

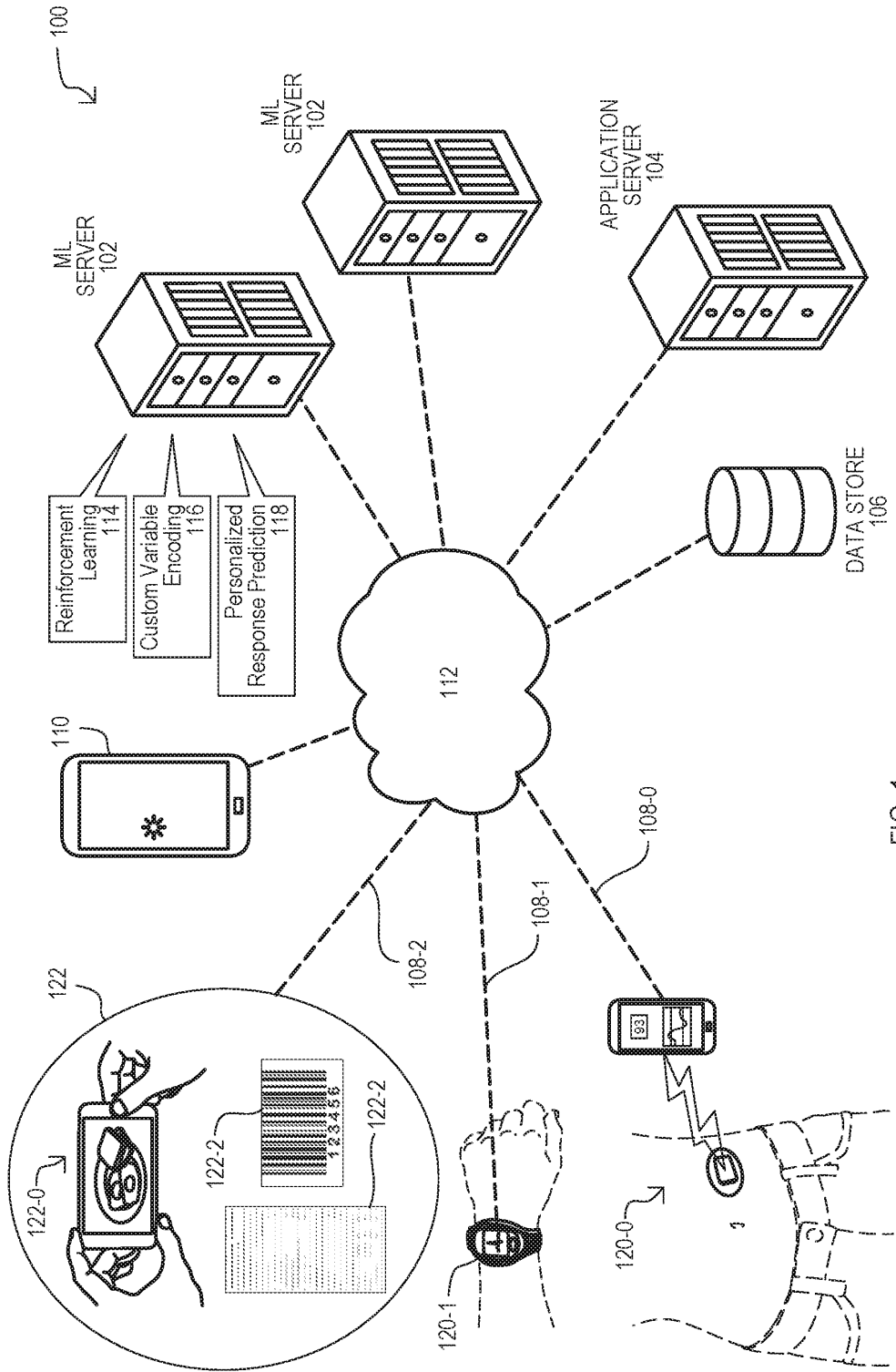


FIG. 1

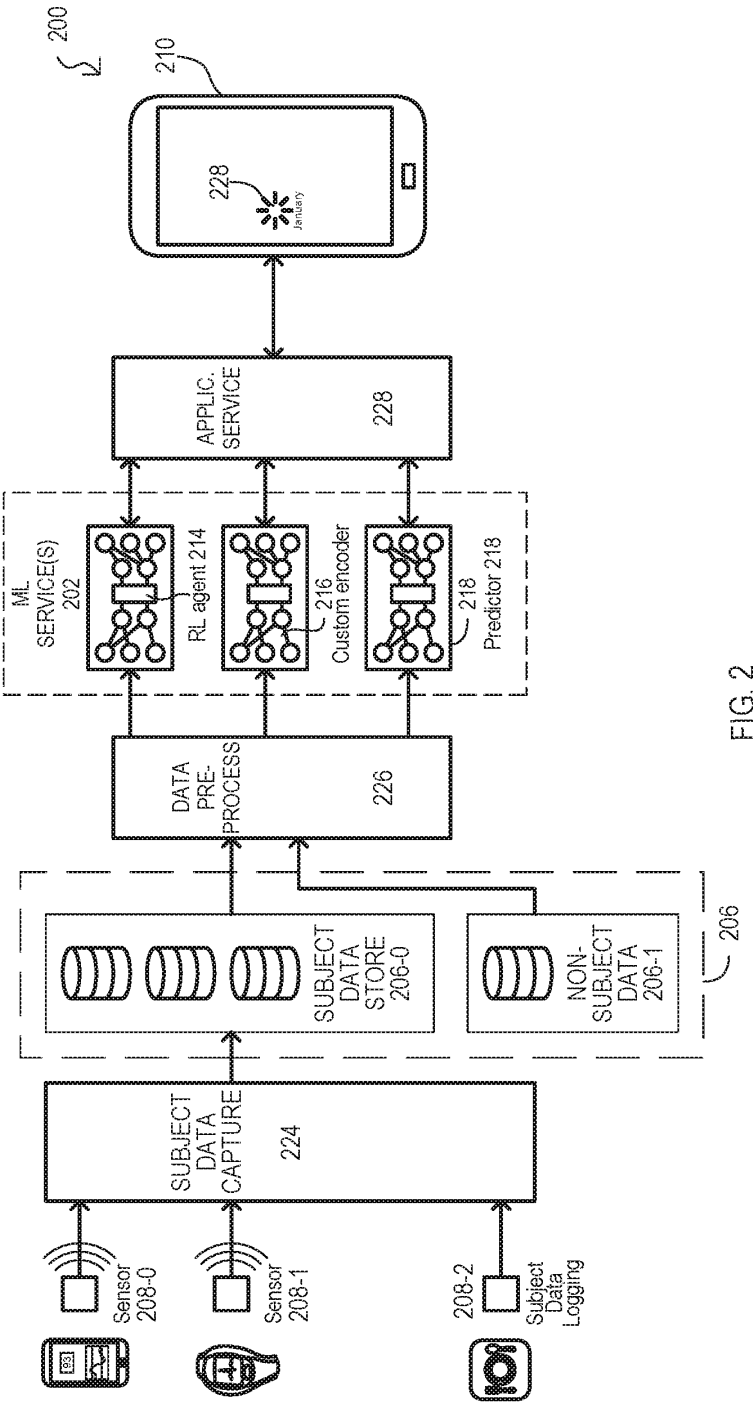


FIG. 2

3/45

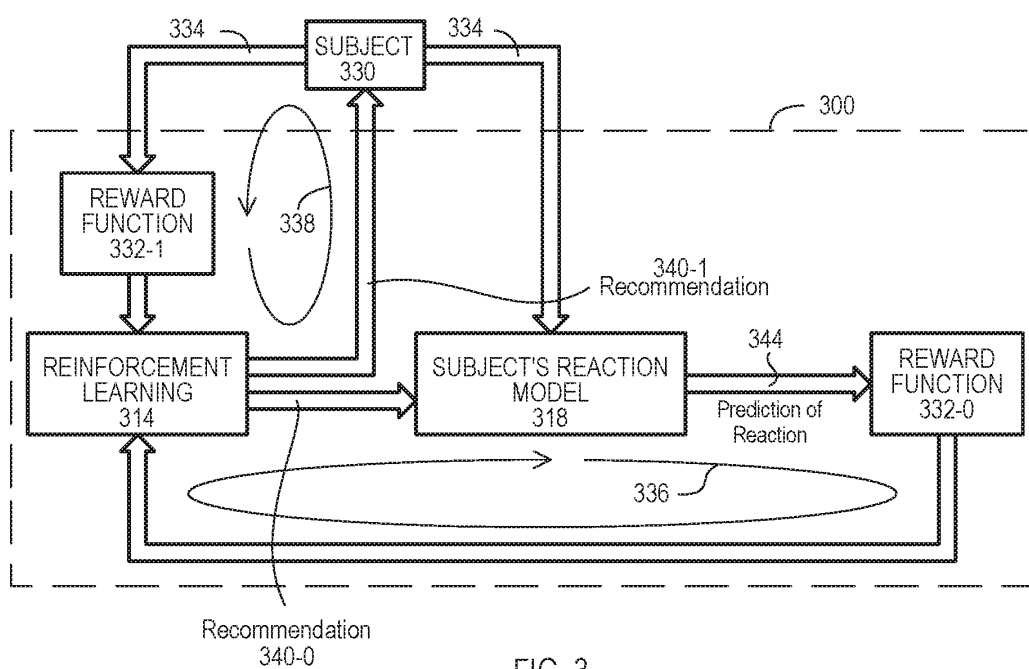


FIG. 3

4/45

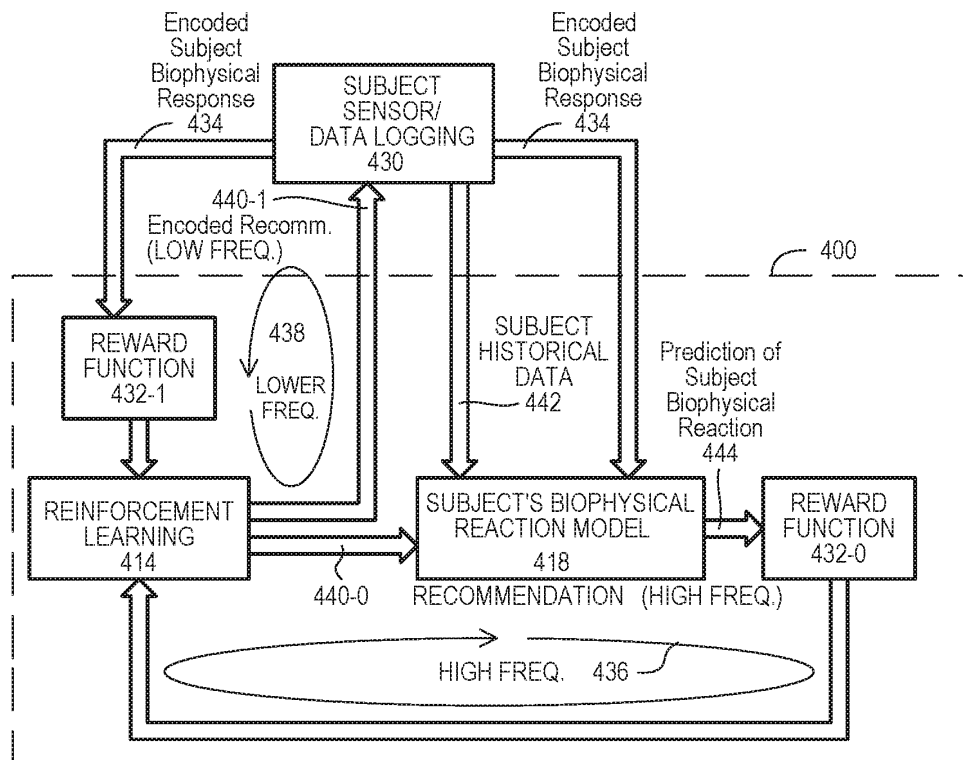


FIG. 4A

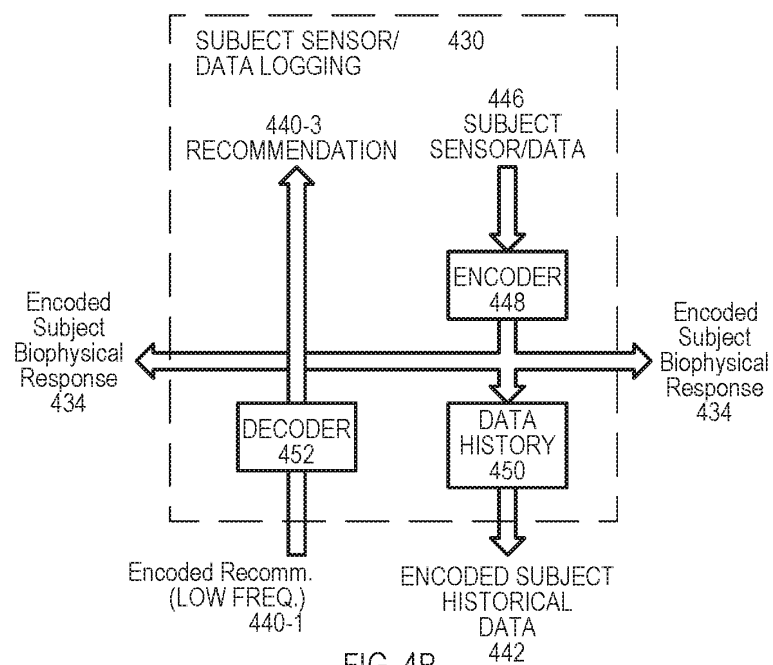


FIG. 4B

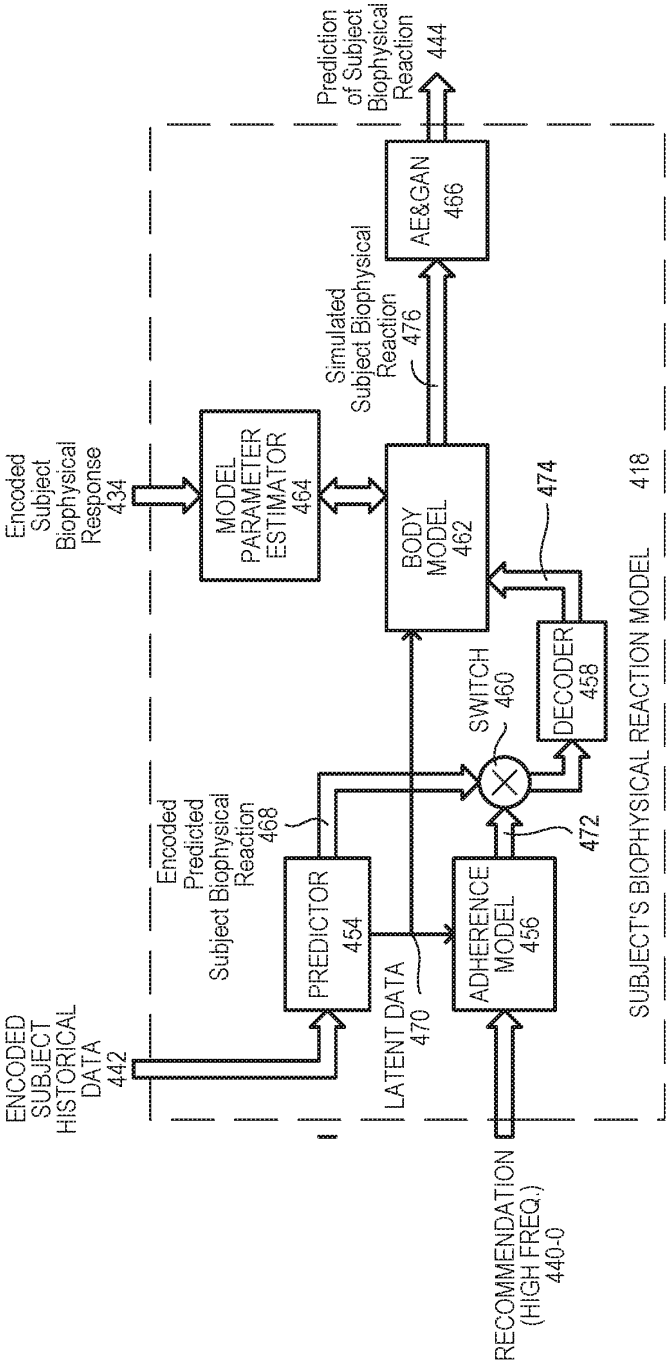


FIG. 4C

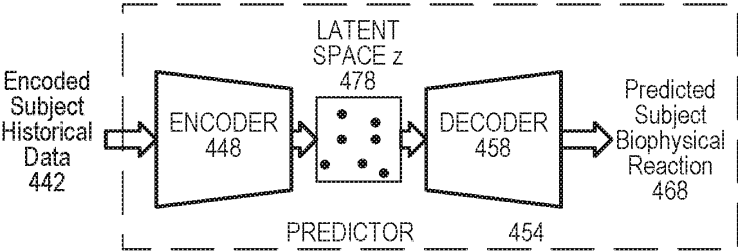


FIG. 4D

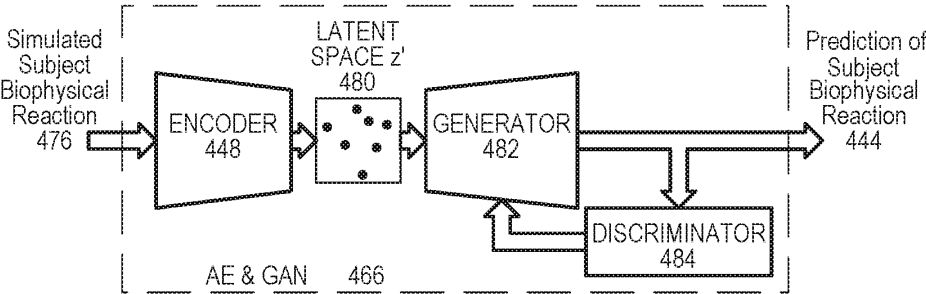


FIG. 4E

7/45

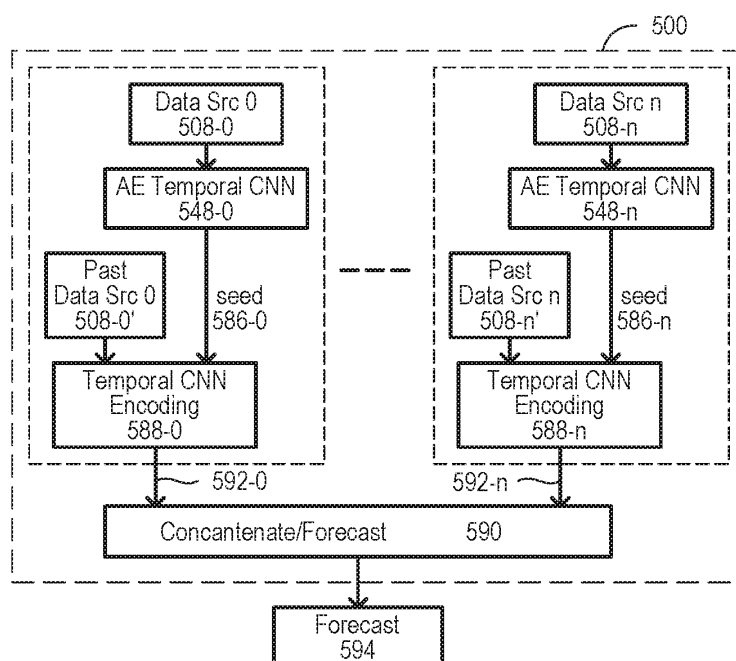


FIG. 5

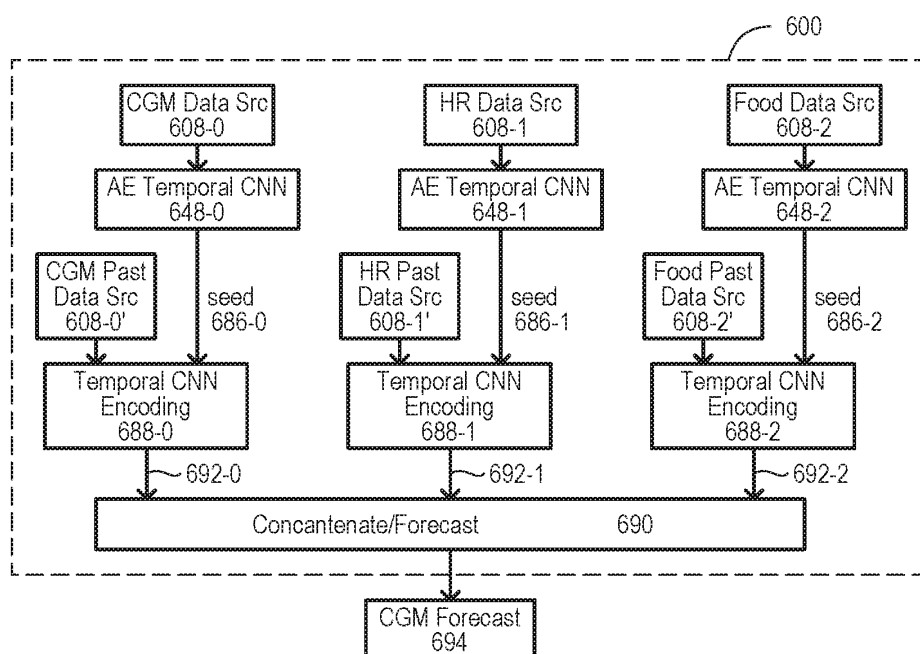


FIG. 6

8/45

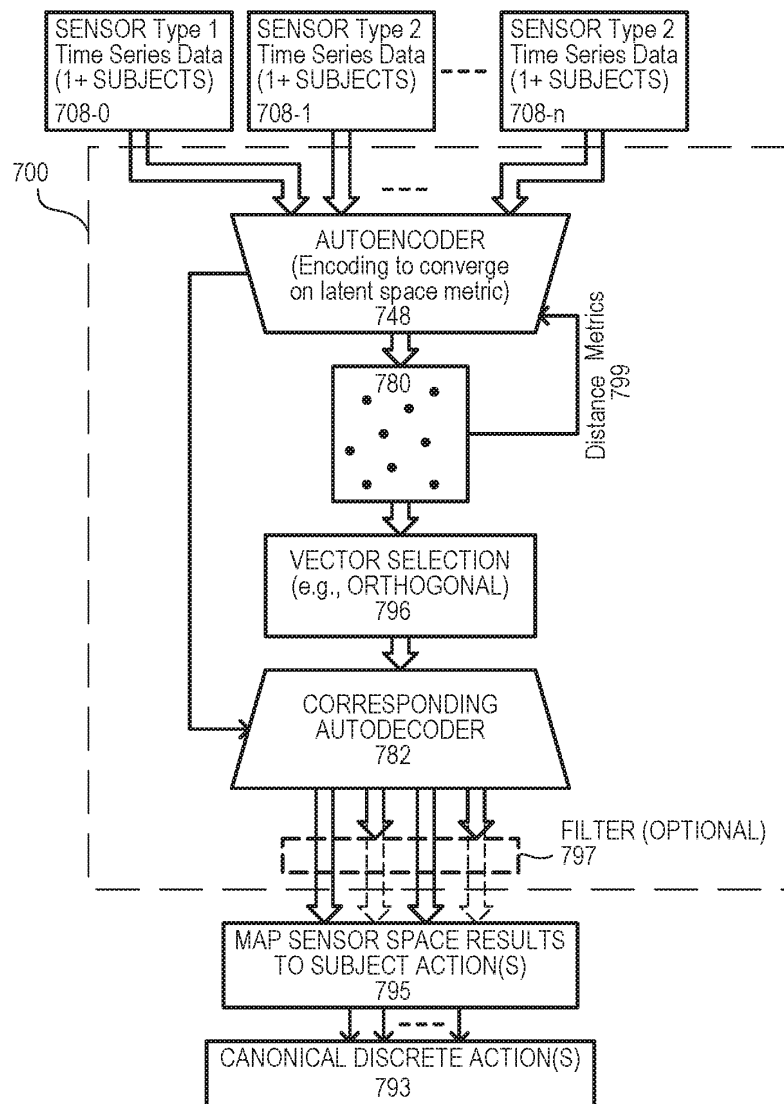


FIG. 7

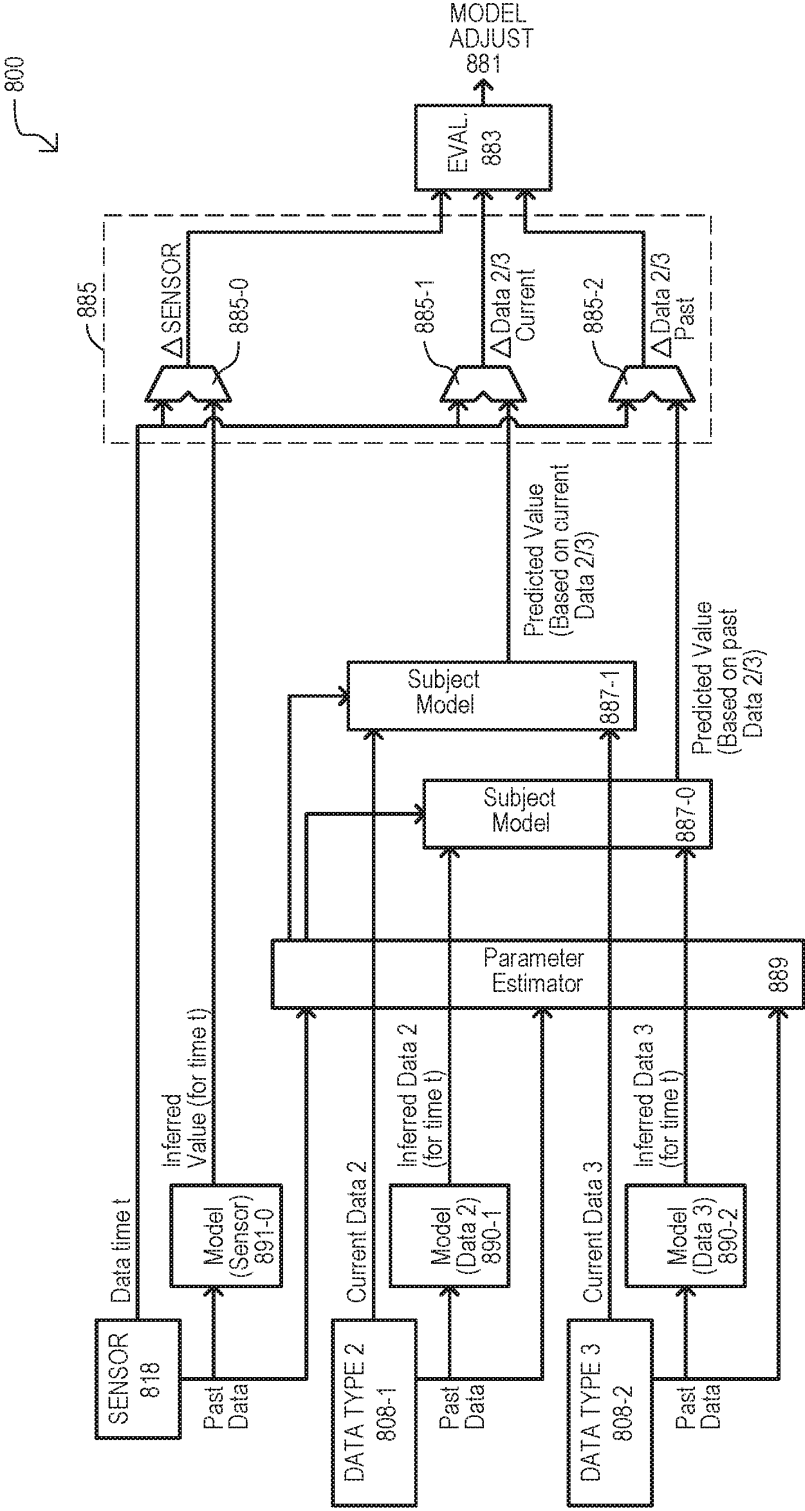


FIG. 8

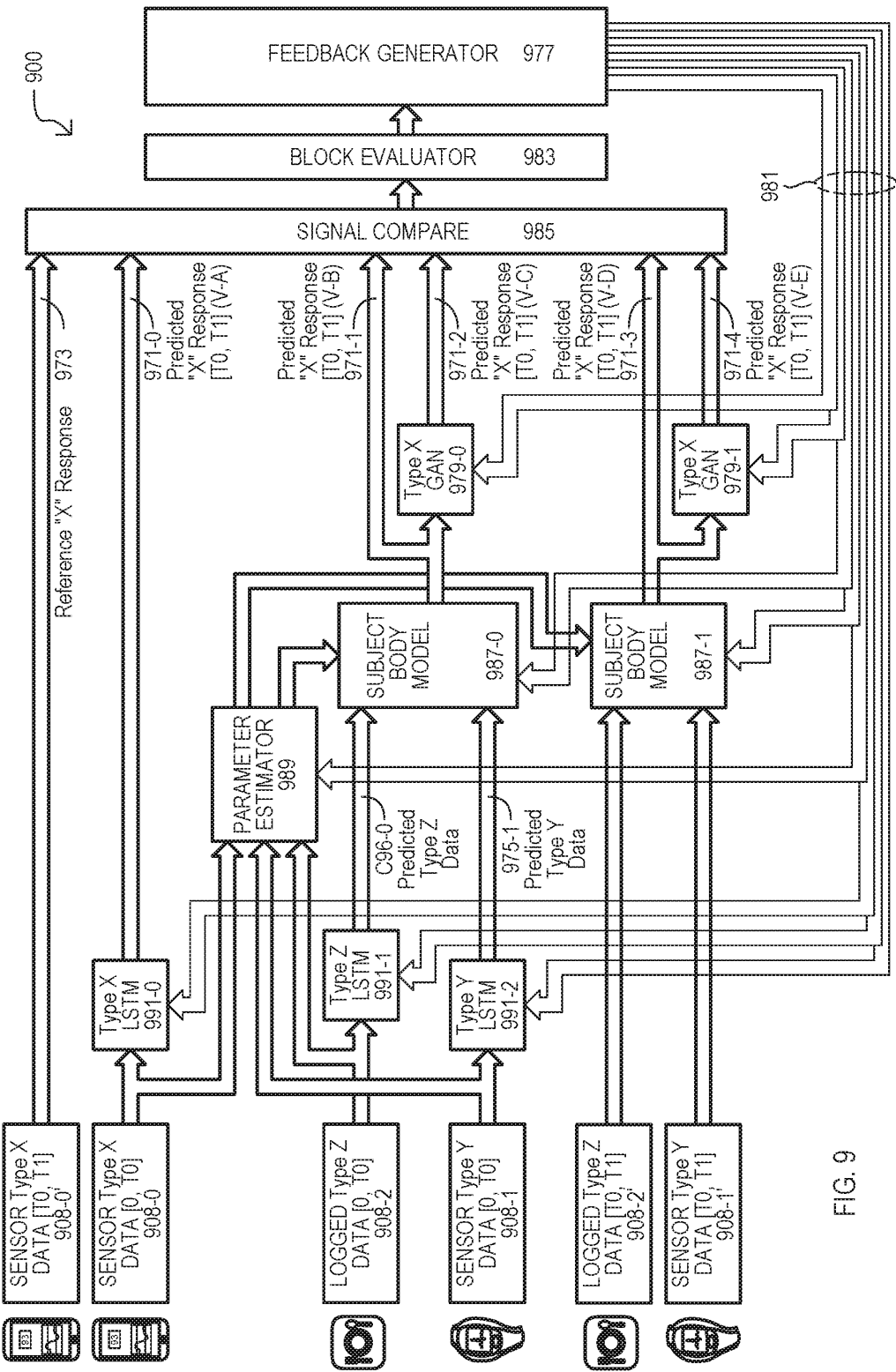
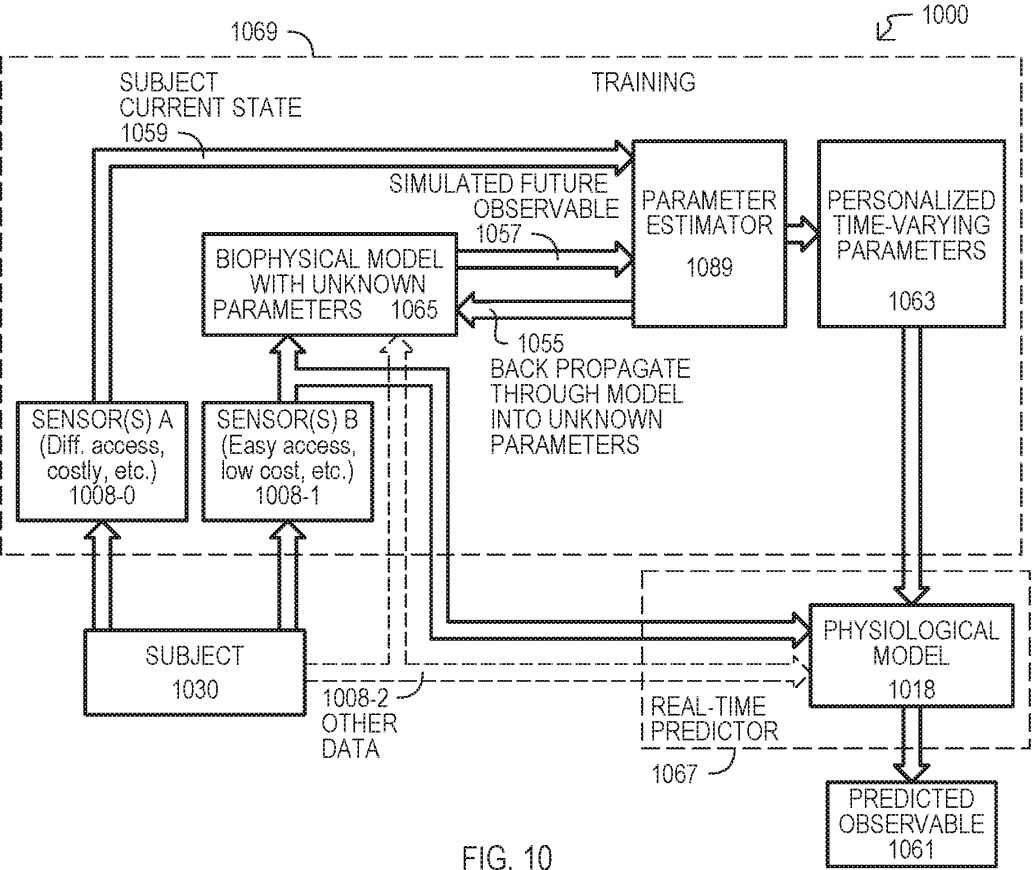


FIG. 9



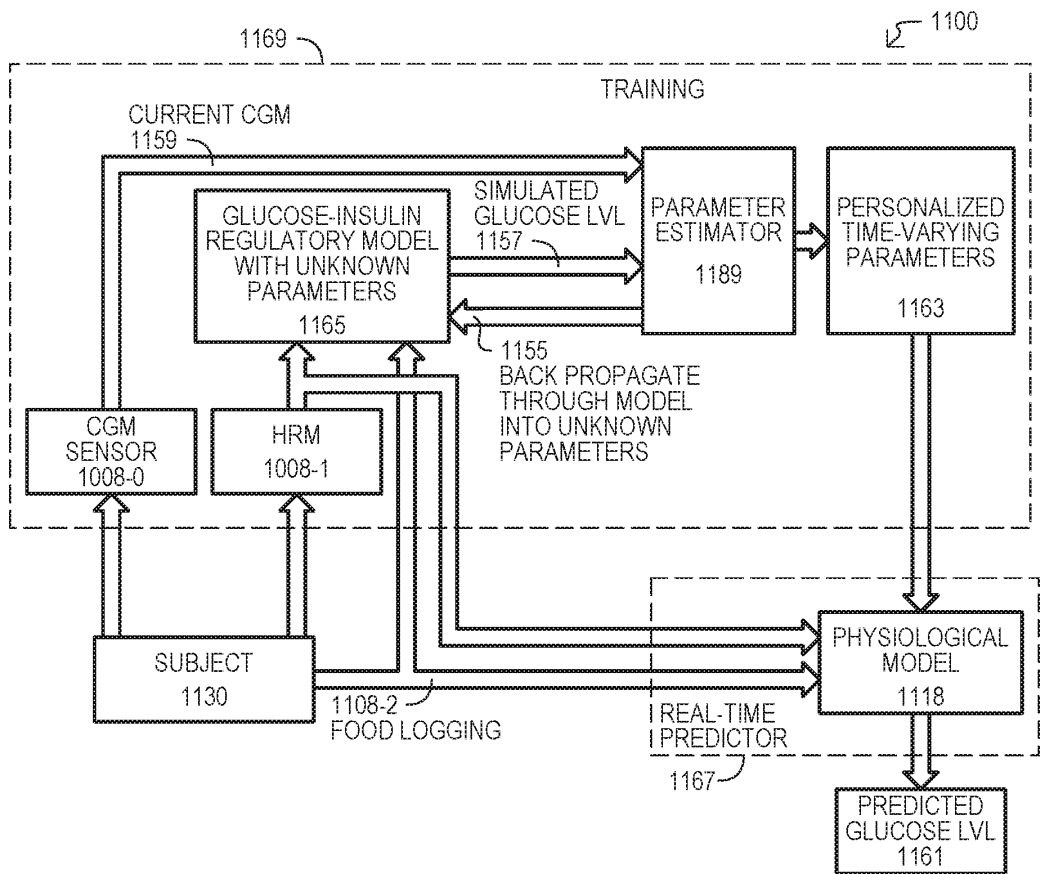


FIG. 11

13/45

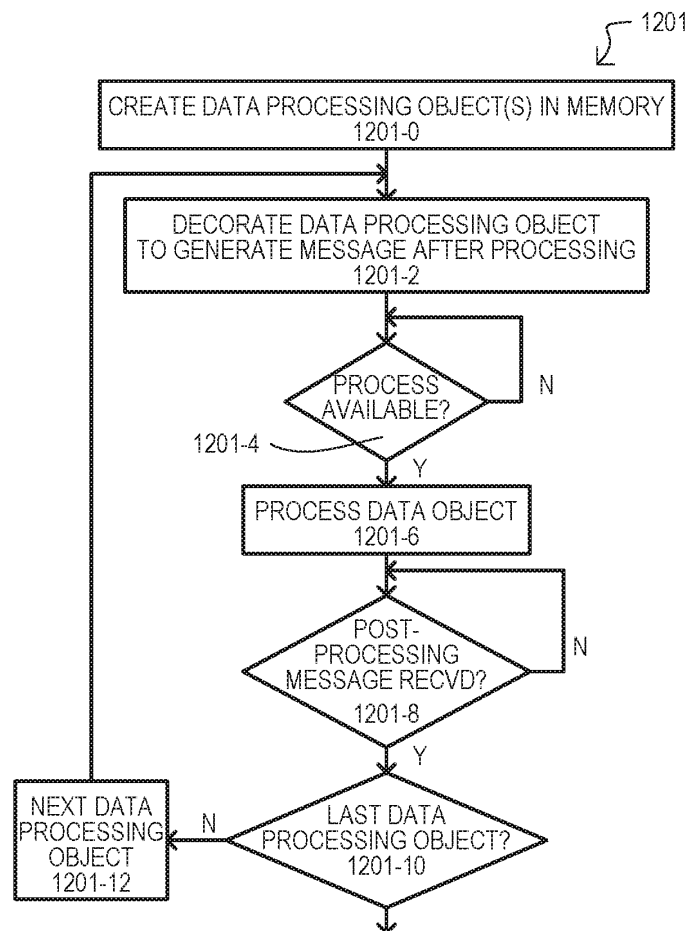


FIG. 12A

14/45

1253

