AN ALTERNATIVE WAY TO PREDICT BIOMETRICS BASED ON REGRESSION CNN AND ITS EXPLAINABILITY

A PREPRINT

David S. Hippocampus*
Department of Computer Science
Cranberry-Lemon University
Pittsburgh, PA 15213
hippo@cs.cranberry-lemon.edu

Elias D. Striatum
Department of Electrical Engineering
Mount-Sheikh University
Santa Narimana, Levand
stariate@ee.mount-sheikh.edu

October , 2021

ABSTRACT

In clinical medicine, doctors need to judge the patient's physical condition based on specific biometrics. However, due to the tight doctor-patient ratio, doctors cannot manually make accurate measurements and judgments for every disease of every patient. Therefore, various equipment and technologies based on computer-aided design came into being. Among them, medical image processing technology based on deep learning occupies the mainstream today because of its high efficiency and accuracy. Some biometrics are obtained indirectly through calculation results. For instance, the fetal head circumference and cardiac diseases. The measurements of fetal head circumference (HC) is a key biometric to monitor fetus growth during pregnancy. This biometric is measured on ultrasound images via segmentation and fitting of an ellipse. In this paper, we propose an alternative regression CNN to directly measure the head circumference without segmentation results. Our method is performed on the public HC18 dataset. We compare the prediction results with the segmentation based results. Besides, we utilise regression CNNs to predict the volume of cardiac structures directly. And this is performed on the public ACDC dataset. In order to prove the explainability of networks, we use several analysis methods to interpret the networks.

***K*eywords** Medical image · Biometrics · Segmentation · Regression · Explainability

1 Introduction

Medical image processing technology based on deep learning occupies the mainstream today because of its high efficiency and accuracy, such as medical image reconstruction, medical image enhancement, medical image segmentation, medical image registration, etc. [1]. However, these technologies are for doctors to make better decisions. Thus in this paper, we propose an idea which is an alternative way to directly predict biometrics using regression CNN. We valid our idea on two applications, one is head circumference prediction, the other one is volume of cardiac structures prediction.

1.1 Head circumference prediction

Automated measurement of fetal head circumference (HC) is performed throughout the pregnancy as a key biometric to monitor fetus growth and estimate gestational age. In clinical routine, this measurement is performed on ultrasound (US) images, via manually tracing of the skull contour or fitting it to an ellipse. Indeed, identifying the head contour is challenging due to low signal-to-noise ratio in US images, and also because the contours have fuzzy (and sometimes missing) borders (Fig. 1). Manual contouring is an operator-dependant operation, prone to intra and inter-variability, which provokes inaccurate measurements [2]. More precisely, the 95% limits of agreement are $\pm$7mm for the intra-operator variability and $\pm$12mm for the inter-operator variability [2, Tab. 1 p. 272]. Several approaches have been

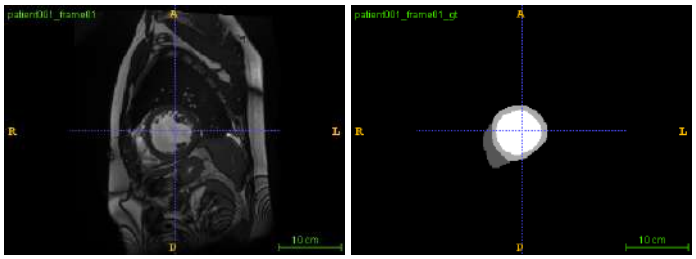
*Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

Figure 1: Ultrasound images of fetal head from [3]. Corresponding head circumference (HC) is displayed in millimeters and pixels.

proposed in the literature to measure the head circumference in US images, based on image segmentation [4, 5, 6]. Usually they follow at two-step approach, namely fetal head localization and segmentation refinement. In [7], the first step consists in locating the fetal head via machine learning, with Haar-like features used to train a random forest classifier; and the second step consists in the measurement of the HC, via ellipse fitting and Hough transform. Similar method is used in [4]. Other approaches build upon deep segmentation models also in a two-step process, prediction and ellipse fitting [8]. In [9], the standard segmentation model U-Net [10] is trained using manually labeled images, and segmentation results are fitted to ellipses. In [11], authors build upon the same idea, combining image segmentation and ellipse tuning together in a multi-task network.

1.2 Cardiac structure prediction

Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year according to statistics released by the World Health Organization. Therefore, effective diagnosis and treatment are very necessary. Medical image analysis is an important link for finding the lesion and classifying diseases. In the public "Automatic Cardiac Diagnosis Challenge" dataset (ACDC) [12], there are 3 cardiac structures, namely left ventricular (LV), myocardium (MYO), right ventricle (RV). And 4 types of disease, previous myocardial infarction (MINF), dilated cardiomyopathy (DCM), hypertrophic cardiomyopathy (HCM), abnormal right ventricle (RV), which are diagnosed by computing the volume of each structure of a heart, then judge them if the volume belongs to normal range. Fig. 2 is the MRI image of a patient and its ground truth annotated by experts.



(a) MRI image of a patient (b) Annotations of a patient

Figure 2: MRI cardiac image and its ground truth annotated by clinical expert. The white color in (b) is left ventricle (295.5078), the light grey is myocardium (164.2578 ml), the dark grey is right ventricle (139.7217 ml).

In literature, there are two kinds of methods to predict the volume of cardiac structures. One is segmentation based methods, then calculating the volume through segmented areas multiplying thickness of each slice. In ACDC challenge, almost all the deep learning architectures are used in segmentation contest [13, 14, 15, 16, 17, 18, 19, 20, 21]. The other one is direct predict the volume. In [22], authors propose using machine learning techniques including both handcrafted feature extraction and regression to directly estimate cardiac ventricular volumes without segmentation. Afterwards, they utilise CNN and random forests to automatically predict volume directly [23].

1.3 Explainability of regression CNN

It is known that the high accuracy of deep learning methods comes at the cost of a low interpretability, i.e. the model is seen as a black box, which does not provide explanations along with the prediction. In this paper, our goal is to investigate how explanation methods can help us to get some insights on the regression network and to appreciate its

behavior [24]. Therefore, Explainable AI (XAI) techniques has come up as a branch of AI to make AI more transparent. In classification networks, explanations may take the form of saliency or sensitivity maps [25], highlighting the areas that particularly contributed to a decision. Explanations are especially crucial for task such as survival prediction, stage classification or biometrics estimation, i.e. when the decision is taken outside the image domain. The saliency maps have been applied on different neural networks such as CNN, LSTM, and in various tasks, for example classification, detection and image segmentation [26]. Compared to a classification model, the last layer of a regression CNN model is a linear or sigmoid activation function, instead of the softmax layer. Also, the regression loss function is metric-inspired, for instance, it can be the Mean Absolute Error (MAE) or the Mean Squared Error (MSE).

The rest of the paper is organized as follows. Section 2 designed and performed experiments about head circumference prediction based on segmentation. Section 3 describes the proposed regression CNN architecture and the loss functions as well as experiments. In Section 4, we describe the explainability on regression CNN of head circumference prediction. Section 5 is another application of regression CNN on ACDC dataset. The conclusion and future works are drawn in Section 6.

2 Head circumference prediction based on segmentation

2.1 Method

2.1.1 Segmentation CNN model

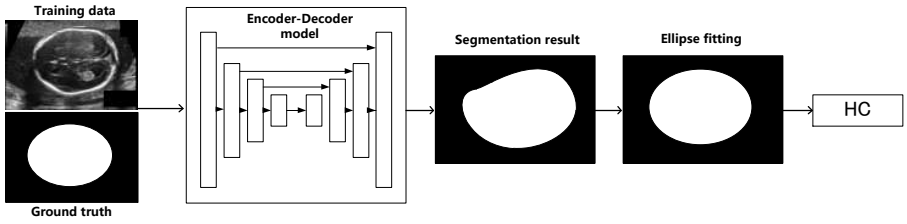


Figure 3: Overview of head circumference prediction process. Segmentation through U-Net model, then ellipse fitting, ellipse circumference is the HC value.

Given the training images and ground truth, we use U-Net [10] CNN model to train and predict the contour of fetus head. See Fig. 3. For the encoder part of U-Net, we can load different pretrained CNN architectures to improve the segmentation performance, e.g.VGG [27].

2.1.2 Loss functions

We use Dice loss [28], Kappa loss [29] and design a composite loss function composed of Focal loss [30] and Kappa loss with a hyper parameter λ. Because Focal loss is upgraded version of Cross entropy loss which is logarithmic based function, and Kappa loss is improved version of Dice loss which is metric based function. For the λ, it should put more weight on Kappa loss in segmentation task.

$$\text{Dice loss} = 1 - \frac{2\sum_{i=1}^{N} p_i g_i}{\sum_{i=1}^{N} p_i + \sum_{i=1}^{N} g_i} \tag{1}$$

$$\text{Kappa loss} = 1 - \frac{2\sum_{i=1}^{N} p_i g_i - \sum_{i=1}^{N} p_i \cdot \sum_{i=1}^{N} g_i / N}{\sum_{i=1}^{N} p_i + \sum_{i=1}^{N} g_i - 2\sum_{i=1}^{N} p_i g_i / N} \tag{2}$$

$$\text{Composite\_loss} = (1-\lambda) * \text{Focal} + \lambda * \text{Kappa} \tag{3}$$

2.1.3 Post processing

The predicted image usually include more than one component, see Fig. 4 (a). Post processing is necessary in order to fitting an ellipse. We extract the contour information in the image after binarization, and the vector will store the contour information, and each element in a vector is composed of continuous points which are contours of isolated areas. The number of elements in a vector stands for the number of contours. Then we only keep the largest area as predicted head contour, while the other small regions are performed with background color filling, just like Fig. 4 (b).

