전방추락 v.s. 뛰어내림 분류 알고리즘 경량화

0. 요약

전방추락과 뛰어내림 동작 이진 분류 모델 경량화 진행
특성벡터수(15->8), 윈도우크기(25->9) 변경으로 정확도 100%
유지하면서 용량 및 연산량 44% 감축

1. 목적

- 개발된 전방추락과 뛰어내림 분류 알고리즘의 특성벡터 수와 쉬프팅 윈도우 크기 경량화

2. 방법

- 연대에서 공유된 추락보호복 1~2차년도 데이터를 기반으로 유전 알고리즘과 Grid Search 기법을 적용하여 최적화

3. 특성벡터 수 최적화 결과: 9개

- 최적화 방법
  : GeneticSelectionCV와 RandomForestClassifier 적용

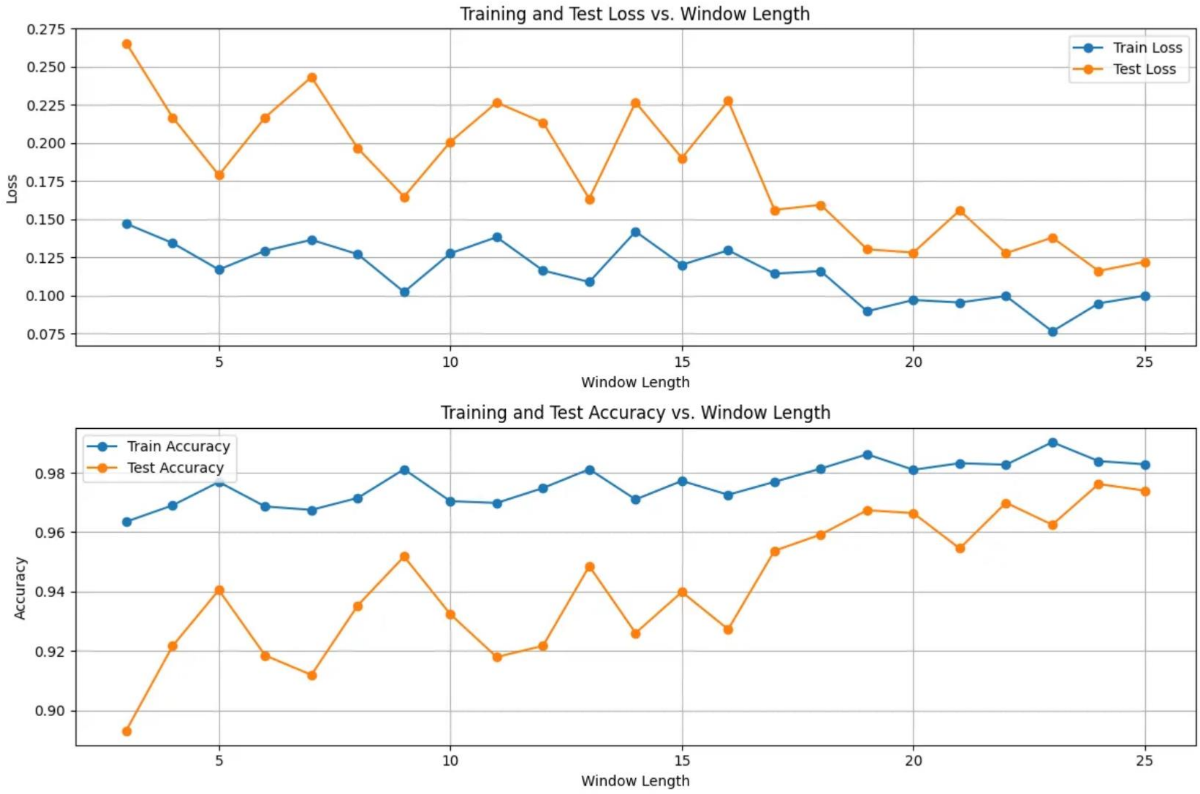
```
Selecting features with genetic algorithm.
gen nevals  avg                               std                                      min                                   max
0   50      [ 0.799664  6.38       0.039431] [ 0.101257  3.321987  0.01747 ] [ 0.511637  1.         0.003339] [ 0.90655   12.         0.090782]
1   33      [-199.153766    8.6       200.041087] [ 1400.120892     2.57682    1399.99413 ] [-10000.          2.          0.011683] [     0.90655     13.        10000.       ]
2   34      [-399.143816    9.58      400.039039] [ 1959.766562     1.844885  1959.583825] [-10000.         5.         0.01306]  [     0.908447     13.        10000.       ]
3   34      [-199.117909    9.72      200.041107] [ 1400.126013     1.575309  1399.994128] [-10000.          5.          0.015225] [     0.912031     14.        10000.       ]
4   31      [ 0.903234  9.5        0.041653]        [ 0.011368  1.268858  0.008833]          [ 0.845203  7.         0.015225]          [ 0.912031  12.         0.060409]
5   27      [-199.110277    9.42      200.040084] [ 1400.127103     1.429545  1399.994274] [-10000.          7.          0.018099] [     0.916906     13.        10000.       ]
6   30      [ 0.908824  8.78       0.039177]        [ 0.004442  1.136486  0.006773]          [ 0.8901    7.         0.016508]          [ 0.916906  12.         0.048927]
