```
      model = keras.models.load_model(load_model)
      history = None
      time_train = 0

    # model test
    if subjs_test:
      x_data_test, y_data_test, _ = partition_data(subjs_test[perm_index] if type(subjs_test[0]) == tuple else subjs_test)
      if pca:
        x_data_test, _ = reduce_matrix(x_data_test, Vpca)
        y_data_test = adjust_size(x_data_test, y_data_test)
      print(f'\n### testing with {subjs_test[perm_index] if type(subjs_test[0]) == tuple else subjs_test}, {len(x_data_test)} inputs')
      start_time = time.monotonic()
      eval_metrics = model.evaluate(x_data_test, y_data_test)
      time_test = time.monotonic() - start_time
    else:
      eval_metrics = [0., 0.]
    if write_report:
      write_values()
    if save_model:
      # save in both directory and h5 formats (we had problems with both of them sometimes)
      model_file_name = f'{working_dir}/{file_id}model_w{window:04d}_o{overlap:03d}_d{decimation:03d}_e{epochs}_t{round(eval_metrics[1] *
      #model.save(model_file_name)
      model.save(model_file_name + '.h5')
  if write_report:
    out_f.close()

  return model, x_data_test, y_data_test, eval_metrics[1]  # can be needed by other blocks
```

With RPCA

```
"""---
**Start of actual program blocks:**
"""

# create dataset, create model, train and test
# if __name__ == '__main__':
dense1 = 0
lstm1 = 8
lstm2 = 8
lstm3 = 0

window = 256
overlap = window // 2
oversample = True
decimation = 0
pca = True  # compute PCA on full data matrix
rpca = True  # compute RPCA before PCA
spikes = 1/500
rpca_mu = 0.1

subjs_train_perm = ( (tuple(i for i in range(2, 10)) + tuple(i for i in range(20, 32)), ()), )
# subjs_train_perm = ( (tuple(i for i in range(2, 15)) + tuple(i for i in range(18, 35)), ()), )
subjs_test = (0, 1, 15, 16, 17)  # 2 for N, 3 for AD
epochs = 10

  # if decimation:
  #   window //= decimation
  #   overlap //= decimation

x_data, y_data, subj_inputs = create_dataset(window, overlap, decimation)
```

```
(('01', 'N'), ('02', 'N'), ('03', 'N'), ('04', 'N'), ('05', 'N'), ('06', 'N'), ('07', 'N'), ('08', 'N'), ('09', 'N'), ('10', 'N'),

### creating dataset

tot samples: 5954304
x_data: (46483, 256, 16)
y_data: (46483, 1)
windows per subject: [868, 759, 739, 1560, 1139, 910, 1256, 1879, 1562, 1136, 1130, 1087, 1109, 1498, 1285, 1384, 1430, 1440, 1525,
class distribution: [17917, 28566]
```

```
model, x_data_test, y_data_test, test_acc = train_session(save_model = False, earlystop = 0)  # returned variables can be optionally use
```

```
class distribution (training subset): [10181, 14380]
After oversampling (training subset):
x_data_train: (28760, 256, 16)
y_data_train: (28760, 1)
class distribution: [14380, 14380]

Performing (R)(MS)PCA...
