Fetal Brain Development Monitoring using Ultrasound Images: A U-Net based approach

Jai Tyagi[1] **Manya Goyal**[2] **Rahul Baliyan**[3]
Bhagwan Parshuram Institute Of Technology[1,2,3]

Abstract

Monitoring fetal brain development is essential for detecting abnormalities early in pregnancy. Ultrasound imaging is widely used for this purpose due to its safety and real-time capabilities. In this paper, we present a method that uses the U-Net architecture to analyze ultrasound images of the fetal brain. U-Net is effective in segmenting medical images, which helps in identifying important structures in the brain. By training the model on ultrasound data, we aim to improve the detection of developmental issues in the fetal brain. Our approach offers a non-invasive way to assist medical professionals in making timely decisions.

1 Introduction

Fetal brain development is a critical aspect of prenatal health, as early detection of abnormalities can significantly improve outcomes for both the mother and child. Ultrasound imaging is the most common tool used in obstetrics for monitoring fetal growth and development due to its non-invasive nature and widespread availability. The brain is one of the most complex organs to monitor during development, and identifying anomalies early on can help in diagnosing conditions like neural tube defects, ventriculomegaly, and other structural brain disorders [**?**]

Recent advancements in medical imaging and deep learning have opened the door to automating the analysis of ultrasound images. Traditional methods often rely heavily on manual interpretation by specialists, which can lead to variability and subjective bias in the diagnosis [**?**] This has led to the adoption of machine learning methods, which offer more consistent and accurate results. In particular, convolutional neural networks (CNNs) have shown great promise in image segmentation tasks, helping in the precise identification of structures within the fetal brain [**?**]

The U-Net architecture, specifically designed for medical image segmentation, has proven to be highly effective in delineating complex structures from imaging data [**?**] Its ability to work with limited data and produce accurate segmentations makes it an ideal candidate for fetal brain analysis using ultrasound images. In this study, we focus on leveraging U-Net to automatically segment the fetal brain in ultrasound scans. By automating this process, we aim to reduce the reliance on manual interpretation and improve early detection of developmental disorders.

Our approach is intended to enhance the current standard of care by providing an additional tool for medical professionals, allowing them to make faster, more informed decisions. This paper outlines our methodology, experimental results, and the potential applications of this system in clinical settings.

2 U-Net in Ultrasound Analysis

In this section, we explain the key reasons behind selecting the U-Net architecture for fetal brain segmentation from ultrasound images. U-Net has gained popularity in medical imaging due to its

specific design for segmentation tasks, especially when data is limited and high precision is required. Below, we break down the main factors influencing this choice.

2.1 Specialized for Medical Image Segmentation

U-Net was developed with medical image segmentation in mind [**?**] Unlike typical CNN architectures, U-Net's design focuses on producing high-quality segmentation maps by combining both local and global context from an image. This is particularly important when working with ultrasound images, which often have low contrast and noisy details. U-Net's "U-shaped" architecture, with its encoder-decoder design, allows it to extract features and then recover spatial resolution to predict detailed masks for segmentation [**?**] This capability is ideal for accurately identifying key regions in the fetal brain.

It became imperative to seek solutions that could help decrease the training duration while permitting the neural network to adapt to a relatively smaller dataset without compromising accuracy and training efficiency. Given GoogLeNet's advocacy for a neural network with an extensive amount of parameters, our primary concern was to prevent overfitting on the provided data. To mitigate the overfitting issue and reduce training time, we experimented with the integration of batch normalization [2] to the network. Similar applications such as this one can be found in the works of [7, 9, 10].

2.2 Handling Limited Data Effectively

In medical applications, large labeled datasets are often hard to obtain. The U-Net architecture addresses this by being highly efficient, even when trained on small datasets. Its ability to generate accurate segmentations without the need for vast amounts of data makes it a strong choice for our application, where the availability of labeled fetal brain ultrasound images is limited [**?**] Moreover, U-Net applies data augmentation techniques during training, allowing it to learn more robust patterns from fewer images.
The working of the Batch Normalization is as follows :

Inputs	Values of x over a mini-batch: B = $\{x_{1....m}\}$; Parameters to be learned: $\boldsymbol{\gamma}, \boldsymbol{\beta}$
Outputs	$\{y_i = \mathbf{BN}_{\gamma,\beta}(x_i)\}$
Mini-Batch Mean	$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$
Mini-Batch Variance	$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$
Normalize	$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$
Scale and Shift	$y_i \leftarrow \boldsymbol{\gamma}\hat{x}_i + \boldsymbol{\beta} \equiv \mathbf{BN}_{\gamma,\beta}(x_i)$

Table 1: Batch Normalizing Transform, applied to activation x over a mini-batch

It has been proven to significantly enhance convergence rates without introducing overfitting, as demonstrated in [2]. Thus, in our pursuit of training the network to achieve notable advancements in specific application domains, we turned to batch normalization as a viable technique for implementation in our network.

There are several prior works which have inspired our use of batch normalization in this work. The works of [9, 10] provided a fair incentive for the usage of batch normalization. As well as the works of [11, 12, 13, 14] provided motivation for using convolutional architectures for the problem at hand.

