Reconhecimento de atividade humana usando deep learning e video datasets

Projeto Prático de Inteligência Artificial

2021

Autores:	Josué Braz nº 35558 Diogo Araújo nº 35128
Docente:	Prof. José Manuel Torres

Índice

1. Introdução

Este documento dedica-se à descrição detalhada da elaboração do projeto final para a disciplina de Inteligência Artificial, que consiste em explorar esquemas de Deep-Learning, manipular datasets em linguagem Python, com auxílio do Keras e de outras bibliotecas. Através de vários datasets foi-nos possível treinar vários modelos cujo foco é o reconhecimento de atividade humana

1.1. Motivação

O projeto desenvolvido aprofunda o estudo de Machine Learning, lecionado ao longo das aulas de Inteligência Artificial do Mestrado em Engenharia Informática, proporcionando-nos um melhor conhecimento para o desenvolvimento deste projeto e a compreensão necessária para alguns dos esquemas trabalhados.

Este projeto trata de um conjunto de dados sobre atividade humana, dados estes interessantes para serem usados em reconhecer e prever ações através de imagens, que potencialmente possui vários usos.

1.2. Objetivos

Ao escolher este conjunto de dados, pretendíamos, com o auxílio de ferramentas e esquemas de Deep-Learning, como a biblioteca Keras, tratar de avaliar a predição de atividades humanas através de um vídeo, verificando isto através dos diversos valores apresentados para cada frame do mesmo, a sua comparação e como estes podem ter impacto na decisão final. Após isto, é-nos possível obter uma percentagem aproximada da veracidade dos dados através do seu nível de precisão e do seu nível de perda. Após esta avaliação e tratamento rígido dos dados, é possível verificar a probabilidade de a previsão estar certa.

2. Descrição do Problema

O desafio principal deste projeto envolve a análise e comparação de vários métodos de classificação de vídeo, que fazem uma previsão da atividade com base em modelos já previamente treinados. Para este trabalho foram implementados, estudados e comparados 2 métodos, o método Single-Frame CNN e CNN com LSTMs.

O conjunto de dados escolhido foi um grupo de imagens que possuem variadas atividades humanas, com 50 categorias de ações. A maioria dos data sets não são realistas e são encenados por atores.

Posteriormente, em código Python, foi necessário carregar estes dados, para então os visualizar graficamente e manipular. Para isto, foram utilizadas várias bibliotecas, referidas mais abaixo, que proporcionaram a visualização dos dados em diferentes tipos de gráficos para melhor compreender o problema. Após a compreensão dos dados deste conjunto, procedeu-se ao real objetivo, que é avaliar a precisão destes dados. Após importar os dados é necessário, então, proceder à implementação do modelo de redes neuronais em Keras (modelo NN).

Com o modelo pronto a ser usado é necessário de seguida compila-lo. Esta compilação utiliza o Tensorflow que oferece várias bibliotecas para executar esta função. Após a sua compilação, o modelo sequencial está pronto a ser trabalhado com os dados. Nesta fase podemos então calcular as percentagens de precisão do modelo, ou seja a fiabilidade dos dados e o valor das perdas. A precisão varia á medida que trabalhamos e ajustamos o modelo ao conjunto de dados. Os valores são posteriormente representados graficamente o que nos permite fazer uma avaliação dos mesmos.

3. Estado da Arte

Sabemos que a inteligência artificial pode ser aplicada em diversas áreas e contextos e que diversos estudos já foram realizados, e sendo o nosso tema relacionado com reconhecimento de imagem, esperava-mos que existisse muita informação. Como mencionado previamente neste relatório apenas serão analisados 2 métodos.

