
KineX

WE INTERVIEWED PHYSICAL THERAPY PATIENTS, AS WELL AS PHYSICAL THERAPISTS, TO LEARN ABOUT THE CHALLENGES THEY ENCOUNTERED...

WE CONDUCTED A SURVEY AND CONCLUDED THAT THE MAIN REASONS INDIVIDUALS DID NOT CONTINUE PT WERE:

- COST (60%)
- LACK OF INSURANCE COVERAGE (40%)
- LIMITED TIME (80%)
- TRAVELLING OFTEN (50%)
- FORGETTING EXERCISES (90%)

KineX

CONNECTING PHYSICAL THERAPY WITH AI TO YOUR BUSY LIFE

THEME: PHYSICAL HEALTH AND MENTAL WELL-BEING MANAGEMENT

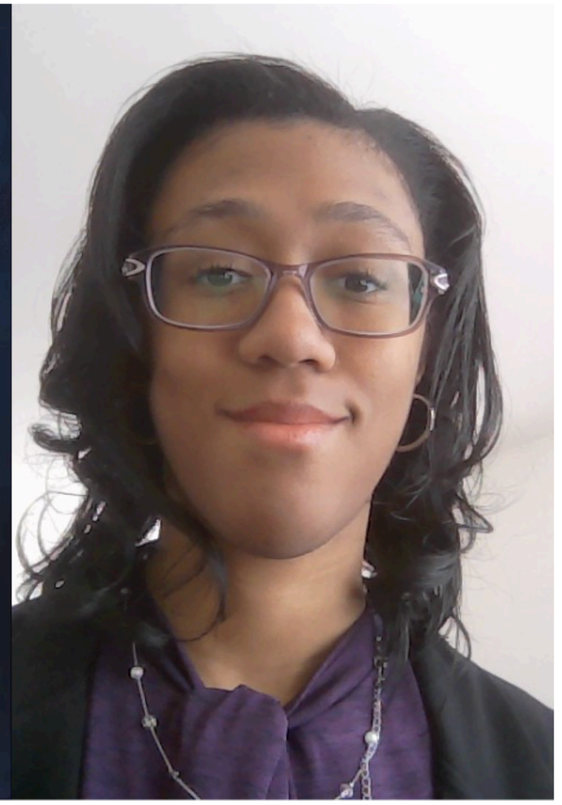
KineX

Team 11

