No Pain, Just Gains:
Keypoint Prediction For Safer Exercises

50.035 Computer Vision
Final Project
Associate Professor Ngai-Man Cheung

Singapore University of Technology and Design

December 2024

Prepared By: Team 15
Wang Jun Long Ryan (1005923)
Vy Dinh (1006124)
Sarang Nirwan (1006403)
Rachel Leow (1006071)

Contents

1 Introduction

With the third goal of the United Nations Sustainable Development Goals emphasizing the importance of ensuring healthy lives and promoting well-being for all at all ages [1], this project aims to leverage advancements in computer vision to support better exercise practices. Human pose estimation, a subfield of computer vision, focuses on localizing and identifying human body joints in an image or video sequence. By utilizing this concept, our project seeks to track body landmarks and assess whether a user's posture is correct while performing specific exercises, thereby contributing to healthier lifestyles.

To determine whether the user is doing an exercise (for example squats) properly, we need a model capable of making accurate but high-speed inferences, such as the BlazePose model [2] (see Appendix B for details on the original model), about the location of the user's joint locations. Using the coordinates from the joints, we can determine the angles between the joints and obtain a sinusoidal relationship.

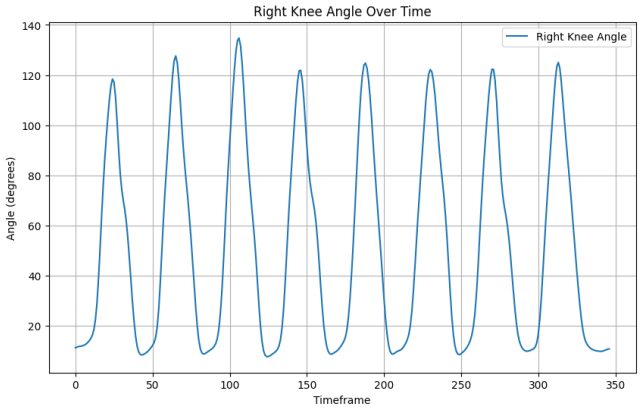

Figure 1: Angle between the right knee, right hip and right ankle against frames

From figure 1, we can see that there is a clear sinusoidal pattern, and by determining the appropriate threshold value we can determine the exercise repetition count. The task of determining body posture is more complicated as the changes in the data pattern are not as clear. Therefore, the human pose estimation model that supplies landmark coordinate data must provide very accurate coordinate data to reduce noise and obtain clear data relationships when a user moves.

In order to receive more accurate data from the human pose estimation model, we modified existing models through our understanding of the U-Net [3] architecture and Vision Transformers [4]. These enhancements aim to improve the model's accuracy in predicting skeletal key points, which would be salient in evaluating exercise forms. The ability to determine proper posture hinges on precise landmark predictions, enabling the system to compute joint angles and establish relationships between them. As the downstream task of predicting proper forms for squats depends on the accuracy of the skeletal keypoints, this project focuses on improving keypoint predictions as a measure of the system's overall capability to provide meaningful and actionable feedback for exercise correction.

2 Project Objectives

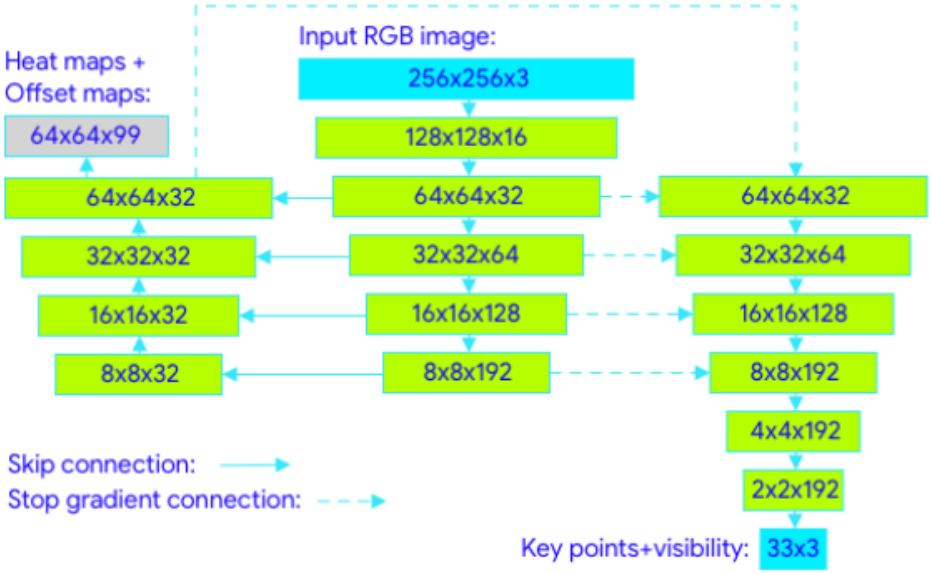

Figure 2: BlazePose architecture from the original paper [2]

Blazepose uses a combined approach of heat maps (left branch) and regression (right branch) to predict keypoints. During training, heatmaps are used to supervise lightweight embedding, which is then utilized by the regression encoder network. Using the current BlazePose architecture as the foundation, we experimented with different approaches to change the architecture and detailed the changes in performance. We aim to

- Modify the existing BlazePose model architecture and train it based on sourced dataset.
- Improve keypoint predictions, using Percentage of Correct Keypoints (PCK) as the evaluation metric.
- Detail trade-offs of the modified component between processing speed and accuracy.

3 Methodology

We adapted the original work (See Appendix B for more details) to form our base model for further experimentation later on. Instead of predicting 33 keypoints, the base model would predict 14 keypoints to align to our chosen dataset. There are three branches in the base mode, the backbone, heatmap branch and the regression branch.

