Inteligência Artificial

Human Activity Recognition usando deep learning (ML) e video datasets

Mestrado em Computação Móvel 2021/2022

Aluna:
Maria Barbosa, Nº 36502

Índice

Índice de Imagens

1. Introdução

Neste projeto pretende-se, utilizando vários datasets disponibilizados adicionando mais informação aos datasets escolhidos, utilizar sistematicamente o software escrito em python para testar e comparar diversos esquemas de deep learning para reconhecimento de atividade humana nesses datasets. O cenário de utilização escolhido é “smart home”.

1.1. Motivação

A aplicação prática dos conceitos abordados neste documento permite classificar ou prever a atividade / ação que está a ser realizada por alguém - reconhecimento de atividade.

1.2. Objetivos

O objetivo deste projeto é treinar o modelo, este aprende a distinguir entre duas ações semelhantes usando o contexto ambiental, perceber como funciona e testar a sua eficiência.

2.Descrição do problema

O Reconhecimento de Atividade Humana é um tipo de problema de classificação de série temporal em que precisamos de dados de uma série de etapas de tempo para classificar corretamente a ação que está sendo executada.

A técnica mais comum e eficaz é anexar um sensor vestível (por exemplo, um smartphone) a uma pessoa e, em seguida, treinar um modelo temporal como um LSTM na saída dos dados do sensor.

Neste projeto será realizada uma análise e comparação Single-Frame CNN e CNN com LSTM.

3. Estado da Arte

Um dos cenários que irei abordar neste relatório é o single-frame cnn. O single-frame é um exemplo de classificação de vídeos simplesmente agregando previsões em quadros / imagens individuais.

A implementação mais básica da classificação de vídeo é o uso de uma rede de classificação de imagem. Executarei um modelo de classificação de imagem em cada quadro do vídeo e, em seguida, a média de todas as probabilidades individuais para obter o vetor de probabilidades final.

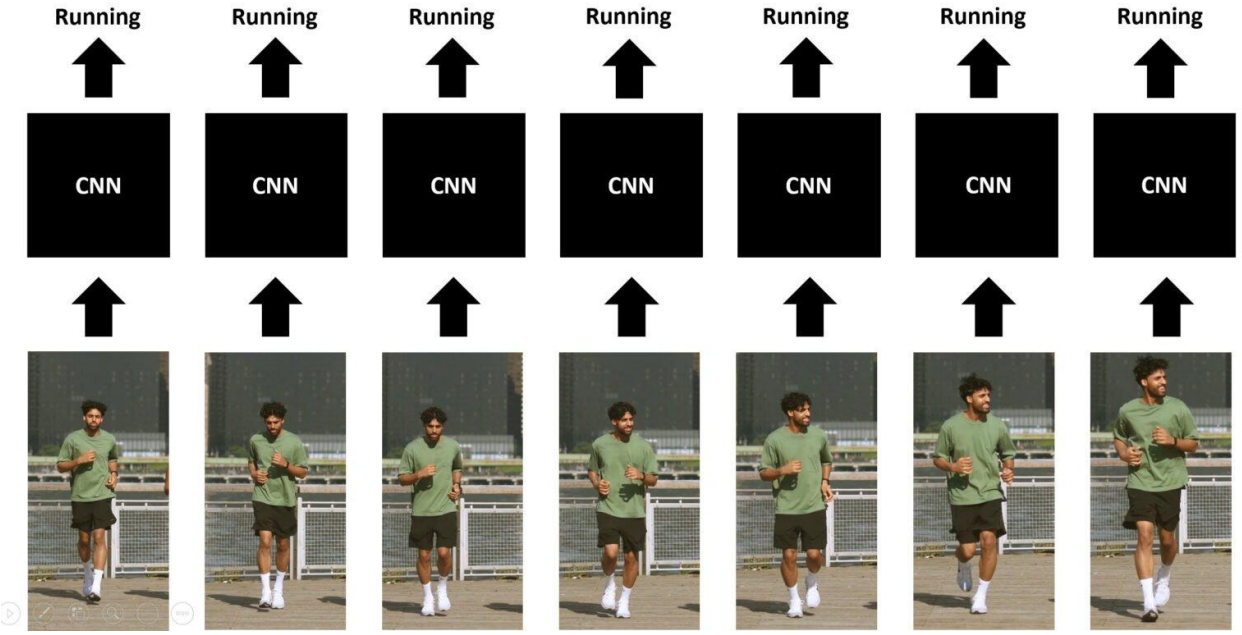

Fig.1 - Single Frame CNN

O outro cenário que irei abordar é CNN com LSTMs, o objetivo nesta abordagem é utilizar redes convolucionais para extrair características locais de cada quadro. As saídas dessas redes convolucionais independentes são alimentadas a uma rede LSTM multicamadas muitos-para-um para fundir temporariamente essas informações extraídas.

