US 20220327807A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2022/0327807 A1**

COOK et al. (43) **Pub. Date: Oct. 13, 2022**

(54) **CONTINUALLY LEARNING AUDIO FEEDBACK ENGINE**

(71) Applicant: **EXER Labs, Inc.**, Denver, CO (US)

(72) Inventors: **Sean Christopher COOK**, Denver, CO (US); **Zaw Lynn THET**, Denver, CO (US); **Clint J. GEHDE**, Denver, CO (US); **Nicole Xiaoxuan AW**, Denver, CO (US); **Daniel James FISHER**, Denver, CO (US); **Nicolas ARELLANO**, Denver, CO (US)

(73) Assignee: **EXER Labs, Inc.**, Denver, CO (US)

(21) Appl. No.: **17/709,381**

(22) Filed: **Mar. 30, 2022**

Related U.S. Application Data

(60) Provisional application No. 63/169,777, filed on Apr. 1, 2021, provisional application No. 63/169,778, filed on Apr. 1, 2021.

Publication Classification

(51) **Int. Cl.**

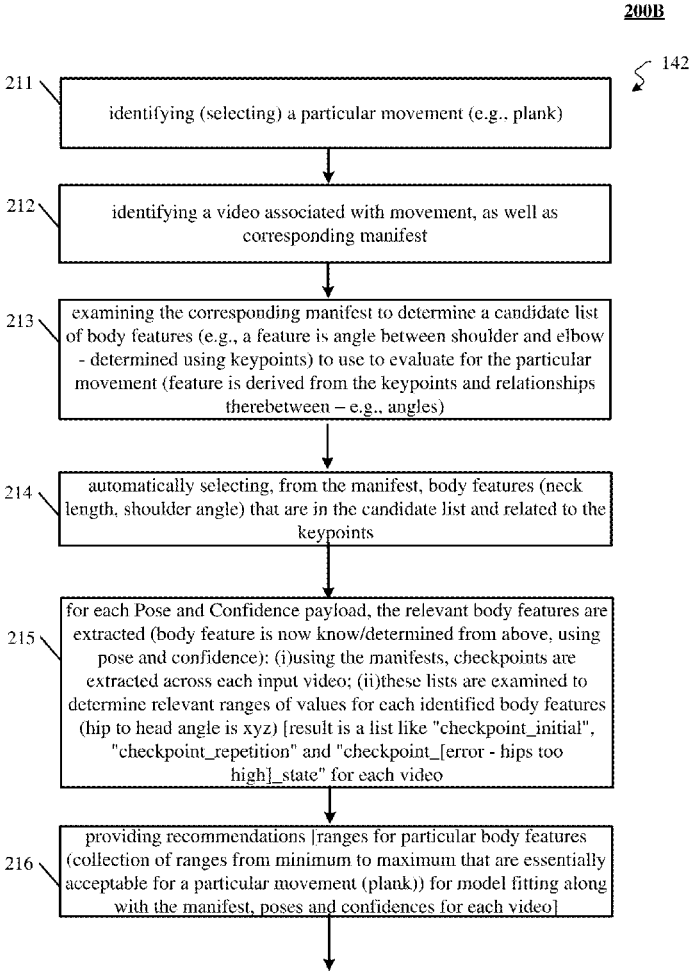
G06V 10/774	(2006.01)
G06V 10/82	(2006.01)
G06T 13/40	(2006.01)
G06T 7/77	(2006.01)

(52) **U.S. Cl.**

CPC ***G06V 10/774*** (2022.01); ***G06V 10/82*** (2022.01); ***G06T 13/40*** (2013.01); ***G06T 7/77*** (2017.01); *G06T 2207/30196* (2013.01)

(57) **ABSTRACT**

The present technology provides systems, methods and computer program instructions implementing machine learning techniques to enable program processes to learn more effective feedback mechanisms to achieve desired results (e.g., reduce errors, improve form, duration, speed, and so forth) of motions and poses comprising tasks being taught or guided. In implementations an automated technology for automated creation of movement assessments from labeled video and continually learning audio, video or other feedback for use with machine learning techniques enable program processes to learn more effective feedback mechanisms to achieve desired results (e.g., reduce errors, improve form, duration, speed, and so forth) of motions and poses comprising tasks being taught or guided.



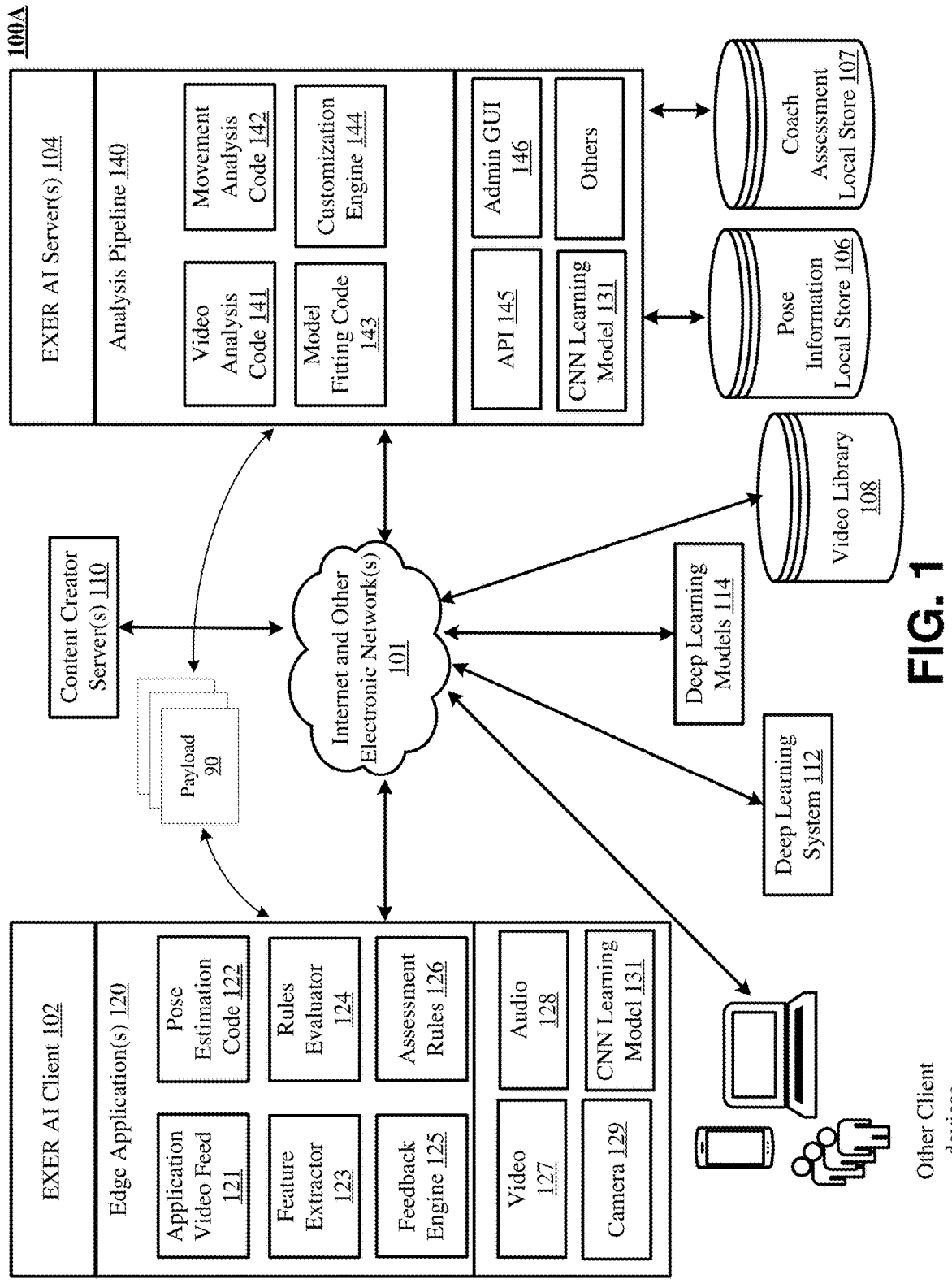

100A
EXER AI Server(s) 104
Analysis Pipeline 140
Video Analysis Code 141
Movement Analysis Code 142
Model Fitting Code 143
Customization Engine 144
API 145
Admin GUI 146
CNN Learning Model 131
Others
Coach Assessment Local Store 107
Pose Information Local Store 106
Video Library 108
Content Creator Server(s) 110
Internet and Other Electronic Network(s) 101
Payload 90
Deep Learning Models 114
Deep Learning System 112
EXER AI Client 102
Edge Application(s) 120
Application Video Feed 121
Pose Estimation Code 122
Feature Extractor 123
Rules Evaluator 124
Feedback Engine 125
Assessment Rules 126
Video 127
Audio 128
Camera 129
CNN Learning Model 131
Other Client devices

FIG. 1

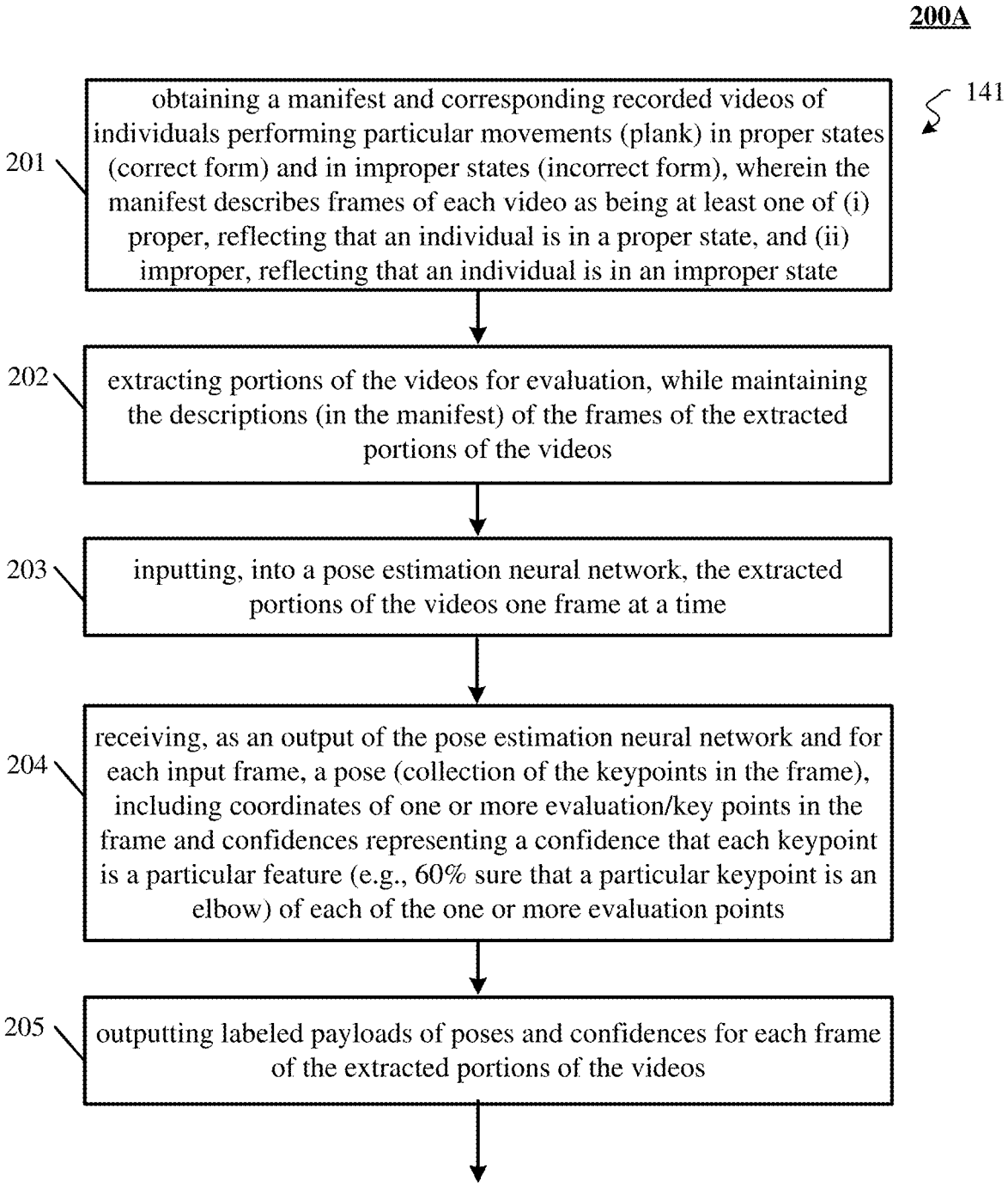

200A
141
201
obtaining a manifest and corresponding recorded videos of individuals performing particular movements (plank) in proper states (correct form) and in improper states (incorrect form), wherein the manifest describes frames of each video as being at least one of (i) proper, reflecting that an individual is in a proper state, and (ii) improper, reflecting that an individual is in an improper state
202
extracting portions of the videos for evaluation, while maintaining the descriptions (in the manifest) of the frames of the extracted portions of the videos
203
inputting, into a pose estimation neural network, the extracted portions of the videos one frame at a time
204
receiving, as an output of the pose estimation neural network and for each input frame, a pose (collection of the keypoints in the frame), including coordinates of one or more evaluation/key points in the frame and confidences representing a confidence that each keypoint is a particular feature (e.g., 60% sure that a particular keypoint is an elbow) of each of the one or more evaluation points
205
outputting labeled payloads of poses and confidences for each frame of the extracted portions of the videos

FIG. 2A

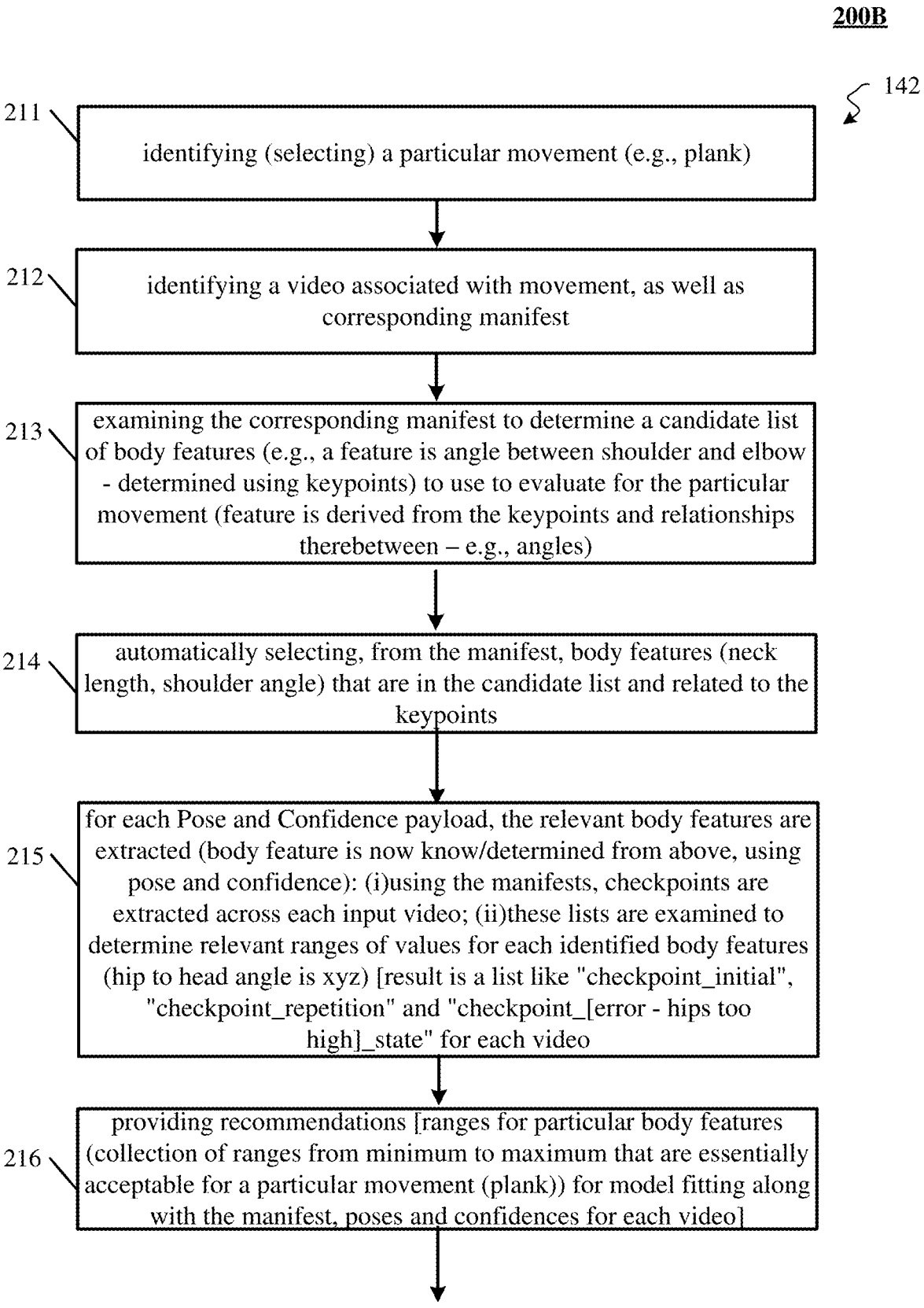

200B
142
211
identifying (selecting) a particular movement (e.g., plank)
212
identifying a video associated with movement, as well as corresponding manifest
213
examining the corresponding manifest to determine a candidate list of body features (e.g., a feature is angle between shoulder and elbow - determined using keypoints) to use to evaluate for the particular movement (feature is derived from the keypoints and relationships therebetween – e.g., angles)
214
automatically selecting, from the manifest, body features (neck length, shoulder angle) that are in the candidate list and related to the keypoints
215
for each Pose and Confidence payload, the relevant body features are extracted (body feature is now know/determined from above, using pose and confidence): (i)using the manifests, checkpoints are extracted across each input video; (ii)these lists are examined to determine relevant ranges of values for each identified body features (hip to head angle is xyz) [result is a list like "checkpoint_initial", "checkpoint_repetition" and "checkpoint_[error - hips too high]_state" for each video
216
providing recommendations [ranges for particular body features (collection of ranges from minimum to maximum that are essentially acceptable for a particular movement (plank)) for model fitting along with the manifest, poses and confidences for each video]

FIG. 2B

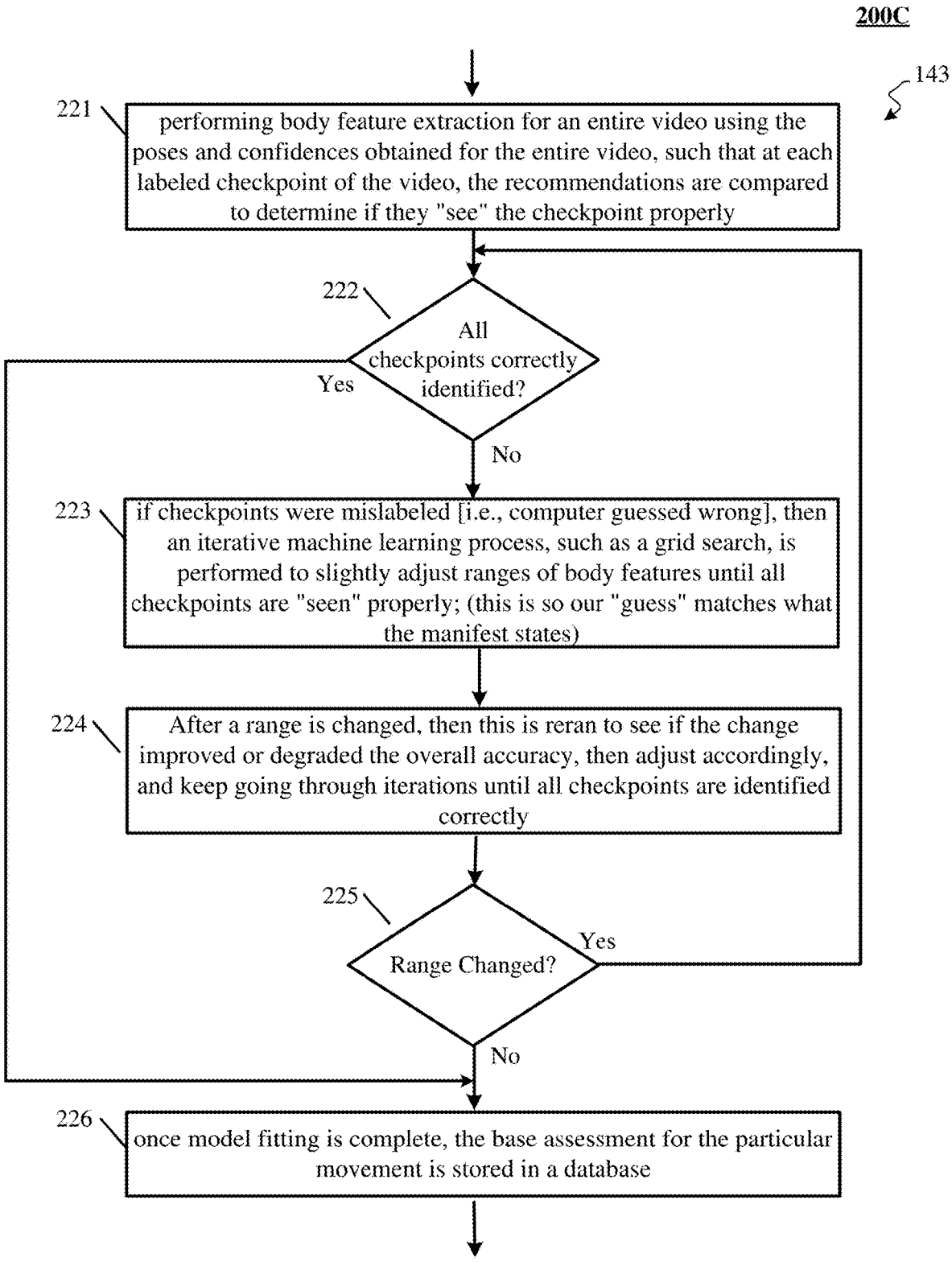

200C
143
221
performing body feature extraction for an entire video using the poses and confidences obtained for the entire video, such that at each labeled checkpoint of the video, the recommendations are compared to determine if they "see" the checkpoint properly
222
All checkpoints correctly identified?
Yes
No
223
if checkpoints were mislabeled [i.e., computer guessed wrong], then an iterative machine learning process, such as a grid search, is performed to slightly adjust ranges of body features until all checkpoints are "seen" properly; (this is so our "guess" matches what the manifest states)
224
After a range is changed, then this is reran to see if the change improved or degraded the overall accuracy, then adjust accordingly, and keep going through iterations until all checkpoints are identified correctly
225
Range Changed?
Yes
No
226
once model fitting is complete, the base assessment for the particular movement is stored in a database

FIG. 2C

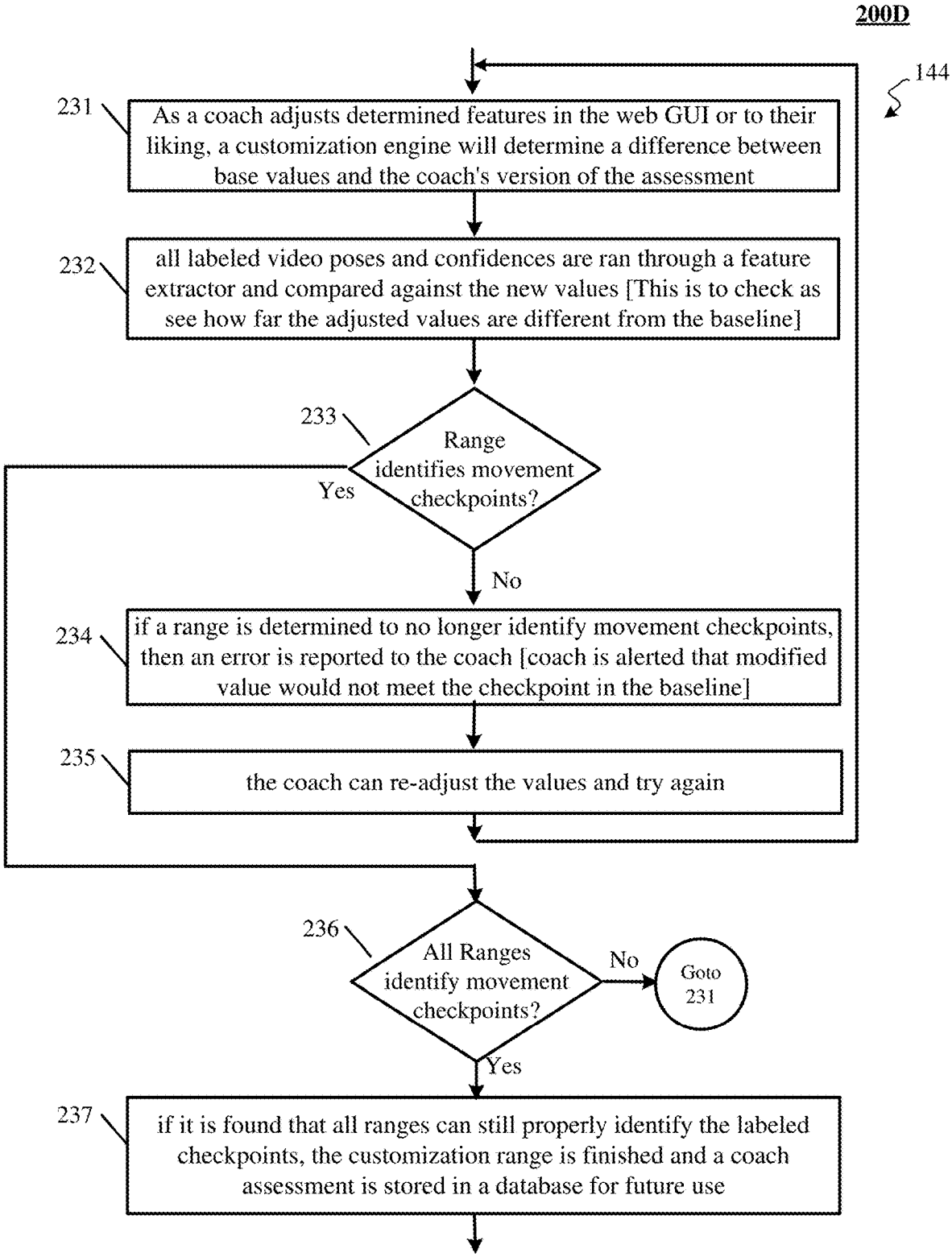

200D
144
231
As a coach adjusts determined features in the web GUI or to their liking, a customization engine will determine a difference between base values and the coach's version of the assessment
232
all labeled video poses and confidences are ran through a feature extractor and compared against the new values [This is to check as see how far the adjusted values are different from the baseline]
233
Range identifies movement checkpoints?
Yes
No
234
if a range is determined to no longer identify movement checkpoints, then an error is reported to the coach [coach is alerted that modified value would not meet the checkpoint in the baseline]
235
the coach can re-adjust the values and try again
236
All Ranges identify movement checkpoints?
No
Goto 231
Yes
237
if it is found that all ranges can still properly identify the labeled checkpoints, the customization range is finished and a coach assessment is stored in a database for future use