(a) Predicted image (b) Denoised image

Figure 4: Removing the isolated points

Figure 5: The process of ellipse fitting. Detecting the contour of segmented results and performing least square fitting on detected contour, a, b is two axes of an ellipse.

2.1.4 Ellipse fitting

The segmented results predicted by U-Net can not directly compute HC value. As Fig. 5 shown, they are first performed with contour detection (e.g, Canny detection), afterward, those contours are fitted into ellipses by least square fitting method. Then we use method Ramanujan approximation II [31] (Eq. 4 to calculate the circumference of ellipse which is also head circumference, a is long axis of an ellipse, the half distance of detected contour points between minimum point and maximum point in horizontal direction, b is short axis, the half distance of detected contour points between minimum point and maximum in vertical direction. $h = \frac{(a-b)^2}{(a+b)^2}$).

$$\text{HC} = \pi(a+b)(1 + \frac{3h}{10 + \sqrt{4-3h}}) \tag{4}$$

2.2 Experimental settings

2.2.1 HC18 dataset

We use the HC18 *training* dataset [3], that contains 999 US images acquired at varying times during the pregnancy, along with the corresponding fulfilled head ground truth[2]. We randomly split the dataset into a training (600), a validation (200) and a test set (199), except for the images that were made during one echographic examination, that are assigned the same set. Image preprocessing includes a resizing from 800 $\times$ 540 pixels to 224$\times$224, and normalization by subtracting the mean and dividing by standard deviation.

2.2.2 Data augmentation

As we know, medical images are often not sufficient to feed into deep neural networks. Therefore, the data augmentation techniques bring up the possibility of data diversity. In this work, the data type is ultrasound images of fetus head among three different trimesters supplied in one hospital. Except for the color and intensity that can be negligible in the

[2]The original ground truth of dataset are a set of ellipse contours, so we fill the inside of ellipse as foreground.

Table 1: 5 fold cross validation Original Performance of different loss functions and models (U-Net trained from scratch and U-Net loaded with pretrained VGG). Evaluation metrics are Dice Index (DI), Hausdorff Distance (HD) in pixels, Mean Absolute Error (MAE) in millimeters, Percentage MAE (PMAE).

	U-Net (train from scratch)				U-Net (VGG)			
Loss	DI	HD(px)	MAE(mm)	PMAE(%)	DI	HD(px)	MAE(mm)	PMAE(%)
Kappa	0.892±0.06	9.024±1.79	33.578±32.38	21.064±12.20	0.928±0.05	7.666±0.87	10.096±5.62	10.76±9.76
Dice	0.844±0.09	10.594±3.58	46.51±47.23	28.28±17.04	0.926±0.05	7.512±0.63	9.386±4.00	8.966±7.73
Focal+Kappa	0.906±0.05	7.992±0.79	19.968±7.64	14.734±8.26	0.934±0.04	7.586±0.76	10.968±4.47	10.574±9.10

images, they mainly have various positions and sizes. In order to expand and simulate the diversity of fetus head. We augment the data of training set to 1800 images from following two aspects:

- Position: observing the original dataset, the position of fetus head varies from one to another. Thus we perform rotation (10 degrees counterclockwise) and translation (40 pixels horizontally, 1/20 of the original image width and 27 pixels vertically, 1/20 of the original image height) on the images respectively. At the same time, we delete the augmented images whose heads are beyond the borders.

- Size: the fact of original images consists of different size of fetus head in three trimesters. Thus we first crop (left upper (100,50) righter downer (700,500)) the marginal non-fetus head area, then we resize the images to the original size (800,540), and we slightly move the position (5 pixels) to make them look different from before. At the same time, we delete the augmented images whose heads are beyond the borders.

2.2.3 experiment protocol

We use U-Net segmentation model to train the HC18 dataset, the U-Net has 1943761 trainable parameters and besides, the pretrained VGG [27] model was loaded in U-Net, which has 19027891 trainable parameters. Batch Normalization is included. Protocol is a 5-fold cross validation. The optimizer is Adam with a learning rate of $1e^{-3}$. The batch size is 16. Evaluation metrics for the fitted ellipse are the Dice index (DI) and the Hausdorff distance (HD), which is the maximum point-to-point distance between two contours. For HC value, we use mean absolute error (MAE) to compare the predicted HC and ground truth HC. Note that all the metrics are compared under the original image size (800,540). The implementation tool is based on Keras and Tensorflow 1.0.Tesla P100 GPU server is used in the experiments.

2.3 Experiment results

We trained 6 neural networks in this experiment, using different loss functions and U-Net with and without pretrained models. The learning curves are shown in Fig. 8. For the U-Net trained from scratch, it took about 4 hours and 28 minutes. For the U-Net loaded with pretrained VGG, it took about 8 hours and 8 minutes. We compare 3 loss functions, we can see that the Dice loss (in purple) and Kappa loss (in blue) has the similar stable performance, while the composite loss function in green color (Focal+Kappa loss) has unstable performance. In Fig. 9, the models trained from scratch are over fitting in valid stage, while the pretrained models perform well in training and valid stage.
The quantitative results are shown in Table 1. It's obvious that the U-Net with pretrained VGG models have better performance in each metrics. There are two types of evaluation metrics, one is segmentation metrics (Dice index, Hausdorff distance), the other one is clinical metrics (Mean absolute error, Percentage Mean absolute error). We can see that the segmentation metrics are satisfying, while the clinical metrics are not as good as the results of state of the art [32, 33, 34]. The key point that can affect the clinical metrics is the fitting process.
In testing stage, the predicted images vary from each other. For example, in Fig. 10, the segmentation result is quite close to the shape of ellipse, thus, there is only one fitted ellipse in an image, in this case, the HC value is therefore very close to the ground truth value. However, in bad segmentation result (Fig. 11), the foreground information is incomplete or discrete, which can be ambiguous to fitting, because in fitting process, the first step is to detect contour, so there will be several isolated contours, and corresponding number of ellipses are generated. Then the two axis of ellipse are far away from the ground truth. Unless we manually remove the unnecessary ellipses that are not belong to the ground truth, but this operation is undoubtedly a very time-consuming and tedious work.

Table 2: Original Best model Performance of different loss functions and models (U-Net trained from scratch and U-Net loaded with pretrained VGG) on the same test set. Evaluation metrics are Dice Index (DI), Hausdorff Distance (HD) in pixels, Mean Absolute Error (MAE) in millimeters, Percentage MAE (PMAE).

	U-Net (train from scratch)				U-Net (VGG)			
Loss	DI	HD(px)	MAE(mm)	PMAE(%)	DI	HD(px)	MAE(mm)	PMAE(%)
Kappa	0.93±0.09	7.08±1.55	9.31±15.82	6.69±13.19	0.95±0.10	6.91±1.50	4.52±9.98	4.14±12.38
Dice	0.94±0.10	7.14±1.67	7.61±10.79	6.48±14.03	0.96±0.08	6.79±1.13	3.96±7.70	3.43±9.72
Focal+Kappa	0.91±0.10	7.60±2.08	13.91±9.78	9.74±13.34	0.94±0.08	7.10±1.31	6.71±10.63	5.03±11.38

Table 3: Post-processed Best model Performance of different loss functions and models (U-Net trained from scratch and U-Net loaded with pretrained VGG) on the same test set. Evaluation metrics are Dice Index (DI), Hausdorff Distance (HD) in pixels, Mean Absolute Error (MAE) in millimeters, Percentage MAE (PMAE).

