Deep Learning-Based Classification of Fetal Ultrasound Planes

Kareem Khaleel
Department of Computer Science
University College Dublin
kareem.khaleel@ucdconnect.ie

Abstract

Ultrasound imaging is one of the most important tools in the medical field. Automating the classification process for fetal ultrasound imaging improves diagnostic accuracy and consistency. This paper explores different deep learning approaches to classifying fetal ultrasound images. It compares baseline convolutional neural networks (CNNs), an improved CNN model, and several state-of-the-art architectures. The experiments demonstrate the impact of different transformers, architectures, and hyperparameter settings. The enhanced model achieved an accuracy of 90%, while state-of-the-art models such as ResNet18 and DenseNet121 reached up to 91.79% and 91.46%, respectively, on the same dataset. Further insights into the methodology and results are discussed.

1 Introduction

Ultrasound imaging is one of the most used techniques when it comes to prenatal care to be able to assess the fetal and check its development and detect any anomalies. It's important to standardize the fetal ultrasound plane identification to have more accurate and quicker clinical assessment. Manual classification for ultrasound images, especially fetal images, is subject to observer variability and inefficiency. Human errors can occur in the classification and detection of these images. Recent advancements in deep learning, particularly CNNs, open the door for promising solutions to automate this process.

In this research, we investigate deep learning models to classify fetal ultrasound images accurately. Based on the recent work done in this area, a literature review was conducted to understand the problem clearly, check the existing approaches, what the challenges are, and to have an idea of best practices that can be done. After analyzing the results, a baseline model was created along with different transformers to improve the data quality. In our testing, we compare our models with different settings. Finally, we compare the results of our work with several state-of-the-art networks, including ResNet18, ResNet50, DenseNet121, EfficientNet-B0, and MobileNetV2.

This paper is organized as follows: Section 2 will show related research on ultrasound image classification. Section 3 explains the experimental setup, including datasets, transformations, and models. Section 4 shows the results of our experiments, and Section 5 provides the conclusions of the work, with future research lines.

2 Related Work

Recent advances in ultrasound image classification have primarily concentrated on Convolutional Neural Networks (CNNs) due to their ability to learn spatial hierarchies from medical image data.

Preprint. Under review.

Baumgartner et al. [1] proposed the SonoNet, a CNN-based system that automatically detects and localizes standard fetal ultrasound planes from freehand scans in real-time. Their method provided an impressive productivity boost to clinical workflow that exceeded traditional methods both in terms of speed and accuracy, and featured a handy real-time localization capability.

Burgos-Artizzu et al. [2] conducted one of the earliest and most comprehensive evaluations of CNNs for the classification of fetal ultrasound planes. Using a VGG-based architecture, they achieved superior classification performance on six benchmark maternal-fetal ultrasound planes. Their work also released the FETAL_PLANES_DB dataset used in our research, providing a useful benchmark for future work in this arena.

To address the issue of limited medical data, Chen et al. [3] employed a transfer learning approach by fine-tuning pre-trained ResNet models on large-scale natural image datasets. Their results showed improved generalization and robustness, confirming that pre-trained CNNs can be transferred to fetal ultrasound images with minimal retraining.

Liu et al. [4] extended this work by examining DenseNet architectures for fetal ultrasound standard plane classification. They identified the advantage of dense connectivity to preserve gradient flow and enable feature reuse to provide improved accuracy and computational efficiency.

Tan and Le [5] introduced the EfficientNet family, which regulates network depth, width, and resolution systematically through neural architecture search. While not specifically an ultrasound, EfficientNet has been applied with great success in medical imaging applications and is well-suited to ultrasound image classification due to its parameter efficiency and scalability.

Aside from model structure, data augmentation and preprocessing are of utmost concern in model robustness improvement. Shorten and Khoshgoftaar [6] presented a comprehensive list of augmentation methods, including flipping, rotation, cropping, brightness adjustment, and elastic deformations. They have been effective, particularly against overfitting and improving model generalization in fields of application where there is limited labeled data available, for example, prenatal imaging.

In terms of model optimization, Bergstra and Bengio [7] advocated random search as a more efficient alternative to traditional grid search for hyperparameter tuning. Their paper laid the ground for automated tuning strategies, which are now seen as essential for fine-tuning CNN performance in many areas, including medical image classification.