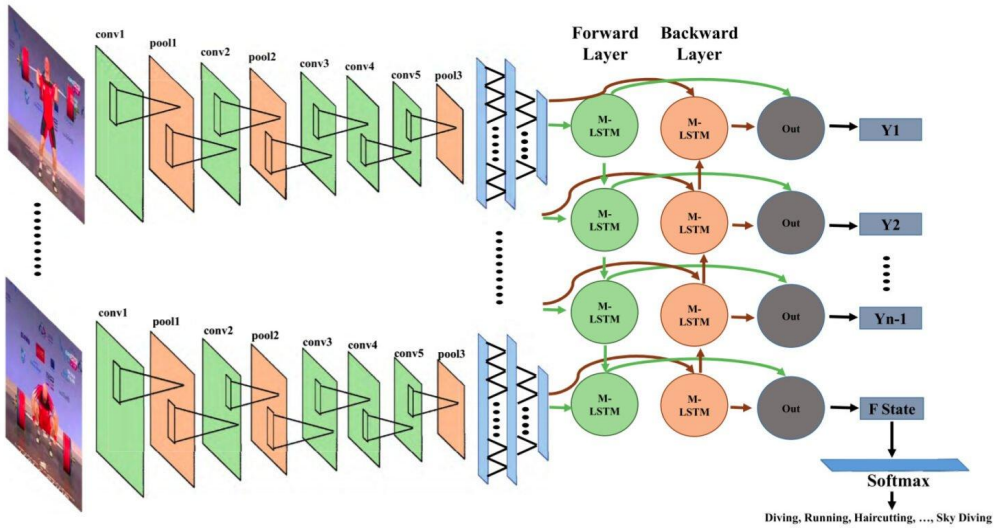

Fig.2 - CNN com arquitetura LSTM bidirecional

4. Descrição do trabalho realizado pelo autor

Para o desenvolvimento deste projeto como referi anteriormente eu vou utilizar dois algoritmos de classificação de vídeo diferentes e compará-los entre si. Utilizei a biblioteca Keras e repliquei o código presente no artigo [1].

Visto que a minha máquina possui algumas limitações executei e implementei o algoritmo no colab, onde implementei 3 classes diferentes Walking With Dog, TaiChi e Horse Race. Para a comparação ser consistente usei as mesmas classes nos duas implementações.

```
features, labels = create_dataset()

Extracting Data of Class: WalkingWithDog
Extracting Data of Class: TaiChi
Extracting Data of Class: Swing
Extracting Data of Class: HorseRace
Extracting Data of Class: PushUps
Extracting Data of Class: Punch
```

Fig.3 - Extração dos modelos

Para treinar o modelo no algoritmos foi necessário definir um dado número de épocas, sendo que maior o número de épocas maior a accuracy, o número escolhido foi 50.

Utilizei o dataset UCF50 [2], que é um conjunto de dados de reconhecimento de ações que contém várias categorias de ação.Está organizado por pastas, cada pasta é referente a uma categoria e contém vídeos dessa mesma categoria. Coloquei o dataset no google drive e colab passo por parâmetro o caminho até esta pasta.

Nas imagens que se seguem podemos observar o resultado da criação dos dois modelos (imagem 4 e 5) e em seguida a estrutura dos respectivos modelos finais (imagem 5 e 6).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 64)        1792
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 60, 60, 64)        36928
_________________________________________________________________
batch_normalization (BatchNo (None, 60, 60, 64)        256
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 30, 30, 64)        0
_________________________________________________________________
global_average_pooling2d (Gl (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 256)               16640
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 1542
=================================================================
Total params: 58,182
Trainable params: 57,542
Non-trainable params: 640
_________________________________________________________________
Model Created Successfully!
```

Fig.4 - Criação do modelo Single Frame CNN

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
time_distributed (TimeDistri (None, None, 62, 64)      640
_________________________________________________________________
time_distributed_1 (TimeDist (None, None, 60, 64)      12352
_________________________________________________________________
time_distributed_2 (TimeDist (None, None, 60, 64)      0
_________________________________________________________________
time_distributed_3 (TimeDist (None, None, 30, 64)      0
_________________________________________________________________
time_distributed_4 (TimeDist (None, None, 1920)        0
_________________________________________________________________
lstm (LSTM)                  (None, 100)               808400
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0
_________________________________________________________________
dense (Dense)                (None, 64)                6464
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
=================================================================
Total params: 828,051
Trainable params: 828,051
Non-trainable params: 0
_________________________________________________________________
Model Created Successfully!
```