FIG. 2D

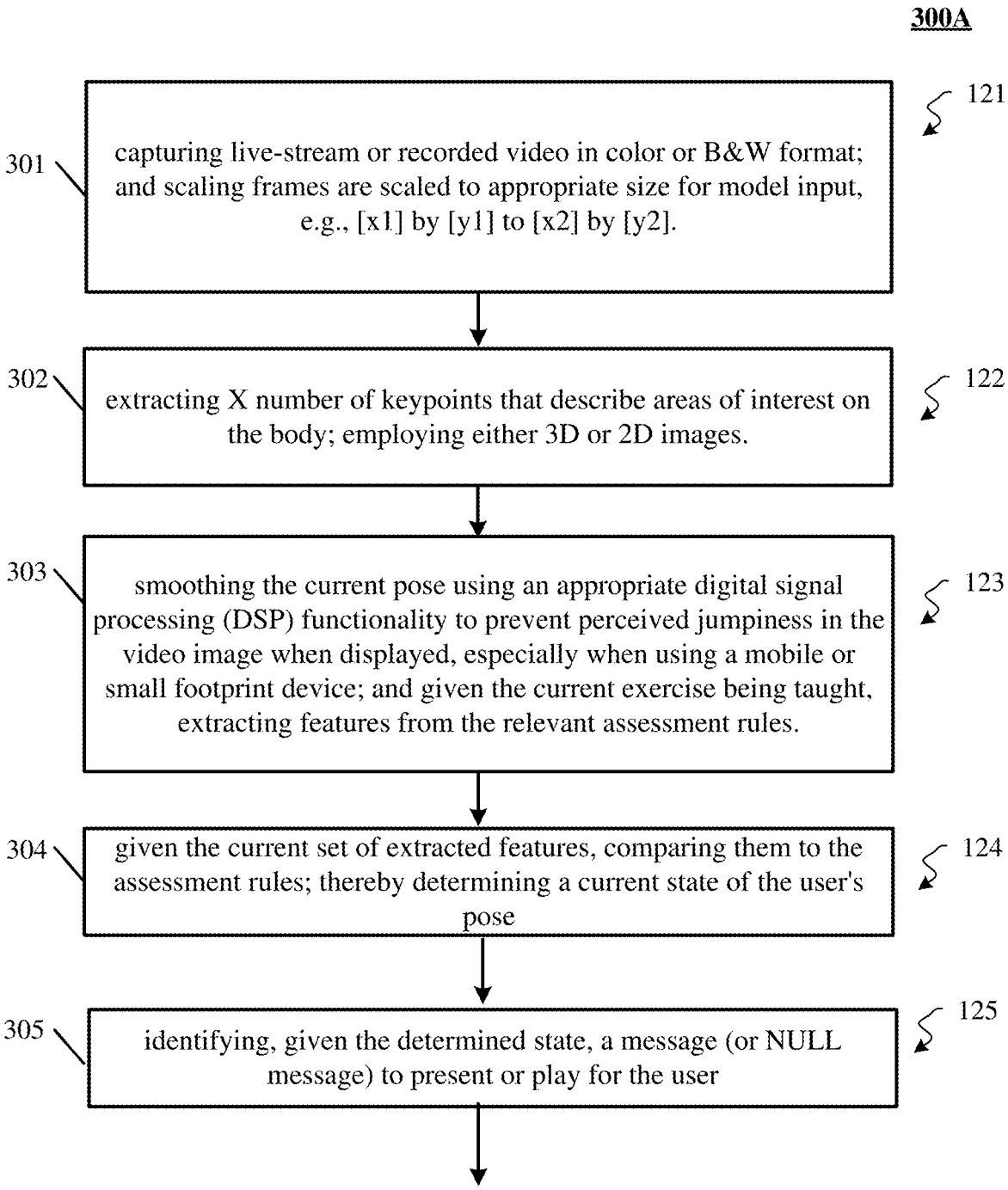

300A
301
capturing live-stream or recorded video in color or B&W format; and scaling frames are scaled to appropriate size for model input, e.g., [x1] by [y1] to [x2] by [y2].
121
302
extracting X number of keypoints that describe areas of interest on the body; employing either 3D or 2D images.
122
303
smoothing the current pose using an appropriate digital signal processing (DSP) functionality to prevent perceived jumpiness in the video image when displayed, especially when using a mobile or small footprint device; and given the current exercise being taught, extracting features from the relevant assessment rules.
123
304
given the current set of extracted features, comparing them to the assessment rules; thereby determining a current state of the user's pose
124
305
identifying, given the determined state, a message (or NULL message) to present or play for the user
125

FIG. 3A

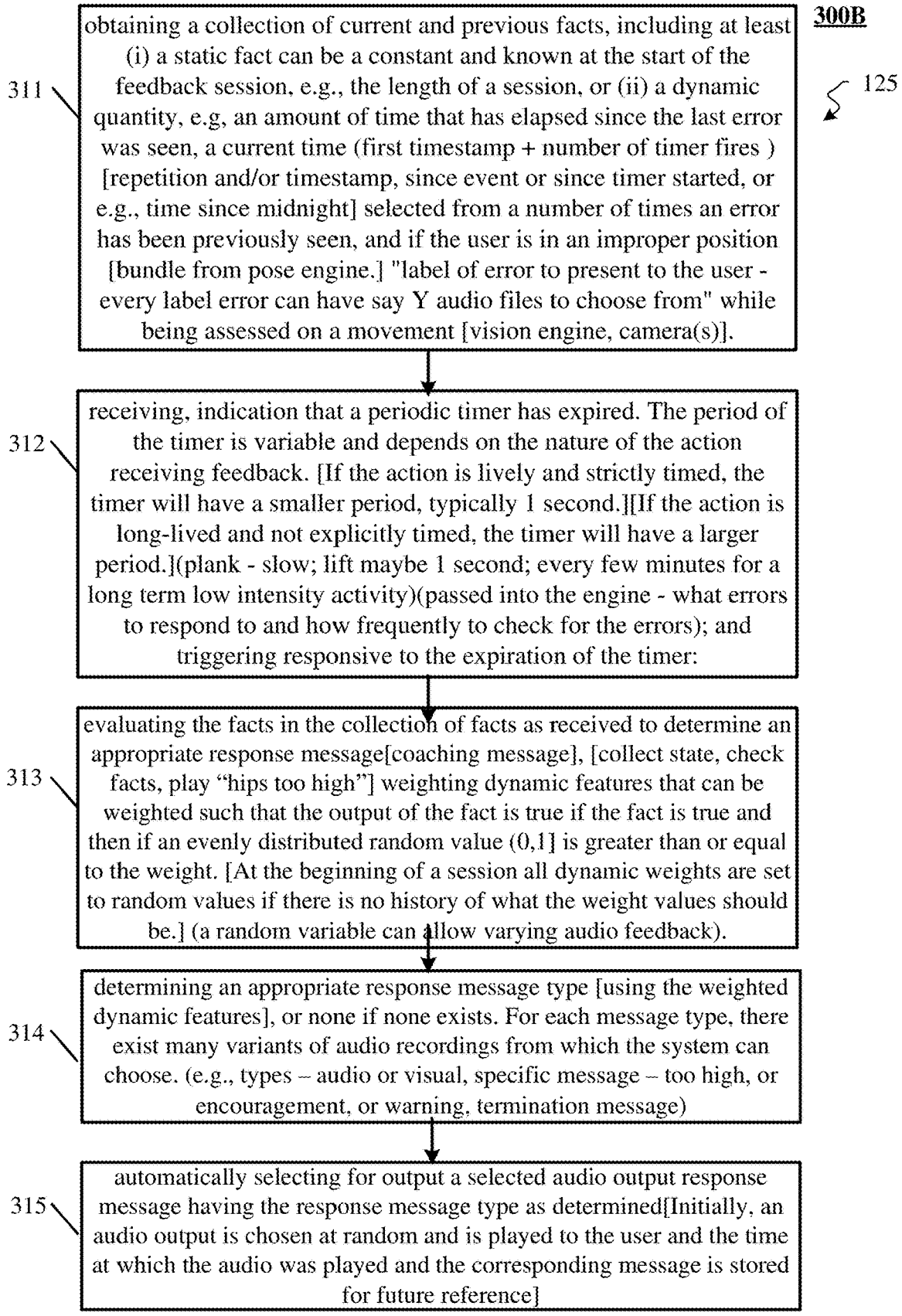

300B
125
311
obtaining a collection of current and previous facts, including at least (i) a static fact can be a constant and known at the start of the feedback session, e.g., the length of a session, or (ii) a dynamic quantity, e.g, an amount of time that has elapsed since the last error was seen, a current time (first timestamp + number of timer fires) [repetition and/or timestamp, since event or since timer started, or e.g., time since midnight] selected from a number of times an error has been previously seen, and if the user is in an improper position [bundle from pose engine.] "label of error to present to the user - every label error can have say Y audio files to choose from" while being assessed on a movement [vision engine, camera(s)].
312
receiving, indication that a periodic timer has expired. The period of the timer is variable and depends on the nature of the action receiving feedback. [If the action is lively and strictly timed, the timer will have a smaller period, typically 1 second.][If the action is long-lived and not explicitly timed, the timer will have a larger period.](plank - slow; lift maybe 1 second; every few minutes for a long term low intensity activity)(passed into the engine - what errors to respond to and how frequently to check for the errors); and triggering responsive to the expiration of the timer:
313
evaluating the facts in the collection of facts as received to determine an appropriate response message[coaching message], [collect state, check facts, play "hips too high"] weighting dynamic features that can be weighted such that the output of the fact is true if the fact is true and then if an evenly distributed random value (0,1] is greater than or equal to the weight. [At the beginning of a session all dynamic weights are set to random values if there is no history of what the weight values should be.] (a random variable can allow varying audio feedback).
314
determining an appropriate response message type [using the weighted dynamic features], or none if none exists. For each message type, there exist many variants of audio recordings from which the system can choose. (e.g., types – audio or visual, specific message – too high, or encouragement, or warning, termination message)
315
automatically selecting for output a selected audio output response message having the response message type as determined[Initially, an audio output is chosen at random and is played to the user and the time at which the audio was played and the corresponding message is stored for future reference]

FIG. 3B

300C

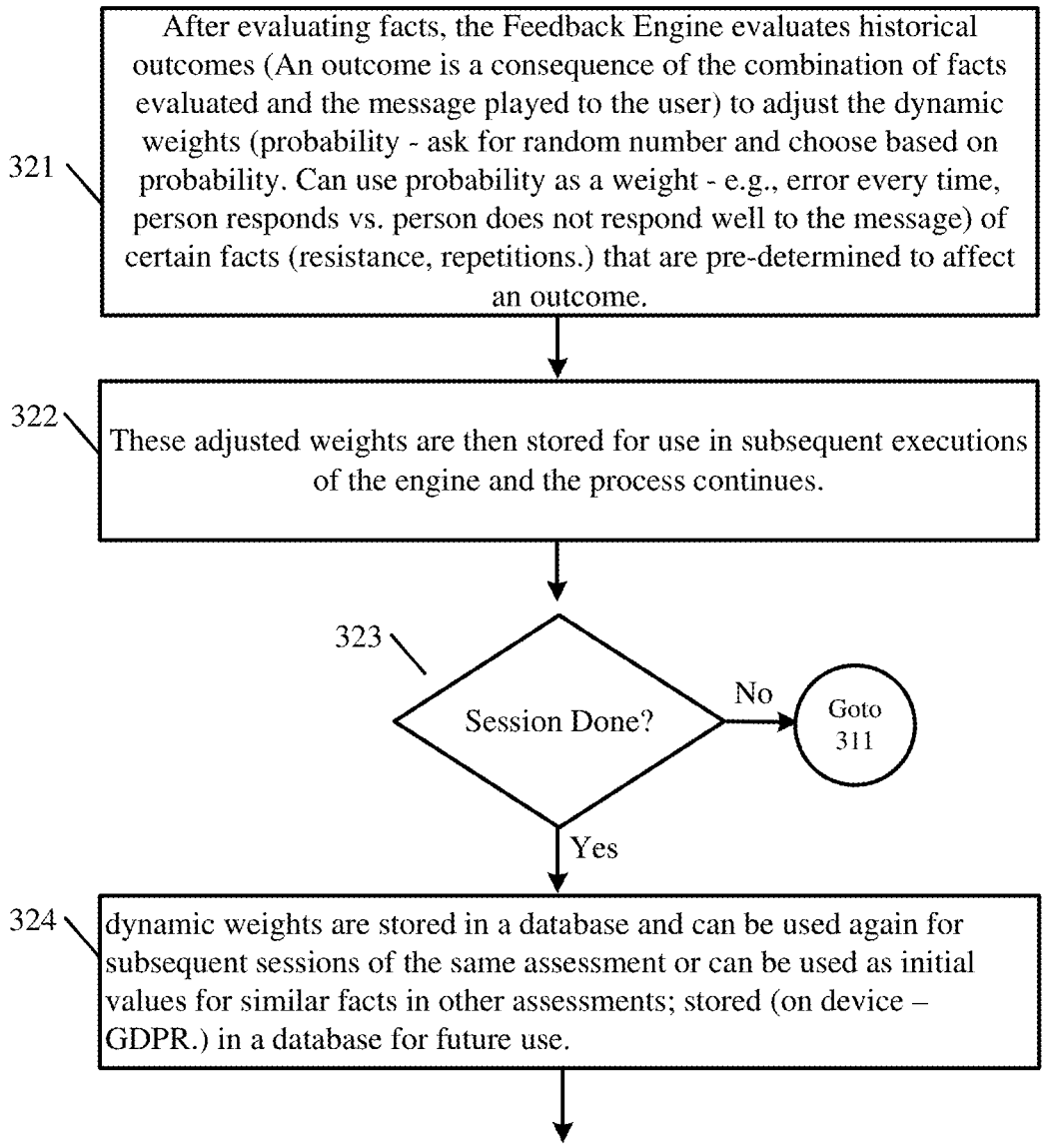

After evaluating facts, the Feedback Engine evaluates historical outcomes (An outcome is a consequence of the combination of facts evaluated and the message played to the user) to adjust the dynamic weights (probability - ask for random number and choose based on probability. Can use probability as a weight - e.g., error every time, person responds vs. person does not respond well to the message) of certain facts (resistance, repetitions.) that are pre-determined to affect an outcome.
321
These adjusted weights are then stored for use in subsequent executions of the engine and the process continues.
322
323
Session Done?
No
Goto 311
Yes
dynamic weights are stored in a database and can be used again for subsequent sessions of the same assessment or can be used as initial values for similar facts in other assessments; stored (on device – GDPR.) in a database for future use.
324

FIG. 3C

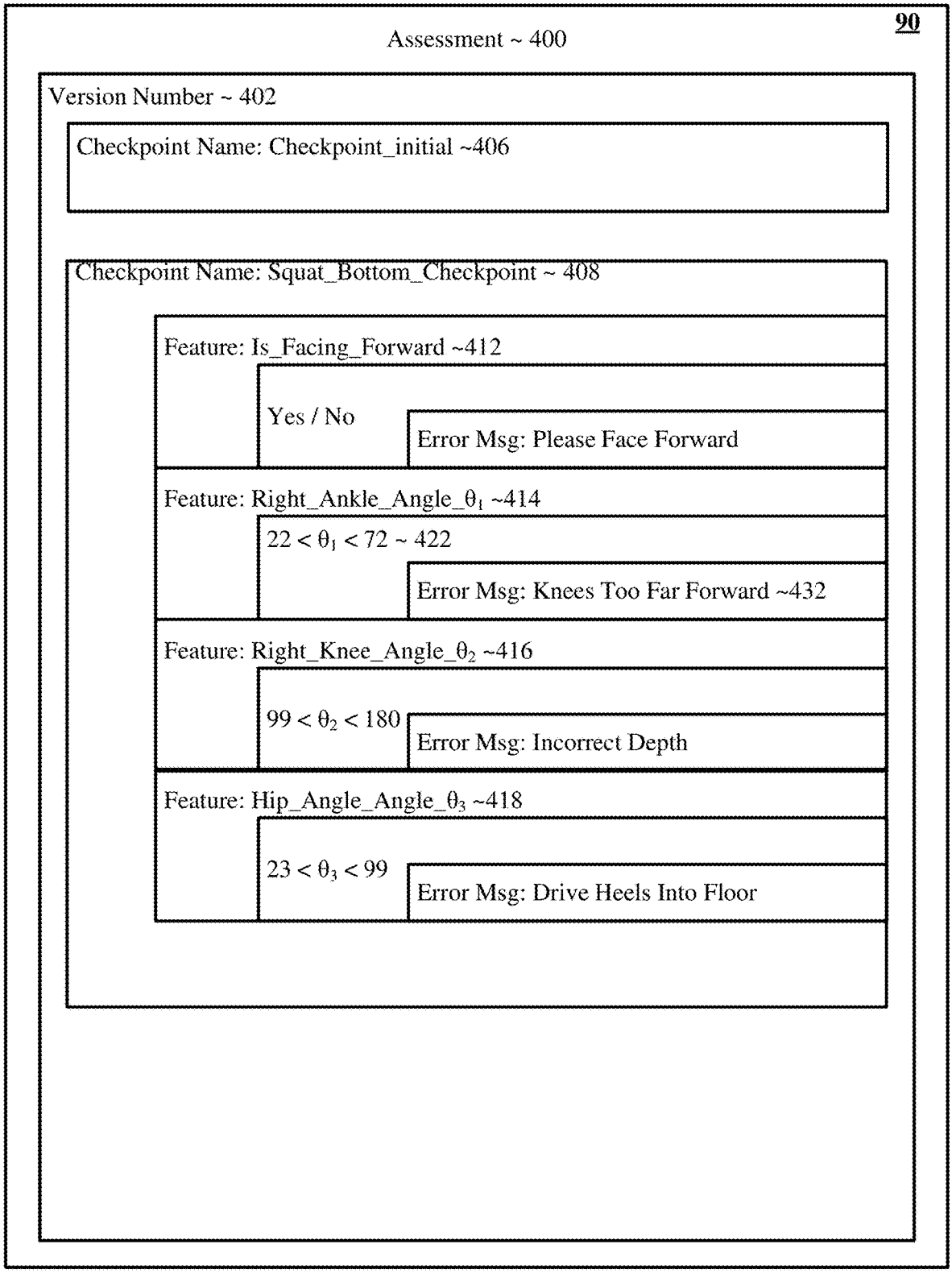

90
Assessment ~ 400
Version Number ~ 402
Checkpoint Name: Checkpoint_initial ~406
Checkpoint Name: Squat_Bottom_Checkpoint ~ 408
Feature: Is_Facing_Forward ~412
Yes / No
Error Msg: Please Face Forward
Feature: Right_Ankle_Angle_θ1 ~414
22 < θ1 < 72 ~ 422
Error Msg: Knees Too Far Forward ~432
Feature: Right_Knee_Angle_θ2 ~416
99 < θ2 < 180
Error Msg: Incorrect Depth
Feature: Hip_Angle_Angle_θ3 ~418
23 < θ3 < 99
Error Msg: Drive Heels Into Floor

FIG. 4

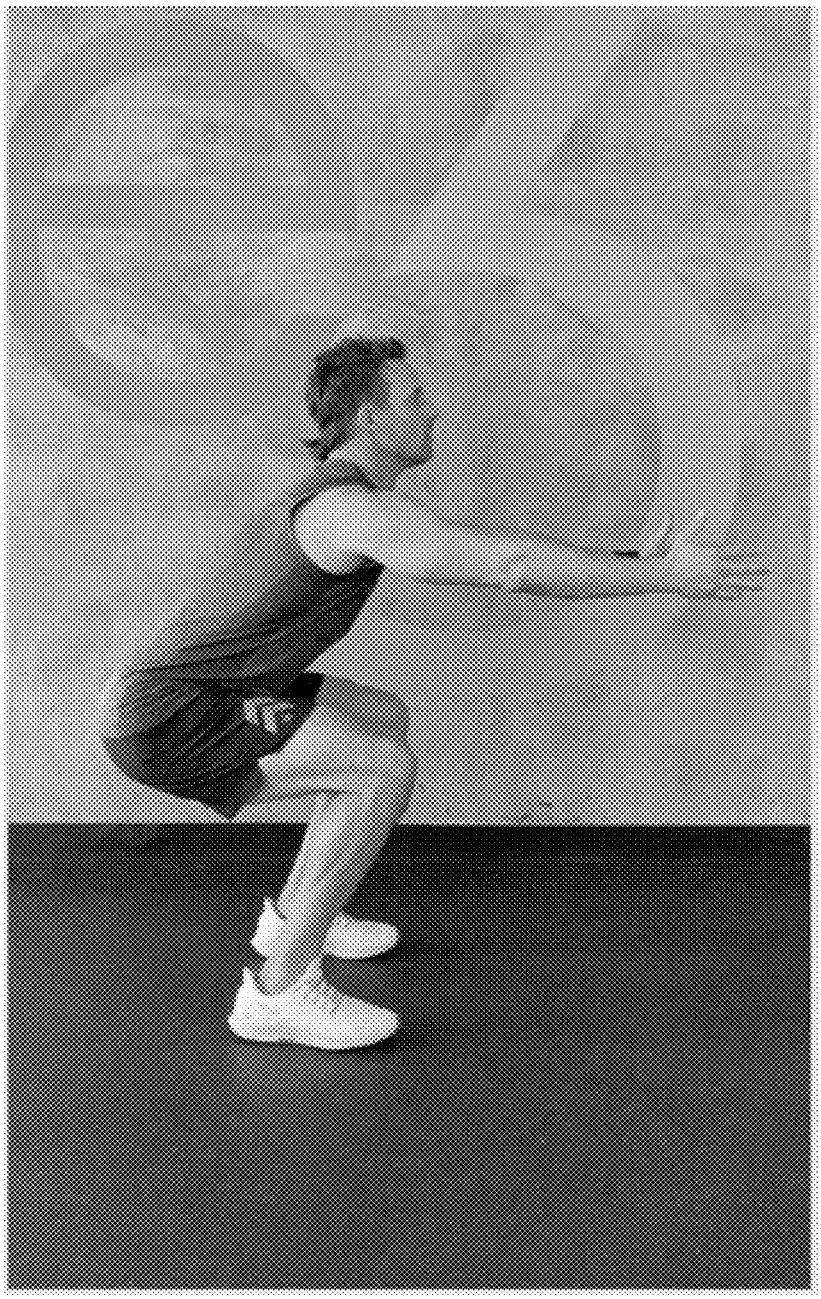


FIG. 5A

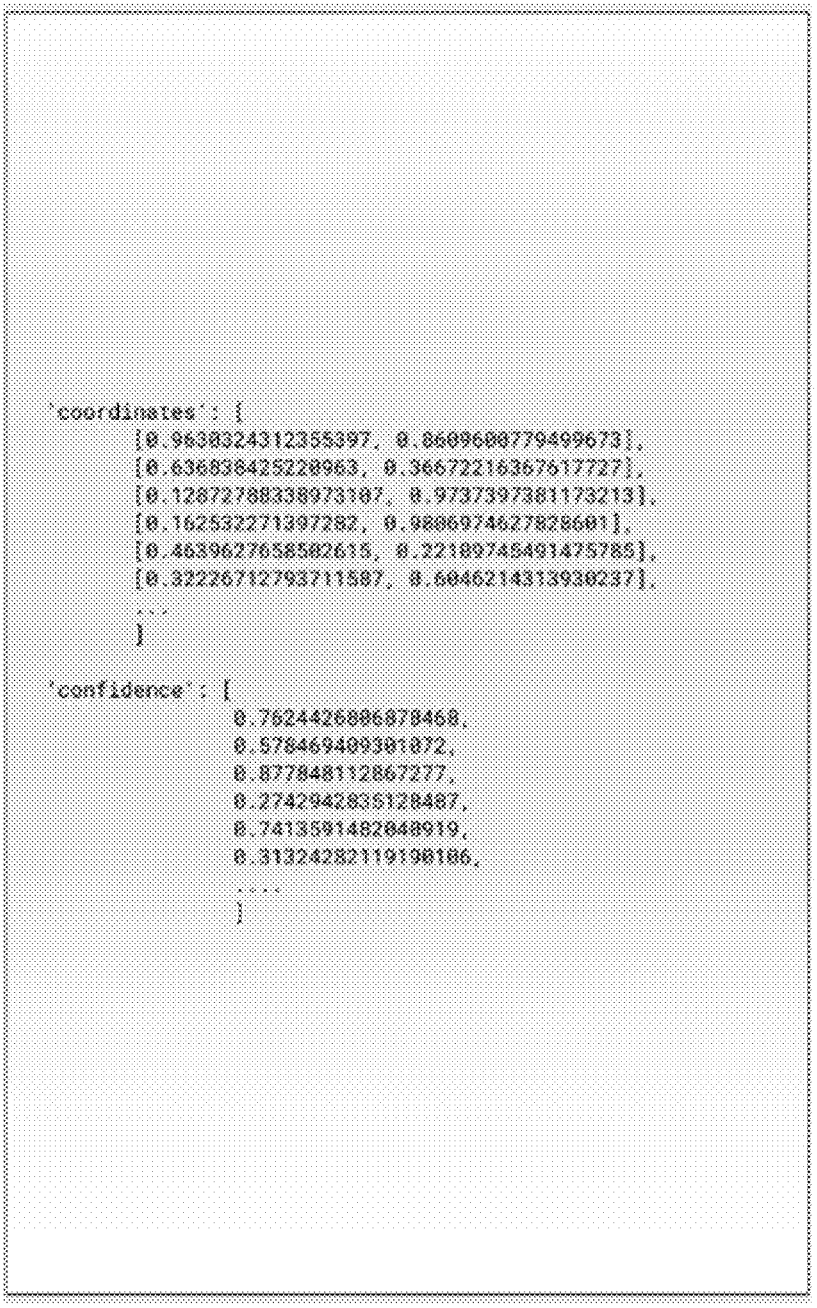

'coordinates': [
[0.9638324312355397, 0.8609600779499673],
[0.636838425228963, 0.3667221636761727],
[0.12872708238973107, 0.9737397381173213],
[0.1625322713997282, 0.9886974627828601],
[0.4639627658582615, 0.22189745491475785],
[0.3222671279371587, 0.6046214313930237],
...
]
'confidence': [
0.7624426896878468,
0.5784694099301072,
0.877848112867277,
0.27429428351284487,
0.7413591482848919,
0.31324282119198186,
...
]

FIG. 5B

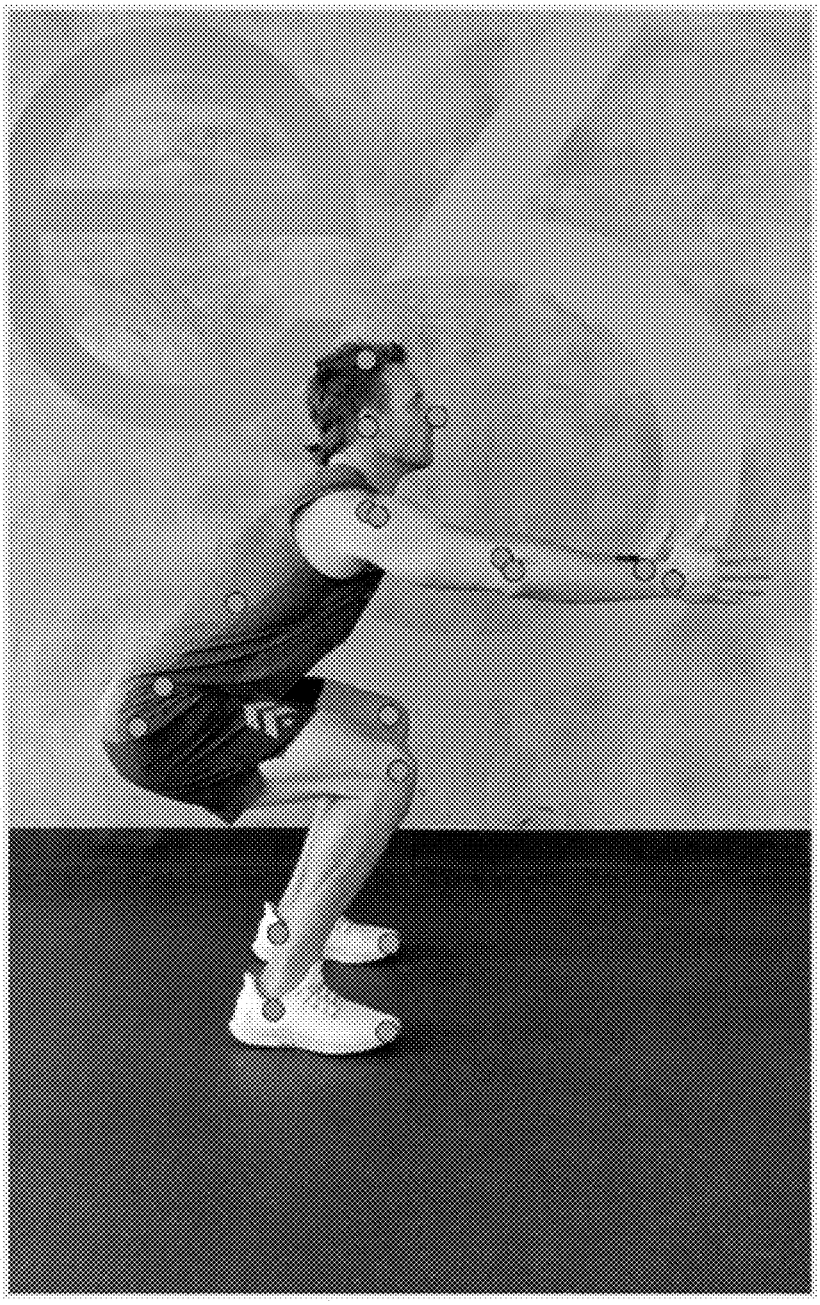


FIG. 5C

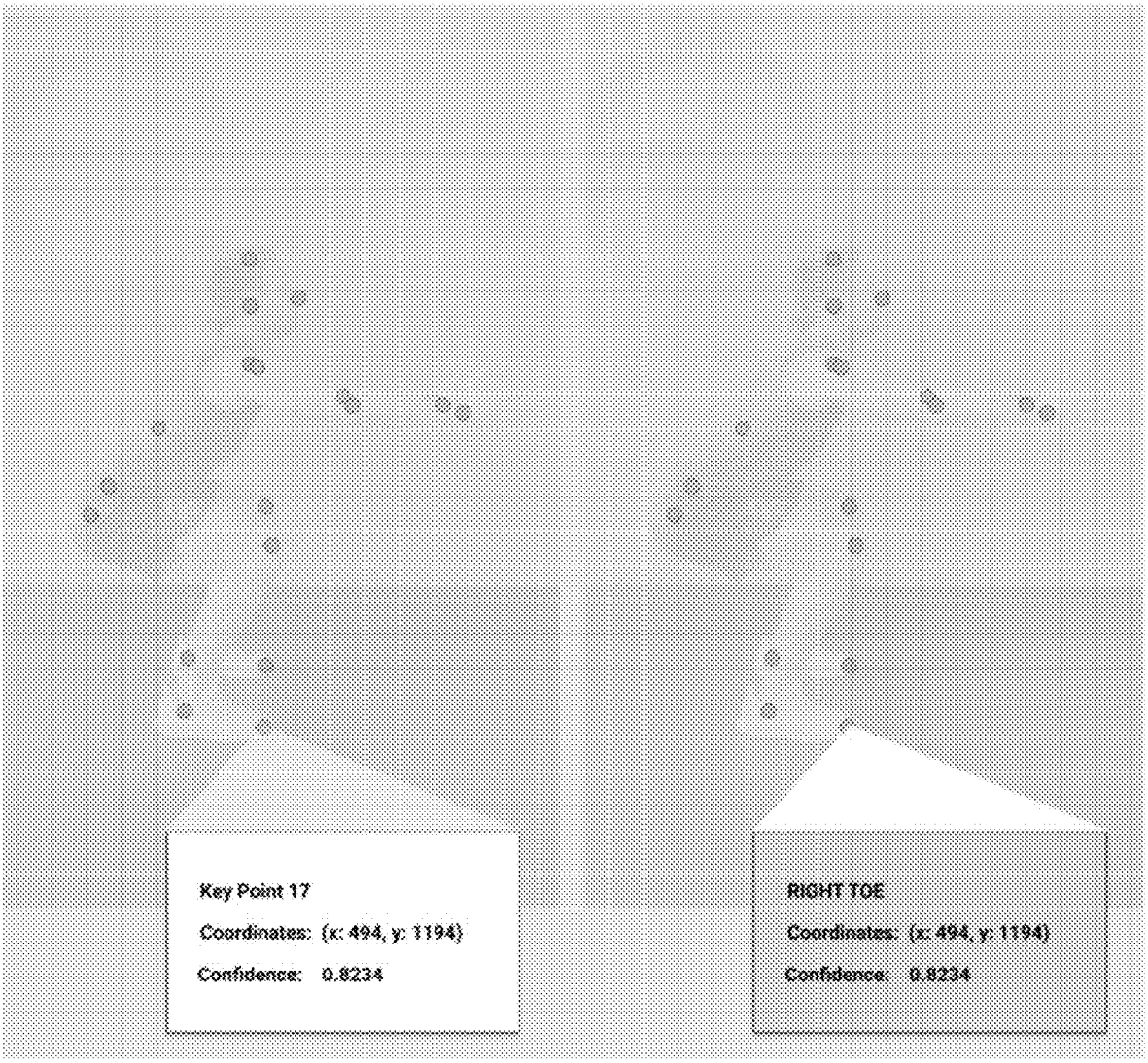

Key Point 17
Coordinates: (x: 494, y: 1194)
Confidence: 0.8234
RIGHT TOE
Coordinates: (x: 494, y: 1194)
Confidence: 0.8234