7   26      [ 0.911427  8.46       0.039181]        [ 0.00217   0.963535  0.004137]          [ 0.90744   7.         0.018099]          [ 0.916906  11.         0.045886]
8   33      [ 0.911327  8.22       0.039019]        [ 0.007816  0.782049  0.004956]          [ 0.859011  7.         0.021929]          [ 0.916906  10.         0.061556]
9   36      [ 0.913113  8.2        0.038999]        [ 0.003227  0.69282   0.00258 ]          [ 0.89661   7.         0.035157]          [ 0.917387  10.         0.048284]
10  32      [ 0.914438  8.14       0.03834 ]        [ 0.002069  0.4005    0.001949]          [ 0.907469  7.         0.034724]          [ 0.918051  9.        0.045031]
11  30      [ 0.913824  8.02       0.039087]        [ 0.004452  0.468615  0.003184]          [ 0.895632  7.         0.034724]          [ 0.918241  10.         0.054889]
12  22      [ 0.91557   8.02       0.037421]        [ 0.001951  0.244131  0.003003]          [ 0.907462  7.         0.02105 ]          [ 0.918241  9.        0.044061]
13  26      [ 0.915777  8.04       0.037382]        [ 0.002192  0.344093  0.003295]          [ 0.90539  7.        0.01974]           [ 0.918241  10.         0.046495]
14  29      [ 0.916059  8.         0.037797]        [ 0.001716  0.2       0.001644]          [ 0.911651  7.         0.035157]          [ 0.918241  9.        0.042317]
15  31      [ 0.915521  7.94       0.038114]        [ 0.002603  0.310483  0.00212 ]          [ 0.905134  7.         0.034919]          [ 0.918241  9.        0.046741]
Selected Features: [False False False  True False  True False False  True  True  True  True  False  True  True]
GryoX, GyroZ, lpf_acc, lpf_gyro, snr, vv, pitch, jerk
```

