```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=Tr
```

```
import pandas as pd
import os
import re

# ------------------------------
# Paths
# ------------------------------
folder_path = "/content/drive/MyDrive/DV project /Simplified"
output_base = "/content/drive/MyDrive/DV project /Simplified/Separated_Files_By_PositionSeparated_Files_By_Position"

folders = ["standing", "sitting", "wheelchair", "other"]
for f in folders:
    os.makedirs(os.path.join(output_base, f), exist_ok=True)

# Dictionary for CSV
position_dict = {f: [] for f in folders}

# ------------------------------
# Process files (NO COPYING)
# ------------------------------
for file in os.listdir(folder_path):
    if not file.endswith(".txt"):
        continue

    match = re.match(r"(\d+)_(\d+)_(\d+)_(\d+)_(\d+)_(\w+)\.txt", file)

    if not match:
        position = "other"
    else:
        pos = match.group(6).lower()
        if pos in ["stand", "standing"]:
            position = "standing"
        elif pos in ["sit", "sitting"]:
            position = "sitting"
        elif pos in ["wheel", "wheelchair", "chair"]:
            position = "wheelchair"
        else:
            position = "other"

    # ✅ ONLY record filename (no file operations)
    position_dict[position].append(file)

# ------------------------------
# Create / Update CSVs only
# ------------------------------
for pos in position_dict:
    csv_path = os.path.join(output_base, pos, f"{pos}_filenames.csv")
    df = pd.DataFrame(position_dict[pos], columns=["Filename"])
    df.to_csv(csv_path, index=False)
    print(f"📄 CSV refreshed: {csv_path}")

print("\n✅ Safe run completed — no files moved or copied.")
```

```
📄 CSV refreshed: /content/drive/MyDrive/DV project /Simplified/Separated_Files_By_PositionSeparated_Files_By_Position/stanc
📄 CSV refreshed: /content/drive/MyDrive/DV project /Simplified/Separated_Files_By_PositionSeparated_Files_By_Position/sitti
📄 CSV refreshed: /content/drive/MyDrive/DV project /Simplified/Separated_Files_By_PositionSeparated_Files_By_Position/wheel
📄 CSV refreshed: /content/drive/MyDrive/DV project /Simplified/Separated_Files_By_PositionSeparated_Files_By_Position/other

✅ Safe run completed — no files moved or copied.
```

This code organizes all skeleton .txt files into separate folders based on the body position found in the filename. It performs the following steps:

Reads all .txt files from the input directory

Extracts metadata from filenames using regular expressions

Detects the body position (standing, sitting, wheelchair, or other)

Copies each file into its corresponding folder

Stores filenames in a dictionary for each category

Generates four CSV files listing filenames for each class

Automatically handles invalid or unknown filename formats

Prints completion messages for user confirmation

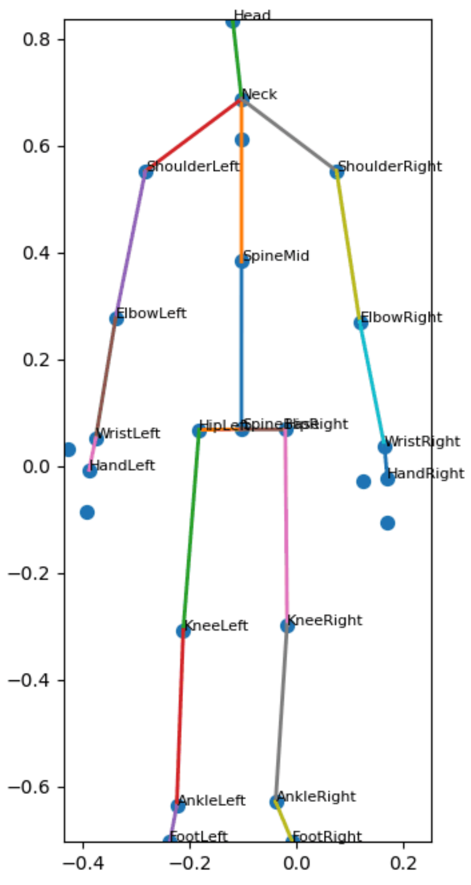
This helps in cleaning and structuring the dataset for further analysis or machine learning tasks.

```
import os

standing_folder = "/content/drive/MyDrive/DV project /Simplified/Separated_Files_By_PositionSeparated_Files_By_Position/sta

# List all TXT files
txt_files = [f for f in os.listdir(standing_folder) if f.endswith(".txt")]

# Count
print("Total TXT files in standing folder:", len(txt_files))
```

```
Total TXT files in standing folder: 0
```

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation, PillowWriter
from IPython.display import Image, display

def load_skeleton(file_path):
    data = np.loadtxt(file_path, delimiter=',')
    n_joints = data.shape[1] // 3
    return data.reshape((-1, n_joints, 3))

joint_labels = [
    "SpineBase", "SpineMid", "Neck", "Head",
    "ShoulderLeft", "ElbowLeft", "WristLeft", "HandLeft",
    "ShoulderRight", "ElbowRight", "WristRight", "HandRight",
    "HipLeft", "KneeLeft", "AnkleLeft", "FootLeft",
    "HipRight", "KneeRight", "AnkleRight", "FootRight"
]

connections = [
    (0,1),(1,2),(2,3),
    (2,4),(4,5),(5,6),(6,7),
    (2,8),(8,9),(9,10),(10,11),
    (0,12),(12,13),(13,14),(14,15),
    (0,16),(16,17),(17,18),(18,19)
]

def animate_skeleton_2d_labeled(file_path, interval=50, save_path='skeleton_2d_fixed.gif'):
    data = load_skeleton(file_path)
    n_frames, n_joints, _ = data.shape

    x_min, x_max = np.min(data[:,:,0]), np.max(data[:,:,0])
    y_min, y_max = np.min(data[:,:,1]), np.max(data[:,:,1])

    fig, ax = plt.subplots(figsize=(6,8))
    ax.set_xlim(x_min, x_max)
    ax.set_ylim(y_min, y_max)   # ✅ NO inversion
    ax.set_aspect('equal')

    scatter = ax.scatter([], [], s=50)
    text_labels = [ax.text(0,0,"", fontsize=8) for _ in range(n_joints)]
    lines = [ax.plot([], [], lw=2)[0] for _ in connections]

    def update(frame):
        scatter.set_offsets(data[frame,:,:2])
        for i, label in enumerate(joint_labels):
            text_labels[i].set_position((data[frame,i,0], data[frame,i,1]))
            text_labels[i].set_text(label)
        for k, (i,j) in enumerate(connections):
            lines[k].set_data(
                [data[frame,i,0], data[frame,j,0]],
                [data[frame,i,1], data[frame,j,1]]
            )
        return scatter, *lines, *text_labels

    ani = FuncAnimation(fig, update, frames=n_frames, interval=interval)
    ani.save(save_path, writer=PillowWriter(fps=10))
    plt.close(fig)
    display(Image(filename=save_path))

file_path = '/content/drive/MyDrive/DV project /Simplified/Separated_Files_By_Position/standing/101_18_0_1_1_stand.txt'
```