Auto tol: 0.0009507554366505457 used tol: 5e-06
mu 0.1 lambda 0.0017022170495076935
iteration: 1, error: 94.94793486696643, ||S||: 94.88110235008492
iteration: 2, error: 6.565819780371327, ||S||: 185.3408196499094
iteration: 3, error: 4.42453273528766, ||S||: 273.0050906757732
iteration: 4, error: 4.546682015769854, ||S||: 357.5900098476372
iteration: 5, error: 4.930932859476592, ||S||: 438.676217439168
iteration: 6, error: 5.34899012602163, ||S||: 515.8078198338236
iteration: 7, error: 5.735833941831613, ||S||: 588.5517249696509
iteration: 8, error: 6.046589607713281, ||S||: 656.5797820127156
iteration: 9, error: 6.262119815175569, ||S||: 719.7525340133992
iteration: 10, error: 6.398718700035524, ||S||: 778.1572362585783
||A,L,S||: 9507.554366505457 9311.192378083506 778.1572362585783
PCA truncation 256 -> 50
x_data_train: (21570, 50, 16)
y_data_train: (21570, 1) [10820, 10750]
x_data_val: (7190, 50, 16)
y_data_val: (7190, 1) [3560, 3630]

### creating model
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ lstm1 (LSTM)                    │ (None, 50, 8)          │           800 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ drop2 (Dropout)                 │ (None, 50, 8)          │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ lstm2 (LSTM)                    │ (None, 8)              │           544 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ drop3 (Dropout)                 │ (None, 8)              │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense2 (Dense)                  │ (None, 2)              │            18 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 1,362 (5.32 KB)
 Trainable params: 1,362 (5.32 KB)
 Non-trainable params: 0 (0.00 B)

### training with (2, 3, 4, 5, 6, 7, 8, 9, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29), 21570 inputs, 7190 validation
Epoch 1/10
675/675 ━━━━━━━━━━━━━━━━━━━━ 32s 43ms/step - accuracy: 0.6215 - loss: 0.5988 - val_accuracy: 0.9135 - val_loss: 0.2397
Epoch 2/10
675/675 ━━━━━━━━━━━━━━━━━━━━ 39s 40ms/step - accuracy: 0.9247 - loss: 0.2223 - val_accuracy: 0.9346 - val_loss: 0.1779
Epoch 3/10
675/675 ━━━━━━━━━━━━━━━━━━━━ 41s 40ms/step - accuracy: 0.9515 - loss: 0.1495 - val_accuracy: 0.9606 - val_loss: 0.1166
Epoch 4/10
675/675 ━━━━━━━━━━━━━━━━━━━━ 40s 38ms/step - accuracy: 0.9644 - loss: 0.1141 - val_accuracy: 0.9648 - val_loss: 0.1051
Epoch 5/10
675/675 ━━━━━━━━━━━━━━━━━━━━ 43s 40ms/step - accuracy: 0.9696 - loss: 0.1008 - val_accuracy: 0.9683 - val_loss: 0.0939
Epoch 6/10
675/675 ━━━━━━━━━━━━━━━━━━━━ 26s 39ms/step - accuracy: 0.9733 - loss: 0.0889 - val_accuracy: 0.9694 - val_loss: 0.0921
Epoch 7/10
675/675 ━━━━━━━━━━━━━━━━━━━━ 41s 39ms/step - accuracy: 0.9748 - loss: 0.0815 - val_accuracy: 0.9711 - val_loss: 0.0899
Epoch 8/10
675/675 ━━━━━━━━━━━━━━━━━━━━ 41s 39ms/step - accuracy: 0.9787 - loss: 0.0749 - val_accuracy: 0.9744 - val_loss: 0.0841
Epoch 9/10
675/675 ━━━━━━━━━━━━━━━━━━━━ 41s 39ms/step - accuracy: 0.9796 - loss: 0.0715 - val_accuracy: 0.9733 - val_loss: 0.0890