3.1 System Design

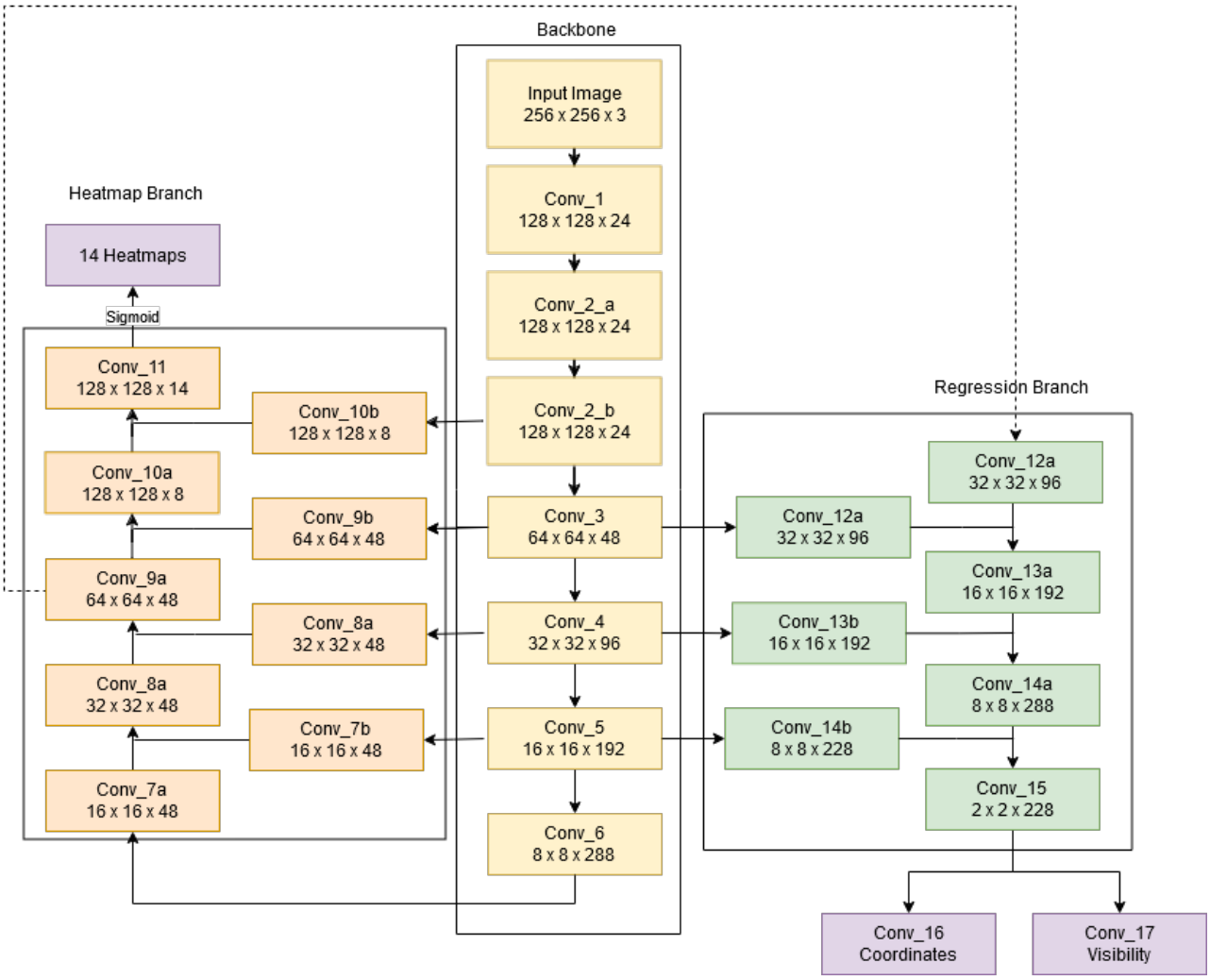

Figure 3: **Base Model** - This project's adaptation of the blazepose model

3.1.1 Backbone

The backbone serves as the feature extraction module of the model. It begins by processing the input image through `Conv_1`, a convolutional layer with a stride of 2, which reduces the spatial resolution while increasing the depth to 24 channels. This initial layer captures low-level image features such as edges and textures. Following `Conv_1`, the backbone employs two separable convolutional layers, `Conv_2_1` and `Conv_2_2`, as implemented in the MobileNet's architecture [5]. These convolutions are a type of factorized convolution that splits the standard convolution into two operations: a depthwise convolution, which applies a single filter to each input channel, and a pointwise convolution, which combines the outputs of the depthwise convolution using 1x1 convolutions. This approach significantly reduces computational cost and the number of parameters while maintaining performance.

3.1.2 Heatmap Branch

After extracting features, the heatmap branch performs a series of upsampling and refinement operations to generate heatmaps that localize each of the 14 keypoints on the input image. Each heatmap corresponds to a specific keypoint and indicates the probability (after passing through a sigmoid activation) of that keypoint being at each pixel location in the image. The upsampling layers (e.g., `Conv_7a`, `Conv_8a`, `Conv_9a`) progressively increase the spatial resolution, while skip connections (e.g., adding outputs from `Conv_7b`, `Conv_8b`, and

Conv_9b) help preserve spatial information from earlier feature maps. The final heatmaps are generated by applying a sigmoid activation function to the outputs of the Conv_11 layer.

3.1.3 Regression Branch

Building upon the feature maps processed by the heatmap branch, the regression branch directly regresses the precise numerical coordinates for each of the 14 keypoints. It begins with intermediate feature maps whose gradients are stopped, ensuring that the training of the regression branch does not interfere with the optimization of the heatmap branch. The regression branch employs a sequence of convolutional blocks (e.g., Conv_12a, Conv_13a, and Conv_14a) that progressively refine the feature representations. Skip connections (e.g., adding outputs from Conv_12b, Conv_13b, and Conv_14b) further enhance the information flow and maintain spatial context.

The final regression layers, Conv_16 and Conv_17, output the coordinates and visibility scores of the keypoints. The coordinates are produced using a dense layer followed by reshaping, while visibility scores are computed with a sigmoid activation to represent the likelihood of each keypoint being visible. This direct regression approach, combined with the structured refinement from earlier layers, ensures precise keypoint predictions suitable for applications requiring high accuracy and robustness.

3.1.4 Model Outputs

The base model produces three distinct outputs during inference, corresponding to different aspects of pose estimation:

i. **Heatmap (H)**

The heatmap of dimensions 128×128 is a spatial probability distribution of keypoints, produced by the heatmap branch. It provides a probabilistic representation of each keypoint's position in the input image. The dimensions of the heatmap output are:

$$H \in \mathbb{R}^{B \times 128 \times 128 \times K}$$

ii. **Coordinates (C)**

The coordinates are numerical (x, y) positions of each keypoint, predicted by the regression branch. These are the precise locations of keypoints in the image, with dimensions:

$$C \in \mathbb{R}^{B \times K \times 2}$$

iii. **Visibility (V)**

The visibility is a probability score indicating whether each keypoint is visible in the input image, also predicted by the regression branch. The dimensions of the visibility output are:

$$V \in \mathbb{R}^{B \times K \times 1}$$

where:

- B: Batch size
- K: Number of keypoints

The final output of the model is structured as:

$$\text{Model}(x) = (H, C, V)$$

where x is the batch of input images, and H, C, and V represent the heatmap, coordinates, and visibility predictions, respectively.

3.2 Training Process

The base model and the experiments will be trained with a similar strategy to the original BlazePose model (See Appendix B.3 for details). Training the base model involved the following steps:

i. Data Preparation from the dataset

ii. Pre-training the Backbone and the Heatmap Branch with a **frozen Regression Branch**.

iii. Training the Regression Branch, with a **frozen backbone and Heatmap Branch**.

3.2.1 Data Preparation

The Leeds Sport Pose [6] (LSP) dataset was used for our training and experimentation. The data preparation process begins with loading the LSP dataset and its annotations, followed by preprocessing the images and generating corresponding heatmaps for training. This is to transform the annotated input data into a format suitable for training the base model.

The annotated keypoints and visibility flags are loaded and reshaped from a `.mat` file using the `loadmat` function and are structured as $(x, y, \text{visibility})$ for each point. Afterwhich, the images are read, resized, and normalized to a fixed dimension of $256 \times 256 \times 3$. The keypoint coordinates in the annotations are adjusted to reflect the resizing of the images to ensure accurate positions relative to the new image dimensions. For each image, Gaussian heatmaps are generated for all keypoints. The `getGaussianMap` function is used to create a 2D Gaussian distribution centered around the joint coordinates, scaled to a heatmap size of 128×128. This involves calculating the usable Gaussian range within the image dimensions and overlaying the Gaussian distribution onto the corresponding heatmap. These heatmaps provide a probabilistic representation of each keypoint location to aid the model in learning spatial patterns for each joint during training.

After preparing the annotations, the training set is prepared as follows:

$$
\begin{aligned}
x_{\text{train}} &\in \mathbb{R}^{N\times 256\times 256\times 3}, \\
y_{\text{train}} &= \left(\text{heatmap}_{\text{train}}, \text{coordinates}_{\text{train}}, \text{visibility}_{\text{train}}\right), \\
\text{heatmap}_{\text{train}} &\in \mathbb{R}^{N\times 128\times 128\times K}, \\
\text{coordinates}_{\text{train}} &\in \mathbb{R}^{N\times K\times 2}, \\
\text{visibility}_{\text{train}} &\in \mathbb{R}^{N\times K\times 1},
\end{aligned}
$$

where N is the number of training samples, and K is the number of keypoints. The validation and test sets follow the same structure as the training sets where N would be adjusted for the number of validation or test samples.

Data Splits: The dataset is then split into training, validation, and test sets based on predefined proportions in the config.py file. The default parameters are 60%, 20% and 20% for training, validation and testing respectively. The images (x) and their corresponding outputs (y)—including heatmaps, coordinates, and visibility—are divided into these subsets. The training data is used to optimize the model, the validation data is used to monitor performance and tune hyperparameters, and the test data evaluates the model's generalization ability.