```
animate_skeleton_2d_labeled(file_path)
```



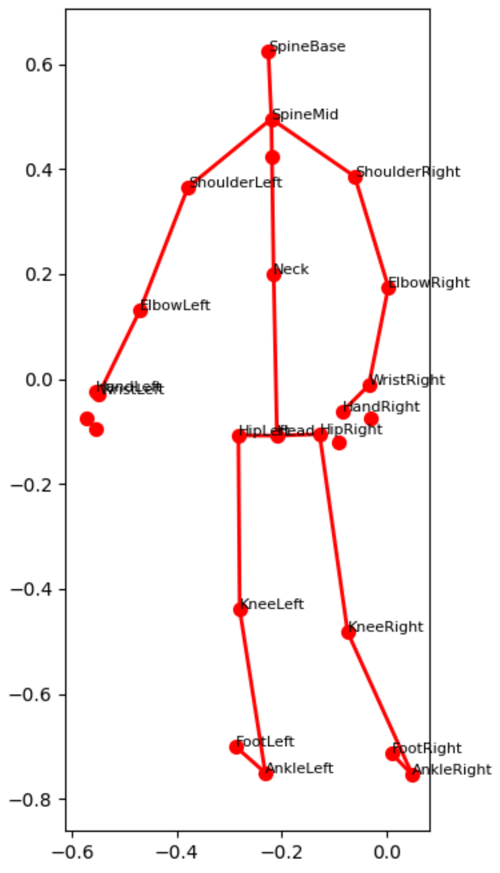
```
import os
import pandas as pd

# Path to standing folder
folder_path = '/content/drive/MyDrive/DV project /Simplified/Separated_Files_By_Position/standing'

# List to store metadata
data_list = []

# Loop through all .txt files
for file in os.listdir(folder_path):
    if file.endswith('.txt'):
        # Split filename based on underscores
        # Example filename: 101_18_0_1_1_stand.txt
        name_part = file.replace('.txt', '').split('_')
        if len(name_part) == 6:
            subject_id, date_id, gesture_label, repetition_number, correct_label, position = name_part
            data_list.append({
                "SubjectID": subject_id,
                "DateID": date_id,
                "GestureLabel": gesture_label,
                "RepetitionNumber": repetition_number,
                "CorrectLabel": correct_label,
                "Position": position,
                "Filename": file
            })

# Create DataFrame
df = pd.DataFrame(data_list)

# Check if DataFrame is empty before attempting to reorder columns or save
if not df.empty:
    # Optional: reorder columns
    df = df[["SubjectID", "DateID", "GestureLabel", "RepetitionNumber", "CorrectLabel", "Position", "Filename"]]
    # Save to CSV
    csv_path = '/content/drive/MyDrive/DV project /Simplified/Separated_Files_By_PositionCombinedCSV/standing_metadata.csv
    os.makedirs(os.path.dirname(csv_path), exist_ok=True) # Ensure output directory exists
    df.to_csv(csv_path, index=False)

    print(f"CSV saved with {len(df)} records at {csv_path}")
else:
    print(f"No .txt files found in {folder_path}. No CSV generated.")
```

```
CSV saved with 1222 records at /content/drive/MyDrive/DV project /Simplified/Separated_Files_By_PositionCombinedCSV/standing
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from IPython.display import HTML, display

# Increase animation embed limit to accommodate larger animations
plt.rcParams['animation.embed_limit'] = 50.0 # Set to 50 MB, adjust as needed

# Load metadata
csv_path = '/content/drive/MyDrive/DV project /Simplified/Separated_Files_By_PositionCombinedCSV/standing_metadata.csv'
df = pd.read_csv(csv_path)

# Load skeleton data
def load_skeleton(file_path):
    data = np.loadtxt(file_path, delimiter=',')
    n_joints = data.shape[1] // 3
    data = data.reshape((-1, n_joints, 3))
    return data

# Skeleton connections
connections = [
    (0,1),(1,2),(2,3),      # Spine
    (2,4),(4,5),(5,6),(6,7),# Left arm
    (2,8),(8,9),(9,10),(10,11), # Right arm
    (0,12),(12,13),(13,14),(14,15), # Left leg
    (0,16),(16,17),(17,18),(18,19)  # Right leg
]

# Joint labels ordered head-to-foot
joint_labels = [
    "Head", "Neck", "SpineMid", "SpineBase",
    "ShoulderLeft", "ElbowLeft", "WristLeft", "HandLeft",
    "ShoulderRight", "ElbowRight", "WristRight", "HandRight",
    "HipLeft", "KneeLeft", "AnkleLeft", "FootLeft",
    "HipRight", "KneeRight", "AnkleRight", "FootRight"
]

# 2D Animation function (head-to-foot)
def animate_skeleton_2d(file_path, correct=True, interval=100):
    data = load_skeleton(file_path)
    n_frames, n_joints, _ = data.shape

    x_min, x_max = np.min(data[:,:,0]), np.max(data[:,:,0])
    y_min, y_max = np.min(data[:,:,1]), np.max(data[:,:,1])

    fig, ax = plt.subplots(figsize=(6,8))
    ax.set_xlim(x_min - 0.05*(x_max-x_min), x_max + 0.05*(x_max-x_min))
    ax.set_ylim(y_min - 0.05*(y_max-y_min), y_max + 0.05*(y_max-y_min))
    ax.set_aspect('equal')  # Do NOT invert y-axis for head-to-foot

    color = 'green' if correct else 'red'

    scatter = ax.scatter([], [], c=color, s=50)
    text_labels = [ax.text(0,0,"", fontsize=8, color='black') for _ in range(n_joints)]
    lines = [ax.plot([], [], c=color, lw=2)[0] for _ in connections]

    def update(frame):
        scatter.set_offsets(data[frame,:,:2])
        for idx, label in enumerate(joint_labels):
            text_labels[idx].set_position((data[frame,idx,0], data[frame,idx,1]))
            text_labels[idx].set_text(label)
        for idx, (i,j) in enumerate(connections):
            lines[idx].set_data([data[frame,i,0], data[frame,j,0]],
                                [data[frame,i,1], data[frame,j,1]])
        return scatter, *lines, *text_labels