	U-Net (train from scratch)				U-Net (VGG)			
Loss	DI	HD(px)	MAE(mm)	PMAE(%)	DI	HD(px)	MAE(mm)	PMAE(%)
Kappa	0.93±0.09	6.97±1.57	8.41±7.84	5.89±10.18	0.95±0.11	6.87±1.50	4.55±7.25	3.89±10.79
Dice	0.94±0.09	7.00±1.84	7.6±8.03	5.68±11.21	0.96±0.08	6.69±1.01	4.19±5.01	3.1±7.65
Focal+Kappa	0.90±0.10	7.47±1.99	14.05±7.57	9.42±10.92	0.94±0.08	6.82±1.25	6.99±7.21	4.85±9.37

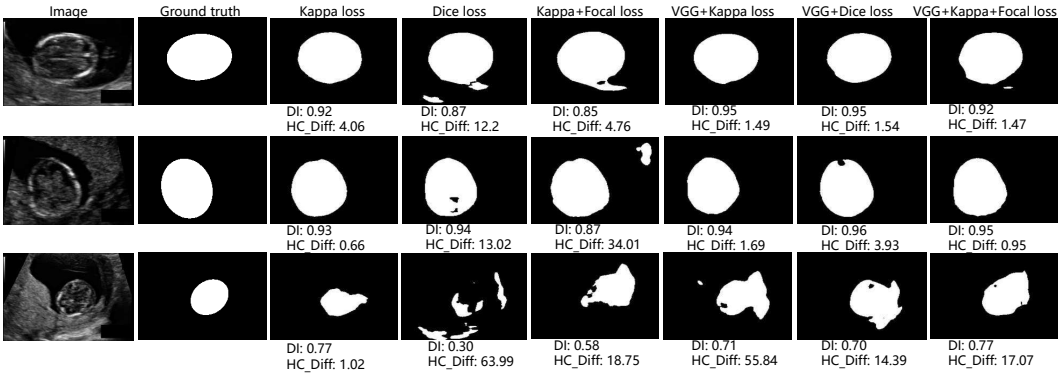


Figure 6: Segmentation instance under U-Net and U-Net loaded with pretrained VGG models and loss functions. Below images are evaluation metrics: Dice index (DI) and Difference HC value from ground truth (HC_Diff).

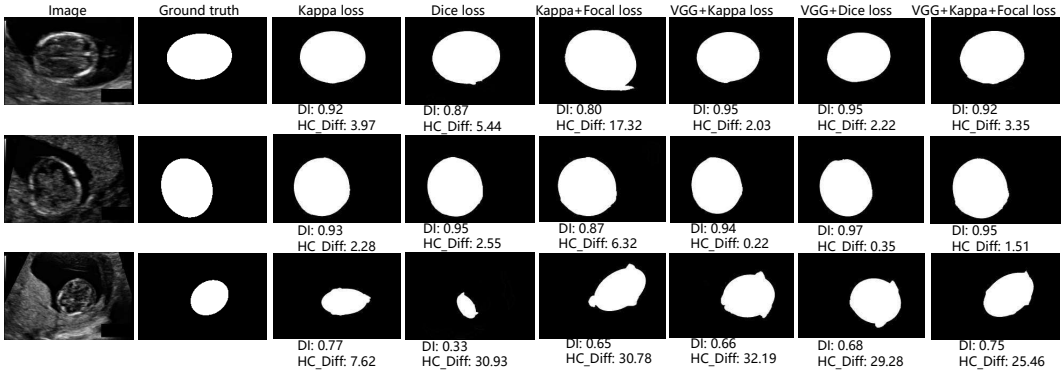
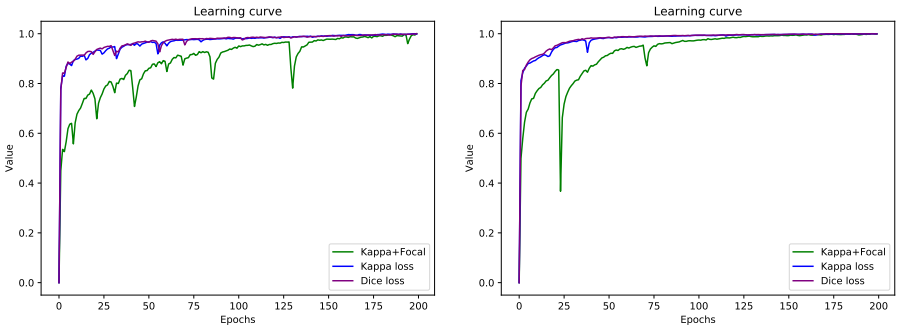
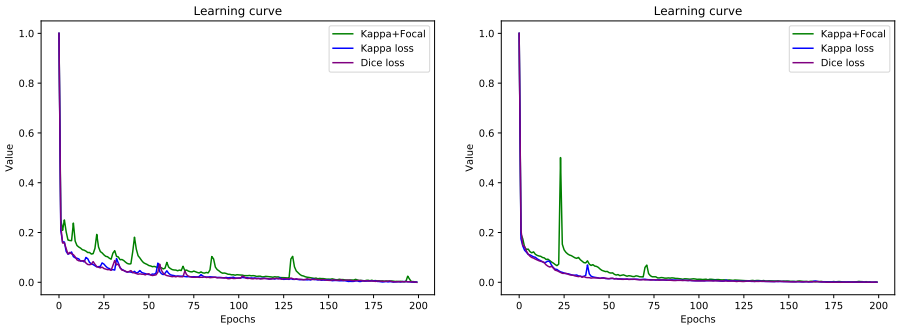


Figure 7: Post-processed segmentation instance under U-Net and U-Net loaded with pretrained VGG models and loss functions. Below images are evaluation metrics: Dice index (DI) and Difference HC value from ground truth (HC_Diff).

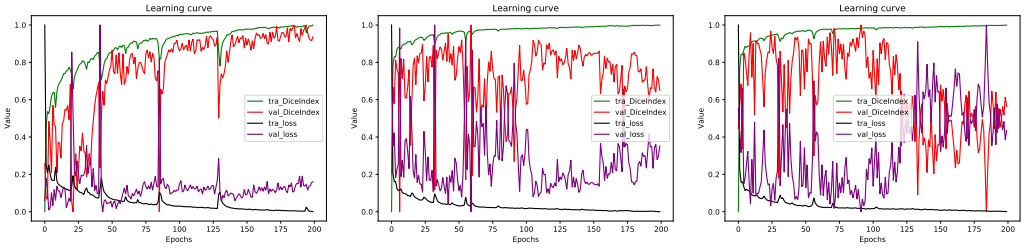


(a) Dice index, train from scratch (b) Dice index, fine tuning

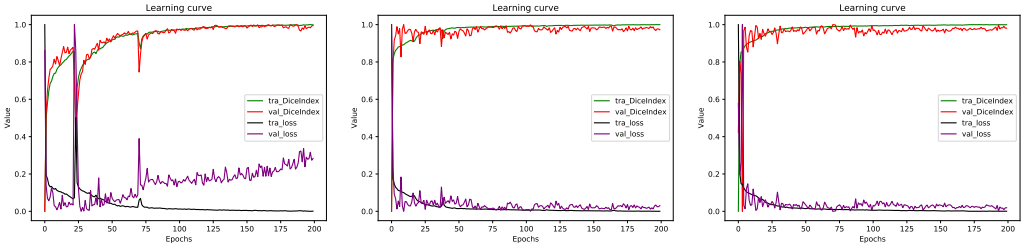


(c) Loss, train from scratch (d) Loss, fine tuning

Figure 8: Learning curve of models in training stage under loss functions: Kappa+Focal (in green), Kappa loss (in blue), Dice loss (in purple). (a) is Dice index learning curve of U-Net trained from scratch; (b) is Dice index learning curve of fine tuning U-Net; (c) and (d) is loss curve of two models.



(a) Kappa+Focal, train from scratch (b) Kappa loss, train from scratch (c) Dice loss, train from scratch



(d) Kappa+Focal loss, fine tuning (e) Kappa loss, fine tuning (f) Dice loss, fine tuning

Figure 9: Learning curve of different models in training and valid stage under 3 losses: Kappa+Focal (in green), Kappa loss (in blue), Dice loss (in purple). (a), (b), (c) is Dice index and loss learning curve of U-Net trained from scratch; (d), (e), (f) is learning curve of fine tuned U-Net.

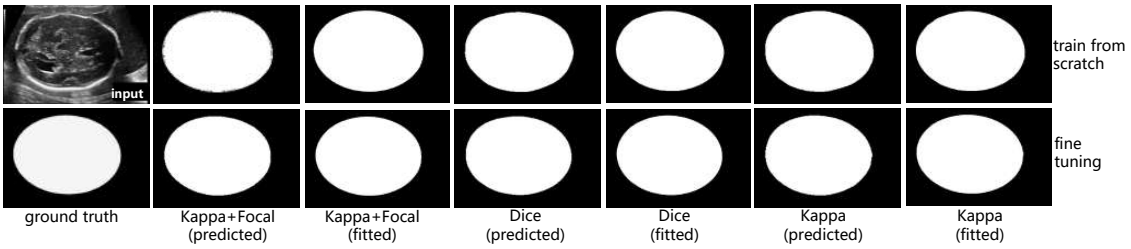


Figure 10: Good prediction results (segmented and fitted results) under models trained from scratch (first line) and fine tuned model (second line) with 3 loss functions respectively.

