Universidade da Beira Interior
Departamento de Informática

Nº Trabalho Prático: *Human Recognition in Surveillance Settings*

Elaborado por:

João Miguel Baltazar Martins

Orientador:

Professor Doutor Hugo Pedro Proença

10 de junho de 2024

Acrónimos

CNN Convolutional Neural Network

MSE *Mean Squared Error*

MAE *Mean Absolute Error*

GAN *Generative Adversarial Network*

CNN *Convolutional Neural Network*

MAPE *Mean Absolute Percentage Error*

ROI *Region of Interest*

Conteúdo

Lista de Tabelas

Lista de Figuras

Lista de Excertos de Código

Capítulo 1

Introdução

1.1 Enquadramento

O presente relatório aborda o trabalho prático da unidade curricular de *Computer Vision*, que se concentra em desenvolver uma solução para dada uma imagem da cara de “má qualidade” de uma pessoa obter a correspondente imagem de "boa qualidade".

1.2 Objetivo

O objetivo principal deste trabalho é, dada uma imagem de "má qualidade", obter a correspondente imagem de "boa qualidade". Este objetivo principal pode ser subdividido em várias tarefas específicas, conforme listado abaixo:

1. Pré-Processamento das Imagens:
    a) Desenvolver uma função para padronizar as proporções das imagens, tornando-as quadradas, com uma banda preta de ambos os lados.
    b) Obter a estimativa da pose da cabeça (*Head Pose Estimation*.
    c) Realizar a segmentação facial (*Face Segmentation*).
2. Treinar o modelo *Pix2Pix* no *dataset* deste projeto.
3. Utilizar um modelo *Pix2Pix* pré-treinado.
4. Análise critica dos resultados provenientes destes.

1.3 Organização do Documento

1. O primeiro capítulo – **Introdução** – apresenta o projeto, o enquadramento para o mesmo, o objetivo do projeto e a respetiva organização do documento.

2. O segundo capítulo – **Desenvolvimento** – será detalhada a constituição do conjunto de dados deste projeto, o modo de divisão do mesmo, a implementação do *Pix2Pix* do zero e a utilização de um modelo *Pix2Pix* pré-treinado.

3. O terceiro capítulo – **Exposição e Análise dos Resultados** – onde são expostos os resultados do modelo *Pix2Pix* treinado sem segmentação e do modelo *Pix2Pix* com segmentação, acompanhada de uma reflexão crítica.

4. O quarto capítulo – **Conclusão** – refere as conclusões principais a tirar deste trabalho e o que poderá ainda ser feito a respeito do mesmo.

Capítulo 2

Desenvolvimento

2.1 Introdução

Neste capítulo, será detalhada a estrutura do *dataset* deste projeto e a implementação do modelo *Pix2Pix*.

2.2 Estrutura do *Dataset*

Este estudo é composto por vídeos no formato *.mp4* de 159 utilizadores. Para cada utilizador, existem quatro vídeos, sendo dois de boa qualidade e dois de má qualidade, gravados em dias diferentes. Cada vídeo tem a duração de trinta segundos, totalizando um minuto de vídeo por utilizador.

Os vídeos têm uma taxa de dez *frames* por segundo, em formato horizontal (*landscape*). Nos vídeos de boa qualidade, o utilizador está a olhar diretamente para a câmara numa pose descontraída. Nos vídeos de má qualidade, foram introduzidas variações em termos de distância, ângulo, perspetiva, ambiente, luminosidade e oclusões.