FIG. 5D

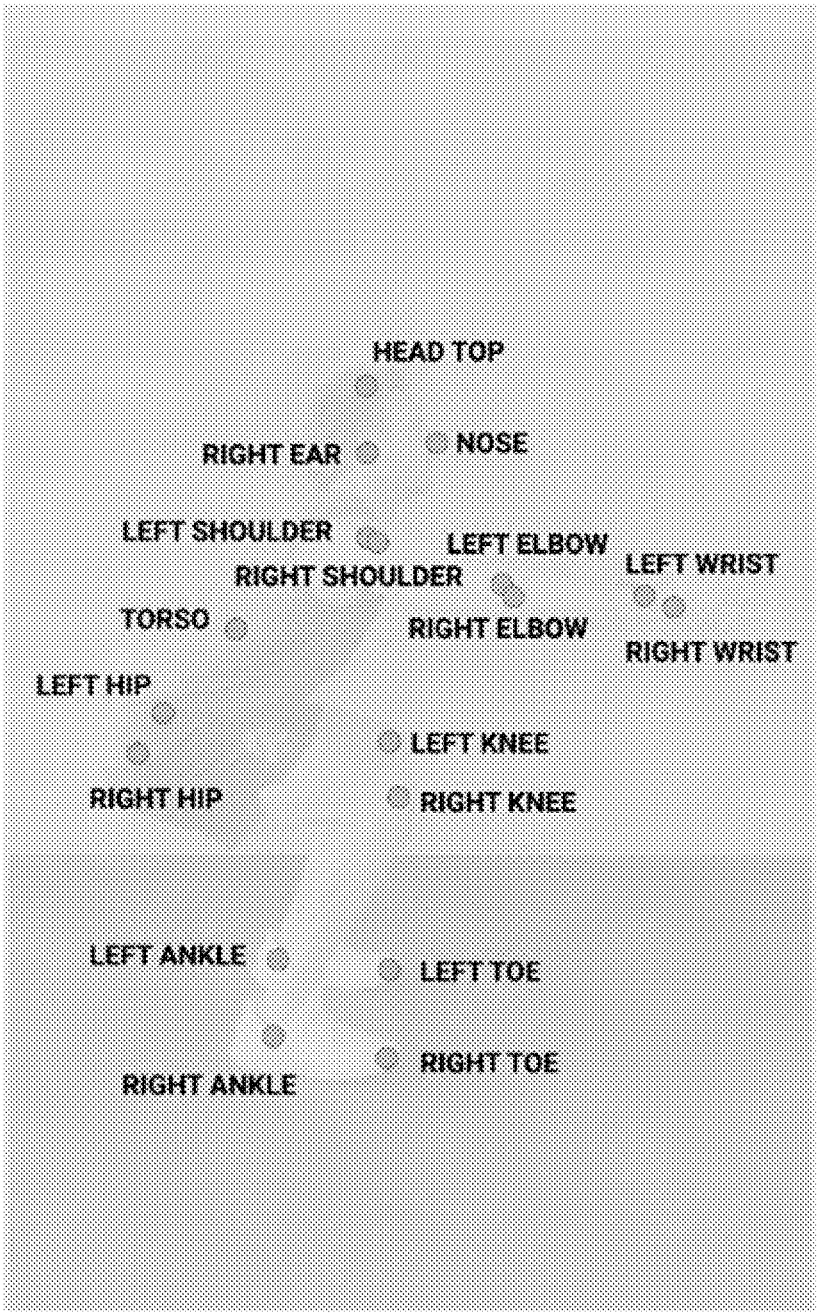

HEAD TOP
NOSE
RIGHT EAR
LEFT SHOULDER
LEFT ELBOW
LEFT WRIST
RIGHT SHOULDER
TORSO
RIGHT ELBOW
RIGHT WRIST
LEFT HIP
LEFT KNEE
RIGHT HIP
RIGHT KNEE
LEFT ANKLE
LEFT TOE
RIGHT TOE
RIGHT ANKLE

FIG. 5E

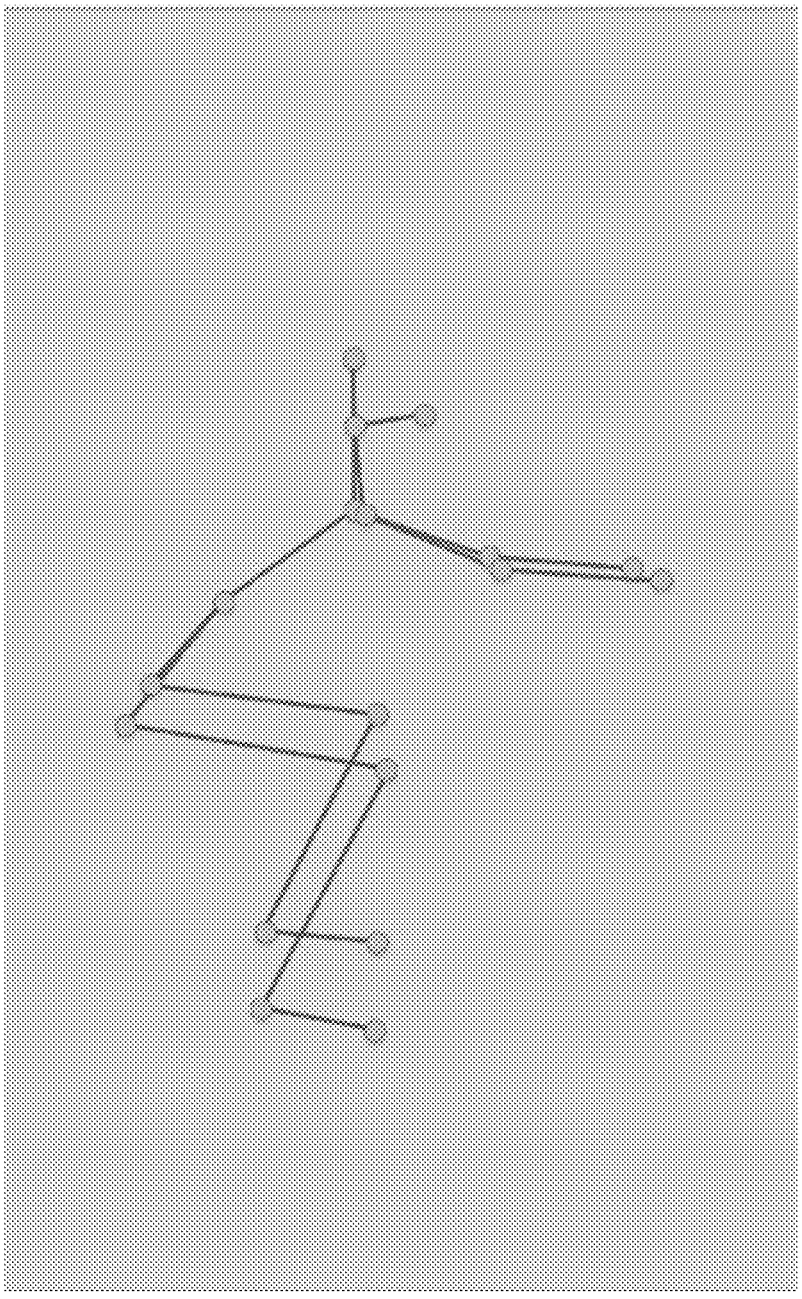


FIG. 5F

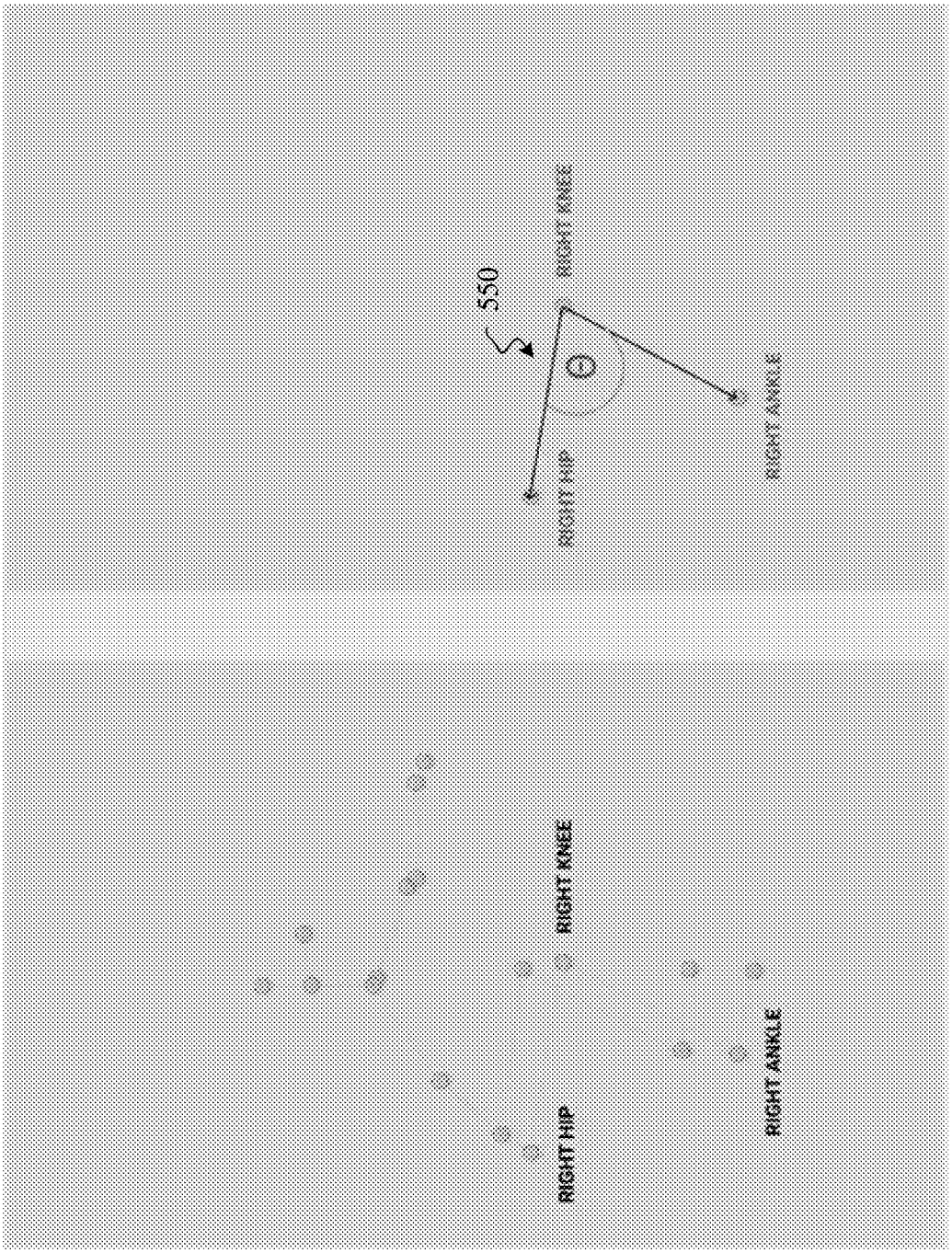

550
RIGHT KNEE
RIGHT HIP
RIGHT ANKLE
RIGHT HIP
RIGHT KNEE
RIGHT ANKLE

FIG. 5G

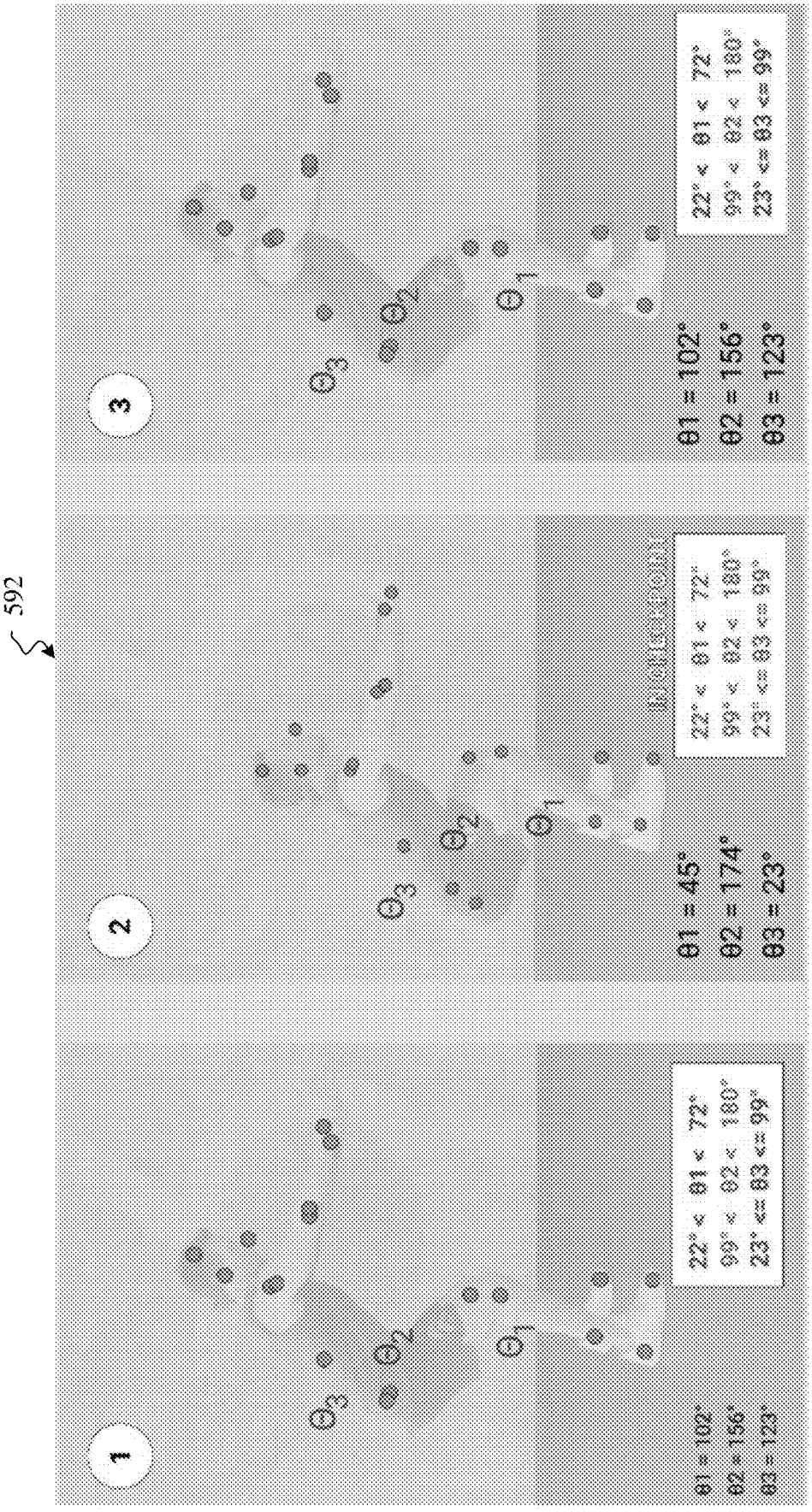

592
1
2
3
θ1 = 102°
θ2 = 156°
θ3 = 123°
22° < θ1 < 72°
99° < θ2 < 180°
23° <= θ3 <= 99°
θ1 = 45°
θ2 = 174°
θ3 = 23°
MODIFICATION

FIG. 5H

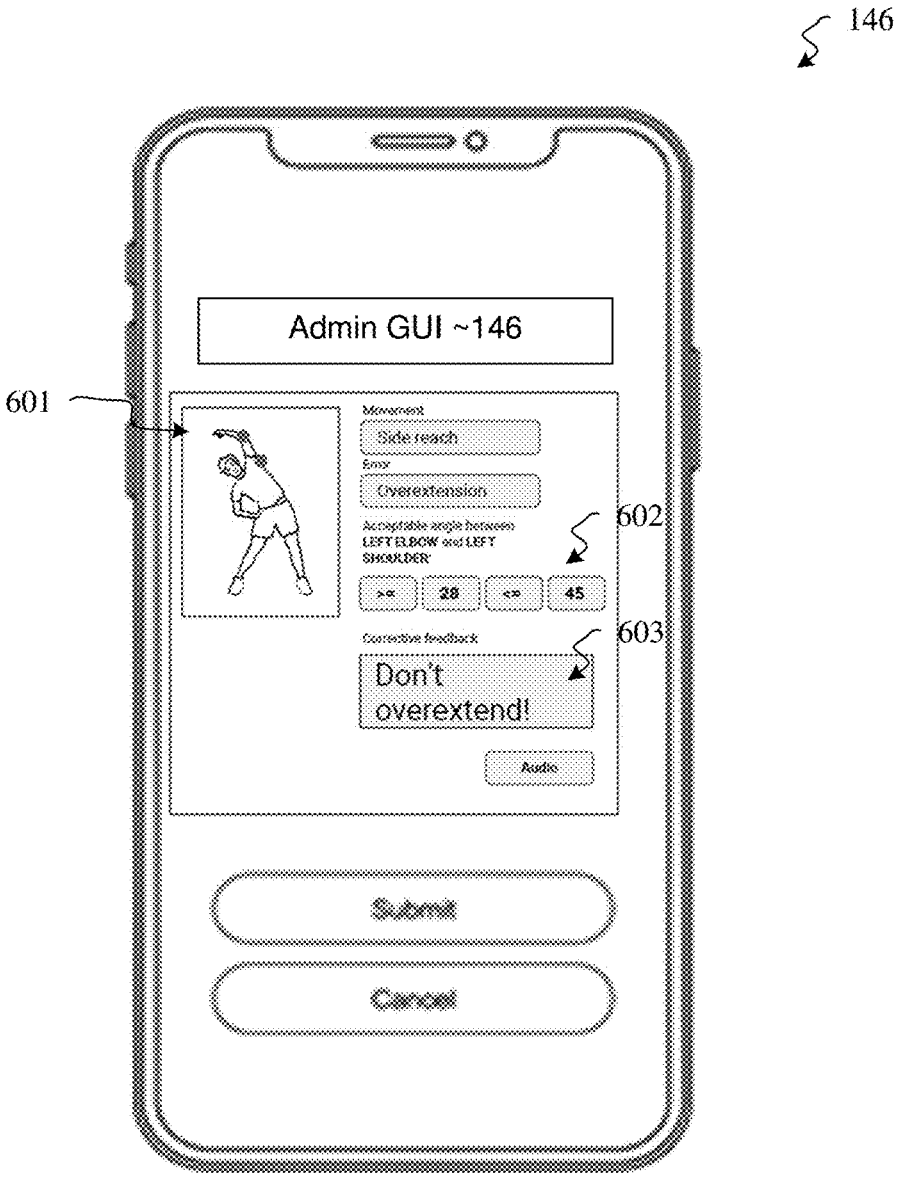

146
Admin GUI ~146
601
Movement
Side reach
Error
Overextension
602
Acceptable angle between
LEFT ELBOW and LEFT SHOULDER
>= 28
<= 45
603
Corrective feedback
Don't overextend!
Audio
Submit
Cancel

FIG. 6

700

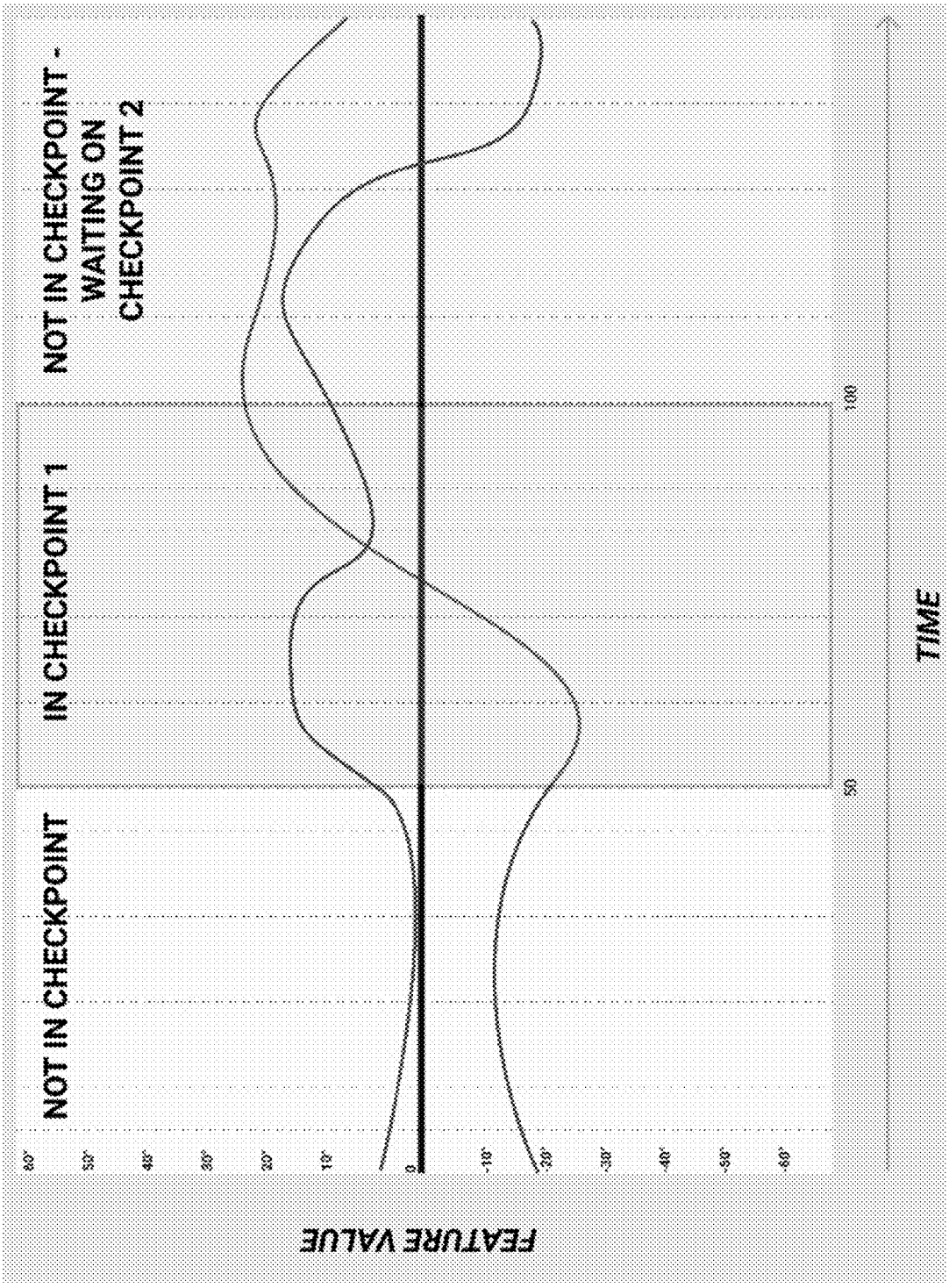

NOT IN CHECKPOINT
IN CHECKPOINT 1
NOT IN CHECKPOINT - WAITING ON CHECKPOINT 2
FEATURE VALUE
TIME
50
100

FIG. 7

800

Encoded Human Motion Reference

Motion Encoding Row	Positions in Sequence							
	θ1	θ2	θ3	.....		Audio Feedback	Danger	
Squat	102	156	123	.....	Outside checkpoint	Deeper!	0	0
Squat	45	174	23		Inside checkpoint	Good Job	0	0
Squat	45	174	20		Outside checkpoint	Watch your Knees!	!	0
Squat	102	156	123		Outside checkpoint	Head Up	0	0
	0	0	0				0	0
	0	1	0				0	0
	0	0	0				0	0
	0	0	0				0	0
	0	0	0				0	0
	1	0	1				0	0
	0	0	0				0	0
	0	0	0				0	0
	0	0	0				0	0
	0	0	0				0	0
	0	0	0				0	0
	0	0	0				1	0
	0	0	0				0	1
	0	0	0				0	0
	0	0	0				0	0
	0	0	0				0	0
				.....		.....		