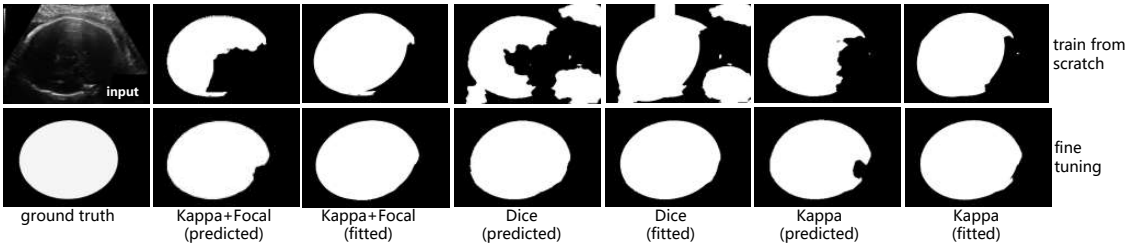


Figure 11: Bad prediction results (segmented and fitted results) under models trained from scratch (first line) and fine tuned model (second line) with 3 loss functions respectively.

3 Head circumference prediction based on regression

3.1 Method

3.1.1 Regression CNN model

We propose regression CNN to train the HC18 dataset. Specifically, they are regression VGG16 [27] and regression ResNet50 [35] pretrained on ImageNet [36]. As shown in Fig. 12, the regression CNN are composed of CNN architecture and regression layer (linear activation function).

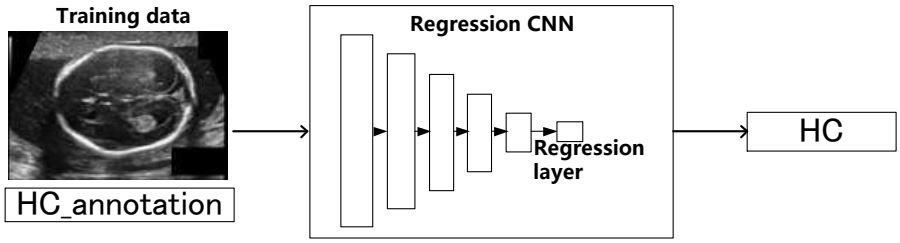


Figure 12: Overview of head circumference prediction process. The HC value is directly predicted through Regression CNN models.

3.1.2 Loss functions

We use 3 different regression loss functions in the regression CNN models. They are Mean Absolute Error (MAE) loss, Mean Square Error (MSE) loss and Huber loss respectively, which are all aiming to reduce the error between predicted value and ground truth value.

$$\text{MAE loss} = \frac{1}{N}\sum_{i=1}^{N}|p_i - g_i| \tag{5}$$

$$\text{MSE loss} = \frac{1}{N}\sum_{i=1}^{N}(p_i - g_i)^2 \tag{6}$$

$$\text{Huber loss} = \begin{cases} \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(p_i - g_i)^2, & \text{for} \quad |p_i - g_i| < \delta \\ \frac{1}{N}\sum_{i=1}^{N}\delta * (|p_i - g_i| - \frac{\delta}{2}), & \text{otherwise} \end{cases} \tag{7}$$

3.2 Experiment settings

3.2.1 HC8 dataset preprocessing

As described in Section 2.2, we still use HC18 dataset to predict head circumference. The input size and number are the same with segmentation experiment. For the input ultrasound images, we perform normalization, for the ground truth HC annotations, we perform normalization divided by the maximum value of HC. In data augmentation, we perform rotation, flipping and translation to augment the training amount.

3.2.2 experiment protocol

We train and predict the HC18 dataset using regression VGG16 [27] and regression ResNet50 [35] pretrained on ImageNet [36] with 3 different loss functions. The basic experiment settings are the same with Section 2.2. But in this experiment, we use Early stopping (epoch=160/200) mechanism when the loss does not changing any more.

3.3 Experiment results

3.3.1 Regression CNN HC prediction

The HC value prediction results are given in Table 4. We using 3 regression loss functions and two deep CNN architectures. We can see that the Regression ResNet50 with MAE loss function has the best average prediction results over 5 fold cross validation.

Table 4: 5-fold cross validation: Performance of different loss functions and models (Regression VGG16 and Regression ResNet50 pretrained with ImageNet). Evaluation metrics are Mean Absolute Error (MAE) in millimeters, Percentage MAE (PMAE).

	Regression VGG16			Regression ResNet50		
Loss	MAE(px)	MAE(mm)	PMAE(%)	MAE(px)	MAE(mm)	PMAE(%)
MAE	33.84±40.16	4.81±6.54	3.27±5.52	29.69±32.89	**4.4**±6.25	2.8±4.7
MSE	34.79±45.16	5.58 ±9.3	3.2±5.15	43.61±49.47	6.44±9.05	5.14±7.73
HL	35.54±41.69	5.2±8.13	3.15±4.89	39.39±38.98	5.76±7.24	2.7±3.58

3.3.2 HC prediction of 3 different trimesters

Inspired by [7], the HC dataset contains 999 ultrasound images of fetal head of 3 trimesters respectively. See Fig. 13. We can see that the fetus in the first trimester is so small that there is almost no skull, while the skull in the second and third trimester is becoming bigger and clearer.

Figure 13: Fetal head images of 3 trimesters and its head circumference in millimeter respectively.

Thus, we manually split the HC dataset into 3 parts in order to train and predict each trimester separately. In [7] Table 1, the number of training images in each trimester are 165, 693, 141. Because they don't make the index of test set public.

We augment these 3 training subset to 999 images and split them into training, valid and test set. The experimental setting is the same with the above description. Table 5 and Table 6 show the results of each trimester. We can see that the HC prediction results of single trimester are better than the results on overall 3 trimesters, which means that the regression ResNet or regression VGG model are more focus on the data that has the consistent features.

Table 5: Performance of Regression ResNet50 of 3 trimesters

Phase	Trimester 1		Trimester 2		Trimester 3	
Loss	MAE(mm)	PAME(%)	MAE(mm)	PAME(%)	MAE(mm)	PAME(%)
MAE	2.66(±3.6)	1.6(±2.26)	1.74(±2.67)	1.05(±1.74)	5.03(±3.74)	1.71(±1.21)
MSE	3.02(±4.51)	1.84(±3.11)	3.73(±5.31)	2.28(±3.62)	7.2(±5.95)	2.44(±1.95)
HL	3.47(±4.84)	2.12(±3.41)	3.95(±6.61)	2.48(±5.32)	6.46(±5.62)	2.2(±1.88)

Table 6: Performance of Regression VGG16 of 3 trimesters

Phase	Trimester 1		Trimester 2		Trimester 3	
Loss	MAE(mm)	PAME(%)	MAE(mm)	PAME(%)	MAE(mm)	PAME(%)
MAE	3.73(±4.36)	1.24(±1.38)	3.12(±2.98)	1.05(±0.97)	2.93(±2.85)	0.98(±0.92)
MSE	4.19(±4.04)	1.4(±1.26)	5.56(±4.62)	1.88(±1.53)	4.74(±5.33)	1.58(±1.71)
HL	6.5(±5.48)	2.2(±1.8)	3.48(±3.54)	1.16(±1.11)	4.25(±4.76)	1.42(±1.55)

3.3.3 HC prediction based on Segmentation vs Regression

Both segmentation CNN and regression CNN methods can predict HC value. However, the predicting process of them is very different. From previous description we know that the segmentation based U-Net model has tedious post processing. While the regression CNN model can directly predict the HC value. To compare the performance of these two kinds of models, we list the performance of different models that predict on the same test set. See Table 7. We can see that almost all of the regression CNN models are better than the U-Net models except for the U-Net loading with VGG model which can also predict comparable results, but it's time consuming and it need manual post processing.

Table 7: HC prediction based on Segmentation vs Regression. All the trainded models use the same test set (200 images).

Regression	MAE(mm)	MAE(pixel)	PMAE(%)	**Segmentation**	MAE(mm)	MAE(pixel)	PMAE(%)
VGG(MAE)	4.24(±6.46)	28.17(±36.81)	2.49(±3.47)	U-Net(Dice)	7.6(±8.03)	58.57(±75.84)	5.68(±11.21)
VGG(MSE)	4.8(±8.8)	29.24(±42.06)	2.56(±3.48)	U-Net(Kappa)	8.41(±7.84)	62.83(±69.44)	5.89(±10.18)
VGG(HL)	**4.03**(6.46)	29.53(±35.98)	2.48(±3.48)	U-Net(K-F)	14.05(±7.57)	105.42(±77.41)	9.42(±10.92)
ResNet(MAE)	4.69(±7.69)	31.85(±37.78)	2.87(±3.91)	U-Net(VGG,Dice)	**4.19**(±5.01)	31.92(±49.52)	3.1(±7.65)
ResNet(MSE)	6.99(±8.96)	49.23(±53.63)	4.73(±6.79)	U-Net(VGG,Kappa)	4.55(±7.25)	37.17(±74.83)	3.89(±10.79)
ResNet(HL)	5.79(±6.65)	41.09(±42.97)	3.78(±5.0)	U-Net(VGG,K-F)	6.99(±7.21)	52.96(±73.69)	4.85(±9.37)

4 Explainability on regression CNN of head circumference prediction

To open the black box of CNN model, there are plenty of methods to explain the inside of networks. One of the manifestations is to show the highlighting features learned by the model through a certain analysis (interpretation) method, which is called saliency maps. Another method is to judge the ability of a certain analysis method to identify features by adding perturbation.

