university of groningen / faculty of science and engineering / biomedical engineering

Prediction of Postnatal Fetal Renal Function: A Deep Learning Approach

Jeremi Olejnik, BSc

S5728932

Department of Obstetrics and Gynecology / Data Science Center in Health (DASH), UMCG

Period: 15/04/2024 - 05/07/2024

Internship

1st Examiner: [dr. ir. P.M.A. (Peter) van Ooijen / Faculty of Medical Sciences, RUG]

2nd Examiner: [dr. Federica Fontanella / Obstetrics and Gynecology, UMCG]

Abstract

Congenital obstructive uropathies present a significant challenge in pediatric care, being the leading cause of renal failure in children. These anomalies, characterized by urinary tract obstruction (UTO), underscore the importance of early detection and intervention. Fetal renal assessment through ultrasound imaging offers the potential for timely diagnosis; however, the lack of standardized assessment methods makes it difficult to accurately predict renal function. This internship aimed to explore the integration of deep learning techniques to enhance the ultrasound-based evaluation of postnatal fetal renal function.

The project was structured into three comprehensive phases: preprocessing, model development, and model evaluation. In the preprocessing phase, extensive image corrections, including inpainting measurements, were performed, ensuring the quality and consistency of input images. Model development phase involved adjustments to various pertained functional model architectures and weights, along with hyperparameter tuning using cross-validation to optimize model performance. Additionally, explainable AI (XAI) techniques, specifically GradCAM, were incorporated to generate visual explanations of the model's predictions, enhancing interpretability. The final evaluation phase focused on assessing the model's accuracy and reliability in predicting renal function from ultrasound images.

Despite the small size of the dataset, which consisted of only 97 images, the results demonstrate the potential of deep learning to improve the accuracy of predicting postnatal fetal renal function from ultrasound images. The best-performing model achieved an accuracy of 68.42%, an F1 score of 66.(6)%, a sensitivity of 75%, a specificity of 63.64%, and an AUC of 69.32%. These promising outcomes highlight the need for further research and larger datasets to enhance the model's predictive capabilities before they could be used in clinical practice.

Table of contents

1. Introduction

Congenital obstructive uropathies are the most common urinary tract anomalies, which are the largest recognizable cause of renal failure in infants and children [1]. These anomalies result from urinary tract obstructions that can occur at different levels (Figure 1): at the renal pelvis or fetal ureter, in the case of upper urinary tract obstruction (UUTO); below the bladder neck, in the case of lower urinary tract obstruction (LUTO) [1]. UUTO is typically characterized by unilateral hydronephrosis and possible dilatation of the renal calyces, with severe cases potentially leading to massive dilatation, rupture of the collecting system, and perinephric ureter or urinary ascites. LUTO, often caused by partial or complete blockage of urination due to posterior urethral valves, is associated with antenatal symptoms such as enlarged urinary bladder, bilateral hydronephrosis, hydroceles, and oligohydramnios after the second trimester of pregnancy. If untreated, LUTO has a mortality rate of 45% [2], with approximately 25-30% of surviving infants developing renal failure requiring dialysis or transplant in childhood [3].

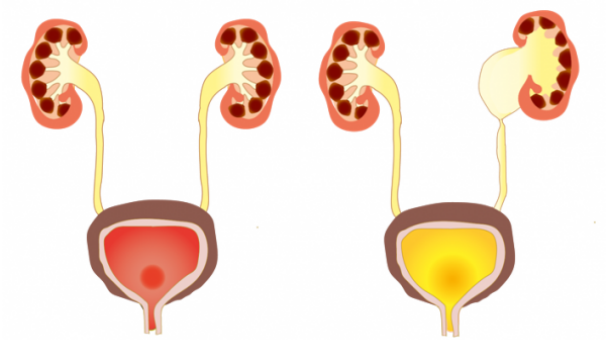

Figure 1. Normal (left) and abnormal (right) urinary tract system, with stenosis (narrowing) between the ureter and the kidney, inducing accumulation of urine in renal pelvis and calyces (hydronephrosis) [4]

Congenital defects of the urinary tract constitute 20-30% of all congenital defects detected antenatally, with detection rates reaching approximately 90% [5]. Despite this high detection rate, effective diagnosis in the prenatal period is limited by the lack of comprehensive follow-up studies and a commonly accepted approach to objectively assess fetal renal function. The current best predictors of postnatal kidney functioning rely on ultrasound imaging to assess renal cortical parenchymal appearance and renal echogenicity, which refers to the ability of the renal tissue to reflect ultrasound waves, indicating its density and composition. However, this assessment is based on the subjective evaluation of the sonographer, which has shown poor agreement between experts and low accuracy. Therefore, there is a need for standardized diagnostic procedures that allow for a more precise and objective assessment of fetal renal function, which would significantly improve early intervention techniques, prenatal care, and outcomes for affected children.

In recent years, artificial intelligence (AI) has significantly impacted healthcare [6]. Machine learning (ML) techniques, particularly deep learning (DL) [7], utilize artificial neural networks to process extensive datasets and identify important features that predict outcomes of interest. Convolutional neural networks (CNNs), a specific DL architecture, are algorithms specifically designed to process images [8]. CNNs have been widely used in classification tasks involving ultrasound imaging, demonstrating their ability to detect diverse patterns and subtle changes in this type of medical images [9]. Each CNN layer contains small matrices of weights, called filters - or kernels, which are applied to image pixels through a mathematical operation known as convolution. In this process, each filter slides over the input image,

performing element-wise multiplication and summing the results to produce a single value. This operation is repeated across the entire image, resulting in a new image known as a feature map, that emphasizes certain patterns or features detected by the filter. These kernels showed an excellent ability at detecting diverse patterns and changes in ultrasound images, enabling early and precise identification of kidney abnormalities [10]. Training deep learning networks can be challenging due to various factors, including the choice of network architecture, the depth of the network, and the specific parameters used. Different CNN models may perform differently even when trained for the same task on identical datasets; thus, a thorough evaluation is needed to determine the most effective one. AI has significantly advanced healthcare by automating complex tasks and analyzing large datasets; however, a major limitation in this domain is the difficulty in interpreting the models' outcomes. This lack of transparency can undermine trust and acceptance among healthcare professionals, who need to understand how decisions are made to confidently use AI in clinical practice. To address this issue, Explainable AI (XAI) techniques have been introduced, providing a way to enhance the interpretability of AI models. XAI techniques enable the creation of visual explanations of the models' predictions, making it easier for clinicians to understand the decision-making process.

By leveraging deep learning and XAI, this project attempted to improve the objective evaluation of fetal renal function, enabling for earlier and more accurate diagnosis with a consequent improvement of prenatal care and outcomes for children with congenital obstructive uropathies. This internship aimed to perform ultrasound image classification in obstructive uropathies by using a deep learning approach based on CNNs. Specifically, the project focused on addressing the following research questions:

1. How accurately can deep learning models predict fetal renal function based on ultrasound imaging?
2. Which deep learning framework performs best for the binary classification of fetal renal function using ultrasound images?
3. What are the strengths and limitations of using deep learning for predicting fetal renal function based on ultrasound imaging?
4. How can the integration of explainable artificial intelligence (XAI) techniques enhance the interpretation and decision-making processes in healthcare systems regarding fetal renal health using ultrasound imaging?

2. Materials

2.1 Data Description

Between 2013 and 2023 a retrospective, cross-sectional study was conducted at the Fetal Medicine units of the University Medical Center Groningen (UMCG). For this project, fetuses with unilateral obstructive uropathies were selected, resulting in a total of 97 cases. One ultrasound image was available for each case, which was selected by the gynecologist based on its quality and optimal display of the fetal renal pathology. Along each ultrasound image, segmentation masks of the kidney and renal pelvis (Figure 2) were created by the gynecologist. Each patient's ultrasound image was categorized into one of two outcome classes based on clinical criteria:

- “favorable outcome” (label 0): normal renogram (> 40% function of the affected kidney) and no history of surgical intervention related to the kidney problem

- “not favorable outcome” (label 1): abnormal renogram (≤ 40% function of the affected kidney) and/or a history of surgical intervention related to the kidney problem

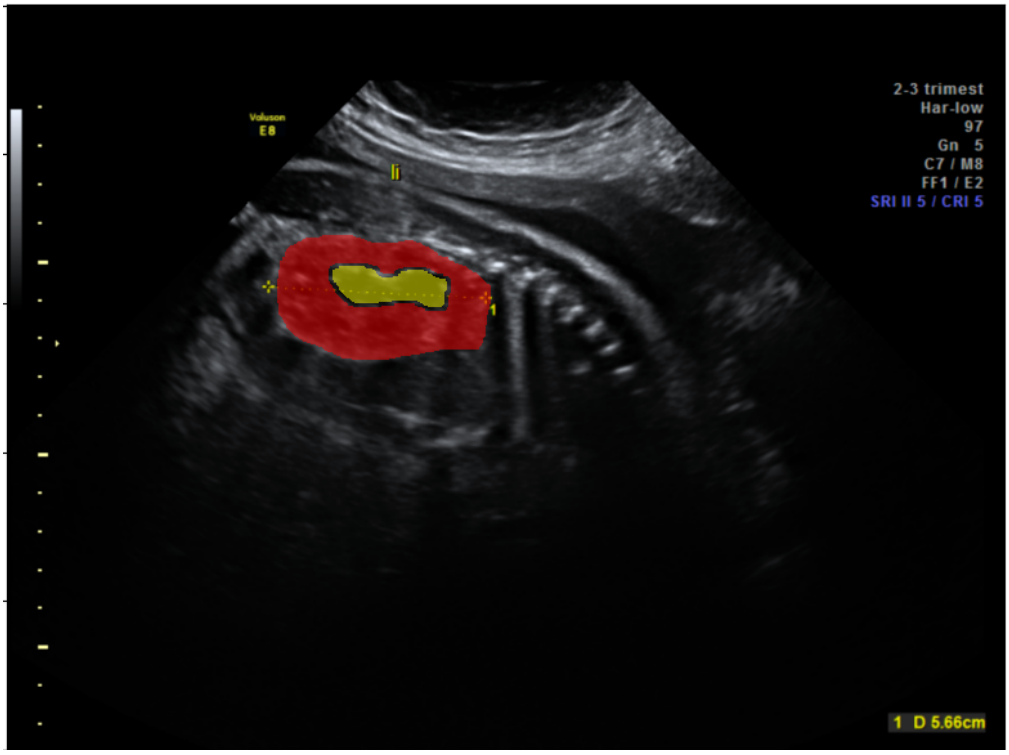

Figure 2. Example ultrasound image with segmentation masks, where the kidney is highlighted in red and the renal pelvis in yellow

2.2 Data Preprocessing

Preprocessing is a crucial step in the creation of datasets to use to train deep learning models, particularly for medical applications [11]. This process aims to standardize images, minimizing variations so the model focuses on relevant features. Effective preprocessing ensures the dataset is clean and consistent, enhancing the model's accuracy and reliability.