FIG. 8

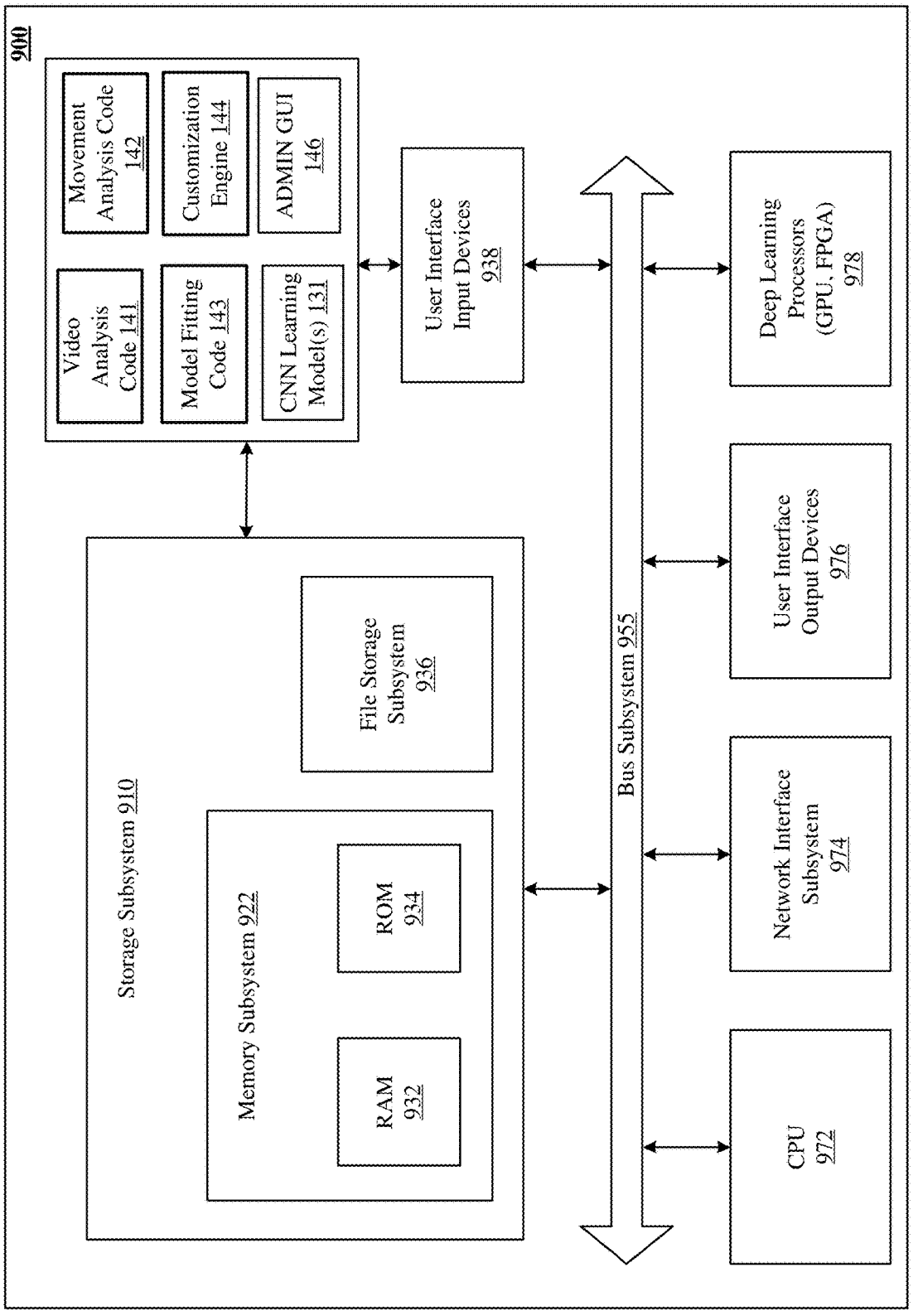

900
Video Analysis Code 141
Movement Analysis Code 142
Model Fitting Code 143
Customization Engine 144
CNN Learning Model(s) 131
ADMIN GUI 146
User Interface Input Devices 938
Deep Learning Processors (GPU, FPGA) 978
Storage Subsystem 910
File Storage Subsystem 936
Memory Subsystem 922
ROM 934
RAM 932
Bus Subsystem 955
User Interface Output Devices 976
Network Interface Subsystem 974
CPU 972

FIG. 9

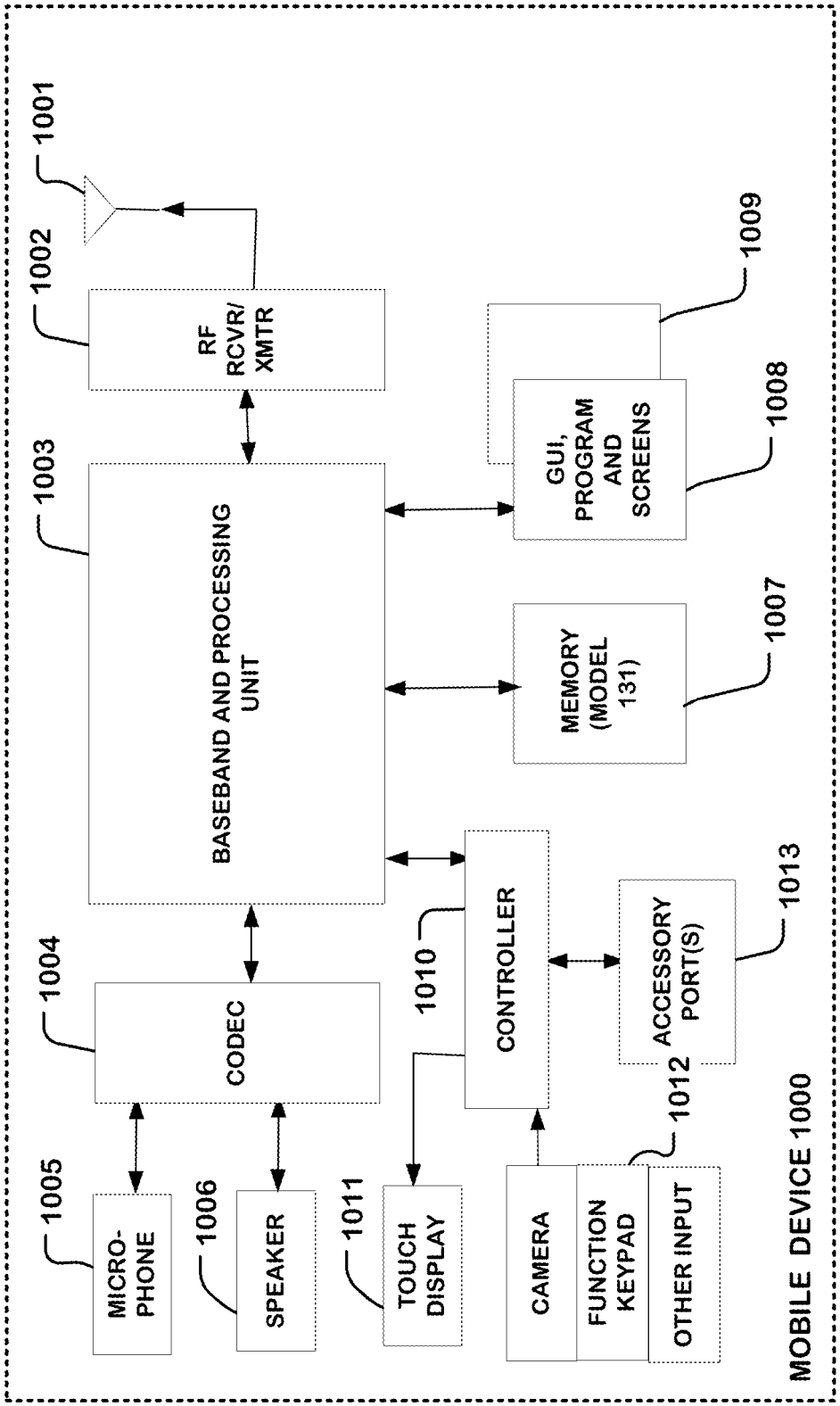

MOBILE DEVICE 1000
1001
1002
RF RCVR/ XMTR
1003
BASEBAND AND PROCESSING UNIT
1004
CODEC
1005
MICRO-PHONE
1006
SPEAKER
1007
MEMORY (MODEL 131)
1008
GUI, PROGRAM AND SCREENS
1009
1010
CONTROLLER
1011
TOUCH DISPLAY
1012
CAMERA
FUNCTION KEYPAD
OTHER INPUT
1013
ACCESSORY PORT(S)

FIG. 10

1100

Convolutional Neural Network

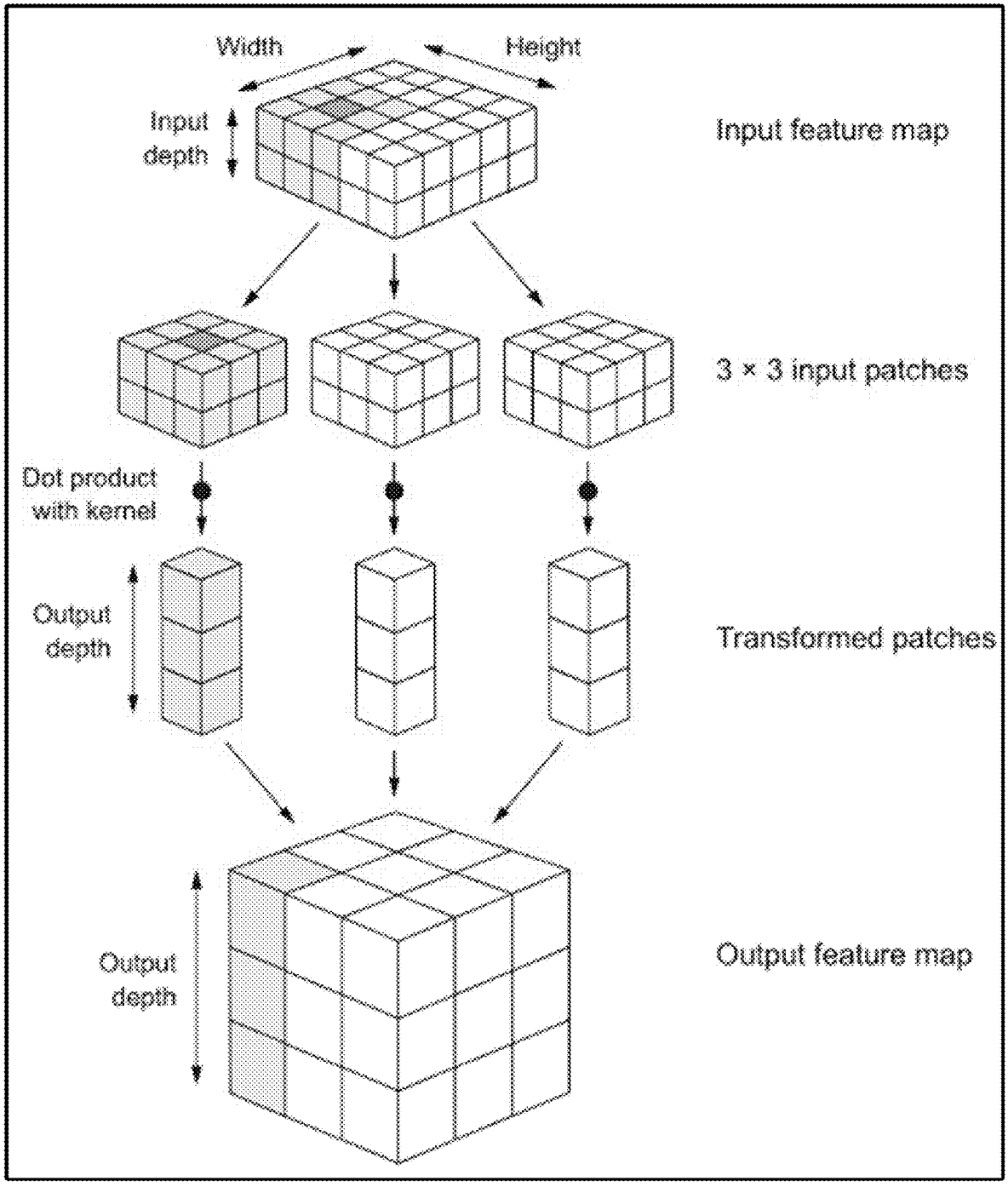


FIG. 11

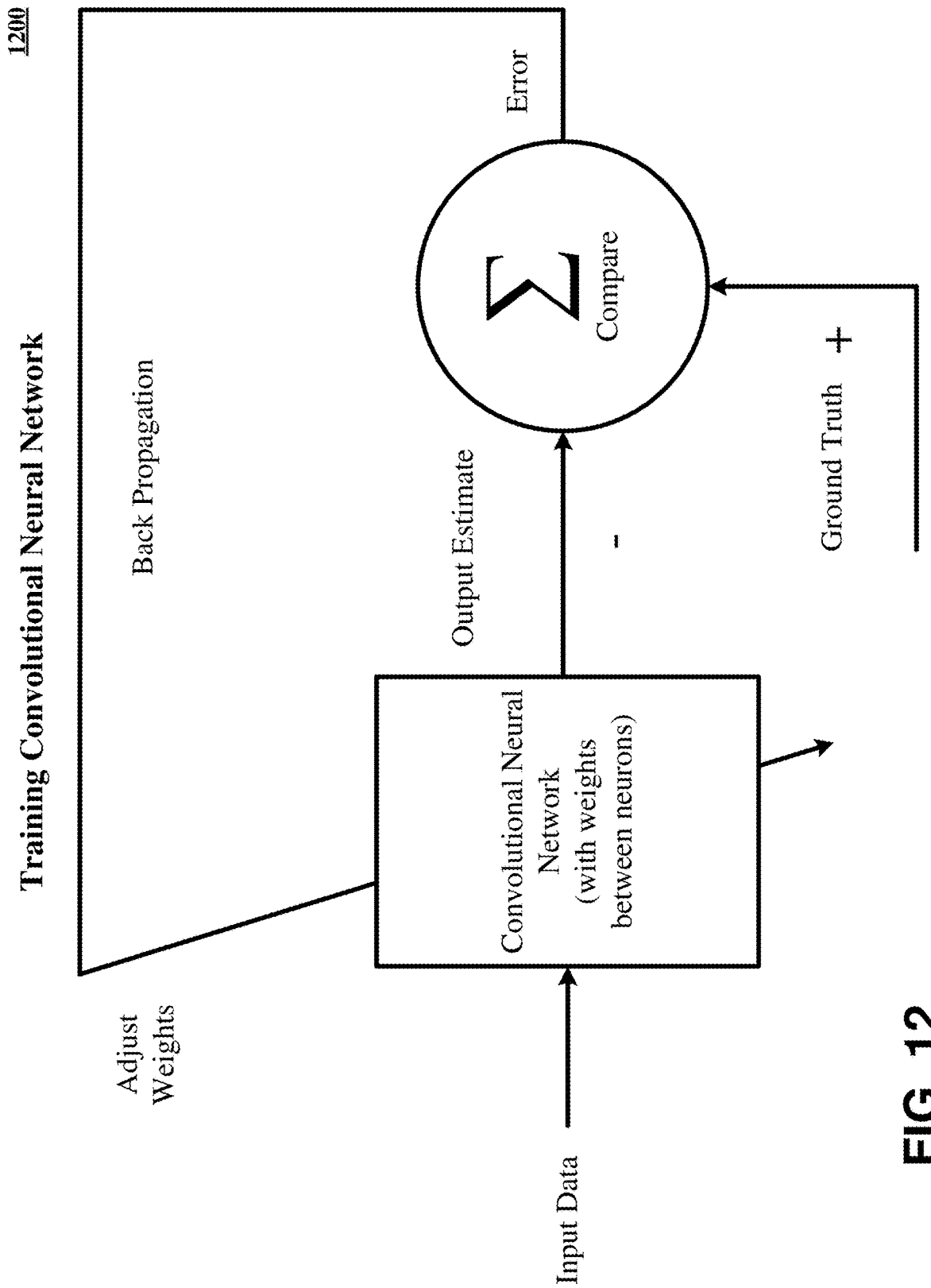

1200
Training Convolutional Neural Network
Back Propagation
Error
Compare
Output Estimate
-
+
Ground Truth
Convolutional Neural Network (with weights between neurons)
Adjust Weights
Input Data

FIG. 12

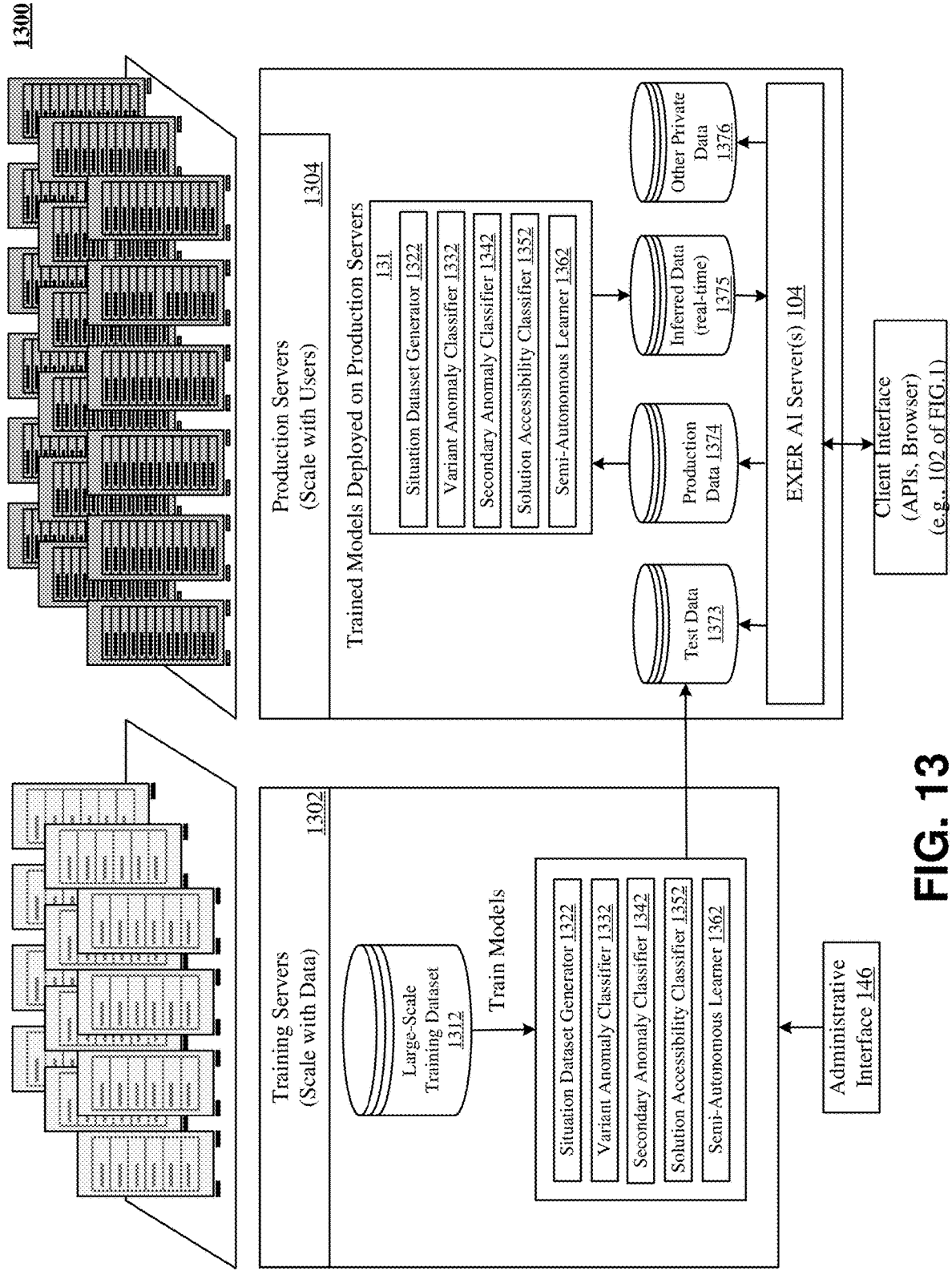

1300
Training Servers
(Scale with Data)
1302
Large-Scale
Training Dataset
1312
Train Models
Situation Dataset Generator 1322
Variant Anomaly Classifier 1332
Secondary Anomaly Classifier 1342
Solution Accessibility Classifier 1352
Semi-Autonomous Learner 1362
Administrative
Interface 146
Production Servers
(Scale with Users)
1304
Trained Models Deployed on Production Servers
131
Situation Dataset Generator 1322
Variant Anomaly Classifier 1332
Secondary Anomaly Classifier 1342
Solution Accessibility Classifier 1352
Semi-Autonomous Learner 1362
Test Data
1373
Production
Data 1374
Inferred Data
(real-time)
1375
Other Private
Data
1376
EXER AI Server(s) 104
Client Interface
(APIs, Browser)
(e.g., 102 of FIG.1)

FIG. 13

1402

	f1_upper	f1_lower	f2_upper	f2_lower	f3_upper	f3_lower	f4_upper	f4_lower	in_checkpoint
0	37.807529	31.513744	87.454028	85.994337	-5.369709	258.639425	22.843872	99.481821	0
1	33.462397	32.381499	88.199415	85.831451	-6.997473	249.676413	22.762868	99.312433	1
2	37.646549	31.146272	86.788342	85.809256	-7.607501	192.099128	22.594796	99.266220	0
3	34.736345	31.635518	86.866897	85.866690	-6.744744	274.284982	22.427212	99.599940	0
4	35.313985	31.985234	87.546953	85.847695	-4.405462	136.372367	22.509915	99.192137	0
...	...	...	...	...	...	...	...	...	...
995	35.035885	31.551618	88.773141	85.543874	-8.406590	150.837669	22.644453	99.866593	1
996	32.496273	32.204220	87.389772	85.751694	-10.824351	132.054682	22.845299	99.885844	0
997	33.917318	32.642749	86.859497	85.802385	-4.709685	260.912411	22.584492	99.261009	0
998	37.580747	32.314146	87.766427	85.618062	-6.957556	191.748314	22.828462	99.726120	0
999	34.041836	31.646079	87.228284	85.783186	-11.181092	162.192218	22.486865	99.286210	0

FIG. 14A

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33)
clf = make_pipeline(StandardScaler(), SVC())
clf.fit(X_train, y_train.ravel())
predicted = clf.predict(X_test)
accuracy_score(y_test, predicted)
```

1412

1414

0.7

FIG. 14B

```
parameters = [{'svc__kernel':('linear', 'rbf'), 'svc__gamma': ('scale', 'auto'), 'svc__C':[1, 10, 100]}]
```

1422

```
pipeline = make_pipeline(StandardScaler(), SVC())

clf = GridSearchCV(estimator=pipeline, param_grid=parameters,return_train_score=True, verbose=0)
clf.fit(X_train, y_train.ravel())
clf.get_params()