Cada vídeo encontra-se convertido para *frames* e estas estão armazenadas na respetiva diretoria. As designações para cada diretoria são da seguinte forma:

A pasta de dados deste projeto está armazenada numa pasta chamada `face_square`. Esta pasta contém múltiplas sub-pastas, cada uma representando um vídeo específico. A convenção de nomenclatura para estas subpastas é a seguinte:

```
<ID_SUJEITO>_<TIPO_VIDEO>_<INDICE_VIDEO>
```

Onde:

- `ID_SUJEITO` é o ID ou número de cidadão do voluntário (o sujeito no vídeo).
- `TIPO_VIDEO` indica o tipo de vídeo. "E"representa enrolamento, o que significa que o vídeo é de boa qualidade. "U"representa não-constrangido, o que significa que o vídeo é de má qualidade.
- `INDICE_VIDEO` indica a sessão de registos. Pode ser Sessão 1 ou Sessão 2.

Por exemplo, uma pasta chamada `001_E_1` conteria o primeiro vídeo de boa qualidade do voluntário "001", e uma pasta chamada `002_U_2` conteria o segundo vídeo de má qualidade do voluntário "002".

Cada uma destas sub-pastas contém um conjunto de *frames*, representados como imagens *.jpg*, correspondentes ao utilizador, à qualidade e à respetiva sessão.

É importante realçar que foi necessário utilizar o *YOLO* para definir as *Region of Interest* (ROI) tanto da cara quanto do corpo.

2.3 Divisão dos Dados

O conjunto de dados, constituído por 159 utilizadores, foi dividido da seguinte forma:

- Conjunto de Treino: 128 Utilizadores (cerca de 80% dos dados)
- Conjunto de Validação: 16 Utilizadores (cerca de 10% dos dados)
- Conjunto de Validação: 15 Utilizadores (cerca de 10% dos dados)

Sabendo que, um utilizador que pertence já a um conjunto, não poderá pertencer a outro.

2.4 Pré-Processamento das *Frames*

Colocar Frame *Quadrada*

A pré-processamento das imagens para serem quadradas com bandas pretas *padding* é importante principalmente para garantir a uniformidade de entrada e preservar o aspecto original da imagem, evitando distorções que podem afetar o desempenho do modelo.

Nas figuras 2.1, 2.2, está ilustrado o resultado deste processo.

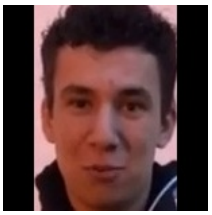

Figura 2.1: Imagem de boa qualidade quadrada

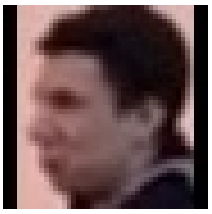

Figura 2.2: Imagem de má qualidade quadrada

Face Segmentation

A segmentação facial permite que o modelo se concentre na parte mais relevante da imagem, o rosto, ignorando o restante da imagem que pode conter informações irrelevantes ou enganosas. Além disso, ao segmentar o rosto, reduz-se a quantidade de dados que o modelo precisa processar, tornando o treino do modelo mais rápido e eficiente.

A segmentação facial foi realizada usando um modelo pré-treinado do *PyTorch* chamado *FCN-ResNet50*, que é uma *Convolutional Neural Network* (CNN).

Para cada imagem, são aplicas as transformações, passa a imagem pelo modelo e obtém a saída do modelo, que é uma máscara indicando a localização do rosto na imagem. Esta máscara é então aplicada à imagem original para criar uma nova imagem onde apenas o rosto é visível e o restante da imagem é preenchido com verde.

Nas figuras 2.3, 2.4, está ilustrado o resultado da segmentação à imagem de boa qualidade e de má qualidade, respetivamente.

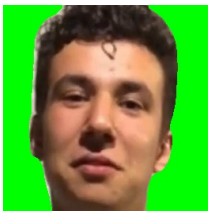

Figura 2.3: Imagem de boa qualidade quadrada e segmentada

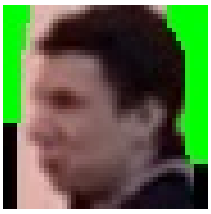

Figura 2.4: Imagem de má qualidade quadrada e segmentada

Head Pose Estimation

A estimativa de pose da cabeça é uma técnica de visão computacional que determina a orientação da cabeça de uma pessoa em uma imagem ou vídeo. Ela fornece a rotação da cabeça em torno de três eixos: inclinação (movimento para cima e para baixo), guinada (movimento para a esquerda e para a direita) e rolagem (inclinação da cabeça para a esquerda ou para a direita).