In the context of ultrasound imaging, preprocessing helps in managing various issues such as the presence of annotations, patient personal health information (PHI), and inconsistencies in image dimensions and quality. In the following subsections we will describe all the steps taken in this study to obtain the final dataset utilized for the analysis.

2.2.1 Data Import and Anonymization

The first step of data preprocessing involved importing all DICOM ultrasound images and image segmentations to establish an organized workflow. DICOM (Digital Imaging and Communications in Medicine) is a standard format for medical imaging that stores both the image data and extensive metadata, such as patient information and imaging parameters. Ultrasound raw images are stored in three channels, representing the RGB (Red, Green, Blue) color model, with each channel containing intensity values for its respective color. To protect PHI and anonymize the dataset, the pixel array along with relevant imaging parameters were extracted from the DICOM files, and the images were converted to NIfTI format, which does not retain the sensitive metadata. Next, all PHI present on the ultrasound images were removed using zero-covering (Figure 3). Finally the dataset was pseudonymized by renaming all images with a randomly generated number and split into training (~80%) and testing (~20%) subsets, maintaining the ratio between the two classes.

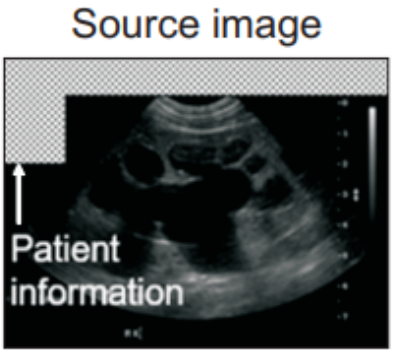

Figure 3. Visualization of PHI removal process using zero-covering [12]

2.2.2 Standardization of Image Dimensions

In medical imaging, pixel-spacing (PS) is defined as the physical distance in a patient between the centers of each two-dimensional pixel, specified by two numerical values (Figure 4): the vertical and horizontal pixel sizes [13]. In our dataset, each image had same pixel dimensions (default ultrasound machine output) but different zoom, which resulted in differences in the PS values across all patients. To address this issue, PS values were collected for all patients from the (0028,0030) tag of the DICOM header of the ultrasound images. These images were then resized based on the average PS across all patients to ensure uniformity.

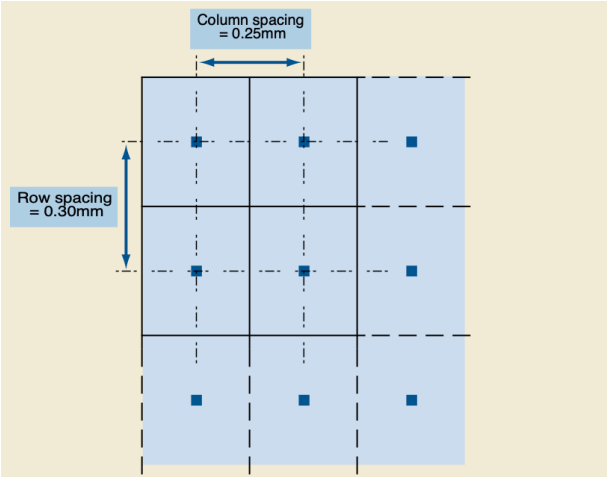

Figure 4. Example of pixel-spacing [13]

Subsequently, to ensure a consistent pixel size and eliminate unwanted elements, a standard region of interest (ROI) focused on the kidneys was established for the images. Width, height, and centers of mass of all kidneys were calculated using segmentation masks. The size of the bounding box was then defined by adding the margin to the maximum kidney dimensions across patients, to ensure that surrounding tissues were also included. Images smaller than the bounding box size were zero-padded [14] to match the required dimensions. As result, bounding boxes of a fixed size of (410, 350) pixels were extracted around the calculated centers of mass (Figure 5) per each ultrasound image.

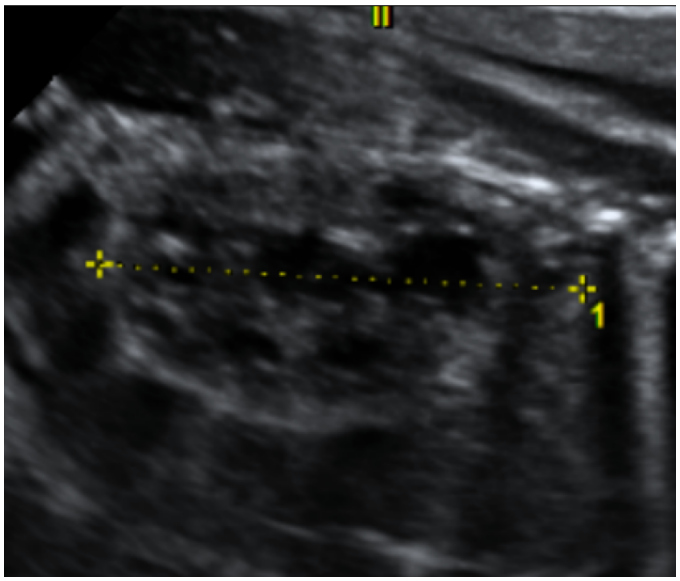

Figure 5. Example of an extracted bounding box from an ultrasound image

2.2.3 Image Inpainting

In the dataset, many images contained colored annotations, including calipers and texts, which could introduce bias during the model training process. To address the issue, we performed inpainting, a technique used in image processing to restore missing or corrupted parts of an image, by filling in the missing areas in a visually realistic manner. Inpainting is particularly useful in applications where the image structure is compromised by noise, annotations, or other unwanted artifacts.

OpenCV, a popular computer vision library, provides two inpainting algorithms: Navier-Stokes [15] and Telea [16]. The Navier-Stokes inpainting algorithm is based on fluid dynamics and aims to propagate linear structures (isophotes) from the boundary of the damaged area inwards, preserving the continuity of these structures. The Telea algorithm, on the other hand, uses a fast-marching method to gradually fill in the missing pixels from the boundaries towards the center, ensuring smoothness and coherence by considering the known pixels' information around the boundary. MagicInpaint [17] is an algorithm from an open-source inpainting library that uses variations of the Normalized Cross-Correlation calculation methods [18]. It fills in the pixels from the missing noisy regions using the data from the undamaged area in the same image, also called low noise area. This process involves assigning each pixel, or group of pixels, to the so-called image keys. These keys are extracted from the neighborhood of non-noisy pixels and are then used to find the best match for the corresponding noisy pixels.

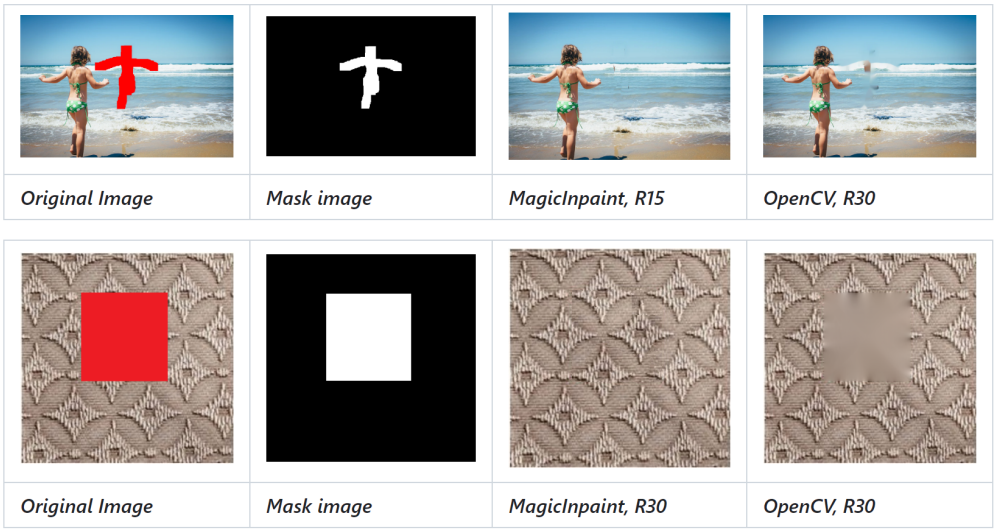

Figure 6. Example results of inpaint algorithms [17]

All three inpainting algorithms cited above require as input an image, the radius of a circular neighborhood, and a corresponding binary mask that indicates the areas to be inpainted (Figure 6). For each ultrasound image a mask was created using two different techniques, ensuring that the algorithms would effectively target the areas needing correction. The following subsections provide a detailed description of the mask creation process.