1251

```

1 def run_service(self, user_id, msg)
  # identify target
  if self.target is None:
    self.producer.send(user_id, msg)

  # User based transformation target
  else:

    # Create DB structure
    in_mem_dir = self.in_memory_db_location
    random_dir = ".join(random.choices(string.ascii_uppercase + string.digits, k=1)
    name_container_db = in_mem_dir + random_dir + 'container.db'

    # Write container to in-memory database
    self.write_sqlite_blob(name_container_db=name_container_db, user_id=user_id)

    # Decorate target process to produce message output when complete
    new_target = post_process_decorator(send_func=self.producer.send, user_id=user_id,
    target_func=self.target, ingress_msg=msg,
    db_name=name_container_db)

    # Fork of separate asynchronous process
    p = Process(target=new_target, args=())
    self.containers.append(Container(process=p, container_dir=name_container_db))
    p.start()
    logging.info("Executed contain process pid: {}".format(p.pid))

```

Encapsulate Egress Message according to ML Service Protocol

```

def service_encode(query, results, db_name=""):

    :type db_name: str
    :type query: dict
    :type results: dict
    :return:

    query_json = query
    results['version'] = 1
    results['db_name'] = db_name
    query_json['results'] = results

    return query_json

```

1249-0

1249-1

```

# Decorator helper for container transformation
def post_process_decorator(send_func, user_id, target_func, ingress_msg, db_name):
    def new_target():
        send_func(user_id=user_id,
        msg_json=service_encode(query=ingress_msg,
        results=target_func(db_name, ingress_msg),
        db_name=db_name))

    return new_target