Epoch 10/10
675/675 ━━━━━━━━━━━━━━━━━━━━ 26s 39ms/step - accuracy: 0.9797 - loss: 0.0689 - val_accuracy: 0.9752 - val_loss: 0.0804

### testing with (0, 1, 15, 16, 17), 5881 inputs
184/184 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.9353 - loss: 0.2276
```

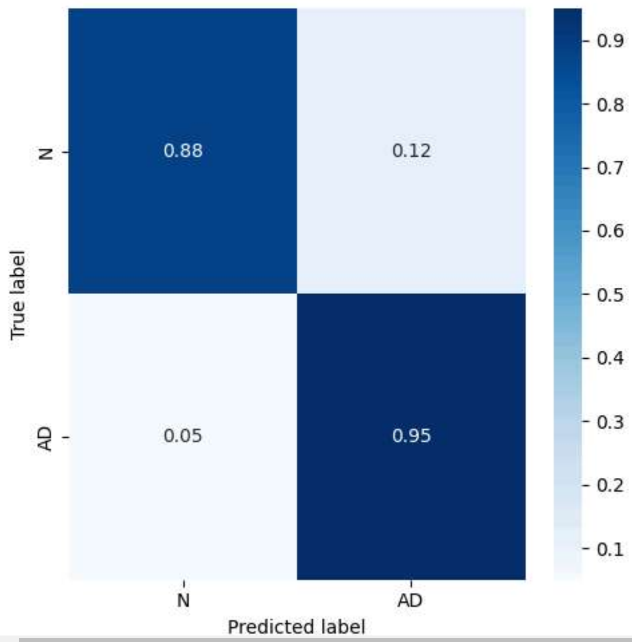
```python
# model test only
# if False and __name__ == '__main__':
print(f'### testing with {len(x_data_test)} inputs')
eval_metrics = model.evaluate(x_data_test, y_data_test)
```

```
### testing with 5881 inputs
184/184 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.9353 - loss: 0.2276
```

```python
# # create confusion matrix
# if False and __name__ == '__main__':
import pandas as pd
import seaborn
y_pred = np.argmax(model.predict(x_data_test), axis=-1)
con_mat = tf.math.confusion_matrix(labels = y_data_test, predictions = y_pred).numpy()
con_mat_norm = np.around(con_mat.astype('float') / con_mat.sum(axis = 1)[:, np.newaxis], decimals = 2)
classes = ['N', 'AD']
```

```
con_mat_df = pd.DataFrame(con_mat_norm, index = classes, columns = classes)
figure = plt.figure(figsize = (5, 5))
seaborn.heatmap(con_mat_df, annot = True, cmap = plt.cm.Blues)
plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.tight_layout()
plt.savefig(working_dir + '/confusion_matrix.eps', format='eps')
plt.show()
```

```
184/184 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step
```



Without RPCA (PCA only)

```
"""---
**Start of actual program blocks:**
"""

# create dataset, create model, train and test
# if __name__ == '__main__':
dense1 = 0
lstm1 = 8
lstm2 = 8
lstm3 = 0

window = 256
overlap = window // 2
oversample = True
decimation = 0
pca = True  # compute PCA on full data matrix
rpca = False  # compute RPCA before PCA
spikes = 1/500
rpca_mu = 0.1

subjs_train_perm = ( (tuple(i for i in range(2, 10)) + tuple(i for i in range(20, 32)), ()), )
# subjs_train_perm = ( (tuple(i for i in range(2, 15)) + tuple(i for i in range(18, 35)), ()), )
subjs_test = (0, 1, 15, 16, 17)  # 2 for N, 3 for AD
epochs = 10

  # if decimation:
  #   window //= decimation
  #   overlap //= decimation

x_data, y_data, subj_inputs = create_dataset(window, overlap, decimation)
```

```
(('01', 'N'), ('02', 'N'), ('03', 'N'), ('04', 'N'), ('05', 'N'), ('06', 'N'), ('07', 'N'), ('08', 'N'), ('09', 'N'), ('10', 'N'),

### creating dataset

tot samples: 5954304
x_data: (46483, 256, 16)
```

```
y_data: (46483, 1)
windows per subject: [868, 759, 739, 1560, 1139, 910, 1256, 1879, 1562, 1136, 1130, 1087, 1109, 1498, 1285, 1384, 1430, 1440, 1525,
class distribution: [17917, 28566]
```

```
model, x_data_test, y_data_test, test_acc = train_session(save_model = False, earlystop = 0)  # returned variables can be optionally use
```

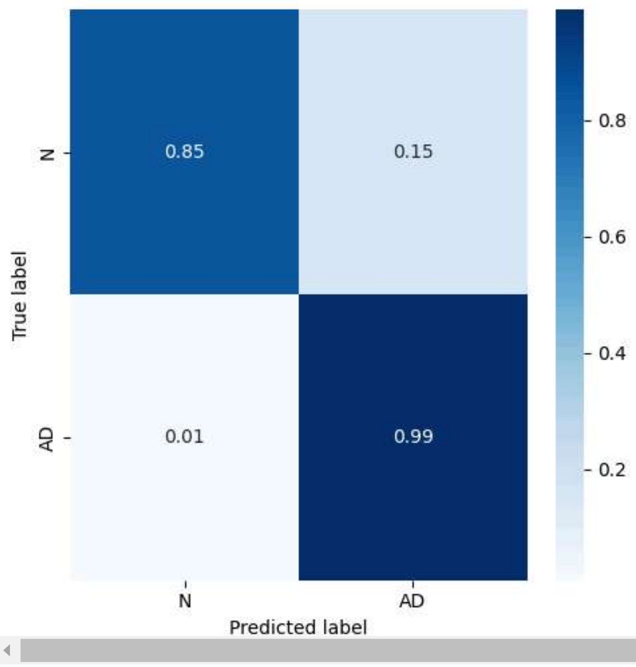
```
class distribution (training subset): [10181, 17252]
After oversampling (training subset):
x_data_train: (34504, 256, 16)
y_data_train: (34504, 1)
class distribution: [17252, 17252]

Performing (R)(MS)PCA...