Our research builds on these foundations by incorporating state-of-the-art CNN architectures systematically, embracing data augmentation techniques, and performing controlled hyperparameter tuning to improve classification accuracy for fetal ultrasound images.

3 Experimental Setup

3.1 Dataset

We made use of the publicly accessible FETAL_PLANES_DB dataset, which consists of 5,271 grayscale ultrasound images from actual clinical pregnancy scans. There are six anatomical classes - fetal_abdomen, fetal_brain, fetal_femur, fetal_thorax, maternal_cervix, and other - to which images are labeled and annotated. Classes represent standard fetal views typically reviewed in obstetric ultrasound scanning. Both normal planes and non-diagnostic views make up the dataset, giving a realistic and varied distribution of fetal image data.

In addition, the dataset includes metadata such as patient IDs, exam operator, and ultrasound machine manufacturer, facilitating additional investigation of model robustness to different acquisition conditions. To allow for fair training and testing, we split the dataset into training (60%), validation (20%), and testing (20%) sets with a balanced class distribution in each set. This allows for reproducible performance benchmarking as well as prevents class imbalance during model training.

3.2 Preprocessing Transformers

To further improve the generalization and strength of the model, we employed three different data transformation pipelines at training, and they were labeled as Transformer 1, Transformer 2, and Transformer 3. These kinds of transformations are especially important when one is working with relatively small-sized datasets, such as FETAL_PLANES_DB, containing a relatively small number of labeled fetal ultrasound images. All transformation pipelines include varying degrees of augmentation to mimic the real-world variation found in ultrasound image acquisition, thereby enhancing the variety in the training dataset and reducing the risk of overfitting.

Transformer 1:

Does some basic prepossessing by resizing all input images to 128×128 pixels and randomly flipping them horizontally. The procedure provides a fixed input size with variation in orientation to simulate various fetal positions or probe orientations.

Pixel values are normalized by a mean and standard deviation of 0.5 to move the distribution of data towards the center, thereby facilitating improved convergence during training. This is an optimized augmentation form, ideal for rapid training, although it may be constrained in representing sophisticated anatomical features.

Transformer 2:

Increases the input resolution to 224×224 pixels, thereby allowing the model to capture and learn additional spatial information, e.g., fine organ contours and tissue texture. This enhancement is especially helpful in separating anatomically similar classes, e.g., fetal thorax and abdomen. As in Transformer 1, random horizontal flipping and normalization are also performed. Moreover, the increased resolution renders this configuration suitable for deeper network models that can take advantage of more detailed input.

Transformer 3:

Complements the capacities of Transformer 2 with further augmentation techniques intended to mimic real-world imaging variability. It adds random rotations of ±10 degrees to simulate probe orientation differences, and brightness and contrast perturbations to simulate differences in ultrasound machine settings and image quality. This aggressive augmentation approach effectively raises dataset variability and reinforces the model's ability to learn robust features that generalize across a broad array of acquisition conditions. Collectively, the incremental improvement strategies of these three transformers allow systematic exploration of how input variability affects model performance. Transformer 3 was, however, found to be optimal in boosting classification performance and generalizability, particularly in challenging and unclear cases.

3.3 Model Architectures

We explored two primary CNN architectures:

CNNBaseline:

As a baseline for performance, we developed a bespoke convolutional neural network named CNNBaseline. Our design takes a minimalist but powerful approach, comprising both a feature extraction module and a classification head.

The feature extraction module is composed of four convolutional layers, each of which is followed by a ReLU activation function, batch normalization, and max pooling.

The number of output channels grows systematically across the layers, i.e.: 32, 64, 128, and 256. The systematic architecture enables the capacity of the network to capture increasingly abstract and finer features in the deeper layers. The application of batch normalization after every convolution brings stability and a higher convergence rate by normalizing feature maps, while the max pooling layers progressively decrease the spatial resolution, thereby inducing translation invariance and accelerating computational complexity.