```

FIG. 12B

15/45

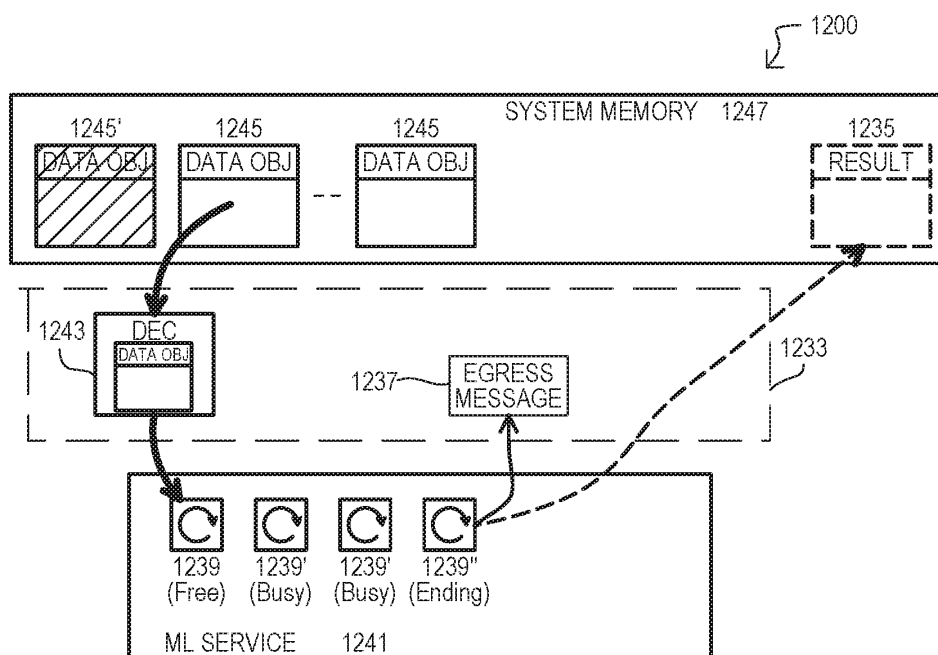


FIG. 12C

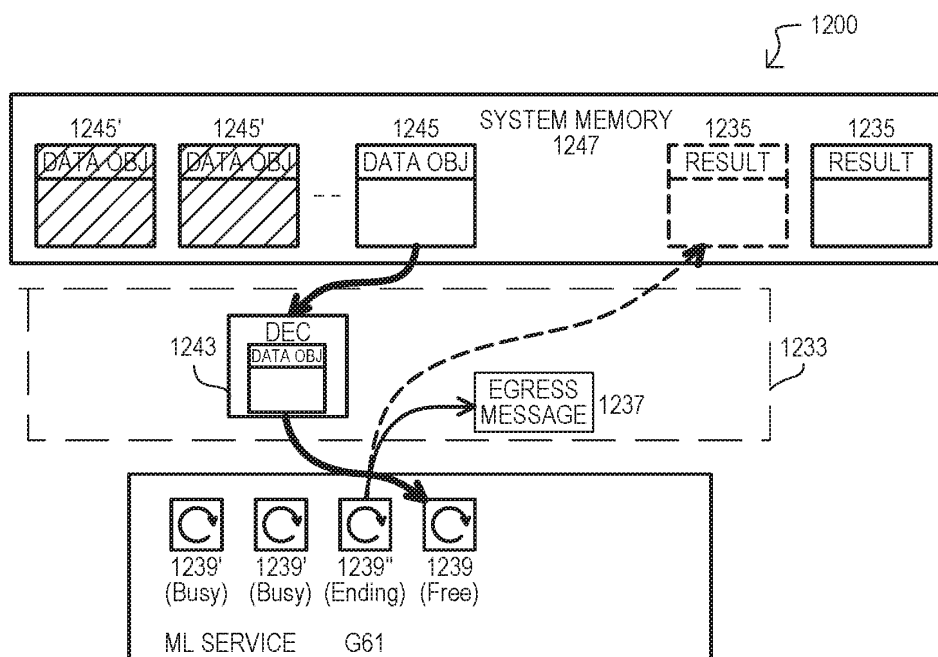


FIG. 12D

16/45

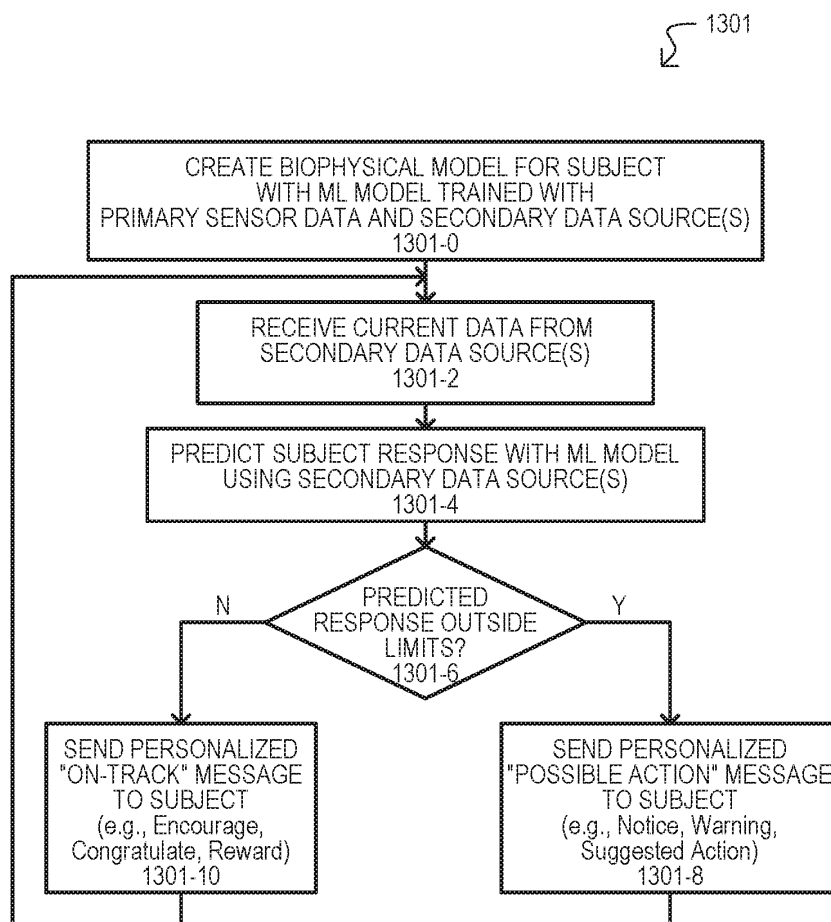


FIG. 13

17/45

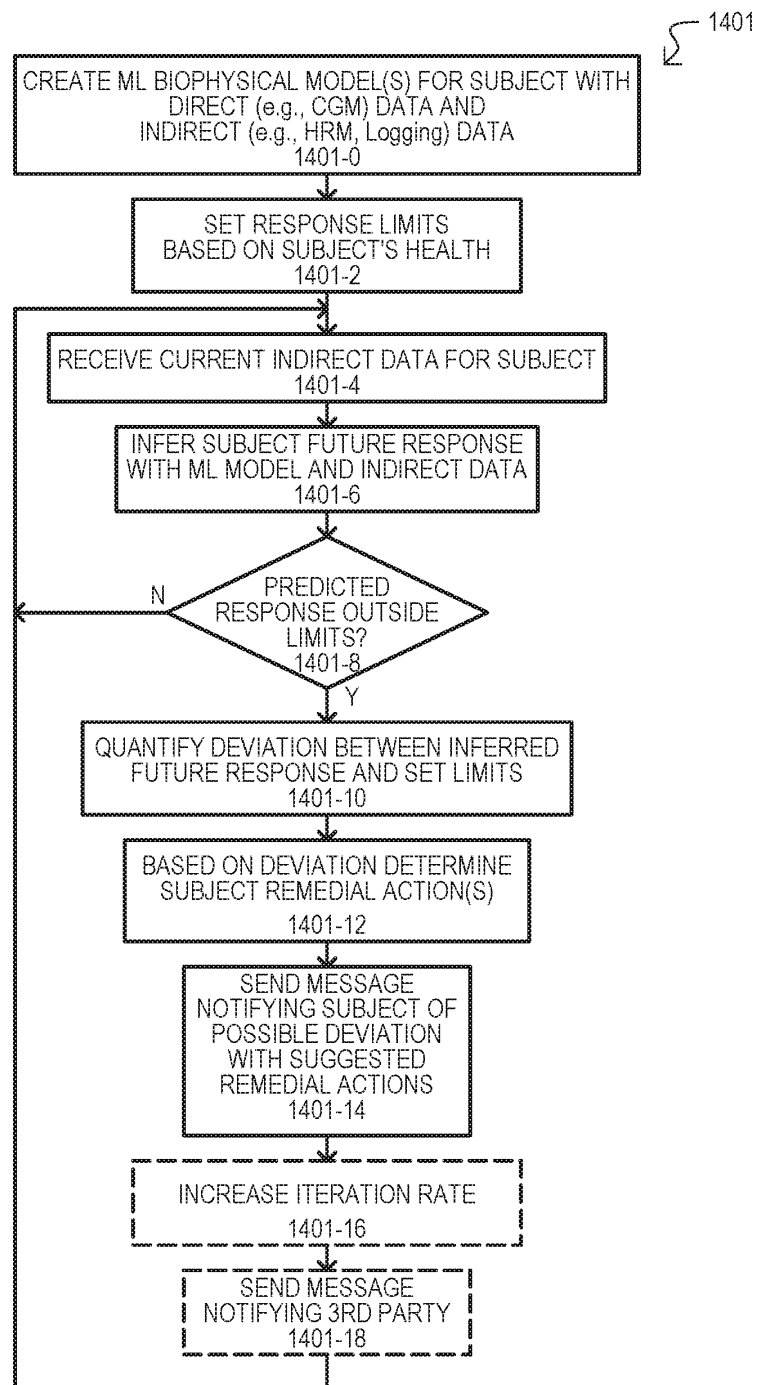


FIG. 14

18/45

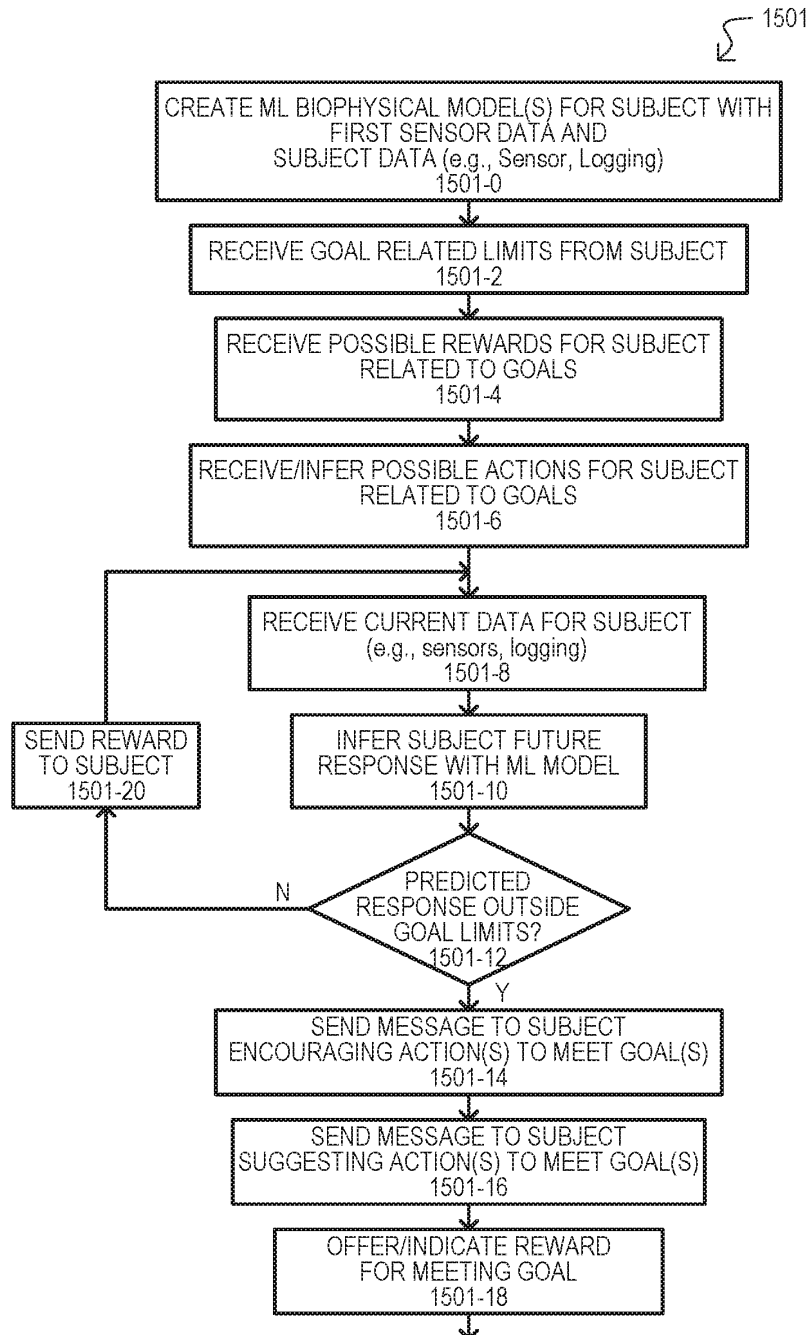


FIG. 15

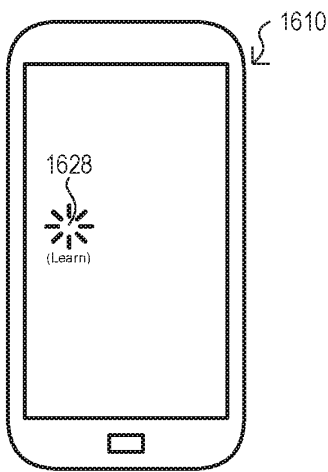


FIG. 16A

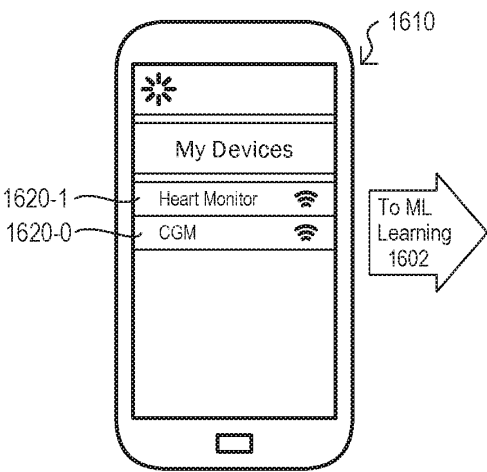


FIG. 16B

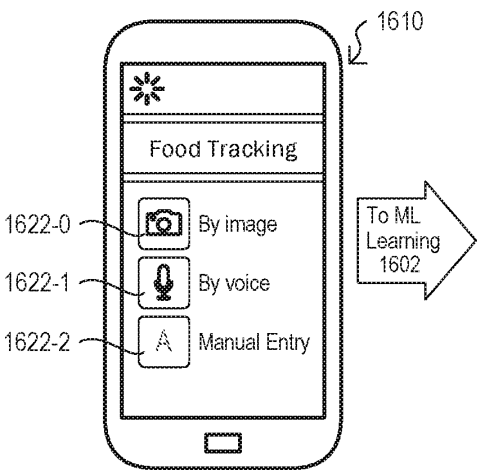


FIG. 16C

20/45

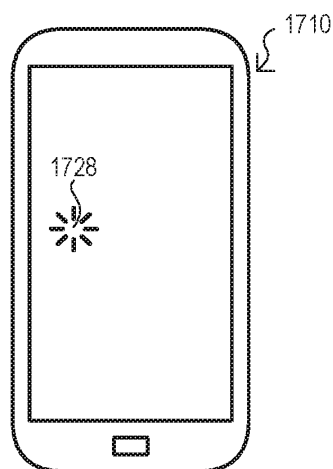


FIG. 17A