Os valores representam a rotação da cabeça em torno dos eixos x, y e z, respectivamente. Aqui está uma análise dos resultados:

1. **Imagem da figura (boa qualidade) 2.1 (2.079658, -7.495624, 1.346217)**: Esses valores sugerem que a cabeça está ligeiramente inclinada para a direita (rotação positiva em torno do eixo x), ligeiramente virada para a direita (rotação negativa em torno do eixo y) e ligeiramente inclinada para a direita (rotação positiva em torno do eixo z).

2. **Imagem da figura (má qualidade) 2.2 (65.150192, -11.407316, -4.553188)**: Esses valores sugerem que a cabeça está significativamente inclinada para a direita (rotação positiva em torno do eixo x), ligeiramente virada para a direita (rotação negativa em torno do eixo y) e ligeiramente inclinada para a esquerda (rotação negativa em torno do eixo z).

A diferença significativa na rotação em torno do eixo x entre os dois vídeos pode ser devido à diferença na qualidade dos vídeos. Em um vídeo de má

qualidade, pode ser mais difícil para o algoritmo detectar com precisão os pontos de referência faciais, o que pode levar a uma estimativa de pose menos precisa.

Além disso, a iluminação, a resolução e outros fatores também podem afetar a precisão da estimativa de pose da cabeça.

2.5 Modelo *Pix2Pix* do Zero

O modelo *Pix2Pix* é um tipo de *Generative Adversarial Network* (GAN) cujo objetivo é fazer o mapeamento de uma imagem de entrada para uma imagem de saída.

Este consiste em duas componentes principais: um gerador e um discriminador. O gerador tenta criar imagens que pareçam reais, enquanto o discriminador tenta distinguir entre imagens reais e falsas. O gerador e o discriminador são treinados juntos, com o gerador a tentar enganar o discriminador e o discriminador a tentar classificar corretamente as imagens como reais ou falsas.

No caso deste projeto, as imagens de entrada são as *frames* dos vídeos de má qualidade, e as imagens de saída são as *frames* dos vídeos de de boa qualidade. O objetivo é treinar o modelo para gerar imagens de boa qualidade a partir de imagens de má qualidade.

O modelo é definido na função `create_model`. Esta função constrói uma *U-Net Architecture* que utiliza uma série de camadas de amostragem *encoder* para reduzir as dimensões espaciais da imagem de entrada enquanto aumenta a profundidade (número de canais). Isto é seguido por uma série de camadas de amostragem *decoder* que aumentam as dimensões espaciais enquanto reduzem a profundidade. As camadas de amostragem *encoder* e *decoder* são ligadas por conexões de atalho, que ajudam a preservar a informação espacial.

Este é treinado usando o otimizador *Adam* e a função de *loss* utilizado foi o *Mean Squared Error* (MSE). Esta mede a diferença quadrática média entre os valores previstos e os valores reais, o que a torna adequada para tarefas de regressão como esta.

O modelo é treinado num conjunto de treino de imagens, e o seu desempenho é avaliado num conjunto de validação de imagens durante o treino. O processo de treino continua até que o desempenho no conjunto de validação pare de melhorar, conforme determinado pela *callback* do *early stopping*.

Após o treino, o desempenho do modelo é avaliado num conjunto de teste de imagens. O modelo é então guardado num ficheiro para uso posterior.