O single frame CNN foi o primeiro método estudado e analisado, ele consiste com base em algumas frames espalhadas pelo vídeo em análise, calcular a probabilidade de uma determinada atividade estar a ocorrer, fazendo posteriormente a análise a média de todas as probabilidades individuais, obtendo assim um vetor de probabilidades. No geral é um método que possui um bom desempenho e resultados promissores. Neste trabalho o CNN foi implementado com um dataset que possui vários vídeos demonstrando atividades humanas, nomeadamente as usadas: corridas de cavalos, andar de baloiço, passear um cão, tai chi.

No caso da arquitetura CNN com LSTMs, é feito o uso de camadas CNN para extrair recursos em dados de entrada combinados com LSTMs para oferecer suporte á previsão de sequência. Isto difere do CNN pois ele não é recorrente, ou seja, não retém memória de padrões de séries temporais anteriores. Ele só consegue treinar com base em dados inseridos num determinado instante. Em suma, a arquitetura CNN com LSTMs foi desenvolvida para problemas de previsão visual de séries temporais com o objetivo de gerar descrições provenientes das sequências de imagens.

4. Descrição do Trabalho

Como início foi implementado o algoritmo de de classificação de vídeo single-frame CNN, desenvolvido em python através da biblioteca keras. De maneira a organizar o código e ter uma abordagem por blocos foi decidido que este ia ser desenvolvido via Jupiter. Como foi mencionado previamente foi necessário fazer uso de um dataset de vídeos de maneira a testar os vários métodos de reconhecimento de atividade humana. Para o tal, foi utilizado o Human activity Recognition using smatphones data set, pelo UCI Machine Learning repository. Posteriormente foi carregado este dataset para o Jupiter.

Após ter sido implementado o código foram selecionadas 4 classes: WalkingWithdog, TaiChi, Swing e HorseRace. Estas são as classes para quais o modelo será treinado, usando vídeos relacionados com o tópico de cada uma.

```
[4]: image_height, image_width = 64, 64
     max_images_per_class = 8000

     dataset_directory = "UCF50"
     classes_list = ["WalkingWithDog", "TaiChi", "Swing", "HorseRace"]

     model_output_size = len(classes_list)
```

Figura 1- modelos a ser treinados

Após extrair as classes foi necessário treinar o modelo. Para o treino dos modelos, foi necessário definir um número de epochs, ou épocas de treino. Quantas mais épocas escolhermos mais accurate o modelo vai ser. Para este projeto foram selecionadas 50 épocas pois aparentam ser suficientes para obter resultados satisfatórios e não levam muito tempo a serem processadas.

```
[12]:
# Adding the Early Stopping Callback to the model which will continuously monitor the validation loss metric for every epoch.
# If the models validation loss does not decrease after 15 consecutive epochs, the training will be stopped and the weight which reported the lowest validation loss will be retored in t
early_stopping_callback = EarlyStopping(monitor = 'val_loss', patience = 15, mode = 'min', restore_best_weights = True)

# Adding loss, optimizer and metrics values to the model.
model.compile(loss = 'categorical_crossentropy', optimizer = 'Adam', metrics = ["accuracy"])

# Start Training
model_training_history = model.fit(x = features_train, y = labels_train, epochs = 50, batch_size = 4 , shuffle = True, validation_split = 0.2, callbacks = [early_stopping_callback])
```

Figura 2- treinar os modelos

Após o treinos dos nossos modelos o algoritmo procede a traçar os gráficos de resultados dos mesmos.
As métricas que tivemos em conta foram a accuracy e as perdas.

```
[15]: def plot_metric(metric_name_1, metric_name_2, plot_name):
        # Get Metric values using metric names as identifiers
        metric_value_1 = model_training_history.history[metric_name_1]
        metric_value_2 = model_training_history.history[metric_name_2]

        # Constructing a range object which will be used as time
        epochs = range(len(metric_value_1))

        # Plotting the Graph
        plt.plot(epochs, metric_value_1, 'blue', label = metric_name_1)
        plt.plot(epochs, metric_value_2, 'red', label = metric_name_2)

        # Adding title to the plot
        plt.title(str(plot_name))

        # Adding legend to the plot
        plt.legend()

[19]: plot_metric('loss', 'val_loss', 'Total Loss vs Total Validation Loss')
```