Our study introduces three neural network architectures, differing in the number of batch normalization layers. The three proposed architectures are as follows: we initially conduct experiments with our modified GoogLeNet, which lacks batch normalization layers. Subsequently, we repeat the experimentation with successive models, gradually increasing the number of batch normalization layers while maintaining a constant network parameter count of ∼8.3M.

A simple representation of batch normalization for a minibatch $\mathcal{B}$, and input $\mathbf{x} \in \mathcal{B}$ to Batch Normalization (BN). And thus, so the batch normalization will be defined as [3] :

$$\mathrm{BN}(\mathbf{x}) = \boldsymbol{\gamma} \odot \frac{\mathbf{x} - \hat{\boldsymbol{\mu}}_{\mathcal{B}}}{\hat{\boldsymbol{\sigma}}_{\mathcal{B}}} + \boldsymbol{\beta}.$$

2.3 Preserving Fine Details

One of the main challenges in fetal brain segmentation is capturing the fine details of brain structures from ultrasound images, which may have low resolution. U-Net incorporates skip connections between its encoder and decoder parts, enabling the model to retain detailed features from the earlier layers [**?**] This helps ensure that even the small and subtle structures of the fetal brain are accurately segmented. The skip connections are critical in preserving both high-level and low-level features during the upsampling process.

1. The increased size of the resources needs increased computation.
2. Due to limited computational budget, efficient distribution of resources is required.

To solve both of the problems, the Inception module is used as a sparse alternative for the fully connected architecture of the neural network which is connected through multiple layers in between. For solving these issues, and add the element of sparsity, the Inception module is used in the network containing convolutional and pooling layers which are concatenated in the end. In the original work on the GoogLeNet both the naïve and the dimension reduction forms are present [1].

2.4 Flexibility and Extensibility

U-Net is highly flexible and can be adapted to different medical imaging tasks beyond just segmentation. The architecture can be easily modified and extended to incorporate different types of input data (e.g., 2D or 3D images), which is beneficial in case we want to expand the model to include different imaging modalities or datasets in future work [**?**] This flexibility ensures that U-Net can be tailored to specific challenges posed by fetal brain ultrasound images.

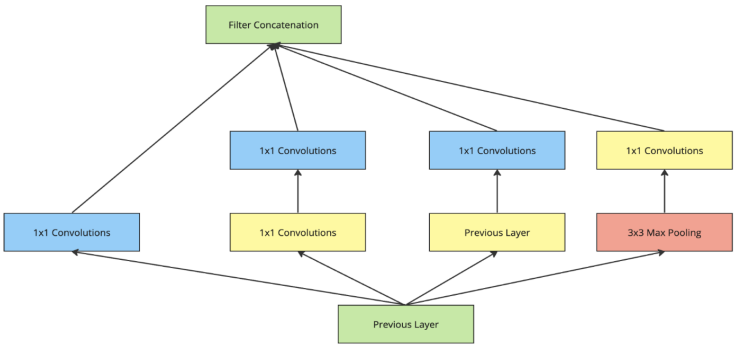

Figure 1: Inception Module

The other two different Inception Modules we have used in our experiments are implementations of the same along with layers of batch normalization. The first implementation BN GoogLeNet contains 2 batch normalization layers on top of the computaionally expensive 3 x 3 and 5 x 5

convolutions, and the ExpGoogLeNet contains 4 batch normalization layers each before the final concatenation of the layers.

Both of the Inception modules are designed as follows :