Color-Based Binary Mask Creation

Most of the colored annotations were yellow (RGB 255,255,0), with some exceptions in green (RGB 0,255,0). In order to easily isolate all of them at once, all green measurements were firstly converted to yellow by setting the red RGB value to 250. Subsequently, a binary mask was created based on the yellow color bounds (Figure 7). These bounds were carefully chosen through an iterative process to strike a balance between minimizing image noise and retaining important information.

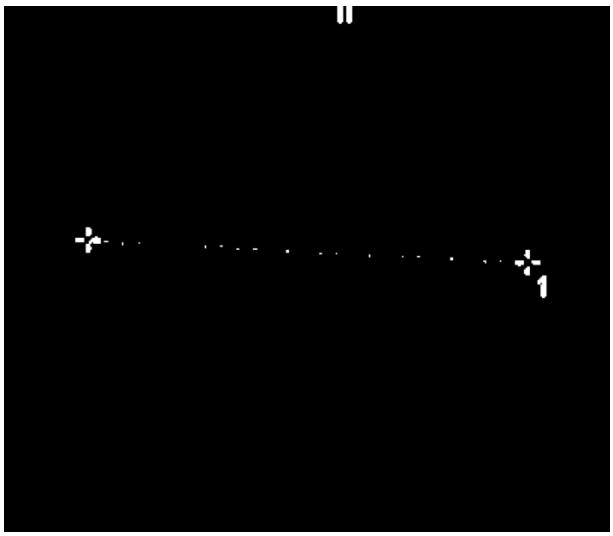

Figure 7. Example of a binary mask created based solely on yellow color bounds

The unwanted features in the images not only appeared as yellow annotations, but also included black shadows which complicated the masking process. To address this problem, the binary mask was dilated [19] with a 5x5 kernel (Figure 8). A custom dilation algorithm was also tested, which first detected the boundaries of the shadows and then expanded the mask accordingly. However, the 5x5 kernel showed better overall results.

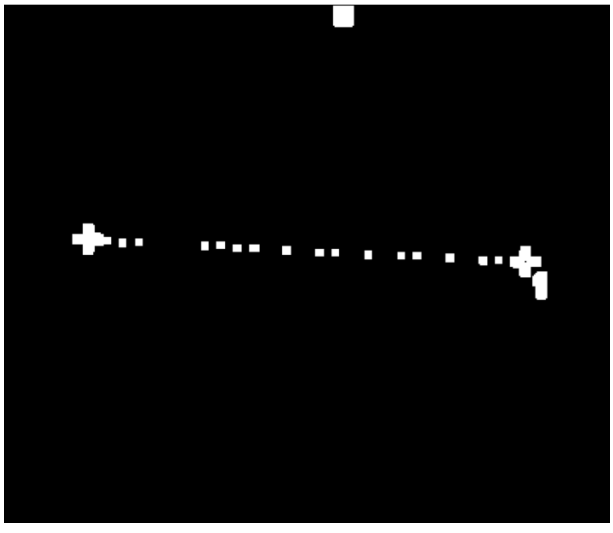

Figure 8. Example of a binary mask after the delation process

Edge-Detection Binary Mask Refinement

Some binary masks created with the logic described in the previous section either contained a lot of salt noise [20] or were missing small regions due to the faded yellow color on the input images. To address these problems, a set of improvement steps was developed.

Firstly, a Canny edge detection algorithm [21] was used on each RGB channel to identify measurement edges (Figure 9). Since each RGB channel contained different parts of the measurements, each Canny edge mask contained different segments. To combine these segments into one edge mask, a set of mathematical operations (addition and subtraction) was implemented (Figure 10). The resulted edge mask was dilated and used to remove noise from the binary mask obtained from the previous subsection. If both masks had a pixel labeled “1”, it was considered valid; otherwise, it was relabeled as “0”.

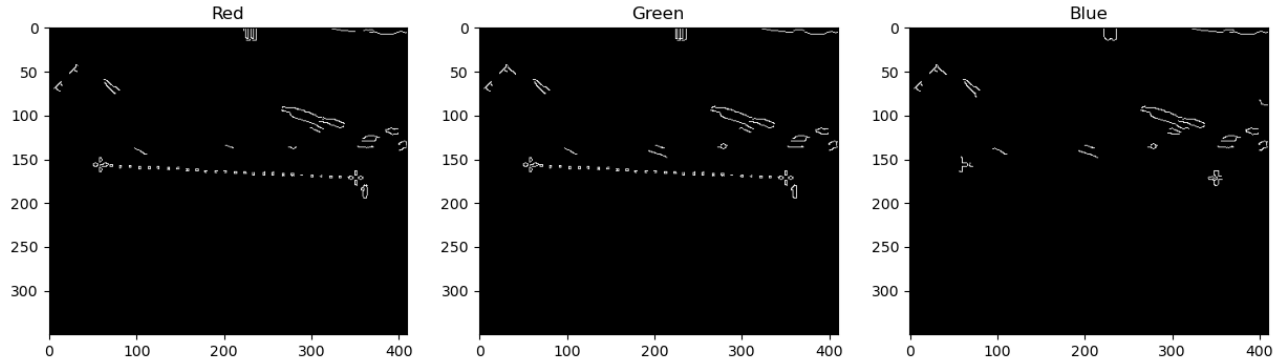

Figure 9. Result of a Canny edge detection algorithm

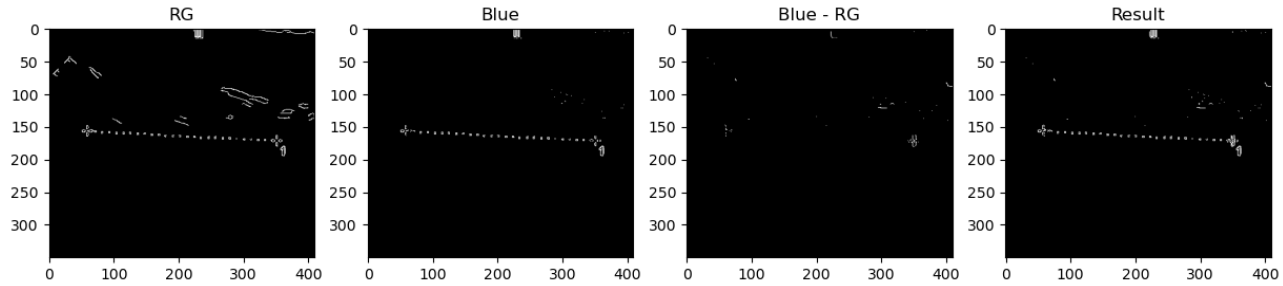

Figure 10. Measurements extracted from edges masks

Secondly, the resulted edge mask was combined with the - now cleaned - binary mask to fill in the missing regions. Some salt noise was introduced by the last step, therefore eroding [19] was performed on the final mask with a 3x3 kernel. The resulting final binary masks accurately detect all annotations present on the ultrasound images, including their shadows.

Inpainting results

Only the blue RGB channel of the ultrasound images was used as the input image for inpainting. In this channel, in fact, the yellow annotations appear black, since yellow consists of only red and green components. By using the blue channel, any annotations that were present on black pixel regions were effectively rendered invisible, improving the overall quality of the inpainted images. Additionally, by using only one channel, the overall image size was reduced by a factor of 3 to an 8-bit grayscale format. The MagicInpaint algorithm was selected as it visually demonstrated to maintain image quality and showed the best performance in blending accuracy, making it the preferred method for this specific preprocessing task.