predicted = clf.predict(X_test)
accuracy_score(y_test, predicted)
```

1424

0.9848484848484849

FIG. 14C

CONTINUALLY LEARNING AUDIO FEEDBACK ENGINE

RELATED APPLICATIONS

[0001] This application claims priority under 35 U.S.C. § 119(e) to: (i) U.S. Provisional Patent Application No. 63/169,778, entitled "Continually Learning Audio Feedback Engine" filed on Apr. 1, 2021 (Attorney Docket No. EXER 1001-2) and (ii) U.S. Provisional Patent Application No. 63/169,777, entitled "Motion Engine" filed on Apr. 1, 2021 (Attorney Docket No. EXER 1000-2), which applications are incorporated herein in their entirety by reference for all purposes.

INCORPORATIONS

[0002] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in arXiv:1707.01083, 2017;

[0003] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications," in arXiv:1704.04861, 2017;

[0004] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in arXiv:1801.04381v3, 2018;

[0005] Z. Qin, Z. Zhang, X. Chen, and Y. Peng, "FD-MobileNet: Improved MobileNet with a Fast Downsampling Strategy," in arXiv:1802.03750, 2018;

[0006] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proc. of CVPR, 2016;

[0007] K. He, X. Zhang, S. Ren, and J. Sun, "DEEP RESIDUAL LEARNING FOR IMAGE RECOGNITION," arXiv:1512.03385, 2015;

[0008] J. Wu, "INTRODUCTION TO CONVOLUTIONAL NEURAL NETWORKS," Nanjing University, 2017;

[0009] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "CONVOLUTIONAL NETWORKS," Deep Learning, MIT Press, 2016;

[0010] F. Yu and V. Koltun, "MULTI-SCALE CONTEXT AGGREGATION BY DILATED CONVOLUTIONS," arXiv:1511.07122, 2016;

[0011] R. K. Srivastava, K. Greff, and J. Schmidhuber, "HIGHWAY NETWORKS," arXiv: 1505.00387, 2015;

[0012] G. Huang, Z. Liu, L. van der Maaten and K. Q. Weinberger, "DENSELY CONNECTED CONVOLUTIONAL NETWORKS," arXiv:1608.06993, 2017;

[0013] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "GOING DEEPER WITH CONVOLUTIONS," arXiv: 1409.4842, 2014;

[0014] S. Ioffe and C. Szegedy, "BATCH NORMALIZATION: ACCELERATING DEEP NETWORK TRAINING BY REDUCING INTERNAL COVARIATE SHIFT," arXiv: 1502.03167, 2015;

[0015] Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan, "DROPOUT: A SIMPLE WAY TO PREVENT NEURAL NETWORKS FROM OVERFITTING," The Journal of Machine Learning Research, 15 (1):1929-1958, 2014;

[0016] L. C. Piqueras, "AUTOREGRESSIVE MODEL BASED ON A DEEP CONVOLUTIONAL NEURAL NETWORK FOR AUDIO GENERATION," Tampere University of Technology, 2016;

[0017] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, and G. Wang, "RECENT ADVANCES IN CONVOLUTIONAL NEURAL NETWORKS," arXiv:1512.07108, 2017;

[0018] M. Lin, Q. Chen, and S. Yan, "Network in Network," in Proc. of ICLR, 2014;

[0019] L. Sifre, "Rigid-motion Scattering for Image Classification, Ph.D. thesis, 2014;

[0020] L. Sifre and S. Mallat, "Rotation, Scaling and Deformation Invariant Scattering for Texture Discrimination," in Proc. of CVPR, 2013;

[0021] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in Proc. of CVPR, 2017;

[0022] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated Residual Transformations for Deep Neural Networks," in Proc. of CVPR, 2017;

[0023] F. Chaubard, R. Mundra, and R. Socher, "CS 224D: DEEP LEARNING FOR NLP, LECTURE NOTES: PART I," 2015;

[0024] F. Chaubard, R. Mundra, and R. Socher, "CS 224D: DEEP LEARNING FOR NLP, LECTURE NOTES: PART II," 2015;

[0025] F. Chaubard, R. Mundra, and R. Socher, "CS 224D: DEEP LEARNING FOR NLP, LECTURE NOTES: PART III," 2015;

[0026] F. Chaubard, R. Mundra, and R. Socher, "CS 224D: DEEP LEARNING FOR NLP, LECTURE NOTES: PART IV," 2015;

[0027] F. Chaubard, R. Mundra, and R. Socher, "CS 224D: DEEP LEARNING FOR NLP, LECTURE NOTES: PART V," 2015;

[0028] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WAVENET: A GENERATIVE MODEL FOR RAW AUDIO," arXiv:1609.03499, 2016; and

[0029] S. Ö. Arik, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, A. Ng, J. Raiman, S. Sengupta and M. Shoeybi, "DEEP VOICE: REAL-TIME NEURAL TEXT-TO-SPEECH," arXiv:1702.07825, 2017.

REFERENCE TO COMPUTER PROGRAM LISTING APPENDIX

[0030] A Rules Engine Listing Appendix submitted electronically via EFS-Web in pdf format accompanies this application and is incorporated by reference. The name of the PDF file is "V7_Plank_App_Dialog_Engine_Rules_Appx," created on 9 Feb. 2021 and is forty-five (45) pages.

TECHNICAL FIELD

[0031] The technology disclosed relates to artificial intelligence type computers and digital data processing systems and corresponding data processing methods and products for emulation of intelligence (i.e., knowledge based systems, reasoning systems, and knowledge acquisition systems); and including systems for reasoning with uncertainty (e.g., fuzzy logic systems), adaptive systems, machine learning systems, and artificial neural networks. In particular, the technology

disclosed relates to using deep neural networks such as convolutional neural networks (CNNs) and fully-connected neural networks (FCNNs) for analyzing data and particularly to continually learning movement analysis in implementations implementing feedback.

BACKGROUND

[0032] The subject matter discussed in this section should not be assumed to be prior art merely as a result of its mention in this section. Similarly, a problem mentioned in this section or associated with the subject matter provided as background should not be assumed to have been previously recognized in the prior art. The subject matter in this section merely represents different approaches, which in and of themselves can also correspond to implementations of the claimed technology.

[0033] Motion capture technologies enable a variety of uses in motion picture animation, video game production and the like. However, diagnostic techniques based upon motion capture have been more elusive in realizing commercial practicality. The simple variability in correct performance of movements and indeed in actor's sizes and body types makes effectively evaluating motions captured in images problematic. Limitations on the processing power and capabilities of computing machinery configured to address these types of applications makes conventional approaches commercially impracticable.

[0034] Accordingly, an opportunity arises to introduce new methods and systems to evaluate motion captured from images and determine root causes of failure analysis during human execution of captured motions from images.

SUMMARY

[0035] The present technology provides systems, methods and computer program instructions implementing machine learning techniques to enable program processes to learn more effective feedback mechanisms to achieve desired results (e.g., reduce errors, improve form, duration, speed, and so forth) of motions and poses comprising tasks being taught or guided. In implementations an automated technology for automated creation of movement assessments from labeled video and continually learning audio, video or other feedback for use with machine learning techniques enable program processes to learn more effective feedback mechanisms to achieve desired results (e.g., reduce errors, improve form, duration, speed, and so forth) of motions and poses comprising tasks being taught or guided.

[0036] In an aspect of the present technology a method of conducting evaluation of collected facts about performance of a task and determining output instructions for a user performing the task is described that includes receiving state information comprising a set of collected facts describing a user pose state, including (i) at least one static fact that is constant over a time-period in which at least the task is performed and (ii) at least one dynamic fact based on an amount of time that has elapsed since a last error was detected. Upon lapse of a periodic timer, the method includes in a first process of evaluating facts in the set of collected facts as received to determine a response message to be output as instructions for performing the task, by weighting at least some facts with dynamic weightings; selecting based on the at least some facts as dynamically weighted, a response message to be output; and performing the output response message as selected, and capturing results for evaluation as historical outcomes. The method further includes in a second process, evaluating by a feedback engine historical outcomes, wherein an outcome is a consequence of a combination of facts evaluated and response message(s) played to the user, of a sample set of previous outcomes, thereby identifying a time between similar outcomes; and applying a machine learning process to the time between similar outcomes to obtain an improved selection of response messages to obtain similar outcomes exhibiting a desired result. The method further includes storing dynamic weights in a database to personalize task performance training to the user thereby bringing about desired outcome for that user.

[0037] In another aspect of the present technology, whenever the user is in an improper position as determined using information from a pose engine, state information received can further includes a label of error to present to the user wherein each label of error comprises one or more audio, video or other output-type files from which feedback is selected for output to the user while being assessed on a movement.

[0038] In further aspects, dynamic facts are selected from a set including at least (i) an amount of time that has elapsed since a last error was observed, (ii) a repetition or timestamp since an event or a timer started or a time since midnight selected from a number of times an error has been previously observed. Constant facts are quantities determinable at session start of the time period in which the task is to be performed selected from a set including at least (i) a length of a session in which tasks are to be performed, (ii) number of tasks to be performed, (iii) data collected and feedback given, and (iv) other quantities capable of being held constant during a session.

[0039] In a yet further aspect of the present technology, duration of the timer is variable and dependent on an action type of the task to be performed, in a range of 0.85 seconds corresponding to higher repletion rate actions to 5 minutes corresponding to lower repetition rate actions.

[0040] In a still further aspects of the present technology, selecting feedback can be probabilistic. Weight(s) can be applied to one or more dynamic features such that output of the fact will be true if (i) the fact is true, and then (ii) if an evenly distributed random variable with value in a range of 0 to 1 is greater than or equal to the weight, thereby enabling varying feedback selected for a particular fact. At the beginning of a session setting dynamic weights are to random values if there is no historical values for the weight values. Determining an appropriate response message type using the weighted dynamic features, including: (i) none when none exists, (ii) selecting an output message based upon output type in a set of at least audio message, visual message, and (iii) when for each message type, there exist multiple variants of recorded responses from which to choose selecting based upon message type in a set of at least specific message—too high, specific message—encouragement, specific message—warning, and specific message—termination message. Output can be automatically selected audio output response message having the response message type as determined. Initially, an audio output can be chosen at random and playing the audio output as chosen to the user and storing a time at which the audio was played and a corresponding message for future reference.

[0041] In a still yet further aspect of the present technology, applying a machine learning process can include evaluating time between two errors reported to the user by fitting a curve to data points representing previous results for the user; and applying association rules and linear regression to maximize time/reps between errors, using gradient descent, variable times, max time between consecutive errors to provide coaching to a user.

[0042] In a still yet further aspect of the present technology, historical data from a plurality (e.g., 10 or more) of previous sessions can be used to adjust the dynamic weights; and the dynamic weights as adjusted can be stored to be used in subsequent executions of the method. Dynamic weights can be stored on a device of the user, when protecting user privacy is a concern. For example, a probability of a particular type of response to a message is used to weigh future selections of responses. The probability of a particular type of response can be one of a set comprising: an error every time, a person responds to the message with a successful outcome, a person does not respond well to the message or the message appears to have no effect on the person's performance of the task. The probability of a particular type of response is one of a set task related facts comprising: resistance, repetitions, format (e.g., Tabata, as many repetitions as possible, drop-set, superset, etc.) or other factors characterizing performance of the task.

[0043] In an aspect of the present technology a method of performing video analysis includes capturing a live-stream or recorded video in color or B&W format. Scaling frames to appropriate size for model input is also part of the method. The method also includes extracting a plurality of keypoints that describe areas of interest on the body (e.g., a location of a keypoint corresponding to the user's head is found to be at (x: 200, y: 300)). Implementations of the method also call for smoothing the current pose using an appropriate digital signal processing (DSP) functionality to prevent perceived jumpiness in the video image when displayed, especially when using a mobile or small footprint device. Given a current exercise being taught, the method further calls for extracting features from the relevant assessment rules, and comparing the current set of keypoints as extracted to features extracted from the relevant assessment rules to determine a current state of the user's pose. Given the determined state, the method includes identifying a message (or NULL message) to present or play for the user.

[0044] In another aspect of the present technology, facts for determining correctness of task performance are gathered by a server and automatically labeled using machine learning processes.

[0045] A system including one or more processors and memory accessible by the processors is also described. The memory can be loaded with computer instructions which can be executed on the processors. The computer instructions when executed on the processors can implement the method for self-disinfecting a touch surface of a touch sensitive display device. Computer program products which can be executed by computer systems are also described herein.

[0046] Other aspects and advantages of the present technology can be seen on review of the drawings, the detailed description and the claims, which follow.

BRIEF DESCRIPTION OF THE DRAWINGS

[0047] The patent or application file contains at least one drawing executed in color. Copies of this patent or patent application publication with color drawing(s) will be provided by the Office upon request and payment of the necessary fee.

[0048] The disclosure will be understood more fully from the detailed description given below and from the accompanying figures of embodiments of the disclosure. The figures are used to provide knowledge and understanding of embodiments of the disclosure and do not limit the scope of the disclosure to these specific embodiments. Furthermore, the figures are not necessarily drawn to scale.

[0049] FIG. **1** illustrates a system **100**A that implements content management, analysis and delivery functionality using machine learning techniques.

[0050] FIG. **2**A illustrates a flowchart depicting an example process for performing video analysis as may be embodied by video analysis code **141** in a representative implementation.

[0051] FIG. **2**B illustrates a flowchart depicting an example process for performing movement analysis as may be embodied by movement analysis code **142** in a representative implementation.

[0052] FIG. **2**C illustrates a flowchart depicting an example process for performing model fitting as may be embodied by model fitting code **143** in a representative implementation.

[0053] FIG. **2**D illustrates a flowchart depicting an example process for performing customization as may be embodied by customization engine **144** in a representative implementation.

[0054] FIG. **3**A illustrates a flowchart depicting an example processing conducted by edge application **120** for performing training using feedback and rule based motion analysis gathered from a user session in a representative implementation.

[0055] FIG. **3**B illustrates a flowchart depicting an example process for performing rule-based feedback of motion analysis as may be embodied by feedback engine code **125** in a representative implementation.

[0056] FIG. **3**C illustrates a flowchart depicting an example process for performing continual training of a machine learning automaton based upon feedback and rule-based motion analysis gathered from a user session in a representative implementation.

[0057] FIG. **4** illustrates a typical payload exchanged between an AI server implementation and an AI client implementation in an embodiment.

[0058] FIG. **5**A illustrates an example raw image frame in an example application of the disclosed technology to conduct training using feedback and rule-based motion analysis gathered from a user session in a representative implementation.

[0059] FIG. **5**B illustrates an example raw model output in an example application of the disclosed technology to conduct training using feedback and rule-based motion analysis gathered from a user session in a representative implementation.

[0060] FIG. **5**C illustrates an example annotated image output in an example application of the disclosed technology to conduct training using feedback and rule-based motion analysis gathered from a user session in a representative implementation.

[0061] FIG. **5**D illustrates an example of translating key points to body joints in an example application of the disclosed technology to conduct training using feedback and

rule-based motion analysis gathered from a user session in a representative implementation.

[0062] FIG. **5**E illustrates an example labelling in an example application of the disclosed technology to conduct training using feedback and rule-based motion analysis gathered from a user session in a representative implementation.

[0063] FIG. **5**F illustrates an example keypoints in a pose in an example application of the disclosed technology to conduct training using feedback and rule-based motion analysis gathered from a user session in a representative implementation.

[0064] FIG. **5**G illustrates an example feature definition in an example application of the disclosed technology to conduct training using feedback and rule-based motion analysis gathered from a user session in a representative implementation.

[0065] FIG. **5**H illustrates an example checkpoints definition in an example application of the disclosed technology to conduct training using feedback and rule-based motion analysis gathered from a user session in a representative implementation.

[0066] FIG. **6** illustrates an example graphical user interface (GUI) implemented in some embodiments.

[0067] FIG. **7** illustrates assessing poses over time of checkpoints in an example application of the disclosed technology to conduct training using feedback and rule-based motion analysis gathered from a user session in a representative implementation.

[0068] FIG. **8** illustrates a labelling of an exemplary input training set for training a deep neural network implementation.

[0069] FIG. **9** illustrates one implementation of a computer system **900** that can be used to implement the technology disclosed.

[0070] FIG. **10** is a simplified diagram of a mobile phone computing platform **1000**, representative of computing devices which can be used as an apparatus for edge application **120** described herein.

[0071] FIG. **11** illustrates an implementation of a convolutional neural network suitable for implementing the disclosed technology.

[0072] FIG. **12** depicts a block diagram of training a convolutional neural network in accordance with one implementation of the technology disclosed.

[0073] FIG. **13** illustrates a deep learning system in a supervised or semi-supervised implementation.

[0074] FIG. **14**A illustrates an example iterative machine learning process such as a grid search in a representative implementation.

[0075] FIG. **14**B shows use of a simple example **1412** of a C-Support Vector Classification (SVC) provided by the machine learning toolkit.

[0076] FIG. **14**C shows performing a Grid Search **1422** to determine new hyperparameters for the given model (in this case an SVC).

DETAILED DESCRIPTION

[0077] Aspects of the present disclosure relate to automated creation of movement assessments from labeled video and continually learning audio feedback.

[0078] Our technology addresses the problem of improving the efficiency of adding movements (e.g., exercises) for evaluation by an automated system product. Applications could include any type of specific movement (e.g., choreographed dance movements, High Intensity Interval Training (HIIT) movements, martial arts, and so forth), physical therapy applications, and kinematic uses such as gait analysis.

[0079] Each user responds to feedback differently. For example, some enjoy constant praise or corrections, while other prefer silence. Our technology enables adaptively self-configuring by the computer system to meet the needs of an individual user while still achieving a goal. Machine learning techniques can be applied to various situations where messages are sent to attempt to change outcome. Implementations include portable devices or systems in teaching or training applications.

[0080] Our technology calls for inputs of videos recorded of individuals performing movements in proper and incorrect states. These videos are labeled on a per frame basis with labels including "proper", as well as error states (e.g., "elbow out too far"). The videos are processed using a convolutional neural network trained for human pose estimation that identifies the coordinates of keypoints (e.g., "head", "right elbow") in each frame of the input videos. For each frame's identified keypoints, a collection of body features is evaluated. A body feature is a description of one or more body keypoints in relation to another keypoint or a fixed point in space.

[0081] Using the feature outputs, a pattern detection algorithm is executed on each of the video outputs to identify patterns of motion using the coordinates of each identified body part, as well as the ranges that determine a successful repetition of the exercise. Using these ranges as an initial starting point as well as the initial video outputs and other "test videos", the system then performs a statistical fitting operation to ensure that each frame of a video is labeled correctly. Once the automated process has completed, a user (coach, trainer, professional) can "tweak" the ranges subjectively to better match how they tend to coach someone who is performing the movement.

[0082] The output of the system is a payload that describes each movement's proper and error states, as well as additional boundaries as customized by the user's coach. Additionally, the payload can include error details such as coaching instructions to be displayed when an error is detected. This output is then utilized by an edge device (e.g., phone, computer, fitness hardware, etc) that processes a live video feed with a similar human pose estimation Convolutional Neural Network (CNN). The edge device runs software that interprets the same collection of body features and compares each frame's features against the payload to determine the current state of the user in relation to the expected movement.

[0083] Given the collection of current facts, a feedback engine determines an appropriate response message type, or none if none exists. For each message type, there exist many variants of audio recordings from which the system can choose. Initially, an audio output is chosen at random and is played to the user and the time at which the audio was played and the corresponding message is stored for future reference.

[0084] After evaluating facts, the Feedback Engine evaluates historical outcomes. An outcome is a consequence of the combination of facts evaluated and the message played to the user. By evaluating all previous outcomes, the time between similar outcomes can be identified. For instance, the time between two errors reported to the user can be

evaluated. The system can then use machine learning approaches such as association rules and linear regression to adjust the dynamic weights of certain facts that are predetermined to affect an outcome. These adjusted weights are then used in subsequent executions of the engine and the process continues.

[0085] When a feedback session ends, the dynamic weights are stored and can be used again for subsequent sessions of the same assessment or can be used as initial values for similar facts in other assessments.

[0086] Some implementations can reduce the processing time of analyzing video of captured motion from weeks to days. Some implementations can improve the usability and performance over the current state of the art where all messages are just played to users.

System Overview

[0087] The technology disclosed describes system and method implementations of human motion analysis using deep learning-based approaches to identify and isolate anomalies and to identify and trigger appropriate remedial actions including selection and initiation of instructional content representations. FIG. **1** shows an architectural level schematic of a system in accordance with an implementation. Because FIG. **1** is an architectural diagram, certain details are intentionally omitted to improve the clarity of the description.

[0088] The discussion of FIG. **1** will be organized as follows. First, the elements of the figure will be described, followed by their interconnections. Then, the use of the elements in the system will be described in greater detail.

[0089] FIG. **1** includes the system **100**A that implements content management, analysis and delivery functionality using machine learning techniques. The system **100**A includes one or more AI server(s) **104**, disposed to process, manage and distribute instructional content to users at AI client devices **102**. Content can be provided by content creators **110** and stored in a video library **108**. Once analyzed, content can be stored in a pose information local store **106**. A deep learning system **112** can be used to train one or more neural networks or other learning model(s) **114**, such as for example a convolutional neural network (CNN) that can be trained using techniques described herein to recognize human poses and movements, and Internet and/or other electronic communications network(s) **101**.

[0090] The interconnection of the elements of system **100**A will now be described. Network(s) **101** couples the client(s) **102**, the one or more servers **104**, the pose information local store **106**, with the other servers and other entities that comprise a content delivery network **100**A, e.g., the other client devices, video library **108**, content creator's server(s) **110** accessing private collections data stored for each organization, the deep learning system **112**, the learning model(s) **114**, and other devices not shown in FIG. **1** for clarity sake, that can be in communication with each other (indicated by solid double-arrowed lines). The actual communication path can be point-to-point over public and/or private networks comprising network(s) **101**. The communications can occur over a variety of networks, e.g., private networks, VPN, MPLS circuit, or Internet, and can use appropriate application programming interfaces (APIs) and data interchange formats, e.g., Representational State Transfer (REST), JavaScript Object Notation (JSON), Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Java Message Service (JMS), and/or Java Platform Module System. At least some of the communications can be encrypted. The communication is generally over a network such as the LAN (local area network), WAN (wide area network), telephone network (Public Switched Telephone Network (PSTN), Session Initiation Protocol (SIP), wireless network, point-to-point network, star network, token ring network, hub network, Internet, inclusive of the mobile Internet, via protocols such as EDGE, 3G, 4G LTE, Wi-Fi, and WiMAX. Additionally, a variety of authorization and authentication techniques, such as username/password, Open Authorization (OAuth), Kerberos, SecureID, digital certificates and more, can be used to secure the communications. The engines or system components of FIG. **1**, such as AI client(s) **102**, AI server(s) **104**, deep learning system **112**, content creators' server(s) **110**, and other server(s) not shown in FIG. **1** for clarity sake are implemented by software running on varying types of computing devices. Example devices are a workstation, a server, a computing cluster, a blade server, and a server farm.

[0091] AI Server(s) **104** implement an analysis pipeline **140** that takes as input video information, such as may be provided by content creators using content creator server(s) **110** and stored in video library **108**, and provides as output a payload **90** comprising a manifest, assessment rules, and some introductory weights that will be used when determining what audio to play to client devices such as AI client **102**. In a minimal implementation, these weights can be set to a trivial starting point (e.g., 1.0— meaning that the all audio should play when the system detects an error.) Payload **90** will also include source locations for associated audio to play to the user. This will be a mapping of error identifiers (e.g., ERROR-001 represents the state "head too high" for a plank) to a remote storage URL (e.g., AWS S3) for the audio file to play when we encounter that error. It is noteworthy that portions of payload **90** can be modified by client **102** and the modified values returned to server **104**. For example, dynamic weightings in the neural network sent to the device **102** can be modified by the device **102** during usage of the neural network. These modified dynamic weightings can be sent back to the server **104**, enabling the system to continue the learning process based upon collective learning during use by multiple clients in the field. Analysis pipeline **140** includes a video analysis code **141**, movement analysis code **142**, a model fitting code **143**, and customization engine **144**.

[0092] Video analysis code **141** takes as input video, such as provided by content creators using content creator server (s) **110** and stored in video library **108**, for example. Input videos are accompanied by a manifest that describes each video as well as relevant timeframes in the video, such as the start and end of a repetition, relevant keypoints that should be examined during an exercise, a "working side" indicator that says what side of the body should be examined, and the number of peak checkpoints per repetition—which is typically 2—initial and peak. The video analysis code begins by extracting relevant parts of each video while maintaining the appropriate labels (e.g. "shoulder rotation", "improper form", "repetition start", and others). The videos can be fed a frame at a time into a pose estimation convolutional neural network (CNN) **131**, which can be drawn from one of the deep learning models **114**, that outputs a Pose (coordinates of X number of keypoints in the frame) and Confidences (the confidence of each identified keypoint). The outputs of the

video analysis code stage are labeled payloads of Poses and Confidences, as well as the initial manifest describing the overall movement.

[0093] Movement analysis code **142** examines the manifest to determine a candidate list of body features to use to assess the movement. Features involving the relevant keypoints (indicated in the manifest) are automatically selected based upon a data structure associating keypoints to appropriate features. For example, if "head" is a relevant keypoint, features like "head->neck distance" and "head->shoulder angle" are selected for use in the initial analysis. Keypoint-feature data structure implementations include a linked list, a doubly linked list, search trees, tables, and the like. For each Pose and Confidence payload, the relevant features are extracted. Using the manifests, checkpoints are extracted across each input video. These lists are examined to determine relevant ranges of values for each feature. This results in a list like "checkpoint_initial", "checkpoint_repetition", and "checkpoint_[error]_state" for each video. These recommendations are output to the model fitter cod **143** along with the manifest and Poses and Confidences for each video.

[0094] Model fitting code **143** performs feature extraction upon the Poses and Confidences for an entire video. At each labeled checkpoint, the recommendations are compared to determine if they "see" the checkpoint properly, e.g., if the system is able to adequately assess the movement by evaluating the features (commonly the angle between certain keypoints) and determining if the evaluated ranges fall within those that the system learned during the previous stage, the system "sees" the checkpoint. Practically this could mean that we have trained the system to look for an elbow angle in the range of 30-36 degrees and if during this stage we see an angle of 34 degrees we can say that the system properly "sees" the checkpoint. If all checkpoints were identified correctly, the model fitting processing stage is complete. If checkpoints were mislabeled, an iterative machine learning process such as a grid search such as described below in greater detail with reference to FIG. **14**A-FIG. **14**C is performed to slightly adjust values for features until all checkpoints are "seen" properly. Once the model fitting is complete, the "base" assessment for the exercise is stored in a database, e.g., pose information local store **106**.

[0095] Customization engine **144** enables a coach to adjust determined features in the web GUI **146** to his or her liking. The customization engine **144** will determine the difference between the base values e.g., received from the model fitting **143** and the coach's version of the assessment, as prompted for by the customization engine **144** via the admin GUI **146** and/or via API **145**. The labeled video Poses and Confidences are then ran through a feature extractor and compared against the new values. If a range is determined to no longer identify movement checkpoints, and error is reported to the coach. The coach can then re-adjust the values and try again. If it's found that all ranges can still properly identify all labeled checkpoints, the customization stage is finished. A "coach" assessment is stored in a coach assessment local store **107** database for future use.

[0096] AI Client(s) **102** implement an edge application **120** that takes as input manifest comprising assessment rules **126**, as may be provided by AI Server(s) **104**, and implements a continually learning feedback session for the user of the AI client **102**. Edge application **120** includes an application video feed code **121**, pose estimation code **122**, a feature extractor code **123**, a rules evaluator **124**, and a feedback engine **125**.

[0097] Application video feed code **121** implements capture of live-stream or recorded video in color or B&W format. While embodiments can be realized in virtually any size video, one implementation presently in use employs video in 1920×1080 (landscape) and 1080×1920 (portrait) sizes), however other sizes are appropriate to various applications. Frames are scaled to appropriate size for model input, such as for example a currently implementation resizes to 224×416 (portrait), 416×224 (landscape), however other sizes are appropriate to various applications.

[0098] Pose estimation code **122** extracts X number of keypoints that describe areas of interest on the body. For example: in one implementation, the location of a keypoint corresponding to the user's head is found to be at (x: 200, y: 300). Implementations can employ either 3D or 2D images.

[0099] Feature extractor code **123** smooths the current pose using an appropriate digital signal processing (DSP) functionality to prevent perceived jumpiness in the video image when displayed, especially when using a mobile or small footprint device. One implementation uses Kalman filters to remove jumpiness (see e.g., //en.wikipedia.org/wiki/Kalman_filter; which is incorporated herein by reference in its entirety for all purposes). Another implementation uses 1 € ("One Euro") Filters as described in the paper here: //dl.acm.org/doi/10.1145/2207676.2208639?cid=81100168597 which is incorporated herein by reference in its entirety for all purposes. Given the current exercise being taught, extract features from the relevant assessment rules **126**.

[0100] Rules evaluator **124** implements given the current set of extracted features, comparing them to the assessment rules **126**. Thus, the rules evaluator **124** is able to determine a current state of the user's pose.

[0101] Feedback engine **125** implements given the determined state, identify a message (or NULL message) to present or play for the user.