Netra Ramesh

Zenia Haroon

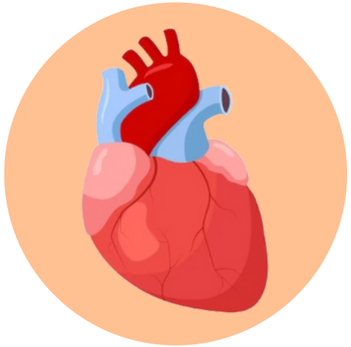
Annika Sachdeva



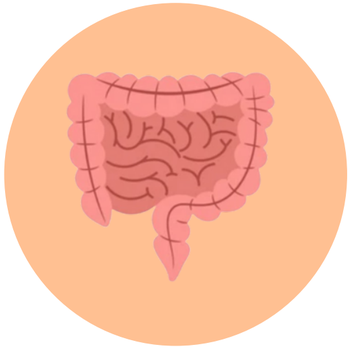
Erin Brunson

Team 11 Mentor

Abhi Hanchate



22%



22%

18%

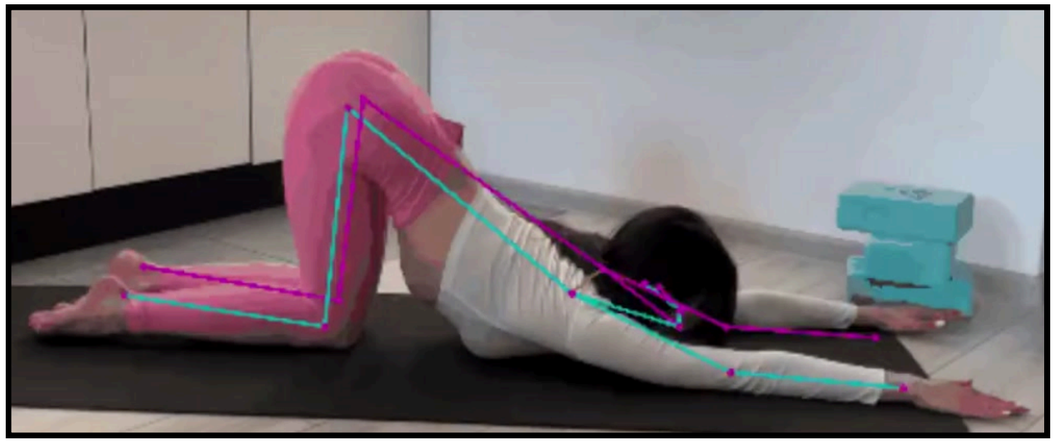
85% of people in the world lead <u>sedentary</u> lifestyles

Our audience:

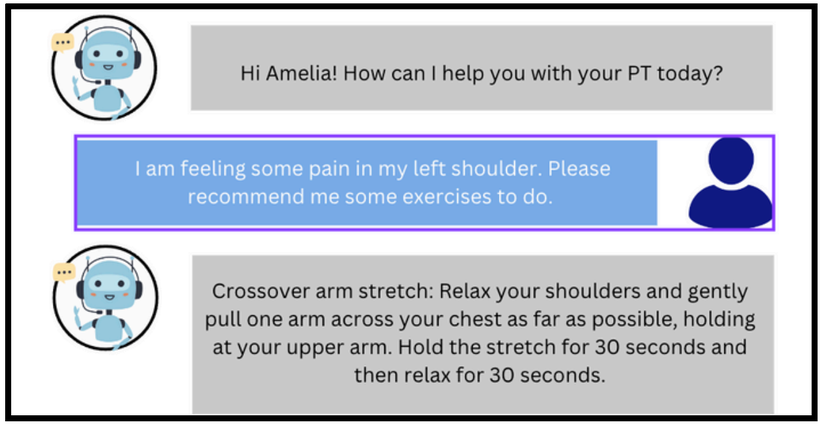
Those who have little time/are living a sedentary lifestyle

There is nothing on the market that combines computer vision with wearable tech and AI recommendations → which makes KineX unique