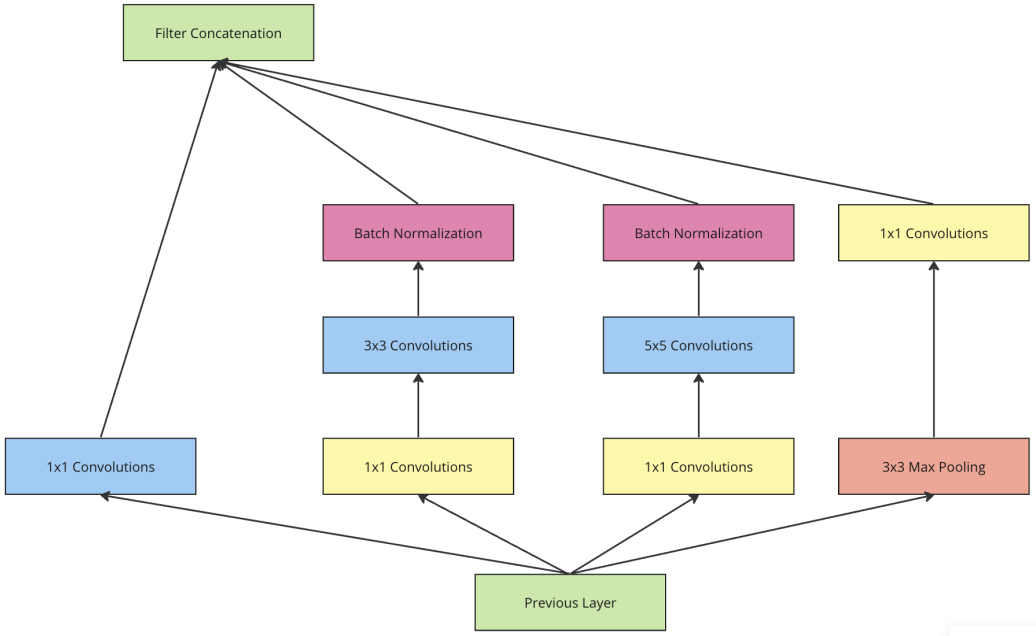

Figure 2: BN GoogLeNet Inception Module

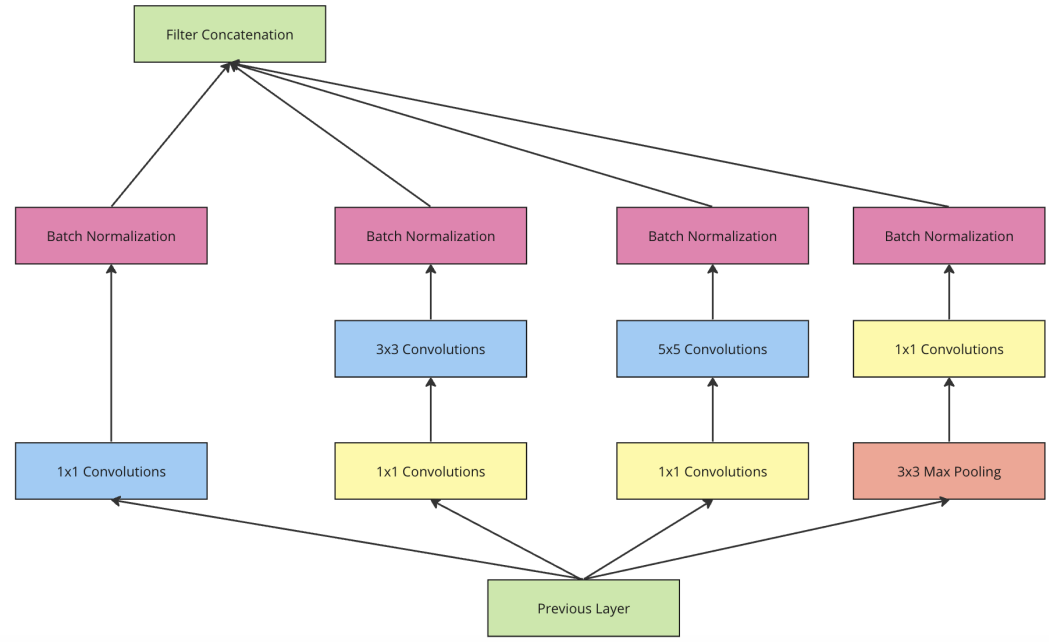

Figure 3: ExpGoogLeNet Inception Module

2.5 Strong Performance in Previous Studies

U-Net has consistently demonstrated superior performance in medical image segmentation tasks, including brain tissue segmentation in MRI and CT scans [**?**] Given its success in identifying complex anatomical structures in other medical imaging domains, it provides a strong foundation for

applying similar techniques to fetal brain ultrasound images. This past performance makes U-Net a reliable choice for our study.

3 Dataset

The dataset consists of ultrasound images of fetal brains, along with corresponding annotations for segmentation tasks. The data is split into training, validation, and test sets to ensure the model is properly trained and evaluated.

3.1 Overview of the dataset :

The dataset comprises a total of 2,333 images, of which 1334 are ultrasound images each corresponding to a unique subject. These images capture fetal brain development ranging from gestational age of 20 to 30 weeks. The dataset is carefully annotated to assist in training the U-Net model for segmentation:

- Total Images: 2,333
- Subjects: 1334
- Gestational Age: 20 to 30 weeks
- Training Images: 1,998 (999 images with corresponding 999 annotations)
- Test Images: 335

The dataset has been made available on Kaggle for open source use [5, 6].

The classification of the number images and classes in the dataset are as follows :

Class	**Number Of Images**
Mild Demented	8960
Moderate Demented	6464
Non Demented	9600
Very Mild Demented	8960

Table 2: Overview Of Distribution Of MRI Images

Along with this a bar chart describing the classes and images is shown as follows :