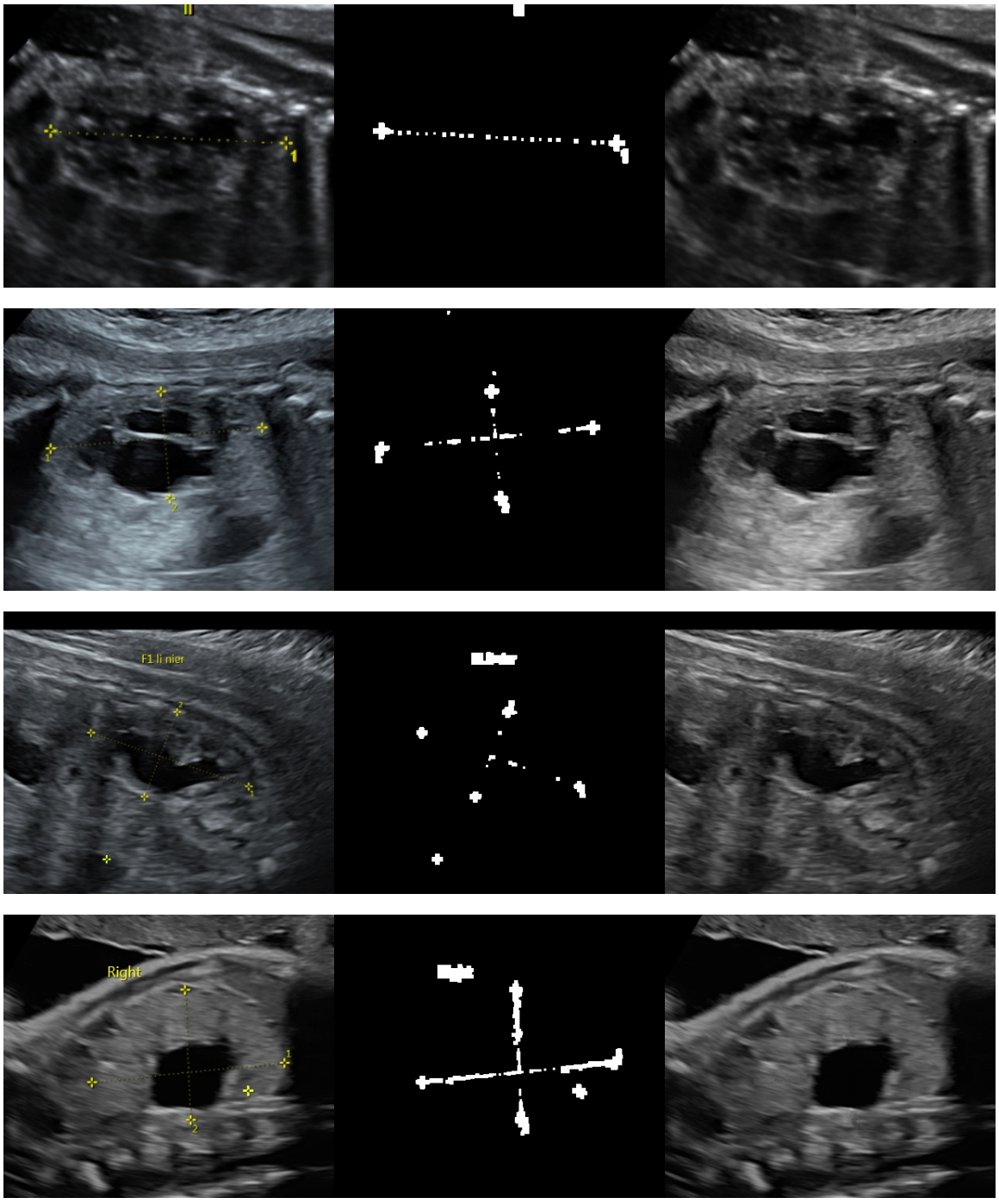

Figure 11. Example results of the preprocessing pipeline, with the original image on the left, the binary mask used for inpainting in the middle, and the resulted image on the right

3. Methods

Deep learning is a subset of machine learning that uses artificial neural networks with multiple layers to model complex patterns and relationships in data. Neural networks, particularly convolutional neural networks (CNNs) [22], are designed to process structured grid data, like images. CNNs are very effective in computer vision because they can learn hierarchical features from raw pixel values. This makes them suitable for tasks like image classification, object detection, and segmentation.

Supervised learning [23] involves training a model on labeled data, where the input data are paired with the correct output labels. The goal is to enable the model to accurately classify new, unseen images into these categories. In this project, we focus on a supervised learning binary classification task, aiming to predict fetal renal function based on 2D ultrasound images.

In the first part of this chapter, data augmentation techniques and the various model architectures used in this study are discussed. This is followed by an explanation of transfer learning and the process of hyperparameter tuning to optimize model performance. The final sections describe the application of explainable AI methods and outline the measures of performance used to evaluate the models.

3.1 Data augmentation

Data augmentation is a technique used to artificially expand the training dataset by applying various transformations to the input images. Only Random Rotation [24] was considered appropriate for this study, as other transformations, like flipping, could misrepresent anatomical structures (e.g., flipping could swap the left and right kidneys). Ultrasound images of fetuses' kidneys can be captured from various angles, therefore a random rotation layer was added to all pre-trained models after the input layer to improve the models' ability to handle rotations in the input images.

3.2 Model architectures

Based on a literature review of similar studies, four model architectures were selected: ResNet50 [25], EfficientNetB4 [26], VGG16 [27], and InceptionV3 [28]. Each of these architectures has unique strengths and weaknesses, making them suitable for different types of classification tasks. ResNet50 is renowned for its use of residual connections, allowing for the training of much deeper networks without a significant performance loss, although it demands high computational resources. EfficientNetB4 uses a compound scaling method that scales all dimensions of depth, width, and resolution uniformly, providing a balance between accuracy and computational cost, though due to its complex design implementation can be challenging. VGG16 is characterized by its simplicity, using small (3x3) convolution filters throughout the network, making it easy to implement but relatively large and computationally expensive. InceptionV3 uses inception modules that capture multi-scale features with filters of different sizes simultaneously, leading to a highly efficient model with excellent performance and fewer parameters, though its complexity can complicate implementation. Throughout this study, each architecture was evaluated to determine which one offered the best performance for predicting fetal renal function based on ultrasound images, aiming to identify the most effective model for this classification task.

3.3 Transfer learning

Transfer learning is a technique where a model developed for a particular task is reused as the starting point for a model on a second task. Fine-tuning is a technique that goes a step further by allowing the pre-trained model's parameters to be adjusted during training on the new dataset. These techniques offer significant benefits, such as reducing training time and improving model performance, as the pre-trained models have already learned useful features from a large amount of data. This is especially beneficial when working with small datasets, such as the one used in this study, as it helps to mitigate issues related to overfitting and insufficient training data.

All four models based on the architectures described in the previous section were pre-trained on the ImageNet dataset [29] and accessed from Keras [30], a popular deep learning API. Before they could be used for this study, two main challenges had to be overcome. First, the pre-trained models were configured to accept 3-channel (RGB) images as input, whereas our dataset consisted of single-channel (grayscale) images. Therefore, the weights of the first convolutional layer were averaged across the three channels to fit the single-channel input. Second, each pre-trained model included different preprocessing layers tailored to their original training, which were removed to ensure compatibility with our input data (Figure 12). Finally, the architectures' last dense layer was modified to suit the binary classification task. The output dense layer of 1000 nodes (corresponding to 1000 classes from the ImageNet dataset) from the pre-trained models was replaced with a dense layer of 2 nodes (one for each class) with a softmax activation function [31].

Model: "resnet50"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 350, 410, 3)	0	-
conv1_pad (ZeroPadding2D)	(None, 356, 416, 3)	0	input_layer[0][0]
conv1_conv (Conv2D)	(None, 175, 205, 64)	9,472	conv1_pad[0][0]
conv1_bn (BatchNormalizatio…	(None, 175, 205, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 175, 205, 64)	0	conv1_bn[0][0]
pool1_pad (ZeroPadding2D)	(None, 177, 207, 64)	0	conv1_relu[0][0]
pool1_pool (MaxPooling2D)	(None, 88, 103, 64)	0	pool1_pad[0][0]

Model: "resnet50_grayscale"

Layer (type)	Output Shape	Param #	Connected to
input_layer_3 (InputLayer)	(None, 350, 410, 1)	0	-
RandomRotation (RandomRotation)	(None, 350, 410, 1)	0	input_layer_3[0]…
conv1_conv (Conv2D)	(None, 172, 202, 64)	3,200	RandomRotation[0…
conv1_bn (BatchNormalizatio…	(None, 172, 202, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 172, 202, 64)	0	conv1_bn[0][0]
pool1_pad (ZeroPadding2D)	(None, 174, 204, 64)	0	conv1_relu[0][0]
pool1_pool (MaxPooling2D)	(None, 86, 101, 64)	0	pool1_pad[0][0]

Figure 12. Section of a ResNet50 architecture before (left) and after (right) adjustments

3.4 Hyperparameter tuning

Hyperparameters are settings that are not learned from the data but set before the training begins. These parameters control the learning process and significantly influence the model's performance during training. Hyperparameter tuning is the process of choosing a set of optimal hyperparameters that will result in the best model performance after training. For this study, three of these parameters were optimized: batch size, learning rate, and model architecture.

The first one, batch size, refers to the number of training samples processed in one iteration before the model's parameters are updated. A smaller batch size can lead to more frequent updates but may introduce more noise. On the contrary, a larger batch size makes the training faster, but may result in poorer generalization, leading to overfitting, where the model performs well on training data but fails to capture important patterns and nuances in unseen data.

The learning rate controls the size of the steps the model takes towards minimizing the loss function during training. A high learning rate can accelerate training but may overshoot the optimal solution, while a low learning rate ensures more precise updates but can cause the process to slow down or even stop making progress towards improving the model's performance.

The batch sizes of 2, 4, and 8 were chosen based on the dataset size, while learning rates of 1e-2, 1e-3, and 1e-4 were selected because these values are commonly used in optimizing neural network models across various tasks. A grid search was performed to systematically explore the combinations of these hyperparameters, resulting in 36 different configurations.

Cross-validation is a technique where the dataset is divided into k subsets or folds. In each round of cross-validation, one of these folds is held out as the validation set (also knows as the testing set), while the remaining $k - 1$ folds are used as the training set. This process is repeated k times, with each fold serving as the validation set exactly once. Therefore, each data point ends up in the validation set exactly once across all rounds of cross-validation. The goal of this technique is to assess how well the model generalizes to new, unseen data by averaging the performance across all folds. This is especially useful with small datasets, as it maximizes the use of limited data and provides a better estimate of the model's performance. In this study, five fold ($k = 5$) stratified cross-validation [32] was performed, which ensures that each fold maintains the same class distribution as the original dataset (Figure 13) in both subsets. Evaluation metrics for each fold were stored, then averaged and saved for later analysis. This method helps achieve a reliable assessment of model performance and helps select optimal hyperparameters.