Data Augmentation: Initially, 2000 images from the Leeds Sports Pose (LSP) dataset [6] were utilized for training the model. This dataset was too small to train a robust model and thus we decided to perform some data augmentation from the LSP dataset to increase training data size. We flipped the existing 2000 images and their corresponding ground truth joint coordinates and heatmaps. This doubled our training data to 4000 images.

(a) Training image with joints annotated

(b) Flipped training image with joints annotated

Figure 4: Comparison of training images

3.2.2 Pretraining

The pretraining process focuses on training the backbone and heatmap branch of the base model to localize keypoints accurately through the heatmaps. The objective is to train the model to learn important features that contribute to the probability distributions of keypoint locations. This stage serves as a foundation for the subsequent regression branch training.

The prepared training data (See Section 3.2.1), consisting of resized images x_{train} and their corresponding labels y_{train}, is fed into the model. Training is performed for a predefined number of epochs with a batch size (specified by `total_epoch` and `batch_size` in `config.py` respectively). Validation data (x_{val}, y_{val}) is used to monitor the model's performance during training, providing a mechanism to detect overfitting.

The model architecture is instantiated using the `BlazePose` class, and it is compiled with a combination of loss functions tailored to each output: binary cross-entropy (`BCE`) for the heatmap output, mean squared error (`MSE`) for the coordinates, and binary cross-entropy for the visibility predictions. The optimizer used is Adam with a learning rate of 0.001, providing a balance between stability and efficiency during training. To ensure that only the backbone and heatmap branch are trained during this phase, **layers associated with the regression branch are frozen** by setting their `trainable` attribute to `False`. This prevents the weights in the regression branch from being updated, allowing the model to focus on refining the heatmap predictions. Additionally, callbacks were utilized to facilitate model checkpointing and logging. The `ModelCheckpoint` callback saves the model's weights after each epoch, enabling recovery in case of interruptions and allowing evaluation of the model at various training stages.

3.2.3 Training Regression Branch

The regression branch training focuses on refining the precise coordinates and visibility predictions of the keypoints, leveraging the backbone and heatmap branch outputs learned during pretraining. The objective is to predict the numerical coordinates and visibility scores of the keypoints.

The process for training the regression branch will be the same as training the backbone and heatmap branch, except that **the weights for both the backbone and heatmap branch are frozen**. This means that the model is still compiled with the same combination of loss functions (BCE, MSE, BCE) for each corresponding heatmap, coordinates and visibility output, and optimized with an Adam optimizer. During training, the model leverages on the features extracted by the backbone and the intermediate heatmap representation from the heatmap branch,

using them as input to the regression branch.

4 Experimentation

4.1 Architecture Modifications

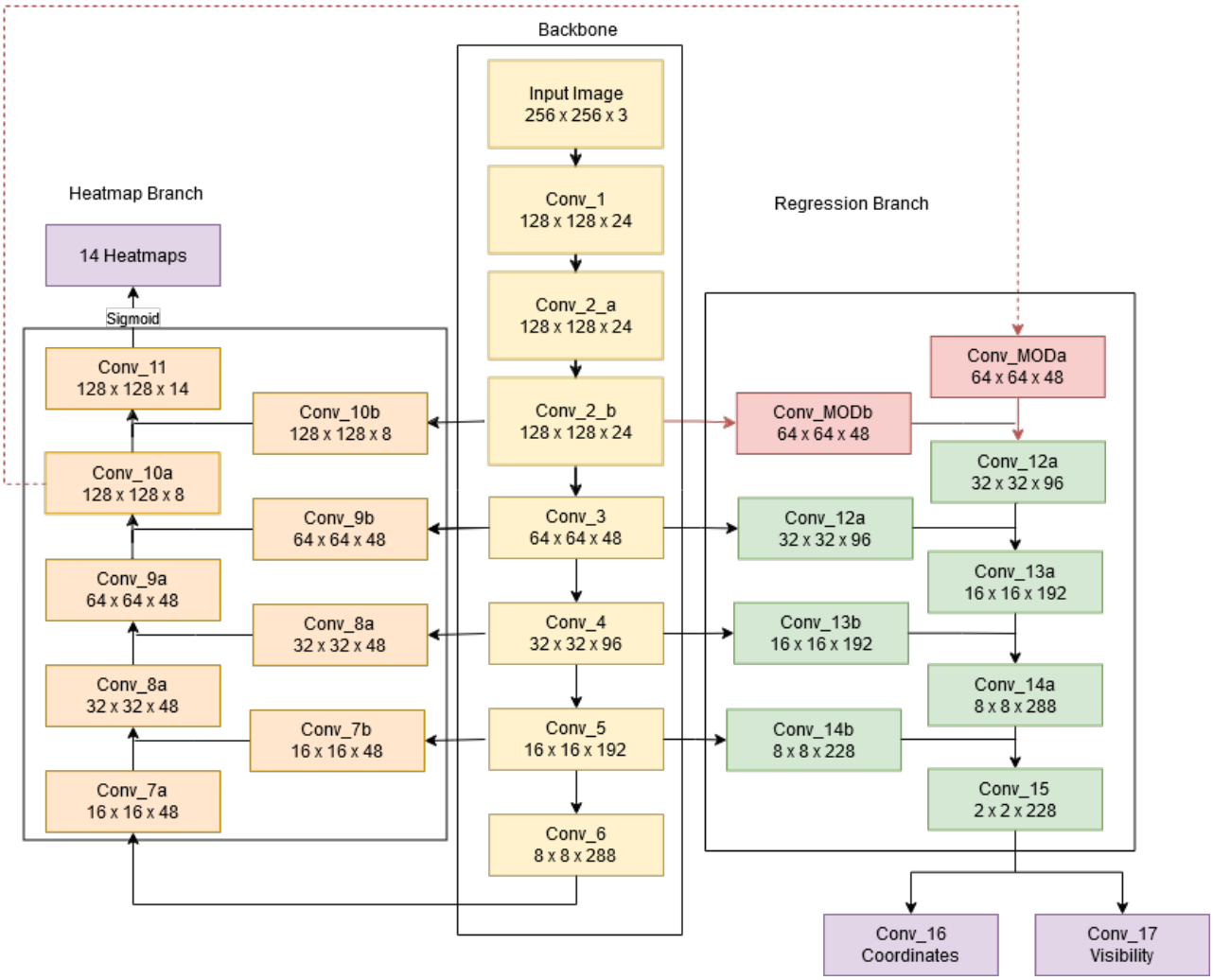

Figure 5: Initial modifications to the base architecture

The initial adaptation of the blazepose model (See Figure 3) utilizes the intermediate heatmap representation from the Conv_9a layer in the regression branch. Hence, our initial experimentation involved modifying the heatmap branch to improve representations and output more accurate heatmaps that correspond to each joint.

We first experimented with using a higher resolution heatmap (Conv_10a) representation and connected it to the regression branch (See Figure 5). To accept this new input, two more convolutional blocks were constructed (Conv_MODa and Conv_MODb) to maintain a similar structure to the rest of the regression branch. The main motivation for using a higher resolution representation is to capture more details and nuances from the probability distribution of each joint. However, this affected the training of the heatmap branch. As the skip connection is now higher up in the upsampling branch (adding to the inputs before Conv_10a), the gradients in the pre-training phase would also propagate to Conv_9a and Conv_8a and their upstream modules which affected the confidence and precision negatively (See Figure 6).