Figura 3- traçar os gráficos

Após termos traçado as métricas do algoritmo para gráficos temos então os modelos prontos para fazer previsões em tempo real com vídeos de teste. Para isto foi feito o download de vídeos da plataforma Youtube.

```
# Creating The Output directories if it does not exist
output_directory = 'Youtube_Videos'
os.makedirs(output_directory, exist_ok = True)

# Downloading a YouTube Video
video_title = download_youtube_videos('https://www.youtube.com/watch?v=8u0qjmHIOcE', output_directory)

# Getting the YouTube Video's path you just downloaded
input_video_file_path = f'{output_directory}/{video_title}.mp4'
```

Figura 4- download vídeo youtube

Posteriormente, após ter o video guardado localmente e preparado para ter as labels colocadas, prosseguiu-se á previsão do mesmo utilizando a Moving average e sem Moving average. Se a window_size for colocada com o valor 1, a função comporta-se como um classificador normal para prever frames de vídeos.

```
[22]:
# Setting sthe Widow Size which will be used by the Rolling Averge Proces
window_size = 1

# Construting The Output YouTube Video Path
output_video_file_path = f'{output_directory}/{video_title} -Output-WSize {window_size}.mp4'

# Calling the predict_on_live_video method to start the Prediction.
predict_on_live_video(input_video_file_path, output_video_file_path, window_size)

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

t:   0%|          | 0/866 [00:00<?, ?it/s, now=None]
Moviepy - Building video __temp__.mp4.
Moviepy - Writing video __temp__.mp4


Moviepy - Done !
Moviepy - video ready __temp__.mp4
```

Figura 5- previsão sem moving average

Utilizando Moving Average:

```
[23]: # Setting the Widow Size which will be used by the Rolling Averge Process
window_size = 25

# Constructing The Output YouTube Video Path
output_video_file_path = f'{output_directory}/{video_title} -Output-WSize {window_size}.mp4'

# Calling the predict_on_live_video method to start the Prediction and Rolling Averge Process
predict_on_live_video(input_video_file_path, output_video_file_path, window_size)

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

```
t:   0%|          | 0/866 [00:00<?, ?it/s, now=None]
Moviepy - Building video __temp__.mp4.
Moviepy - Writing video __temp__.mp4


Moviepy - Done !
Moviepy - video ready __temp__.mp4
```

Figura 6- previsão com moving average

Apesar de os resultados não serem perfeitos pode-se observar que são bastante melhores que o método que não usa moving average.

De seguida foi utilizado então o método single-frame CNN. A função dá um output que faz uma unica previsão para o vídeo todo, ou seja, pega em N frames do vídeo inteiro e faz previsões. Posteriormente o algoritmo faz a média das previsões dessas N frames e dá-nos a previsão final da atividade desse vídeo.

```
[25]:
# Downloading The YouTube Video
video_title = download_youtube_videos('https://www.youtube.com/watch?v=ceRjxW4MpOY', output_directory)

# Constructing 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: TaiChi   AVERAGED PROBABILITY: 1e+02
CLASS NAME: HorseRace   AVERAGED PROBABILITY: 0.0018
CLASS NAME: WalkingWithDog   AVERAGED PROBABILITY: 2.5e-05
CLASS NAME: Swing   AVERAGED PROBABILITY: 1.5e-05
chunk:   0%|          | 0/1215 [00:00<?, ?it/s, now=None]
Moviepy - Building video __temp__.mp4.
MoviePy - Writing audio in __temp__TEMP_MPY_wvf_snd.mp3
t:   0%|          | 0/1322 [00:00<?, ?it/s, now=None]
MoviePy - Done.
Moviepy - Writing video __temp__.mp4