4.1 Saliency maps

There are two categories of saliency maps are generally considered, perturbations-based or propagation-based. In perturbation-based approaches, the goal is to estimate how perturbation applied to the input image, such as blurring or injecting noise, changes the predicted class [37, 38]. In propagation-based techniques, the idea is to backpropagate a relevance signal from the output to the input. In this paper, we will focus on the latter category of methods that actually

encompass (i) sensitivity (or gradient-based) analysis, (ii) deconvolution methods, and (iii) Layer-wise Relevance Propagation (LRP) variants.

The sensitivity analyzers include the **Gradient** [39] method, that simply computes the gradient of the output w.r.t. input image, and expresses how much the output value changes w.r.t. a small change in input; the **SmoothGrad** [40], that averages the gradient over random samples in a neighborhood of the input with added noise, and which is an improvement of Gradient method that can sharpen the saliency map; the **Input*Gradient** [41] technique, that strengthens the saliency map by multiplying Gradient with input information; and the **Integrated Gradients** [42], that computes the integration of the gradient along a path from the input to a baseline black image.

Deconvolution methods are the **DeConvNet** [43] that acts equivalently as a decoder of CNN models, which reverses the CNN layers, and the **Guided BackProp** [44] that combines backpropagation and DeConvNet.

The core idea of **Layer-wise Relevance Propagation (LRP)** [45] is to compute a relevance score for each input pixel layer by layer in backward direction. It first forward-passes the image so as to collect activation maps and backpropagates the error taking into account the network weights and activations. The **DeepTaylor** [46] method identifies the contribution of input features as the first-order of a Taylor expansion, through Taylor decomposition, then it can estimate the attribution of each neuron one by one.

In the classification setting, a saliency map provides an estimation of how much each pixel contributes to the class prediction. In the regression setting, the saliency map will provide an estimation of how much each pixel is impacting the model, and is contributing to decrease the prediction error, as measured by the loss function, that is in general the MAE or MSE as well as Huber loss. Fig. 14 shows saliency maps of different methods. We can see that the methods DeConvNet and Gradients are insensitive to regression CNN models and loss functions, and there are no highlight features that can be understood by humans. While the other methods show the highlight features that the regression CNN models have learned, which offers people reliability of predicting HC values.

4.2 Perturbation based evaluation

Explanation methods (also called analyzers) perform differently depending on the model, the task at hand, the data, etc. In order to quantitatively evaluate those analyzers, we build upon the perturbation analysis of [47], originally designed to assess explainability methods in classification networks. Let us first describe the perturbation process and then the evaluation metric. Fig. 15 is an example of the perturbation process of Gradient analyzer.

First, the input image to be analyzed is subsampled by a grid. Each subwindow of the grid is ranked according to its importance w.r.t. to the pixel-wise saliency scores assigned by the analyzers. Then, the information content of the image is gradually corrupted by adding perturbation (Gaussian noise) to each subwindow, starting with the most relevant subwindow, w.r.t. the ranking just mentioned. The effect of this perturbation on the model performance is measured with the prediction error. This procedure is repeated for each subwindow. Generally, the accuracy of model will drop quickly when important information is removed and remains largely unaffected when perturbing unimportant regions. Thus, the analyzers can be compared by measuring how quickly their performance drops. That is to say, the quicker the model performance drops after introducing perturbation, the better the analyzer is capable of identifying the input components responsible for the output of the model.

The quantitative evaluation proposed in [47] for classification network, consists in computing the difference between the score $f(x)$ indicating the certainty of the presence of an object in the image x, in the presence and in the absence of perturbation. This difference is called Area over Perturbation Curve (AOPC) and defined more precisely defined in in [47] as:

$$\mathrm{AOPC}_{Analyzer} = \frac{1}{N}\sum_{n=0}^{N}(f(x_n)^{(0)} - \frac{1}{K}\sum_{k=0}^{K} f(x_n)^{(k)}) \tag{8}$$

where N is the number of images, K is the number of perturbation steps, x is the input image.

Here, we propose to adapt the AOPC to the regression case, and if we denote by $\epsilon(x)^{(0)}$ the prediction error of initial image evaluated by the analyzer and $\epsilon(x_n)^{(k)}(1 \leq k \leq K)$ the prediction error of the perturbed image $(x_n)^{(k)}$ at step k, we can define the $\mathrm{AOPC}^{\mathrm{regression}}_{\mathrm{Analyzer}}$ as:

$$\mathrm{AOPC}^{regression}_{Analyzer} = \frac{1}{N}\sum_{n=0}^{N}(\epsilon(x_n)^{(0)} - \frac{1}{K}\sum_{k=0}^{K} \epsilon(x_n)^{(k)}) \tag{9}$$

A larger AOPC in absolute value means that an analyzer has a steep decrease in accuracy (increase in predicion error, see Fig. 16.) while the perturbation steps is increasing. Table. 8 is the AOPC score of different analyzers on different

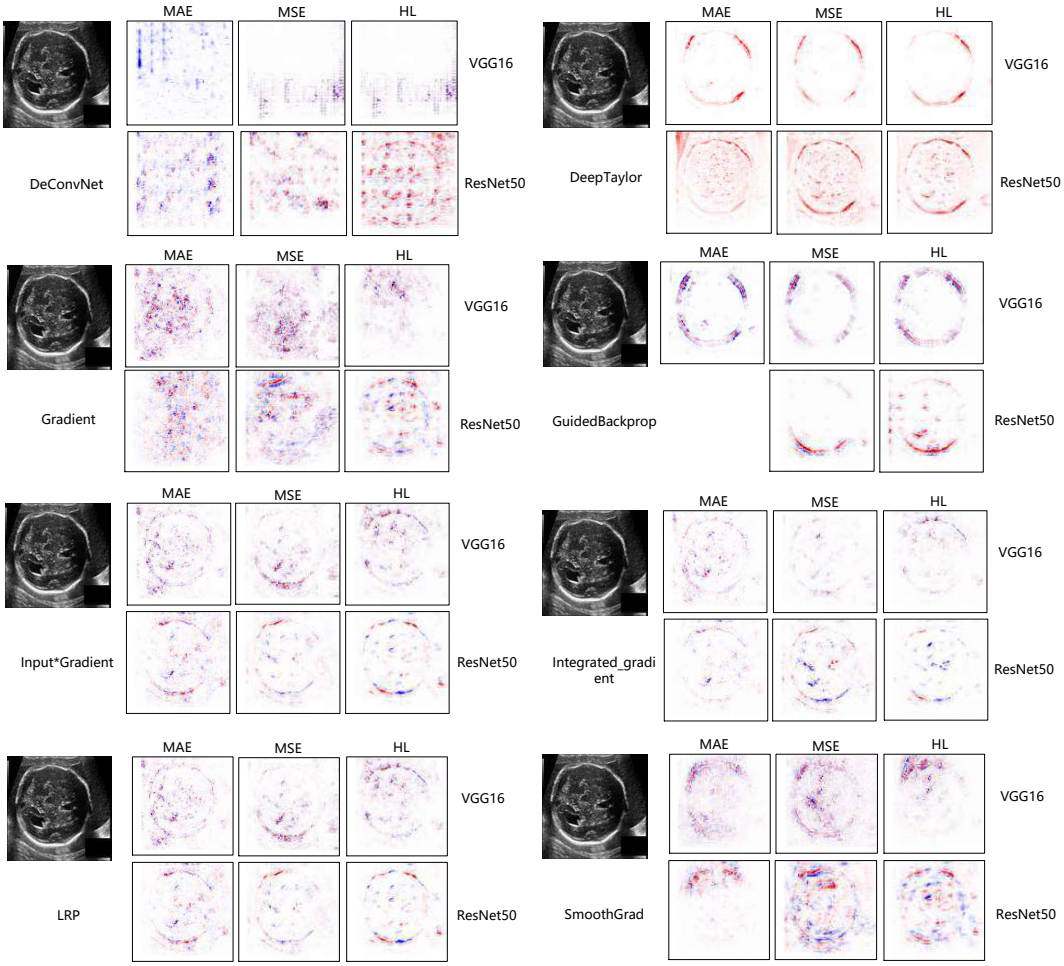


Figure 14: Saliency maps of different explanation methods under 3 different loss functions and regression CNN model VGG16 and ResNet50 on the same input image.

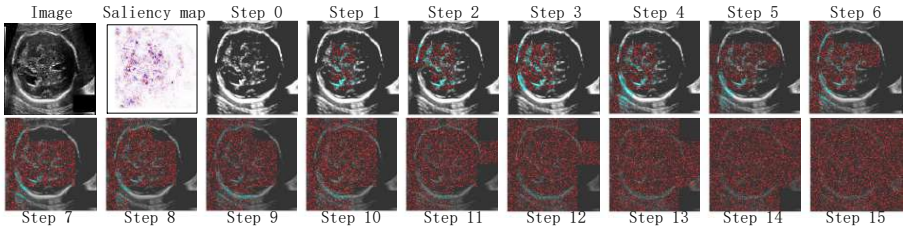


Figure 15: Perturbation process for the saliency map produced by the Gradient method. Step 0 is the original input image. From step 1 to step 15, Gaussian noise is added gradually on the image subwindows. The perturbation order of these subwindows corresponds to the saliency scores assigned by the Gradient method analysis, i.e. the most contributing pixels are perturbed first. Red: noise, blue: original image pixels.

models. We can see that basicilly the methods Input*Gradient and LRP perform more stable and robust among the other methods.

