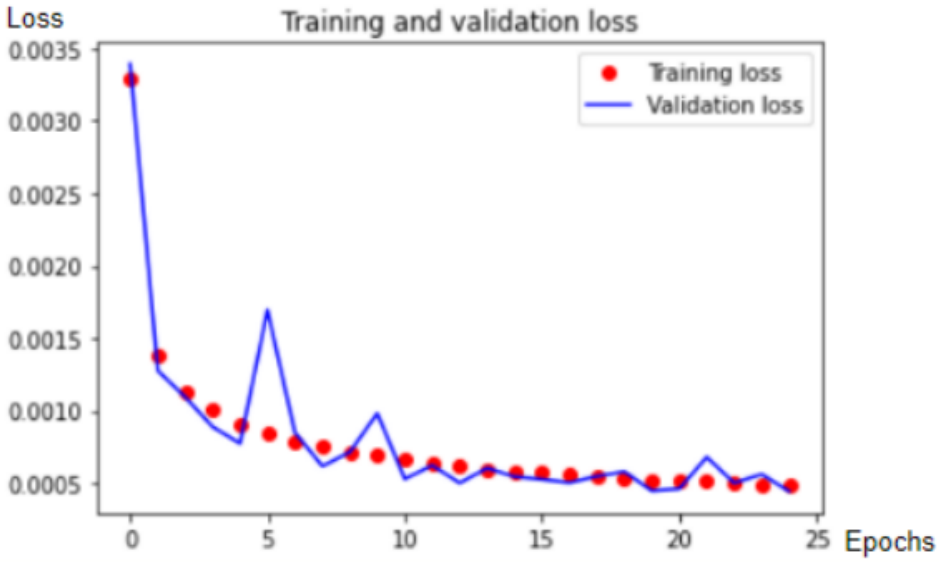


FIG. 7: **Training and validation loss graph.**

D. Results

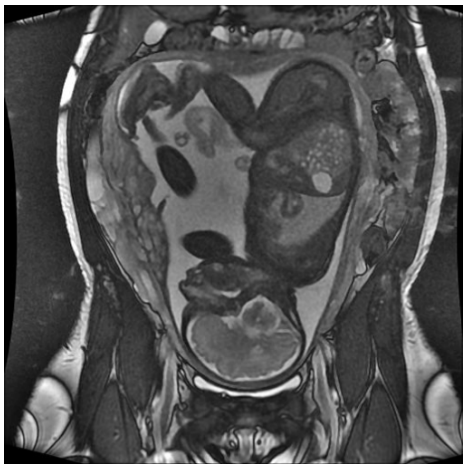
All the tests were performed on a secured Google Drive connecting to Google Colab Pro+ (50 GB RAM and GPU).

One last parameter was not discussed yet : the number of patches used for training, validating and testing. We begin our test with the naive autoencoder on 80.000 different patches, distributed among the 3 categories as follows : 70% for training, 20% for validation and 10% for test. We set up the batch size to 128 and the number of epochs to 100. Then we increase our number of data to 300 000 patches after having noticed overfitting. Finally, we replace the naive autoencoder by our final architecture to reach lower loss values. The results are summarized in Table IV. We notice that the loss values stagnate from 25 epochs and so we end with the following training and validation loss in Figure 7. Final value of training and validation loss is 0.0005.