Moviepy - Done !
Moviepy - video ready __temp__.mp4
```

Figura 7- previsão CNN

CNN com LSTMs

Após desenvolvermos o single frame CNN procedeu-se á implementação de um CNN LSTM network model, esta arquitetura envolve usar camadas CNN para a extração de dados combinados com LSTMs para dar suporte á previsão de sequência.

CNN LSTMs foram desenvolvidos para problemas de previsão de séries temporais e a aplicação de geração de descrições textuais a partir de sequência de imagens.

O modelo CNN LSTM lerá subsequências da sequência principal em blocos, extrairá recursos de cada bloco e, em seguida, permitirá que o LSTM interprete os recursos extraídos de cada bloco.

Uma abordagem para implementar este modelo é dividir cada janela de 128 etapas de tempo em subsequências para o modelo CNN processar. Por exemplo, as 128 etapas de tempo em cada janela podem ser divididas em quatro subsequências de 32 etapas de tempo.

```
# reshape data into time steps of sub-sequences
n_steps, n_length = 4, 32
trainX = trainX.reshape((trainX.shape[0], n_steps, n_length, n_features))
testX = testX.reshape((testX.shape[0], n_steps, n_length, n_features))
```

Figura 8-data subsequencias

Podemos então definir um modelo CNN que espera ler em sequências com um comprimento de 32 intervalos de tempo e nove recursos.

Todo o modelo CNN pode ser empacotado numa camada TimeDistributed para permitir que o mesmo modelo CNN leia em cada uma das quatro subsequências na janela. Os recursos extraídos são então nivelados e fornecidos ao modelo LSTM para leitura, extraindo os seus próprios recursos antes que um mapeamento final para uma atividade seja feito.

```
# define model
model = Sequential()
model.add(TimeDistributed(Conv1D(filters=64, kernel_size=3, activation='relu'), in
model.add(TimeDistributed(Conv1D(filters=64, kernel_size=3, activation='relu')))
model.add(TimeDistributed(Dropout(0.5)))
model.add(TimeDistributed(MaxPooling1D(pool_size=2)))
model.add(TimeDistributed(Flatten()))
model.add(LSTM(100))
model.add(Dropout(0.5))
model.add(Dense(100, activation='relu'))
model.add(Dense(n_outputs, activation='softmax'))
```

Figura 9- definir modelo

Uma extensão adicional da ideia do CNN com LSTM é realizar as convoluções do CNN como parte do LSTM. Essa combinação é chamada de LSTM convolucional ou, abreviadamente, ConvLSTM e, como o CNN LSTM, também é usado para dados espaço-temporais.

Ao contrário de um LSTM que lê os dados diretamente para calcular o estado interno e as transições de estado, e ao contrário do CNN LSTM que está interpreta a saída dos modelos CNN, o ConvLSTM usa convoluções diretamente como parte da leitura de entrada nas próprias unidades LSTM

A biblioteca Keras fornece a classe ConvLSTM2D que suporta o modelo ConvLSTM para dados 2D. Ele pode ser configurado para classificação de série temporal multivariada 1D.