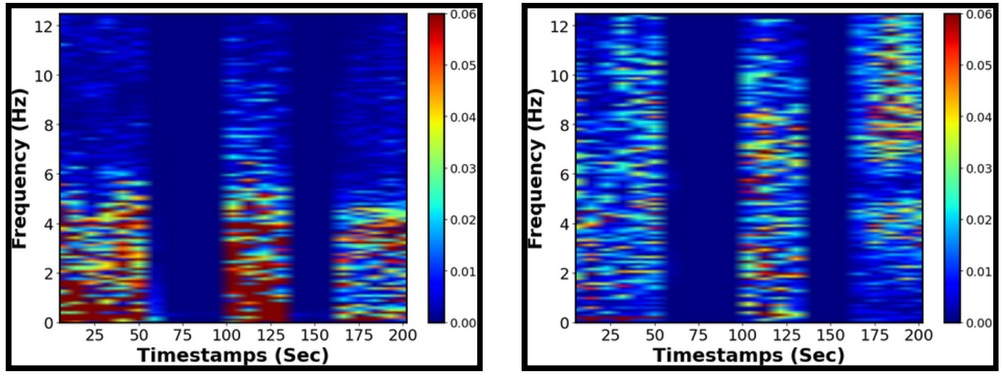
Computer Vision: Pose Estimation



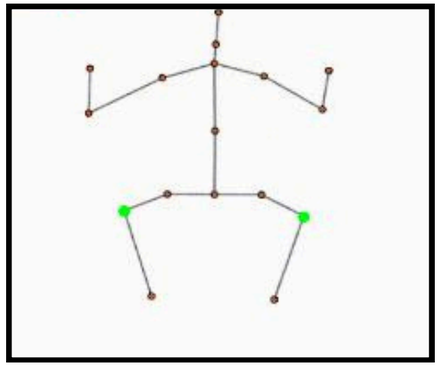
LLM-based Recommendations



Signal Processing: Multimodal Data Analysis



Post Analytics and Visualization



Angle: 59
Left Knee: Good

Angle: 89.5
Right Knee: Good

DEMO

OpenCV

TensorFlow

PyTorch

UC Irvine
Machine Learning
Repository

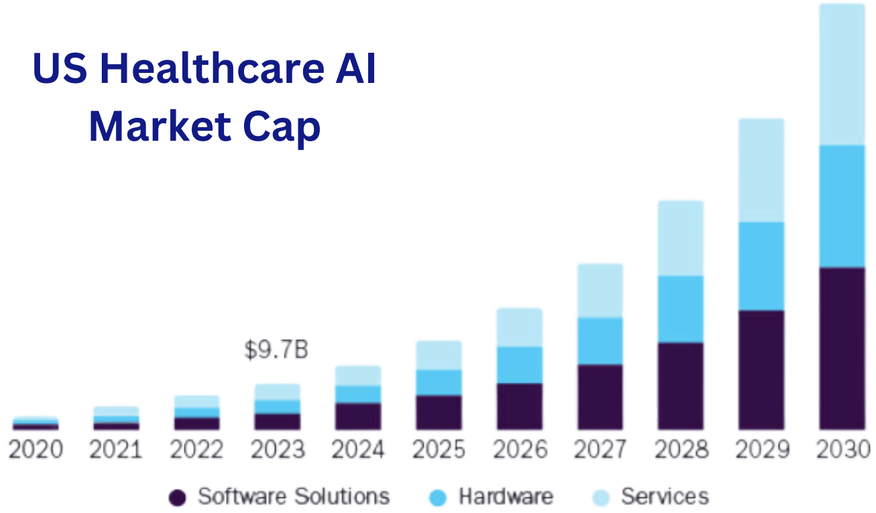
K

Kinex

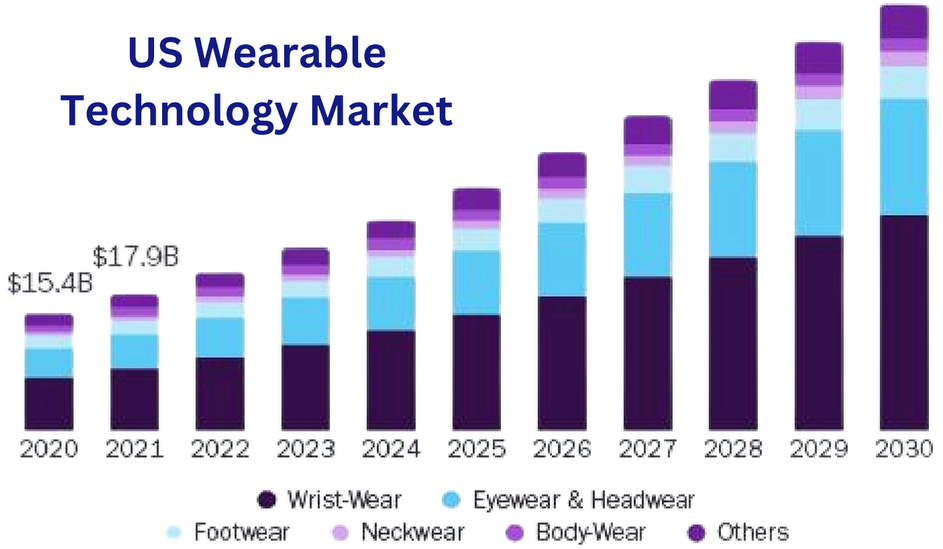
Connecting Physical Therapy
with AI to your busy life

