**INTERNSHIP PROGRESS REPORT**

**Project Title:**

Image Size Optimization for Patient Data (Background Removal & Compression)

**Organization:**

Newro Kaaya (under PRS Neurosciences)

**Internship Duration:**

July – September 2025

**Team Members:**

* Ankit Ghosh
* Kironmoy Mukherjee
* Mayukh Tilak

**Supervisor:**

Mr Stafford Michahial

Mr Pratitah Sharan

**WEEK 1**

**Objective**

* Build a foundational understanding of **Artificial Intelligence (AI)** and **Machine Learning (ML)**.
* Get familiar with **TensorFlow**, the main framework we plan to use.
* Establish a common knowledge base for the team before moving into project-specific tasks.

**Activities & Work Done**

1. **Introduction to Core Concepts**
   * Understood the differences between **AI, ML, and Deep Learning**.
   * Discussed the types of ML: **supervised learning, unsupervised learning, and reinforcement learning**.
   * Explored how models are trained using **datasets**, and the importance of **training vs testing sets**.
2. **Understanding TensorFlow**
   * Installed and set up TensorFlow in our working environment.
   * Learned about **tensors** as the basic data structure in TensorFlow.
   * Studied the role of **layers, models, optimizers, and loss functions** in building ML models.
3. **Team Learning Sessions**
   * Each member explored different tutorials and documentation.
   * Shared insights within the group to ensure everyone had a clear grasp of the basics.
   * Discussed simple examples like **linear regression** and **classification** to see how models learn patterns.

**WEEK 2**

**Objective**

* To take our **first practical step** into applying the AI/ML basics learned in Week 1.
* Try out simple **image classification** and **face detection** tasks, not for perfect results but to **understand the end-to-end process** of building a model.
* Get familiar with the workflow: **dataset → preprocessing → model → training → testing → saving**.

**Activities & Work Done**

**1. Image Classification (Happy/Sad Classifier)**

* We started with a **basic CNN model** to classify images into two categories (happy vs sad).
* The main goal was to see how images are loaded, preprocessed (resizing, scaling), and fed into a deep learning pipeline.
* We experimented with a simple **Convolutional Neural Network (CNN)** to learn the idea of feature extraction through convolution and pooling layers.
* Training was done for multiple epochs, and we monitored how **loss decreases** and **accuracy improves**.
* The model was then tested on some new images and saved for reuse.

👉 This exercise helped us understand how a **classification problem** is structured in practice.

**2. Face Detection (First Attempt)**

* We then attempted a **face detection model**, which was more complex since it required not just classifying but also **locating faces with bounding boxes**.
* We collected images and annotated them using **MakeSense.ai**, exported in **YOLO format**.
* Learned how to **organize datasets** into training, validation, and test sets.
* Tried **data augmentation** (flipping, rotation, scaling) to increase dataset variety.
* Built a model using **VGG16 as a backbone** and customized it to give both classification and bounding box outputs.
* Defined separate loss functions for classification and localization — a new concept for us at this stage.
* Ran training and did some initial testing, even trying **real-time webcam input** to see it work.

👉 This gave us our first experience of how **object detection tasks** differ from simple classification.

**Findings & Challenges**

* We realized this was our **first “dip” into real ML model building**, and much of the time went into just understanding **how the pieces fit together**.
* Handling datasets and annotations was trickier than expected — especially making sure augmented images and labels matched correctly.
* Performance wasn’t our focus yet; the **main takeaway** was getting comfortable with the **model development pipeline**.

**Next Steps**

* Use the learnings from this week to move into more refined experiments.
* Start exploring **Mediapipe** as a tool for face detection and compare its workflow with our custom models.
* Look into efficiency and accuracy improvements once we’re confident with the process.

Link to work:-   
<https://github.com/mukhokironmoy/Tensorflow-Face-Detection-v1.git>  
https://github.com/mukhokironmoy/Deep-CNN-Image-Classifier.git

**WEEK 3**

**Objective**

* Understand how **Mediapipe** works for human body tracking and face detection.
* Learn how it **maps landmark points** to different parts of the human body and how this data can be used for downstream ML tasks.
* Explore potential applications such as **pose estimation, face detection/recognition, and background removal**.