    ani = FuncAnimation(fig, update, frames=n_frames, interval=interval, blit=False)
    plt.close(fig)
    return HTML(ani.to_jshtml())
```

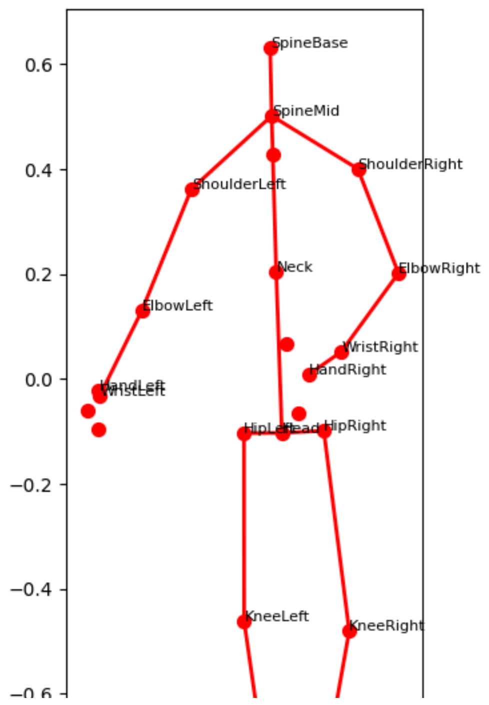
```
# Animate first 3 files for each GestureLabel 0–8
for gesture in range(9):
    gesture_df = df[df['GestureLabel'] == gesture].reset_index(drop=True)
    print(f"\n=== GestureLabel {gesture} ===")
    for idx, row in gesture_df.head(3).iterrows():  # first 3 samples per gesture
        file_path = '/content/drive/MyDrive/DV project /Simplified/Separated_Files_By_Position/standing/' + row['Filename']
        correct = row['CorrectLabel'] == 1
        print(f"Animating {row['Filename']} - {'Correct' if correct else 'Incorrect'}")
        display(animate_skeleton_2d(file_path, correct=correct))
```

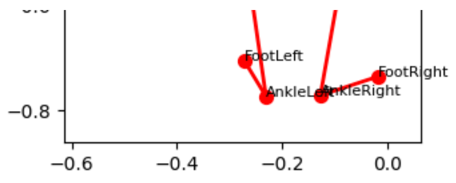
```
=== GestureLabel 0 ===