PCA truncation 256 -> 50
x_data_train: (25878, 50, 16)
y_data_train: (25878, 1) [12947, 12931]
x_data_val: (8626, 50, 16)
y_data_val: (8626, 1) [4305, 4321]

### creating model
Model: "sequential_1"
┌───────────────────────────────┬──────────────────────────┬──────────────┐
│ Layer (type)                  │ Output Shape             │      Param # │
├───────────────────────────────┼──────────────────────────┼──────────────┤
│ lstm1 (LSTM)                  │ (None, 50, 8)            │          800 │
├───────────────────────────────┼──────────────────────────┼──────────────┤
│ drop2 (Dropout)               │ (None, 50, 8)            │            0 │
├───────────────────────────────┼──────────────────────────┼──────────────┤
│ lstm2 (LSTM)                  │ (None, 8)                │          544 │
├───────────────────────────────┼──────────────────────────┼──────────────┤
│ drop3 (Dropout)               │ (None, 8)                │            0 │
├───────────────────────────────┼──────────────────────────┼──────────────┤
│ dense2 (Dense)                │ (None, 2)                │           18 │
└───────────────────────────────┴──────────────────────────┴──────────────┘
 Total params: 1,362 (5.32 KB)
 Trainable params: 1,362 (5.32 KB)
 Non-trainable params: 0 (0.00 B)

### training with (2, 3, 4, 5, 6, 7, 8, 9, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31), 25878 inputs, 8626 validation
Epoch 1/10
809/809 ━━━━━━━━━━━━━━━━━━━━ 36s 41ms/step - accuracy: 0.7298 - loss: 0.5079 - val_accuracy: 0.9408 - val_loss: 0.1736
Epoch 2/10
809/809 ━━━━━━━━━━━━━━━━━━━━ 39s 39ms/step - accuracy: 0.9419 - loss: 0.1719 - val_accuracy: 0.9616 - val_loss: 0.1163
Epoch 3/10
809/809 ━━━━━━━━━━━━━━━━━━━━ 41s 39ms/step - accuracy: 0.9613 - loss: 0.1249 - val_accuracy: 0.9682 - val_loss: 0.0982
Epoch 4/10
809/809 ━━━━━━━━━━━━━━━━━━━━ 34s 42ms/step - accuracy: 0.9663 - loss: 0.1092 - val_accuracy: 0.9728 - val_loss: 0.0894
Epoch 5/10
809/809 ━━━━━━━━━━━━━━━━━━━━ 38s 39ms/step - accuracy: 0.9696 - loss: 0.0977 - val_accuracy: 0.9754 - val_loss: 0.0799
Epoch 6/10
809/809 ━━━━━━━━━━━━━━━━━━━━ 41s 39ms/step - accuracy: 0.9739 - loss: 0.0862 - val_accuracy: 0.9762 - val_loss: 0.0758
Epoch 7/10
809/809 ━━━━━━━━━━━━━━━━━━━━ 41s 39ms/step - accuracy: 0.9749 - loss: 0.0812 - val_accuracy: 0.9789 - val_loss: 0.0739
Epoch 8/10
809/809 ━━━━━━━━━━━━━━━━━━━━ 41s 39ms/step - accuracy: 0.9773 - loss: 0.0740 - val_accuracy: 0.9772 - val_loss: 0.0725
Epoch 9/10
809/809 ━━━━━━━━━━━━━━━━━━━━ 41s 39ms/step - accuracy: 0.9791 - loss: 0.0708 - val_accuracy: 0.9790 - val_loss: 0.0689
Epoch 10/10
809/809 ━━━━━━━━━━━━━━━━━━━━ 41s 39ms/step - accuracy: 0.9804 - loss: 0.0654 - val_accuracy: 0.9798 - val_loss: 0.0649
```

```
# model test only
# if False and __name__ == '__main__':
print(f'### testing with {len(x_data_test)} inputs')
eval_metrics = model.evaluate(x_data_test, y_data_test)
```

```
### testing with 5881 inputs
184/184 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.9301 - loss: 0.2505
```

```
# # create confusion matrix
# if False and __name__ == '__main__':
import pandas as pd
import seaborn
y_pred = np.argmax(model.predict(x_data_test), axis=-1)
con_mat = tf.math.confusion_matrix(labels = y_data_test, predictions = y_pred).numpy()
con_mat_norm = np.around(con_mat.astype('float') / con_mat.sum(axis = 1)[:, np.newaxis], decimals = 2)
classes = ['N', 'AD']
con_mat_df = pd.DataFrame(con_mat_norm, index = classes, columns = classes)
figure = plt.figure(figsize = (5, 5))
seaborn.heatmap(con_mat_df, annot = True, cmap = plt.cm.Blues)
plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.tight_layout()
plt.savefig(working_dir + '/confusion_matrix_withoutRPCA.eps', format='eps')
plt.show()
```

```
184/184 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step
```


True label
Predicted label
N
AD
N
AD