**Activities & Work Done**

1. **Team Learning (Basics of Mediapipe)**
   * Watched the Mediapipe tutorial video ([YouTube link](https://youtu.be/01sAkU_NvOY?feature=shared)) which explained how to:
     + Load Mediapipe’s built-in **Pose solution**.
     + Detect **33 key landmarks** on the human body (like shoulders, elbows, knees).
     + Retrieve landmark coordinates (x, y, z, visibility) as structured data.
     + Overlay these landmarks on a live video stream using OpenCV .
   * Practiced running simple examples to understand how Mediapipe captures real-time movement.
   * Focused on the idea that these landmarks can later be used as **inputs for training models** or for performing tasks like gesture recognition.
2. **Independent Exploration by Team Member**
   * One teammate (with prior AIML experience) went beyond the basics:
     + Experimented with **Mediapipe’s Face Mesh** for detecting 468 facial landmarks.
     + Attempted an early version of **face recognition** by comparing facial landmark embeddings.
     + Began testing **background removal** using Mediapipe’s segmentation capabilities.
3. **Hands-on Exercises (Group)**
   * Extracted pose landmark data from images and video.
   * Learned how Mediapipe outputs landmark data as structured lists (with normalized coordinates).
   * Discussed how this numerical data could be stored in a dataset and later used for ML tasks such as **action recognition** or **movement classification**.

**Findings & Challenges**

* Realized that **Mediapipe simplifies complex CV tasks**, since pose and face detection models are pre-trained and optimized for real-time use.
* Learned how landmark data is accessible as arrays, which makes it easier to integrate into training pipelines.
* Main challenge: balancing **depth vs breadth** — while two members focused on understanding basics, the third member’s deeper dive into face recognition/background removal highlighted the variety of possible directions.

**Next Steps**

* As a group, consolidate our understanding of Mediapipe’s **pose and face modules**.
* Continue experimenting with background removal and compare Mediapipe’s built-in segmentation with other methods (custom CNNs, MODNet).
* Decide on a **consistent workflow** for the team so we progress in sync while still encouraging individual exploration.

Link to work:  
<https://github.com/mukhokironmoy/Mediapipe-basics.git>

**WEEK 4**

**Objective**

* Begin our first structured attempts at **removing backgrounds from images and video frames**.
* Explore **multiple approaches** using Mediapipe for segmentation and pose detection.
* Test how background removal affects **file size reduction** and **dataset usefulness**.

**Activities & Work Done**

We experimented with several methods. For each, we noted the workflow, tested outputs, and saved code/scripts for reference.

**Method 1 – Pose Cropping (Bounding Box around Person)**

* **Idea:** Use Mediapipe Pose to detect human landmarks, compute a bounding box, and crop only the region containing the subject.
* **Process:**
  + Detect visible body keypoints.
  + Calculate minimum and maximum x,y coordinates to form a bounding box.
  + Add a fixed margin (converted from cm to pixels).
  + Save cropped frames as JPEG with compression.
* **Expected Advantage:** Reduces storage by cutting out background pixels entirely.

📂 **Code Link:** pose\_cropping.py  
🖼️ **Screenshots:** *[to be added]*  
📝 **Remarks:** *[space reserved]*

**Method 2 – Pose Erasing with Alpha Matte**

* **Idea:** Create a **mask of body landmarks and connections**, then erase/remove the background by applying an alpha channel.
* **Process:**
  + Build a mask by filling circles around each joint and connecting bones.
  + Smooth and dilate the mask to create a continuous silhouette.
  + Merge mask with original frame to produce RGBA images with transparent backgrounds.
* **Expected Advantage:** Allows saving **PNG with transparency** for flexibility in downstream ML tasks.

📂 **Code Link:** pose\_erase.py  
🖼️ **Screenshots:** *[to be added]*  
📝 **Remarks:** *[space reserved]*

**Method 3 – Real-time Segmentation and Cropping**

* **Idea:** Use **Mediapipe Selfie Segmentation** in live webcam feed to isolate person from background.
* **Process:**
  + Generate segmentation mask in real time.
  + Apply binary threshold to extract person.
  + Crop around the detected region with margin.
  + Save output frames as JPEG with compression.
* **Expected Advantage:** Useful for **real-time applications** and efficient dataset collection.

📂 **Code Link:** realtime.py  
🖼️ **Screenshots:** *[to be added]*  
📝 **Remarks:** *[space reserved]*

**Method 4 – Batch Segmentation and Compression**

* **Idea:** Process entire folders of saved frames with segmentation + JPEG compression.
* **Process:**
  + Loop through stored images.
  + Apply Mediapipe segmentation mask for background removal.
  + Crop subject and save with **JPEG optimization (quality 95 → 65 → 55)**.
  + Combines both background removal and lossy compression for maximum file size reduction.
* **Expected Advantage:** Scalable for **large datasets**; integrates background removal + compression in one pipeline.