In particular, all convolutions have a kernel size of 3×3, with padding being 1, thereby preserving the spatial size before pooling. The max pooling layers down-sample the input size by a factor of 2 at each stage, leading to a final feature map size of 14×14 (given an initial input resolution of 224×224, by Transformer 2 and Transformer 3). This setup leads to the final tensor having a shape of (256, 14, 14) following the convolutional backbone. The classification block begins with the flattening layer that reshapes the 3D tensor to a 1D vector of size $256 \times 14 \times 14 = 50{,}176$. The vector is passed through a fully connected layer of 512 units, and ReLU activation, along with a dropout layer having a dropout rate of 0.5, is used to prevent overfitting.

The final layer includes a completely connected linear transform relating the output to six class logits, where the six class logits reflect the six classes in the dataset. This architecture was selected as a baseline model to evaluate the impact of applying different data augmentation techniques (i.e., Transformers 1–3) and as a benchmark to evaluate against deeper and more complex architectures such as ResNet18, DenseNet121, and EfficientNet-B0. The CNNBaseline model, despite its simplicity, performs well and serves to illustrate the role of each component, such as data augmentation, resolution scaling, and regularization techniques.

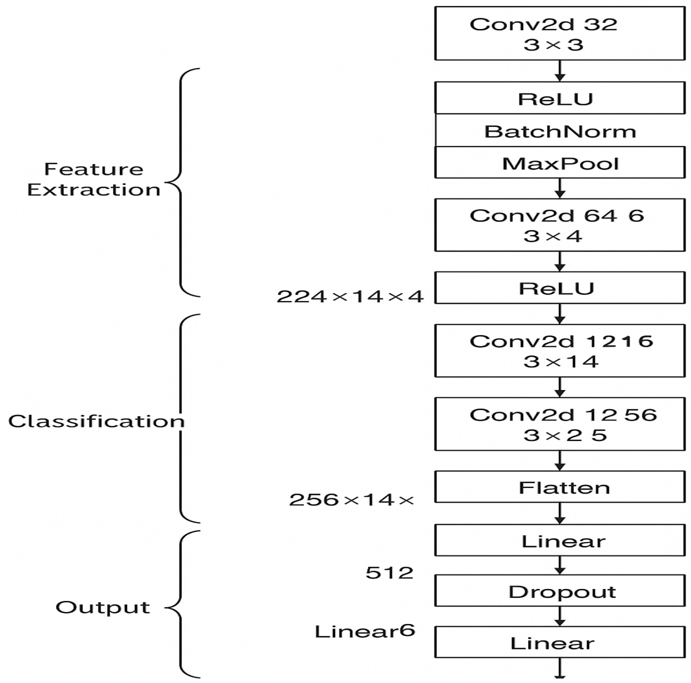

Figure 1: CNNBaseline architecture showing 4 convolutional blocks and fully connected layers for classification.

CNNImproved:

Starting from the baseline CNN, we developed an improved architecture, referred to as CNNImproved, intending to increase model capacity without sacrificing computational efficiency. This version introduces a deeper convolutional feature extractor and employs an adaptive pooling strategy to reduce the complexity of the classification head.

The feature extraction module contains five convolutional blocks, an increase from four in the baseline. Each block has the same pattern: a 2D convolutional layer with 3×3 kernel size and 1 padding to maintain spatial dimensions, followed by a ReLU activation function, batch normalization, and max pooling of size 2×2. The number of output channels systematically goes up through the layers as: 32, 64, 128, 256, and lastly 512 for the fifth block.

This additional fifth block facilitates the model's capacity to capture higher-level features, especially useful for fine-grained distinction between fetal anatomical structures.

After the feature extraction procedure, the output feature map is fed through an Adaptive Average Pooling layer, which downsamples each feature map to a single spatial location, producing an output dimension of (512, 1, 1). This procedure guarantees that the input to the classifier is of a fixed size, irrespective of the spatial dimensions of the preceding convolutional output. It also lowers the overall number of parameters compared to using extremely long flattened vectors, as in the case of the baseline model.

The classifier module then flattens the pooled output into a 1D vector of size 512 and then a fully connected layer with 256 neurons, followed by ReLU activation and a dropout layer with a rate of 0.5 to avoid overfitting. The final layer is a fully connected layer mapping the intermediate representation to six output classes corresponding to the six fetal ultrasound views in the dataset.