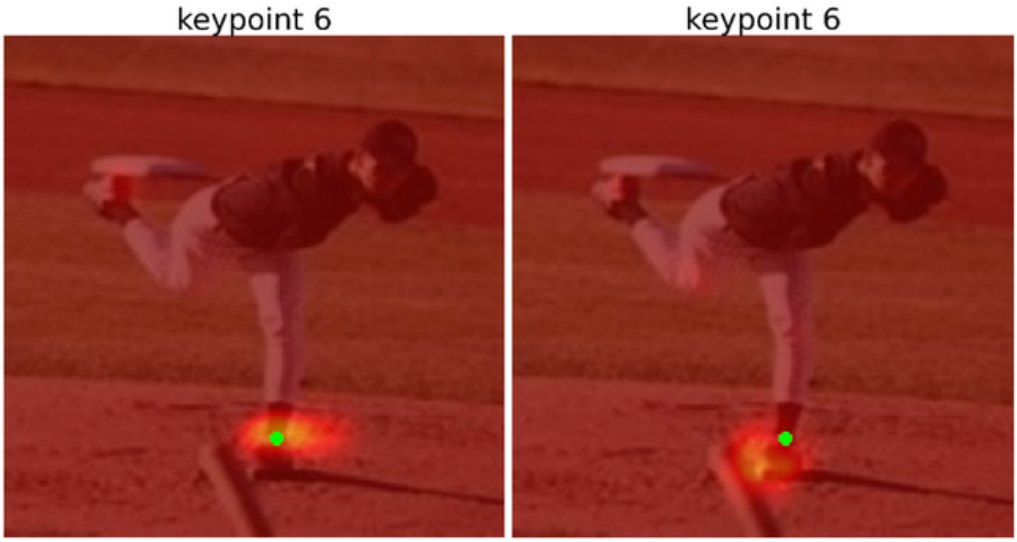

Figure 6: Heatmap comparison of predictions. **Left**: Initial adaptation. **Right**: After modifications. Lower confidence and precision in predicting the left ankle (keypoint 6) after modifications. The brightness of the spot indicate a higher probability for the keypoint.

To address the affected probability distribution of the output heatmaps, we also introduced transpose convolutions within the heatmap branch as opposed to a bilinear interpolation in the up-sampling stream (Conv_6 to Conv_11). The main motivation for this change is to learn features for more robust upsampling and obtain more accurate heatmaps to negate the downsides of our initial modifications. As a trade-off, the model would take longer to train during the pre-training phase. However, the introduction of transpose convolutions had minimal to no impact on the heatmap outputs or the PCK scores. This outcome is likely attributed to the limited robustness of the dataset, as the training set, though of high quality, comprises of 1,600 unique images, with 200 images reserved for validation and testing each.

4.2 Heatmap Loss Modifications

We also observed that the model regularly mixed up the location of opposing joints (e.g. left ankle and right ankle), as seen in Figure 7. We thus had the idea to experiment with adjusting the heatmap loss such that it penalizes the region of the opposing joint more severely.

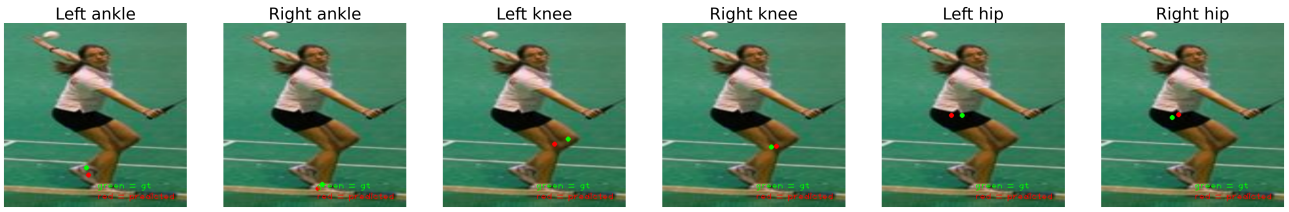

Figure 7: Example of the base model mixing up mirrored joints.

Originally, 14 heatmaps were generated for each input example for joint j = 1..14. Each heatmap H_j was a 128x128 matrix with domain [0, 1], with increasing intensity closer to the specified joint region. This increase in intensity followed a 2D gaussian distribution. The loss function used was Binary Cross Entropy (BCE). For this experiment, we further generated the heatmap $H_{j'}$ for each mirrored joint j'.

In **attempt 1**, we subtracted the original heatmap H_j by the negative heatmap $H_{j'}$ and took the result $A = H_j - H_{j'}$ as the new target heatmap. The resultant heatmap A has domain [-1, 1], wherein intensity is close to 1 for joint j and close to -1 for the opposing joint j', and the in-between region as well as background regions have intensity = 0. To adjust to the new domain, we also replaced the activation of Conv_11 in the heatmap

branch from *sigmoid* to *tanh*, and experimented with replacing the loss function by smooth L1 loss and Huber loss. Unfortunately, the loss stagnated after 20 epochs and the train PCK score remained at 0.22.

In **attempt 2**, we added the negative heatmaps $H_{j'}$'s as another output to Conv_11 with the same *sigmoid* activation, but with a new loss function:

$$L = \begin{cases} -\frac{1}{N} \sum y_i \cdot log(1 - \hat{y}_i) + (1 - y_i) \cdot log(\hat{y}_i), & \text{if } y_i > 0 \\ 0, & \text{otherwise} \end{cases}$$

Here, y_i represents a single pixel intensity in [0, 1] within the ground truth negative heatmap $H_{j'}$. This is equivalent to calculating BCE($H_{j'}$, $1 - \hat{H}_j$) in the region surrounding the mirrored joint j'. Minimizing L means maximizing the difference between the predicted heatmap and the heatmap corresponding to the mirrored joint. The objective is to penalize the intensity values in the mirrored joint region specifically. We also experimented with other loss functions (with the same negative maximizing) including smooth L1 and MSE.

The generated heatmaps are shown in Figure 8, and our test PCK was 0.437. While the inclusion of negative heatmaps made differentiating between mirrored joints slightly clearer (see Figure 9), the predicted pixel locations became less accurate, resulting in a worse PCK score. Our hypothesis was that an additional output might have confused the model, and thus the extracted features became less precise.

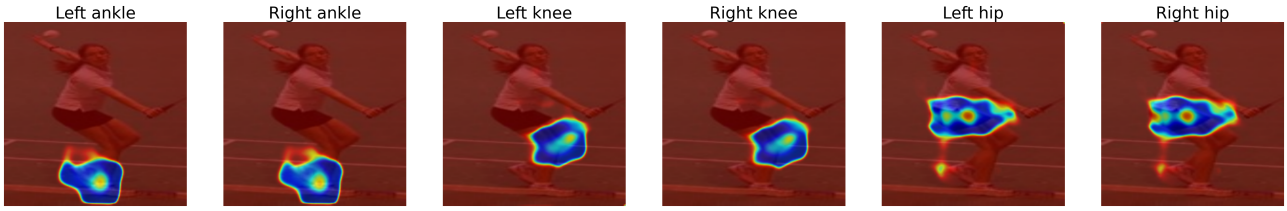

Figure 8: Attempt 2's generated heatmaps.

Figure 9: Attempt 2's predictions.

4.3 Attention Might Be What We Need

Despite our initial efforts in modifying the base model and experimenting with the heatmap loss functions (Sections 4.1 and 4.2), there were clear limits on modifying the base architecture alone. The BlazePose architecture inherently lacks the ability to model long-range dependencies, resulting in output heatmaps that are unaware of the positions of other keypoints, and struggle to accurately determine if a keypoint belongs to the left or the right joint of the target.

In an effort to introduce novelty and address the limitations of convolutional networks in capturing long-range dependencies, transformer blocks were introduced into the backbone of the model. While convolutional layers are highly effective at extracting local spatial features, they often struggle to model relationships across distant regions of an image. By incorporating attention mechanisms, the model would be able to better capture global spatial relationships. This integration enhances the model's ability to represent complex features, complementing the strengths of the convolutional backbone. To this extent, we experimented with two different mechanisms, transformer modules [4] and Convolutional Block Attention Modules (CBAM) [7].