📂 **Code Link:** segment\_and\_compress.py, imagecompression.ipynb  
🖼️ **Screenshots:** *[to be added]*  
📝 **Remarks:** *[space reserved]*

**Method 5 – Combined Background Removal + Compression Pipeline**

* **Idea:** Create a single pipeline that performs both background removal and multi-stage JPEG compression for maximum file size reduction.
* **Process:**
  + Step 1: Apply an initial compression (simulating Guetzli-like optimization).
  + Step 2: Run Mediapipe Pose with segmentation enabled to remove background.
  + Step 3: If segmentation fails, use a convex hull fallback based on visible landmarks to approximate the body region.
  + Step 4: Save the processed output as JPEG.
  + Step 5: Apply a second, stronger compression step to further reduce file size.
* **Expected Advantage:**
  + Produces significant space savings while still keeping the subject pixels.
  + Can be used as a batch processor for entire datasets.
  + Flexible fallback ensures at least partial results even when segmentation is imperfect.

**📂 Code Link: Method 5 - imagecompression.py  
🖼️ Screenshots: *[to be added]*  
📝 Remarks: *[space reserved]***

**Findings & Challenges**

* **Pose Cropping** worked well for quick storage reduction, but cropping alone does not eliminate complex backgrounds.
* **Alpha Matte Erasing** produced cleaner outputs but PNGs were larger in size compared to JPEG.
* **Real-time Segmentation** was efficient but requires tuning confidence thresholds to avoid false negatives.
* **Batch Compression** gave the best balance between **automation and size reduction**, but requires further testing on large datasets.

**Next Steps**

* Compare results of each method quantitatively (file size, quality retention, ML usability).
* Decide on a **primary workflow** for dataset processing.
* Explore whether advanced models like **MODNet** provide better background removal than Mediapipe.

**WEEK 8**

**Objective**

* Reduce storage size of Poise session frames (~10 KB JPEGs at 30 fps).
* Preserve all clinically relevant body parts (hands, arms, legs) while removing unnecessary background.
* Explore and benchmark different background removal and compression methods.
* Move towards a stable, automated pipeline for large-scale processing.

**Activities & Work Done**

1. **Initial Experiments with File Formats**
   * Converted Poise JPEG frames to **WebP**.
     + Finding: On our low-detail frames, WebP **increased file size** instead of reducing.
     + Decision: WebP not viable as a direct replacement for JPEG.
2. **Background Removal with Mediapipe**
   * Tried **Mediapipe Selfie Segmentation** for extracting subjects.
     + Advantage: lightweight and real-time capable.
     + Issues:
       - Low-quality (10 KB) images → masks were inaccurate.
       - Cropping based on masks often cut away limbs or key body parts.
   * Outcome: Not reliable for production but useful for quick tests.
3. **Cropping Approaches Tested**
   * Naïve cropping from segmentation masks.
     + Issue: **unstable bounding boxes** and high risk of cutting important regions.
   * Lesson: **Landmark-aware cropping** (using Poise-provided coordinates) is needed instead of mask-only cropping.

**Proposed Workflow (Pipeline)**

1. **Ingest**
   * Input: Frame folder (JPEGs) + Poise landmark data (coordinates of body joints).
   * Validation: Check missing landmarks, frame consistency.
2. **Background Removal**
   * **Primary:** Use **MODNet** for alpha matte → better isolation of subject compared to Mediapipe.
   * **Fallback:** Mediapipe (with denoise + relaxed thresholds) if MODNet unavailable.
3. **Landmark-Aware Cropping**
   * Compute bounding box from Poise landmarks (shoulders, hips, wrists, ankles).
   * Add configurable margins to ensure no limb truncation.
   * Apply temporal smoothing to reduce jitter between consecutive frames.
4. **Compression**
   * Run cropped frames through **CLI optimizers** (jpegoptim / mozjpeg).
   * Tune JPEG quality (55–70) to balance size vs clarity.
   * WebP only for cases with very simple crops (optional).
5. **Reporting & QA**
   * Track size reduction per frame and per session.
   * Verify all landmarks remain inside the cropped region.
   * Create side-by-side samples for quick human review.

**Findings & Challenges**

* **WebP** is not automatically better — in our data, it performed worse than JPEG.
* **Mediapipe** segmentation struggles with small low-quality frames, producing unstable masks.
* **Naïve Cropping** caused frequent data loss; landmark-guided cropping is essential.
* Key challenge: Balancing **accuracy (preserve subject)** with **compression (reduce size)**.

**Next Steps**

* Implement MODNet on sample sessions to test matte quality.
* Develop **landmark-aware cropping logic** with margins + smoothing.
* Benchmark compression settings to find best quality/size tradeoff.
* Automate reporting pipeline for session-level results.
* Run pilot tests on multiple sessions to validate both storage savings and clinical usability.