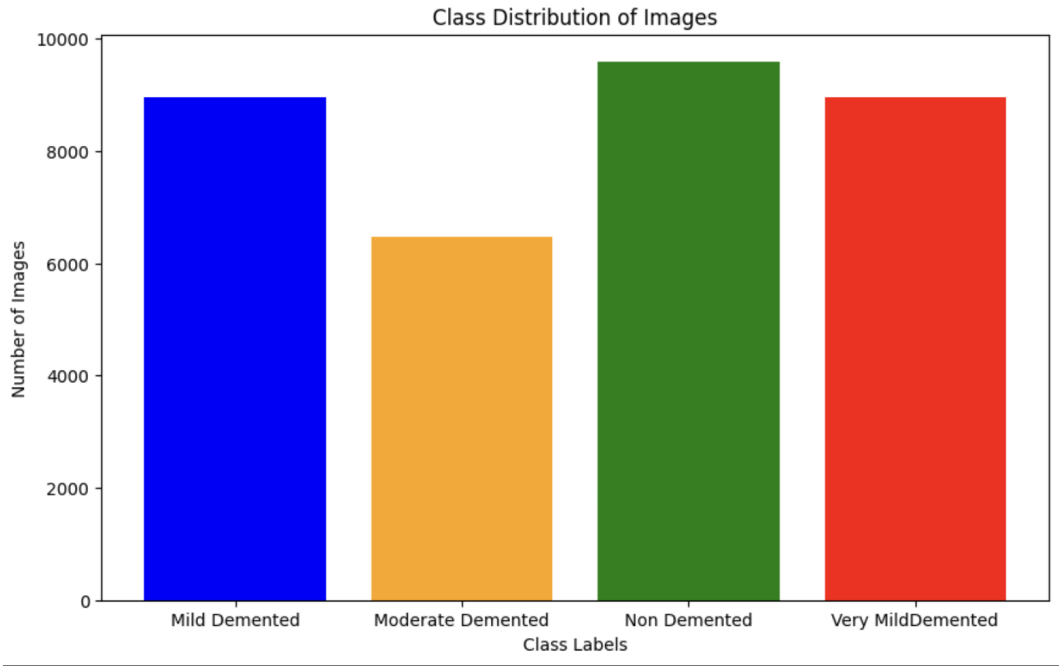

Figure 4: Distribution Of Images in the dataset

Along with this, some sample images from the dataset are as follows :

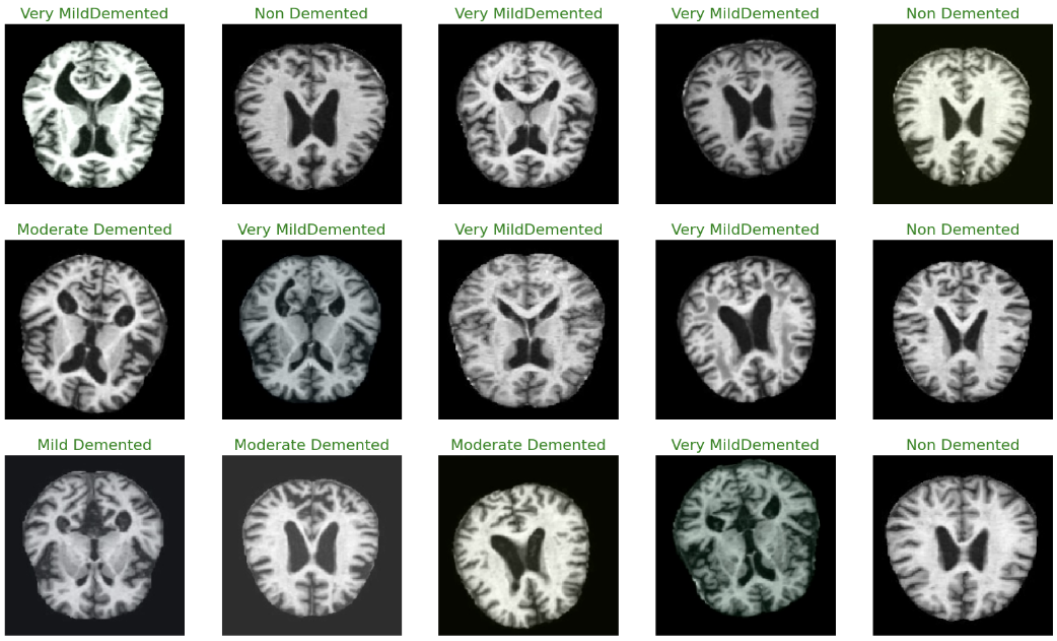

Figure 5: Sample images of classes in the dataset

This is a representation of the images sampled randomly from the dataset. It depicts the classes and the types of images in the classes present.

3.2 Data Splits

To effectively train and evaluate the model, the data is split into three subsets: training, validation, and test sets. This ensures that the model is trained on one portion of the data, tuned on a separate validation set, and finally tested on completely unseen images. The split is as follows:
to be added The training set is sampled with the intention to keep about 62% of the images, along with the validation and the testing set containing images from the remaining as 15% and 23% respectively.

A table containing brief description of the images is shown as follows in the form of a table :

Sets	**Number Of Images**
Training Set	27187
Validation Set	6797
Testing Set	10196

Table 3: Split of data into training, testing and validation

3.3 Data Preprocessing and Ingestion

Before feeding the dataset into the U-Net model, several preprocessing steps are required to ensure consistency and compatibility with the model's input format. These steps include reading the images, processing the annotations, and resizing the images to a specific size. The preprocessing is crucial because U-Net requires input images of a fixed size (812x812), and the annotations (masks) need to be prepared for segmentation tasks.

3.3.1 Reading and Processing Masks

The first step in preprocessing involves reading the annotated masks and preparing them for the segmentation task. The mask images are loaded using OpenCV, and contours are detected to extract the regions of interest. The contours are then drawn to create solid masks, which can be used as ground truth for training the U-Net model. This step ensures that the masks are properly prepared, with clearly defined regions for segmentation. The contours extracted from the masks provide the model with the necessary structure to segment the fetal brain from the ultrasound images.