4.3.1 Transformer Modules Modifications

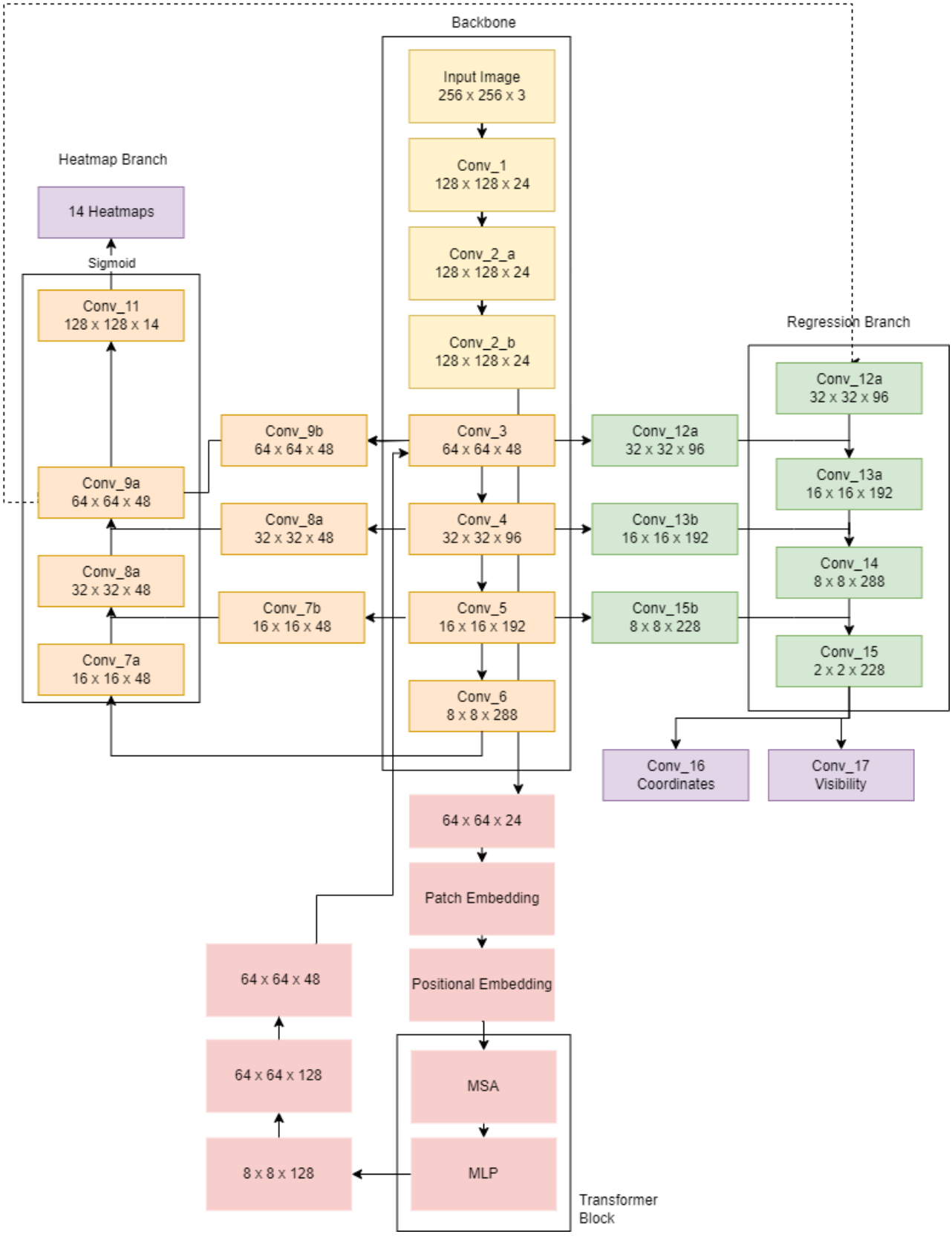

Figure 10: Addition of Transformer Modules to the base model

Initial iterations combined the base model's backbone with modules from the Vision Transformer (ViT). The backbone captures local features, while the ViT modules help model global relationships in an image.

Their combination allows the model to effectively leverage both local feature extraction and global context understanding. This pretraining enables the CNN to extract meaningful features from images, while the ViT enhances these representations for improved performance in downstream tasks. The ViT modules are placed after the backbone to ensure that the transformer has access to the complete spatial features extracted by the backbone. This placement enables the self-attention mechanism to focus on important spatial relationships across the entire feature map, allowing the model to learn where to focus in different regions by weighting the importance of each spatial location. This design is also intended to be modular, allowing the CNN backbone to utilize existing pretrained models (such as ResNet [8] or MobileNet [5]) and then be fine-tuned with the added ViT modules. Multiple blocks of these modules can be stacked to experiment with and optimize the number of layers that yield the best results.

The ViT modules consist of three main components:

i. **Patch Embedding**: Divides the input image into patches and maps each patch into a fixed-dimensional vector representation.

ii. **Positional Embedding**: Adds positional information to the patches, enabling the transformer to consider spatial relationships.

iii. **Transformer Block**: The core building block, which includes:

   (a) Multi-Head Self-Attention (MSA): Captures dependencies between patches, enabling the model to focus on relevant regions.

   (b) Multilayer Perceptron (MLP): Processes the outputs of the MSA mechanism and enhances feature representations.

The Vision Transformer modules (ViT) adds on to the backbone of the base model (See Figure 10). After passing through `conv_2_a` and `conv_2_b`, the output is downsampled from 128×128 to 64×64, while maintaining the number of channels. Reducing the spatial dimensions makes the transformer more computationally efficient. Furthermore, higher-resolution features often preserve fine-grained details, enabling the attention mechanism to extract richer contextual information from the patches. Patch embedding and positional embedding are applied to divide the image into smaller patches and provide locational information for each patch. As the data passes through the transformer layers, each patch is processed using the self-attention mechanism. This mechanism allows the model to simultaneously consider all patches, capturing long-range dependencies. The intention is to enable the model to identify relationships between different parts of the image, even if they are spatially distant. To facilitate gradient flow and improve training stability, residual connections such as $x + \texttt{attn\_output}$ and $x + \texttt{mlp\_output}$ were added. These skip connections were added to address the vanishing gradient issues encountered during the initial iterations of this architecture.

The shape of the output from the final transformer layer is (batch size, number of patches, embed dimensions). To integrate this output with the heatmap branch, it is reshaped from 64×128 to $8 \times 8 \times 128$. Subsequently, the output is passed through a `conv_trans_upsampling` layer, which resizes it to $64 \times 64 \times 128$. Finally, a 1×1 convolutional layer with 48 filters is applied to reduce the number of channels, resulting in an output of $64 \times 64 \times 48$.

The final output of the Transformer Modification is described as follows:

$$\text{Final Output} \in \mathbb{R}^{B \times 64 \times 64 \times 48},$$

where B is the batch size, 64×64 represents the spatial dimensions of the feature map, and 48 is the number of output channels.

In addition to the current CNN-ViT model implemented, calling it ViT2, we further optimized it. In the `Patch_Embedding` class, we used a convolutional layer instead of a dense layer. The convolutional layer acts as a sliding window that directly extracts patches from the input image and embeds them into a higher-dimensional space in one operation. A dense layer, on the other hand, would require flattening the image first, breaking the spatial structure, and manually splitting it into patches, which is computationally more expensive and less

intuitive. Additionally, convolutional layers share weights across the spatial dimensions of the image, making them much more parameter-efficient than dense layers.

Another optimization we did is the portion where we reshaped the output of the ViT so that it can be passed into the heatmap brunch. We used transposed convolution to up sample the output instead of the fixed interpolation bilinear method. Transpose convolution will also apply convolutional operation to refine features and generate richer representations.