[0102] In one implementation, learning model(s) **114** implement multi-layer ensembles of neural subnetworks includes a first anomaly subnetwork, and a second solution accessibility subnetwork. The learning model(s) **114** are further configured to classify inputs indicating various anomalous sensed conditions into probabilistic anomalies using a first anomaly subnetwork. Determined probabilistic anomalies may be classified into remedial application triggers. Remedial application triggers are invoked to recommend or take actions to remediate, and/or report the anomaly. One implementation the learning model(s) **114** can select a feedback type, such as an audio feedback object, video feedback object, haptic feedback object, or other feedback object to submit based upon the situation state. For example within the exercise field, learning model(s) **114** can select whether to play an encouraging audio message responsive to a decision that the user is fatigues, or vibrate slightly to indicate a motion going awry, or the like. One implementation can select a report recipient based upon the situation state. For example within the exercise field, learning model(s) **114** can address reporting to a coach, team coach, personal trainer, physiotherapist, doctor, nurse practitioner, nurse, or other medical professional, or other third party.

[0103] The deep learning system **112** trains some of the learning model(s) **114** implementing neural networks in semi-supervised modalities to recognize anomalies in motions and trigger remedial actions. Further implementations enable learning from gathered results of remedial actions and adjusting the dynamic weightings in the neural network to continue the learning process during use in the field. In one implementation, neural networks are trained on one or more training servers (e.g., **1302** of FIG. **13**) using training datasets (e.g., **1312** of FIG. **13**) and deployed on one or more production servers (e.g., **1304** of FIG. **13**).

[0104] The API **145** is used for communication between the client and server. Each request to the API includes a version number, which represents the version of payload that the client supports. When an assessment from the from the API is requested an Assessment Rules payload. Requests to the API can include the ability to report back range minima and maxima as seen during live processing. This data can be used to further train the model and update assessment rules.

[0105] Having reviewed FIG. **1** illustrating an example system for delivering content capable of interacting with and learning from a user, we now look at the processes conducted by the AI server **104** in more detail with reference to FIGS. **2**A, **2**B, **2**C and **2**D.

Video Analysis

[0106] FIG. **2**A illustrates a flowchart depicting an example process for performing video analysis as may be embodied by video analysis code **141** in a representative implementation.

[0107] In block **201** process for obtaining a manifest and corresponding recorded video of individuals performing particular movements, such as a plank or the like in proper states (correct form) and in improper states (incorrect form) is illustrated. Videos are created and labeled with ground truth (e.g., correct form, incorrect form, etc.) labelling or some implementations convert/leverage/use videos found in the public domain (e.g., world wide web for example). The manifest describes frames of each video as being at least one of (i) proper, reflecting that an individual is in a proper state, and (ii) improper, reflecting that an individual is in an improper state. The manifest may further describe the frames of each video as (i) being a start of a repetition, (ii) being an end of a repetition, (iii) including specific evaluation points (head, shoulder elbow, etc.) to be evaluated and (iv) including a working side to be evaluated, wherein the manifest identifies a number of peak checkpoints to be evaluated per repetition (e.g., initial checkpoint and peak checkpoint (e.g., bottom of the squat)). As used here, the terms evaluation/keypoint are used to denote interesting features, e.g., elbow, wrist and coordinates thereof. As used herein the term checkpoint refers to a collection of keypoints in a known state, e.g., in a squat, bottom of squat, certain angles of knees, shoulders, waist, etc., as the body moves through a series of checkpoints throughout a repetition. In an implementation, automation can be included, where the computer determines some/all of the data in the manifest. In other implementations, a human's input is solicited, collected, or both. For example, a manifest could say from 2-6 seconds the user's hips are too high.

[0108] In block **202** extracting portions of the videos for evaluation, while maintaining the descriptions (in the manifest) of the frames of the extracted portions of the videos.

[0109] In block **203** inputting, into a pose estimation neural network, the extracted portions of the videos one frame at a time.

[0110] In block **204** receiving, as an output of the pose estimation neural network and for each input frame, a pose (collection of the keypoints in the frame), including coordinates of one or more evaluation/key points in the frame and confidences representing a confidence that each keypoint is a particular feature (e.g., 60% sure that a particular keypoint is an elbow) of each of the one or more evaluation points.

[0111] In block **205** outputting labeled payloads of poses and confidences for each frame of the extracted portions of the videos (e.g., this frame includes a pose of someone leaning over, poses of the keypoints and the confidences of the keypoints, then aggregate of confidences over all keypoints for a particular repetition). Confidences can also be used to weight some joints (keypoints) (e.g., confidences of elbow can permeate through labeling of other keypoints—across collections of frames). In implementations, if "labels" in the manifest could be wrong, computer implemented technology can reconcile and validate (question) the labels that were determined; thereby resulting in slices of videos comprising information about whether correct/incorrect position/state, and keypoint and confidence information).

Movement Analysis:

[0112] FIG. **2**B illustrates a flowchart depicting an example process for performing movement analysis as may be embodied by movement analysis code **142** in a representative implementation.

[0113] In block **211** identifying (selecting) a particular movement (e.g., plank).

[0114] In block **212** identifying a video associated with movement, as well as corresponding manifest.

[0115] In block **213** examining the corresponding manifest to determine a candidate list of body features (e.g., a feature is angle between shoulder and elbow—determined using keypoints) to use to evaluate for the particular movement. The feature is derived from the keypoints and relationships therebetween e.g., angles.

[0116] In block **214** automatically selecting, from the manifest, body features (neck length, shoulder angle) that are in the candidate list and related to the keypoints.

[0117] In block **215** for each Pose and Confidence payload, the relevant body features are extracted (body feature is now know/determined from above, using pose and confidence). Using the manifests, checkpoints are extracted across each input video. These lists are examined to determine relevant ranges of values for each identified body features (hip to head angle is xyz) [result is a list like "checkpoint_initial", "checkpoint_repetition" and "checkpoint_[error—hips too high]_state" for each video.

[0118] In block **216** providing recommendations [ranges for particular body features (collection of ranges from minimum to maximum that are essentially acceptable for a particular movement (plank)) for model fitting along with the manifest, poses and confidences for each video. This is still the learning phase, the system is still learning what is acceptable and not acceptable.

Model Fitting:

[0119] FIG. **2**C illustrates a flowchart depicting an example process for performing model fitting as may be embodied by model fitting code **143** in a representative implementation. The goal is to determine which keypoints and/or body are relevant to determining whether (e.g., plank) posture is correct. (This can remove keypoints/body features from analysis and focus on the relevant ones).

[0120] In block **221** performing body feature extraction for an entire video using the poses and confidences obtained for the entire video, such that at each labeled checkpoint of the video, the recommendations are compared to determine if they "see" the checkpoint properly (e.g., the checkpoint is properly modelled). Up to this point, we have all of the data we need to make our first guess (or estimate); for each frame the present technology enables a computer to "guess" whether the move was proper/improper.

[0121] In block **222** if all checkpoints are identified correctly, i.e., corresponds with the "guess" of the computer as to being proper or improper, then this stage is complete.

[0122] In block **223** if checkpoints were mislabeled [i.e., the computer "guessed" wrong], then an iterative machine learning process, such as a grid search, is performed to slightly adjust ranges of body features until all checkpoints are "seen" properly. (This is so our "guess" matches what the manifest states).

[0123] In block **224** and **225** after a range is changed, then this is reran to see if the change improved or degraded the overall accuracy, then adjust accordingly, and keep going through iterations until all checkpoints are identified correctly.

[0124] In block **226** once model fitting is complete, the base assessment for the particular movement is stored in a database.

Customization:

[0125] FIG. **2**D illustrates a flowchart depicting an example process for performing customization as may be embodied by customization engine **144** in a representative implementation. Up to now, we have identified baselines for best case scenarios for each movement.

[0126] In block **231** as a coach adjusts determined features in the web GUI or to their liking, a customization engine will determine a difference between base values and the coach's version of the assessment.

[0127] In block **232** all labeled video poses and confidences are ran through a feature extractor and compared against the new values. This is to check as see how far the adjusted values are different from the baseline.

[0128] In block **233** if a range is determined to no longer identify movement checkpoints, then in block **234** an error is reported to the coach. The coach is alerted that modified value would not meet the checkpoint in the baseline.

[0129] In block **235** the coach can re-adjust the values and try again.

[0130] In block **236** if it is found that all ranges can still properly identify the labeled checkpoints, the customization range is finished and a coach assessment is stored in a database for future use.

[0131] Having reviewed the processes conducted by the AI server **104** in more detail with reference to FIGS. **2**A, **2**B, **2**C and **2**D, we now look at the processes conducted by the AI client **102** in more detail with reference to FIGS. **3**A, **3**B, and **3**D.

Edge Application:

[0132] FIG. **3**A illustrates a flowchart depicting an example processing conducted by an edge application **120** for performing training using feedback and rule-based motion analysis gathered from a user session in a representative implementation.

[0133] In block **301**, live-stream or recorded video is captured by application video feed code **121** in color or B&W format. While embodiments can be realized in virtually any size video, one implementation presently in use employs video in 1920×1080 (landscape) and 1080×1920 (portrait) sizes), however other sizes are appropriate to various applications. Frames are scaled to appropriate size for model input, such as for example a currently implementation resizes to 224×416 (portrait), 416×224 (landscape), however other sizes are appropriate to various applications.

[0134] In block **302**, extracting by pose estimation code **122** X number of keypoints that describe areas of interest on the body. For example: in one implementation, the location of a keypoint corresponding to the user's head is found to be at (x: 200, y: 300). Implementations can employ either 3D or 2D images.

[0135] In block **303**, feature extractor code **123** smooths the current pose using an appropriate digital signal processing (DSP) functionality to prevent perceived jumpiness in the video image when displayed, especially when using a mobile or small footprint device. One implementation uses Kalman filters to remove jumpiness (see e.g., //en.wikipedia.org/wiki/Kalman_filter; which is incorporated herein by reference in its entirety for all purposes). Another implementation uses 1€ ("One Euro") Filters as described in the paper here: //dl.acm.org/doi/10.1145/2207676.2208639?cid=81100168597 which is incorporated herein by reference in its entirety for all purposes. Given the current exercise being taught, extract features from the relevant assessment rules **126**.

[0136] In block **304**, rules evaluator **124** implements given the current set of extracted features, comparing them to the assessment rules **126**. Thus, the rules evaluator **124** is able to determine a current state of the user's pose.

[0137] In block **305**, feedback engine **125** implements given the determined state, identify a message (or NULL message) to present or play for the user.

Feedback Engine:

[0138] FIG. **3**B illustrates a flowchart depicting an example process for performing rule-based feedback of motion analysis as may be embodied by feedback engine code **125** in a representative implementation. In the example process illustrated by flowchart **300**B, feedback engine **125** evaluates a collection of facts and determines an output message and an accompanying audio file to play based on how previous audio messages affected the user's performance of a specific task.

[0139] In block **311**, receiving state information describing a user pose state, including: obtaining a collection of current and previous facts, including at least fact that can be a (i) constant and known at the start of the feedback session, e.g., the length of a session, or (ii) dynamic, e.g., the amount

of time that has elapsed since the last error was seen, a current time (first timestamp+number of timer fires) [repetition and/or timestamp, since event or since timer started, or e.g., time since midnight] selected from a number of times an error has been previously seen, and if the user is in an improper position [bundle from pose engine.] "label of error to present to the user—every label error can have say Y audio files to choose from" while being assessed on a movement [using a vision engine, and/or a camera(s)].

[0140] In block **312**, receiving an indication that a periodic timer has expired. The period of the timer is variable and depends on the nature of the action receiving feedback. [If the action is lively and strictly timed, the timer will have a smaller period, typically 1 second.][If the action is long-lived and not explicitly timed, the timer will have a larger period.](plank—slow; lift maybe 1 second; every few minutes for a long term low intensity activity)(passed into the engine—what errors to respond to and how frequently to check for the errors); and triggering responsive to the expiration of the timer:

[0141] In block **313**, evaluating the facts in the collection of facts as received to determine an appropriate response message [e.g., a coaching message], including: [collect state, check facts, play "hips too high"] weighting dynamic features that can be weighted such that the output of the fact is true if the fact is true and then if an evenly distributed random value (0,1] is greater than or equal to the weight. [At the beginning of a session all dynamic weights are set to random values if there is no history of what the weight values should be.] (a random variable can allow varying audio feedback).

[0142] In block **314**, determining an appropriate response message type [using the weighted dynamic features], or none if none exists. For each message type, there exist many variants of audio recordings from which the system can choose (e.g., types—audio or visual, specific message—too high, or encouragement, or warning, termination message).

[0143] In block **315**, automatically selecting for output a selected audio output response message having the response message type as determined. Initially, an audio output is chosen at random and is played to the user and the time at which the audio was played and the corresponding message is stored for future reference.

[0144] FIG. **3**C illustrates a flowchart depicting an example process for performing continual training of a machine learning automaton based upon feedback and rule-based motion analysis gathered from a user session in a representative implementation. Once performing the audio output response message is selected in block **315** of FIG. **3**B, control passes to the processing of FIG. **3**C in a second thread that analyzes what has happened—e.g., capturing results and evaluating historical outcomes are performed by counting a count of corrections, etc. thereby enabling viewing a stream of decisions over time.

[0145] In a block **321**, After evaluating facts, the Feedback Engine evaluates historical outcomes. An outcome is a consequence of the combination of facts evaluated and the message played to the user. By evaluating all previous outcomes, the time between similar outcomes can be identified. For instance, the time between two errors reported to the user can be evaluated. In implementations, fitting something to data points, e.g., fitting a line to data points. [The system can then use machine learning approaches such as association rules and linear regression] (e.g., maximize time/reps between errors, gradient descent.) Some embodiments implement Variable times. Some embodiments implement a Max time between consecutive errors. This can achieve Mirroring what coach wants to get out of you. Some embodiments implement techniques for tracking for a Long term—e.g., 10 session's worth of data to collect sufficient sample size to adjust the dynamic weights. Some embodiments implement using a probability—ask for random number and choose based on probability—as a weight—e.g., error every time, person responds vs. person does not respond well to the message, of certain facts (resistance, repetitions, etc.) that are pre-determined to affect an outcome.

[0146] In a block **322**, dynamic weights adjusted based upon the outcome of the processing of block **321** are stored (e.g., as floating-point, or other numbers)—useful as personalization to a user to bring about desired outcome for that user. These adjusted weights are then used in subsequent executions of the engine and the process continues.

[0147] In a block **323**, if training is continuing, control passes back to block **311** of FIG. **3**B. Otherwise, when a feedback session ends, in block **324**, the dynamic weights are stored in a database and can be used again for subsequent sessions of the same assessment or can be used as initial values for similar facts in other assessments. Preferably, dynamic weights are stored on the device **102** to meet privacy, e.g., GDPR requirements, in a database for future use.

[0148] The computer implemented methods described above can be practiced in a system that includes computer hardware. The computer implemented system can practice one or more of the methods described above. The computer implemented system can incorporate any of the features of methods described immediately above or throughout this application that apply to the method implemented by the system. In the interest of conciseness, alternative combinations of system features are not individually enumerated. Features applicable to systems, methods, and articles of manufacture are not repeated for each statutory class set of base features. The reader will understand how features identified in this section can readily be combined with base features in other statutory classes.

[0149] As an article of manufacture, rather than a method, a non-transitory computer readable medium (CRM) can be loaded with program instructions executable by a processor. The program instructions when executed, implement one or more of the computer-implemented methods described above. Alternatively, the program instructions can be loaded on a non-transitory CRM and, when combined with appropriate hardware, become a component of one or more of the computer-implemented systems that practice the methods disclosed.

[0150] Each of the features discussed in this particular implementation section for the method implementation apply equally to CRM and system implementations. As indicated above, all the method features are not repeated here, in the interest of conciseness, and should be considered repeated by reference.

Payload Data Model:

[0151] FIG. **4** illustrates a typical payload exchanged between an AI server implementation and an AI client implementation in an embodiment. Payload **90** of FIG. **4**

includes assessment rules **400** that are provided to an AI client **102** by an AI server **104** implement that follow these guidelines:

[0152] 1. Each checkpoint **406**, **408** for an exercise is labeled using "checkpoint_[name]". A checkpoint entitled "checkpoint_initial" (e.g., **406**) is required for all exercises.

[0153] 2. For each checkpoint, e.g., **408**, the working side of the body is listed as well as the combination of features **412**, **414**, **416**, **418** that are used to identify the checkpoint.

[0154] 3. For each feature, a range of values is provided. Some features may be a single value (e.g., checkpoint **408** includes feature **412** "is_facing_forward"), some may have 2 values with a min and max value, e.g., checkpoint **408** also includes feature **414** (Right_Ankle_Angle_θ_1) having min value of 22 degrees and a max value of 72 degrees, see e.g., **422**, and some may have 4 ranges with a lower min max and a higher min max (not shown in FIG. **4** for clarity sake).

[0155] 4. Other metadata provided in the admin GUI, such as error messages to tell the user to coach them in the event of an identified checkpoint, are listed under each checkpoint as well. For example, feature **414** includes an error message **432** "Knees Too Far Forward" that can be triggered for display by client **102** upon detecting a user's Right_Ankle_Angle falling below the min value.

[0156] 5. The Assessment Rules **400** of payload **90** defines a version number 402, which represents the version of the client-side Edge Application **120** implementation that is required to parse it successfully.

Example Image Processing and Pose Recognition

[0157] FIG. **5**A illustrates an example raw image frame in an example application of the disclosed technology to conduct training using feedback and rule-based motion analysis gathered from a user session in a representative implementation. An image frame, typically from a live video feed, is extracted and fed into the pose estimation model.

[0158] FIG. **5**B illustrates an example raw model output in an example application of the disclosed technology to conduct training using feedback and rule-based motion analysis gathered from a user session in a representative implementation. The pose estimation model's outputs are then processed to return a raw list of coordinates and the corresponding confidence of each coordinate pair depicted by FIG. **5**B.

[0159] FIG. **5**C illustrates an example annotated image output in an example application of the disclosed technology to conduct training using feedback and rule-based motion analysis gathered from a user session in a representative implementation. In FIG. **5**C, the coordinates are scaled to match the image's coordinates and are annotated on the input image. Each identified coordinate is a key point.

[0160] FIG. **5**D illustrates an example of translating key points to body joints in an example application of the disclosed technology to conduct training using feedback and rule-based motion analysis gathered from a user session in a representative implementation. The index of the key point in the output from the pose estimation model correlates to the key point that was used for training the model. Each of the points used in training has a semantic meaning; for example, the point at the tip of the right foot of the user is given the semantic label "Right Toe". This label is known as a body joint. A set of coordinates of the point is also provided. The coordinates returned from the model are in the coordinate space of the input image, e.g., 224×416. The outputted coordinates are then translated to that of the display image, e.g., 768×1024. To create the figure we show the coordinates were translated. Also, a confidence level is also included.

[0161] FIG. **5**E illustrates an example labelling in an example application of the disclosed technology to conduct training using feedback and rule-based motion analysis gathered from a user session in a representative implementation. Each of the output key points has a significance. Here the corresponding meanings, or body joints, are illustrated by FIG. **5**E for the example key points.

[0162] FIG. **5**F illustrates an example keypoints in a pose in an example application of the disclosed technology to conduct training using feedback and rule-based motion analysis gathered from a user session in a representative implementation. The collection of points shown in FIG. **5**F is a pose. The pose includes each of the identified key points and the corresponding confidence.

[0163] FIG. **5**G illustrates an example feature definition in an example application of the disclosed technology to conduct training using feedback and rule-based motion analysis gathered from a user session in a representative implementation. Information derived from key point locations is referred to as a "feature". A common feature is the angle between related key points. Indicated as "0" in FIG. **5**G is the "Right_Knee_Hip_Anke Angle" feature **550**.

[0164] FIG. **5**H illustrates an example checkpoints definition in an example application of the disclosed technology to conduct training using feedback and rule-based motion analysis gathered from a user session in a representative implementation. Given a timeline of images (e.g., those from a live video feed), each identified poses' features are analyzed. A checkpoint is a collection of feature ranges that identify a position in a movement of importance. Image **592** of FIG. **5**H shows that the detected features (θ**1**, θ**2**, θ**3**) are within the ranges that are acceptable for the given checkpoint. In some implementations, movement checkpoints outside of a range are displayed in red by a graphical user interface (GUI) and movement checkpoints within a range are displayed in green by the graphical user interface (GUI).

[0165] FIG. **6** illustrates an example graphical user interface (GUI) implemented in some embodiments. The Admin GUI **146** of FIG. **6** is used by the coach to modify errors and attach text and audio feedback for the user. A clickable interface to identify "important" joints **601**. Clickable interface **602** enables manually setting ranges for body features (these will be pre-populated by the automated process). Display window **603** provides text correction for user. Clickable Audio button **604** enables attaching audio of coaching the correction.

[0166] FIG. **7** illustrates assessing poses over time of checkpoints in an example application of the disclosed technology to conduct training using feedback and rule-based motion analysis gathered from a user session in a representative implementation. In FIG. **7**, two feature values are mapped over time. Each vertical dotted line is a clock timer tick. The measured values between clock timer ticks are smoothed to create a line or curve for visualization purposes. At time **50**, the feature values are both within angle ranges that are pre-determined by teaching the neural network to represent a checkpoint. They remain the acceptable ranges for that checkpoint until time **100**. The example then leaves that state and waits for the next checkpoint.

Classifier Inputs and Outputs

[0167] FIG. **8** illustrates a labelling of an exemplary input training set for training a deep neural network implementation. The implementation example in FIG. **8** selects an appropriate audio feedback using a set of inputs to the neural network. Inputs (θ1, θ2, θ3) in the example illustrated by FIG. **8** correspond to angles in FIGS. **5**A-**5**H. Whether structured or unstructured data type data points, inputs can be encoded into fields of a vector (or tensor) representation. Implementations will employ various levels of abstraction in configuring, classification and anomaly detection tasks, e.g., in a physical therapy application, data can be selected to describe movements, rates of repetitions, number of repetitions to failure and so forth. In one example, a neural network ensemble can implement a classifiers that are trained to classify situation states according to input data useful in human movement training applications, such as without limitation physiotherapies, personal training, athletic training, and the like.

[0168] The classifier(s) once trained on a training dataset **800** can determine based on the inputs whether an observed motion is meets success criteria (Correct form, Sufficient repetitions, Sufficient rate) for a particular situation state. The exemplary deep neural network implementation selects an appropriate classification of a failing effort when detected and can select an appropriate audio feedback and detect a dangerous situation from a set of inputs.

[0169] In one exemplary implementation, some neural networks implementing learning model(s) **131** are implemented as an ensemble of subnetworks trained using datasets widely chosen from approved transactions and flagged transactions, with outputs including classifications of anomalies based upon the input sensed data, and/or remedial actions to be triggered by invoking downstream applications such as audio feedback, preparing and submitting reports to a human coach or trainer, as well as the capability to both cluster information and to escalate problems.

Computer System

[0170] FIG. **9** illustrates one implementation of a computer system **900** that can be used to implement the technology disclosed. Computer system **900** includes at least one central processing unit (CPU) **972** that communicates with a number of peripheral devices via bus subsystem **955**. These peripheral devices can include a storage subsystem **910** including, for example, memory devices and a file storage subsystem **936**, user interface input devices **938**, user interface output devices **976**, and a network interface subsystem **974**. The input and output devices allow user interaction with computer system **900**. Network interface subsystem **974** provides an interface to outside networks, including an interface to corresponding interface devices in other computer systems. The analyzer and deep learning system can be communicably linked to the storage subsystem **910** and the user interface input devices **938**.

[0171] User interface input devices **938** can include a keyboard; pointing devices such as a mouse, trackball, touchpad, or graphics tablet; a scanner; a touch screen incorporated into the display; audio input devices such as voice recognition systems and microphones; and other types of input devices. In general, use of the term "input device" is intended to include all possible types of devices and ways to input information into computer system **900**.

[0172] User interface output devices **976** can include a display subsystem, a printer, a fax machine, or non-visual displays such as audio output devices. The display subsystem can include an LED display, a cathode ray tube (CRT), a flat-panel device such as a liquid crystal display (LCD), a projection device, or some other mechanism for creating a visible image. The display subsystem can also provide a non-visual display such as audio output devices. In general, use of the term "output device" is intended to include all possible types of devices and ways to output information from computer system **900** to the user or to another machine or computer system.

[0173] Storage subsystem **910** stores programming and data constructs that provide the functionality of some or all of the modules and methods described herein. These software modules are generally executed by deep learning processors **978**.

[0174] Deep learning processors **978** can be graphics processing units (GPUs) or field-programmable gate arrays (FPGAs). Deep learning processors **978** can be hosted by a deep learning cloud platform such as Google Cloud Platform™, Xilinx™, and Cirrascale™. Examples of deep learning processors **978** include Google's Tensor Processing Unit (TPU)™, rackmount solutions like GX4 Rackmount Series™, GX2 Rackmount Series™, NVIDIA DGX-1™, Microsoft' Stratix V FPGA™, Graphcore's Intelligent Processor Unit (IPU)™, Qualcomm's Zeroth Platform™ with Snapdragon Processors™, NVIDIA's Volta™, NVIDIA's DRIVE PX™, NVIDIA's JETSON TX1/TX2 MODULE™, Intel's Nirvana™, Movidius VPU™, Fujitsu DPI™, ARM's DynamicIQ™, IBM TrueNorth™, and others.

[0175] Memory subsystem **922** used in the storage subsystem **910** can include a number of memories including a main random access memory (RAM) **932** for storage of instructions and data during program execution and a read only memory (ROM) **934** in which fixed instructions are stored. A file storage subsystem **936** can provide persistent storage for program and data files, and can include a hard disk drive, a floppy disk drive along with associated removable media, a CD-ROM drive, an optical drive, or removable media cartridges. The modules implementing the functionality of certain implementations can be stored by file storage subsystem **936** in the storage subsystem **910**, or in other machines accessible by the processor.

[0176] Bus subsystem **955** provides a mechanism for letting the various components and subsystems of computer system **900** communicate with each other as intended. Although bus subsystem **955** is shown schematically as a single bus, alternative implementations of the bus subsystem can use multiple busses.

[0177] Computer system **900** itself can be of varying types including a personal computer, a portable computer, a workstation, a computer terminal, a network computer, a television, a mainframe, a server farm, a widely-distributed set of loosely networked computers, or any other data processing system or user device. Due to the ever-changing nature of computers and networks, the description of computer system **900** depicted in FIG. **9** is intended only as a specific example for purposes of illustrating the preferred embodiments of the present invention. Many other configurations of computer system **900** are possible having more or less components than the computer system depicted in FIG. **9**.

[0178] FIG. **10** is a simplified diagram of a mobile phone computing platform **1000**, representative of computing

devices which can be used as an apparatus for edge application **120** described herein. Other computing devices configured for user motion observation, as described herein, can have a similar platform, including devices which can be modular in form factor for deployment in a variety of settings, and that are configured for communication with local servers, and devices that are configured for wireless communication via the internet, rather than the mobile phone network.

[0179] In this example, the computing platform **1000** includes an antenna **1001** and a radio including a radio frequency RF receiver/transmitter **1002**, by which the computing platform **1000** is coupled to a wireless communication medium, according to one or more of a variety of protocols. In examples described herein, the RF receiver/transmitter **1002** can include one or more radios to support multiprotocol/multiband communications for communication with the wireless service provider of the mobile phone network, as well as the establishment of wireless local radio links using a protocol like Bluetooth® or WIFI protocols. The receiver/transmitter **1002** is coupled to baseband circuitry and a digital processor in processing section **1003**, in which the audio signals are processed and call signals are managed. A codec **1004**, including analog-to-digital and digital-to-analog converters, is coupled to the processing section **1003**. A microphone **1005** and a speaker **1006** are coupled to the codec **1004**.

[0180] Memory **1007** which can be a nonvolatile read-only memory, stores a dynamic weightings, rules, neural networks, and machine learning model artifacts for use in a recognition and classification algorithms, as well as instructions, parameters and other data for execution by the processing section **1003**. In addition, a read/write memory **1008** in the mobile phone stores instructions and parameters for recognition processes and other data for use by the processing section **1003**. There may be multiple types of read/write memory on the computing platform **1000**, such as nonvolatile read/write memory **1008** (flash memory or EEPROM for example) and volatile read/write memory **1009** (DRAM or SRAM for example). Other embodiments include removable memory modules in which instructions, parameters and other data for use by the processing section **1003** are stored.