Animating 211_18_0_1_2_stand.txt - Incorrect
```



Once [✓ Loop] Reflect

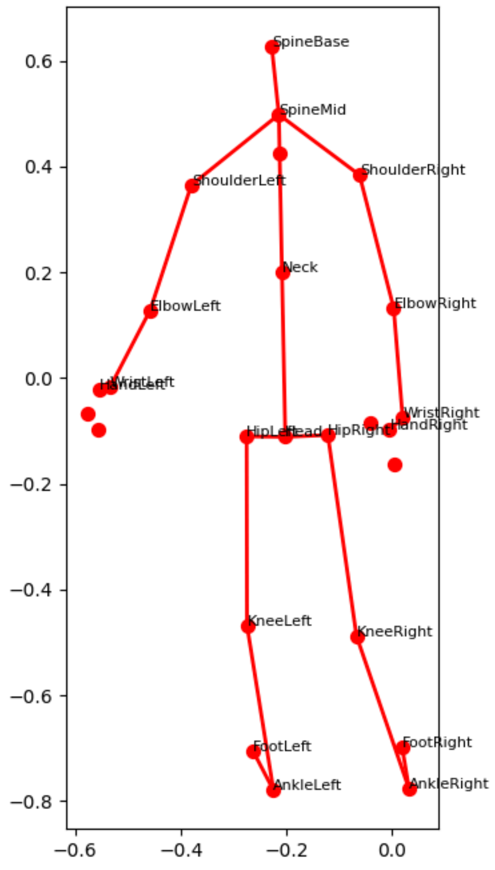
```
Animating 211_18_0_2_2_stand.txt - Incorrect
```





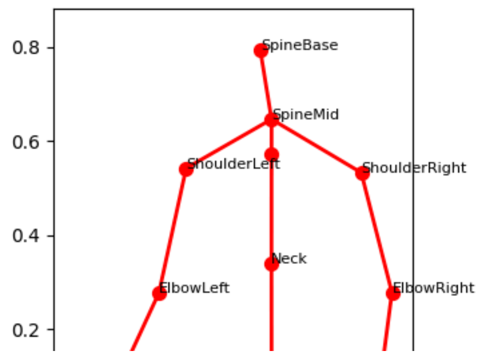
Once [✓ Loop] Reflect

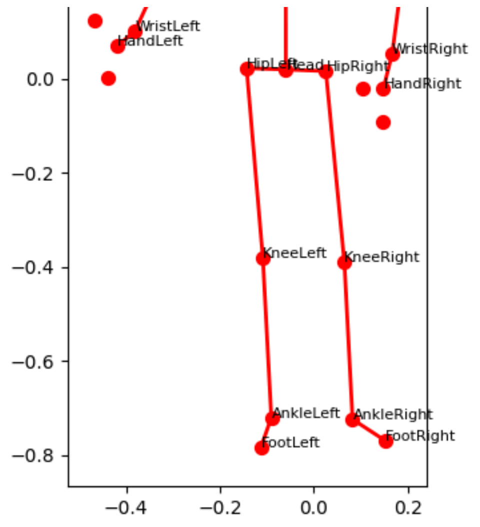
```
Animating 211_18_0_5_2_stand.txt - Incorrect
```



Once [✓ Loop] Reflect

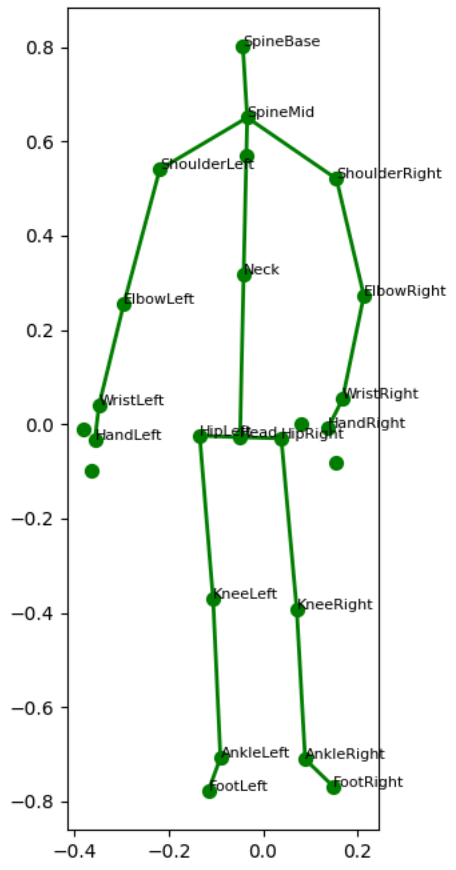
```
=== GestureLabel 1 ===
Animating 207_18_1_15_2_stand.txt - Incorrect
```





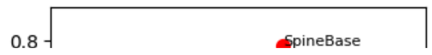
○ Once [✓ Loop] ○ Reflect

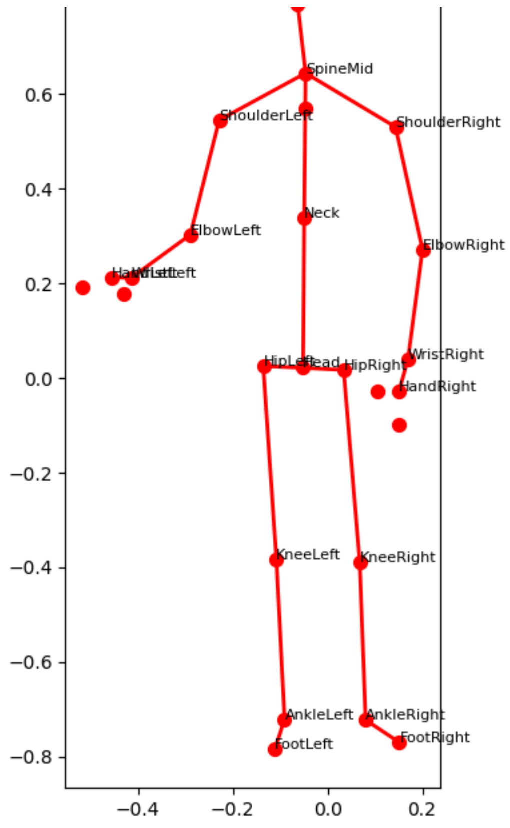
```
Animating 207_18_1_9_1_stand.txt - Correct
```

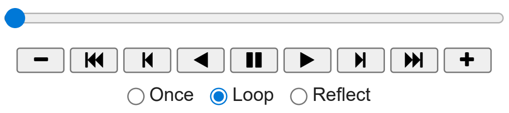


○ Once [✓ Loop] ○ Reflect

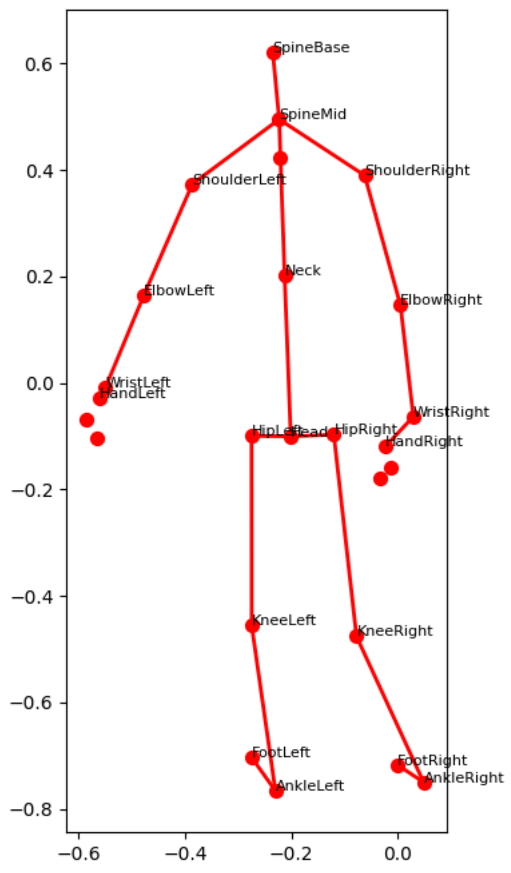
```
Animating 207_18_1_7_2_stand.txt - Incorrect
```





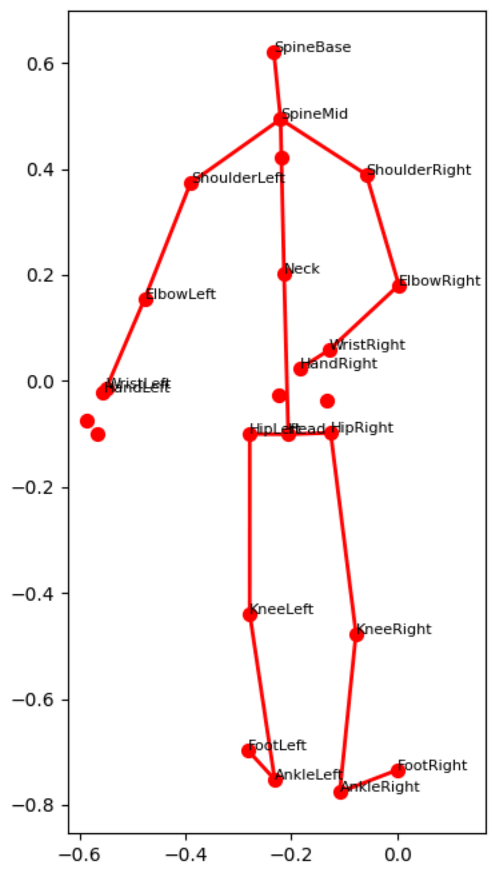