```
# define model
model = Sequential()
model.add(ConvLSTM2D(filters=64, kernel_size=(1,3), activation='relu', input_shape=
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(n_outputs, activation='softmax'))
```

Figura 10- definir modelo conv2D

Podemos então avaliar os modelos como fizemos anteriormente.

5. Análise de Resultados

Começando pelo single-frame CNN, utilizando as classes referidas anteriormente podemos observar os seguintes resultados.

Épocas = 50

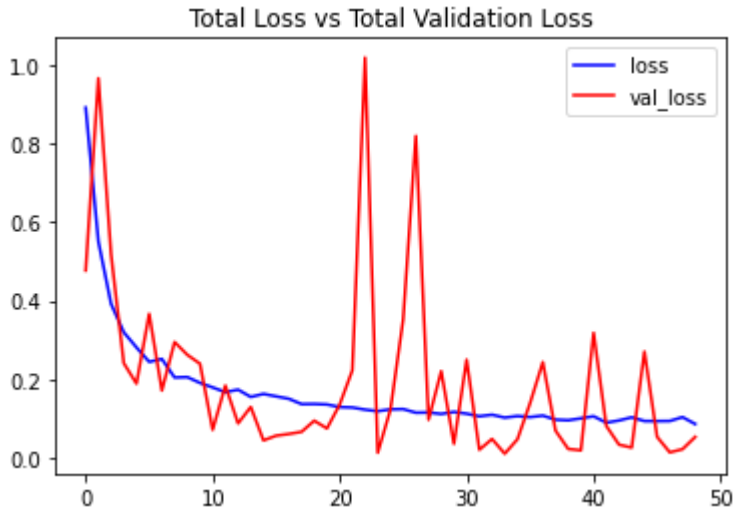


Figura 11 - Modelo de Perdas

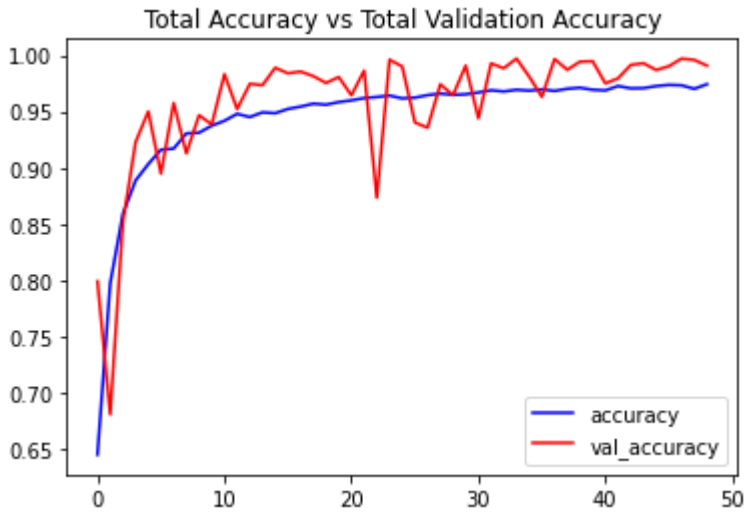


Figura 12- Modelo de Precisão

```
CLASS NAME: Swing   AVERAGED PROBABILITY: 9.4e+01
CLASS NAME: WalkingWithDog   AVERAGED PROBABILITY: 6.1
CLASS NAME: HorseRace   AVERAGED PROBABILITY: 0.043
CLASS NAME: TaiChi   AVERAGED PROBABILITY: 0.0016
chunk:   0%|          | 0/1214 [00:00<?, ?it/s, now=None]
Moviepy - Building video __temp__.mp4.
MoviePy - Writing audio in __temp__TEMP_MPY_wvf_snd.mp3
t:   0%|          | 0/1650 [00:00<?, ?it/s, now=None]
MoviePy - Done.
Moviepy - Writing video __temp__.mp4


