**2.4.4.** **Temporal CNN with Feature Engineering**

To incorporate temporal dependencies in presence detection, a Temporal Convolutional Neural Network (Temporal CNN) was developed. Unlike LSTM-based architectures which are not supported in TensorFlow Lite (TFLite) for Raspberry Pi deployment; this model employs 1D convolutional layers along the time axis. This approach enables the network to efficiently model sequential behavior in thermal data while maintaining compatibility with real-time deployment on edge devices.

Table 5: Temporal CNN Model Architecture

|  |  |  |
| --- | --- | --- |
| **Layer** | **Type** | **Output Shape** |
| Conv1 | 2D Convolution | (Batch, 16, 32, 32) |
| MaxPool1 | Max Pooling (2) | (Batch, 16, 16, 16) |
| Conv2 | 2D Convolution | (Batch, 32, 16, 16) |
| MaxPool2 | Max Pooling (2) | (Batch, 32, 8, 8) |
| Flatten | Fully Connected | (Batch, 2048) |
| Temporal Conv1D | 1D Convolution | (Batch, 32) |
| Feature Input | Handcrafted Data | (Batch, X) |
| Feature Conv1D | 1D Convolution | (Batch, 16) |
| Concatenation | Merged Features | (Batch, 32+16) |
| Dense1 | Fully Connected | (Batch, 128) |
| Dropout | Regularization | (Batch, 128) |
| Output | Sigmoid | (Batch, 1) |

The architecture consists of two parallel branches: a deep learning pipeline that processes raw thermal frames and a handcrafted feature branch that extracts statistical image features using classical image processing techniques as shown below in figure 1. The thermal frame branch operates on a sequence of five thermal images, each of size 32 × 32. A TimeDistributed CNN applies convolutional operations to each frame independently, capturing spatial patterns such as heat blobs, edges, and gradients. This includes two convolutional layers with ReLU activation (16 and 32 filters respectively), each followed by max pooling to reduce dimensionality and retain key spatial characteristics. After processing, each frame is flattened into a 2048-dimensional vector, resulting in a sequence of shape (5, 2048). This is passed through a 1D convolutional layer that learns short-term temporal dependencies, followed by GlobalAveragePooling1D, which reduces the temporal sequence into a single 32-dimensional vector. This pooling layer helps summarize the dominant temporal patterns while reducing sensitivity to slight timing variations between frames and minimizing parameter count.

In parallel, the handcrafted feature branch processes a set of seven features derived from each thermal frame using custom Python functions. These features include the mean and variance of gradients in the X and Y directions (to capture directional heat change), the variance of a small region surrounding the hottest pixel (representing concentrated human heat signatures), and edge intensity and variance computed via the Sobel operator. This results in a 5 × 7 input sequence, which is then passed through a 1D convolutional layer to model temporal evolution of these features. A GlobalAveragePooling1D layer further compresses the sequence into a fixed 16-dimensional representation.

The outputs from both branches—a 32-dimensional vector from the thermal CNN and a 16-dimensional vector from the handcrafted feature pipeline—are concatenated to form a unified representation of both deep-learned and engineered cues. This 48-dimensional feature vector is passed through a fully connected dense layer with 128 neurons and ReLU activation, followed by dropout regularization with a rate of 0.3 to mitigate overfitting. The final dense layer with sigmoid activation outputs the probability of human presence for the given 5-frame sequence.

The complete model architecture, including layer types and output shapes, is detailed in **Table 5**. A simplified schematic overview of the full inference pipeline is illustrated in **Figure 1**, highlighting the dual-branch structure, layer transitions, and feature fusion strategy. This final Temporal CNN model was chosen for deployment due to its balance between efficiency, accuracy, and compatibility with edge inference. The use of Conv1D layers in place of LSTMs allowed for faster inference times and reduced memory usage, making it highly suitable for low-power platforms like the Raspberry Pi. Additionally, the integration of both learned and handcrafted features improved the model’s robustness to variations in motion, posture, and ambient conditions, resulting in improved presence detection performance in dynamic environments.