```
=== GestureLabel 2 ===
Animating 211_18_2_1_2_stand.txt - Incorrect
```



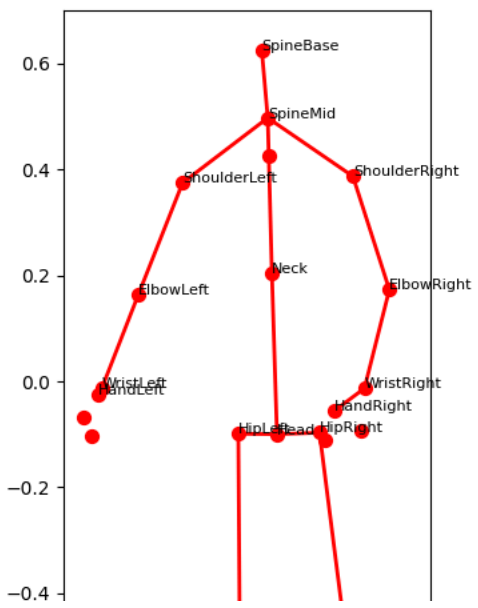
Once [✓ Loop] Reflect

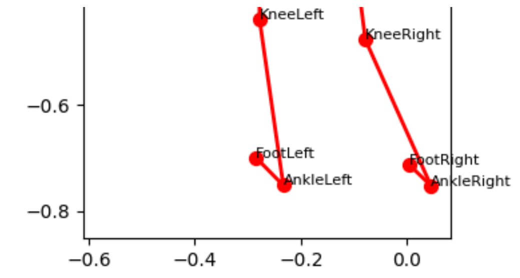
```
Animating 211_18_2_5_2_stand.txt - Incorrect
```



Once [✓ Loop] Reflect

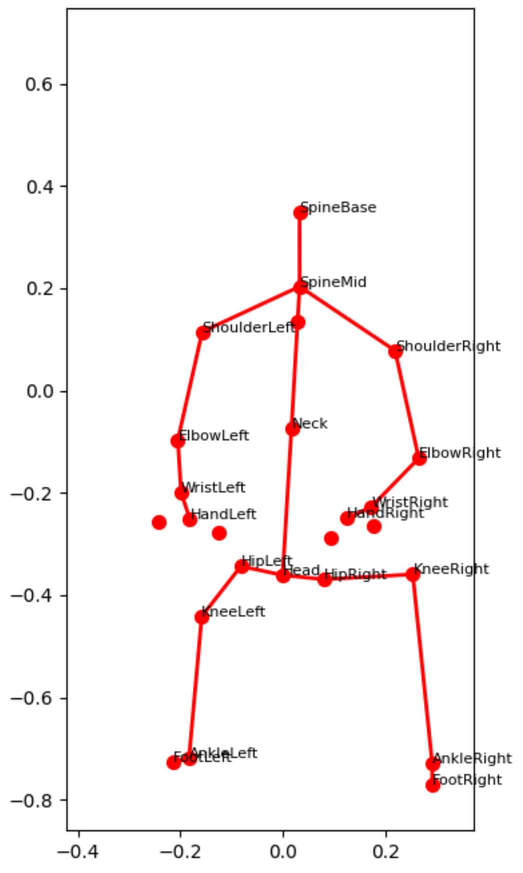
```
Animating 211_18_2_2_2_stand.txt - Incorrect
```





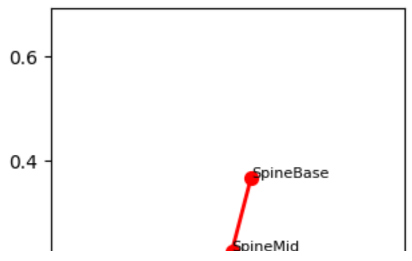
○ Once [✓ Loop] ○ Reflect

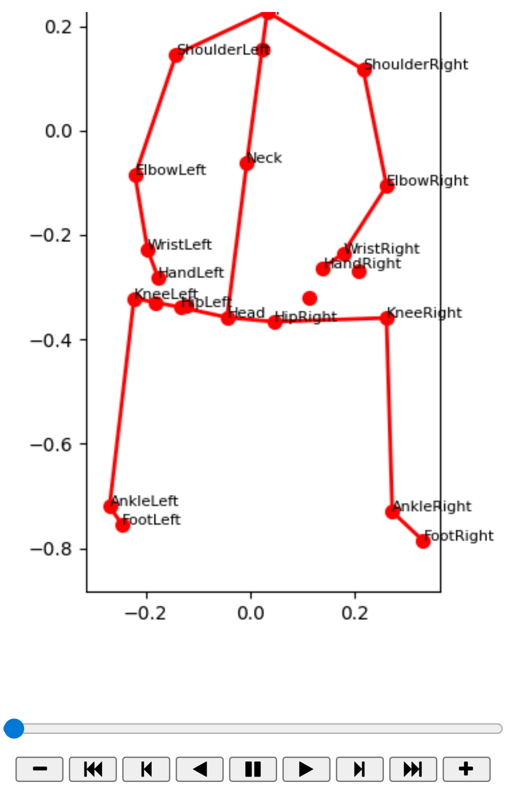
```
=== GestureLabel 3 ===
Animating 207_18_3_9_2_stand.txt - Incorrect
```



○ Once [✓ Loop] ○ Reflect

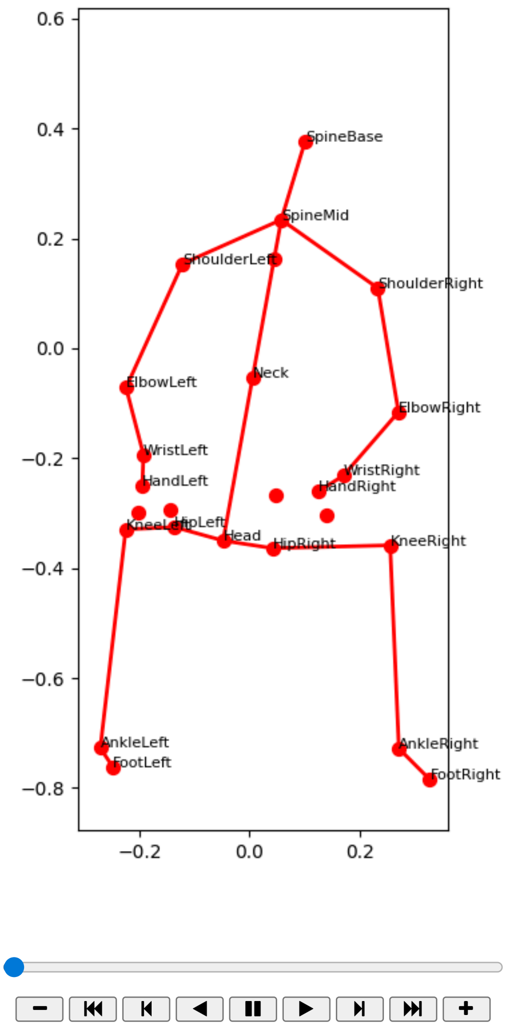
```
Animating 207_18_3_8_2_stand.txt - Incorrect
```





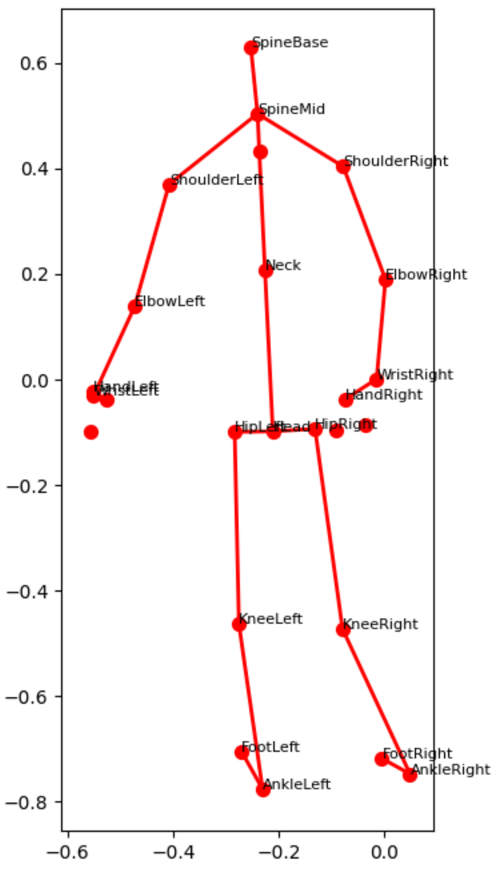
Once [✓ Loop] Reflect