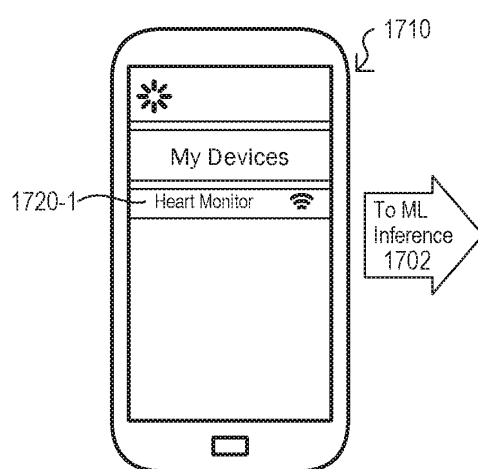


FIG. 17B

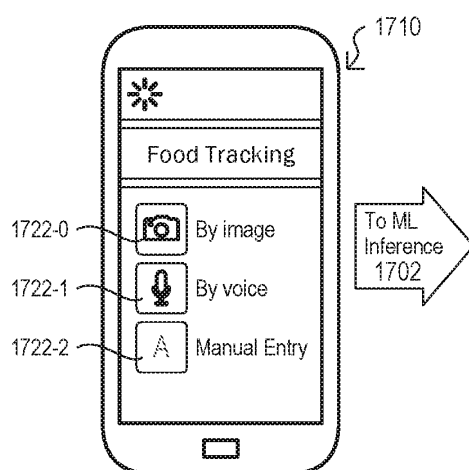


FIG. 17C

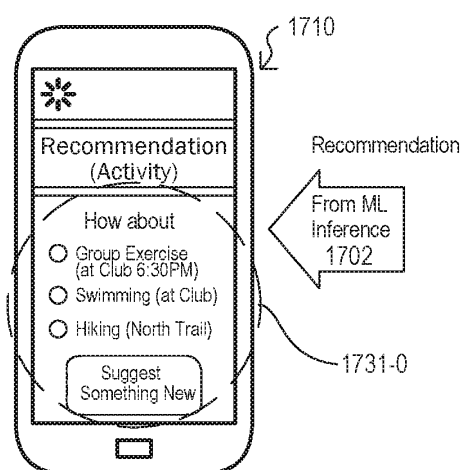


FIG. 17D

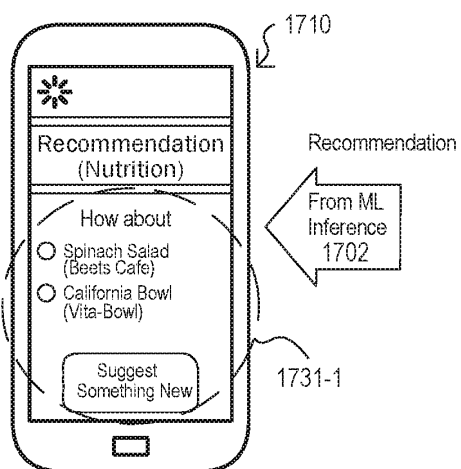


FIG. 17E

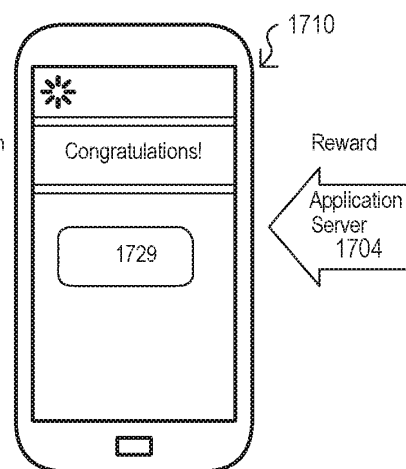


FIG. 17F

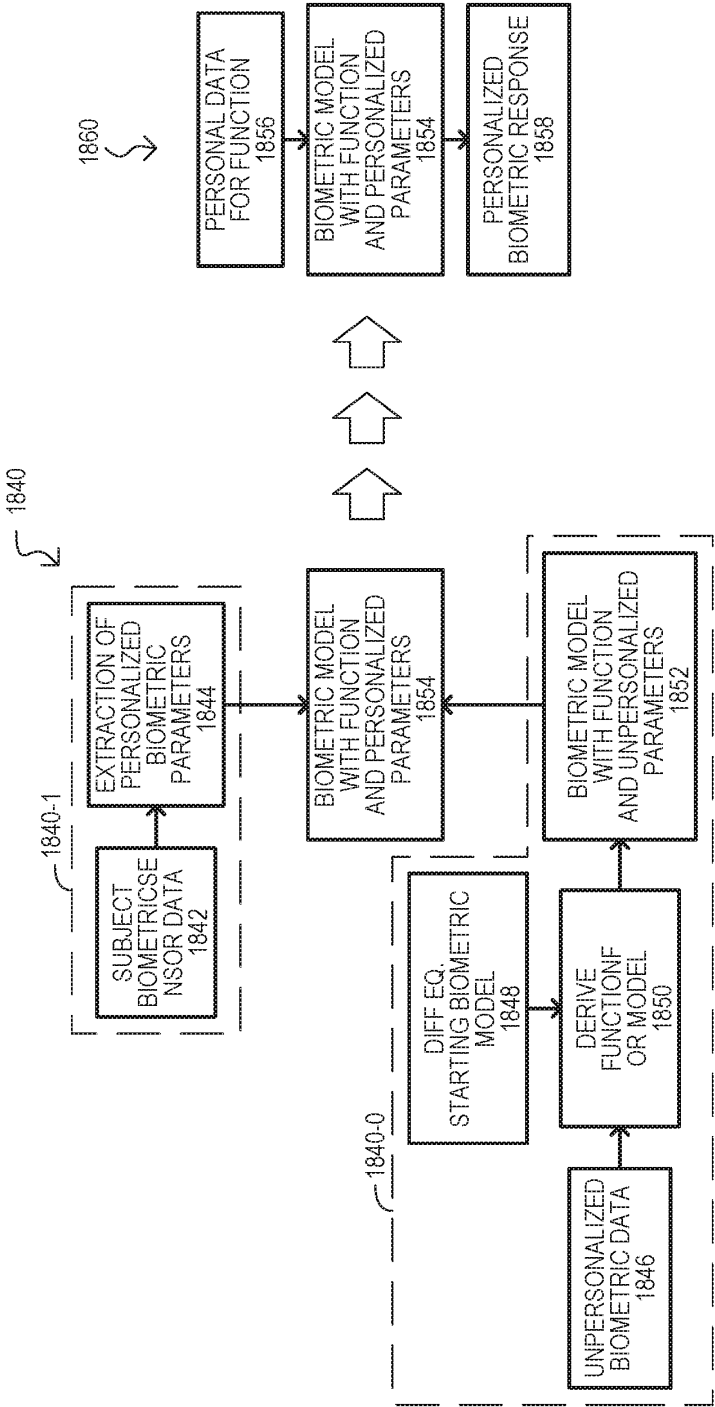


FIG. 18A

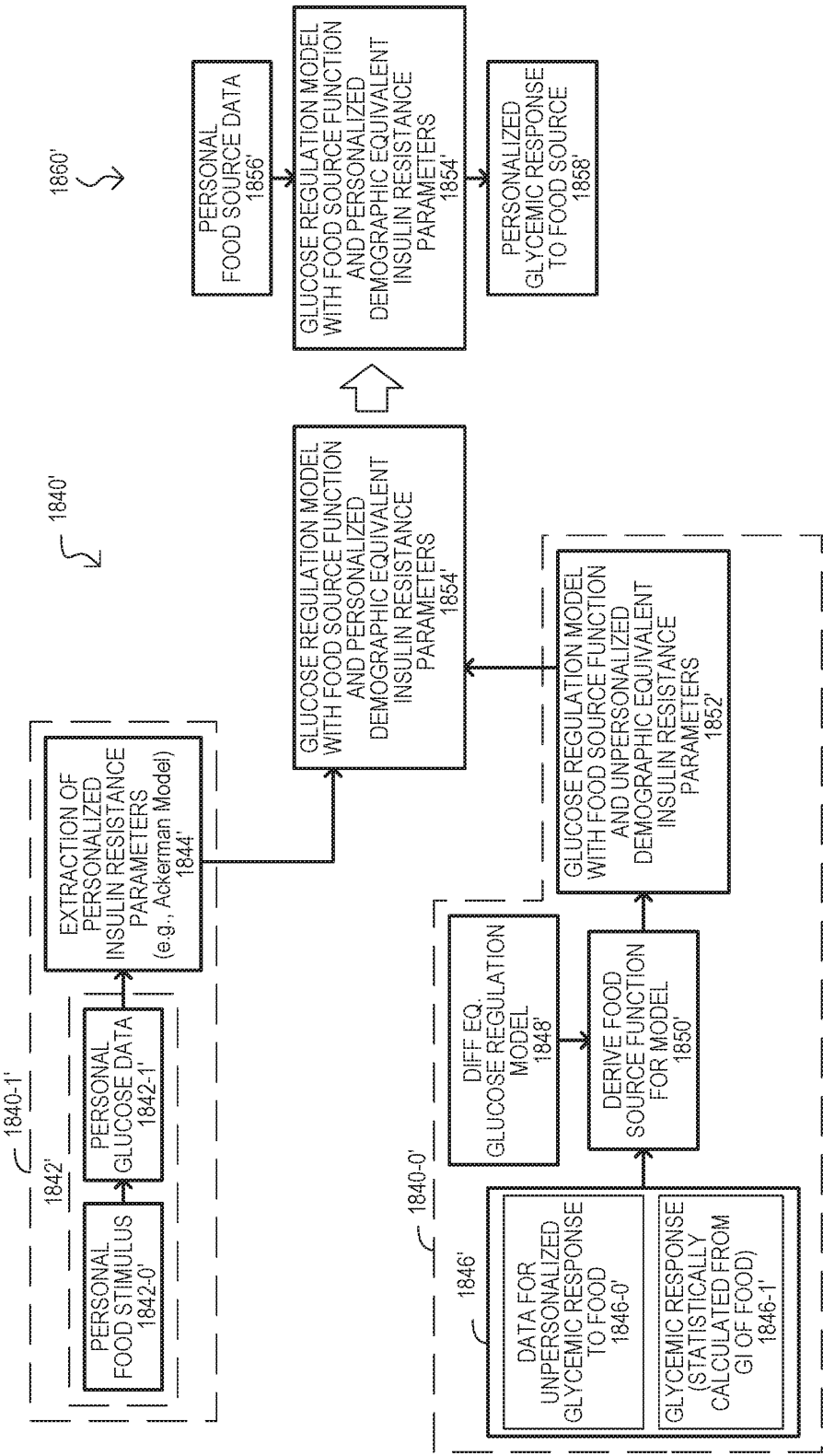


FIG. 18B

23/45

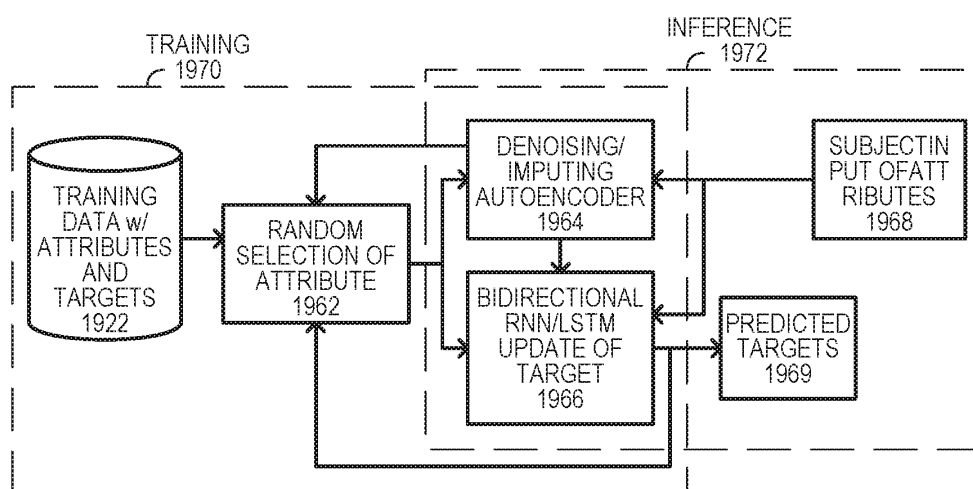


FIG. 19A

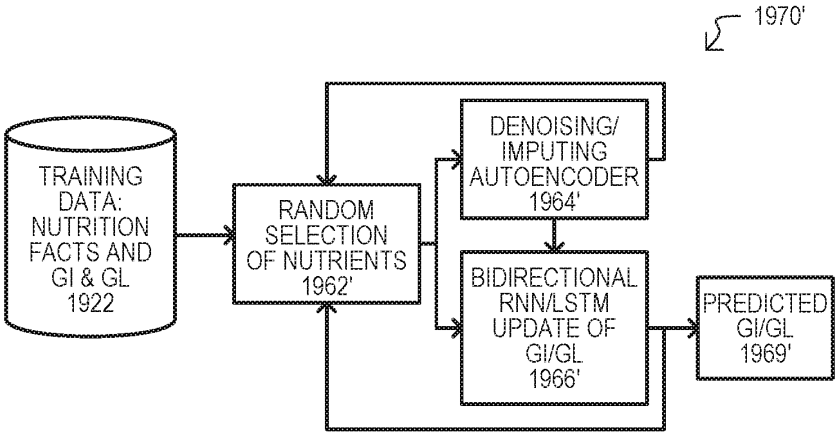


FIG. 19B

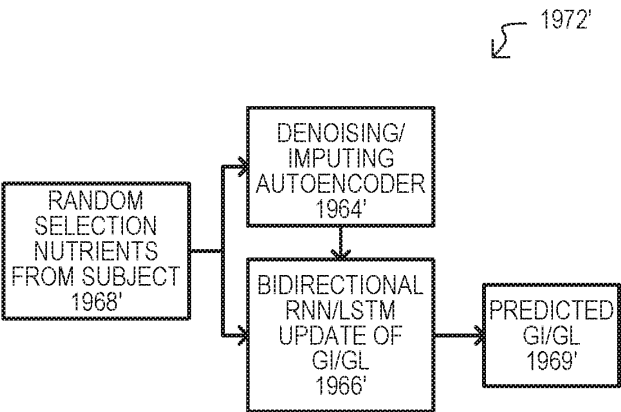


FIG. 19C

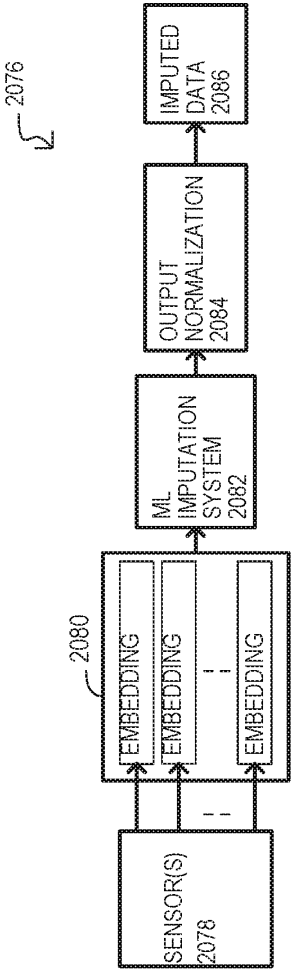