Fig.5 - Criação do modelo CNN com LSTM

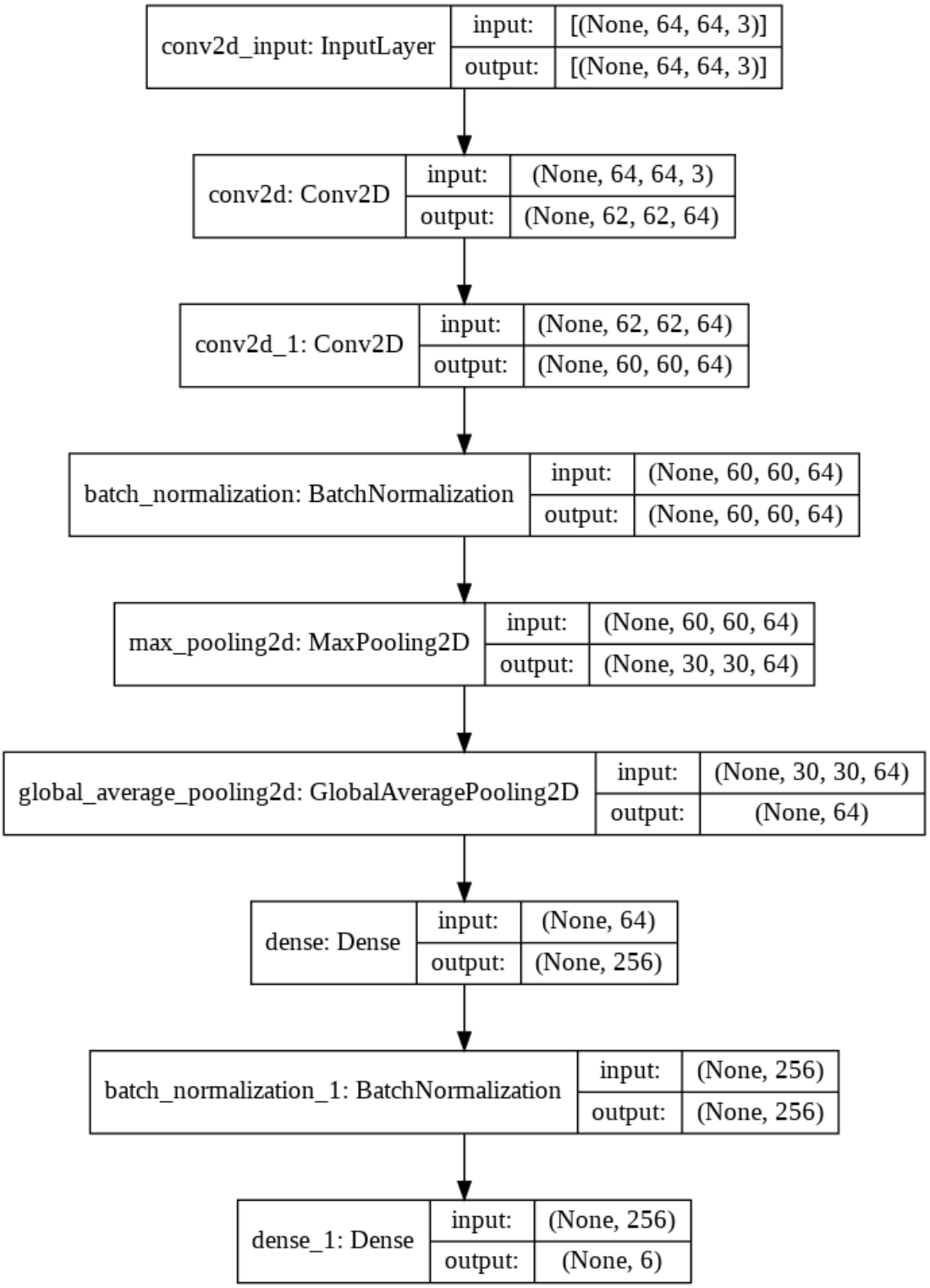

Fig.6 - Modelo Single Frame CNN

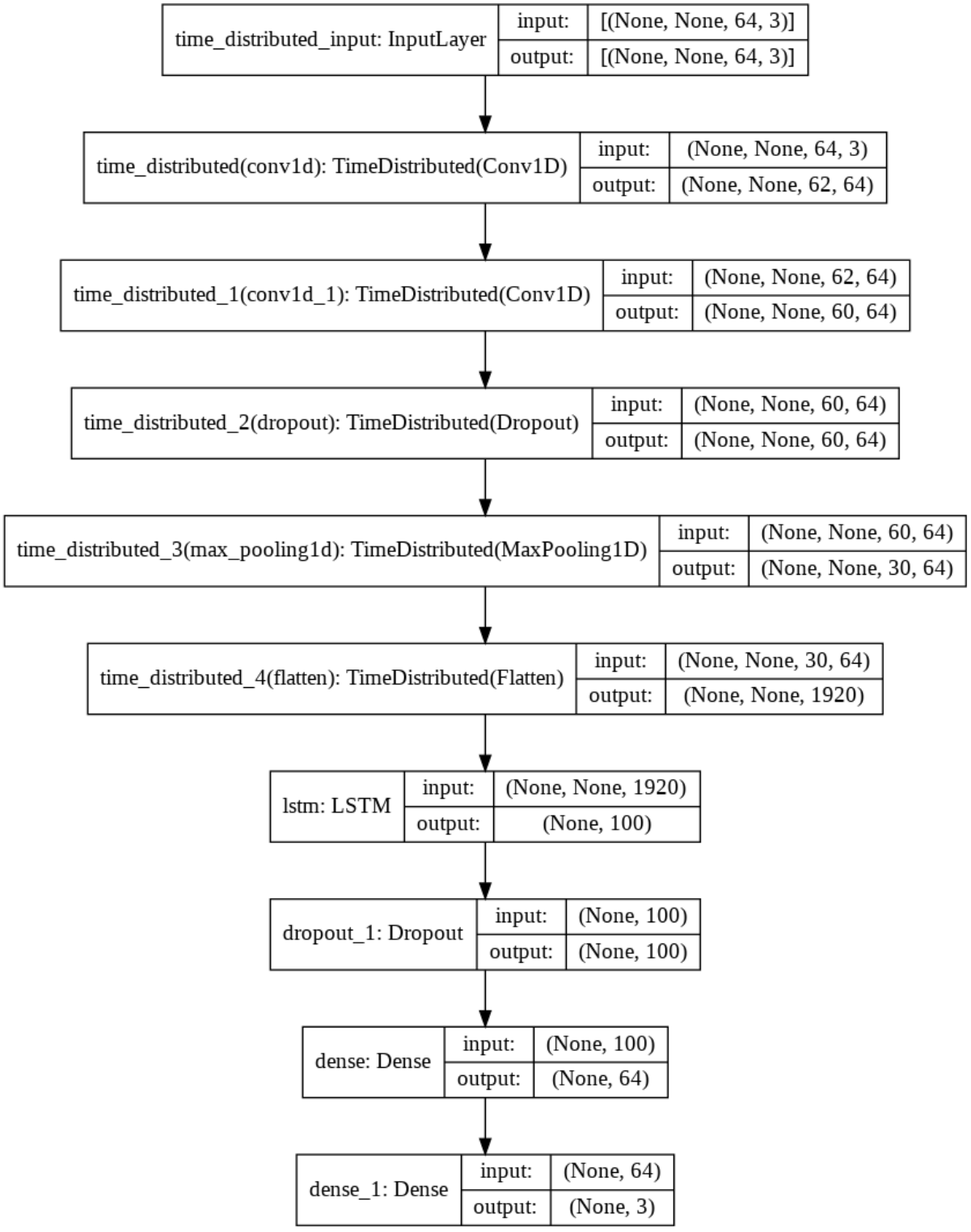

Fig.7 - Modelo CNN com LSTM

5. Análise de Resultados

Como referi anteriormente, para conseguirmos comparar os dois algoritmos e obtermos uma comparação concreta utilizei as mesmas classes nos dois algoritmos.