[0181] An input/output controller **1010** is coupled to a touch sensitive display **1011** and to user input devices **1012**, such as a camera, a function keypad, activity trackers connectible via Bluetooth, WIFI or the like. The camera can be used to capture images for the motion recognition and classification and instruction. An accessory port or ports **1013** coupled to the controller **1010** are used for other types of input/output devices, such as binaural and monaural headphones, connections to processing devices such as PDAs, or personal computers, alternative communication channels such as an infrared port or universal serial bus USB port, a portable storage device port, and other things. The controller **1010** is coupled to the processing section **1003**. User input concerning call set up and call management, and concerning use of the motion recognition and classification and instruction, user preferences and the like received via the input devices **1012** and optionally via accessories. User interaction is enhanced, and the user is prompted to interact, using the touch display **1011** and optionally other accessories. Input may also be received via the microphone **1005** supported by voice recognition programs, and user interaction and prompting may utilize the speaker **1006** for various purposes.

[0182] In the illustrated embodiment, memory **1008** stores a program for displaying a function selection menu user interface on the display **1011**, such that the user can select the functions to be carried out during the motion recognition and classification and instruction discussed herein. Also, the instructions executable by the processing section **1003** and/or the controller **1010**, are stored in a non-transitory medium such as the memory **1007**, **1008**, **1009**, that includes logic for executing the sequence of operations outlined above in connection with FIGS. **2**A, **2**B, **2**C, **2**D and **3**A, **3**B, and **3**C.

Convolutional Neural Networks

[0183] A convolutional neural network is a special type of neural network. The fundamental difference between a densely connected layer and a convolution layer is this: Dense layers learn global patterns in their input feature space, whereas convolution layers learn local patters: in the case of images, patterns found in small 2D windows of the inputs. This key characteristic gives convolutional neural networks two interesting properties: (1) the patterns they learn are translation invariant and (2) they can learn spatial hierarchies of patterns.

[0184] Regarding the first, after learning a certain pattern in the lower-right corner of a picture, a convolution layer can recognize it anywhere: for example, in the upper-left corner. A densely connected network would learn the pattern anew if it appeared at a new location. This makes convolutional neural networks data efficient because they need fewer training samples to learn representations they have generalization power.

[0185] Regarding the second, a first convolution layer can learn small local patterns such as edges, a second convolution layer will learn larger patterns made of the features of the first layers, and so on. This allows convolutional neural networks to efficiently learn increasingly complex and abstract visual concepts.

[0186] A convolutional neural network learns highly non-linear mappings by interconnecting layers of artificial neurons arranged in many different layers with activation functions that make the layers dependent. It includes one or more convolutional layers, interspersed with one or more sub-sampling layers and non-linear layers, which are typically followed by one or more fully connected layers. Each element of the convolutional neural network receives inputs from a set of features in the previous layer. The convolutional neural network learns concurrently because the neurons in the same feature map have identical weights. These local shared weights reduce the complexity of the network such that when multi-dimensional input data enters the network, the convolutional neural network avoids the complexity of data reconstruction in feature extraction and regression or classification process.

[0187] Convolutions operate over 3D tensors, called feature maps, with two spatial axes (height and width) as well as a depth axis (also called the channels axis). For an RGB image, the dimension of the depth axis is 3, because the image has three color channels; red, green, and blue. For a black-and-white picture, the depth is 1 (levels of gray). The convolution operation extracts patches from its input feature map and applies the same transformation to all of these patches, producing an output feature map. This output

feature map is still a 3D tensor: it has a width and a height. Its depth can be arbitrary, because the output depth is a parameter of the layer, and the different channels in that depth axis no longer stand for specific colors as in RGB input; rather, they stand for filters. Filters encode specific aspects of the input data: at a height level, a single filter could encode the concept "presence of a face in the input," for instance.

[0188] For example, the first convolution layer takes a feature map of size (28, 28, 1) and outputs a feature map of size (26, 26, 32): it computes 32 filters over its input. Each of these 32 output channels contains a 26×26 grid of values, which is a response map of the filter over the input, indicating the response of that filter pattern at different locations in the input. That is what the term feature map means: every dimension in the depth axis is a feature (or filter), and the 2D tensor output [:, :, n] is the 2D spatial map of the response of this filter over the input.

[0189] Convolutions are defined by two key parameters: (1) size of the patches extracted from the inputs—these are typically 1×1, 3×3 or 5×5 and (2) depth of the output feature map—the number of filters computed by the convolution. Often these start with a depth of 32, continue to a depth of 64, and terminate with a depth of 128 or 256.

[0190] A convolution works by sliding these windows of size 3×3 or 5×5 over the 3D input feature map, stopping at every location, and extracting the 3D patch of surrounding features (shape (window_height, window_width, input_depth)). Each such 3D patch is then transformed (via a tensor product with the same learned weight matrix, called the convolution kernel) into a 1D vector of shape (output_depth,). All of these vectors are then spatially reassembled into a 3D output map of shape (height, width, output_depth). Every spatial location in the output feature map corresponds to the same location in the input feature map (for example, the lower-right corner of the output contains information about the lower-right corner of the input). For instance, with 3×3 windows, the vector output [i, j, :] comes from the 3D patch input [i−1: i+1, j−1: J+1, :]. The full process is detailed in FIG. **11**, illustrating an implementation of a convolutional neural network suitable for implementing the disclosed technology.

[0191] The convolutional neural network comprises convolution layers which perform the convolution operation between the input values and convolution filters (matrix of weights) that are learned over many gradient update iterations during the training. Let (m, n) be the filter size and W be the matrix of weights, then a convolution layer performs a convolution of the W with the input X by calculating the dot product W·x+b, where x is an instance of X and b is the bias. The step size by which the convolution filters slide across the input is called the stride, and the filter area (m×n) is called the receptive field. A same convolution filter is applied across different positions of the input, which reduces the number of weights learned. It also allows location invariant learning, i.e., if an important pattern exists in the input, the convolution filters learn it no matter where it is in the sequence.

Training a Convolutional Neural Network

[0192] FIG. **12** depicts a block diagram of training a convolutional neural network in accordance with one implementation of the technology disclosed. The convolutional neural network is adjusted or trained so that the input data leads to a specific output estimate. The convolutional neural network is adjusted using back propagation based on a comparison of the output estimate and the ground truth until the output estimate progressively matches or approaches the ground truth.

[0193] The convolutional neural network is trained by adjusting the weights between the neurons based on the difference between the ground truth and the actual output. This is mathematically described as:

$$\Delta w_i = x_i \delta$$

[0194] where δ=(ground truth)−(actual output)

[0195] In one implementation, the training rule is defined as:

$$w_{nm} \leftarrow w_{nm} + \alpha(t_m - \varphi_m)a_n$$

[0196] In the equation above: the arrow indicates an update of the value; t_m is the target value of neuron m; φ_m is the computed current output of neuron m; a_n is input n; and α is the learning rate.

[0197] The intermediary step in the training includes generating a feature vector from the input data using the convolution layers. The gradient with respect to the weights in each layer, starting at the output, is calculated. This is referred to as the backward pass, or going backwards. The weights in the network are updated using a combination of the negative gradient and previous weights.

[0198] In one implementation, the convolutional neural network uses a stochastic gradient update algorithm (such as ADAM) that performs backward propagation of errors by means of gradient descent. One example of a sigmoid function based back propagation algorithm is described below:

$$\varphi = f(h) = \frac{1}{1 + e^{-h}}$$

[0199] In the sigmoid function above, h is the weighted sum computed by a neuron. The sigmoid function has the following derivative:

$$\frac{\partial \varphi}{\partial h} = \varphi(1 - \varphi)$$

[0200] The algorithm includes computing the activation of all neurons in the network, yielding an output for the forward pass. The activation of neuron m in the hidden layers is described as:

$$\varphi_m = \frac{1}{1 + e^{-hm}}$$

$$h_m = \sum_{n=1}^{N} a_n W_{nm}$$

[0201] This is done for all the hidden layers to get the activation described as:

$$\varphi_k = \frac{1}{1+e^{hk}}$$

$$h_k = \sum_{m=1}^{M} \varphi_m v_{mk}$$

[0202] Then, the error and the correct weights are calculated per layer. The error at the output is computed as:

$$\delta_{ok}=(t_k-\varphi_k)\varphi_k(1-\varphi_k)$$

[0203] The error in the hidden layers is calculated as:

$$\delta_{hm} = \varphi_m(1-\varphi_m)\sum_{k=1}^{K} v_{mk}\delta_{ok}$$

[0204] The weights of the output layer are updated as:

$$vmk \leftarrow vmk+\alpha\delta ok\varphi m$$

[0205] The weights of the hidden layers are updated using the learning rate α as:

$$vnm \leftarrow wnm+\alpha\delta hman$$

[0206] In one implementation, the convolutional neural network uses a gradient descent optimization to compute the error across all the layers. In such an optimization, for an input feature vector x and the predicted output ŷ, the loss function is defined as l for the cost of predicting ŷ when the target is y, i.e. l(ŷ, y). The predicted output ŷ is transformed from the input feature vector x using function f. Function f is parameterized by the weights of convolutional neural network, i.e. $\hat{y}=f_w(x)$. The loss function is described as $l(\hat{y}, y)=l(f_w(x), y)$, or $Q(z, w)=l(f_w(x), y)$ where z is an input and output data pair (x, y). The gradient descent optimization is performed by updating the weights according to:

$$v_{t+1} = \mu v_t - \alpha\frac{1}{n}\sum_{i=1}^{N}\nabla wtQ(zt, w_t)$$

$$w_{t+1} = w_t + v_{t+1}$$

[0207] In the equations above, α is the learning rate. Also, the loss is computed as the average over a set of n data pairs. The computation is terminated when the learning rate α is small enough upon linear convergence. In other implementations, the gradient is calculated using only selected data pairs fed to a Nesterov's accelerated gradient and an adaptive gradient to inject computation efficiency.

[0208] In one implementation, the convolutional neural network uses a stochastic gradient descent (SGD) to calculate the cost function. A SGD approximates the gradient with respect to the weights in the loss function by computing it from only one, randomized, data pair, z_t, described as:

$$v_{t+1}=\mu v-\alpha\nabla wQ(z_t, w^t)$$

$$w_{t+1}=w_t+v_{t+1}$$

[0209] In the equations above: Δ is the learning rate; μ is the momentum; and t is the current weight state before updating. The convergence speed of SGD is approximately O(1/t) when the learning rate α are reduced both fast and slow enough. In other implementations, the convolutional neural network uses different loss functions such as Euclidean loss and softmax loss. In a further implementation, an Adam stochastic optimizer is used by the convolutional neural network.

[0210] Having described neural network implementations, the discussion now turns to deep learning approaches.

[0211] FIG. **13** illustrates a deep learning system in a supervised or semi-supervised implementation. As shown, deep learning system **1300** includes training servers **1302** and production servers **1304**. Large scale training dataset **1312** is accessible to training servers **1302** for training the deep convolutional neural network **131**. In an implementation, deep neural network **131** includes a first anomaly subnetwork, and a second solution accessibility subnetwork that are trained on one or more training servers **1302**. The trained deep neural network ensemble including the first trained anomaly subnetwork, and the trained second solution accessibility subnetwork are deployed on one or more production servers **1304** that receive input anomaly information from requesting client devices **102**. The production servers **1304** process the input anomaly information through at least one of the deep neural network **131**, the first anomaly subnetwork, and the second solution accessibility subnetwork to produce outputs that are transmitted to the client devices **102**.

[0212] Training servers **1302** conduct training using models and comprise a situation dataset generator **1322** includes a deep convolutional neural network based variant anomaly classifier, running on numerous processors coupled to memory that prepares training sets comprising data chosen from large scale training dataset **1312** to reflect one or more scenarios being trained, a variant anomaly classifier **1332** includes a deep convolutional neural network based variant anomaly classifier, running on numerous processors coupled to memory that is trained to recognize anomalous situations from sensed data using the scenarios prepared, an optional secondary classifier **1342** includes a deep convolutional neural network based secondary anomaly classifier, running on numerous processors coupled to memory that is trained to recognize special situation anomalies (e.g., radioactive spill, biohazard, etc.), a solution accessibility classifier **1352** includes a deep convolutional neural network based secondary anomaly classifier, running on numerous processors coupled to memory that is trained to recognize anomalies and output identifiers identifying remedial applications that are invoked to trigger remedial actions. A semi-autonomous learner **1362** includes a deep convolutional neural network based variant anomaly classifier, running on numerous processors coupled to memory that progressively augments a set size of the anomaly training set based on the trained ensemble's evaluation of a synthetic set or in implementations, input of live data from a real world scenario.

[0213] In one implementation, the neural networks such as situation dataset generator, variant anomaly classifier, secondary anomaly classifier, solution accessibility classifier, and semi-autonomous learner are communicably linked to the storage subsystem comprised of test data database **1373**,

production data database **1374**, inferred data database **1375** and other private data database **1376** and user interface input devices.

[0214] In one implementation, data used in one or more of large scale training dataset **1312**, test data database **1373**, production data database **1374**, inferred data database **1375** and other private data database **376** is selectively obtained from multiple sources of data: (i) various drug databases (e.g., the FDA Product-Specific Guidance database, which enables searching and clustering by active ingredient(s)) and communications including machine reading of emails on recalls minimizes the need to change notification protocols that can be related to machine-readable data and image recognition (e.g. images of pills) and (ii) user responses to deep learning driven follow-up questions selected by the solution accessibility classifier **1352** and semi-autonomous learner **1362** (allowing for live training and refinement).

Grid Search:

Hyperparameters

[0215] Hyperparameters are model configuration parameters that determine neural network structure as well as how the network is trained. Common examples of hyperparameters are "learning rate", or how quickly the model is adapted to the input problem, and "batch size", or how many pieces of training data are fed into the model at each training step. These parameters are set before training occurs.

Accuracy

[0216] After training completes, a model is assessed to see how well it completes its given task using various metrics including accuracy. Accuracy is simply the fraction of predictions the model correctly labeled.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total Number of predictions}}$$

[0217] If a model's accuracy is not within acceptable range (i.e. it is not performing the given task successfully), we can adjust the hyperparameters, retrain a new model, and then assess the new model's performance.

Optimizing Hyperparameters

[0218] Tuning, or optimizing hyperparameters, is a complex problem aims to identify the optimal collection of parameters for a given model or algorithm. Grid Search is a common technique that performs an exhaustive search of given parameters to determine which collection of parameters results in the best performance of the model or algorithm.

How we Use Grid Search

[0219] FIG. **14**A illustrates an example iterative machine learning process such as a grid search in a representative implementation. Consider a collection of labeled video frames for 1000 videos **1402** where we have stored the upper and lower bounds for four features: f1, f2, f3, f4 as well as whether or not the poses in the frame is considered in a given checkpoint.

Training a Model

[0220] We can use many different models and algorithms to train a classifier that, given a collection of the features f1, f2, f3, and f4, can predict whether or not the given frame is in a given checkpoint.

[0221] Now with reference to FIG. **14**B, showing use of a simple example **1412** of a C-Support Vector Classification (SVC) provided by the machine learning toolkit Sklearn. In alternative implementations, many other algorithm(s) or deep learning model(s) that we create or that is provided by an existing deep learning library can be used.

Improving the Model

[0222] With continuing reference to FIG. **14**B, a first iteration of training a model has resulting in an accuracy **1414** of only 70%. This low accuracy would be experienced during the training pipeline and would result in an evaluation of a test video not being properly labeled as in a checkpoint or not.

[0223] Now with reference to FIG. **14**C, showing performing a Grid Search **1422** to determine new hyperparameters for the given model (in this case an SVC). This will use our existing labeled data that the system has ingested; no new data are added.

Finished

[0224] We now have a model that is 98% accurate (e.g., **1424**). This new, improved accuracy improves the chance that a given test frame is accurately labeled, which would allow the system to proceed with its automatic generation of assessment rules for assessing a given exercise.

Other Particular Implementations

[0225] In an aspect of the present technology a method of automated determination of a base assessment for a pose or movement includes performing video analysis, including: obtaining a manifest and corresponding recorded videos of individuals performing particular movements in proper states ("correct form") and in improper states ("incorrect form"); extracting portions of the videos for evaluation, while maintaining the descriptions from the manifest for the frames of the extracted portions of the videos; and inputting, into a pose estimation neural network, the extracted portions of the videos one frame at a time. The method further includes receiving, as an output of the pose estimation neural network and for each input frame, a pose comprising a collection of the keypoints in the frame, the keypoints corresponding to body parts, including (i) coordinates of one or more keypoints in the frame and (ii) confidences for each keypoint representing a probability that a keypoint is a particular feature of each of a body portion subject to evaluation. The method further includes outputting labeled payloads of poses and confidences for each frame of the extracted portions of the videos, wherein the labeled payload indicates (i) whether the video slice depicts a body that is in a correct form or an incorrect form, (ii) keypoint information and (iii) confidence information).

[0226] The method further includes performing movement analysis using results of the video analysis, including identifying a particular movement, a video associated with the particular movement and a corresponding manifest; examining the corresponding manifest to identify candidate body

features and automatically selecting from the candidate body features, relevant body features to serve as keypoints. For each pose and confidence in the labeled payloads, extracting the relevant body features; using the manifest, extracting checkpoints comprising the relevant body features across one or more input videos; determining for each of the relevant body features identified, a relevant range of values; and providing recommendations including ranges for relevant body features that are acceptable for a particular movement, thereby enabling a model to be fit to the movements in the video within the relevant range of values.

[0227] The method includes performing model fitting using results of the movement analysis to determine keypoints and body features relevant to classifying a pose as correct form or incorrect form, including: performing body feature extraction for a video using the poses and confidences obtained. For each labelled checkpoint, whether the pose at the checkpoint is proper or improper is estimated; and whenever checkpoints were mislabeled in an estimate that the pose at the checkpoint is improper, an iterative machine learning process is applied to adjust ranges of the body features until each checkpoint is identified properly, thereby resulting in the estimate matching the manifest. The method further includes storing a base assessment for the particular movement in a database once the poses at all of the checkpoints are identified correctly, wherein the base assessment includes identified baselines for best case scenarios for each pose/movement.

[0228] In another aspect of the present technology, video analysis can further include labeling a payload indicating (i) that a particular frame includes a particular pose, and poses of each keypoint, (ii) confidences of each keypoints, and (iii) an aggregate of confidences over multiple keypoints for a particular repetition of a movement in the task.

[0229] In further aspect of the present technology, video analysis can also include using one or more confidences to weight some keypoints assigned to joints, thereby enabling confidences of a first keypoint associated with a first joint permeate through labeling of other keypoints associated with other joints and across collections of frames. For example, body features are selected from a set of at least a neck length, a shoulder angle, and a knee angle.

[0230] In a yet further aspect of the present technology, model fitting can also include rerunning the performing body feature extraction after a range is changed; determining whether the change improved or degraded overall accuracy based upon at least whether number of mislabeled checkpoints increased or decreased; and adjusting ranges until all checkpoints are identified correctly.

[0231] In implementations, videos that are self-created are labelled whereas videos received from other sources may or may not be labelled.

[0232] In a still further aspect of the present technology, the manifest describes frames of each video as being at least one of (i) proper, reflecting that an individual is in a proper state, and (ii) improper, reflecting that an individual is in an improper state.

[0233] In a still yet further aspects of the present technology, the manifest describes the frames of each video as (i) being a start of a repetition, (ii) being an end of a repetition, (iii) including specific keypoints to be evaluated and (iv) including a working side to be evaluated. The keypoints to be evaluated are associated with at least one of a set comprising head, shoulder elbow, knee, foot, any other body feature.

[0234] In yet still further aspects of the present technology, the manifest identifies a plurality of peak checkpoints to be evaluated per repetition as an individual's body moves through repetitions, during a repetition the individual's body moves through a series of these checkpoints. The peak check points can include at least one of a set comprising an initial checkpoint and a motion-peak checkpoint. The motion-peak check is a stopping point in a motion. (e.g., bottom of a squat motion). A checkpoint can comprise a set of keypoints in a known state as a body moves through a series of checkpoints throughout a repetition. The keypoints in a squat include a bottom of squat, certain angles of knees, an angle formed by shoulders, an angle formed at a waist.

[0235] In a still further aspect of the present technology, customization of the payload can be performed using an automated method including receiving from a coach user, adjustments to determined features using a web GUI; determining, by a customization engine, a difference between baseline values and a coach user's version of the base assessment determined using the adjustments as received; extracting from labeled video poses and confidences a set of features, comparing the labeled video poses and confidences against new values determined using the adjustments as received, thereby determining a difference between adjusted values and the baseline. If a range is determined to no longer identify movement checkpoints, reporting an error alerting that a modified value would not meet the checkpoint in the baseline and providing the coach user an opportunity to re-adjust the values and retry. If it is found that all ranges can still properly identify the labeled checkpoints, the customization is finished, and a coach assessment is stored in a database for future use.

[0236] Other implementations may include a non-transitory computer readable storage medium storing instructions executable by a processor to perform actions of the system described above. Yet another implementation may include a method performing actions of the system described above.

[0237] This system implementation and other systems disclosed optionally include one or more of the foregoing features. System can also include features described in connection with methods disclosed. In the interest of conciseness, alternative combinations of system features are not individually enumerated. Features applicable to systems, methods, and articles of manufacture are not repeated for each statutory class set of base features. The reader will understand how features identified in this section can readily be combined with base features in other statutory classes.

[0238] Other implementations may include a non-transitory computer readable storage medium storing instructions executable by a processor to perform actions of the system described above. Each of the features discussed in the particular implementation section for other implementations apply equally to this implementation. As indicated above, all the other features are not repeated here and should be considered repeated by reference.

[0239] The preceding description is presented to enable the making and use of the technology disclosed. Various modifications to the disclosed implementations will be apparent, and the general principles defined herein may be applied to other implementations and applications without departing from the spirit and scope of the technology disclosed. Thus, the technology disclosed is not intended to be limited to the implementations shown, but is to be accorded the widest scope consistent with the principles and features disclosed herein. The scope of the technology disclosed is defined by the appended claims.

	A Response	B Type	C Subcategory	D Subtype	E CurrentErrorType
2	systemMessageKickoffShortintro	systemMessage	kickOff	shortIntro	
3	systemMessageKickoffFirsttimerintro	systemMessage	kickOff	firstTimerIntro	
4	systemMessagePauseToomanyerrors	systemMessage	pause	tooManyErrors	
5	systemMessageTimingCountdown	systemMessage	timing	countdown	
6	systemMessageTimingChime	systemMessage	timing	chime	
7	systemMessageTimingCountdownnochime	systemMessage	timing	countdownNoChime	
8					
9	errorHipslowLongcorrection	error	hipsLow	longCorrection	hipsLow
10	errorHipslowLongcorrection	error	hipsLow	longCorrection	hipsLow
11	errorHipslowLongcorrection	error	hipsLow	longCorrection	hipsLow
12	errorHipslowLongcorrection	error	hipsLow	longCorrection	hipsLow
13	errorHipslowContinuederror	error	hipsLow	continuedError	hipsLow
14	errorHipslowContinuederror	error	hipsLow	continuedError	hipsLow
15	errorHipslowContinuederror	error	hipsLow	continuedError	hipsLow
16	errorHipslowContinuederror	error	hipsLow	continuedError	hipsLow
17	errorHipslowRepeaterror	error	hipsLow	repeatError	hipsLow
18	errorHipslowRepeaterror	error	hipsLow	repeatError	hipsLow
19	errorHipslowRepeaterror	error	hipsLow	repeatError	hipsLow
20	errorHipslowRepeaterror	error	hipsLow	repeatError	hipsLow
21	errorHipslowSofterror	error	hipsLow	softError	hipsLow
22	errorHipslowSofterror	error	hipsLow	softError	hipsLow
23	errorHipslowSofterror	error	hipsLow	softError	hipsLow
24	errorHipslowSofterror	error	hipsLow	softError	hipsLow
25	errorHipshighLongcorrection	error	hipsHigh	longCorrection	hipsHigh
26	errorHipshighLongcorrection	error	hipsHigh	longCorrection	hipsHigh
27	errorHipshighLongcorrection	error	hipsHigh	longCorrection	hipsHigh
28	errorHipshighLongcorrection	error	hipsHigh	longCorrection	hipsHigh
29	errorHipshighContinuederror	error	hipsHigh	continuedError	hipsHigh
30	errorHipshighContinuederror	error	hipsHigh	continuedError	hipsHigh

	F Action	G FeedbackSetting	H EncouragementSetting	I ErrorSetting
2				
3				
4	EndPlank( )			T
5				
6				
7				
8				
9	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
10	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
11	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
12	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
13	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
14	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
15	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
16	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
17	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
18	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
19	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
20	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
21	CheckConfidence( ), CheckErrorPriority( )	T		T
22	CheckConfidence( ), CheckErrorPriority( )	T		T
23	CheckConfidence( ), CheckErrorPriority( )	T		F
24	CheckConfidence( ), CheckErrorPriority( )	T		F
25	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
26	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
27	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F

-continued

28	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
29	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
30	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T

1	J TotalErrorCount	K ErrorConfidence	L OverallConfidence	M CalibrationMsgElapsedTimePlayed	N ChimePlayed	O ChimeElapsedTimePlayed	P GoalTime
2							
3							
4	“>”, 5						
5							
6							
7							
8							
9	“<=”, 5	HIGH					
10	“<=”, 5	HIGH					
11		HIGH					
12		HIGH					
13	“<=”, 5	HIGH					
14	“<=”, 5	HIGH					
15		HIGH					
16		HIGH					
17	“<=”, 5	HIGH					
18	“<=”, 5	HIGH					
19		HIGH					
20		HIGH					
21	“<=”, 5	MED					
22	“<=”, 5	MED					
23		MED					
24		MED					
25	“<=”, 5	HIGH					
26	“<=”, 5	HIGH					
27		HIGH					
28		HIGH					
29	“<=”, 5	HIGH					
30	“<=”, 5	HIGH					

1	Q Current Time	R PreviousHeadTooLow	S PreviousHeadTooHighSeen
2	“==”, 0		
3	“==”, 0		
4			
5	“==”, GoalTime − CountdownAL		
6	“==”, GoalTime − 1		
7	“==”, 99999		
8			
9	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
10	“>”, GoalTime		
11	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
12	“>”, GoalTime		
13	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
14	“>”, GoalTime		
15	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
16	“>”, GoalTime		
17	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
18	“>”, GoalTime		
19	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
20	“>”, GoalTime		
21	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
22			
23	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
24	“>”, GoalTime		
25	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
26	“>”, GoalTime		
27	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
28	“>”, GoalTime		
29	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
30	“>”, GoalTime		

-continued

1	T PreviousShouldersSeen	U PreviousHipsLowSeen	V PreviousHipsHighSeen	W TotalTimesPlanked	X LastErrorElapsedTimePlayed
2				“>=”, 1	
3				“==”, 0	
4					
5					
6					
7					
8					
9		F			
10		F			
11		F			
12		F			
13		T			“>=”, 1; “<=” 4
14		T			“>=”, 1; “<=” 4
15		T			“>=”, 1; “<=” 4
16		T			“>”, 4
17		T			“>”, 4
18		T			“>”, 4
19		T			“>”, 4
20		T			“>”, 4
21					
22					
23					
24					
25			F		
26			F		
27			F		
28			F		
29			T		“>=”, 1; “<=” 4
30			T		“>=”, 1; “<=” 4

1	Y LastAcknowledgementElapsedTimePlayed	Z MostRecentErrorNotSoftError	AA LastRemainingTimeElapsedTimePlayed
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			

1	AB LastStandardTimeMsgElapsedTimePlayed	AC IntroElapsedTimePlayed	AD CurrentStreak	AE SystemMessageDetected	AF ErrorDetected	AG TimeDetected
2						
3						
4						
5						
6						
7						

-continued

8				
9				
10				F
11				F
12				F
13				F
14				F
15				F
16				F
17				F
18				F
19				F
20				F
21				F
22				F
23				F
24				F
25				F
26				F
27				F
28				F
29				F
30				F

1	AH AcknowledgementDetected	AI StateIndicatorDetected	AJ TimeofLastEncour- agementMoreRecentThanReminder	AK GoalTimeMoreThanOneMin
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				

1	AL MusicSetting	AM LastAudioElapsedTimePlayed	AN Priority
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			

-continued

17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	

1	A Response	B Type	C Subcategory	D Subtype	E CurrentErrorType
31	errorHipshighContinuederror	error	hipsHigh	continuedError	hipsHigh
32	errorHipshighContinuederror	error	hipsHigh	continuedError	hipsHigh
33	errorHipshighRepeaterror	error	hipsHigh	repeatError	hipsHigh
34	errorHipshighRepeaterror	error	hipsHigh	repeatError	hipsHigh
35	errorHipshighRepeaterror	error	hipsHigh	repeatError	hipsHigh
36	errorHipshighRepeaterror	error	hipsHigh	repeatError	hipsHigh
37	errorHipshighSofterror	error	hipsHigh	softError	hipsHigh
38	errorHipshighSofterror	error	hipsHigh	softError	hipsHigh
39	errorHipshighSofterror	error	hipsHigh	softError	hipsHigh
40	errorHipshighSofterror	error	hipsHigh	softError	hipsHigh
41	errorShouldersLongcorrection	error	shoulders	longCorrection	shoulders
42	errorShouldersLongcorrection	error	shoulders	longCorrection	shoulders
43	errorShouldersLongcorrection	error	shoulders	longCorrection	shoulders
44	errorShouldersLongcorrection	error	shoulders	longCorrection	shoulders
45	errorShouldersContinuederror	error	shoulders	continuedError	shoulders
46	errorShouldersContinuederror	error	shoulders	continuedError	shoulders
47	errorShouldersContinuederror	error	shoulders	continuedError	shoulders
48	errorShouldersContinuederror	error	shoulders	continuedError	shoulders
49	errorShouldersRepeaterror	error	shoulders	repeatError	shoulders
50	errorShouldersRepeaterror	error	shoulders	repeatError	shoulders
51	errorShouldersRepeaterror	error	shoulders	repeatError	shoulders
52	errorShouldersRepeaterror	error	shoulders	repeatError	shoulders
53	errorShouldersSofterror	error	shoulders	softError	shoulders

1	F Action	G FeedbackSetting	H EncouragementSetting	I ErrorSetting
31	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
32	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
33	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
34	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
35	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
36	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
37	CheckConfidence( ), CheckErrorPriority( )	T		T
38	CheckConfidence( ), CheckErrorPriority( )	T		T
39	CheckConfidence( ), CheckErrorPriority( )	T		F
40	CheckConfidence( ), CheckErrorPriority( )	T		F
41	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
42	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
43	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
44	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
45	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
46	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T

-continued

47	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T	F