4.3.2 CBAM Modules Modifications

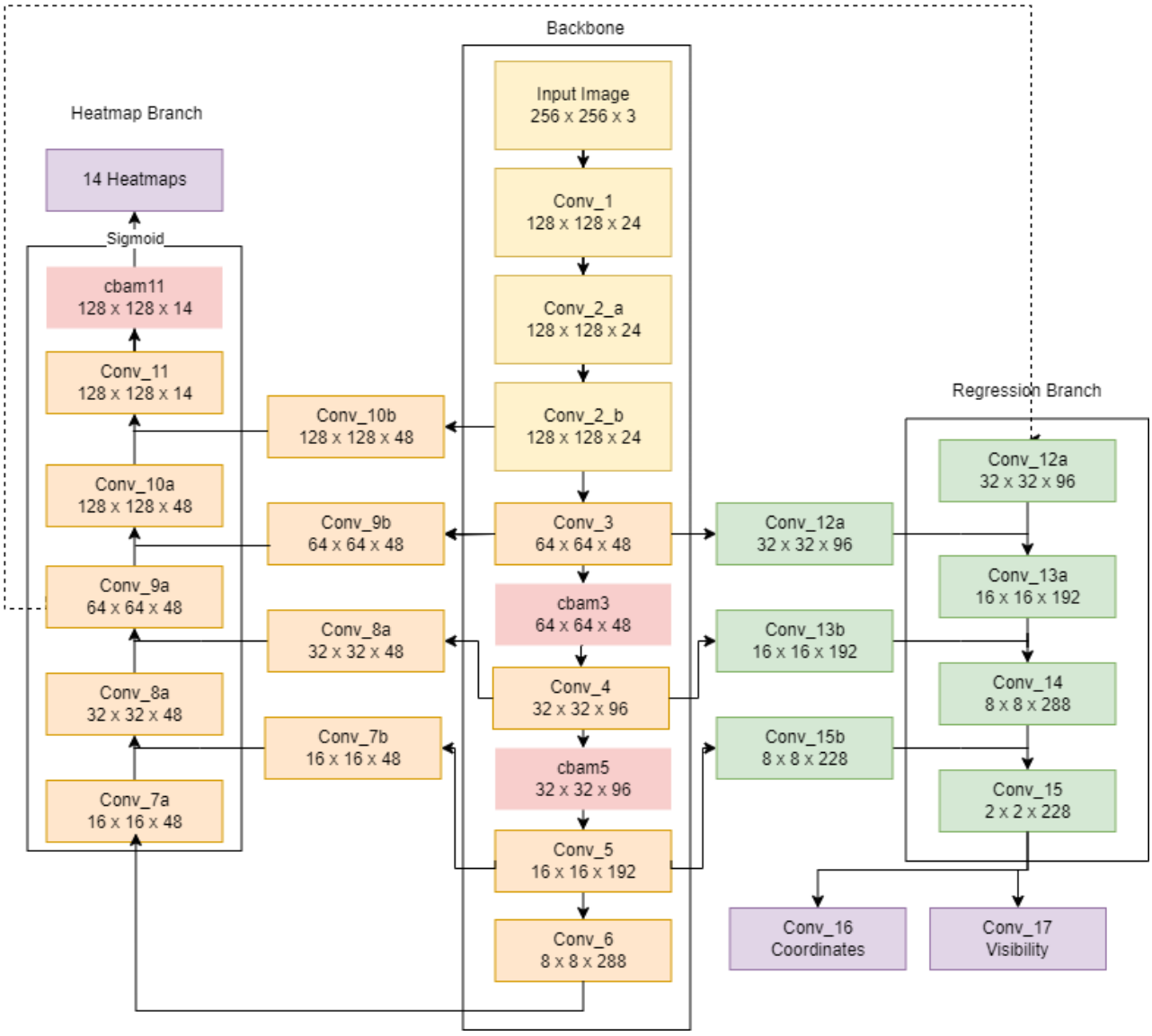

Figure 11: Addition of CBAM Module to the base model

In a separate experiment, Convolutional Block Attention Modules [7] (CBAM) were introduced to the base model as an alternative for the attention mechanism. Each CBAM consists of two sequential segments: the Channel Attention (CA) and the Spatial Attention (SA). CA identifies which feature maps are most relevant, while SA pinpoints the critical regions within those maps. The objective is to refine the feature maps, improving their contribution to the overall model performance.

In each CBAM, CA calculates channel-wise attention weights using global average and max pooling operations whilst retaining the input dimensions of the previous layer. These weights are passed through two fully connected layers, where the first reduces the dimensionality using a ratio of 16 and the second restores it

to match the original input channels. The output of CA is added to the input features to refine their importance. This refined output is then passed through the SA module, which concatenates spatial average and max pooling results before applying a 7x7 convolution to generate a spatial attention map. The final attention-enhanced features from SA are used as input to the next layer.

The CBAM modules are placed after `conv3`, `conv5`, and `conv11` to target specific stages of feature processing within the network. After `conv3`, CBAM refines low-level features early, ensuring subsequent layers receive more meaningful inputs. Placing CBAM after `conv5` enhances mid-level features by amplifying spatial details and abstract patterns, critical for robust intermediate representations. Lastly, applying CBAM after `conv11` focuses on optimizing high-level feature maps, improving the model's attention to critical areas just before generating the heatmap and regression outputs. This selective placement balances computational efficiency and performance improvement.

5 Discussion

The model predictions for the base model (See Figure 12) are generally good for limbs that are visible and clear. However, the model still has issues predicting keypoints with occluded limbs and (lsp_1751 in Figure 12) or noisy images with multiple people in the frame (lsp_1721 in Figure 12). More sample outputs can be observed in Appendix C, Figure 16.

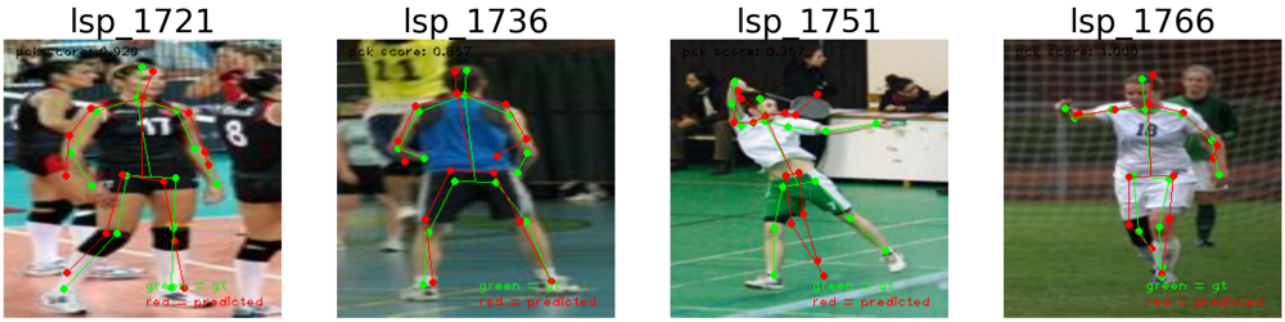

Figure 12: Sample outputs from the base model

5.1 Ablation

Model Variants	PCK Score (%)	Latency (ms)
Base Model	**58.5**	116
Architecture Modifications	39.1	290
Heatmap Loss Modifications	43.7	**93**
ViT Modules Modifications	41.7	117
CBAM Modules Modifications	42.0	124

Table 1: Comparison of Model Variants on PCK Score, Latency, and Total Loss

Evaluation Metrics:

i. **Percentage of Correct Keypoints (PCK):** This metric measures the accuracy of the model's predicted keypoints. A keypoint prediction is considered correct if it lies within a certain threshold distance from the ground truth. The PCK score is computed as:

$$\text{PCK} = \frac{\text{Number of Correct Keypoints}}{\text{Total Number of Keypoints}} \times 100$$

where the threshold distance is expressed as a fraction of a reference length using the head size. The standard threshold is 0.5, which we used.

ii. **Latency:** The time taken by the model to process a single image, measured in milliseconds (ms). This metric reflects the computational efficiency of the model, making it particularly important for real-time applications.