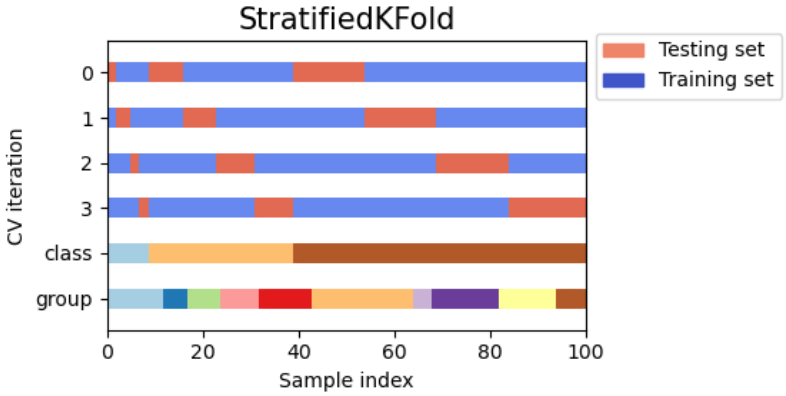

Figure 13. Stratified Cross Validation [32]

3.5 Model training

The training process of a deep learning model involves optimizing its parameters to minimize a specified loss function. In this study, categorical cross-entropy [33] (1) was used as the loss function, and the “Adam” optimizer [34] served to update the model's weights during training to achieve this minimization. The training process was conducted over 150 epochs, which define the number of complete passes through the entire training dataset. To prevent overfitting and ensure efficient training, an early stopping callback was implemented, targeting the validation loss. This feature monitors the specified metric, and if it does not improve after a certain number of epochs (referred to as "patience"), training is stopped and the model is reverted to the state with the best performance on the validation set. The training was performed on an external server in a virtual environment with an 8-core CPU and 56 GB of RAM [35].

$$Loss = -\sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\log(\hat{y}_{ij}) \quad (1)$$

,where y - actual values, ŷ - model predictions, N - number of samples, K - number of classes in the data

3.6 Explainable AI

Explainable AI (XAI) [36] refers to techniques and methods that enable human users to understand and trust the results and output created by machine learning algorithms. In the context of deep learning, especially in complex models like neural networks, the decision-making process can be quite difficult to comprehend; thus, it is often described as a "black box”. XAI aims to make these processes more transparent by providing insights into how the model arrived at its decisions. This is particularly crucial in the medical field, where the predictions made by the model can influence clinical decisions and patient outcomes. By making deep learning models more interpretable, clinicians can gain confidence in using these tools, ensure that these models are making decisions based on relevant medical information, and identify any potential biases or errors in the model's logic.

A popular technique for enhancing explainability in deep learning models, particularly CNNs, is Gradient-weighted Class Activation Mapping (Grad-CAM) [37]. Grad-CAM generates visual explanations (heatmaps) for predictions made by CNNs by highlighting the regions in the input image that were most influential for the model's decision.

The process of creating such explanations can be divided into three steps. It begins with feeding the input image into the trained CNN, then, as the image passes through multiple convolutional and pooling layers (together known as convolutional blocks), the CNN produces feature maps (Figure 14). These structures capture the presence or absence of specific features at different spatial locations in the image. Feature maps created on the lower layers may detect basic features like edges and corners, but as we go deeper, the spatial dimensions tend to decrease; thus, feature maps recognize more complex patterns, such as objects or abstract patterns specific to the given dataset.

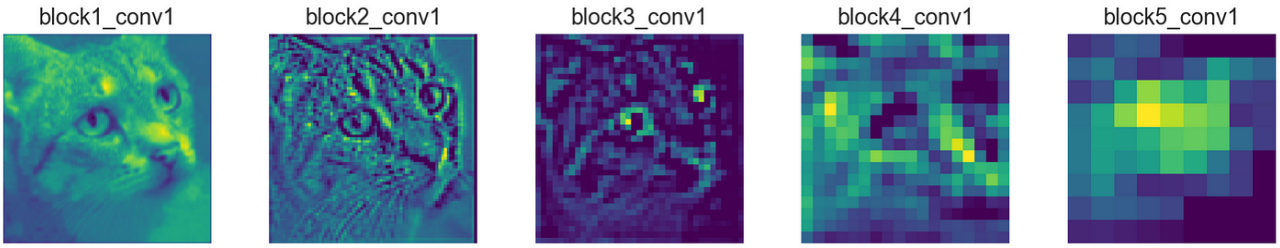

Figure 14. Example feature maps from deeper convolutional blocks [38]

Finally, Grad-CAM computes the gradient of the target class score (i.e., the model's predicted probability for the target class) with respect to the feature maps from a specific convolutional layer of interest (usually the last one). The gradients indicate how much each pixel in the feature maps influences the target class score. The computed gradients are then averaged over the spatial dimensions (height and width) of the feature maps to obtain importance weights, which reflect the significance of each feature map for the particular class being considered.

The Class Activation Map (CAM) is the weighted sum of the feature maps multiplied by their corresponding importance weights (Figure 15). The resulted heatmap is resized to match the dimensions of the input image and is often overlaid on top of the original image for visualization. Generating these images not only helps in validating the model's predictions but also may help clinicians to understand and interpret the results, potentially identifying new patterns or features that are clinically significant.

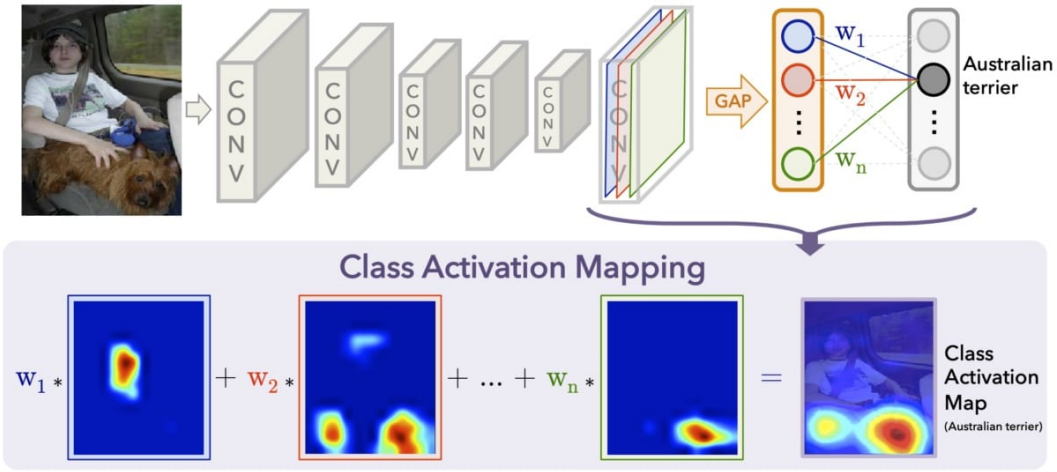

Figure 15. Example of a Gradient-weighted Class Activation Mapping process [39]

3.7 Measures of performance

In binary classification problems, such as the one in this project, data are classified into two possible classes (i.e., Positive and Negative). Each data point can be classified correctly (True) or incorrectly (False). The combination of these classifications forms a confusion matrix, which compares predicted and true classes. This matrix shows the number of correctly classified data points (True Positive - TP, or True Negative - TN) and incorrectly classified data points (False Positive - FP, or False Negative - FN). After each training loop, the following performance metrics based on the confusion matrix were calculated:

Accuracy, which measures the proportion of correctly predicted instances out of the total instances. It provides a general indication of the model's performance on unseen data, but it does not take into account how data is spread between TP and TN.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

Sensitivity (Recall), which measures the proportion of actual positives correctly identified by the model (true positives). High sensitivity indicates that most of the positive cases are correctly detected.

$$Sensitivity = \frac{TP}{TP + FN} \tag{3}$$

Specificity, which measures the proportion of actual negatives correctly identified by the model (true negatives). High specificity indicates that most of the negative cases are correctly detected.

$$Specificity = \frac{TN}{TN + FP} \tag{4}$$

F1 Score, which is the harmonic mean of precision (positive predictive value) and recall (sensitivity), and is ranged between 0 and 1. High F1 score indicates high classification performance. It is particularly useful when the class distribution is imbalanced, since it takes into account both FP and FN values.

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{5}$$

Area Under the Curve (AUC), which measures the area under the Receiver Operating Characteristic (ROC) curve. The ROC plots the True Positive Rate (TPR, or Recall) against the False Positive Rate (FPR) at various threshold settings. A higher AUC indicates better model performance in distinguishing between the two classes, as it signifies a higher TPR and a lower FPR across thresholds. Conversely, a lower AUC suggests poorer performance, indicating that the model struggles to effectively differentiate between the two classes.

$$AUC = \int_0^1 TPR(t)\, d(FPR(t)) \tag{6}$$

4. Results

In this section, the results obtained from hyperparameter tuning and from the models trained with the best hyperparameter selection are described. Hyperparameter tuning was performed for each model architecture to identify the optimal settings. After extensive evaluation, a batch size of 4 and a learning rate of 1e-4 were identified as the most optimal hyperparameters. This hyperparameter configuration was then used to train the three best-performing architectures. Additionally, during final evaluation, Grad-CAM heatmaps were generated to provide explanations for the predictions made by the trained models.

4.1 Hyperparameter tuning

A total of 180 models were trained, 36 hyperparameter combinations with 5-fold cross-validation. In order to visualize the hyperparameter tuning results, the metrics stored during training were uploaded to Weights and Biases (WandB) [40]. This platform is a powerful tool for tracking experiments and visualizing model performance across different hyperparameter configurations (Figure 16). This helped in identifying the optimal set of hyperparameters, by assessing their behavior throughout the training.