Features	Multimodal Quantification	Real-Time Feedback	Wearable Device implementation	Physical Therapist Homework	Community Engagement
KineX AI	✓	✓	✓	✓	✓
PT Timer: Stretch & Exercise				✓	
Hinge Health				✓	
Prehab		✓			
Exer Health	✓	✓			

US Healthcare AI Market Cap



US Wearable Technology Market



$24,000,000,000 PER YEAR

PHYSICAL INACTIVITY U.S. HEALTHCARE EXPENDITURES

KINEX AI

$9.99/month

PHYSICAL THERAPY

>$100 per session

ETHICAL AI MATRIX

Respect For:	Well-Being	Autonomy	Fairness
Patients (those who cannot afford PT, or are busy/cannot fit into schedules)	• Access to health development resources • Prevents idleness with a sedentary lifestyle • Privacy accounted for	• Tailored exercise plans, remote monitoring • Support tools for assistance customized to patient needs	• PT via telehealth and remote monitoring. • HIPAA protocols + transparent data access • Inclusive content for diverse needs
Physical Therapists	• PTs offer personalized exercise plans • Maintains physical health and recovery progress	• Control their therapy schedule • Perform exercises at their convenience • Foster independence in their rehab	• PT accessible to those who can't afford it / have busy schedules • High quality care + tailored resources.
Family members of patients	• Aims for the well-being of all family members • Improving patients' health to ensures a better quality of life	• Family members are able to support patients • Focus on personal accountability .	• Affordable, customizable treatment via phone • Accounts for lower income households, eases medical bills for families.

NEXT STEPS

Patients

- Augmented Reality
- Pain Management Integration

Physical Therapists

- Real-Time biomechanical Analysis
- Personalized Engagement with Therapist Feedback

Overall Enhancements

- Global Reach
- Adaptive Learning for Enhanced Therapy
- Low Cost Subscription, less than PT expenses

AFTER CONDUCTING INTERVIEWS, ALL OF THE PHYSICAL THERAPISTS WE SPOKE TO ASKED FOR US TO IMPLEMENT THIS TOOL IN THEIR CLINIC ONCE IT WAS READY FOR LAUNCH!

WORK CITED

1. Ryan, C. G., Wellburn, S., McDonough, S., Martin, D. J., & Batterham, A. M. (2017). The association between displacement of sedentary time and chronic musculoskeletal pain: an isotemporal substitution analysis. Physiotherapy, 103(4), 471-477.
2. Dzakpasu, F. Q., Carver, A., Brakenridge, C. J., Cicuttini, F., Urquhart, D. M., Owen, N., & Dunstan, D. W. (2021). Musculoskeletal pain and sedentary behaviour in occupational and non-occupational settings: a systematic review with meta-analysis. International Journal of Behavioral Nutrition and Physical Activity, 18, 1-56.
3. Park, J. H., Moon, J. H., Kim, H. J., Kong, M. H., & Oh, Y. H. (2020). Sedentary lifestyle: overview of updated evidence of potential health risks. Korean journal of family medicine, 41(6), 365.
4. World Health Organization. (2002). The world health report 2002: reducing risks, promoting healthy life. World Health Organization.
5. Lewis, S. F., & Hennekens, C. H. (2016). Regular physical activity: forgotten benefits. The American journal of medicine, 129(2), 137-138.

NCBI

World Health Organization

UC Irvine Machine Learning Repository

THANKS FOR LISTENING

KineX

QUESTIONS?