5.2 Limitations

Given the limited time and computational resources available for this project, we chose a dataset that was small and manageable to enable easier experimentation and training. However, this would also become the main limitation for our model and experiments. Though the LSP dataset have high quality annotations with unique poses for different sports, the size of the dataset (2000 samples) was not sufficient in training more complex models. There is a larger version of the LSP dataset, named LSPET, with 10,000 samples annotated by Amazon Mechanical Turk workers. However, this dataset was lower in quality than the original LSP set as some keypoints were either plotted incorrectly with extreme values (See Figure 13) or, to account for occluded limbs (the dataset was not clear on the labels provided). As a result, the LSPET set heavily affecting the pre-training process and the negative effects were cascaded into the regression branch.

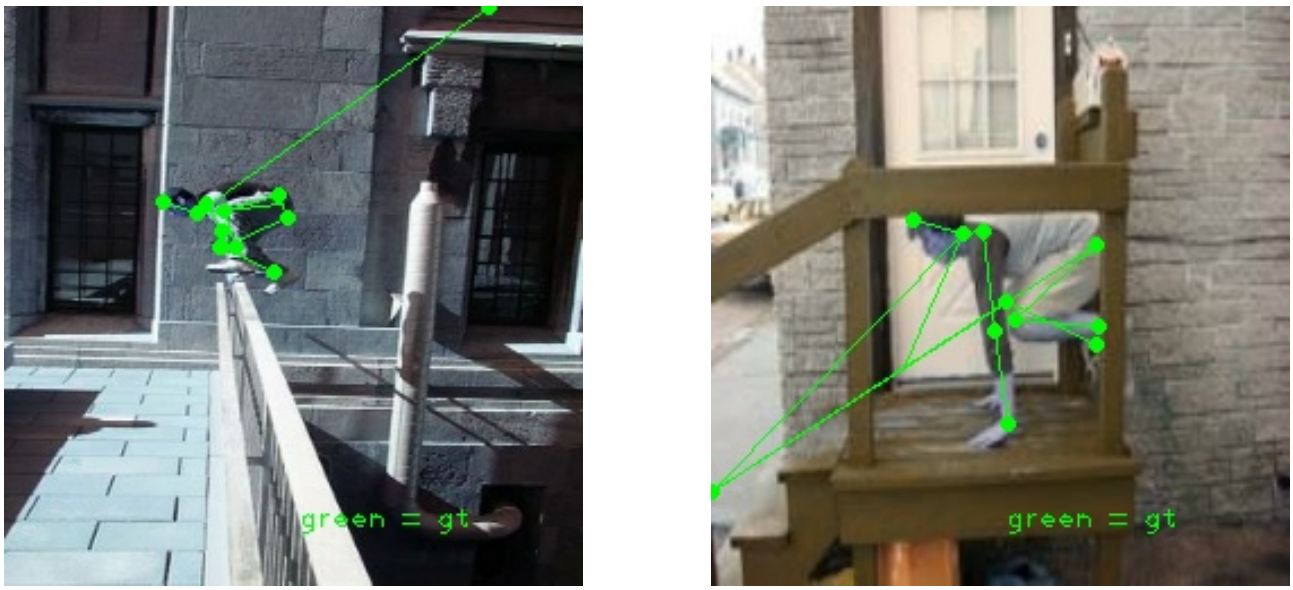

Figure 13: Samples of extreme values in the LSPET dataset

The traditional pain points in human pose estimation, including occluded body parts, blending of opposing joints, and blending of body parts into the background, remain. Occlusion occurs when parts of the body are obscured by other objects, other body parts, or external factors such as clothing. Our experimentations in using transformer models, attention blocks, and heatmap loss variations had aimed to establish relations between different body parts, producing a basis from which the location of occluded joints could be predicted. However, future research is needed to further alleviate this problem.

In addition, while our data augmentation was successful, it might be simplistic and might not capture the full variability of real-world poses and conditions, such as lighting, motion blur, or camera angles. Furthermore, unseen poses or top-down angles posed a challenge to our current model. More advanced augmentation techniques, including rotation, scaling, synthetic data generation, and noise generation, could enhance training diversity.

5.3 Future Work

Future work could capitalize on more recent works with higher quality and larger volume data, such as the JRDB-Pose [9] and H3WB [10] datasets, to have fairer ablations for the modifications made. These newer and larger datasets are also comprehensive with different human poses to help the model generalize better.

Further, more experiments can be done to modify the base model to incorporate other modalities of data. Recent works have introduced datasets with diverse perspectives such as a dataset with fisheye perspectives [11] or a dataset with omnidirectional top-view images [12]. These additional modalities can enable more novel modifications to attain better accuracy for keypoint predictions.

We can also work on more data augmentation techniques whilst utilizing a GPU that could handle the large amounts of data to be used as input. In particular, one of the methods we plan to explore is adversarial augmentation, such that we can leverage on the power of GAN to generate rarer poses and imbalanced demographics (skin tones). In addition, incorporating temporal information to enhance pose estimation in video sequences is another avenue we wish to figure out. With RNNs and more sophisticated transformer networks, the model would better capture motion dynamics and ensure consistency across frames. Combining this additional temporal information with the spatial relations in our transformer architecture additions might be able to extract more precise features in the human body.

These future directions could help address existing limitations, such as motion blur and occlusion of body part, which is important in our goal to predict the location of joints and infer the correctness of users' poses during real-time exercises.

5.4 Conclusion

Overall, the adapted Base Model demonstrated reliable keypoint prediction under the given time and computational constraints, but faced challenges with occlusions and complex backgrounds. Addressing these issues requires richer and more diverse datasets, data augmentation techniques, and methods that handle occlusions and temporal information more effectively. Incorporating these improvements would further improve keypoint predictions, and bring the system closer to fulfilling its original aim as outlined in the introduction: providing precise, actionable feedback to promote healthier exercise practices. By enhancing the model's robustness and accuracy, we can better support the United Nations Sustainable Development Goal of ensuring healthy lives and well-being, ultimately enabling more reliable posture evaluation and exercise guidance for users.

5.5 Source code

The code for the different models as well as instructions on training, testing and processing can be found on the GitHub repository.

References

[1] United Nations, *The 17 goals — sustainable development*, Accessed: 13 December 2024, n.d. [Online]. Available: `https://sdgs.un.org/goals`.

[2] V. Bazarevsky, "Blazepose: On-device real-time body pose tracking," *arXiv preprint arXiv:2006.10204*, 2020.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, Springer, 2015, pp. 234–241.

[4] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[5] A. G. Howard, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[6] dbcollection, *Lspe - leeds sports pose extended*, Accessed: 2024-12-13, n.d. [Online]. Available: `https://dbcollection.readthedocs.io/en/latest/datasets/leeds_sports_pose_extended.html`.

[7] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[9] E. Vendrow, D. T. Le, J. Cai, and H. Rezatofighi, "Jrdb-pose: A large-scale dataset for multi-person pose estimation and tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4811–4820.

[10] Y. Zhu, N. Samet, and D. Picard, "H3wb: Human3. 6m 3d wholebody dataset and benchmark," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 20 166–20 177.

[11] J. Yu, D. Nandi, R. Seidel, and G. Hirtz, "Ntop: Nerf-powered large-scale dataset generation for 2d and 3d human pose estimation in top-view fisheye images," *arXiv preprint arXiv:2402.18196*, 2024.

[12] J. Yu, T. Scheck, R. Seidel, Y. Adya, D. Nandi, and G. Hirtz, "Human pose estimation in monocular omnidirectional top-view images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 6411–6420.

[13] kobiso, *Github - kobiso/cbam-keras: Cbam implementation on keras*, GitHub, 2018. [Online]. Available: `https://github.com/kobiso/CBAM-keras` (visited on 12/13/2024).