3.3.2 Image Visualization

To verify that the mask and image preprocessing steps are working correctly, the images are visualized using matplotlib. This helps in confirming that the masks and images are properly aligned and processed before training. This visualization step helps to inspect both the original image and the corresponding mask after processing.

3.3.3 Resizing and Padding

Since U-Net requires input images of a fixed size (812x812), it is necessary to resize or pad the images to fit this specification. Images are padded to make them square, ensuring they match the required input dimensions without distortion. This resizing is essential for maintaining consistency in the training process.

The process is as follows :

- The difference between the width of the image and the required size (812 pixels) is calculated.
- To avoid stretching the image, padding (extra space) is added evenly on both sides of the image
- The extra padding is filled with black pixels (zeros), so no unnecessary information is introduced. This step is essential for preserving the integrity of the ultrasound images during model training
- The processed images are then used in batches of 16 for the training of our model

4 Model and Training specifications

The U-Net architecture was introduced in 2015 by Olaf Ronneberger, Philipp Fischer, and Thomas Brox in the paper titled "U-Net: Convolutional Networks for Biomedical Image Segmentation" [**?**]. U-Net was specifically designed to address the challenge of segmenting biomedical images, an essential task in medical imaging that involves identifying and labeling regions of interest, such as organs or tissues, in a given image. Before U-Net, traditional methods like manual annotation and early machine learning techniques struggled with the intricacies of biomedical image segmentation, largely due to the variability in image sizes and the need for precise localization of objects. While Convolutional Neural Networks (CNNs) had shown promise in tasks like object detection and classification, they weren't fully suited for the precise pixel-wise prediction required in medical image segmentation.

What makes U-Net unique is its combination of two critical components: (1) the contracting path (encoder), which captures the context of an image by extracting its features, and (2) the expanding path (decoder), which combines these features with high-resolution localization information to generate pixel-level predictions. This dual-path architecture allows U-Net to produce accurate segmentation masks that preserve both the global context of an image and fine-grained details. Another key innovation is the use of skip connections between corresponding layers in the encoder and decoder, which help preserve the original high-resolution information throughout the network, overcoming the loss of spatial resolution inherent in downsampling operations.

One of the main reasons U-Net is particularly well-suited for medical imaging is that it performs well with relatively small datasets. Medical datasets are often limited in size due to the difficulty of data collection and patient privacy concerns. U-Net addresses this by leveraging data augmentation and the ability to learn from fewer examples, making it ideal for tasks like fetal brain development monitoring, where large datasets might not always be available.

4.1 U-Net Architecture

The U-Net architecture is symmetric and consists of two paths: the contracting path (encoder) and the expanding path (decoder), which mirror each other, forming a "U" shape.

4.1.1 Encoder

The left side of the U-Net is the contracting path, responsible for capturing the context of the image by downsampling and extracting high-level features. Each layer in the encoder consists of two 3x3 convolutional layers followed by a ReLU activation function. These layers are grouped into blocks, with each block doubling the number of feature maps compared to the previous one. This progression helps the model learn increasingly abstract representations as the spatial resolution decreases and the depth of the feature maps increases. After each convolutional block, a 2x2 max-pooling layer is applied, reducing the spatial dimensions of the image by half, which allows the model to focus on the most prominent features.

For example, starting with an input image of size 812x812 pixels, the first convolutional block reduces the image size by half, producing an output of size 406x406. As the model moves deeper into the contracting path, the feature maps become smaller but richer in content. By the time the image passes through the fifth convolutional block, it has been reduced to a size of 26x26 pixels, but with a significant increase in the number of feature channels (1024 channels at this stage). This process enables the model to learn various levels of abstraction while compressing spatial information, allowing it to capture the global structure of the image.

4.1.2 Decoder

The right side of the U-Net is the expanding path, which up-samples the compressed feature maps back to the original image resolution. This path mirrors the contracting side but replaces max-pooling with up-convolutions (also known as transposed convolutions or deconvolutions) to restore the spatial resolution. These up-convolutions gradually double the size of the feature maps, reversing the downsampling process. The up-convolutional layers are followed by convolutional layers similar to those in the contracting path.

One of the defining features of U-Net is the use of skip connections between corresponding layers in the contracting and expanding paths. These connections transfer feature maps from the encoder directly to the decoder, allowing the model to recover fine-grained details lost during downsampling. Specifically, the feature maps from the encoder are concatenated with the up-sampled feature maps in the decoder, providing the model with both contextual and detailed information. This mechanism ensures that the model retains the spatial precision necessary for accurate segmentation.

For instance, after the deepest layer (which has 1024 channels and a spatial size of 26x26), an up-convolution is applied to increase the spatial size to 52x52. Simultaneously, the corresponding feature maps from the contracting path, which also have a size of 52x52, are concatenated with these upsampled maps, giving the model access to both high-level and fine-grained features. This process continues until the feature maps are restored to the original input size (812x812 pixels), at which point the final 1x1 convolution is applied to output the segmentation map.