KineX

Example on how it determines torso range

```
def torso_visible(keypoints):
  """Checks whether there are enough torso keypoints.

  This function checks whether the model is confident at predicting one of the
  shoulders/hips which is required to determine a good crop region.
  """
  return ((keypoints[0, 0, KEYPOINT_DICT['left_hip'], 2] >
           MIN_CROP_KEYPOINT_SCORE or
          keypoints[0, 0, KEYPOINT_DICT['right_hip'], 2] >
           MIN_CROP_KEYPOINT_SCORE) and
          (keypoints[0, 0, KEYPOINT_DICT['left_shoulder'], 2] >
           MIN_CROP_KEYPOINT_SCORE or
          keypoints[0, 0, KEYPOINT_DICT['right_shoulder'], 2] >
           MIN_CROP_KEYPOINT_SCORE))

def determine_torso_and_body_range(
    keypoints, target_keypoints, center_y, center_x):
  """Calculates the maximum distance from each keypoints to the center location.

  The function returns the maximum distances from the two sets of keypoints:
  full 17 keypoints and 4 torso keypoints. The returned information will be
  used to determine the crop size. See determineCropRegion for more detail.
  """
  torso_joints = ['left_shoulder', 'right_shoulder', 'left_hip', 'right_hip']
  max_torso_yrange = 0.0
  max_torso_xrange = 0.0
  for joint in torso_joints:
    dist_y = abs(center_y - target_keypoints[joint][0])
    dist_x = abs(center_x - target_keypoints[joint][1])
    if dist_y > max_torso_yrange:
      max_torso_yrange = dist_y
    if dist_x > max_torso_xrange:
      max_torso_xrange = dist_x

  max_body_yrange = 0.0
  max_body_xrange = 0.0
  for joint in KEYPOINT_DICT.keys():
    if keypoints[0, 0, KEYPOINT_DICT[joint], 2] < MIN_CROP_KEYPOINT_SCORE:
      continue
    dist_y = abs(center_y - target_keypoints[joint][0]);
    dist_x = abs(center_x - target_keypoints[joint][1]);
    if dist_y > max_body_yrange:
      max_body_yrange = dist_y
```

Real-time

1. Initialize camera
2. Take picture
3. Pose estimation
4. Repeat (while loop)

Video

1. Upload a video
2. Convert video to gif
3. Pose estimation

Movement Analysis of Pose Estimation

```
Interrupted by user
Movement Data: [[[ 2.58386135e-05 -1.70767307e-04]
  [ 3.67164612e-05 -1.51693821e-04]
  [-5.51342964e-06 -2.50041485e-04]
  [ 1.46031380e-04 -1.71244144e-04]
  [-9.29832458e-06 -3.61740589e-04]
  [ 5.37037849e-05 -1.53660774e-04]
  [-2.31772661e-04 -1.35004520e-04]
  [ 1.86204910e-04 -1.15752220e-04]
  [-6.61015511e-05 -2.21729279e-04]
  [-1.22666359e-04 -2.12192535e-05]
  [ 1.99913979e-04 -2.83718109e-05]
  [ 3.93271446e-04  1.13666058e-04]
  [ 1.31040812e-04 -6.00814819e-05]
  [ 4.70519066e-04 -1.02689862e-03]
  [ 3.13401222e-04  1.08957291e-04]
  [ 4.42266464e-04 -4.82201576e-04]
  [ 2.56538391e-04 -4.00066376e-04]]
```

```
[[-8.34167004e-05  1.17301941e-04]
 [-7.22408295e-05  1.89602375e-04]
 [-1.36494637e-04  1.26659870e-04]
 [-1.03473663e-04  2.26020813e-04]
 [-1.90526247e-04  2.58684158e-05]
 [ 6.84857368e-05  1.87277794e-04]
 [-1.67399645e-04  2.04920769e-04]
 [ 8.60691071e-05  1.58011913e-04]
 [-1.83582306e-04  1.84476376e-04]
 [-1.76370144e-04 -1.33097172e-04]
 [ 2.14576721e-05  4.45425510e-04]
 [ 2.05874443e-04  2.58564949e-04]
 [ 1.23620033e-04 -1.80602074e-05]
 [-3.57627869e-06 -3.48478556e-04]
 [-2.77161598e-05  8.41617584e-05]
 [ 9.01222229e-05 -1.11639500e-04]
 [ 3.42726707e-05  2.37196684e-04]]
```