A Contributions & Acknowledgements

Member Name	Student ID	Contributions
Wang Jun Long Ryan	1005923	i. Implemented and trained the base model (Section 3). ii. Experimented on different architecture modifications (Section 4.1). iii. Experimented with CBAM modules (Section 4.3.2). iv. Wrote and edited the entire report.
Vy Dinh	1006124	i. Implemented and trained the base model (Section 3). ii. Initial coding of the CBAM layer based on [13]. iii. Experimented with negative heatmap modification (Section 4.2).
Sarang Nirwan	1006403	i. Implemented and trained the base model (Section 3). ii. Implemented video processing logic iii. Worked on the report introduction and problem background iv. Github code refactoring
Rachel Leow	1006071	i. Implemented and trained the base model (Section 3). ii. Implemented and researched ViT modules (Section 4.3.1). iii. Experimented with optimizing the ViT modules (Section 4.3.1).

Table 2: Summary of workload and contributions for team members.

This report was written and prepared by the aforementioned team members. To enhance the clarity and quality of the document, Large Language Models (LLMs) were utilized for refining the formatting and correcting linguistic errors, including grammar and spelling.

We extend our gratitude to Associate Professor Ngai-Man Cheung for his valuable guidance and advice throughout the course of this project. Additionally, we would like to acknowledge GitHub user alishsuper for

their implementation of the BlazePose model, which significantly contributed to the construction of our base model.

B Blazepose Model

BlazePose [2] is a lightweight, real-time pose estimation model developed by Google Research, designed to detect and track human body keypoints (See Figure 14) in videos or images. It identifies 33 key landmarks, including those for the face, torso, arms, and legs, enabling precise full-body motion tracking. This model is well-suited for mobile and web applications, achieving high performance on devices with limited computational resources. BlazePose leverages a two-stage pipeline: first, detecting the region of interest (ROI) to focus on the subject, and second, refining keypoint localization using a neural network. It has been applied in fitness, augmented reality, and healthcare for tasks like exercise tracking, posture correction, and interactive experiences.

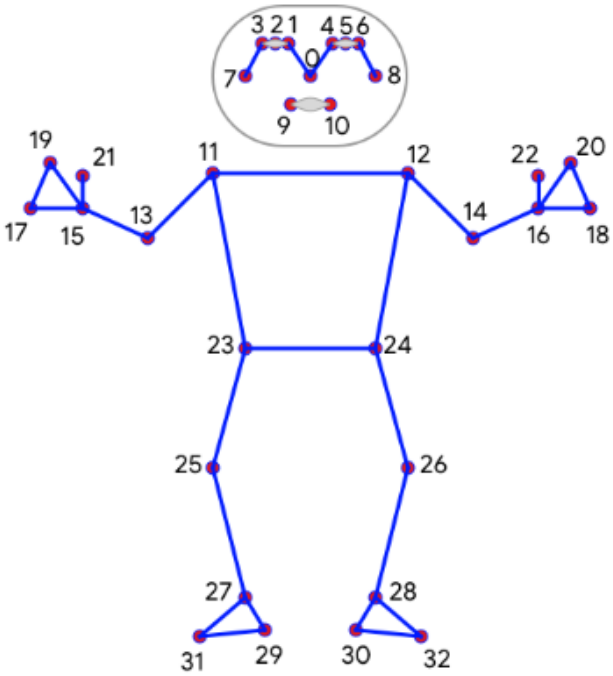

Figure 14: Keypoints predicted in the original work

B.1 Dataset Preparation

The authors of BlazePose constructed the training dataset by combining Google's in-house resources with publicly available data. The dataset comprised 60,000 images depicting one or a few individuals in common poses and an additional 25,000 images featuring fitness-related activities. Each image was meticulously annotated with 33 body keypoints, ensuring that the dataset provided high-quality ground truth for training. The annotation process followed a topology consistent with BlazeFace, BlazePalm, and MS COCO datasets, allowing seamless compatibility with related tasks and datasets.

To enhance the model's robustness, the authors employed significant data augmentation techniques. These included simulating occlusions by overlaying random rectangles filled with varying colors to mimic scenarios where body parts were partially or entirely obscured. Additional augmentations involved scaling, rotation, and translation adjustments, ensuring that the model could generalize effectively across diverse poses and environments.

The authors further refined the dataset by filtering it to include only images where the entire person was visible or where critical keypoints, such as hips and shoulders, could be confidently annotated. This refinement

step ensured that the model received consistent and reliable training data. Moreover, a visibility classifier was incorporated into the training process to identify whether specific keypoints were visible or occluded. This feature allowed the model to maintain accuracy under challenging conditions, such as when only the upper body was visible or when most of the subject's body was outside the frame.

B.2 Model Architecture

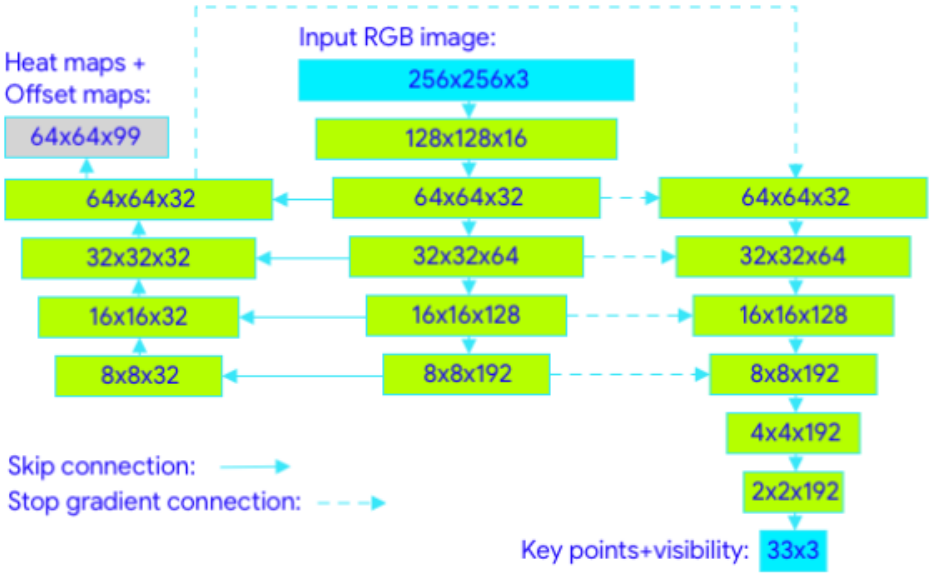

Figure 15: BlazePose model architecture.

BlazePose adopts a hybrid methodology that integrates heatmaps, offsets, and regression for pose estimation. During training, a lightweight encoder-decoder network produces heatmaps that act as guidance for detecting body keypoints. These heatmaps serve as a form of supervision, enabling the regression encoder to estimate the precise coordinates of the keypoints. To enhance efficiency during inference, the heatmap layers are removed, allowing the model to perform effectively on devices with limited computational resources.

B.3 Training in Phases

To refine the network further, the authors incorporated skip connections to ensure a balance between extracting high-level and low-level features. Gradients were intentionally blocked at specific layers to prevent conflicts between heatmap-driven and regression-based learning. This approach found that the heatmap quality and the accuracy of the coordinates greatly improved. Hence, we adopted a similar training strategy in our project to achieve better results. The regression encoder of the original model was designed as a compact module which we try to maintain in our implementation as well.

In the original work, alignment and augmentation processes also played a vital role in preparing the training data. The images were preprocessed to center the subject, aligning the midpoint between the hips at the center of the frame. Rotational adjustments ensured that the line connecting the shoulders and hips was vertically oriented. Additionally, scaling was applied to fit all keypoints within a square bounding box. Augmentations, including small variations in scale and position, simulated natural body movements to improve the model's robustness. The authors found that these strategies contributed to the model's ability to generalize effectively across various poses while maintaining accuracy during inference. Inspired by this, we also implemented various alignment and augmentation pipelines into this project.

C Sample Outputs

Figure 16: Sample outputs for base model.