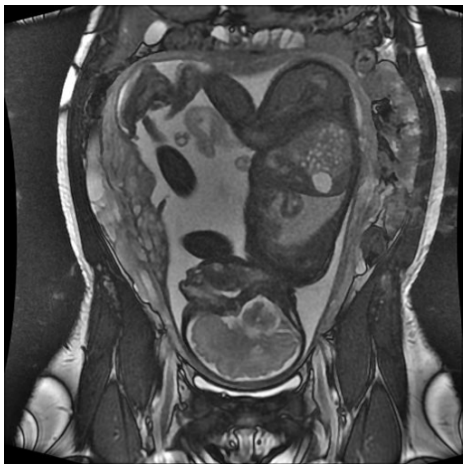
To be able to quantify the success of our implementation, we want to look at the reconstruction power of our inpainting model. We will compare the raw data from LUMIERE Platform with the reconstructed image inferred by our model on the corresponding corrupted patches that composes the image. In Figure 8, you can't see any differences between 8.a : the raw and 8.b : the reconstructed image. We use the Peak Signal to Noise Ratio to quantify the reconstruction power. It is a logarithmic metric that quantifies the differences between two images by using the mean squared error on each pixel value. We compute 3 different values of PSNR : one for the naive autoencoder, one for the architecture chosen in this task and a last one with a bottleneck of 1x1x1024 to check if the bottleneck depth add a substantial improvement. Results are summarized in Table V. With our model, we reach 34 dB. It is in the range of a satisfying reconstruction. [13] However, while the power of reconstruction ensures that our inpainting task works, we are more interested in how relevant the features learned are. Are they really correlated to the fetus body organization? We need to end our task verification with results that can guarantee the usefulness of the bottleneck representation for our final segmentation task.

MODEL	PSNR
Naive	30 dB
Final	34 dB
Deeper (1x1x1024)	34 dB

TABLE V: **Peak Signal to Noise Ratio results**



(a) Raw data from LUMIERE Platform



(b) Reconstructed data through the inpainting model.

FIG. 8: **Raw and reconstructed MRI image.**

MODEL	SVM ACCURACY	RFC ACCURACY
NO-FEATURES PATCH	59%	61%
LATENT REPRESENTATION	75%	87%

TABLE VI: **Fetus or non fetus classifier accuracy**

E. Information stored in the latent representation

We checked that our learned latent representation was meaningful for fetus segmentation so we decided to develop two different verifications. The first one is dimensionality reduction with the t-SNE method and the second one is a fetus or non fetus classifier with our bottleneck representation as input.

1. t-SNE visualisation

t-SNE is a method to visualize high-dimensional data. It converts similarities between data points in high dimension to joint probabilities and tries to minimize the divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. For our task, we manually labeled all the patches to keep track of whether the patch represents a part of the fetus or not. We suppress all the background patches in order not to have a significant difference between patches (we want to only have mother or fetus parts). After extracting the output of our bottleneck representation, we linearize our data and we project them from 7x7x512 (a little more than 25000 dimensions) to 2 dimensions.

We can see the t-SNE output in Figure 9. One can note a tendency as the yellow patches representing the fetus patches are close to each other in the projection dimension. However, we only used 121 patches as manually label patches is time consuming. Moreover the fetus patches do not form a distinct and clear cluster separated from the rest of the patches so this result is not sufficient to conclude on the quality of the learned features.
Due to lack of time, we did not test a UMAP projection.

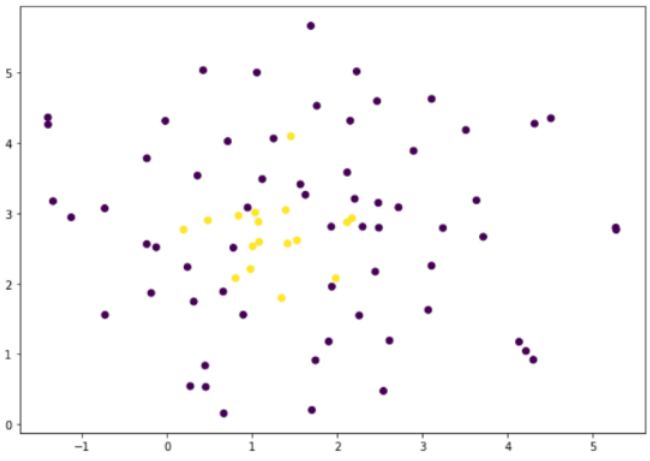


FIG. 9: **t-SNE Visualization of the latent representation from the inpainting model.** In yellow, the fetus part patches, in purple the mother part patches, number of patches = 121

2. Simple fetus classifier

Another idea was to build a linear classifier and take as inputs the output of the bottleneck from our inpainting

model. We can manually label each patch that represents a fetus part and take a look at the final accuracy. If this latter is high, we can conclude that the learnt latent representation helps to discriminate fetus part from non fetus part and thus, understand its general organization.