4.3 Deep Dream based explanation on segmentation

The idea of Deep Dream [48] is to maximize the activation value of the convolutional layer to visualize the characteristics of the convolutional layer by running gradient ascent over an input image.
The process of Deep Dream:

Table 8: Performance (AOPC scores) of different analysis methods after perturbation, with two regression models and three loss functions. G: Gradient, SG: SmoothGrad, DCN: DeConvNet, DT: DeepTaylor, GB: GuidedBackprop, I*G: Input*Gradient, IG: IntegratedGradients. Lower is better. Best scores in bold.

Model	G	SG	DCN	DT	GB	I*G	IG	LRP
VGG16_MAE	-7.312	-7.398	-2.869	-7.401	-1.663	**-9.189**	**-9.490**	**-9.175**
VGG_MSE	-7.805	-7.181	-5.356	-9.098	-2.990	**-14.568**	-	**-14.460**
VGG_HL	-23.389	-21.626	-24.588	-27.779	-18.859	**-29.468**	-	**-29.268**
ResNet50_MAE	-11.533	-11.841	-9.249	-9.890	-9.717	**-14.748**	-5.603	**-14.577**
ResNet50_MSE	-11.305	-	-11.187	-19.408	-	**-32.487**	**-20.495**	**-32.505**
ResNet50_HL	-24.174	-24.269	-	-22.665	-28.421	**-37.125**	-22.814	**-38.120**

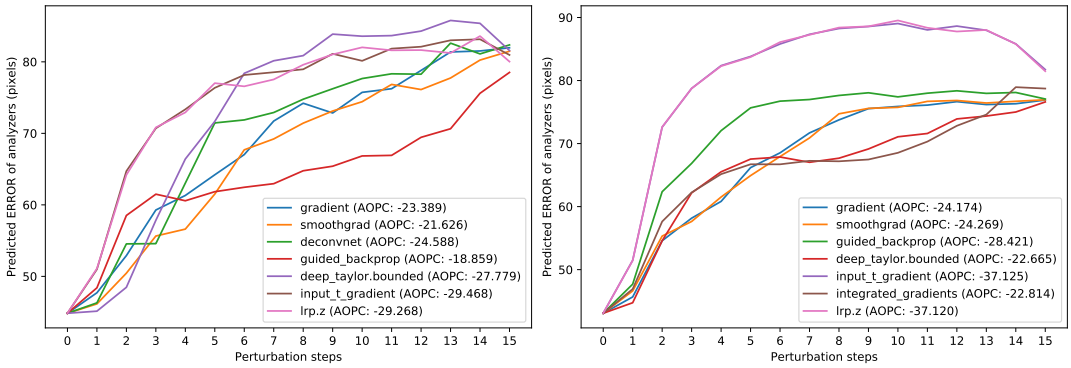


Figure 16: The change of prediction error with adding perturbation of different analyzers under Regression VGG16 and ResNet50 model with loss function HL.

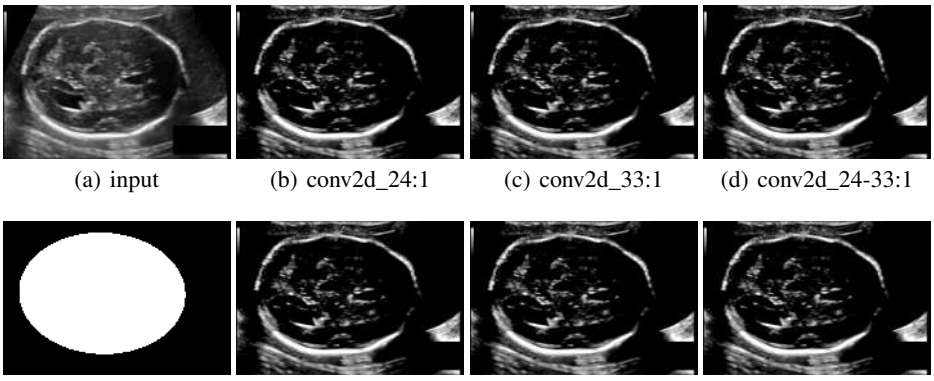
- Enter any image.
- Set a number of different scales, e.g. 3.
- Resize the original image to the smallest scale.
- For every scale, starting with the smallest:
  - Run gradient ascent
  - Upscale image to the next scale
  - Reinject the detail that was lost at upscaling time
- Stop when we are back to the original size. To obtain the detail lost during upscaling, we simply take the original image, shrink it down, upscale it, and compare the result to the (resized) original image.

Since Deep Dream can maximize specific layer (s) about input image, so we can utilize this method to check the inside of CNN models. In this work, we explore the U-Net model, as we know, this model consists of two parts, encoder CNNs and decoder CNNs, we separately "Deep Dream" the shallow and deep layer in encoder and decoder, then we "Deep Dream" the entire encoder and decoder part, respectively. Fig. 17 shows the results of Deep Dream method, we can see that the dark part of input image is weakened and the light part is enhanced, but we hardly see what is changed in different layers when the weight of each layer is set as 1. If we know the importance of a certain layer, we could set a high weight to this layer so that the features can be clearly perceived by the following layers. The question is how to know the importance of each layer. It may be unrealistic to find this answer through exhaustive methods.

5 Cardiac components volume prediction based on 3D regression CNN

5.1 Data Preprocessing

The "Automatic Cardiac Diagnosis Challenge" dataset (ACDC) [12] contains 100 patients, 5 detection types, 1 normal plus 4 diseases. The task is to compute the volume of cardiac components (LV, RV, MYO) in two phase (Dilation and Contraction). They offer the MRI images and corresponding ground truth images (4 classes in all. 0 for background, 1 for RV, 2 for MYO,3 for LV) as well as volumes of each component of each patient. The shape of each patient is different from the others. Thus, it's necessary to preprocesse these data.



(a) input (b) conv2d_24:1 (c) conv2d_33:1 (d) conv2d_24-33:1

(e) ground truth (f) conv2d_34:1 (g) conv2d_46:1 (h) conv2d_34-46:1

Figure 17: The input sample and images generated by Deep Dream on U-Net segmentation model. (b):shallow layer of encoder, (c):deep layer of encoder, (d):whole layers of encoder, (f):shallow layer of decoder, (g):deep layer of decoder, (h):whole layers of decoder, the weight of each layer is 1.

5.1.1 Data cleaning

There are some unacceptable errors in this dataset. That is to say, the volume of cardiac components computed through vol (ml) = vol (pixel) *pixel size in mm in x*pixel size in mm in y*space between slices/1000 is greatly different from what they offer. See Table 9. The absolute errors of left 94 patients are less than 1.

Table 9: Abnormal samples in ACDC dataset

Patient	type	spacing(x,y,z)	shape	volume(ml)	area(pixel)	vol(ml)	ABS
patient019	DCM_33	(1.445,1.445,10)	(11, 256, 216)	868.5903	32269	673.7848	194.8055
patient078	NOR_35	(1.367,1.367,10)	(8, 256, 216)	630.2002	25813	482.3647	147.8355
patient079	NOR_36	(1.367,1.367,10)	(9, 256, 216)	455.7373	18667	348.8282	106.9091
patient080	NOR_37	(1.758,1.758,10)	(6, 256, 216)	223.9502	9173	283.4974	59.5472
patient093	RV_37	(1.563,1.563,7)	(10, 224, 180)	57.3511	23491	401.7145	344.3464
patient099	RV_43	(1.786,1.786,5)	(16, 224, 154)	866.7050	27180	433.4933	433.2117

5.1.2 Data cropping

As Table 9 shows, the shape of each patient is different. Since a 3D CNN model requires fixed size of input data. So we perform cropping on ground truth data first, then the MRI images. The cropping steps are as follows:
1. In each slice of a patient, find the contour of the heart, then get the 4 coordinate points of bounding box which is based on the contour.
2. Among all the slices of a patient, find the largest bounding box and its coordinates.
3. Among all the patients. find the largest bounding box and its coordinates. But in this step, the size of one patient is too larger than the others.The bounding box is also larger than the other images. So we skip this patient.
4. Given the final bounding box coordinates, we start cropping in x, y direction. Because some of the images are smaller than the bounding box, we make the border of them by adding the neighboring pixels.

5.1.3 Data resampling

In z direction, the number of slices are also different from each other (From 6 to 17 slices). So we find a median number 9 as the fixed number of slices. Before removing or adding slice, we first delete some meaningless slices of each patient which are all background information. Afterwards, we add or remove the smallest area of a slice to make the smallest influence on the original data as possible. And the final shape of all the patients are 147*186*9.