FIG. 20A

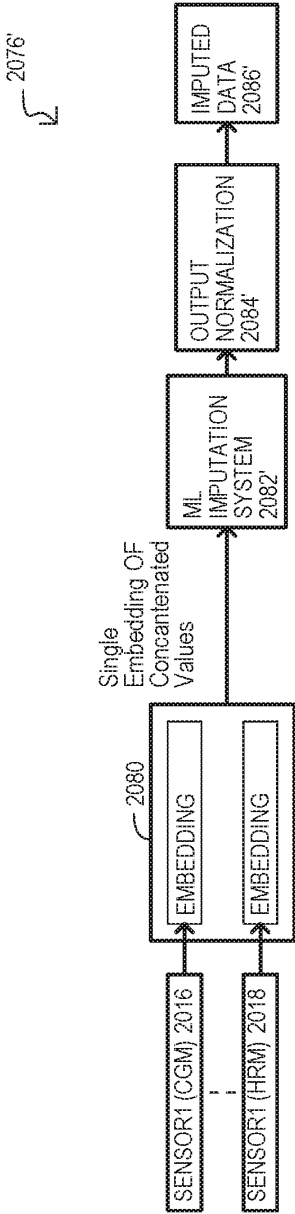


FIG. 20B

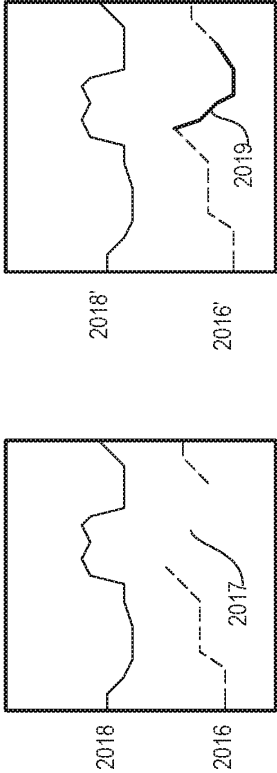


FIG. 20C

FIG. 20D

27/45

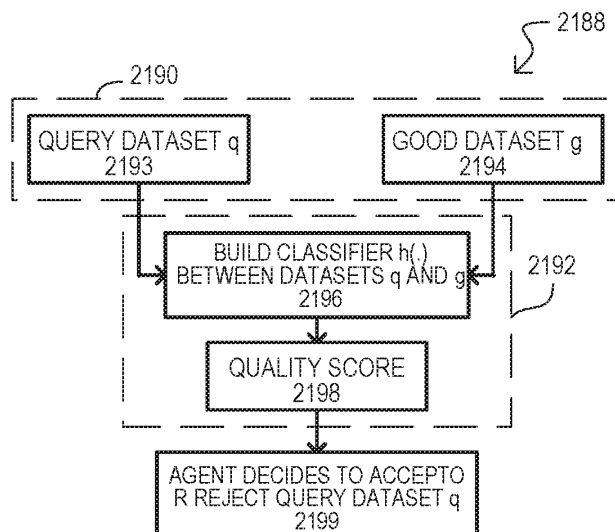


FIG. 21A

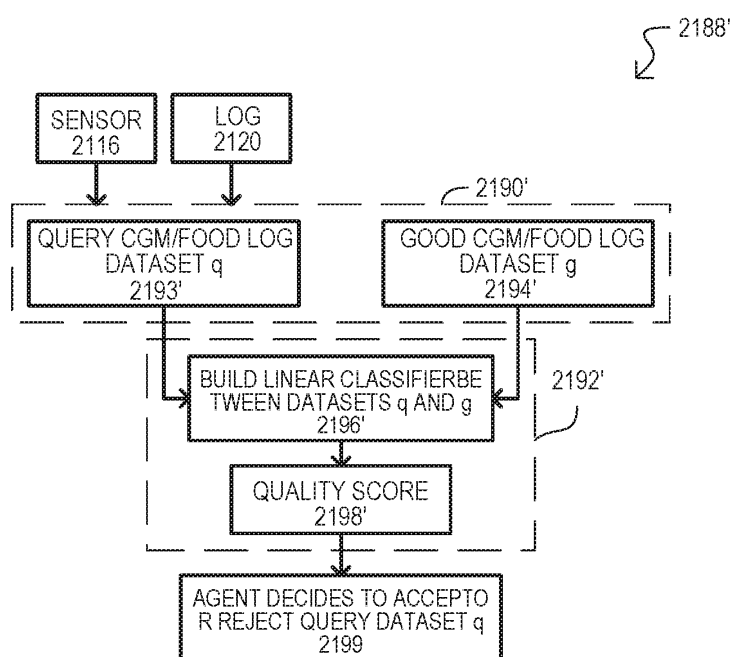


FIG. 21B

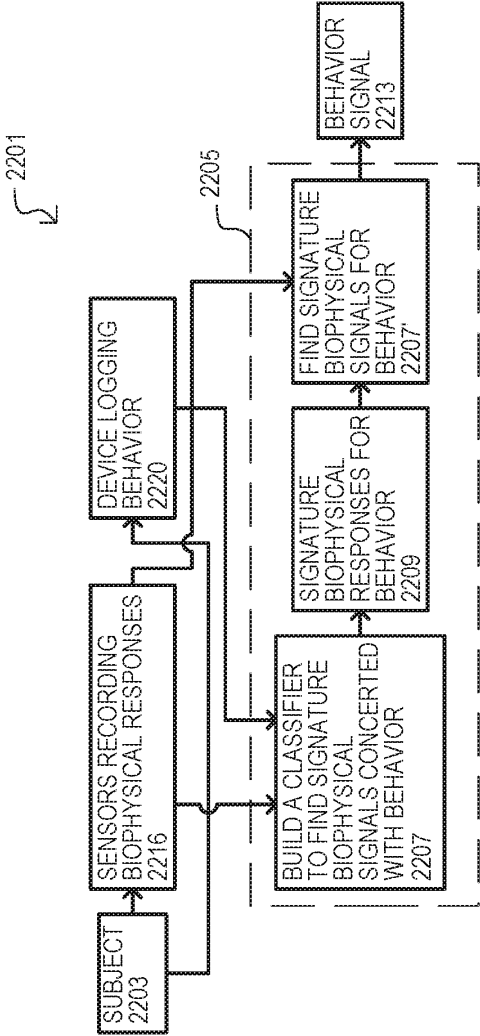


FIG. 22A

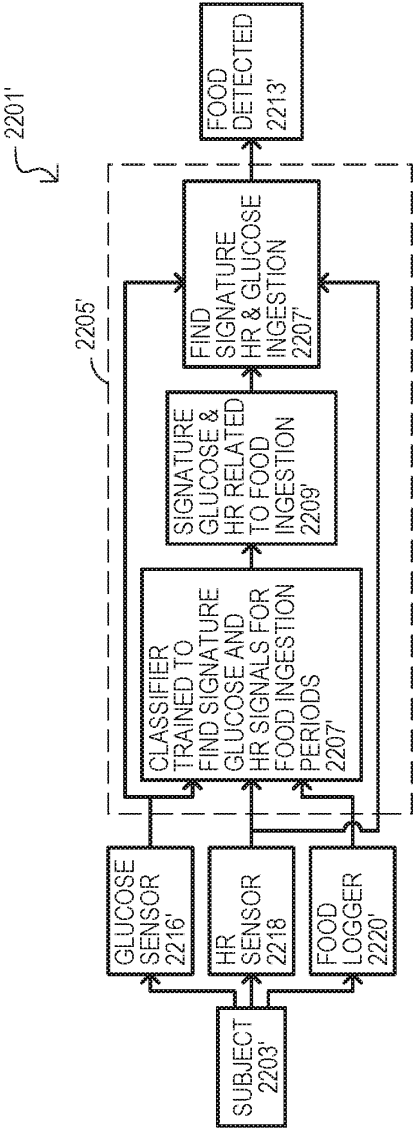


FIG. 22B

30/45

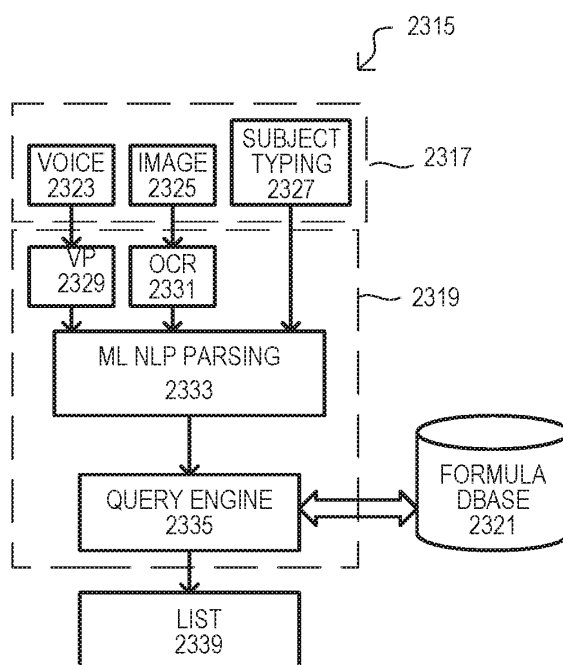


FIG. 23A

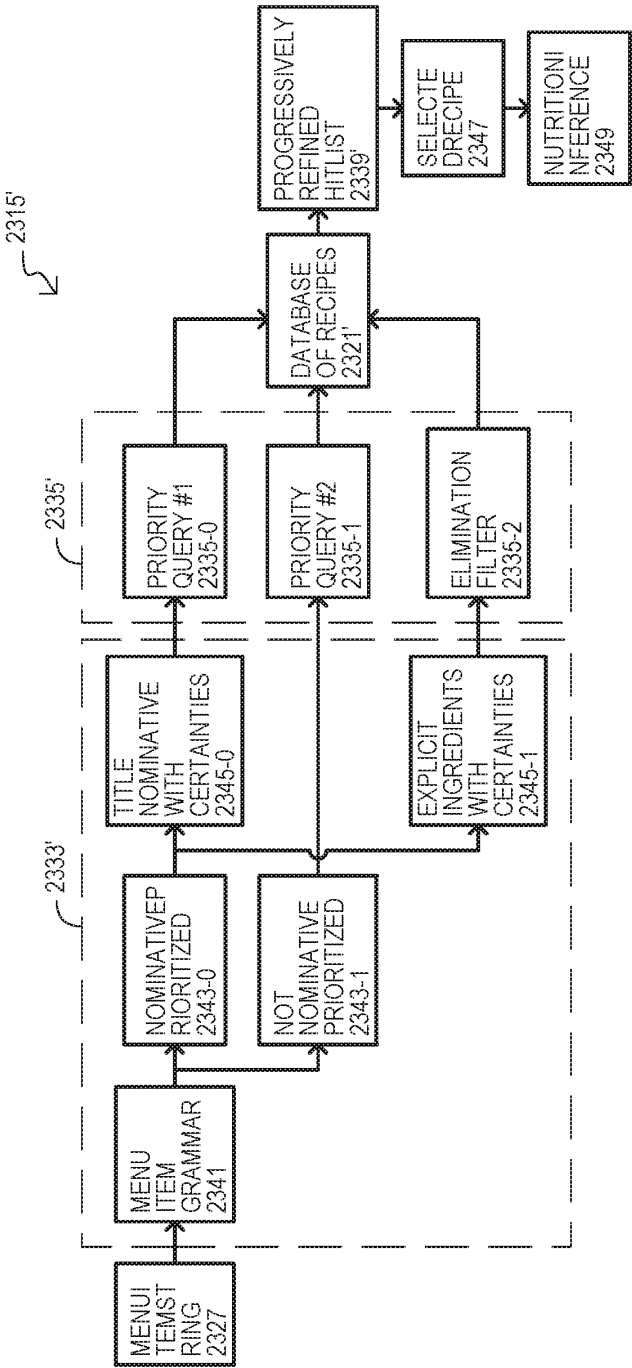
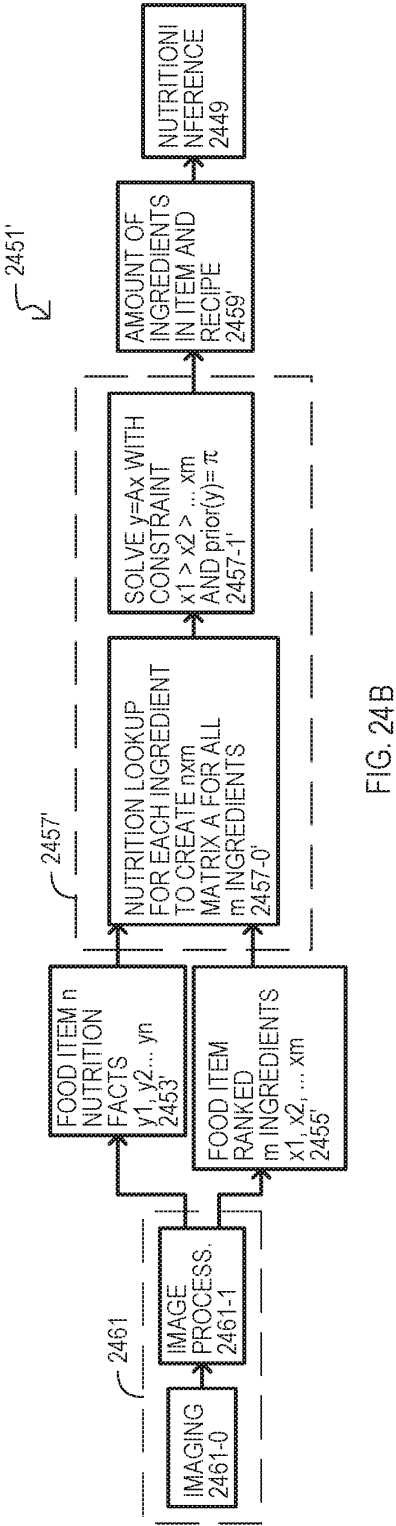
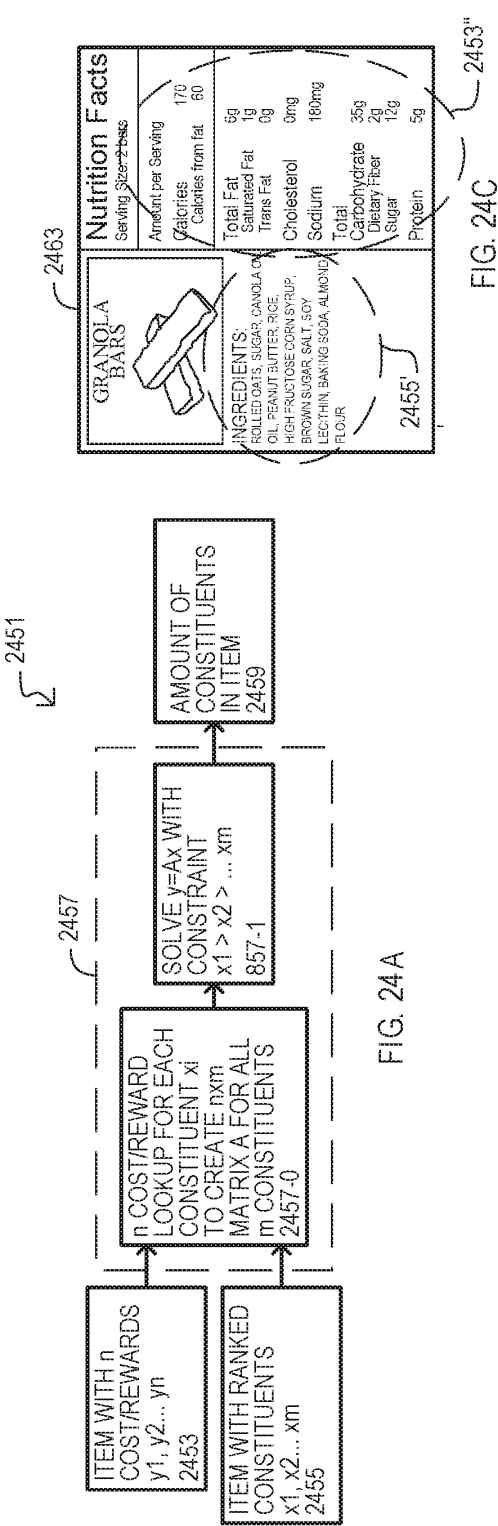