We perform this method on 1147 patches with a balanced number of fetus and non fetus patches. We test two different models : support vector machine and random forest classifier from the scikit-learn library [22]. In order to have a model for comparison, we apply the same procedure to patches of the same dimension but without features.

Table VI describes the results obtained. With our latent representation the accuracy of the random forest classifier reaches **87% of accuracy** versus only 61% for the no-features representation. This last result is much more potent than the t-SNE visualization. Thanks to it, we can conclude that our features seem to characterize, at least in part, the organization of the fetus. We suspect that by increasing the number of patches we could have exceeded 90% of accuracy but again the patches labeling is very time consuming.

F. Perspectives

We were able to build an inpainting model that works efficiently. The model was simple but efficient: extract features of interest to characterize the fetus. Thanks to fetus or non fetus classifier that confirms these features describe the organization of the fetus, we will be able to integrate those features in a segmentation model. Thanks to this representation, we will only require a limited amount of annotated data to have a robust final segmentation.

Nevertheless, it would be interesting to go further in our model by using a more complex loss such as an hybrid one (L2+adversarial loss [13]). One could also have thought to use a diffusion model [38] which would have given us a completely different latent representation.

IV. CONCLUSION

We achieve the two purposes of this study. We first accelerate the constitution of an annotated database by developing a software interface destinated to medical doctor experts. This interface can be used without previous knowledge on any computer. Then, we develop a self-supervised model using inpainting in order to extract meaningful fetus features. These extracted features will allow us to only use a limited amount of annotated data for the future segmentation of the database. Thus, we automate the analysis of the LUMIERE Platform MRI database. As the inpainting pipeline could be re-used for other medical data type, after having build the segmentation network (typically a UNET architecture), one can use the two tools developed on one part of any new medical database, mix it with the other non-analyzed part and perform partial or total segmentation in a faster way.

Acknowledgments. The author would like to thank Charlotte Godard and Robin Cremese for their precious supervision. He also would like to express gratitude to Jean-Baptiste Masson for his advice and warm welcome. More generally, he woud like to thank each member of the decision and bayesian computation team for their welcome.

Quick note on the internship The article describes the work done during a two months internship. The author is aware that some explanations do not seem to be well detailed or documented. Some arguments lack scientific rigor. The author has followed the line to work on two very different tasks and he apologizes if sometimes the quality of the scientific demonstration is affected.

Conflicts of interest. The author declare to have no financial or non-financial conflicts of interest.

[1] Ali Gholipour, Judith A. Estroff, Carol E. Barnewolt, Richard L. Robertson, P. Ellen Grant, Borjan Gagoski, Simon K. Warfield, Onur Afacan, Susan A. Connolly, Jeffrey J. Neil, Adam Wolfberg, and Robert V. Mulkern. Fetal MRI: A Technical Update with Educational Aspirations. *Concepts in magnetic resonance. Part A, Bridging education and research*, 43(6):237–266, November 2014.

[2] Iman A. Hosny and Hamed S. Elghawabi. Ultrafast MRI of the fetus: an increasingly important tool in prenatal diagnosis of congenital anomalies. *Magnetic Resonance Imaging*, 28(10):1431–1439, 2010.

[3] Agustin M. Cardenas, Matthew T. Whitehead, and Dorothy I. Bulas. Fetal Neuroimaging Update. *Seminars in Pediatric Neurology*, 33:100801, 2020.