Arquitetura do Modelo

Camada (tipo)	**Forma de Saída**	**Param #**
input_1 (InputLayer)	(None, 256, 256, 3)	0
sequential (Sequential)	(None, 128, 128, 64)	3072
sequential_1 (Sequential)	(None, 64, 64, 128)	131584
sequential_2 (Sequential)	(None, 32, 32, 256)	525312
sequential_3 (Sequential)	(None, 16, 16, 512)	2099200
sequential_4 (Sequential)	(None, 8, 8, 512)	4196352
sequential_5 (Sequential)	(None, 4, 4, 512)	4196352
sequential_6 (Sequential)	(None, 2, 2, 512)	4196352
sequential_7 (Sequential)	(None, 1, 1, 512)	4196352
sequential_8 (Sequential)	(None, 2, 2, 512)	4196352
concatenate (Concatenate)	(None, 2, 2, 1024)	0
sequential_9 (Sequential)	(None, 4, 4, 512)	8390656
concatenate_1 (Concatenate)	(None, 4, 4, 1024)	0
sequential_10 (Sequential)	(None, 8, 8, 512)	8390656
concatenate_2 (Concatenate)	(None, 8, 8, 1024)	0
sequential_11 (Sequential)	(None, 16, 16, 512)	8390656
concatenate_3 (Concatenate)	(None, 16, 16, 1024)	0
sequential_12 (Sequential)	(None, 32, 32, 256)	4195328
concatenate_4 (Concatenate)	(None, 32, 32, 512)	0
sequential_13 (Sequential)	(None, 64, 64, 128)	1049088
concatenate_5 (Concatenate)	(None, 64, 64, 256)	0
sequential_14 (Sequential)	(None, 128, 128, 64)	262400
concatenate_6 (Concatenate)	(None, 128, 128, 128)	0
conv2d_transpose_7 (Conv2DTranspose)	(None, 256, 256, 3)	6147

Tabela 2.1: Sumário das Camadas do Modelo

- **Parâmetros totais:** 54,425,859 (207.62 MB)
- **Parâmetros treináveis:** 54,414,979 (207.58 MB)
- **Parâmetros não treináveis:** 10,880 (42.50 KB)

Hiperparâmetros no Treino do Modelo *Pix2Pix*

1. **BATCH_SIZE**: Este valor está definido para 32. O tamanho do lote é o número de exemplos de treino usados numa iteração. A escolha do tamanho do lote pode afetar a velocidade e a estabilidade do processo de aprendizagem. Um tamanho de lote menor significa que o modelo

será atualizado mais frequentemente, o que pode tornar o processo de aprendizagem mais rápido, mas também mais instável. Um tamanho de lote maior significa que o modelo será atualizado com menos frequência, o que pode tornar o processo de aprendizagem mais lento, mas também mais estável. A escolha de 32 é uma escolha comum para o tamanho do lote e é muitas vezes um bom equilíbrio entre velocidade e estabilidade.

2. **Paciência no *EarlyStopping***: Este valor está definido para 2. Este é o número de épocas sem melhoria após o qual o treino será interrompido. A escolha da paciência pode afetar a duração do treino do modelo. Uma paciência menor significa que o modelo irá parar o treino mais cedo, o que pode prevenir o *overfitting*, mas também pode resultar em *underfitting*. Uma paciência maior significa que o modelo continuará a treinar por mais tempo, o que pode levar a um melhor desempenho, mas também pode aumentar o risco de *overfitting*. A escolha de 2 é uma escolha comum para a paciência e é muitas vezes um bom equilíbrio entre prevenir o *overfitting* e o risco de *underfitting*.

3. **Número de Épocas**: Foi escolhido um valor de 4 épocas. No entanto, foi implementada uma paragem precoce com uma paciência de 2. Isto significa que, se o desempenho do modelo no conjunto de validação não melhorar durante 2 épocas consecutivas, o treino será interrompido, mesmo que não tenha atingido as 5 épocas. Estava previsto 4 mas teve que ser reduzido (custo computacional).