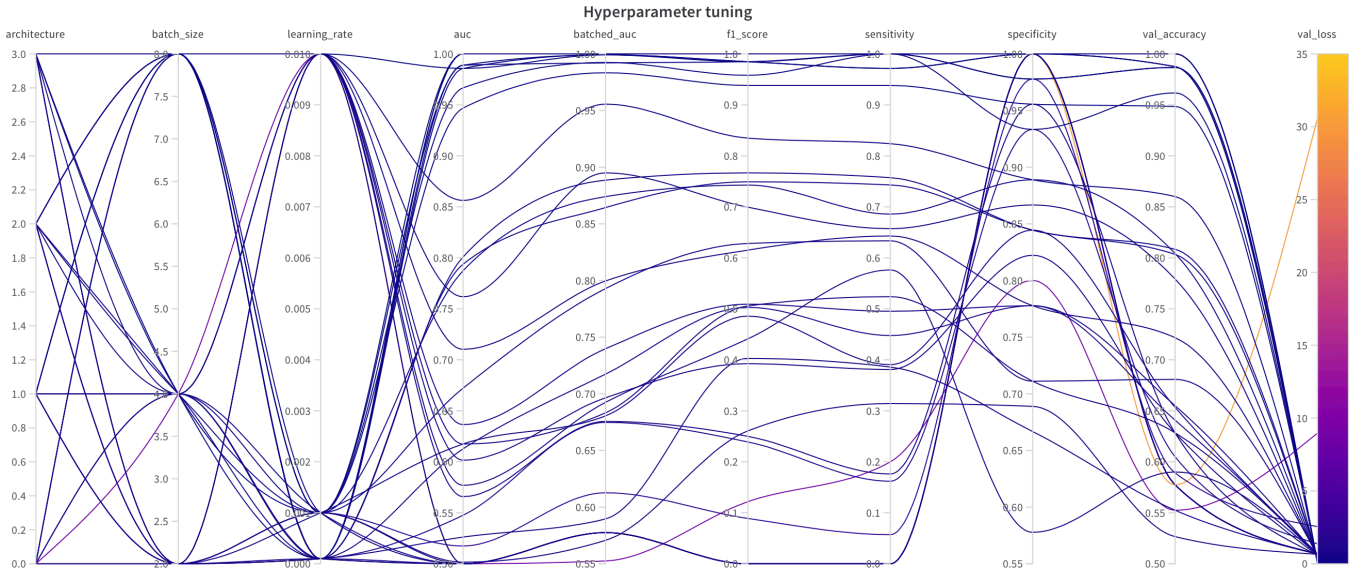

Figure 16. Plotted hyperparameter tuning results from WandB platform. Architecture "0.0" corresponds to VGG16, architecture "1.0" to ResNet50, architecture "2.0" to InceptionV3, and architecture "3.0" to EfficientNetB4.

The VGG16 model architecture and a batch size of 2 consistently underperformed and were therefore excluded from further analysis. For the remaining models, although validation metrics were close across multiple folds, a learning rate of 1e-4 consistently gave the lowest validation loss for most configurations. While all model architectures trained with a batch size of 8 showed good validation results, the validation subset contained only 16 images due to the 5-fold cross-validation, meaning that models were evaluated on just two batches. Therefore, a batch size of 4 was chosen as the optimal value to ensure a more stable and reliable evaluation of the model's performance.

4.2 Model training

After identifying the optimal set of hyperparameters based on the tuning results, the model training process was performed on the three best-performing architectures, with the number of folds decreased to 3, and the patience value for the early-stopping callback set to 20.

Table 1 summarizes the training results, presenting the averaged performance metrics with a 95% confidence interval (CI) calculated on the validation subset from all three cross-validation folds. The overall best-performing model architecture, EfficientNetB4, achieved a mean validation accuracy of 78.2% ± 14.7%, a mean F1 score of 74.91% ± 16.4%, a mean sensitivity of 74.29% ± 19.6%, a mean specificity of 84.44% ± 9.93%, and a mean AUC of 83.88% ± 13.3% (Table 1).

Architecture	Accuracy	F1 Score	Sensitivity	Specificity	AUC
ResNet50	0.5769 ± 0	0 ± 0	0 ± 0	1 ± 0	0.6036 ± 0.0213
InceptionV3	0.7436 ± 0.144	0.6993 ± 0.152	0.6571 ± 0.189	0.8667 ± 0.0729	0.8092 ± 0.119
EfficientNetB4	**0.782 ± 0.147**	**0.7491 ± 0.166**	**0.7429 ± 0.196**	**0.8444 ± 0.0993**	**0.8388 ± 0.133**

Table 1. Averaged performance metrics of the training process, showing a mean value with a 95% CI

The behavior of the accuracy and loss metrics on the validation subset throughout the training process is shown in Figure 17, with the shaded region around the line graph representing the standard deviation.

Figure 17. Plotted validation accuracy and loss, with the standard deviation, show the individual performance curves of these two metrics on the validation subset for each of the three best-performing model architectures

For the final independent evaluation, the test data subset (n = 19) was used. Predictions made by each model on the test subset, were averaged across the three cross-validation folds. Table 2 shows the mean prediction values for both the negative and positive classes for each considered architecture, along with their corresponding standard deviations (μ ± stdev).

	ResNet50		InceptionV3		EfficientNetB4	
	Neg	Pos	Neg	Pos	Neg	Pos
1	0.7166 ± 0.0728	0.2834 ± 0.0728	0.8021 ± 0.0687	0.1979 ± 0.0687	0.8179 ± 0.1885	0.1821 ± 0.1885
2	0.7145 ± 0.0732	0.2855 ± 0.0732	0.4217 ± 0.3941	0.5783 ± 0.3942	0.3719 ± 0.3335	0.6281 ± 0.3335
3	0.7139 ± 0.072	0.2861 ± 0.072	0.9766 ± 0.0114	0.0234 ± 0.0114	0.3895 ± 0.1748	0.6105 ± 0.1748
4	0.717 ± 0.0708	0.283 ± 0.0708	0.4652 ± 0.3261	0.5348 ± 0.3261	0.8094 ± 0.1172	0.1906 ± 0.1172
5	0.713 ± 0.0733	0.287 ± 0.0733	0.4307 ± 0.3114	0.5693 ± 0.3113	0.3243 ± 0.2454	0.6757 ± 0.2454
6	0.7182 ± 0.0712	0.2818 ± 0.0712	0.5353 ± 0.3636	0.4647 ± 0.3636	0.3021 ± 0.2337	0.6979 ± 0.2337
7	0.7174 ± 0.0712	0.2826 ± 0.0713	0.6841 ± 0.0979	0.3159 ± 0.0979	0.824 ± 0.1376	0.176 ± 0.1376
8	0.717 ± 0.0718	0.283 ± 0.0718	0.6233 ± 0.1855	0.377 ± 0.1855	0.3764 ± 0.2431	0.6236 ± 0.2431
9	0.7167 ± 0.072	0.2833 ± 0.072	0.8311 ± 0.1504	0.1689 ± 0.1504	0.4439 ± 0.08	0.5561 ± 0.0800
10	0.7163 ± 0.0724	0.2837 ± 0.0724	0.838 ± 0.1371	0.162 ± 0.1371	0.3363 ± 0.284	0.6637 ± 0.284
11	0.7157 ± 0.0722	0.2843 ± 0.0722	0.8075 ± 0.2203	0.1925 ± 0.2203	0.7935 ± 0.1488	0.2065 ± 0.1488
12	0.7191 ± 0.0728	0.2809 ± 0.0728	0.7354 ± 0.2927	0.2646 ± 0.2927	0.8652 ± 0.1	0.1348 ± 0.1
13	0.7124 ± 0.0741	0.2876 ± 0.0741	0.4039 ± 0.386	0.5961 ± 0.386	0.8383 ± 0.1123	0.1617 ± 0.1123
14	0.7185 ± 0.0713	0.2815 ± 0.0714	0.4743 ± 0.2094	0.5257 ± 0.2093	0.3851 ± 0.2490	0.6149 ± 0.249
15	0.7153 ± 0.0715	0.2847 ± 0.0715	0.4082 ± 0.2969	0.5918 ± 0.2969	0.7203 ± 0.1702	0.2797 ± 0.1702
16	0.718 ± 0.0715	0.282 ± 0.0715	0.714 ± 0.2364	0.286 ± 0.2364	0.7312 ± 0.1363	0.2688 ± 0.1363
17	0.7156 ± 0.0716	0.2844 ± 0.0716	0.5411 ± 0.3562	0.4589 ± 0.3562	0.3533 ± 0.2542	0.6467 ± 0.2542
18	0.7165 ± 0.0722	0.2835 ± 0.0722	0.2983 ± 0.296	0.7017 ± 0.2960	0.5247 ± 0.127	0.4753 ± 0.127
19	0.7167 ± 0.0722	0.2833 ± 0.0721	0.6225 ± 0.0853	0.3775 ± 0.0853	0.3349 ± 0.1128	0.6651 ± 0.1128

Table 2. Pairs of mean prediction values with standard deviations calculated across the three cross-validation folds. Each row corresponds to the same image from the test subset.

Table 3 summarizes the performance metrics of the final evaluation. The best-performing model, again based on EfficientNetB4, achieved a test accuracy of 68.42%, F1 score of 66.(6)%, sensitivity of 0.75 showing that the model correctly identified 75% of the positive cases, specificity of 0.6364 indicating a 63.64% correct identification of negative cases, and AUC of 69.32% (Table 3).