Inicialmente obtive a avaliação dos modelos (presente nas seguintes imagens).

```
model_evaluation_history = model.evaluate(features_test, labels_test)

150/150 [==============================] - 1s 5ms/step - loss: 0.0043 - accuracy: 0.9994
```

Fig.8 - Evaluation do modelo Single Frame CNN

```
model_evaluation_history = model.evaluate(features_test, labels_test)

150/150 [==============================] - 2s 9ms/step - loss: 0.0096 - accuracy: 0.9977
```

Fig.9 - Evaluation do modelo CNN com LSTM

```
# Downloading The YouTube Video
video_title = download_youtube_videos('https://www.youtube.com/watch?v=XqqpZS0c1K0', output_directory)

# Construting The Input YouTube Video Path
input_video_file_path = f'{output_directory}/{video_title}.mp4'

# Calling The Make Average Method To Start The Process
make_average_predictions(input_video_file_path, 50)

# Play Video File in the Notebook
VideoFileClip(input_video_file_path).ipython_display(width = 700)
```

```
CLASS NAME: WalkingWithDog   AVERAGED PROBABILITY: 1e+02
CLASS NAME: HorseRace   AVERAGED PROBABILITY: 0.013
CLASS NAME: TaiChi   AVERAGED PROBABILITY: 1.6e-05
100%|██████████| 1213/1213 [00:01<00:00, 789.34it/s]
100%|██████████| 1651/1651 [00:24<00:00, 68.75it/s]
```

Fig.10 - Resultados para o algoritmo SIngle Frame CNN (Walking with dog)

```
# Downloading The YouTube Video Walking with dog
video_title = download_youtube_videos('https://www.youtube.com/watch?v=XqqpZS0c1K0', output_directory)

# Construting The Input YouTube Video Path
input_video_file_path = f'{output_directory}/{video_title}.mp4'

# Calling The Make Average Method To Start The Process
make_average_predictions(input_video_file_path, 50)

# Play Video File in the Notebook
VideoFileClip(input_video_file_path).ipython_display(width = 700)
```

```
CLASS NAME: WalkingWithDog   AVERAGED PROBABILITY: 1e+02
CLASS NAME: TaiChi   AVERAGED PROBABILITY: 0.0025
CLASS NAME: HorseRace   AVERAGED PROBABILITY: 0.00046
100%|██████████| 1213/1213 [00:01<00:00, 981.87it/s]
100%|██████████| 1651/1651 [00:23<00:00, 69.40it/s]
```

Fig.11 - Resultados para o algoritmo CNN com LSTM

Como podemos observar no algoritmo Single Frame visto que as frames isoladas são analisadas sem contexto temporal para a classificação de atividade verifica-se uma taxa de insucesso quando os inputs não coincidem com as atividades treinadas no modelo.

Quando repetimos o processo mas com o algoritmo CNN com LSTM com os meus inputs, conseguiu-se uma melhoria nos resultados.

Nas seguintes imagens podemos observar uma comparação gráfica entre os dois algoritmos onde comparamos a taxa de acerto e a taxa de perdas dos algoritmos..