FIG. 23B



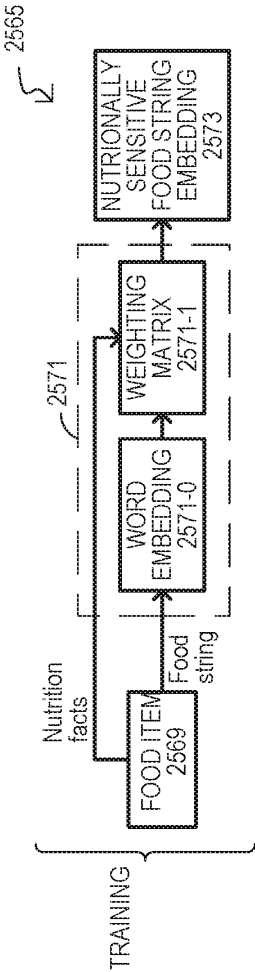


FIG. 25A

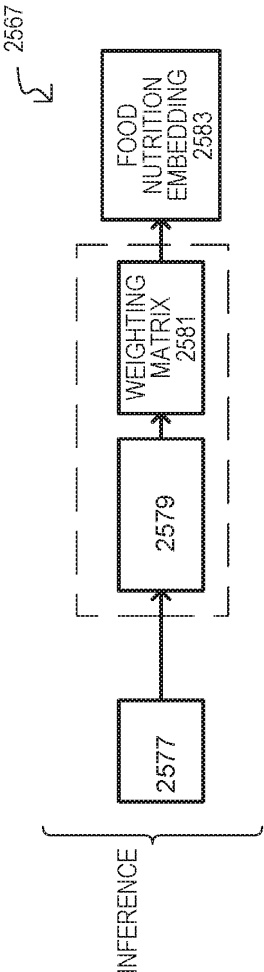


FIG. 25B

34/45

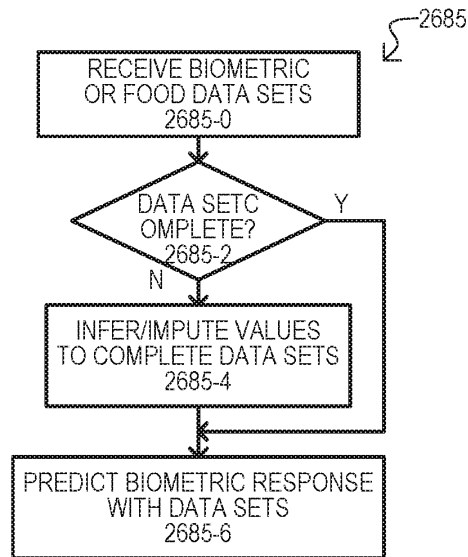


FIG. 26

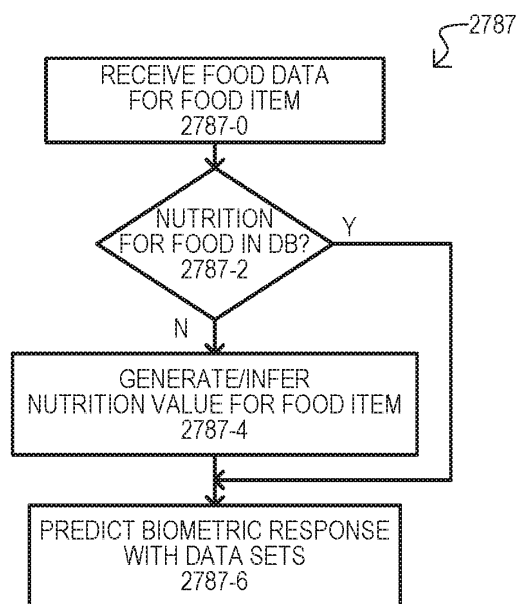


FIG. 27

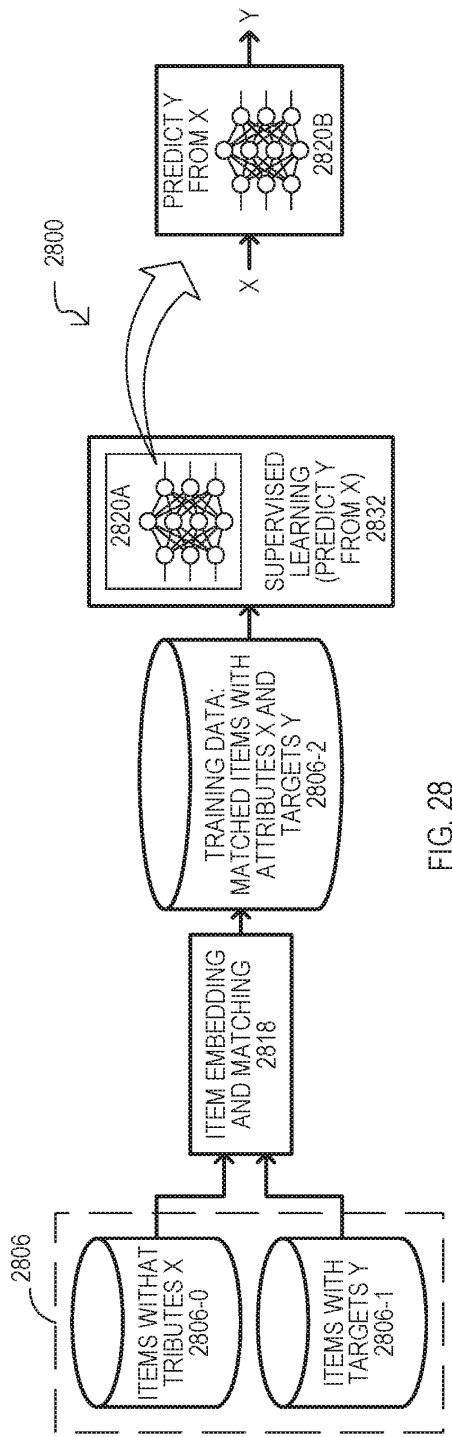


FIG. 28

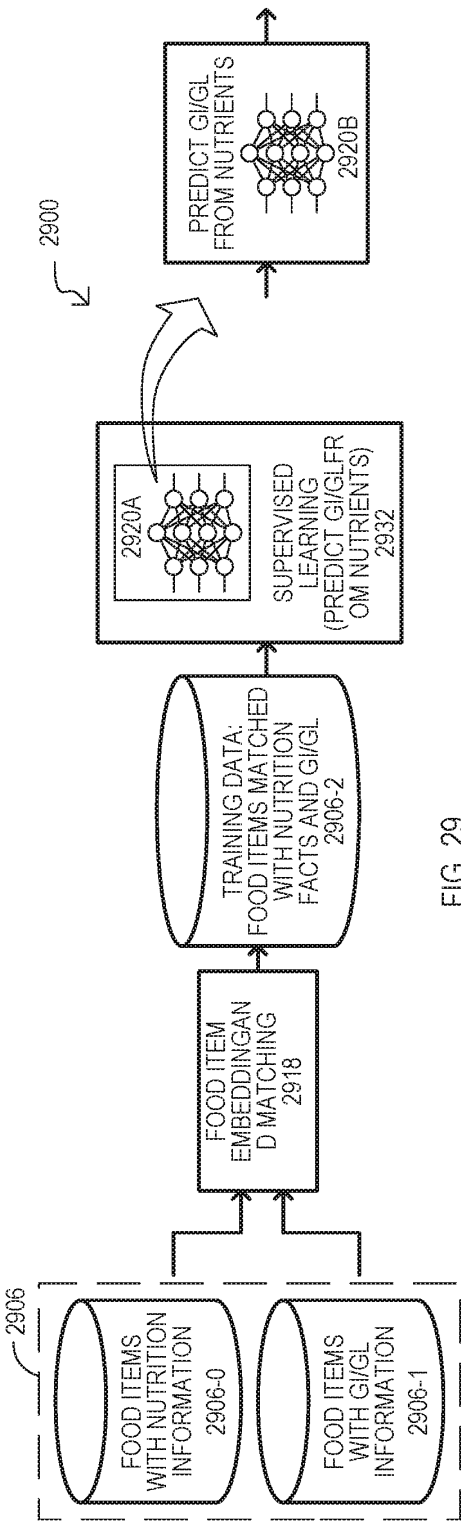


FIG. 29

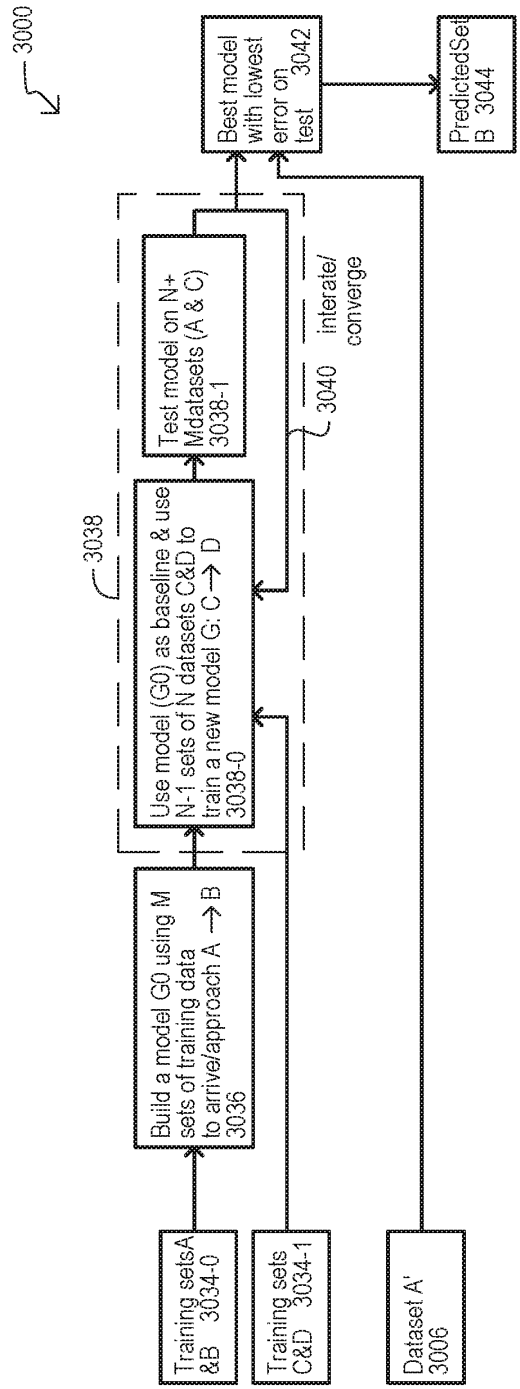


FIG. 30

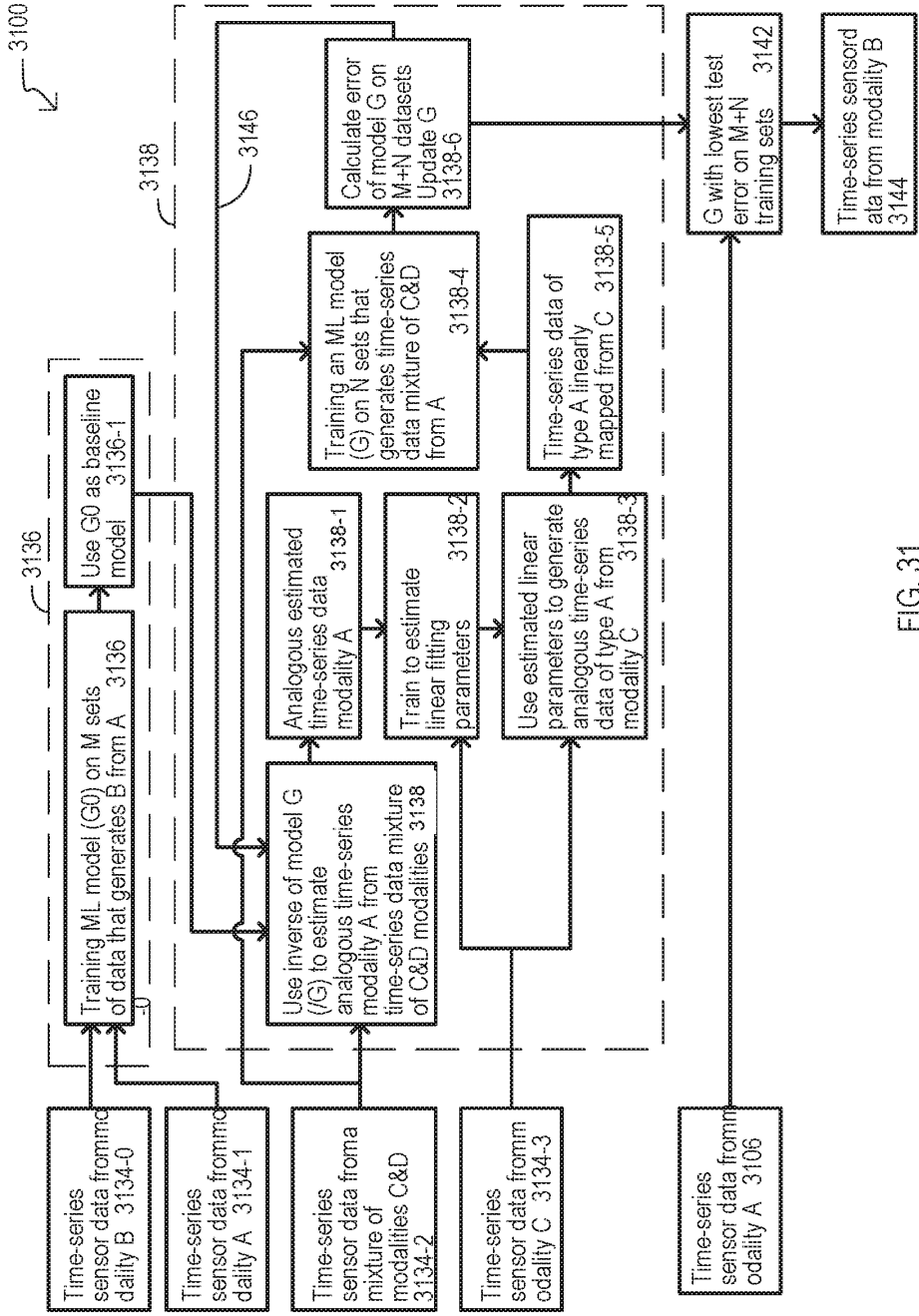


FIG. 31

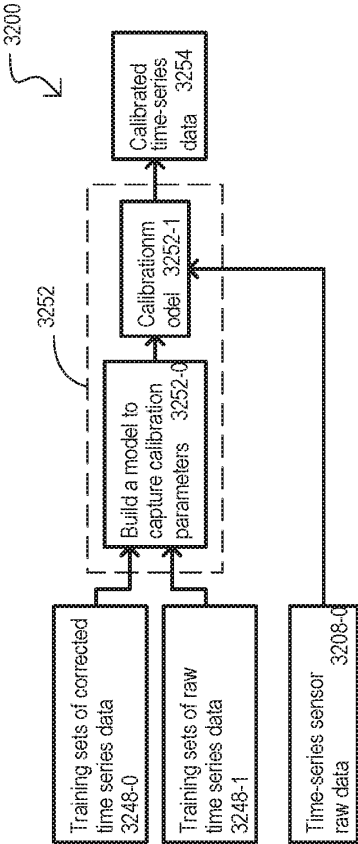
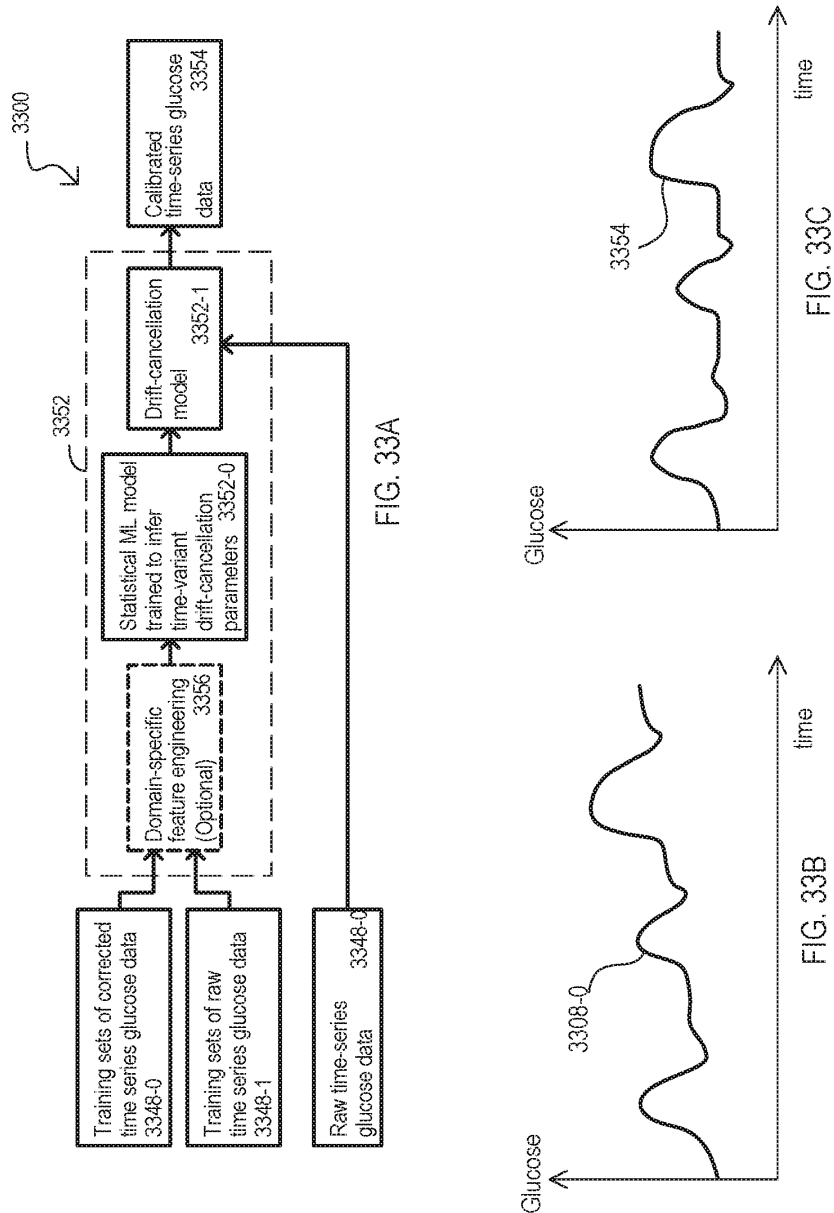