Architecture	Accuracy	F1 Score	Sensitivity	Specificity	AUC
ResNet50	0.5789	0.0	0.0	1.0	0.5
InceptionV3	0.6316	0.5333	0.5	0.7273	0.6136
EfficientNetB4	**0.6842**	**0.6666**	**0.75**	**0.6364**	**0.6932**

Table 3. Averaged performance metrics of the final evaluation on the held-out test subset

During the final evaluation GradCAM visual explanations were generated on the instances from the held-out test subset. The same images were used for each model architecture. Notably, the model based on the EfficientNetB4 architecture most accurately focused on the relevant regions, were other two models failed to do so (Figure 18).

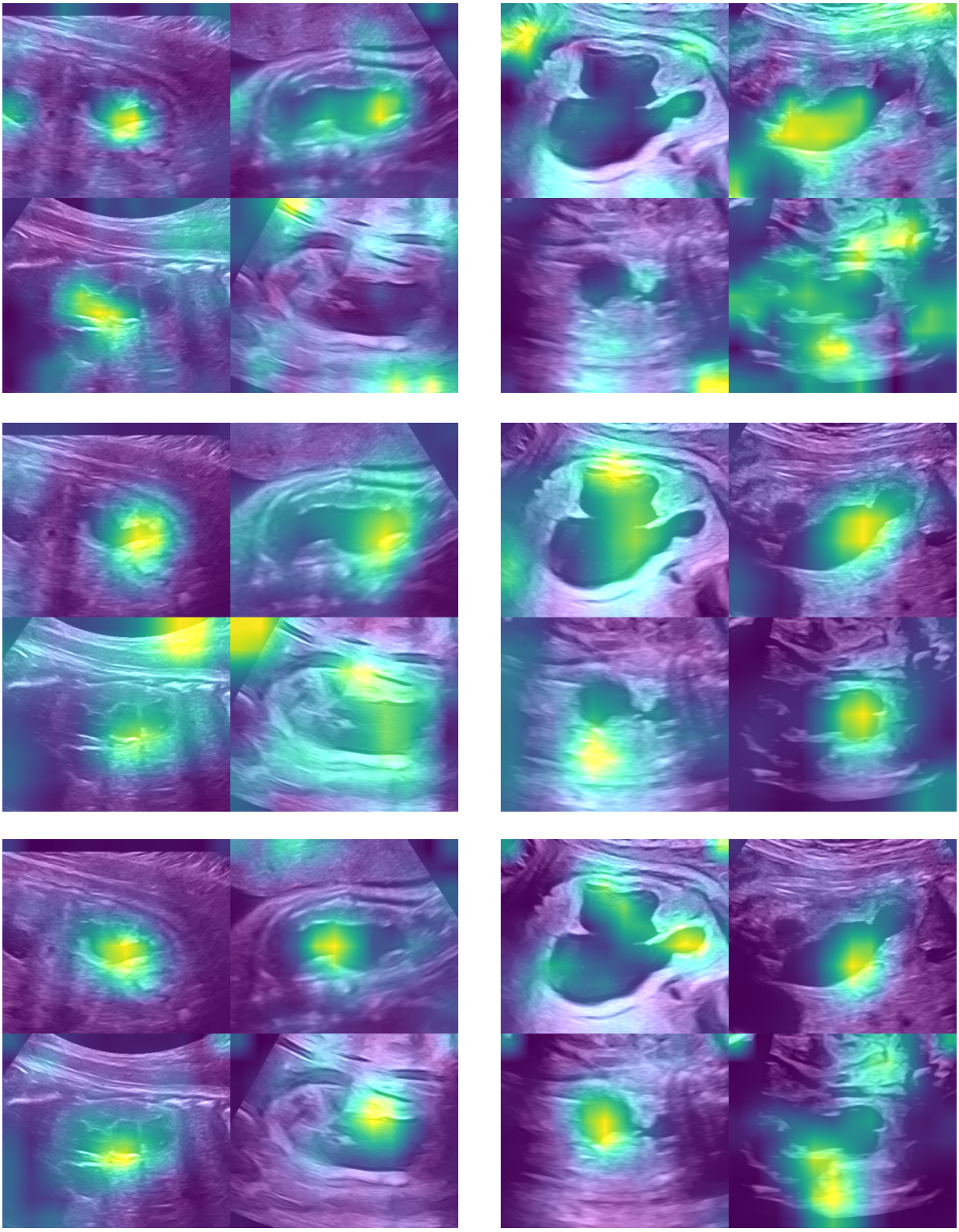

Figure 18. Generated GradCAM visual explanations of 4 random images from the negative (left) and positive (right) classes from the held-out test subset. First row correspond to the ResNet50 model, second to the InceptionV3, and third to the EfficientNetB4.

5. Discussion

The findings of this study demonstrate the potential of deep learning to improve the prediction of postnatal fetal renal function from ultrasound images, achieving notable performance metrics despite the dataset's limited size. The number of images sourced from the UMCG database is relatively small, which presents a limitation. A larger dataset, collected from the Amsterdam University Medical Centre (Amsterdam UMC) during the course of this project, could further validate our model's performance. However, due to time limitations, it could not be incorporated into this study. This expanded dataset would not only provide more diverse training examples but also enable fine-tuning of the model for enhanced accuracy. Additionally, it would allow for a comparison of the model's performance across different clinical settings, ensuring its applicability and reliability in broader clinical practice.

The created preprocessing pipeline demonstrated strong results. The inpainted areas of the images were hardly distinguishable to the naked eye. However, apart from a visual assessment, an attempt was made to numerically quantify the inpainting results to ensure an objective evaluation of the different inpainting algorithms. Several quantification methods were tested, including histogram-based analysis, Structural Similarity Index (SSIM), Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Local Binary Patterns (LBP) in both uniform and non-uniform configurations. These methods were chosen for their ability to measure various aspects of image quality: histogram-based analysis assesses changes in pixel intensity distribution; SSIM evaluates perceived changes in structural content; MSE and PSNR measure the average squared differences and peak signal quality, respectively; and LBP captures texture information (Figure 19). Although all these methods revealed changes in the images post-inpainting, they shared a significant limitation: the absence of a golden standard for comparison. Without a reference image of the ideal outcome, the evaluations could only compare images before and after inpainting, making it challenging to definitively quantify the effectiveness of each algorithm.

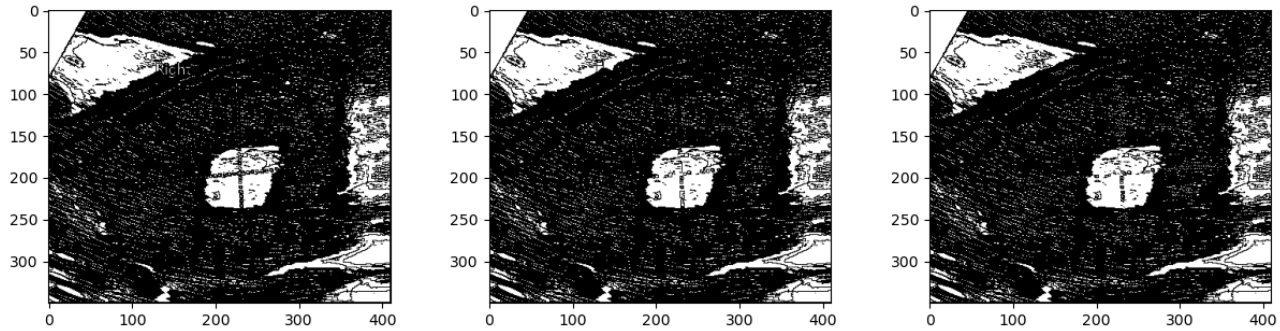

Figure 19. Example results of the LBP analysis

The best-performing model in this study reached a cross-validated accuracy of 68.42% on the test set, along with an F1 score of 66.67%, a sensitivity of 75%, a specificity of 63.64%, and an AUC of 69.32%. These results, although promising, are lower than the ones obtained on the validation set. The drop in the results, at least partially, is likely due to the small size of the test data subset. Because of that, even a single misclassification had a significant impact on the performance metrics. The mean prediction results show that ResNet50 consistently predicted the negative class with a low standard deviation. This suggests that the model did not learn enough features to effectively distinguish between the negative and positive classes. In contrast, InceptionV3 and EfficientNetB4 had more variability in their predictions. However, as indicated by the performance metrics, EfficientNetB4 had a lower standard deviation for each prediction, demonstrating that it is more stable and reliable in its predictions.

During the course of this project, we noted a new study [41] published by a Canadian research team that closely resembled ours. While their study focused on a different classification task, it involved similar preprocessing steps, such as anonymization and inpainting, and utilized a comparable approach to training and analyzing deep learning models. Despite achieving higher performance metrics, with 81.7% mean accuracy on the test set, their study benefited from a dataset ten times larger than ours. This comparison not only confirms that our preprocessing steps and modeling approach are effective but also suggests opportunities to improve our results further.

The selection of EfficientNetB4 as the best-performing model in our study was made based on the comparison of different architectures selected from the literature review. However, exploring different model architectures could potentially yield even better results. For instance, DenseNet169, which has been successfully used in the previously described similar study [41], offers a compelling alternative. DenseNet's architecture, characterized by densely connected layers, promotes feature reuse and mitigates the vanishing gradient problem, which could enhance performance on our classification task.

Grad-CAM heatmaps generated by the best-performing model on the test data demonstrated promising results. These visual explanations reveal the model's effective focus on clinically relevant renal areas associated with renal pathology when applied to a previously unseen test set. This shows that the model's predictions are based on important clinical details, enhancing the interpretability and trustworthiness of its predictions. While Grad-CAM provided valuable visualizations, exploring alternative CAM algorithms such as HiResCAM could offer additional insights. HiResCAM, known for its higher resolution and precise heatmap generation, has the potential to further improve the interpretability of our models. By offering clearer and more detailed visual explanations, HiResCAM could assist clinicians in better understanding the model's decision-making process, particularly in identifying crucial features in ultrasound images.