4.1.3 Skip Connections

Skip connections are a critical innovation in the U-Net design. They allow the model to bypass information directly from the encoder to the decoder, ensuring that fine-grained spatial information is not lost during downsampling. These connections enable U-Net to recover precise boundary details that are essential for tasks like brain segmentation, where small structures must be accurately identified. By concatenating the feature maps from the contracting path with those in the expanding path, the network can produce high-resolution predictions without sacrificing the contextual information learned during encoding.

4.1.4 Output Layes

The output layer of U-Net consists of a 1x1 convolutional layer with a sigmoid activation function. This layer reduces the depth of the final feature map to one channel (or the number of desired output classes, which is one in our case), providing the segmentation map that predicts the likelihood of each pixel belonging to the foreground (brain tissue) or the background. The sigmoid activation ensures that the output values are between 0 and 1, which can be interpreted as probabilities for binary classification tasks.

4.2 Trainning Specification

The training of U-Net involves minimizing the difference between the predicted segmentation map and the ground truth labels using a suitable loss function, typically the binary cross-entropy loss for binary segmentation tasks like ours. This loss function measures how well the model's predictions match the true segmentation maps. During training, we used the Adam optimizer, which is well-suited for this type of deep learning task as it adjusts the learning rate dynamically based on the computed gradients. The learning rate was set to 0.001 to balance the speed of learning with the risk of overshooting the optimal solution.

The model was trained for 10 epochs, with each epoch consisting of multiple forward and backward passes over the training data. During each forward pass, the input images were passed through the U-Net, and the predicted segmentation maps were compared to the true labels to calculate the loss. In the backward pass, the gradients were computed, and the model parameters were updated to minimize the loss. After each epoch, the model's performance was evaluated on a validation set to monitor generalization and prevent overfitting.

The layers of batch normalization, convloutions and max pooling, act as intermediates between the modules as connections between them. After the connections, a global average pooling layer followed by a dropout of 0.4, a linear layer and softmax.

The amount of parameters and the overall description of the number of parameters in the layers is illustrated as follows :

An additional change in the network is the removal of auxiliary classifiers. In the original paper the auxiliary classifiers are used as a means to propagate gradients back to the layers. Given the large depth of the network, it was important that the gradients of the later layers also be propagated back again since they may be able to produce features in the middle of the network which may be

Type	Patch size/stride	#1 x 1	# 3 x 3 Reduce	# 3 x 3	# 5 x 5 Reduce	# 5 x 5	Pool proj
convolution	7 x 7 / 2						
max pool	3 x 3 / 2						
convolution	1 x 1						
convolution	3 x 3 / 1		128	512			
batchnorm							
max pool	3 x 3 / 2						
Inception 1(a)		64	96	128	16	32	32
Inception 1(b)		256	256	192	32	96	64
max pool	3 x 3 / 2						
Inception 2(a)		192	96	208	16	48	64
Inception 2(b)		160	112	224	24	64	64
Inception 2(c)		256	256	512	24	64	64
Inception 2(d)		112	144	288	32	64	64
Inception 2(e)		512	160	320	32	128	128
max pool	3 x 3 / 2						
Inception 3(a)		256	160	320	32	128	128
Inception 3(b)		384	192	384	48	128	128
global avg pool							
dropout (0.4)							
linear							
softmax							
Total Parameters					8,348,020		
Trainable Parameters					8,331,028		

Figure 6: Overview of Layers and Parameters

discriminative and so the auxiliary classifiers were used as a means to calculate the loss along with taking in consideration the losses produced by the auxiliary classifiers as well, taking them weighted by 0.3 and at the time of inference the auxiliary classifiers were discarded.

In our case, we eliminated the requirement of auxiliary classifiers given the fact that batch normalization takes care of all the considerations related to the gradients not being propagated and also helps in the classifier being able to converge in mere 40 epochs with a testing accuracy of 99.1% as opposed to the network without batch normalization which ceases to converge and stops early after 16 epochs with just about $\sim$28%.