```
Animating 207_18_3_6_2_stand.txt - Incorrect
```



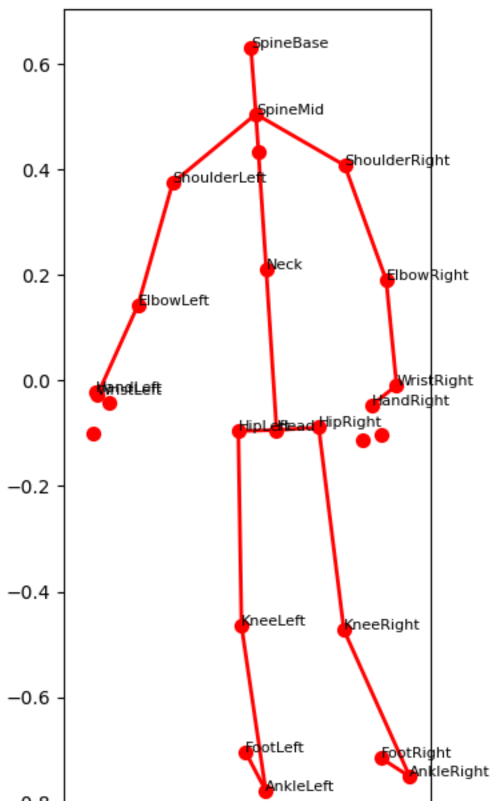
Once [✓ Loop] Reflect

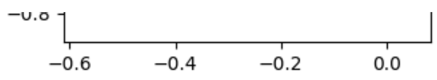
```
=== GestureLabel 4 ===
Animating 211_18_4_2_2_stand.txt - Incorrect
```



Once [✓ Loop] Reflect

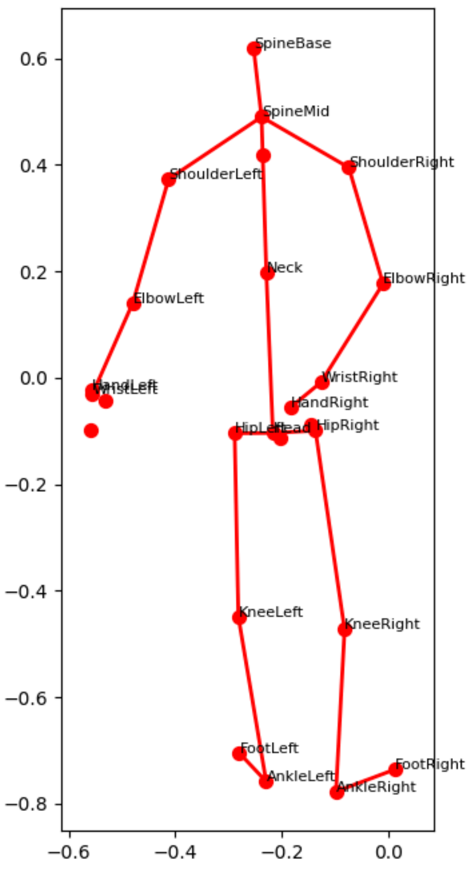
```
Animating 211_18_4_1_2_stand.txt - Incorrect
```





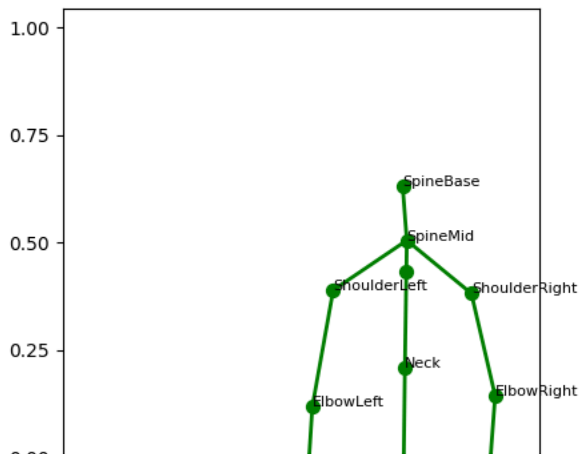
Once [✓ Loop] Reflect

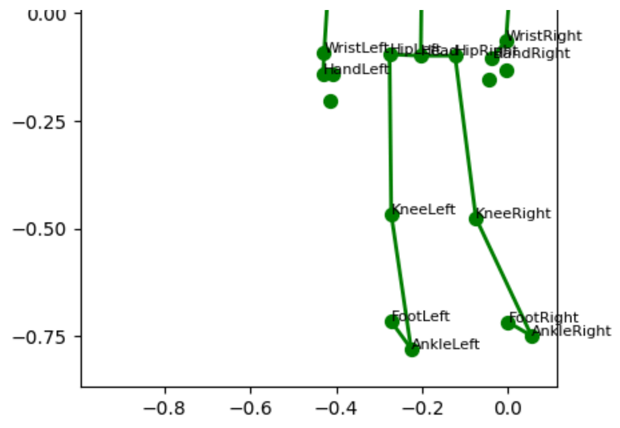
```
Animating 211_18_4_4_2_stand.txt - Incorrect
```



Once [✓ Loop] Reflect

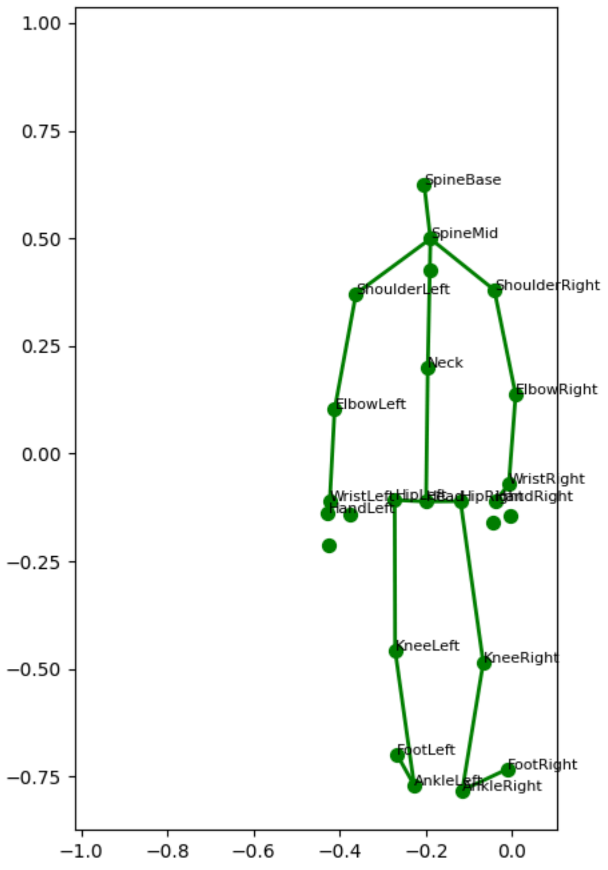
```
=== GestureLabel 5 ===
Animating 211_18_5_2_1_stand.txt - Correct
```





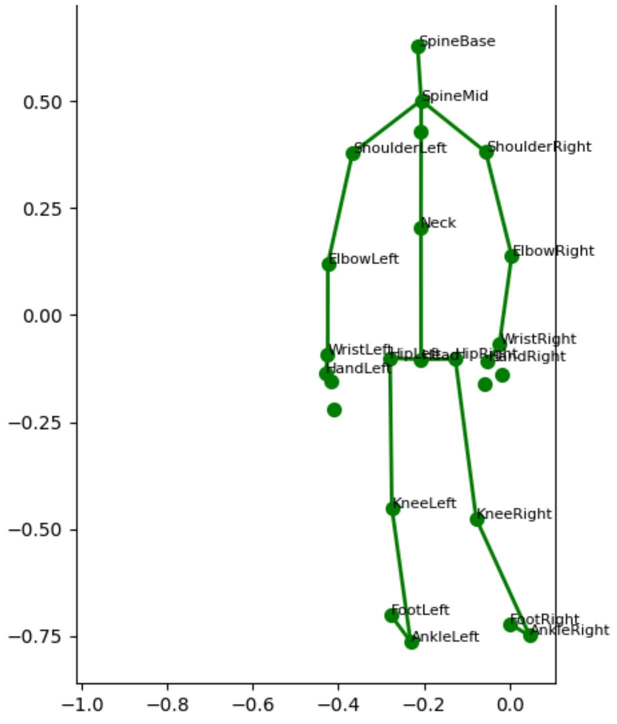
Once [✓ Loop] Reflect

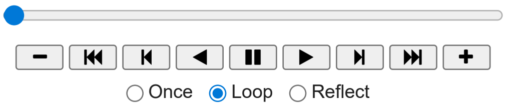
```
Animating 211_18_5_1_1_stand.txt - Correct
```



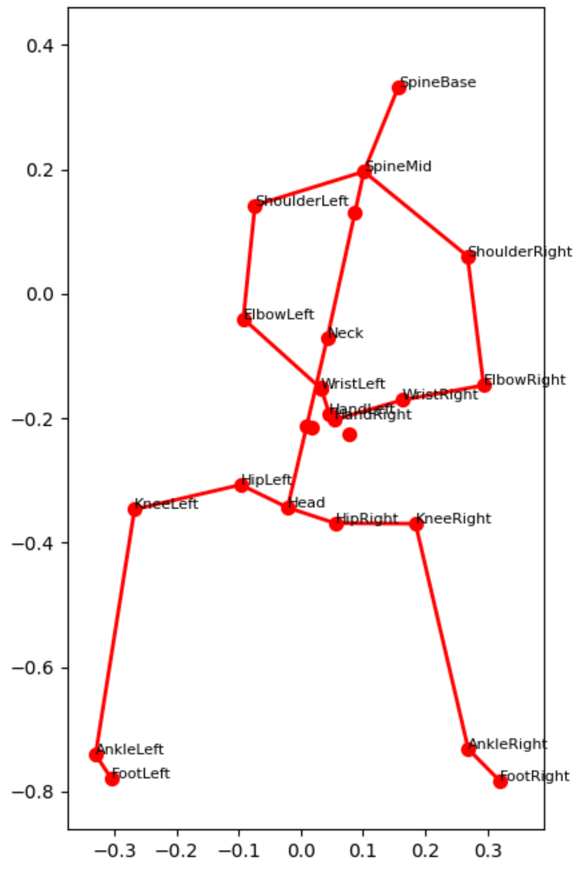
Once [✓ Loop] Reflect