Moviepy - Done !
Moviepy - video ready   temp   .mp4
```

Figura 13- resultados CNN

Como pode ser observado, ao utilizar um vídeo do Youtube de um baloiço como input, o algoritmo CNN, classifica com 94% de certeza, que se trata de facto de um baloiço representado no vídeo. Também é possível observar que ele deteta uma probabilidade de 6% de o vídeo se tratar de passear um cão, mas como a probabilidade é muito inferior á anterior, é descartada. As outras atividades possuem um nivel de certidão semelhante que ronda os 95-96%, á exceção do Tai-chi que surpreendentemente atingiu os 100%.

Utilizando o CNN com LSTMs podemos observar os seguintes resultados:

Épocas = 50

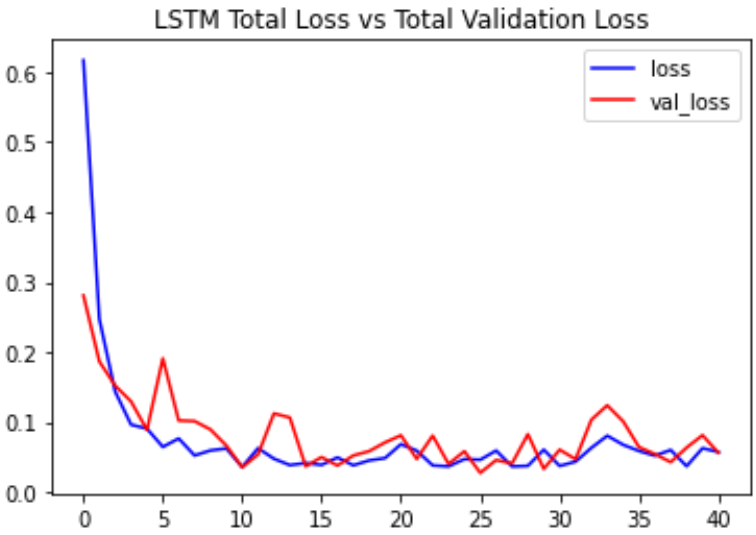


Figura 13- Modelo de perdas

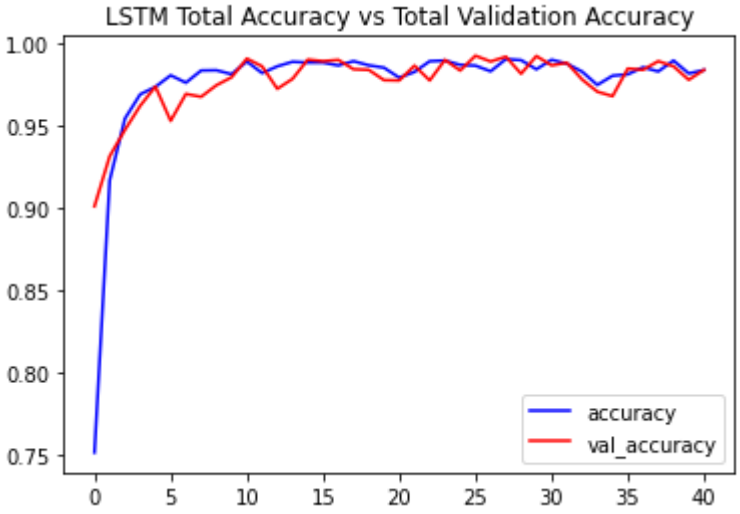
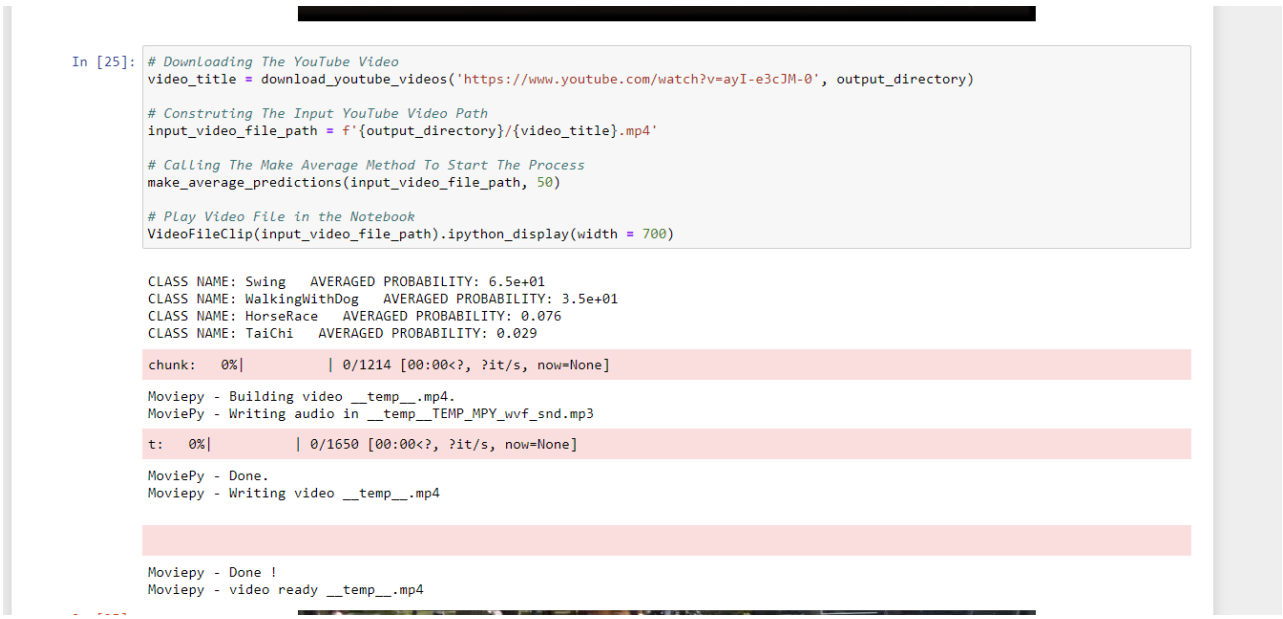


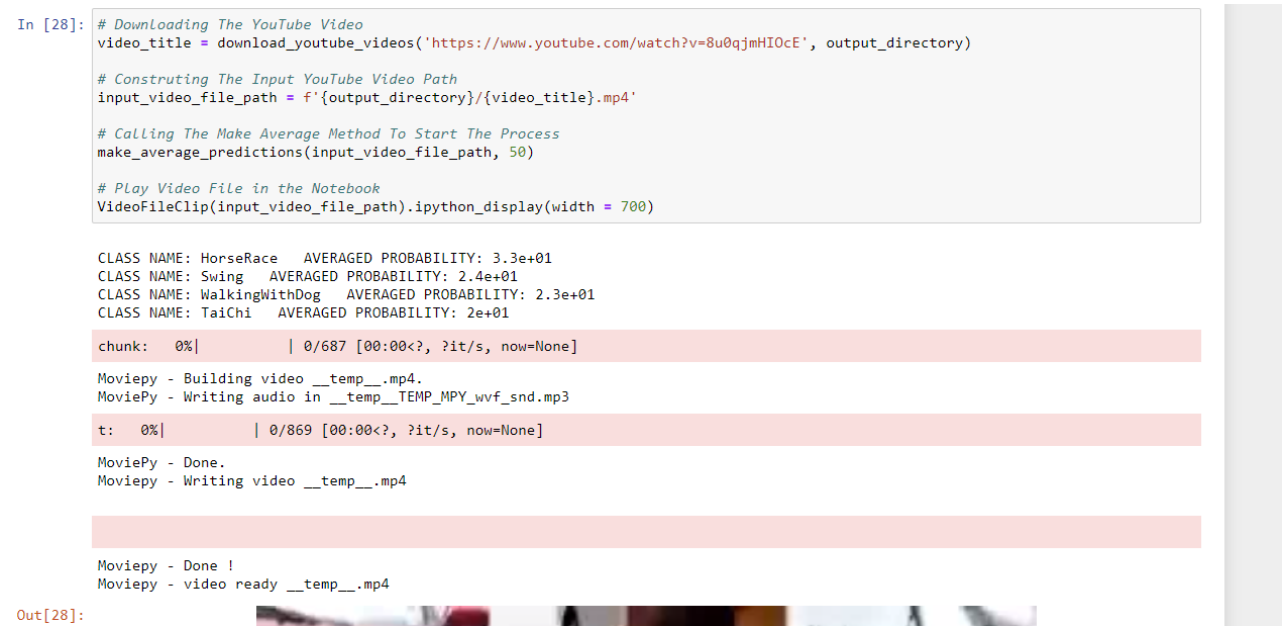
Figura 14-Modelo de precisão