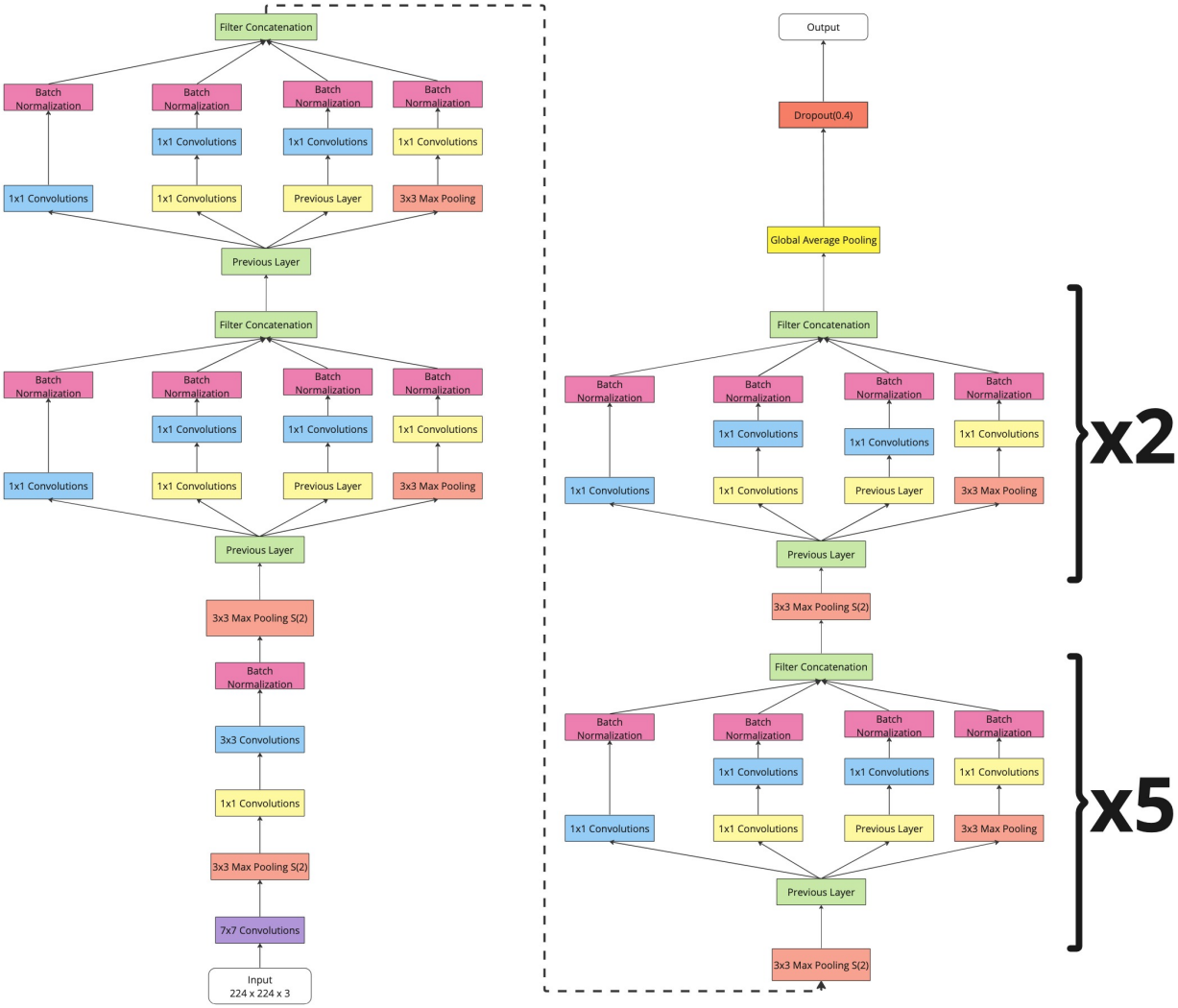

Figure 7: ExpGoogLeNet Network

4.3 Other Comparative Models and Results

Along with the described final model ExpGoogLeNet, we tried two other models with similar architectures, one was our implementation of GoogLeNet and the other was GoogLeNet along with batch normalization which we named BN GoogLeNet architecture. We trained all three of the models with the same training specifications and the results of the three are as follows :

Table 4: Literature Survey on Fetal Brain Health Management using Deep Learning

Author(s) & Year	**Method**	**Dataset**	**Key Findings**
Payette et al. (2021) [**?**]	3D U-Net, 2D U-Net	40 SR Scans	Achieved ∼80% accuracy in detecting brain anomalies
J de Asis et al. (2022) [**?**]	DCGAN & 3D U-Net	71 SR Scans	Predicted developmental outcomes with 97.3% precision
Qu et al. (2020) [**?**]	Deep Convolutional Neural Networks	30,000(Train)+1,200(Test) images	90.1% accuracy with Transfer Learning
D. Selvathi et al. (2022) [**?**]	Deep Convolutional Neural Networks	Dataset used can only be seen on special request	90.43% accuracy with Transfer Learning
Attallah et al. (2020) [**?**]	AlexNet Deep Convolutional Neural Networks	227 images(between 16-39 weeks)	AlexNet gave accuracy of 77.9%
Shinde K. & Thakare A. (2021) [**?**]	Deep Neural Networks + Machine Learning Algorihms	Dataset used can only be seen on special request	94% accuracy observed with a combined architecture
Zhao et al. (2022) [**?**]	3D U-Net	64 SR Scans	Attained 80.6% accuracy on 2D slices
Salehi et al. (2018) [**?**]	2D U-net & Auto-context	250(Train)+17(Test) images	96.52%, Overfitting suspected due to small dataset size
Xie et al. (2020) [**?**]	Deep Learning	10251(Normal images) 2529(Abnormal iamges)	95.9% accuracy observed after 200 epochs of training

5 Conclusion

Through this study, we highlight the application of U-Net architecture for fetal brain segmentation using ultrasound images, showing the potential for deep learning in prenatal diagnostics. While our model demonstrated promising results in segmenting brain regions, there remains significant room for further research and refinement.