By reducing the number of parameters in the fully connected layers, due to the use of AdaptiveAvg-Pool2d, this architecture achieves improved training stability, generalization, and regularization. Furthermore, the deeper model allows it to learn more abstract hierarchical features compared to the baseline CNN, making it more suitable for challenging classification tasks.

Table 1: Comparison between the Baseline CNN and the Improved CNN Architecture

Component	**Baseline CNN**	**CNNImproved**
Convolutional Blocks	4	5
Max Pooling Strategy	Fixed per layer	Fixed per layer
Feature Channels (Final)	256	512
Pooling Before FC	None	Adaptive Average Pooling
FC Layer Input	$256 \times 14 \times 14$ (flattened)	512 (pooled + flattened)
FC Layers	$512 \rightarrow$ num_classes	$512 \rightarrow 256 \rightarrow$ num_classes
Regularization	Dropout (0.5)	Dropout (0.5)

3.4 hyperparameter Tuning

To enhance model performance and encourage good generalization, we conducted systematic hyper-parameter tuning following a grid search strategy. The goal was to identify optimal sets of important training parameters with direct influence on model convergence, regularization, and stability. The grid search was conducted over a number of configurations of learning rate, dropout rate, and batch size, which have been shown to significantly influence training dynamics.

We checked the following ranges:

Learning Rates: [1e-3, 1e-4]: Learning rate determines how much weight correction is allowed during backpropagation. We tried a smaller learning rate (1e-4) to achieve stable and slow convergence and a higher rate (1e-3) to try to accelerate training, but had to closely monitor to prevent divergence or overfitting.

Attrition Rates: [0.3, 0.5]: Dropout is a regularization technique that randomly disables a proportion of neurons during training to decrease overfitting. We experimented with two typical values: 0.3 for light regularization and 0.5 for more aggressive dropout, particularly useful for deeper models.

Batch Sizes: [16, 32]: Batch size influences the noise in gradient estimation and memory usage. A batch size of 16 yields more stable updates at the cost of training time, while a batch size of 32 offers computational efficiency at a slight sacrifice in granularity of the gradient updates. For every configuration, the model was trained for a set number of epochs, and performance on the validation set was evaluated using classification accuracy and confusion matrices as main metrics. Training loss and validation accuracy were also tracked at every epoch to examine convergence behavior and generalization gaps.

Furthermore, we observed trends indicating overfitting by comparing training and validation performance under varying dropout rates. This systematic tuning allowed us to see more clearly the interplay between architectural complexity (i.e., deeper CNN blocks), data augmentation strategies, and training settings. Surprisingly, the learning rate 1e-4, dropout rate 0.5, and batch size 16 combination consistently resulted in high validation accuracy while maintaining models' stability, especially when paired with more powerful augmentations like those in Transformer 3.

4 Results

In this section, we will present our work and explain our findings for each part of the experiment.

4.1 Baseline model

We used this model to train and run on different transformers as an initial step to the classification process. The following table shows the results of the training.

Table 2: Performance of CNNBaseline with different transformers

Transformer	Epoch 5 Acc	Epoch 10 Acc	Epoch 15 Acc	Final Test Acc
Transformer 1	87.65%	88.12%	**88.47%**	88.47%
Transformer 2	86.04%	88.00%	85.35%	85.35%
Transformer 3	84.37%	87.97%	88.41%	88.41%

Based on the table, it's clear that the model is providing good accuracy. However, the training among the epochs is not changing in a good way. Some additional tests were done, and we decided to add one more convolution layer to make the model more accurate

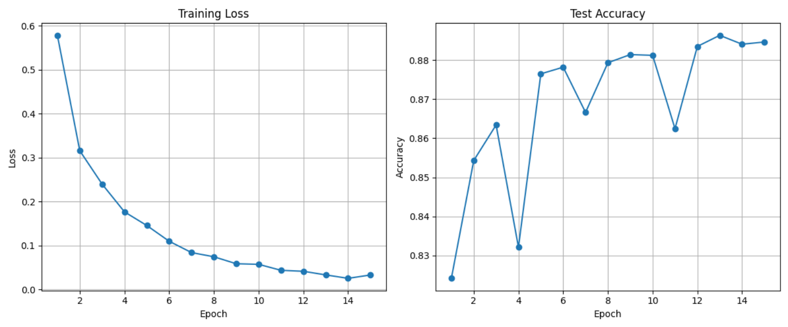

Figure 2: Training Loss and Test accuracy Curve for Baseline model over 15 epochs.