4. **O número de filtros e o tamanho dos filtros nas funções downsample e upsample**: Estes valores estão definidos para 64, 128, 256 e 512 para o número de filtros e 4 para o tamanho dos filtros. O número de filtros é o número de mapas de características na camada convolucional, e o tamanho dos filtros é a altura e a largura dos filtros convolucionais. A escolha destes hiperparâmetros afeta a complexidade do modelo e a quantidade de informação que o modelo pode aprender. Um maior número de filtros ou um maior tamanho de filtros significa que o modelo pode aprender características mais complexas, mas também aumenta o risco de *overfitting* e o custo computacional. A escolha destes valores baseia-se na arquitetura do modelo *Pix2Pix*.

5. **O otimizador**: Este valor está definido para `adam`. O otimizador é o algoritmo usado para atualizar os pesos do modelo. A escolha do otimizador pode afetar a velocidade e a estabilidade do processo de aprendizagem.

Adam é uma escolha popular de otimizador porque combina as vantagens de dois outros métodos, *AdaGrad* e *RMSProp*, e funciona bem na prática numa ampla gama de problemas.

6. **A função de *loss***: Este valor está definido para `mean_square_error`. A função de perda é a função que o modelo tenta minimizar durante o treino. A escolha da função de perda depende do problema. Para problemas de regressão como este, o erro quadrático médio é uma escolha comum porque penaliza erros grandes mais do que erros pequenos.

2.6 *Fine-Tunning* de Modelo *Pix2Pix* Pré-Treinado

O modelo pré-treinado encontrado, foi treinado num conjunto de dados de rostos de celebridades. O gerador foi treinado por 150 épocas, após as quais foi capaz de gerar rostos a partir de puro ruído.

Pretende-se afinar este modelo pré-treinado no próprio conjunto de dados deste projeto. Afinar é um processo que pega num modelo já treinado num conjunto de dados e o treina (ou "ajusta") num conjunto de dados diferente. Isto pode ser benéfico quando o novo conjunto de dados é mais pequeno e se deseja aproveitar as características aprendidas pelo modelo pré-treinado, em vez de treinar um novo modelo do zero. Isto trás também a vantagem de computacionalmente ser menos moroso.

Aqui está como foi irá ser implementa a afinação:

1. Carrega o modelo *Pix2Pix* pré-treinado.

2. Compila o modelo com um otimizador, função de perda e métricas.

3. Define uma *callback* de *checkpoint* para guardar os pesos do modelo a cada 150 *batches*.

4. Afina o modelo nos dados de treino, validando nos dados de validação e usando os *callbacks* de *early stopping* e *checkpoint*. O *callback* de *early stopping* interromperá o processo de treino se a perda de validação não melhorar por um certo número de épocas (neste caso, 3), para evitar *overfitting*.

5. Finalmente, guarda o modelo afinado.

2.7 Conclusão

Nesta secção foram apresentadas as técnicas associados ao pré-processamento das *frames* de forma a potenciar o desempenho do modelo, foi também apresentado o modelo que irá fazer cumprir o nosso objetivo, na componente em que este é treinado pelo *dataset* deste projeto e na componente já pré-treinado num *dataset* de expressões faciais de famosos. Na próxima secção serão apresentados os resultados do modelo *Pix2Pix* treinado do zero em imagens sem segmentação e do modelo *Pix2Pix* treinado do zero em imagens com segmentação junto com a respetiva reflexão critica.

Capítulo 3

Exposição e Análise de Resultados

3.1 Introdução

O foco principal deste capitulo é fornecer *insights* claros sobre o desempenho do modelo *Pix2Pix* treinado do zero em imagens sem segmentação e do modelo *Pix2Pix* treinado do zero em imagens com segmentação, assim como fazer uma comparação e uma análise critica do desempenho de cada um.

