OBUS – Model Architectures

Overview

The overarching goal of the AI models for the four obstetric features of interest—gestational age, fetal presentation, fetal weight, and multiple gestation—is to extract clinically actionable information from ultrasound videos. We have employed the usual deep learning paradigm to solve this problem: we created a many-layered model architecture and tuned the model weights through an iterative training process that involves presenting the model with many examples of video/label pairs repeatedly until the loss function reaches a plateau or a minimum. The model weights are modified by backpropagating the model's output error via a gradient computation.

High-level architecture

The input being video, the data is inherently 3-dimensional. Therefore, the model architecture must extract both the spatial and the temporal information residing in the video. Various deep learning architectures for video understanding have been proposed in the research literature. Three broad categories of model architectures may be defined and within each category a number of variants exist depending on the specific choices for each of the modules. This categorization scheme is illustrated in Table 1, where the entries in the Architecture column are understood to be followed by a fully connected layer, which performs a projection to a lower dimension solution space. This module will act either as a regressor or a classifier, depending on the task. A representative (not exhaustive) list of examples of architectures in the Sequential category are shown, along with comments about their applicability and deployability.

In the sequential architectures, spatial feature extraction occurs first (also called spatial encoding), converting every frame to a feature vector (also known as an embedding). Next, the sequence of frame embeddings is temporally aggregated into a video-level feature vector (also known as a context vector). The video-level context vector is used to make the final prediction via a fully connected layer with an appropriate activation function. One variant of the temporal aggregator is a convolutional LSTM, which replaces multiplication with convolution and thus requires a feature map as input. To achieve this, the CNN output must be derived from a spatial feature map layer, usually chosen to be the last layer before spatial pooling to create a feature vector.

In the unitary group, spatial and temporal analysis are done together in a single module. Variations other than the four listed are conceivable. These architectures tend to be either excessively data hungry (*e.g.*, 3D CNN) or computationally expensive at inference time, or both. The Mamba architecture might be an exception, but time did not allow a thorough exploration of this architecture.

Category	**Architecture**	**Deployability**	**Comment**
Sequential frame encoder→ temporal aggregator	CNN → Attention	good	good for GA, EFW
	CNN → Self-Attention	good	good for FP, TWIN
	CNN → RNN	good	good for FP, TWIN
	2D Vision Transformer → RNN	img size dep.	potential scaling issues
Unitary spatial, temporal in one module	(2 + 1) D CNN	good	needs exploration
	3D CNN	expensive	needs excessive data to train
	3D Vision Transformer	img size dep.	potential scaling issues
	3D Vision Mamba	img size dep.	scaling issues resolved
Parallel spatial, temporal separately but simultaneously	Spatial Temporal }	good	won't work well for ultrasound sweeps because temporal changes derive from transducer movement

Table 1. Deep learning architectures for video understanding

Finally, in the parallel group, spatial and temporal analysis are conducted separately and simultaneously and the results subsequently combined. We believe this arrangement is less suitable for blind sweeps because the temporal pattern is a result of the movement of the transducer, resulting in effectively a 3D image, somewhat like a CT scan or MRI. Again, time did not allow an exploration of this architecture group.

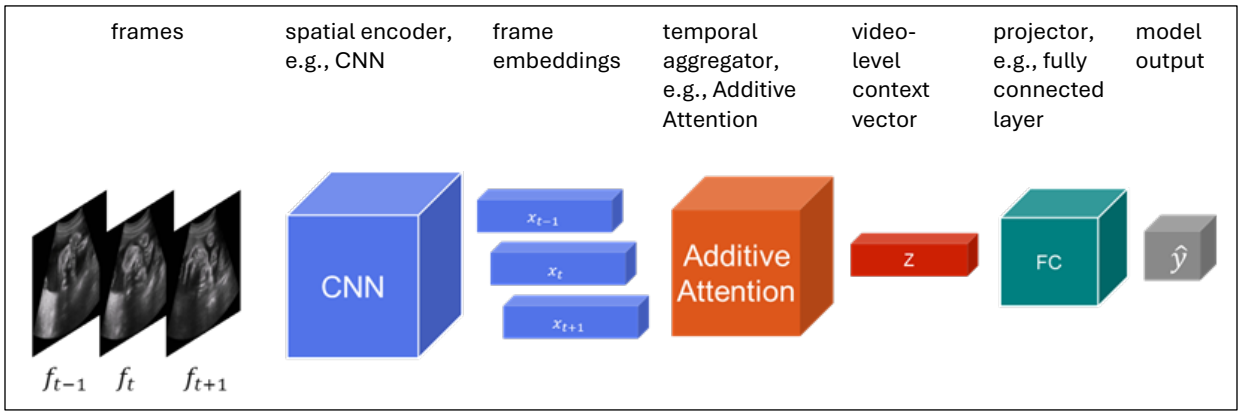

Figure 1. Block diagram of Sequential architecture

The model architecture for all four obstetric features are chosen from the first category, the Sequential group. A block diagram of the Sequential architectures is shown in Figure 1.

Different choices for each of the three modules—spatial encoder, temporal aggregator, and projector—are based on the specific obstetric feature, and these are discussed in the feature-specific model architecture documents.

Exam processing

All four obstetric feature models must output an exam-level result. Each exam consists of multiple videos (usually blind sweeps, but also other types of videos). Regardless of whether the training is performed at the video level or the exam level, the output result must be at the exam level, and the ground truth labels are therefore only available at the exam level, not the video level. For three of the obstetric features, namely gestational age, fetal presentation, and fetal weight, training can be done at the video level because every video carries some information pertinent to the target feature. For these three features, inference is first carried out first at the video level, then the video level results are aggregated to output an exam-level result.

For the multiple gestation feature, every video does not necessarily contain evidence of the target (*e.g.*, twins) and therefore video-level training will try to force the network to output TWIN for a video that may not contain evidence of twins, resulting in classifier confusion. For the multiple gestation model, we use exam-level training via multiple instance learning, which is described more fully in [4.3 Multiple Gestation Model Architecture]. During inference, the entire exam is presented to the multiple gestation model in a single batch, so the two-level approach of video-then-exam aggregation is not necessary.

Frame sampling

Regardless of whether the training is performed at the video level or the exam level, training can only be performed on a fixed number of frames. This requirement is imposed by memory constraints and by the need to have fixed-shape batches for training. On the other hand, video lengths are quite variable, so during video-level training, a fixed number of frames are sampled randomly from each video. During inference, each video is a single batch, so a variable number of frames can be accommodated, even all the frames, if so desired. This applies to the GA and EFW models: a fixed number of frames during training; all the frames during inference.

For the multiple gestation feature, frame sampling occurs at the exam level. Special precautions are needed to keep the average interval between sampled frames constant.

This sampling scheme is more fully discussed in [4.3 Multiple Gestation Model Architecture], as well as the README for the code repository [1].

References

[1] "GHL OBUS GitHub Repository," 2025. [Online]. Available: https://github.com/Global-Health-Labs/OBUS-GHL-DEV.