```
Animating 211_18_5_3_1_stand.txt - Correct
```



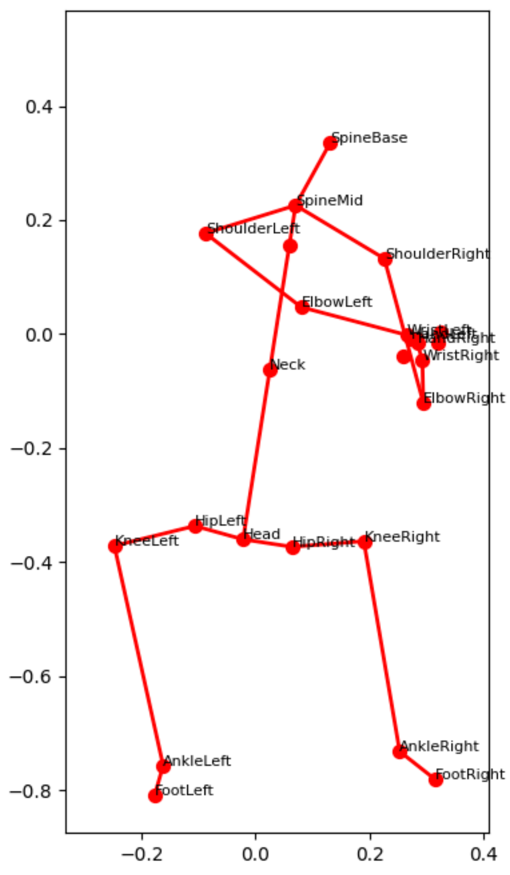