3.2 Resultados do Modelo Treinado do Zero

Resultados Sem *Face Segmentation*

Os resultados obtidos para esta sub-secção foram através da utilização de cerca de vinte cinco por cento do *dataset* disponível. Foram utilizados os utilizadores desde o trinta e seis ao setenta devido custo computacional que acarretava treinar este modelo no *dataset* completo

Foi treinado e testado o modelo nos dados em que as caras não estavam segmentadas e os resultados, para **uma época**, com um **batch size de 32** foram os seguintes:

- **Detected User IDs:** 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70
- **Train Users:** 40, 56, 48, 57, 58, 47, 51, 37, 50, 66, 45, 54, 64, 70, 67, 63, 59, 68, 49, 38, 46, 39, 52, 43, 36, 60, 62, 44
- **Validation Users:** 42, 53, 61
- **Test Users:** 65, 69, 41, 55

```
Treino
532/532 [==============================] - ETA: 0s - loss: 0.0457 -

mean_absolute_error: 0.1576 -
mean_absolute_percentage_error: 20707698.0000
------------------------------------------------------------------------
Teste
76/76 [==============================] - 1086s 14s/step - loss: 0.0492 -
mean_absolute_error: 0.1604 -
mean_absolute_percentage_error: 12222765.0000
```

Análise aos Resultados

Treino

1. **Perda (Loss)**: A perda de treino é 0.0457. Isto é um bom sinal, pois indica que o modelo está a aprender e a melhorar a sua capacidade de prever as variáveis alvo.

2. ***Mean Absolute Error* (MAE)**: O MAE é 0.1576. Isto significa que, em média, as previsões do seu modelo estão aproximadamente 0.1576 unidades afastadas dos valores reais.

3. ***Mean Absolute Percentage Error* (MAPE)**: O MAPE é 20707698.0. Este é um valor muito elevado, o que sugere que as previsões do modelo estão muito distantes dos valores reais. No entanto, o MAPE pode ser distorcido por denominadores pequenos, portanto, se o seu conjunto de dados contiver valores reais próximos de zero, isso pode explicar o MAPE elevado.

No geral, o modelo parece estar a aprender, mas o MAPE elevado é uma preocupação.

Teste

1. **Perda (Loss)**: A perda de teste é 0.0492. Isto é ligeiramente superior à perda de treino de 0.0457. Isto pode indicar que o modelo está a sobreajustar aos dados de treino, o que significa que não está a generalizar bem para dados não vistos. No entanto, a diferença não é muito grande, por isso pode não ser um problema significativo.

2. **Erro Absoluto Médio (MAE)**: O MAE é 0.1604. Isto significa que, em média, as previsões do modelo estão aproximadamente 0.1604 unidades

afastadas dos valores reais no conjunto de teste. Isto é ligeiramente superior ao MAE de treino de 0.1576, o que novamente sugere um pouco de sobreajuste.

3. **Erro Absoluto Percentual Médio (MAPE)**: O MAPE é 12222765.0. Este é um valor muito elevado, o que sugere que as previsões do modelo estão muito afastadas dos valores reais. No entanto, como no caso do MAPE de treino, isto pode ser distorcido por denominadores pequenos se o seu conjunto de dados contiver valores reais próximos de zero.

Em resumo, o seu modelo parece estar ligeiramente a sobreajustar aos dados de treino, como indicado pela perda e MAE mais alto no conjunto de teste em comparação com o conjunto de treino. O MAE elevado é uma preocupação e leva a considerar que as previsões do modelo não são muito precisas.

Demonstração Visual

Para efeitos visuais, deu-se como *input* ao modelo uma *frame* de má qualidade de um utilizador contido no conjunto de teste, aquando o treino do modelo e o resultado pode ser observado na figura 3.1.