The oscillations in the test accuracy show that the model training is not stable, and more tuning and enhancement can be done to this model.

4.2 Improved model

Based on the results of the previous model, we decided to enhance the architecture and do more testing on the same data, using the same transformers.

One additional thing that was done was to increase the number of epochs to check if the results would be enhanced if we trained the model for more time or not.

The following table shows the hyperparameters that the model was trained on. It shows that the results vary from one set to another. All of these results are for the model using Transformer 1 as the data processor.

Table 3: Hyperparameter tuning results for CNNImproved

Configuration	Accuracy	Macro F1	Weighted F1
lr=1e-3, drop=0.3, bs=16	84.33%	0.811	0.840
lr=1e-3, drop=0.3, bs=32	84.82%	0.818	0.847
lr=1e-3, drop=0.5, bs=16	85.37%	0.821	0.852
lr=1e-4, drop=0.3, bs=16	87.67%	0.860	0.877
lr=1e-4, drop=0.3, bs=32	**87.74%**	**0.860**	**0.876**

Using Transformer 3, the results were enhanced, and we were able to achieve better accuracy. The following table shows the results of training over 50 epochs:

The data shows that increasing the epochs alone will not enhance the results, where after 20 epochs the results start to oscillate around 90%.

The confusion matrix shows that the model was able to get good results, but it didn't perform well for the classification "Other" class.

Table 4: Model Performance Across Epochs

Epoch	**Loss**	**Accuracy**
5	0.2805	0.8625
10	0.1682	0.8945
15	0.1200	0.8788
20	0.0803	0.9032
25	0.0680	0.9065
30	0.0564	0.8877
35	0.0485	0.9110
40	0.0446	0.8985
45	0.0498	0.9251
50	0.0467	0.9108

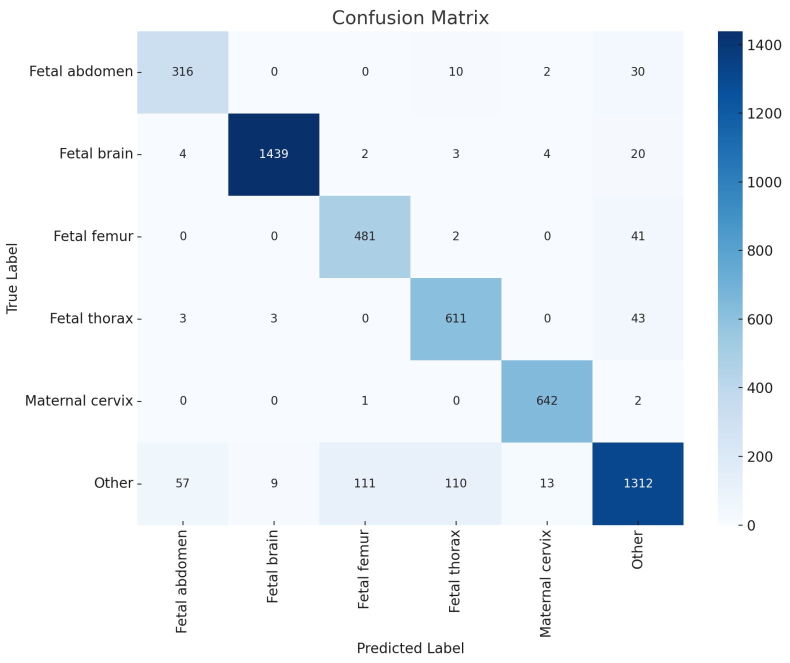

Figure 3: The confusion matrix of the model.

4.3 State-Of-The-Art models

As we are not the only ones who are researching this area, we have to compare our work to others. In this section, we have run the same data through the State-of-the-art to get a reference to compare our work to.

The following table shows some results for networks that we used as a reference.