- 기존 사용되던 특성벡터 입력값: (25, 15)
  : 3축 가속도, 3축 각속도, raw_acc, raw_gyro, lpf_acc, lpf_gyro, snr, vv, roll, pitch, jerk
- 최적화된 특성벡터 입력값: (25, 8)
  : GryoX, GyroZ, lpf_acc, lpf_gyro, snr, vv, pitch, jerk

4. window length 최적화 결과: 9 frames

- 최적화 방법

  : Grid Search로 window length의 변화(3~25)에 따른 훈련 정확도와 평가 정확도 분석



- 분석결과

  - window length가 증가할수록 loss가 감소하고, 정확도 증가하는 경향 보이지만, 생각보다 성능 차이가 크지는 않음

  - window length 15 frames 이하 조건들 중에서는 9 frames이 가장 작은 loss와 큰 정확도를 보임

5. 딥러닝 모델 및 추가 평가 기법 최적화

- **최적화 방법**

  : 추가 평가 기법들 (threshold, voting_type, voting_size) 변화에 따른 모델의 train_acc와 test_acc 모니터링

- **최종 딥러닝 모델**

  - Advanced DNN 모델

    > Dense 레이어 쌓아갈수록 unit을 작게두어 압축하는 구조

    > Bottleneck과 shortcut을 두어 기울기 소실 방지하면서, 적은 용량으로 층을 비교적 깊게 쌓는 구조

    > Batch Normalization과 Drop out으로 오버피팅 방지

Model: "model_8"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	[(None, 9, 8)]	0	[]
flatten (Flatten)	(None, 72)	0	['input_layer[0][0]']
reduce_initial (Dense)	(None, 64)	4672	['flatten[0][0]']
bn_initial (BatchNormalization)	(None, 64)	256	['reduce_initial[0][0]']
reduce1 (Dense)	(None, 32)	2080	['bn_initial[0][0]']
bn1 (BatchNormalization)	(None, 32)	128	['reduce1[0][0]']
tf.nn.relu_44 (TFOpLambda)	(None, 32)	0	['bn1[0][0]']
dropout_bottleneck1 (Dropout)	multiple	0	['tf.nn.relu_44[0][0]', 'tf.nn.relu_45[0][0]', 'bn_expand1[0][0]']
bottleneck1 (Dense)	(None, 16)	528	['dropout_bottleneck1[0][0]']
tf.nn.relu_45 (TFOpLambda)	(None, 16)	0	['bottleneck1[0][0]']
expand1 (Dense)	(None, 32)	544	['dropout_bottleneck1[1][0]']
bn_expand1 (BatchNormalization)	(None, 32)	128	['expand1[0][0]']
projection1 (Dense)	(None, 32)	2080	['bn_initial[0][0]']
bn_projection1 (BatchNormalization)	(None, 32)	128	['projection1[0][0]']
tf.__operators__.add_12 (TFOpLambda)	(None, 32)	0	['dropout_bottleneck1[2][0]', 'bn_projection1[0][0]']
tf.nn.relu_46 (TFOpLambda)	(None, 32)	0	['tf.__operators__.add_12[0][0
reduce2 (Dense)	(None, 32)	1056	['tf.nn.relu_46[0][0]']
bn2 (BatchNormalization)	(None, 32)	128	['reduce2[0][0]']
tf.nn.relu_47 (TFOpLambda)	(None, 32)	0	['bn2[0][0]']
dropout_bottleneck2 (Dropout)	multiple	0	['tf.nn.relu_47[0][0]', 'tf.nn.relu_48[0][0]', 'bn_expand2[0][0]']
bottleneck2 (Dense)	(None, 16)	528	['dropout_bottleneck2[0][0]']
tf.nn.relu_48 (TFOpLambda)	(None, 16)	0	['bottleneck2[0][0]']
expand2 (Dense)	(None, 32)	544	['dropout_bottleneck2[1][0]']
bn_expand2 (BatchNormalization)	(None, 32)	128	['expand2[0][0]']
projection2 (Dense)	(None, 32)	1056	['tf.nn.relu_46[0][0]']
bn_projection2 (BatchNormalization)	(None, 32)	128	['projection2[0][0]']
tf.__operators__.add_13 (TFOpLambda)	(None, 32)	0	['dropout_bottleneck2[2][0]', 'bn_projection2[0][0]']
tf.nn.relu_49 (TFOpLambda)	(None, 32)	0	['tf.__operators__.add_13[0][0]
output_dense1 (Dense)	(None, 16)	528	['tf.nn.relu_49[0][0]']
bn_output1 (BatchNormalization)	(None, 16)	64	['output_dense1[0][0]']
dropout1 (Dropout)	(None, 16)	0	['bn_output1[0][0]']
output_dense2 (Dense)	(None, 8)	136	['dropout1[0][0]']
bn_output2 (BatchNormalization)	(None, 8)	32	['output_dense2[0][0]']
dropout2 (Dropout)	(None, 8)	0	['bn_output2[0][0]']
output_dense3 (Dense)	(None, 4)	36	['dropout2[0][0]']
bn_output3 (BatchNormalization)	(None, 4)	16	['output_dense3[0][0]']
dropout3 (Dropout)	(None, 4)	0	['bn_output3[0][0]']
output_layer (Dense)	(None, 2)	10	['dropout3[0][0]']