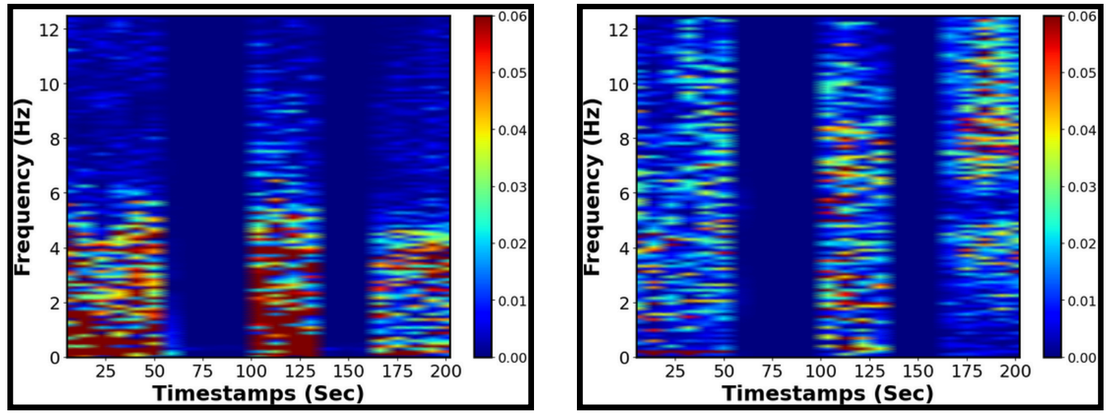
```python
KEYPOINT_DICT = {
    'nose': 0,
    'left_eye': 1,
    'right_eye': 2,
    'left_ear': 3,
    'right_ear': 4,
    'left_shoulder': 5,
    'right_shoulder': 6,
    'left_elbow': 7,
    'right_elbow': 8,
    'left_wrist': 9,
    'right_wrist': 10,
    'left_hip': 11,
    'right_hip': 12,
    'left_knee': 13,
    'right_knee': 14,
    'left_ankle': 15,
    'right_ankle': 16
}

# Maps bones to a matplotlib color name.
KEYPOINT_EDGE_INDS_TO_COLOR = {
    (0, 1): 'm',
    (0, 2): 'c',
    (1, 3): 'm',
    (2, 4): 'c',
    (0, 5): 'm',
    (0, 6): 'c',
    (5, 7): 'm',
    (7, 9): 'm',
    (6, 8): 'c',
    (8, 10): 'c',
    (5, 6): 'y',
    (5, 11): 'm',
    (6, 12): 'c',
    (11, 12): 'y',
    (11, 13): 'm',
    (13, 15): 'm',
    (12, 14): 'c',
    (14, 16): 'c'
}
```

```
Classifications: ['good', 'good', 'good', 'good', 'good', 'good', 'bad', 'bad', 'good', 'good', 'good', 'good', 'good']
```

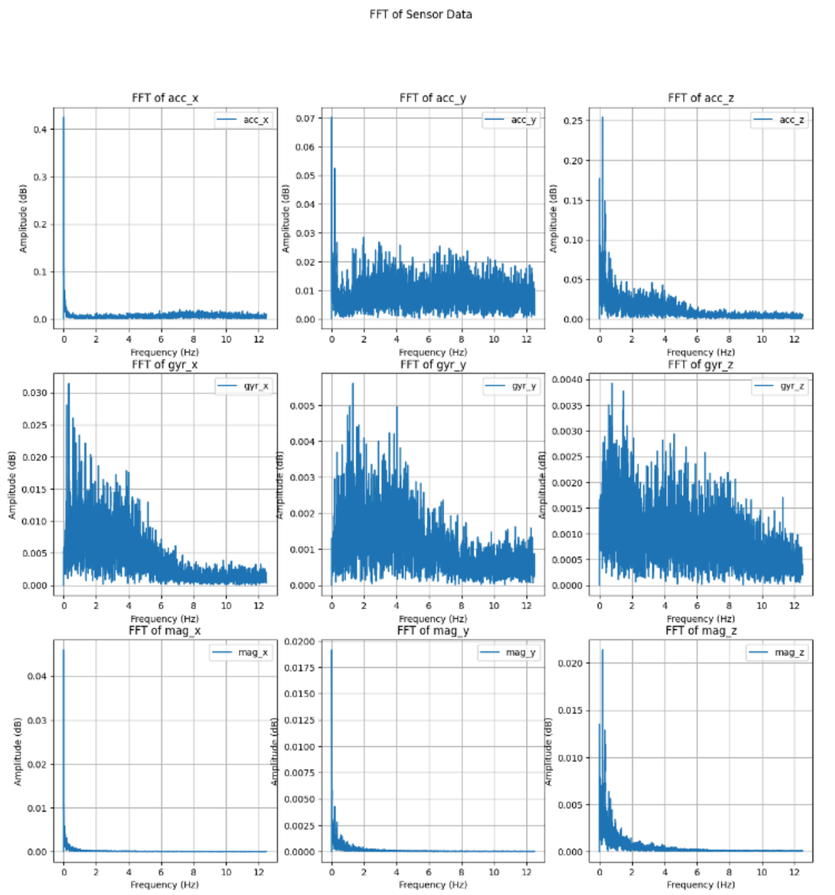
5 people -> 8 exercises -> five MTx sensors -> three tri-axial sensors: an accelerometer, a gyroscope, and a magnetometer