```
=== GestureLabel 6 ===
Animating 207_18_6_15_2_stand.txt - Incorrect
```



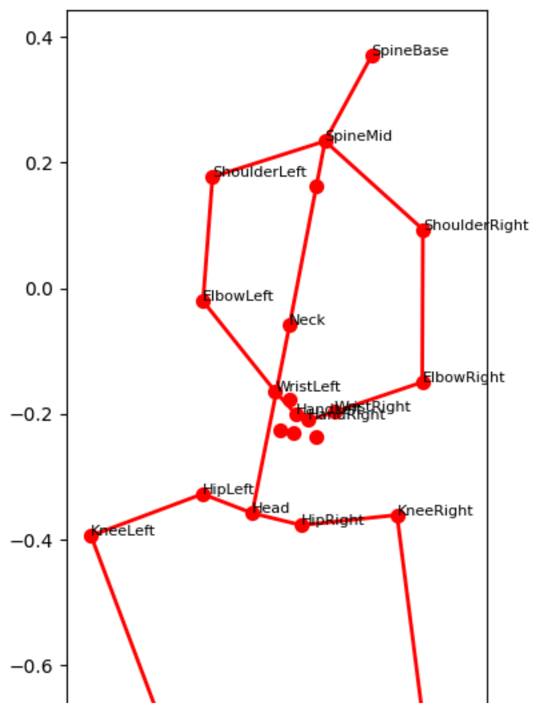
Once [✓ Loop] Reflect

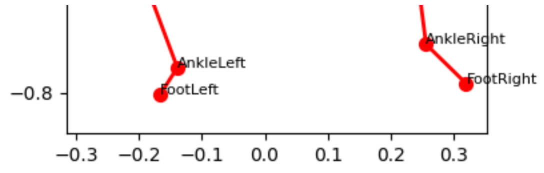
```
Animating 207_18_6_2_2_stand.txt - Incorrect
```

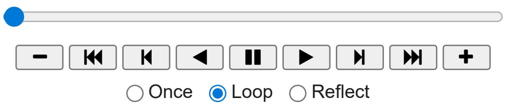


Once [✓ Loop] Reflect

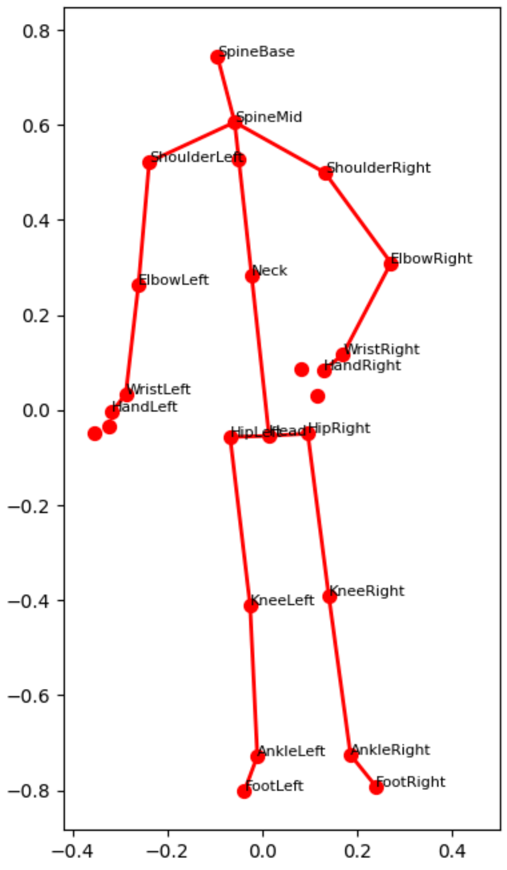
```
Animating 207_18_6_19_2_stand.txt - Incorrect
```

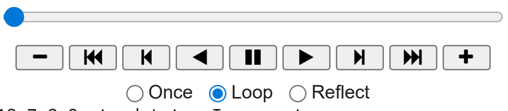




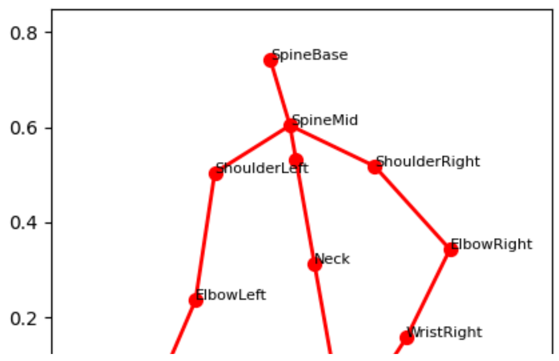


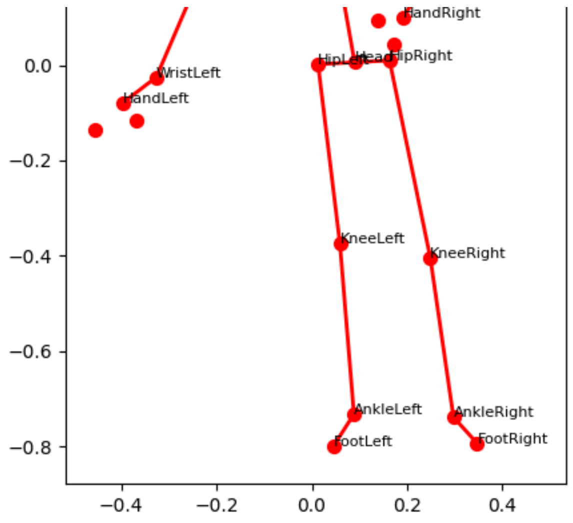
```
=== GestureLabel 7 ===
Animating 207_18_7_2_2_stand.txt - Incorrect
```





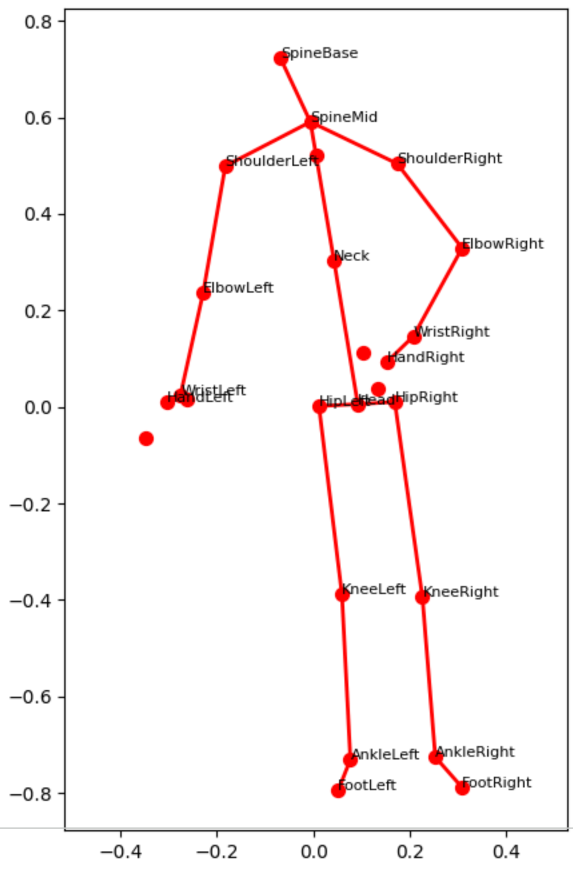
```
Animating 207_18_7_8_2_stand.txt - Incorrect
```





Once [✓ Loop] Reflect

```
Animating 207_18_7_1_2_stand.txt - Incorrect
```



Once [✓ Loop] Reflect

```
=== GestureLabel 8 ===
Animating 207_18_8_2_2_stand.txt - Incorrect
```

0.8
SpineBase