[4] Jebrane Bouaoud, Mohamed El Beheiry, Eve Jablon, Thomas Schouman, Chloé Bertolus, Arnaud Picard, Jean-Baptiste Masson, and Roman H. Khonsari. DIVA, a 3D virtual reality platform, improves undergraduate craniofacial trauma education. *Journal of Stomatology, Oral and Maxillofacial Surgery*, 122(4):367–371, September 2021.

[5] http://fondation-lumiere.org/.

[6] Mohamed El Beheiry, Charlotte Godard, Clément Caporal, Valentin Marcon, Cécilia Ostertag, Oumaima Sliti, Sébastien Doutreligne, Stéphane Fournier, Bassam Hajj, Maxime Dahan, and Jean-Baptiste Masson. DIVA: Natural Navigation Inside 3D Images Using Virtual Reality. *Journal of Molecular Biology*, 432(16):4745–4749, 2020.

[7] Corentin Guérinot, Valentin Marcon, Charlotte Godard, Thomas Blanc, Hippolyte Verdier, Guillaume Planchon, Francesca Raimondi, Nathalie Boddaert, Mariana Alonso, Kurt Sailor, Pierre-Marie Lledo, Bassam Hajj, Mohamed El Beheiry, and Jean-Baptiste Masson. New Approach to Accelerated Image Annotation by Leveraging Virtual Reality and Cloud Computing. *Frontiers in Bioinformatics*, 1, 2022.

[8] Thomas Blanc, Mohamed El Beheiry, Clément Caporal, Jean-Baptiste Masson, and Bassam Hajj. Genuage: visualize and analyze multidimensional single-molecule point cloud data in virtual reality. *Nature Methods*, 17(11):1100–1102, November 2020.

[9] Thomas Blanc, Hippolyte Verdier, Louise Regnier, Guillaume Planchon, Corentin Guérinot, Mohamed El Beheiry, Jean-Baptiste Masson, and Bassam Hajj. Towards Human in the Loop Analysis of Complex Point Clouds: Advanced Visualizations, Quantifications, and Communication Features in Virtual Reality. *Frontiers in Bioinformatics*, 1, 2022.

[10] https://napari.org.

[11] Christoph Sommer, Christoph Straehle, Ullrich Kothe, and Fred A. Hamprecht. ilastik: Interactive Learning and Segmentation, 2011.

[12] Ignacio Arganda-Carreras, Verena Kaynig, Curtis Rueden, Kevin W Eliceiri, Johannes Schindelin, Albert Cardona, and H Sebastian Seung. Trainable Weka Segmentation: a machine learning tool for microscopy pixel classification. *Bioinformatics*, 33(15):2424–2426, August 2017.

[13] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context Encoders: Feature Learning by Inpainting. *arXiv:1604.07379 [cs]*, November 2016. arXiv: 1604.07379.

[14] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollar, and Ross Girshick. Masked Autoencoders Are Scalable Vision Learners. *arXiv:2111.06377 [cs]*, December 2021. arXiv: 2111.06377.

[15] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas Huang. Free-Form Image Inpainting with Gated Convolution. *arXiv:1806.03589 [cs]*, October 2019. arXiv: 1806.03589.

[16] Charles Burlin, Yoann Le Calonnec, and Louis Duperier. Deep image inpainting, 2017.

[17] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-Shot Video Object Segmentation. *arXiv:1611.05198 [cs]*, April 2017. arXiv: 1611.05198.

[18] Amirreza Shaban, Shray Bansal, Zhen Liu, Irfan Essa, and Byron Boots. One-Shot Learning for Semantic Segmentation. *arXiv:1709.03410 [cs]*, September 2017. arXiv: 1709.03410.

[19] Xiaolin Zhang, Yunchao Wei, Yi Yang, and Thomas S. Huang. SG-One: Similarity Guidance Network for One-Shot Semantic Segmentation. *IEEE Transactions on Cybernetics*, 50(9):3855–3865, September 2020. Conference Name: IEEE Transactions on Cybernetics.