Signal Processing: Multimodal Data Analysis



Multimodal Data from UC Irvine Machine Learning Repo

```
time index;acc_x;acc_y;acc_z;gyr_x;gyr_y;gyr_z;mag_x;mag_y;mag_z
1;-6.141071;-6.262912;4.317228;-0.002403;0.009854;0.010958;0.138293;0.663202;-0.42293
2;-6.155866;-6.225835;4.339744;0.017658;0.034767;0.00723;0.137879;0.664667;-0.422957
3;-6.15578;-6.210988;4.332379;0.005799;0.026025;0.018196;0.1382;0.663977;-0.422114
4;-6.118875;-6.270708;4.369582;0.007551;0.015273;0.018247;0.137913;0.662528;-0.42239
5;-6.141526;-6.31513;4.406433;0.005023;0.047466;0.010807;0.139829;0.66463;-0.422329
6;-6.23028;-6.195628;4.346838;-0.002168;0.032747;-0.020154;0.14067;0.662888;-0.422007
7;-6.141422;-6.300253;4.391606;-0.027539;0.043918;-0.003814;0.139175;0.663693;-0.421798
8;-6.170992;-6.226069;4.429176;-0.014973;0.02414;-0.001887;0.138755;0.6641;-0.421955
9;-6.155984;-6.248092;4.347061;0.004931;0.016791;-0.003654;0.139797;0.664712;-0.420405
10;-6.141051;-6.226103;4.377166;-0.006027;0.002585;4.3e-05;0.139326;0.664348;-0.421522
11;-6.126316;-6.255919;4.392009;-0.017675;0.027768;-8e-05;0.138829;0.664992;-0.420583
12;-6.126005;-6.211287;4.347528;-0.002307;0.018642;-0.003668;0.13908;0.665;-0.420986
13;-6.200869;-6.284777;4.346482;-0.014882;0.03114;-0.012868;0.139635;0.665521;-0.42083
14;-6.185726;-6.240384;4.331961;-0.016752;0.019503;-0.012818;0.139575;0.664688;-0.419451
15;-6.171087;-6.299777;4.331686;0.019344;0.005099;0.001884;0.137945;0.664434;-0.419239
16;-6.133525;-6.226045;4.347374;-0.015072;0.009906;0.007295;0.138903;0.665618;-0.41921
17;-6.133685;-6.277791;4.317187;-0.000451;0.016662;-0.016428;0.139198;0.666267;-0.41879
18;-6.155523;-6.210575;4.227911;0.001244;0.004287;4.1e-05;0.137803;0.666692;-0.419262
19;-6.170284;-6.166089;4.250475;-0.007727;0.025934;0.005408;0.139535;0.666035;-0.42015
20;-6.118428;-6.24057;4.242921;-0.002262;0.031271;0.004483;0.140069;0.664775;-0.417919
21;-6.118158;-6.188648;4.228336;0.001367;0.045806;0.021755;0.14034;0.666232;-0.417919
22;-6.118569;-6.255506;4.272673;-0.013049;0.063887;0.025304;0.141725;0.66674;-0.417592
23;-6.171117;-6.314536;4.316664;0.021362;0.058172;0.018074;0.142089;0.667355;-0.415931
24;-6.178778;-6.34423;4.346262;-0.042015;0.073144;0.032514;0.143705;0.666688;-0.416412
25;-6.261175;-6.403078;4.404954;0.03505;0.079584;0.017991;0.146277;0.66767;-0.41439
```