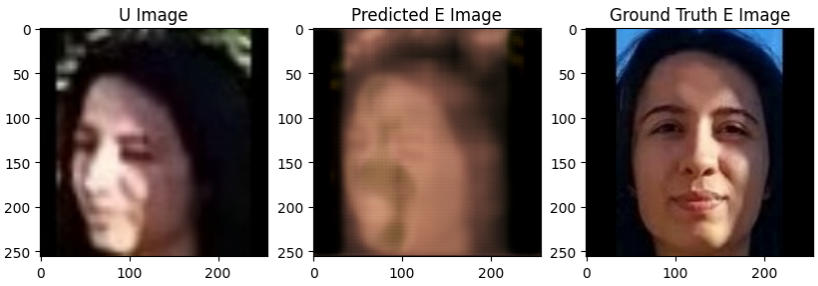

Figura 3.1: Teste visual do modelo treinado sem segmentação

Resultados Com *Face Segmentation*

Os resultados obtidos para esta sub-secção foram através da utilização de cerca de trinta e dois por cento do *dataset* disponível. Foram utilizados 50 utilizadores

Foi treinado e testado o modelo nos dados em que as caras estavam segmentadas e os resultados, para **uma época**, com um **batch size de 32** foram os seguintes:

- **Detected User IDs:** 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 64, 65, 66, 67, 68, 69
- **Train Users:** 45, 14, 18, 13, 36, 33, 19, 68, 43, 40, 39, 9, 17, 8, 25, 23, 24, 35, 3, 42, 30, 4, 67, 65, 21, 69, 11, 5, 38, 6, 1, 10, 29, 16, 44, 64, 31, 7, 66, 12
- **Validation Users:** 2, 27, 37, 32, 22
- **Test Users:** 41, 28, 34, 20, 26, 15

```
Treino / Validação
738/738 [==============================] - 28062s 38s/step - loss: 0.0375 -
mean_absolute_error: 0.1327 - mean_absolute_percentage_error: 7208888.0000
val_loss: 0.0601 - val_mean_absolute_error: 0.1777
val_mean_absolute_percentage_error: 7284660.5000
----------------------------------------------------------------------
Teste
111/111 [==============================] - 1677s 15s/step - loss: 0.0456 -
mean_absolute_error: 0.1457 - mean_absolute_percentage_error: 5648949.0000
```

Análise aos Resultados

Treino

- **Perda (*Loss*): 0.0375**: A perda do modelo é bastante baixa, o que sugere que está desempenhar-se dados de treino.
- **MAE: 0.1327**: O erro absoluto médio do modelo é relativamente baixo, o que sugere que está a fazer previsões bastante precisas.
- **MAPE: 7208888.0000**: este valor é bastante alto, o que sugere que as previsões do modelo podem estar erradas por uma grande percentagem. Isto pode ser devido a *outliers* nos dados ou a erros no modelo.
- **Perda Validação (val_loss): 0.0601**: Este é o valor da função de perda do seu modelo no conjunto de validação. O fato de ser mais alto do que a perda de treino sugere que o seu modelo pode estar a dar *overfitting* aos dados de treino.
- **Erro Absoluto Médio de Validação (val_mean_absolute_error): 0.1777**: É mais alto do que o erro absoluto médio de treino, o que novamente sugere que o seu modelo pode estar a *overfitting* aos dados de treino.

- **Erro Percentual Absoluto Médio de Validação (val_mean_absolute_percentage_error): 7284660.5000**: Este é o erro percentual absoluto médio para o conjunto de validação. Tal como o erro percentual absoluto médio de treino, este valor é bastante alto, o que sugere que as previsões do seu modelo podem estar erradas por uma grande percentagem no conjunto de validação.

No geral, o modelo parece estar a aprender, mas o MAPE elevado é uma preocupação.

Na figura 3.2 está ilustrada a evolução da *loss* durante a época 1, que é a única.