48	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T	F
49	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T	T
50	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T	T
51	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T	F
52	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T	F
53	CheckConfidence( ), CheckErrorPriority( )	T	T

1	J TotalErrorCount	K ErrorConfi=dence	L OverallConfi-dence	M Calibra-tionMsgElapsedTimePlayed	N ChimePlayed	O ChimeElapsedTimePlayed	P GoalTime
31		HIGH					
32		HIGH					
33	“<=”, 5	HIGH					
34	“<=”, 5	HIGH					
35		HIGH					
36		HIGH					
37	“<=”, 5	MED					
38	“<=”, 5	MED					
39		MED					
40		MED					
41	“<=”, 5	HIGH					
42	“<=”, 5	HIGH					
43		HIGH					
44		HIGH					
45	“<=”, 5	HIGH					
46	“<=”, 5	HIGH					
47		HIGH					
48		HIGH					
49	“<=”, 5	HIGH					
50	“<=”, 5	HIGH					
51		HIGH					
52		HIGH					
53	“<=”, 5	MED					

1	Q CurrentTime	R PreviousHeadTooLow	S PreviousHeadTooHighSeen
31	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
32	“>”, GoalTime		
33	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
34	“>”, GoalTime		
35	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
36	“>”, GoalTime		
37	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
38	“>”, GoalTime		
39	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
40	“>”, GoalTime		
41	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
42	“>”, GoalTime		
43	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
44	“>”, GoalTime		
45	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
46	“>”, GoalTime		
47	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
48	“>”, GoalTime		
49	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
50	“>”, GoalTime		
51	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
52	“>”, GoalTime		
53	“>”, IntroAL; “<”, GoalTime − CorrectionAL		

1	T PreviousShouldersSeen	U PreviousHipsLowSeen	V PreviousHipsHighSeen	W TotalTimesPlanked	X LastErrorElapsedTimePlayed
31			T		“>=”, 1; “<=” 4
32			T		“>=”, 1; “<=” 4
33			T		“>”, 4
34			T		“>”, 4

-continued

35		T	">", 4
36		T	">", 4
37			
38			
39			
40			
41	F		
42	F		
43	F		
44	F		
45	T		">=", 1; "<=" 4
46	T		">=", 1; "<=" 4
47	T		">=", 1; "<=" 4
48	T		">=", 1; "<=" 4
49	T		">", 4
50	T		">", 4
51	T		">", 4
52	T		">", 4
53			

1	Y LastAcknowledgementElapsedTimePlayed	Z MostRecentErrorNotSoftError	AA LastRemainingTimeElapsedTimePlayed
31			
32			
33			
34			
35			
36			
37			
38			
39			
40			
41			
42			
43			
44			
45			
46			
47			
48			
49			
50			
51			
52			
53			

1	AB LastStan- dardTimeMsgElapsedTimePlayed	AC IntroElapsedTimePlayed	AD CurrentStreak	AE SystemMes- sageDetected	AF ErrorDetected	AG TimeDetected
31				F		
32				F		
33				F		
34				F		
35				F		
36				F		
37				F		
38				F		
39				F		
40				F		
41				F		
42				F		
43				F		
44				F		
45				F		
46				F		
47				F		
48				F		
49				F		
50				F		
51				F		
52				F		
53				F		

-continued

1	AH AcknowledgementDetected	AI StateIndicatorDetected	AJ TimeofLastEncour- agementMoreRecentThanReminder	AK GoalTimeMoreThanOneMin
31				
32				
33				
34				
35				
36				
37				
38				
39				
40				
41				
42				
43				
44				
45				
46				
47				
48				
49				
50				
51				
52				
53				

1	AL MusicSetting	AM LastAudioElapsedTimePlayed	AN Priority
31			
32			
33			
34			
35			
36			
37			
38			
39			
40			
41			
42			
43			
44			
45			
46			
47			
48			
49			
50			
51			
52			
53			

1	A Response	B Type	C Subcategory	D Subtype	E CurrentErrorType
54	errorShouldersSofterror	error	shoulders	softError	shoulders
55	errorShouldersSofterror	error	shoulders	softError	shoulders
56	errorShouldersSofterror	error	shoulders	softError	shoulders
57	errorHeadtoohighLongcorrection	error	headTooHigh	longCorrection	headTooHigh
58	errorHeadtoohighLongcorrection	error	headTooHigh	longCorrection	headTooHigh
59	errorHeadtoohighLongcorrection	error	headTooHigh	longCorrection	headTooHigh
60	errorHeadtoohighLongcorrection	error	headTooHigh	longCorrection	headTooHigh
61	errorHeadtoohighContinuederror	error	headTooHigh	continuedError	headTooHigh
62	errorHeadtoohighContinuederror	error	headTooHigh	continuedError	headTooHigh
63	errorHeadtoohighContinuederror	error	headTooHigh	continuedError	headTooHigh
64	errorHeadtoohighContinuederror	error	headTooHigh	continuedError	headTooHigh
65	errorHeadtoohighRepeaterror	error	headTooHigh	repeatError	headTooHigh
66	errorHeadtoohighRepeaterror	error	headTooHigh	repeatError	headTooHigh
67	errorHeadtoohighRepeaterror	error	headTooHigh	repeatError	headTooHigh
68	errorHeadtoohighRepeaterror	error	headTooHigh	repeatError	headTooHigh

-continued

69	errorHeadtoohighSofterror	error	headTooHigh	softError	headTooHigh
70	errorHeadtoohighSofterror	error	headTooHigh	softError	headTooHigh
71	errorHeadtoohighSofterror	error	headTooHigh	softError	headTooHigh
72	errorHeadtoohighSofterror	error	headTooHigh	softError	headTooHigh
73	errorHeadtoolowLongcorrection	error	headTooLow	longCorrection	headTooLow
74	errorHeadtoolowLongcorrection	error	headTooLow	longCorrection	headTooLow
75	errorHeadtoolowLongcorrection	error	headTooLow	longCorrection	headTooLow
76	errorHeadtoolowLongcorrection	error	headTooLow	longCorrection	headTooLow
77	errorHeadtoolowContinuederror	error	headTooLow	continuedError	headTooLow

1	F Action	G FeedbackSetting	H EncouragementSetting	I ErrorSetting
54	CheckConfidence( ), CheckErrorPriority( )	T		T
55	CheckConfidence( ), CheckErrorPriority( )	T		F
56	CheckConfidence( ), CheckErrorPriority( )	T		F
57	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
58	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
59	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
60	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
61	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
62	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
63	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
64	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
65	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
66	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
67	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
68	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
69	CheckConfidence( ), CheckErrorPriority( )	T		T
70	CheckConfidence( ), CheckErrorPriority( )	T		T
71	CheckConfidence( ), CheckErrorPriority( )	T		F
72	CheckConfidence( ), CheckErrorPriority( )	T		F
73	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
74	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
75	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
76	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
77	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T

1	J TotalErrorCount	K ErrorConfi-dence	L OverallConfi-dence	M Calibra-tionMsgElapsedTimePlayed	N ChimePlayed	O ChimeElapsedTimePlayed	P GoalTime
54	"<=", 5	MED					
55		MED					
56		MED					
57	"<=", 5	HIGH					
58	"<=", 5	HIGH					
59		HIGH					
60		HIGH					
61	"<=", 5	HIGH					
62	"<=", 5	HIGH					
63		HIGH					
64		HIGH					
65	"<=", 5	HIGH					
66	"<=", 5	HIGH					
67		HIGH					
68		HIGH					
69	"<=", 5	MED					
70	"<=", 5	MED					

-continued

71		MED
72		MED
73	“<=”, 5	HIGH
74	“<=”, 5	HIGH
75		HIGH
76		HIGH
77	“<=”, 5	HIGH

1	Q CurrentTime	R PreviousHeadTooLow	S PreviousHeadTooHighSeen
54	“>”, GoalTime		
55	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
56	“>”, GoalTime		
57	“>”, IntroAL; “<”, GoalTime − CorrectionAL		F
58	“>”, GoalTime		F
59	“>”, IntroAL; “<”, GoalTime − CorrectionAL		F
60	“>”, GoalTime		F
61	“>”, IntroAL; “<”, GoalTime − CorrectionAL		T
62	“>”, GoalTime		T
63	“>”, IntroAL; “<”, GoalTime − CorrectionAL		T
64	“>”, GoalTime		T
65	“>”, IntroAL; “<”, GoalTime − CorrectionAL		T
66	“>”, GoalTime		T
67	“>”, IntroAL; “<”, GoalTime − CorrectionAL		T
68	“>”, GoalTime		T
69	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
70	“>”, GoalTime		
71	“>”, IntroAL; “<”, GoalTime − CorrectionAL		
72	“>”, GoalTime		
73	“>”, IntroAL; “<”, GoalTime − CorrectionAL	F	
74	“>”, GoalTime	F	
75	“>”, IntroAL; “<”, GoalTime − CorrectionAL	F	
76	“>”, GoalTime	F	
77	“>”, IntroAL; “<”, GoalTime − CorrectionAL	T	

1	T PreviousShouldersSeen	U PreviousHipsLowSeen	V PreviousHipsHighSeen	W TotalTimesPlanked	X LastErrorElapsedTimePlayed
54					
55					
56					
57					
58					
59					
60					
61					“>=”, 1; “<=” 4
62					“>=”, 1; “<=” 4
63					“>=”, 1; “<=” 4
64					“>=”, 1; “<=” 4
65					“>”, 4
66					“>”, 4
67					“>”, 4
68					“>”, 4
69					
70					
71					
72					
73					
74					
75					
76					
77					“>=”, 1; “<=”, 4

1	Y LastAcknowledgementElapsedTimePlayed	Z MostRecentErrorNotSoftError	AA LastRemainingTimeElapsedTimePlayed
54			
55			
56			
57			
58			
59			
60			
61			
62			

-continued

63						
64						
65						
66						
67						
68						
69						
70						
71						
72						
73						
74						
75						
76						
77						

1	AB LastStandardTimeMsgElapsedTimePlayed	AC IntroElapsedTimePlayed	AD CurrentStreak	AE SystemMessageDetected	AF ErrorDetected	AG TimeDetected
54				F		
55				F		
56				F		
57				F		
58				F		
59				F		
60				F		
61				F		
62				F		
63				F		
64				F		
65				F		
66				F		
67				F		
68				F		
69				F		
70				F		
71				F		
72				F		
73				F		
74				F		
75				F		
76				F		
77				F		

1	AH AcknowledgementDetected	AI StateIndicatorDetected	AJ TimeofLastEncouragementMoreRecentThanReminder	AK GoalTimeMoreThanOneMin
54				
55				
56				
57				
58				
59				
60				
61				
62				
63				
64				
65				
66				
67				
68				
69				
70				
71				
72				
73				
74				
75				
76				
77				

-continued

1	AL MusicSetting	AM LastAudioElapsedTimePlayed	AN Priority
54			
55			
56			
57			
58			
59			
60			
61			
62			
63			
64			
65			
66			
67			
68			
69			
70			
71			
72			
73			
74			
75			
76			
77			

1	A Response	B Type	C Subcategory	D Subtype	E CurrentErrorType
78	errorHeadtoolowContinuederror	error	headTooLow	continuedError	headTooLow
79	errorHeadtoolowContinuederror	error	headTooLow	continuedError	headTooLow
80	errorHeadtoolowContinuederror	error	headTooLow	continuedError	headTooLow
81	errorHeadtoolowRepeaterror	error	headTooLow	repeatError	headTooLow
82	errorHeadtoolowRepeaterror	error	headTooLow	repeatError	headTooLow
83	errorHeadtoolowRepeaterror	error	headTooLow	repeatError	headTooLow
84	errorHeadtoolowRepeaterror	error	headTooLow	repeatError	headTooLow
85	errorHeadtoolowSofterror	error	headTooLow	softError	headTooLow
86	errorHeadtoolowSofterror	error	headTooLow	softError	headTooLow
87	errorHeadtoolowSofterror	error	headTooLow	softError	headTooLow
88	errorHeadtoolowSofterror	error	headTooLow	softError	headTooLow
89					
90	timeStandardtime	time	standardTime		
91	timeBackuptime	time	backUpTime		
92	timeBackuptime	time	backUpTime		
93	timeBackuptime	time	backUpTime		
94	timeBackuptime	time	backUpTime		
95	timeRemainingtime	time	remainingTime		
96					
97	acknowledgement	acknowledgement			
98	acknowledgement	acknowledgement			
99	acknowledgement	acknowledgement			
100	acknowledgement	acknowledgement			
101					
102	statusIndicatorGoalExceedinggoal	statusIndicator	goal	exceedingGoal	
103	statusIndicatorPRNewpr	statusIndicator	pr	newPr	
104	statusIndicatorPrevioustimeBeatprevioustime	statusIndicator	previousTime	beatPreviousTime	
105	statusIndicatorStreaksTwoday	statusIndicator	streaks	twoDay	
106	statusIndicatorStreaksThreeday	statusIndicator	streaks	threeDay	
107	statusIndicatorStreaksFiveday	statusIndicator	streaks	fiveDay	
108	statusIndicatorStreaksTenday	statusIndicator	streaks	tenDay	
109	statusIndicatorStreaksFifteenday	statusIndicator	streaks	fifteenDay	
110	statusIndicatorStreaksTwentyday	statusIndicator	streaks	twentyDay	
111	statusIndicatorStreaksOnemonth	statusIndicator	streaks	oneMonth	

1	F Action	G FeedbackSetting	H EncouragementSetting	I ErrorSetting
78	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
79	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F

-continued

80	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
81	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
82	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		T
83	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
84	CheckConfidence( ), CheckNewError( ), CheckErrorPriority( ), TotalErrorCount += 1	T		F
85	CheckConfidence( ), CheckErrorPriority( )	T		T
86	CheckConfidence( ), CheckErrorPriority( )	T		T
87	CheckConfidence( ), CheckErrorPriority( )	T		F
88	CheckConfidence( ), CheckErrorPriority( )	T		F
89				
90				
91				
92				
93				
94				
95				
96				
97		T		
98		T		
99		T		
100		T		
101				
102				
103				
104				
105				
106				
107				
108				
109				
110				
111				

1	J TotalErrorCount	K ErrorConfidence	L OverallConfidence	M CalibrationMsgElapsedTimePlayed	N ChimePlayed	O ChimeElapsedTimePlayed	P GoalTime
78	“<=”, 5	HIGH					
79		HIGH					
80		HIGH					
81	“<=”, 5	HIGH					
82	“<=”, 5	HIGH					
83		HIGH					
84		HIGH					
85	“<=”, 5	MED					
86	“<=”, 5	MED					
87		MED					
88		MED					
89							
90							
91							
92							
93							
94							
95							
96							
97		HIGH					
98		HIGH					
99		HIGH					
100		HIGH					
101							
102							
103							
104							
105							
106							
107							

-continued

108					
109					
110					
111					

1	Q Current Time	R PreviousHeadTooLow	S PreviousHeadTooHighSeen
78	">", GoalTime	T	
79	">", IntroAL; "<", GoalTime − CorrectionAL	T	
80	">", GoalTime	T	
81	">", IntroAL; "<", GoalTime − CorrectionAL	T	
82	">", GoalTime	T	
83	">", IntroAL; "<", GoalTime − CorrectionAL	T	
84	">", GoalTime	T	
85	">", IntroAL; "<", GoalTime − CorrectionAL		
86	">", GoalTime		
87	">", IntroAL; "<", GoalTime − CorrectionAL		
88	">", GoalTime		
89			
90	"!=", GoalTime − 30 && "==", StandardTime && "!=", GoalTime		
91	"!=", GoalTime − 30 && "==", BackupTime && "!=", GoalTime		
92	"!=", GoalTime − 30 && "==", BackupTime && "!=", GoalTime		
93	"!=", GoalTime − 30 && "==", BackupTime && "!=", GoalTime		
94	"!=", GoalTime − 30 && "==", BackupTime && "!=", GoalTime		
95	"==", GoalTime − 30		
96			
97	">", 0; "<", GoalTime − 4		
98	">", 0; "<", GoalTime − 4		
99	">", GoalTime		
100	">", GoalTime		
101			
102	"==", GoalTime + 10		
103	"==", PR + 5 && "!=", GoalTime + 10		
104	"==", PreviousTime + 5 && "!=", PR + 5 && "!=", GoalTime + 10		
105			
106			
107			
108			
109			
110			
111			

1	T PreviousShouldersSeen	U PreviousHipsLowSeen	V PreviousHipsHighSeen	W TotalTimesPlanked	X LastErrorElapsedTimePlayed
78					">=", 1; "<=" 4
79					">=", 1; "<=" 4
80					">=", 1; "<=" 4
81					">", 4
82					">", 4
83					">", 4
84					">", 4
85					
86					
87					
88					
89					
90					
91					
92					
93					
94					
95					
96					
97					">=", 1; "<=" 4
98					">=", 1; "<=" 4
99					">=", 1; "<=" 4
100					">=", 1; "<=" 4
101					
102					
103				">", 0	
104					
105					
106					
107					

-continued

108						
109						
110						
111						

1	Y LastAcknowledgementElapsedTimePlayed	Z MostRecentErrorNotSoftError	AA LastRemainingTimeElapsedTimePlayed
78			
79			
80			
81			
82			
83			
84			
85			
86			
87			
88			
89			
90			
91			“==”, 0-1
92			“>”, 40
93			“==”, 0-1
94			“>”, 40
95			
96			
97	“>”, 5	T	
98	“==”, 0-1	T	
99	“>”, 5	T	
100	“==”, 0-1	T	
101			
102			
103			
104			
105			
106			
107			
108			
109			
110			
111			

1	AB LastStan-dardTimeMsgElapsedTimePlayed	AC IntroElapsedTimePlayed	AD CurrentStreak	AE SystemMes-sageDetected	AF ErrorDetected	AG TimeDetected
78				F		
79				F		
80				F		
81				F		
82				F		
83				F		
84				F		
85				F		
86				F		
87				F		
88				F		
89						
90				F	F	
91	“>”, 40			F	F	
92	“>”, 40			F	F	
93	“==”, 0-1			F	F	
94	“==”, 0-1			F	F	
95				F	F	
96						
97				F	F	F
98				F	F	F
99				F	F	F
100				F	F	F
101						
102				F	F	F
103				F	F	F
104				F	F	F
105		“>”, 1; “<” 3	“==”, 1	F	F	F
106		“>”, 1; “<” 3	“==”, 2	F	F	F

-continued

107		">", 1; "<" 3	"==", 4	F	F	F
108		">", 1; "<" 3	"==", 9	F	F	F
109		">", 1; "<" 3	"==", 14	F	F	F
110		">", 1; "<" 3	"==", 19	F	F	F
111		">", 1; "<" 3	"==", 29	F	F	F

1	AH AcknowledgementDetected	AI StateIndicatorDetected	AJ TimeofLastEncour-agementMoreRecentThanReminder	AK GoalTimeMoreThanOneMin
78				
79				
80				
81				
82				
83				
84				
85				
86				
87				
88				
89				
90				
91				
92				
93				
94				
95				
96				
97				
98				
99				
100				
101				
102	F			
103	F			
104	F			
105	F			
106	F			
107	F			
108	F			
109	F			
110	F			
111	F			

1	AL MusicSetting	AM LastAudioElapsedTimePlayed	AN Priority
78			
79			
80			
81			
82			
83			
84			
85			
86			
87			
88			
89			
90			
91			
92			
93			
94			
95			
96			
97			
98			
99			
100			
101			
102			T
103			
104			
105			

-continued

106	
107	
108	
109	
110	
111	

1	A Response	B Type	C Subcategory	D Subtype	E CurrentErrorType
112					
113	encouragementFirstthirtyseconds	encouragement	firstThirtySeconds		
114	encouragementFirstthirtyseconds	encouragement	firstThirtySeconds		
115	encouragementLastthirtyseconds	encouragement	lastThirtySeconds		
116	encouragementInbetween	encouragement	inBetween		
117	encouragementInbetween	encouragement	inBetween		
118	encouragementInbetween	encouragement	inBetween		
119	encouragementInbetween	encouragement	inBetween		
120	encouragementInbetween	encouragement	inBetween		
121	encouragementInbetween	encouragement	inBetween		
122	encouragementInbetween	encouragement	inBetween		
123	encouragementInbetween	encouragement	inBetween		
124	encouragementInbetween	encouragement	inBetween		
125	encouragementInbetween	encouragement	inBetween		
126	encouragementAfterchime	encouragement	afterChime		
127					
128	reminderGeneral	reminder	general		
129	reminderGeneral	reminder	general		
130	reminderGeneral	reminder	general		
131	reminderGeneral	reminder	general		
132	reminderGeneral	reminder	general		
133	reminderGeneral	reminder	general		
134	reminderGeneral	reminder	general		
135	reminderGeneral	reminder	general		
136	reminderGeneral	reminder	general		
137	reminderGeneral	reminder	general		
138	reminderGeneral	reminder	general		
139	reminderGeneral	reminder	general		
140	reminderOveronemin	reminder	overOneMin		
141	reminderOveronemin	reminder	overOneMin		

1	F Action	G FeedbackSetting	H EncouragementSetting	I ErrorSetting
112				
113			T	
114			T	
115			T	
116			T	
117			T	
118			T	
119			T	
120			T	
121			T	
122			T	
123			T	
124			T	
125			T	
126			T	
127				
128		T		
129		T		
130		T		
131		T		
132		T		
133		T		
134		T		
135		T		
136		T		

-continued

137	T	
138	T	
139	T	
140	T	
141	T	

1	J TotalErrorCount	K ErrorConfi-dence	L OverallConfi-dence	M Calibra-tionMsgElapsedTimePlayed	N ChimePlayed	O ChimeElapsedTimePlayed	P GoalTime
112							
113							“<=”, 30
114							“>”, 30
115							
116							
117							
118							“>”, 30
119							“>”, 30
120							
121							
122							“>”, 30
123							“>”, 30
124							
125							
126					T	“==”, 2	
127							
128							
129							
130							
131							
132							“<=”, 30
133							“>”, 30
134							“>”, 30
135							“<=”, 30
136							“>”, 30
137							“>”, 30
138							
139							
140							
141							

1	Q Current Time	R PreviousHeadTooLow	S PreviousHeadTooHighSeen
112			
113	“>”, IntroAL && GoalTime − EncouragementAL − CountdownAL		
114	“>”, IntroAL && 25 − EncouragementAL		
115	“>”, GoalTime − 30 && “>”, GoalTime − CountdownAL − EncouragementAL		
116	“>”, IntroAL && GoalTime − EncouragementAL − CountdownAL && “!=”, GoalTime − 30 − EncouragementAL		
117	“>”, IntroAL && GoalTime − EncouragementAL − CountdownAL && “!=”, GoalTime − 30 − EncouragementAL		
118	“>”, 30 && GoalTime − EncouragementAL − CountdownAL && “!=”, GoalTime − 30 − EncouragementAL		
119	“>”, 30 && GoalTime − EncouragementAL − CountdownAL && “!=”, GoalTime − 30 − EncouragementAL		
120	“>”, GoalTime		
121	“>”, GoalTime		
122	“>”, GoalTime		
123	“>”, GoalTime		
124	“>”, 300		
125	“>”, 300		
126			
127			
128	“>”, IntroAL && “>”, GoalTime − ReminderAL − CountdownAL && “!=”, GoalTime − 30 − ReminderAL		
129	“>”, IntroAL && “>”, GoalTime − ReminderAL − CountdownAL && “!=”, GoalTime − 30 − ReminderAL		
130	“>”, GoalTime		
131	“>”, GoalTime		
132	“>”, IntroAL && “>”, GoalTime − ReminderAL − CountdownAL && “!=”, GoalTime − 30 − ReminderAL		
133	“>”, IntroAL && “>”, GoalTime − ReminderAL − CountdownAL && “!=”, GoalTime − 30 − ReminderAL		

-continued

134	">", GoalTime
135	">", IntroAL && ">", GoalTime − ReminderAL − CountdownAL && "!=", GoalTime − 30 − ReminderAL
136	">", IntroAL && ">", GoalTime − ReminderAL − CountdownAL && "!=", GoalTime − 30 − ReminderAL
137	">", GoalTime
138	">", 300
139	">", 300
140	">", 60; ">", 120
141	">", 60; ">", 120

1	T PreviousShouldersSeen	U PreviousHipsLowSeen	V PreviousHipsHighSeen	W TotalTimesPlanked	X LastErrorElapsedTimePlayed
112					
113					
114					
115					
116					
117					
118					
119					
120					
121					
122					
123					
124					
125					
126					
127					
128					
129					
130					
131					
132					
133					
134					
135					
136					
137					
138					
139					
140					
141					

1	Y LastAcknowledgementElapsedTimePlayed	Z MostRecentErrorNotSoftError	AA LastRemainingTimeElapsedTimePlayed
112			
113			
114			
115			
116			
117			">", 0; "<", 25 − EncouragementAL
118			
119			
120			
121			">", 0; "<", 28 − EncouragementAL
122			
123			
124			
125			
126			
127			
128			
129			">", 0; "<", 25 − ReminderAL
130			
131			">", 0; "<", 28 − ReminderAL
132			
133			
134			
135			
136			
137			

-continued

138	
139	
140	
141	

1	AB LastStandardTimeMsgElapsedTimePlayed	AC IntroElapsedTimePlayed	AD CurrentStreak	AE SystemMessageDetected	AF ErrorDetected	AG TimeDetected
112						
113				F	F	F
114				F	F	F
115				F	F	F
116	“>”, 0; “<”, 2			F	F	F
117				F	F	F
118	“>”, 0; “<”, 25 – EncouragementAL			F	F	F
119	“>”, 0; “<”, 25 – EncouragementAL			F	F	F
120	“>”, 0; “<”, 5			F	F	F
121				F	F	F
122	“>”, 0; “<”, 25 – EncouragementAL			F	F	F
123	“>”, 0; “<”, 25 – EncouragementAL			F	F	F
124				F	F	F
125				F	F	F
126				F	F	F
127						
128	“>”, 8; “<”, 15			F	F	F
129				F	F	F
130	“>”, 8; “<”, 20			F	F	F
131				F	F	F
132				F	F	F
133	“>”, 0; “<”, 25 – ReminderAL			F	F	F
134	“>”, 0; “<”, 25 – ReminderAL			F	F	F
135				F	F	F
136	“>”, 0; “<”, 25 – ReminderAL			F	F	F
137	“>”, 0; “<”, 25 – ReminderAL			F	F	F
138				F	F	F
139				F	F	F
140	“>”, 0; “<”, 25 – ReminderAL			F	F	F
141	“>”, 0; “<”, 25 – ReminderAL			F	F	F

1	AH AcknowledgementDetected	AI StateIndicatorDetected	AJ TimeofLastEncouragementMoreRecentThanReminder	AK GoalTimeMoreThanOneMin
112				
113	F	F		
114	F	F		
115	F	F		T
116	F	F	F	
117	F	F	F	
118	F	F	F	
119	F	F	F	
120	F	F	F	
121	F	F	F	
122	F	F	F	
123	F	F	F	
124	F	F	F	
125	F	F	F	
126	F	F		
127				
128	F	F	T	
129	F	F	T	
130	F	F	T	
131	F	F	T	
132	F	F	T	
133	F	F	T	
134	F	F	T	
135	F	F	T	
136	F	F	T	
137	F	F	T	
138	F	F	T	
139	F	F	T	
140	F	F	T	
141	F	F	T	

-continued

1	AL MusicSetting	AM LastAudioElapsedTimePlayed	AN Priority
112			
113		">", 2	
114		">", 2	
115		">", 5	
116			
117		">", 10	
118	F	">", 10	
119	T	">", 20	
120			
121		">", 10	
122	F	">", 10	
123	T	">", 20	
124	F	">", 10	
125	T	">", 20	
126			
127			
128			
129		">", 8	
130			
131		">", 8	
132	F	">", 10	
133	F	">", 10	
134	F	">", 10	
135	T	">", 20	
136	T	">", 20	
137	T	">", 20	
138	F	">", 10	
139	T	">", 20	
140	F	">", 10	
141	T	">", 20	

What is claimed is:

1. A method of conducting evaluation of collected facts about performance of a task and determining output instructions for a user performing the task, the method comprising:

receiving state information comprising a set of collected facts describing a user pose state, including (i) at least one static fact that is constant over a time-period in which at least the task is performed and (ii) at least one dynamic fact based on an amount of time that has elapsed since a last error was detected;

upon lapse of a periodic timer, in a first process:

evaluating facts in the set of collected facts as received to determine a response message to be output as instructions for performing the task, by weighting at least some facts with dynamic weightings;

selecting based on the at least some facts as dynamically weighted, a response message to be output; and

performing the output response message as selected, and capturing results for evaluation as historical outcomes; and

in a second process:

evaluating by a feedback engine historical outcomes, wherein an outcome is a consequence of a combination of facts evaluated and response message(s) played to the user, of a sample set of previous outcomes, thereby identifying a time between similar outcomes; and

applying a machine learning process to the time between similar outcomes to obtain an improved selection of response messages to obtain similar outcomes exhibiting a desired result; and

storing dynamic weights in a database to personalize task performance training to the user thereby bringing about desired outcome for that user.

2. The method of claim **1**, wherein whenever the user is in an improper position as determined using information from a pose engine, state information received further includes a label of error to present to the user wherein each label of error comprises one or more audio, video or other output-type files from which feedback is selected for output to the user while being assessed on a movement.

3. The method of claim **1**, wherein dynamic facts are selected from a set including at least (i) an amount of time that has elapsed since a last error was observed, (ii) a repetition or timestamp since an event or a timer started or a time since midnight selected from a number of times an error has been previously observed.

4. The method of claim **1**, wherein constant facts are quantities determinable at session start of the time period in which the task is to be performed selected from a set including at least a length of a session in which tasks are to be performed, data collected and feedback given, a number of tasks to be performed, and data collected and feedback given.

5. The method of claim **1**, wherein duration of the timer is variable and dependent on an action type of the task to be performed, in a range of 0.85 seconds corresponding to higher repletion rate actions to 5 minutes corresponding to lower repetition rate actions.

6. The method of claim **1**, wherein weighting facts with dynamic weightings includes:

applying a weight to one or more dynamic features such that output of the fact will be true if (i) the fact is true, and then (ii) if an evenly distributed random variable

with value in a range of 0 to 1 is greater than or equal to the weight, thereby enabling varying feedback selected for a particular fact.

7. The method of claim **6**, further comprising at beginning of a session setting dynamic weights are to random values if there is no historical values for the weight values.

8. The method of claim **6**, wherein selecting a response to be output includes:

determining an appropriate response message type using the weighted dynamic features, including:

(i) none when none exists,

(ii) selecting an output message based upon output type in a set of at least audio message, visual message, and

(iii) when for each message type, there exist multiple variants of recorded responses from which to choose selecting based upon message type in a set of at least specific message—too high, specific message—encouragement, specific message—warning, and specific message—termination message; and

automatically selecting for output a selected audio output response message having the response message type as determined.

9. The method of claim **8**, further comprising initially choosing an audio output at random and playing the audio output as chosen to the user and storing a time at which the audio was played and a corresponding message for future reference.

10. The method of claim **1**, wherein applying a machine learning process further includes:

evaluating time between two errors reported to the user by fitting a curve to data points representing previous results for the user; and

applying association rules and linear regression to maximize time/reps between errors, using gradient descent, variable times, max time between consecutive errors to provide coaching to a user.

11. The method of claim **1**, wherein applying a machine learning process further includes:

using historical data from a plurality of previous sessions to adjust the dynamic weights; and

storing the dynamic weights as adjusted to be used in subsequent executions of the method.

12. The method of claim **11**, wherein a probability of a particular type of response to a message is used to weigh future selections of responses.

13. The method of claim **12**, wherein the probability of a particular type of response is one of a set comprising: an error every time, a person responds to the message with a successful outcome, a person does not respond well to the message.

14. The method of claim **12**, wherein the probability of a particular type of response is one of a set task related facts comprising: resistance, repetitions and format.

15. The method of claim **11**, wherein historical data from at least 10 sessions is used.

16. The method of claim **1**, wherein dynamic weights are stored on a device of the user, thereby protecting user privacy.

17. The method of claim **1**, wherein facts for determining correctness of task performance are gathered by a server and automatically labeled using machine learning processes by:

performing video analysis, including:

obtaining a manifest and corresponding recorded videos of individuals performing particular movements in proper states (correct form) and in improper states (incorrect form),

wherein the videos that are self-created are labelled or videos found from other sources that may or may not be labelled,

wherein the manifest describes frames of each video as being at least one of (i) proper, reflecting that an individual is in a proper state, and (ii) improper, reflecting that an individual is in an improper state,

wherein the manifest further describes the frames of each video as (i) being a start of a repetition, (ii) being an end of a repetition, (iii) including specific keypoints comprising at least one of head, shoulder, knee, and elbow, to be evaluated and (iv) including a working side to be evaluated,

wherein the manifest identifies a number of peak checkpoints to be evaluated per repetition including at least an initial checkpoint and a peak checkpoint based upon a position of a motion as an individual's body moves through repetitions, during a repetition the individual's body moves through a series of these checkpoints,

wherein a difference between a keypoint and coordinates thereof and checkpoint is that checkpoints include collection of keypoints in a known state including at least one of keypoints in identified position in a movement, certain angles of one or more of knees, shoulders, and waist, as an individual's body moves through a series of checkpoints throughout a repetition,

wherein the manifest can include information indicating, in a range of from 2 to 6 seconds or from frames in a range of 60 to 360 frames that position of a particular body part of an individual is within or outside of a tolerance,

extracting portions of the videos for evaluation, while maintaining the descriptions in the manifest of the frames of the extracted portions of the videos;

inputting, into a pose estimation neural network, the extracted portions of the videos one frame at a time;

receiving, as an output of the pose estimation neural network and for each input frame, a pose comprising a collection of the keypoints in the frame, including (i) coordinates of one or more keypoints in the frame and (ii) confidences representing a confidence that each keypoint is a particular feature of each of the one or more evaluation points; and

outputting labeled payloads of poses and confidences for each frame of the extracted portions of the videos,

wherein a labeled payload can indicate (i) that this frame includes a pose of someone leaning over, and poses of the keypoints, (ii) the confidences of the keypoints, and (iii) an aggregate of confidences over all keypoints for a particular repetition,

wherein one or more confidences used to weight some keypoints at joints, such that, confidences of a keypoint at a first joint permeate through labeling of at least one other of the keypoints and across collections of frames,

whenever the labels in the manifest are wrong, reconciling and validating the labels that were so determined,

thereby for slices of videos, providing in the labeled payload information indicating (i) whether the body is in the correct or incorrect position/state, (ii) keypoint information and (iii) confidence information;

performing movement analysis, including:

identifying or selecting a particular movement;

identifying or selecting a video associated with particular movement, and a corresponding manifest;

examining the corresponding manifest of the video to determine a candidate list of body features,

wherein, a body feature includes an angle between a first body part and a second body part that is determined using keypoints that can be used to evaluate a particular movement,

wherein the body feature can be derived from the keypoints and relationships including distances, and angles therebetween;

automatically selecting, from the candidate list of body features, body features including one or more of at least a neck length, a shoulder angle, and a body measurement, and that are in the candidate list and related to the keypoints;

for each pose and confidence in the payload, extracting the relevant body features;

using the manifests, extracting checkpoints across each input video;

determining relevant ranges of values for each identified body features, thereby resulting in a list having form: "checkpoint_initial", "checkpoint_repetition" and "checkpoint_[error—hips too high]_state" for each video or portion thereof; and

providing recommendations including (i) ranges for particular body features that are acceptable for a particular movement, for model fitting along with the manifest, poses and confidences for each video,

thereby forming a collection of ranges from minimum to maximum that are essentially acceptable for the particular movement;

performing model fitting to determine which keypoints and/or body features relevant to determining whether a particular posture is correct, including:

performing body feature extraction for a video using the poses and confidences obtained,

whereby at each labeled checkpoint of the video, recommendations are compared to determine if the checkpoint is being modelled properly,

whereby, all of the data needed to make a first estimate is available, thereby enabling for each frame, determining an estimate whether the pose at the checkpoint is proper or improper;

if all checkpoints are identified correctly based on the estimate, then the performing of the model fitting is complete;

if checkpoints were mislabeled based on estimating that the pose at the checkpoint is improper, then an iterative machine learning process, including a grid search, is performed to adjust ranges of the body features until each of the checkpoints is identified properly, whereby the estimate resulting matches what the manifest states,

after a range is changed, the performing of the body feature extraction from above is reran to see if the change improved or degraded the overall accuracy, then the ranges are adjusted until all checkpoints are identified correctly; and

storing a base assessment for the particular movement in a database once the performing of the model fitting determines that the poses at all of the checkpoints are identified correctly, the base assessment includes identified baselines for best case scenarios for each pose/movement.

18. The method of claim **17**, further including performing customization, including:

receiving from a coach user, adjustments to determined features using a web GUI;

determining, by a customization engine, a difference between base values and a coach user's version of the assessment;

extracting from labeled video poses and confidences a set of features, comparing the labeled video poses and confidences against the new values determined using the adjustments as received, thereby determining a difference between the adjusted values and the baseline;

if a range is determined to no longer identify movement checkpoints, reporting an error alerting that a modified value would not meet the checkpoint in the baseline and providing the coach user an opportunity to re-adjust the values and retry; and

if it is found that all ranges can still properly identify the labeled checkpoints, finishing the customization and storing a coach assessment in a database for future use.

19. The method of claim **1**, wherein the frames of the videos are labelled using a neural network trained for human pose estimation that identifies coordinates of key points of individuals.

20. A method of performing video analysis, the method comprising:

capturing a live-stream or recorded video in color or B&W format;

scaling frames to appropriate size for model input;

extracting a plurality of keypoints that describe areas of interest on the body;

smoothing a current pose using a digital signal processing (DSP) functionality to prevent perceived jumpiness in the video image when displayed using a mobile or small footprint device;

given a current exercise being taught, extracting features from the relevant assessment rules;

comparing a current set of keypoints as extracted to features extracted from the relevant assessment rules to determine a current state of a user's pose; and

given the determined state, identifying a message or a NULL message to present or play for the user.

21. A system comprising:

a memory storing instructions; and

a processor, coupled with the memory and to execute the instructions, the instructions when executed cause the processor to perform the method of claim **1**.

22. A non-transitory computer readable medium comprising stored instructions, which when executed by a processor, cause the processor to perform the method of claim **1**.

23. The method of claim **18**, wherein movement checkpoints outside of a range are displayed in red by the

graphical user interface (GUI) and movement checkpoints within a range are displayed in green by the graphical user interface (GUI).

* * * * *