FIG. 32



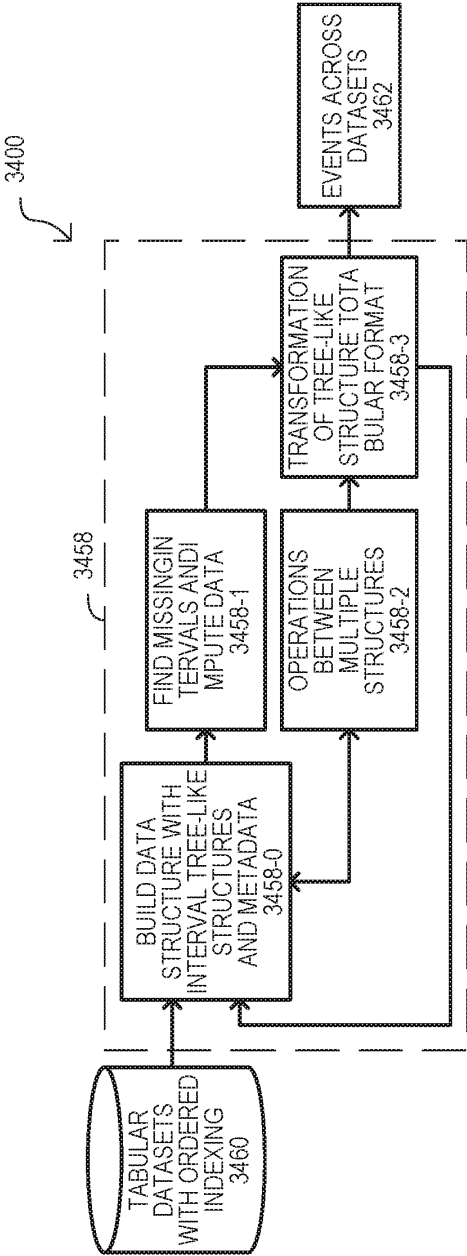


FIG. 34

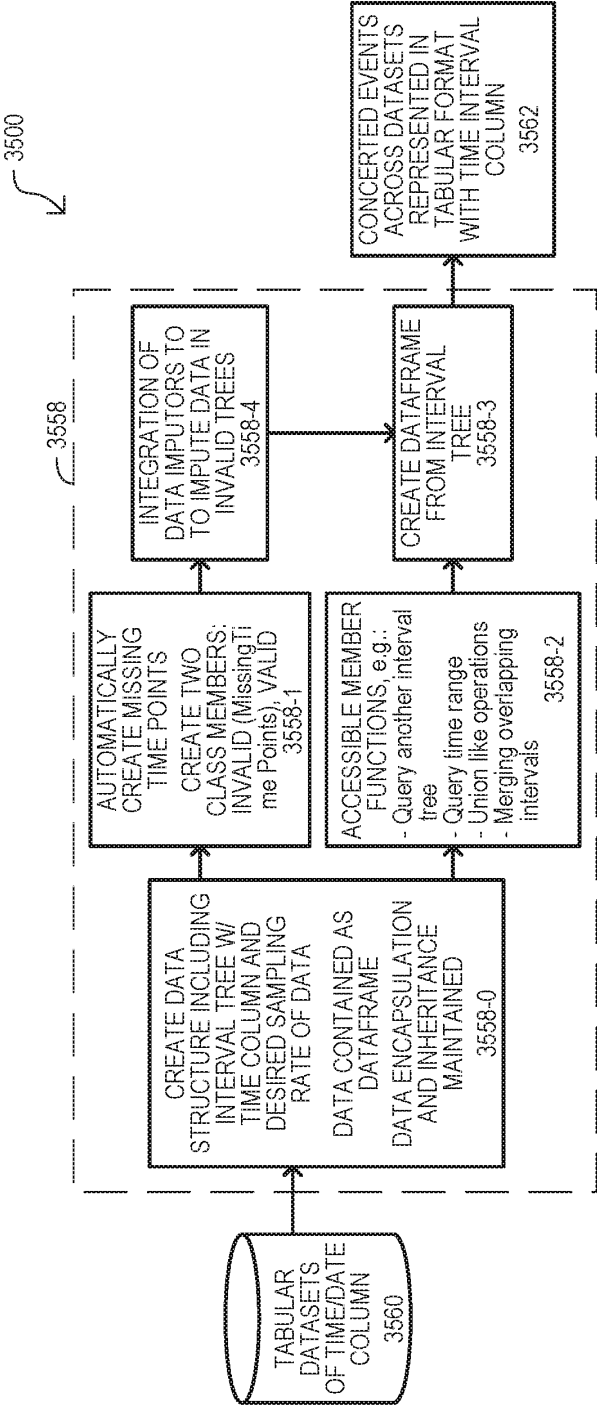


FIG. 35

42/45

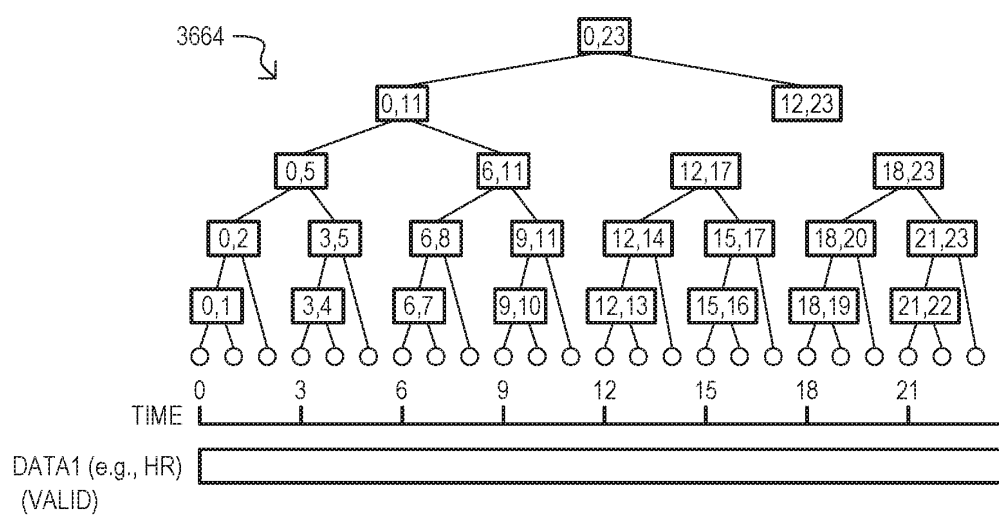


FIG. 36A

43/45

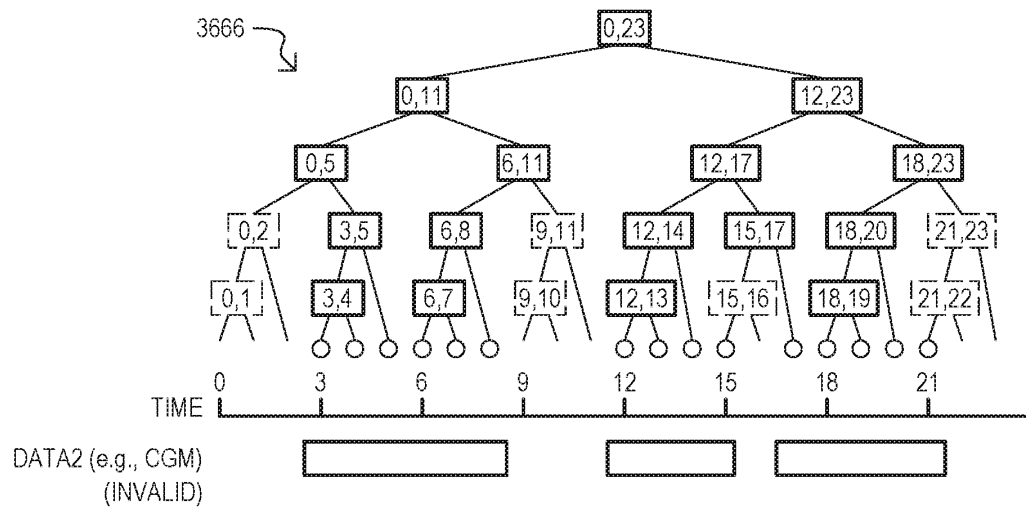


FIG. 36B

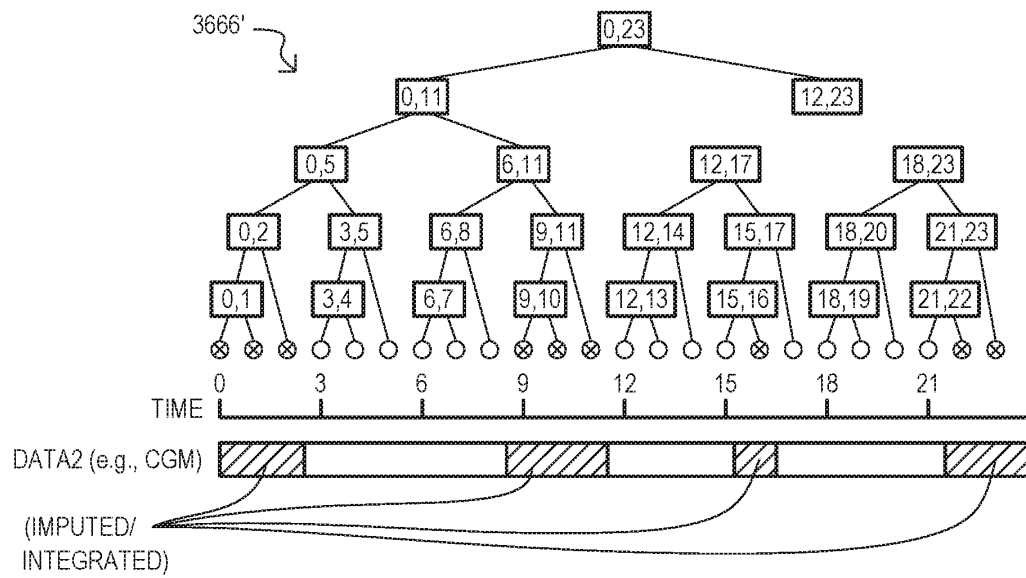


FIG. 36C

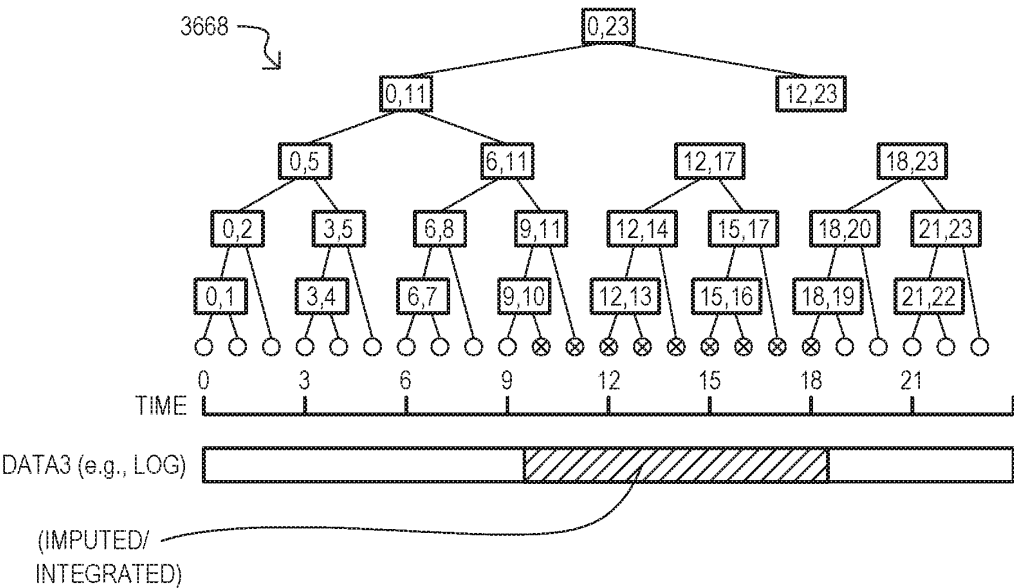


FIG. 36D

QUERY(time=1-4, DATA1, DATA2, DATA3)

TIME	DATA1	DATA2	DATA3
12	72	65	Turkey Sandwich, Iced Tea
13	76	82	Nothing
14	74	70	Banana
15	72	81	Nothing

FIG. 36E

45/45

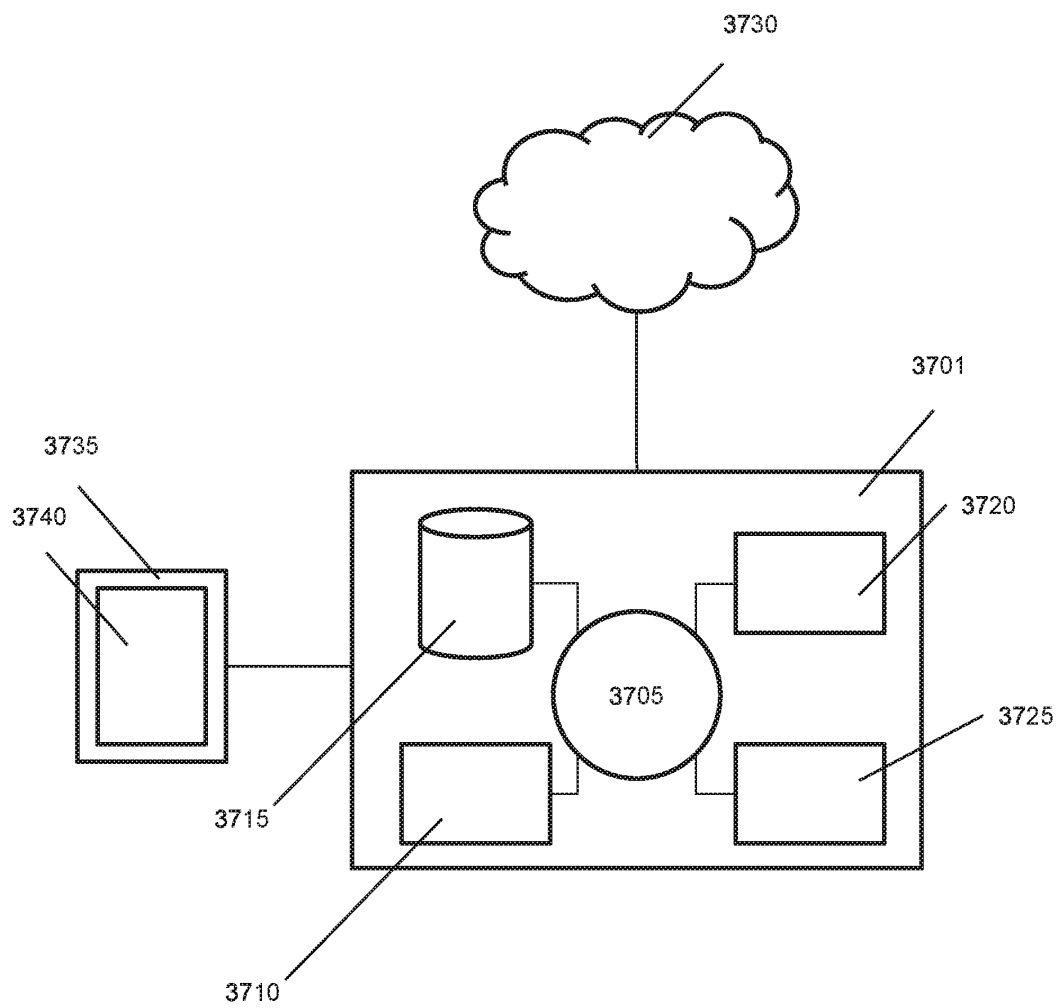


FIG. 37