Table 5: Comparison with pre-trained models

Model	Accuracy	Loss
ResNet18	**91.79%**	0.0402
ResNet50	89.68%	0.0606
DenseNet121	91.46%	**0.0383**
EfficientNet-B0	90.51%	0.0372
MobileNetV2	90.89%	0.0510
Baseline Model	88.40%	0.0881
Improved Model	91.08%	0.0467

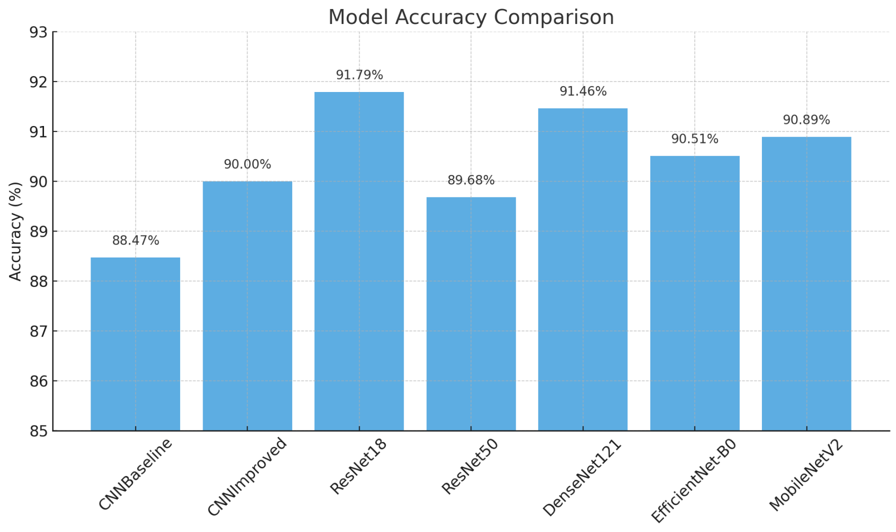

Figure 4: Comparison of accuracy across CNNBaseline, CNNImproved, and several state-of-the-art models.

5 Conclusion and Future Work

This study presents a comprehensive exploration of deep learning approaches for fetal ultrasound plane classification. Starting from a simple CNNBaseline, we progressively improved model architecture, preprocessing strategies, and hyperparameter tuning to significantly enhance classification accuracy. The enhanced CNNImproved model achieved a strong performance of 90% accuracy, while state-of-the-art models like ResNet18 and DenseNet121 exceeded 91%

This study presents an initial testing of CNNs with ultrasound image classifications for fetal image classification. We started with a simple CNN baseline model and used different transformers and settings to fine-tune the networks. We were able to get good results with an accuracy of 91% for the enhanced model. However, the data shows that there are still some problems that were not covered by this research. For example:

- The models still have room for enhancement, where the accuracy is not optimal yet.
- It was clear from the confusion matrix for us and for the other related papers that the models face issues detecting the “Other” class.
- Explanation of the classification results

This research shows that it’s possible to automatically classify the ultrasound images. This will enhance the medical field and reduce human errors. Before jumping to conclusions, the models must be enhanced even more to be more accurate and realistic since we are working with a medical AI system

For future work, we suggest investigating:

- The application of attention-based mechanisms (e.g., Vision Transformers) to capture long-range dependencies.
- Ensemble learning to combine multiple models for more robust predictions.
- Use different datasets to make sure the results are correct

References

[1] Baumgartner, C. F., et al. (2017). Sononet: real-time detection and localisation of fetal standard scan planes in freehand ultrasound. IEEE Transactions on Medical Imaging.

[2] Burgos-Artizzu, X. P., et al. (2020). Evaluation of deep convolutional neural networks for automatic classification of common maternal-fetal ultrasound planes. Scientific Reports.

[3] Chen, H., et al. (2019). Standard plane detection in fetal ultrasound images by deep convolutional neural network. IEEE Access.

[4] Liu, X., et al. (2021). Automatic classification of ultrasound fetal standard planes using DenseNet. Medical Image Analysis.

[5] Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. Proceedings of the 36th International Conference on Machine Learning.

[6] Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of Big Data.

[7] Bergstra, J., & Bengio, Y. (2012). Random search for hyperparameter optimization. Journal of Machine Learning Research.