Future work could explore the integration of more complex data augmentation techniques, such as synthetic data generation using GANs, to improve model robustness across diverse patient populations and ultrasound variations. Additionally, expanding the dataset to include a wider range of gestational ages and conditions could enhance the generalizability of the model, making it applicable in different clinical settings. Transfer learning approaches might also be beneficial, especially in cases where annotated data is limited.

In terms of use cases, this model could be further developed for early detection of neurological abnormalities, helping clinicians monitor brain development more effectively. Automated fetal brain segmentation could aid in the diagnosis of conditions such as microcephaly or hydrocephalus, reducing the time and effort required for manual analysis.

Another area worth investigating is multi-class segmentation, which would allow the model to differentiate between various brain structures and provide a more detailed assessment of fetal development. Moreover, integrating this technology into real-time clinical workflows or mobile health platforms could open new doors for remote diagnostics and prenatal care, especially in low-resource settings.

In summary, while this work presents a solid foundation, there is considerable scope for improvement and expansion, particularly in enhancing model robustness, scalability, and clinical applicability. By addressing these aspects, future research can make significant strides toward more accurate and accessible fetal brain monitoring solutions.

References

[1] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 07-12-June-2015. https://doi.org/10.1109/CVPR.2015.7298594

[2] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. 32nd International Conference on Machine Learning, ICML 2015, 1.

[3] Batch Normalization `http://d2l.ai/chapter_convolutional-modern/batch-norm.html`

[4] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 2.

[5] Alzheimer's Dataset (4 class of Images) `https://www.kaggle.com/datasets/tourist55/alzheimers-dataset-4-class-of-images`

[6] Augmented Alzheimer MRI Dataset `https://www.kaggle.com/datasets/uraninjo/augmented-alzheimer-mri-dataset`

[7] Darma, A. S., & Mohamad, F. S. B. (2021). The Regularization Effect of Pre-activation Batch Normalization on Convolutional Neural Network Performance for Face Recognition System Paper. International Journal of Advanced Computer Science and Applications, 12(11). https://doi.org/10.14569/IJACSA.2021.0121135

[8] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11). https://doi.org/10.1109/5.726791

[9] Thakkar, V., Tewary, S., & Chakraborty, C. (2018). Batch Normalization in Convolutional Neural Networks - A comparative study with CIFAR-10 data. Proceedings of 5th International Conference on Emerging Applications of Information Technology, EAIT 2018. https://doi.org/10.1109/EAIT.2018.8470438

[10] Wang, S. H., Tang, C., Sun, J., Yang, J., Huang, C., Phillips, P., & Zhang, Y. D. (2018). Multiple sclerosis identification by 14-layer convolutional neural network with batch normalization, dropout, and stochastic pooling. Frontiers in Neuroscience, 12(NOV). https://doi.org/10.3389/fnins.2018.00818

[11] Antunes, A., Ferreira, B., Marques, N., & Carriço, N. (2023). Hyperparameter Optimization of a Convolutional Neural Network Model for Pipe Burst Location in Water Distribution Networks. Journal of Imaging, 9(3). https://doi.org/10.3390/jimaging9030068

[12] Han, X., Hu, Z., Wang, S., & Zhang, Y. (2023). A Survey on Deep Learning in COVID-19 Diagnosis. In Journal of Imaging (Vol. 9, Issue 1). https://doi.org/10.3390/jimaging9010001

[13] Jasim, R. M., & Atia, T. S. (2023). An evolutionary-convolutional neural network for fake image detection. Indonesian Journal of Electrical Engineering and Computer Science, 29(3). https://doi.org/10.11591/ijeecs.v29.i3.pp1657-1667

[14] Li, H., Tan, Y., Miao, J., Liang, P., Gong, J., He, H., Jiao, Y., Zhang, F., Xing, Y., & Wu, D. (2023). Attention-based and micro designed EfficientNetB2 for diagnosis of Alzheimer's disease. Biomedical Signal Processing and Control, 82. https://doi.org/10.1016/j.bspc.2023.104571

[15] Lin, H. I., Mandal, R., & Wibowo, F. S. (2024). BN-LSTM-based energy consumption modeling approach for an industrial robot manipulator. Robotics and Computer-Integrated Manufacturing, 85. https://doi.org/10.1016/j.rcim.2023.102629

[16] Saeedi, S., Rezayi, S., Keshavarz, H., & R. Niakan Kalhori, S. (2023). MRI-based brain tumor detection using convolutional deep learning methods and chosen machine learning techniques. BMC Medical Informatics and Decision Making, 23(1). https://doi.org/10.1186/s12911-023-02114-6

[17] Priyatama, A., Sari, Z., & Azhar, Y. (2023). Deep Learning Implementation using Convolutional Neural Network for Alzheimer's Classification. Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi), 7(2). https://doi.org/10.29207/resti.v7i2.4707

[18] Zhao, Z., Chuah, J. H., Lai, K. W., Chow, C. O., Gochoo, M., Dhanalakshmi, S., Wang, N., Bao, W., & Wu, X. (2023). Conventional machine learning and deep learning in Alzheimer's disease diagnosis using neuroimaging: A review. In Frontiers in Computational Neuroscience (Vol. 17). https://doi.org/10.3389/fncom.2023.1038636