5.2 Multi-class prediction with regression 3D VGG networks

As ACDC dataset has 3 classes to predict, they are RV, MYO and LV respectively. Therefore, we propose multi-class prediction with regression 3D VGG networks, see Fig. 19. There are 3 branches of CNNs, each branch predict one class.

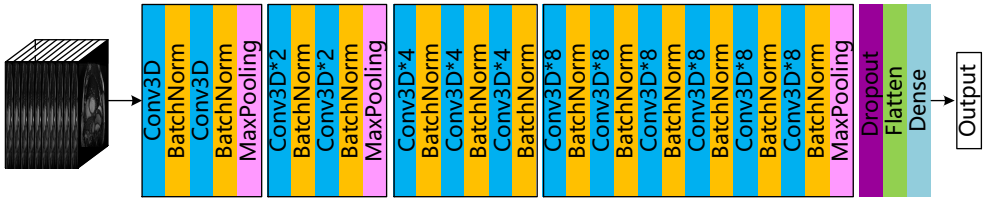


Figure 18: The adatped 3D VGG network

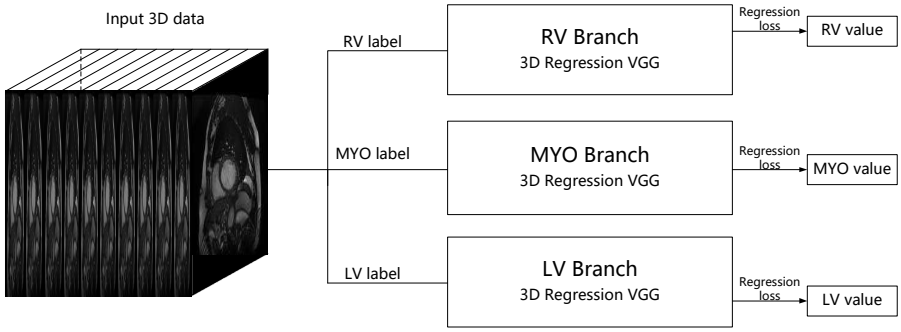


Figure 19: Workflow of direct ACDC prediction.

In each branch, we use adapted 3D VGG networks, see Fig. 18. Because the size of preprocessed data is 9*224*224, the number of slice is not sufficient to the original VGG16 networks. One paper's method [20] is to resize the data to gain the number of slices. What we do is to remove two MaxPooling layers to ensure it can output without errors. Besides, after each convolutional layer, a Batch Normalization layer is added to avoid gradient vanishing problem. As well as an activation function ReLU. The initial number of convolutional kernel number is 8, the kernel size is 3*3*3, the pool size and stride is 3 and 2 respectively. The Dropout rate is 0.5. In the last layer, the activation function is linear, and the number of output is one.

5.3 Experiments and results

5.3.1 Experimental settings

There area totally 200 patients in this dataset. We split these patients into training set (120), validation set (40) and test set (40). In the training set, we do data augmentation beforehand (horizontal flipping, height and width translation with 5 degrees respectively.), so there are 480 samples in it. Note, in 3D data augmentation, the rotation operation may affect the shape of image, because the image is interpolated during the rotation operation, which will cause the image a serious mathematical mapping change, which will affect the network training performance.Before training, we resize the data to 224*224*9, and normalize them by $(x - \mu)/\sigma$. Also the ground truth are normalized by $y/max(y)$.
The experiment is performed under 5 fold cross validation. The regression model is 3D VGG with linear activation layer and regression loss functions. The model is trained from scratch. Because the thickness of data is 9, so we changed the kernel size to (3*3*2). The optimizer is Adam. The learning rate is $1e^{-3}$, the batch size is 8. The model is completed using Keras library and run in GPU p100. The training epochs is 100.

5.3.2 Results

Table 10: Performance of Regression VGG of ED and ES

Type	ED						ES					
Structure	RV		MYO		LV		RV		MYO		LV	
Models	MAE	PAME	MAE	PAME	MAE	PAME	MAE	PAME	MAE	PAME	MAE	PAME
2D-VGG-Multi-Output-Volume(35min)	54.73(±32.00)	32.03(±22.45)	34.86(±19.34)	50.94(±39.56)	67.18(±29.39)	46.30(±30.97)						
2D-VGG-Multi-Output-Pixel(35min)	54.26(±38.94)	27.93(±18.54)	31.68(±27.11)	40.72(±38.91)	41.98(±25.63)	40.32(±29.79)						
2D-VGG-Multi-Label-Volume(13min)	55.08(±34.80)	29.06(±17.68)	41.17(±21.54)	51.29(±31.37)	60.434(±31.68)	48.84(±33.72)						
2D-VGG-Multi-Label-Pixe(18min)l	47.70(±33.02)	25.37(±18.02)	34.67(±24.38)	48.37(±38.29)	44.59(±30.70)	41.33(±32.02)						
3D-VGG-Multi-Output-Volume(2h48min)	47.93(±32.56)	33.04(±32.27)	48.83(±21.65)	54.71(±36.65)	84.28(±30.45)	55.60(±35.79)						
3D-VGG-Multi-Output-Pixel(2h49min)	57.19(±39.63)	31.19(±20.84)	44.36(34.29)	52.40(±46.03)	89.63(±41.52)	64.62(±48.07)						
3D-VGG-Multi-Label-Volume(57min)	64.85(±38.68)	33.69(±14.79)	51.16(±21.83)	59.55(±33.10)	87.64(±29.65)	60.17(±32.51)						
3D-VGG-Multi-Label-Pixel(57min)	58.80(±41.40)	31.45(±20.90)	47.08(±36.18)	57.35(±49.88)	86.80(±41.10)	62.85(±47.20)						

Table 11: Performance of Regression VGG of ED and ES, the input is ground truth of cropped 3D cardiac data, the first two model output predicted 3 volumes, the last model predict the whole volume of cardiac.

	ED			ES		
	RV(ml)	MYO(ml)	LV(ml)	RV	MYO	LV
2D-VGG-Multi-Lable-Pixel	41.17(±26.68)	13.85(±11.56)	27.05(±13.23)			
2D-VGG-Multi-Output-Pixel	56.75(±41.01)	12.62(±9.03)	16.07(±13.60)			
2D-VGG-Single-Label-Pixel		36.63(±34.81)				

Table 12 is the performance of adapted regression 3D VGG networks on directly prediction the volume of cardiac structures. It took about 22 hours to train the model from scratch, 160 seconds an epoch. From the results we can see that the difference from true value is large.

Table 12: Performance of Regression 3D VGG on prediction volume of cardiac structures.

Loss	RV		MYO		LV	
	MAE(ml)	PAME(%)	MAE(ml)	PAME(%)	MAE(ml)	PAME (%)
MAE	76.488(±61.14)	58.95(±76.59)	54.255(±32.611)	89.24(±173.89)	104.959(±49.57)	162.93 (±255.87)
MSE	73.13(±51.40)	89.48(±157.11)	62.519(±39.65)	104.47(±210.03)	107.315(±53.63)	159.66 (±265.99)
HL	67.232(±52.00)	75.46(±129.64)	55.199(±30.68)	129.57(±249.42)	105.845(±49.6)	171.82 (±270.37)

Table 13 is the performance of adapted regression 3D VGG networks on cleaned data, in which we removed 6 abnormal patients (see Table 9). We can see that the predicted volumes of three structures on cleaned data are better than that in Table 12.

Table 13: Performance of Regression 3D VGG on prediction volume of cardiac structures on cleaned data.

Loss	RV		MYO		LV	
	MAE(ml)	PAME(%)	MAE(ml)	PAME(%)	MAE(ml)	PAME (%)
MAE	72.838(±61.04)	42.92(±31.76)	43.408(±26.37)	47.33(±38.76)	94.412(±37.81)	95.71 (±75.75)
MSE	53.602(±35.05)	43.68(±45.57)	45.391(±26.34)	42.82(±29.17)	97.163(±40.55)	111.62 (±88.89)
HL	61.89(±39.59)	50.48(±53.55)	48.051(±25.07)	62.10(±47.34)	87.973(±38.10)	82.47 (±65.51)

We found that training 3D VGG networks takes long time, 22 hours or so on GPU p100. So we change the networks to 2D VGG, which the number of channels is the number of slices of input data. And the training time only takes 30 minutes. 3 seconds an epoch, which greatly reduce the training time. And the performance is shown in Table 14. We can see that it performs similarly with 3D VGG.

References

[1] S Kevin Zhou, Hayit Greenspan, Christos Davatzikos, James S Duncan, Bram van Ginneken, Anant Madabhushi, Jerry L Prince, Daniel Rueckert, and Ronald M Summers. A review of deep learning in medical imaging: Image traits, technology trends, case studies with progress highlights, and future promises. *arXiv preprint arXiv:2008.09104*, 2020.

[2] I Sarris, C Ioannou, P Chamberlain, E Ohuma, F Roseman, L Hoch, DG Altman, AT Papageorghiou, and INTERGROWTH-21st. Intra-and interobserver variability in fetal ultrasound measurements. *Ultrasound in obstetrics & gynecology*, 39(3):266–273, 2012.