[20] Kaixin Wang, Jun Hao Liew, Yingtian Zou, Daquan Zhou, and Jiashi Feng. PANet: Few-Shot Image Semantic Segmentation With Prototype Alignment. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9196–9205, Seoul, Korea (South), October 2019. IEEE.

[21] Jean-Baptiste Masson, Francois Laurent, Albert Cardona, Chloé Barré, Nicolas Skatchkovsky, Marta Zlatic, and Tihana Jovanic. Identifying neural substrates of competitive interactions and sequence transitions during mechanosensory responses in Drosophila. *PLOS Genetics*, 16(2):e1008589, February 2020. Publisher: Public Library of Science.

[22] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Muller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Ãdouard Duchesnay. Scikit-learn: Machine Learning in Python. *arXiv:1201.0490 [cs]*, June 2018. arXiv: 1201.0490.

[23] YS Abu-Mostafa, M Magdon-Ismail, and HT Lin. Overfitting. In *Learning from data: A short course*, pages 145–150. AMLBook, 2012.

[24] Yann le C.- https://ai.facebook.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/.

[25] Ronan Collobert and Jason Weston. A Uniï¬¬*edArchitectureforNaturalLanguageProcessing* : *DeepNeuralNetworkswithMultitaskLearning.page* 8.

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[27] Ross Goroshin, Joan Bruna, Jonathan Tompson, David Eigen, and Yann LeCun. Unsupervised Learning of Spatiotemporally Coherent Metrics. *arXiv:1412.6056 [cs]*,

September 2015. arXiv: 1412.6056.

[28] Vignesh Ramanathan, Kevin Tang, Greg Mori, and Li Fei-Fei. Learning Temporal Embeddings for Complex Video Analysis. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4471–4479, Santiago, Chile, December 2015. IEEE.

[29] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised Visual Representation Learning by Context Prediction. *arXiv:1505.05192 [cs]*, January 2016. arXiv: 1505.05192.

[30] Mehdi Noroozi and Paolo Favaro. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. *arXiv:1603.09246 [cs]*, August 2017. arXiv: 1603.09246.

[31] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a Proxy Task for Visual Understanding. *arXiv:1703.04044 [cs]*, August 2017. arXiv: 1703.04044.

[32] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised Representation Learning by Predicting Image Rotations. *arXiv:1803.07728 [cs]*, March 2018. arXiv: 1803.07728.

[33] Aiham Taleb, Winfried Loetzsch, Noel Danz, Julius Severin, Thomas Gaertner, Benjamin Bergner, and Christoph Lippert. 3D Self-Supervised Methods for Medical Imaging. In *Advances in Neural Information Processing Systems*, volume 33, pages 18158–18172. Curran Associates, Inc., 2020.

[34] Hannah Spitzer, Kai Kiwitz, Katrin Amunts, Stefan Harmeling, and Timo Dickscheid. Improving Cytoarchitectonic Segmentation of Human Brain Areas with Self-supervised Siamese Networks. *arXiv:1806.05104 [cs]*, June 2018. arXiv: 1806.05104.

[35] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional Image Generation with PixelCNN Decoders. *arXiv:1606.05328 [cs]*, June 2016. arXiv: 1606.05328.

[36] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *arXiv:1602.07261 [cs]*, August 2016. arXiv: 1602.07261.

[37] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. High-Resolution Image Inpainting using Multi-Scale Neural Patch Synthesis. *arXiv:1611.09969 [cs]*, April 2017. arXiv: 1611.09969.

[38] Fazil Altinel, Mete Ozay, and Takayuki Okatani. Deep Structured Energy-Based Image Inpainting. *arXiv:1801.07939 [cs]*, August 2018. arXiv: 1801.07939.