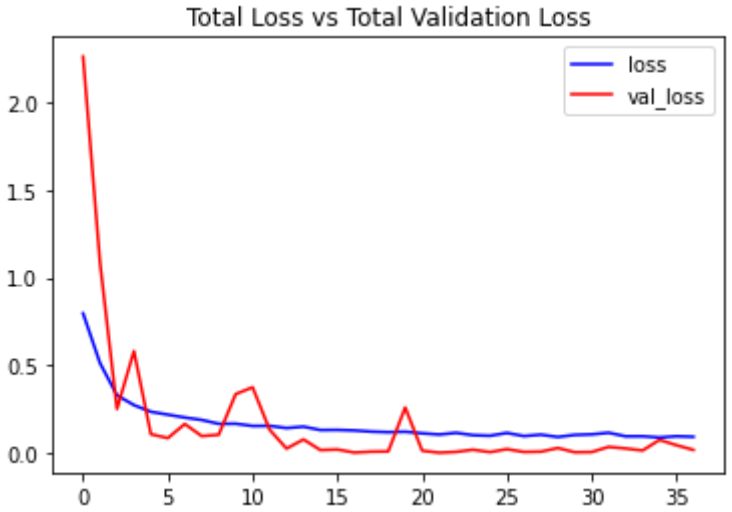

Fig.12 - Single Frame CNN Loss

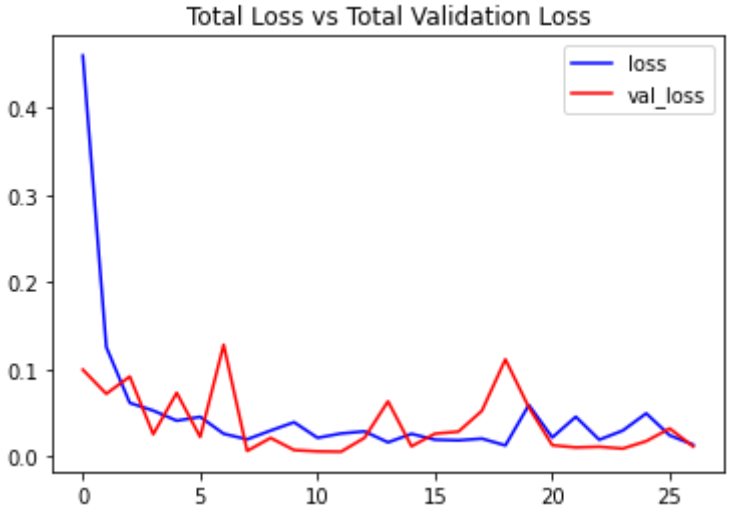

Fig.13 - CNN com LSTM Loss

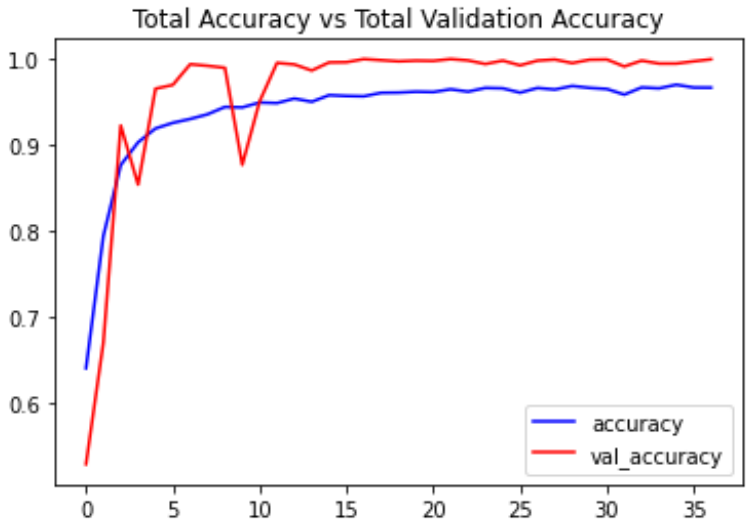

Fig.14 - Single Frame CNN Accuracy

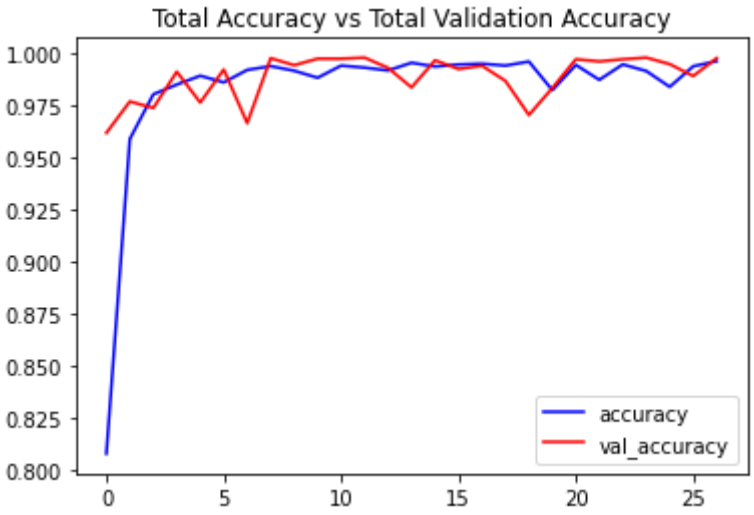

Fig.15 - CNN com LSTM Accuracy

6. Conclusões e Perspectivas de Desenvolvimento

Com a implementação e testagem destes dois algoritmos consigo concluir que com o CNN com LSTM conseguimos atingir , normalmente, uma taxa de acerto geral em relação ao SIngle Frame CNN.

Posso também concluir que um LSTM é projetado para funcionar de forma diferente de uma CNN porque um LSTM é geralmente usado para processar e fazer previsões dadas sequências de dados (em contraste, uma CNN é projetada para explorar "correlação espacial" em dados e funciona bem em imagens e voz).

7. Referências

[1] https://learnopencv.com/introduction-to-video-classification-and-human-activity-recognition

[2] https://machinelearningmastery.com/cnn-long-short-term-memory-networks/

https://medium.com/ai-ml-at-symantec/should-we-abandon-lstm-for-cnn-83accaeb93d6

Repositório: https://github.com/InesCapela/IA/