# Task 1

##### 1. Model & Techniques

* AI Model: MediaPipe BlazePose (Lite version) – Optimized for mobile, provides 3D landmarks (33 keypoints) in real-time.
* Unity Integration:
  + Use Unity’s Barracuda (for on-device ML) or MediaPipe Unity Plugin for direct inference.
  + Convert pose landmarks to Unity GameObjects for visualization.
* Feedback System:
  + Calculate angles (e.g., elbow/shoulder) to detect posture deviations.
  + Use color-coded UI (e.g., red/green) for immediate feedback.

##### 2. Potential Challenges

* Performance: Mobile CPUs/GPUs may struggle with heavy models → Optimize with quantization (e.g., TFLite).
* Latency: Real-time tracking requires <30ms/frame → Use model pruning and multi-threading.
* Occlusions: Hands/objects blocking the body → Augment training data with synthetic occlusions.
* Variability: Different body types/lighting → Test on diverse datasets.

# Task 2

**B**. Can you explain how synthetic data helps improve model accuracy in motion tracking?

**Answer**:

* - Real data may lack diversity (e.g., limited lighting/backgrounds/body types), Synthetic data introduces controlled variations
* - Real-world annotations (e.g., 3D joint positions) are noisy or incomplete. Synthetic data provides pixel-perfect labels (joint coordinates, segmentation masks) automatically.
* - Synthesize bowing motions in Blender with precise 3D joint labels. Add variations:
  + - Different violin sizes
    - Left-handed players
    - Occlusions from music stands
* Fine-tune the model on synthetic + real data.

**C**. If our AI model struggles with detecting a violinist’s bowing motion, how would you debug

and improve its accuracy? Would you adjust the dataset, change the model architecture,

adjust data augmentation, retrain the model, fine-tune hyperparameters, or use another

method?

**Answer**:

##### Adjust Data Augmentation

* For detection robustness:
  + Add motion blur (simulates fast bowing).
  + Random occlusions (e.g., fake "music stand" patches).
  + Various lighting conditions can be added
* For temporal consistency:
  + Use frame stacking (feed 3-5 consecutive frames to the model).

##### Model Level Fixes

* If detection is the issue, switching to a hybrid model might help:
  + Use YOLOv8 (fast) to detect the bow + ViTPose for wrist/elbow tracking.
  + Use the Mediapipe model to get the skeleton and then another model for wrist and elbow tracking. Fine-tune on cropped bow/wrist regions.
* If motion classification is the issue:
  + Use optical flow (e.g., Farneback) as an additional input channel.
  + Training or fine tuning the model with loss function to smooth out the jitters

##### Post-Processing Heuristics

* Enforce physical constraints: Bowing motion is mostly linear (left-right in camera space). Reject erratic predictions. Or try to post processes output with Kalman filter to get smooth predictions.

ng the violin)

**H**. What strategies would you use to keep FPS high while running real-time AI in a mobile

game?

• Would you optimize model size? Reduce inference time? Cache results?

**Answer**:

* Optimize the model: various pruning strategies and quantization of the models can be explored
* Inference time can be reduced by skipping frames, this helps to reduce the number of frames to be processed
* Caching results: Caching strategies in terms of storing the state of the previous frames can be useful

**F.** Which AI architectures would you choose for real-time motion tracking on mobile?

• Would you prefer CNN-based models (e.g., OpenPose, MediaPipe) or

Transformer-based models (e.g., ViTPose)?

• How would you optimize inference speed while maintaining accuracy?

Answer:

* MediaPipe BlazePose (CNN-based) runs 30+ FPS on mid-tier phones
* To optimize inference speed:
  + Input Resolution: Downscale the images to lower resolution
  + Frame Skipping: Process every 2nd frame + interpolate (saves 50% compute).
  + Quantize and prune the model.