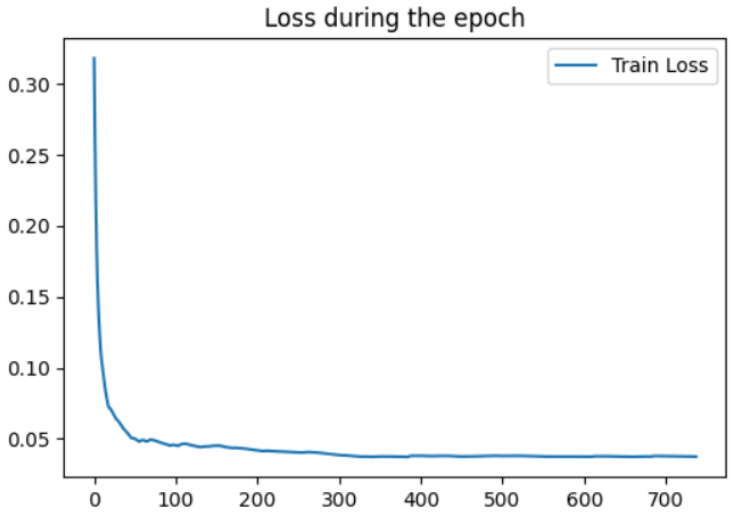

Figura 3.2: Evolução da *loss* durante a época

Teste

- **Perda (loss): 0.0601**
- **MAE: 0.1777**
- **MAPE: 7284660.5**

Podemos ver que o desempenho do modelo nos dados de teste é ligeiramente pior do que nos dados de treino. A perda, MAE e MAPE são todos mais altos nos dados de teste, o que sugere que o modelo pode estar a sobreajustar-se aos dados de treino.

Isto pode ser do facto de o modelo aprender os detalhes e o ruído nos dados de treino ao ponto de impactar negativamente o desempenho do modelo

em novos dados. Isto significa que o ruído ou as flutuações aleatórias nos dados de treino são captados e aprendidos como conceitos pelo modelo.

Pode-se considerar estratégias para melhorar a generalização do modelo, como regularização, *dropout*, ou treinar com maior percentagem do *dataset* de forma a diversificar o modelo.

Demonstração Visual

Para efeitos visuais, deu-se como *input* ao modelo uma *frame* de má qualidade de um utilizador contido no **conjunto de teste**, aquando o treino do modelo e o resultado pode ser observado na figura 3.3.

Figura 3.3: Teste visual do modelo treinado com segmentação

3.3 Comparação e Conclusão

Analisando os resultados do modelo treinado com segmentação e do modelo treinado sem segmentação, não é possível dizer com clareza qual se desempenhou melhor porque o com segmentação foi treinado num maior conjunto de dados, mas mesmo assim é de notar que os resultados são idênticos. Isto porque, o modelo só foi treinado numa época, nos dois casos, e num subconjunto do *dataset*.

É de assinalar que mesmo com estas condições, o modelo, nos dois casos, já começa a ter algumas noções.

Capítulo 4

Conclusão

4.1 Conclusões Principais

Neste documento foi então, apresentadas as técnicas de pré-processamento das *frame*s para otimizar o desempenho do modelo. Além disso, é introduzido o modelo que visa atingir o objetivo do projeto, sendo treinado com o *dataset* específico do projeto e também, um exemplo de modelo pré-treinado com um *dataset* de expressões faciais de famosos. De seguida foram apresentados os resultados do modelo *Pix2Pix* treinado do zero em imagens sem segmentação e do modelo *Pix2Pix* treinado do zero em imagens com segmentação, com uma análise crítica dos mesmos.

4.2 Próximos Passos

Os próximos passos deste projeto poderão ser:

- Evoluir o plano do *Google Colab* de forma a obter-se mais poder de processamento para que seja possível treinar o modelo com o *dataset* completo, por várias épocas, por várias vezes afinando os hiperpârametros.
- Incluir o *Head Pose Estimation* de forma a potenciar o desempenho do modelo.
- Afinar um modelo pré-treinado para o nosso *dataset*.