Total params: 14,934
Trainable params: 14,366
Non-trainable params: 568

None
Flops: 28992

- **Grid Search 결과**

threshol	voting_typ	voting_siz	train_accurac	test_accurac	train_lead_time_mear	train_lead_time_std	test_lead_time_mear	test_lead_time_std
0.6	consecutive	9	100	100	539.3333333	108.9525891	559.047619	91.64656551
0.5	consecutive	10	100	100	532.4444444	108.2621419	542.8571429	86.91226321
0.55	consecutive	10	100	100	524.2222222	107.9297897	540.952381	90.39093728
0.7	consecutive	8	99.44444444	100	548.0898876	107.7541564	569.5238095	95.94026864
0.65	consecutive	9	99.44444444	100	535.2808989	105.4481063	556.1904762	93.11928809
0.6	consecutive	10	99.44444444	100	522.6966292	104.8140573	539.047619	91.64656551
0.9	majority	10	99.44444444	100	493.0337079	121.1221363	512.3809524	122.431896
0.75	consecutive	8	98.88888889	100	540.6818182	111.087552	567.6190476	97.8777061
0.8	consecutive	8	98.88888889	100	528.6363636	114.0093878	560.952381	105.0126868
0.9	consecutive	6	98.88888889	100	531.3636364	134.2248945	554.2857143	151.3297072
0.7	consecutive	9	98.88888889	100	529.3181818	104.7485675	549.5238095	95.94026864
0.85	consecutive	8	98.88888889	100	509.5454545	126.7057175	548.5714286	121.6664337
0.75	consecutive	9	98.88888889	100	518.4090909	112.299178	547.6190476	97.8777061
0.8	consecutive	9	98.88888889	100	508.6363636	114.0093878	540.952381	105.0126868
0.65	consecutive	10	98.88888889	100	513.8636364	105.458219	536.1904762	93.11928809
0.7	consecutive	10	98.88888889	100	506.5909091	107.5225808	529.5238095	95.94026864
0.85	consecutive	9	98.88888889	100	484.7727273	129.263449	528.5714286	121.6664337
0.9	consecutive	7	98.88888889	100	507.0454545	134.7064071	527.6190476	145.9910226
0.75	consecutive	10	98.88888889	100	498.4090909	112.299178	527.6190476	97.8777061
0.8	consecutive	10	98.88888889	100	485.9090909	115.6074506	520.952381	105.0126868
0.85	consecutive	10	98.88888889	100	464.7727273	129.263449	508.5714286	121.6664337
0.9	consecutive	8	98.88888889	100	484.7727273	135.4787578	507.6190476	145.9910226
0.9	consecutive	9	98.88888889	100	458.8636364	137.9841214	487.6190476	145.9910226
0.9	consecutive	10	98.88888889	100	436.3636364	140.2771754	467.6190476	145.9910226
0.85	consecutive	4	100	97.61904762	591.3333333	128.0989201	628.5714286	121.6664337
0.8	consecutive	5	100	97.61904762	587.3333333	120.533997	620.952381	105.0126868
0.85	consecutive	5	100	97.61904762	571.3333333	128.0989201	608.5714286	121.6664337
0.85	majority	6	100	97.61904762	572	118.764005	605.7142857	103.4901159
0.8	consecutive	6	100	97.61904762	565.1111111	122.8390496	600.952381	105.0126868
0.85	majority	7	100	97.61904762	560.8888889	115.6589281	591.4285714	96.91850844

- **최종 검출 성능**

- 훈련 데이터셋

```
Sensitivity: 100.00 % (90/90)
Specificity: 100.00 % (90/90)
Accuracy: 100.00 % (180/180)
Lead Time: 539.33 ± 108.95 ms
```

- 평가 데이터셋

```
Sensitivity: 100.00 % (21/21)
Specificity: 100.00 % (21/21)
Accuracy: 100.00 % (42/42)
Lead Time: 559.05 ± 91.65 ms
```

- **파라미터 수 & FLOPs 비교**

	노인낙상모델	기개발모델	최적화된모델
입력버퍼크기	(25, 6)	(25, 15)	(9, 8)
파라미터수	22,274	26,750	14,934
FLOPs	44,300	53,392	28,992