```
In [25]: # Downloading The YouTube Video
video_title = download_youtube_videos('https://www.youtube.com/watch?v=ayI-e3cJM-0', 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: Swing   AVERAGED PROBABILITY: 6.5e+01
CLASS NAME: WalkingWithDog   AVERAGED PROBABILITY: 3.5e+01
CLASS NAME: HorseRace   AVERAGED PROBABILITY: 0.076
CLASS NAME: TaiChi   AVERAGED PROBABILITY: 0.029

chunk:   0%|          | 0/1214 [00:00<?, ?it/s, now=None]

Moviepy - Building video __temp__.mp4.
MoviePy - Writing audio in __temp__TEMP_MPY_wvf_snd.mp3

t:   0%|          | 0/1650 [00:00<?, ?it/s, now=None]

MoviePy - Done.
Moviepy - Writing video __temp__.mp4


Moviepy - Done !
Moviepy - video ready __temp__.mp4
```

Figura 15- resultados CNN com LSTMs video 1 atividade

Como podemos observar neste exemplo de vídeo que consiste apenas de uma atividade o método de CNN com LSTMs obteve um resultado significativamente pior comparativamente ao CNN single-frame, apenas apontando uma probabilidade de 65% de estar correto.



```
In [28]: # Downloading The YouTube Video
video_title = download_youtube_videos('https://www.youtube.com/watch?v=8u0qjmHIOcE', 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: HorseRace   AVERAGED PROBABILITY: 3.3e+01
CLASS NAME: Swing   AVERAGED PROBABILITY: 2.4e+01
CLASS NAME: WalkingWithDog   AVERAGED PROBABILITY: 2.3e+01
CLASS NAME: TaiChi   AVERAGED PROBABILITY: 2e+01

chunk:   0%|          | 0/687 [00:00<?, ?it/s, now=None]

Moviepy - Building video __temp__.mp4.
MoviePy - Writing audio in __temp__TEMP_MPY_wvf_snd.mp3

t:   0%|          | 0/869 [00:00<?, ?it/s, now=None]

MoviePy - Done.
Moviepy - Writing video __temp__.mp4


Moviepy - Done !
Moviepy - video ready __temp__.mp4
Out[28]:
```

Figura 16-CNN com LSTms video com vária atividades

Como podemos observar na figura quando se trata de um vídeo que contem várias atividades o CNN com LSTMs possui uma probabilidade de certeza bastante superior á do CNN single frame. Como o CNN single frame basea-se em médias de predições de todas as frames de um vídeo, se este for variado e caótico, vai encontrar problemas em ser certeiro e determinar a atividade principal, entretanto o CNN com LSMTs não retém memória de padrões de séries temporais anteriores e funciona com base em dados inseridos num determinado instante, o que é bastante mais adequado para esse tipo de vídeos.

Em exemplos que possuem apenas uma atividade o CNN single frame demonstra uma performance significativamente melhor, que ronda os 95%, em comparação com o CNN LSMTs que varia entre os 60% e os 80%.

6. Conclusões e Perspetivas de Desenvolvimento

Tendo concluído o trabalho e após uma análise dos dados, podemos concluir que o CNN LSTM atinge na maioria dos casos uma taxa de acerto mais elevada em relação ao CNN single-frame, pelo menos no que toca a vídeos mais próximos da realidade em que se podem observar várias atividades no mesmo vídeo. Seria possível reduzir as taxas de erros e acertos na classificação do CNN LSTM em relação ao CNN single-frame com o acerto do algoritmo e com treino adicional.

No geral, em média observou-se que o CNN com LSTMs obteve um melhoramento de cerca de 6% em relação ao single-frame CNN. Futuramente seria relevante aprimorar o algoritmo do CNN com LSTMs de maneira a obter melhores resultados, assim como a exploração de diferentes algoritmos de reconhecimento de imagens.

7. Referências

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

https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones.

https://machinelearningmastery.com/how-to-develop-rnn-models-for-human-activity-recognition-time-series-classification/.