Due to time limitations, certain steps were not feasible during this study. Future work could involve preprocessing the Amsterdam UMC dataset, so it could be used in fine-tuning and cross-institutional validation of the models. Additionally, exploring different architectures, such as DenseNet169, could further enhance model performance and provide insights into optimal architectural choices for similar tasks in fetal renal function prediction.

6. Conclusions

How accurately can deep learning models predict fetal renal function based on ultrasound imaging?

This study demonstrated that deep learning models have potential for predicting postnatal fetal renal function based on ultrasound imaging, but with varying degrees of accuracy. The best-performing model achieved an accuracy of 68%, which indicates that while it has the ability to predict renal function from ultrasound images above a random chance, there is still room for improvement. The best model's performance highlights the complexity of the task and suggests that further work with a possibly larger and more diverse dataset could enhance predictive accuracy. Despite the current limitations, the application of deep learning in this context provides a promising step toward more objective, accurate and automated assessment methods in prenatal care.

Which deep learning framework performs best for the binary classification of fetal renal function using ultrasound images?

Among the various deep learning frameworks evaluated in this study, EfficientNetB4 emerged as the best-performing framework for the given binary classification task. The model based on this architecture achieved the highest accuracy and F1 score, surpassing other architectures such as VGG16, ResNet50, and InceptionV3. Additionally, the XAI GradCAM visual explanations generated for the EfficientNetB4 model demonstrated that it consistently focused on the relevant areas in the ultrasound images, where the other models failed to do so.

What are the strengths and limitations of using deep learning for predicting fetal renal function based on ultrasound imaging?

Deep learning models have shown promising results in predicting fetal renal function, providing objective assessments that can reduce human error and variability in diagnosis. These models can learn and extract relevant features from raw ultrasound images, capturing subtle differences in texture and structure that may be indicative of postnatal renal function. Once trained, they can process and analyze images quickly, offering faster diagnostic insights compared to manual evaluations and making them suitable for large-scale screening programs and studies. However, deep learning models require large amounts of data for training to achieve high accuracy. In this study, the small dataset size posed a significant challenge, potentially limiting the model's generalizability and performance. Furthermore, models trained on data from one clinical center may not perform well on data from different centers due to variations in imaging protocols and equipment; thus, cross-institutional validation is crucial. In conclusion, while deep learning offers high potential for enhancing the prediction of postnatal fetal renal function from ultrasound images, addressing its limitations, particularly concerning data availability and diversity, is essential for its successful integration into clinical practice.

How can the integration of explainable artificial intelligence (XAI) techniques enhance the interpretation and decision-making processes in healthcare systems regarding fetal renal health using ultrasound imaging?

Integrating explainable artificial intelligence (XAI) techniques, such as Grad-CAM visual explanations of the model's predictions, allows healthcare professionals to understand and trust the model's decision-making process, reducing the "black box" nature of deep learning models. By clearly showing which areas of the kidney are influencing the prediction, XAI can help clinicians verify that the model is focusing on medically relevant features, ensuring that the predictions align with clinical knowledge and expertise. This transparency not only aids in validating the model's reliability but also provides valuable insights that can guide further clinical investigations and interventions.

References

[1] Morris RK, Kilby MD. Congenital urinary tract obstruction. Best Pract Res Clin Obstet Gynaecol. 2008;22(1):97–122

[2] Cheung KW, Morris RK, Kilby MD. Congenital urinary tract obstruction. Best Pract Res Cl Ob. 2019;58:78-92

[3] Capone V, Persico N, Berrettini A, Decramer S, De Marco EA, De Palma D, et al. Definition, diagnosis and management of fetal lower urinary tract obstruction: consensus of the ERKNet CAKUT-Obstructive Uropathy Work Group. Nature Reviews Urology. 2022;19(5):295-303

[4] Pediatric Nephrology
https://mosaiques-diagnostics.de/mosaiques-diagnostics/Pediatric-Nephrology

[5] Grandjean H, Larroque D, Levi S. The performance of routine ultrasonographic screening of pregnancies in the Eurofetus Study. Am J Obstet Gynecol. 1999;181(2):446-54

[6] Shen D, Wu G, Suk H-I. Deep learning in medical image analysis. Annu. Rev. Biomed. Eng. 2017;19:221–248. doi: 10.1146/annurev-bioeng-071516-044442

[7] Hinton G. Deep learning—a technology with the potential to transform health care. Jama. 2018;320:1101–1102. doi: 10.1001/jama.2018.11100

[8] Soffer S, et al. Convolutional neural networks for radiologic images: A radiologist's guide. Radiology. 2019;290:590–606. doi: 10.1148/radiol.2018180547

[9] Cheng, P.M., Malhi, H.S. Transfer Learning with Convolutional Neural Networks for Classification of Abdominal Ultrasound Images. *J Digit Imaging* **30**, 234–243 (2017). https://doi.org/10.1007/s10278-016-9929-2

[10] Sudharson S, Kokil P, An ensemble of deep neural networks for kidney ultrasound image classification, Computer Methods and Programs in Biomedicine, Volume 197, 2020, 105709, ISSN 0169-2607, https://doi.org/10.1016/j.cmpb.2020.105709

[11] Get Started with Image Preprocessing and Augmentation for Deep Learning
https://www.mathworks.com/help/images/get-started-with-image-preprocessing-and-augmentation-for-deep-learning.html

[12] Song SH, Han JH, Kim KS, Cho YA, Youn HJ, Kim YI, Kweon J. Deep-learning segmentation of ultrasound images for automated calculation of the hydronephrosis area to renal parenchyma ratio. Investig Clin Urol. 2022 Jul;63(4):455-463. doi: 10.4111/icu.20220085. Epub 2022 May 25. PMID: 35670007; PMCID: PMC9262488.

[13] Pixel Spacing Attribute
https://dicom.innolitics.com/ciods/ultrasound-image/image-pixel/00280034

[14] Zero-padding
https://numpy.org/doc/stable/reference/generated/numpy.pad.html

[15] Bertalmio, Marcelo, Andrea L. Bertozzi, and Guillermo Sapiro. "Navier-stokes, fluid dynamics, and image and video inpainting." In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, vol. 1, pp. I-355. IEEE, 2001.

[16] Telea, Alexandru. "An image inpainting technique based on the fast marching method." Journal of graphics tools 9.1 (2004): 23-34.

[17] Anton Milev, MagicInpaint - image processing Python
https://github.com/antonmilev/magicinpaintpython?tab=readme-ov-file

[18] Briechle K, Hanebeck U. Template matching using fast normalized cross correlation. 2001/03/20, doi: 10.1117/12.421129 Proceedings of SPIE - The International Society for Optical Engineering 4387

[19] Morphological transformations
https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html

[20] Maity A, Chatterjee R. Impulsive noise in images: a brief review. ACCENTS Transactions on Image Processing and Computer Vision, Vol 4(10) ISSN (Online): 2455-4707 http://dx.doi.org/10.19101/TIPCV.2017.39025

[21] Canny edge detection
https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html

[22] Venkatesan, R., & Li, B. (2017). Convolutional Neural Networks in Visual Computing: A Concise Guide (1st ed.). CRC Press. https://doi.org/10.4324/9781315154282

[23] Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar (2012) Foundations of Machine Learning, The MIT Press ISBN 9780262018258.

[24] Random Rotation Geometrical Operation
https://keras.io/api/layers/preprocessing_layers/image_augmentation/random_rotation/

[25] He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition
https://doi.org/10.48550/arXiv.1512.03385

[26] Tan M, Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks
https://doi.org/10.48550/arXiv.1905.11946

[27] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition
https://doi.org/10.48550/arXiv.1409.1556

[28] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the Inception Architecture for Computer Vision
https://doi.org/10.48550/arXiv.1512.00567

[29] J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

[30] Keras - Multi-framework Deep Learning API
https://keras.io/about/

[31] Softmax Activation
https://en.wikipedia.org/wiki/Softmax_function

[32] Cross-validation: evaluating estimator performance, SciKit Learn
https://scikit-learn.org/stable/modules/cross_validation.html

[33] Categorical Cross-Entropy Loss
https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy

[34] Kingma D, Ba J. Adam: A Method for Stochastic Optimization
https://doi.org/10.48550/arXiv.1412.6980

[35] anDREa Research Environment
https://andrea-cloud.com

[36] van der Velden B, Kuijf H, Gilhuijs K, Viergever M. Explainable artificial intelligence (XAI) in deep learning-based medical image analysis, Medical Image Analysis, Volume 79, 2022, 102470, ISSN 1361-8415
https://doi.org/10.1016/j.media.2022.102470

[37] Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R. and Parikh, D., Das (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization, In Proc. of ICCV (pp. 618-626)

[38] Dertat A, Applied Deep Learning - Part 4: Convolutional Neural Networks
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

[39] Zhou B, Khosla A, Lapedriza A, Oliva A, Torralba A. Learning Deep Features for Discriminative Localization
https://doi.org/10.48550/arXiv.1512.04150

[40] Weights & Biases
https://wandb.ai/site

[41] Miguel OX, Kaczmarek E, Lee I, Ducharme R, Dingwall-Harvey ALJ, Rennicks White R, Bonin B, Aviv RI, Hawken S, Armour CM, Dick K, Walker MC. Deep learning prediction of renal anomalies for prenatal ultrasound diagnosis. Sci Rep. 2024 Apr 19;14(1):9013. doi: 10.1038/s41598-024-59248-4. PMID: 38641713; PMCID: PMC11031588