fast fourier transform data

Code Snippets of LLM: Includes data imports (training/testing), transformer import used, and query output example

```python
# Step 3: Load the conversational model
from transformers import AutoModelForCausalLM, AutoTokenizer, Trainer, TrainingArguments

# Use a valid model name from the Hugging Face Model Hub
model_name = "microsoft/DialoGPT-small"
tokenizer = AutoTokenizer.from_pretrained(model_name)

# Add a padding token to the tokenizer
tokenizer.pad_token = tokenizer.eos_token

model = AutoModelForCausalLM.from_pretrained(model_name)
```

```python
import datasets
# Tokenize the data
def tokenize_function(examples):
    return tokenizer(examples['prompt'],  # Tokenize the 'prompt' column
                     examples['response'], # Tokenize the 'response' column
                     padding="max_length",
                     truncation=True)

train_dataset = datasets.Dataset.from_pandas(train_df)
test_dataset = datasets.Dataset.from_pandas(test_df)

train_dataset = train_dataset.map(tokenize_function, batched=True)
test_dataset = test_dataset.map(tokenize_function, batched=True)

# Convert tokens to tensors and ensure proper alignment
train_dataset.set_format(type='torch', columns=['input_ids', 'attention_mask'])
test_dataset.set_format(type='torch', columns=['input_ids', 'attention_mask'])
```

```
Map: 100% 80/80 [00:00<00:00, 762.81 examples/s]
Map: 100% 4/4 [00:00<00:00, 73.72 examples/s]
```

```python
# Step 2: Load the data
import json
import pandas as pd

datapath = '/content/drive/MyDrive/SureStart Mentorship/LLM/'

# Load training data
with open(datapath + 'training_data.json', 'r') as f:
    train_data = json.load(f)

# Load test data
with open(datapath + 'test_data.json', 'r') as f:
    test_data = json.load(f)
```

```python
def json_to_dataframe(data):
    records = []
    for category, details in data.items():
        for exercise in details['exercises']:
            prompt = f"What exercises can I do for {category} pain?"
            response = exercise['explanation']
            records.append({
                'prompt': prompt,
                'response': response
            })
    return pd.DataFrame(records)

train_df = json_to_dataframe(train_data)
test_df = json_to_dataframe(test_data)
```

```python
def generate_response(prompt, max_length=50, temperature=0.7):
    inputs = tokenizer(prompt, return_tensors="pt", padding=True, truncation=True).to('cuda')
    outputs = model.generate(**inputs, max_length=max_length, temperature=temperature, do_sample=False, pad_token_id=tokenizer.eos_token_id)
    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return response

prompt = "What exercises can I do for lower back pain?"
response = generate_response(prompt, max_length=50, temperature=0.8)
print(response)
```

```
/usr/local/lib/python3.10/dist-packages/transformers/generation/configuration_utils.py:540: UserWarning: `do_sample` is set to `False`. However, `temperature` is set to `0.8` -- this flag is on
  warnings.warn(
What exercises can I do for lower back pain?Lie on your back with your knees bent. Flatten your back against the floor by tightening your abdominal muscles and tilting your pelvis up slightly.
```

```python
# Example usage (Modified)
prompt = "What exercises can I do for lower neck pain? "
response = generate_response(prompt, max_length=50, temperature=0.8)  # Adjusted parameters
print(response)
```