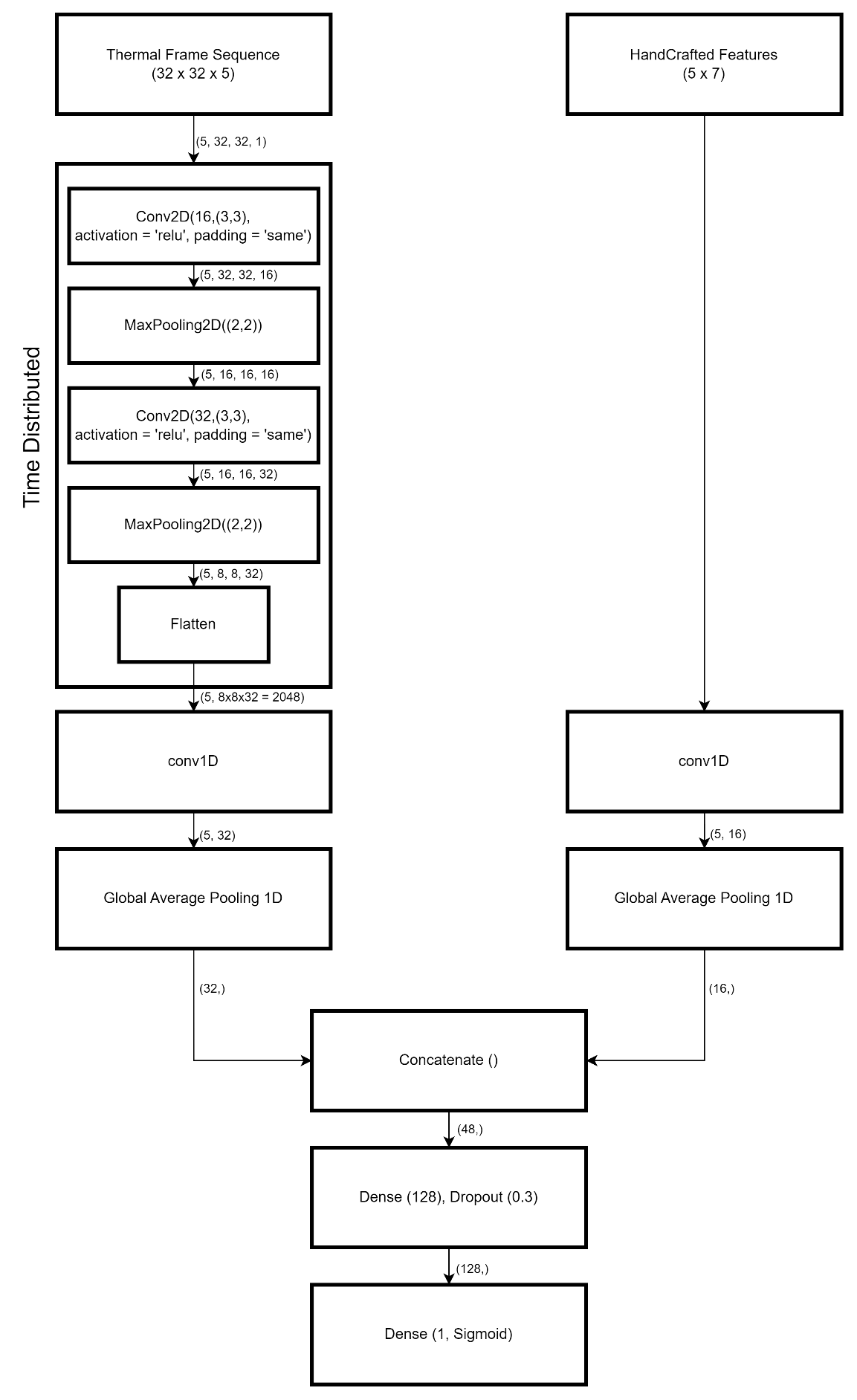
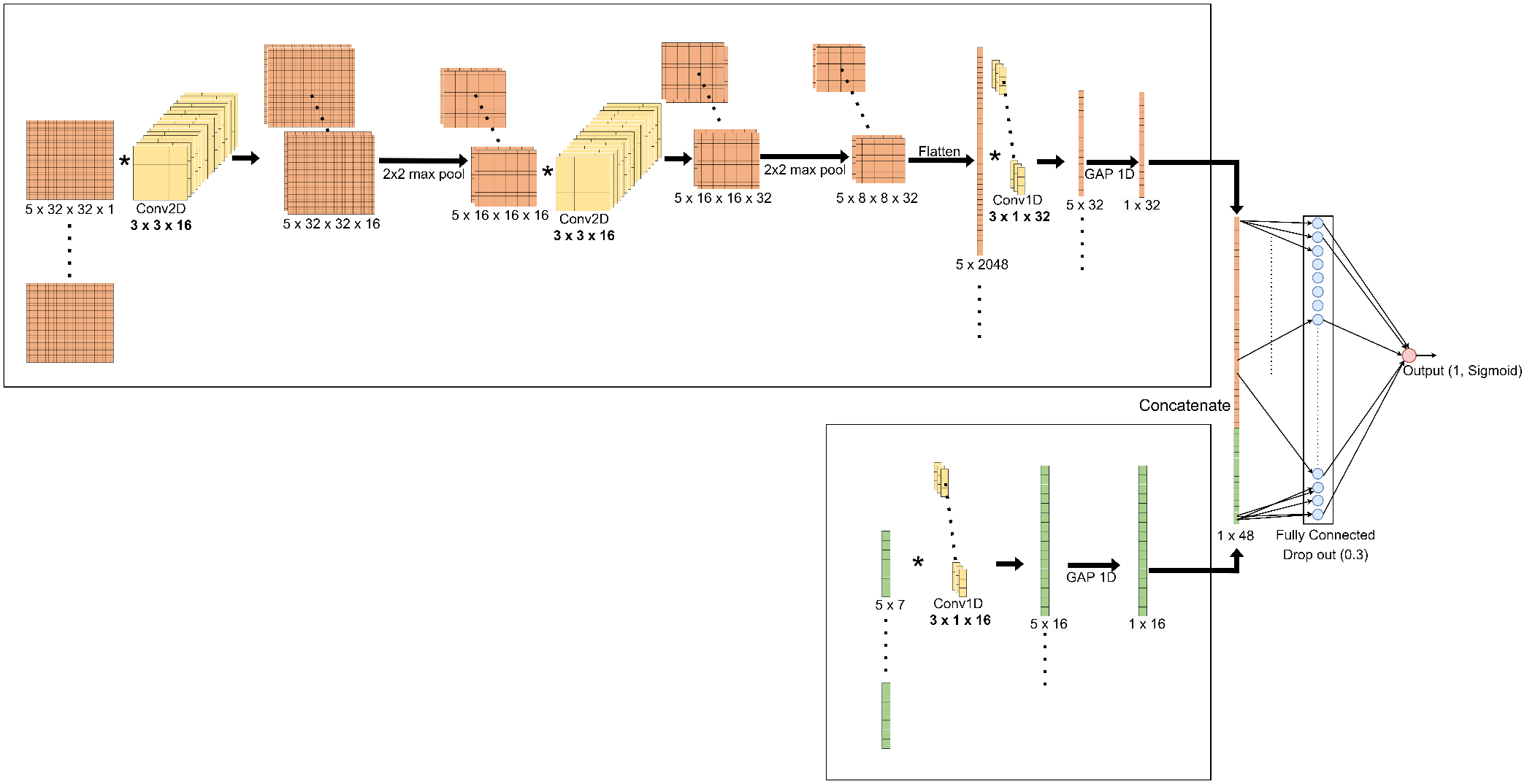


Figure 1: Flowchart of the Temporal CNN model



Note Replace Figure 1 with 2nd Drawing.