[3] Thomas L. A. van den Heuvel, Dagmar de Bruijn, Chris L. de Korte, and Bram van Ginneken. Automated measurement of fetal head circumference using 2d ultrasound images [data set]. *Zenodo*, 2018.

[4] Jing Li, Yi Wang, Baiying Lei, Jie-Zhi Cheng, Jing Qin, Tianfu Wang, Shengli Li, and Dong Ni. Automatic fetal head circumference measurement in ultrasound using random forest and fast ellipse fitting. *IEEE journal of biomedical and health informatics*, 22(1):215–223, 2017.

[5] Wei Lu, Jinglu Tan, and Randall Floyd. Automated fetal head detection and measurement in ultrasound images by iterative randomized hough transform. *Ultrasound in Medicine & Biology*, 31(7):929 – 936, 2005.

[6] Sandra M.G.V.B. Jardim and Mário A.T. Figueiredo. Segmentation of fetal ultrasound images. *Ultrasound in Medicine and Biology*, 31(2):243 – 250, 2005.

[7] Thomas L. A. van den Heuvel, Dagmar de Bruijn, Chris L. de Korte, and Bram van Ginneken. Automated measurement of fetal head circumference using 2d ultrasound images. *PLOS ONE*, 13(8):1–20, 08 2018.

Table 14: Performance of Regression 2D VGG on prediction volume of cardiac structures on cleaned data.

Loss	RV		MYO		LV	
	MAE(ml)	PAME(%)	MAE(ml)	PAME(%)	MAE(ml)	PAME (%)
MAE	74.329(±54.66)	44.23(±28.19)	50.898(±25.80)	58.74(±44.46)	90.555(±32.82)	75.72 (±55.84)
MSE	57.641(±42.13)	41.06(±37.96)	47.299(±23.82)	59.59(±43.81)	92.53(±36.57)	93.17 (±73.60)
HL	65.399(±48.95)	41.32(±31.87)	41.234(±26.65)	56.09(±42.28)	78.184(±38.09)	85.74 (±78.47)

[8] Hwa Pyung Kim, Sung Min Lee, Ja-Young Kwon, Yejin Park, Kang Cheol Kim, and Jin Keun Seo. Automatic evaluation of fetal head biometry from ultrasound images using machine learning. *Physiological measurement*, 40(6):065009, 2019.

[9] Samuel Budd, Matthew Sinclair, Bishesh Khanal, Jacqueline Matthew, David Lloyd, Alberto Gomez, Nicolas Toussaint, Emma C Robinson, and Bernhard Kainz. Confident head circumference measurement from ultrasound with real-time feedback for sonographers. In *MICCAI*, pages 683–691, 2019.

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[11] Zahra Sobhaninia, Shima Rafiei, Ali Emami, Nader Karimi, Kayvan Najarian, Shadrokh Samavi, and SM Reza Soroushmehr. Fetal ultrasound image segmentation for measuring biometric parameters using multi-task deep learning. In *Conference of the IEEE EMBC*, pages 6545–6548, 2019.

[12] Olivier Bernard, Alain Lalande, Clement Zotti, Frederick Cervenansky, Xin Yang, Pheng-Ann Heng, Irem Cetin, Karim Lekadir, Oscar Camara, Miguel Angel Gonzalez Ballester, et al. Deep learning techniques for automatic mri cardiac multi-structures segmentation and diagnosis: is the problem solved? *IEEE transactions on medical imaging*, 37(11):2514–2525, 2018.

[13] Christian F Baumgartner, Lisa M Koch, Marc Pollefeys, and Ender Konukoglu. An exploration of 2d and 3d deep learning techniques for cardiac mr image segmentation. In *International Workshop on Statistical Atlases and Computational Models of the Heart*, pages 111–119. Springer, 2017.

[14] Fabian Isensee, Paul F Jaeger, Peter M Full, Ivo Wolf, Sandy Engelhardt, and Klaus H Maier-Hein. Automatic cardiac disease assessment on cine-mri via time-series segmentation and domain specific features. In *International workshop on statistical atlases and computational models of the heart*, pages 120–129. Springer, 2017.

[15] Yeonggul Jang, Yoonmi Hong, Seongmin Ha, Sekeun Kim, and Hyuk-Jae Chang. Automatic segmentation of lv and rv in cardiac mri. In *International Workshop on Statistical Atlases and Computational Models of the Heart*, pages 161–169. Springer, 2017.

[16] Mahendra Khened, Varghese Alex, and Ganapathy Krishnamurthi. Densely connected fully convolutional network for short-axis cardiac cine mr image segmentation and heart diagnosis using random forest. In *International Workshop on Statistical Atlases and Computational Models of the Heart*, pages 140–151. Springer, 2017.

[17] Jay Patravali, Shubham Jain, and Sasank Chilamkurthy. 2d-3d fully convolutional neural networks for cardiac mr segmentation. In *International Workshop on Statistical Atlases and Computational Models of the Heart*, pages 130–139. Springer, 2017.

[18] Marc-Michel Rohé, Maxime Sermesant, and Xavier Pennec. Automatic multi-atlas segmentation of myocardium with svf-net. In *International Workshop on Statistical Atlases and Computational Models of the Heart*, pages 170–177. Springer, 2017.

[19] Jelmer M Wolterink, Tim Leiner, Max A Viergever, and Ivana Išgum. Automatic segmentation and disease classification using cardiac cine mr images. In *International Workshop on Statistical Atlases and Computational Models of the Heart*, pages 101–110. Springer, 2017.

[20] Xin Yang, Cheng Bian, Lequan Yu, Dong Ni, and Pheng-Ann Heng. Class-balanced deep neural network for automatic ventricular structure segmentation. In *International workshop on statistical atlases and computational models of the heart*, pages 152–160. Springer, 2017.

[21] Clement Zotti, Zhiming Luo, Olivier Humbert, Alain Lalande, and Pierre-Marc Jodoin. Gridnet with automatic shape prior registration for automatic mri cardiac segmentation. In *International Workshop on Statistical Atlases and Computational Models of the Heart*, pages 73–81. Springer, 2017.

[22] Xiantong Zhen, Zhijie Wang, Ali Islam, Mousumi Bhaduri, Ian Chan, and Shuo Li. Direct volume estimation without segmentation. *SPIE Progress in Biomedical Optics and Imaging*, 9413, 2015.

[23] Xiantong Zhen, Zhijie Wang, Ali Islam, Mousumi Bhaduri, Ian Chan, and Shuo Li. Multi-scale deep networks and regression forests for direct bi-ventricular volume estimation. *Medical image analysis*, 30:120–129, 2016.

[24] Wojciech Samek and Klaus-Robert Müller. Towards explainable artificial intelligence. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 5–22. Springer, 2019.

[25] Niels JS Morch, Ulrik Kjems, Lars Kai Hansen, Claus Svarer, Ian Law, Benny Lautrup, Steve Strother, and Kelly Rehm. Visualization of neural networks using saliency maps. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 2085–2090, 1995.

[26] Amitojdeep Singh, Sourya Sengupta, and Vasudevan Lakshminarayanan. Explainable deep learning models in medical image analysis. *Journal of Imaging*, 6(6:52), 2020.

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[28] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. IEEE, 2016.

[29] Jing Zhang, Caroline Petitjean, and Samia Ainouz. Kappa loss for skin lesion segmentation in fully convolutional network. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pages 2001–2004. IEEE, 2020.

[30] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[31] Roger W Barnard, Kent Pearce, and Lawrence Schovanec. Inequalities for the perimeter of an ellipse. *Journal of mathematical analysis and applications*, 260(2):295–306, 2001.

[32] Thomas LA van den Heuvel, Dagmar de Bruijn, Chris L de Korte, and Bram van Ginneken. Automated measurement of fetal head circumference using 2d ultrasound images. *PloS one*, 13(8):e0200412, 2018.

[33] Zahra Sobhaninia, Shima Rafiei, Ali Emami, Nader Karimi, Kayvan Najarian, Shadrokh Samavi, and SM Reza Soroushmehr. Fetal ultrasound image segmentation for measuring biometric parameters using multi-task deep learning. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 6545–6548. IEEE, 2019.

[34] Samuel Budd, Matthew Sinclair, Bishesh Khanal, Jacqueline Matthew, David Lloyd, Alberto Gomez, Nicolas Toussaint, Emma C Robinson, and Bernhard Kainz. Confident head circumference measurement from ultrasound with real-time feedback for sonographers. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 683–691. Springer, 2019.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[37] Ruth Fong and Andrea Vedaldi. Interpretable Explanations of Black Boxes by Meaningful Perturbation. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3449–3457, October 2017. arXiv: 1704.03296.

[38] Luisa M. Zintgraf, Taco S. Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis, 2017.

[39] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2014.

[40] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. In *Workshop on Visualization for Deep Learning, ICML*, 2017.

[41] Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. *CoRR*, abs/1605.01713, 2016.

[42] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328. JMLR. org, 2017.

[43] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[44] J.T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR (workshop track)*, 2015.

[45] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7), 2015.

[46] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[47] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2016.

[48] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. *Google Research Blog*, 2015.