SISTEMAS INTELIGENTES

# Reconhecimento facial

GABRIELA CRISTINA SCHMITT
JENNIFER MAYARA DE PAIVA GOBERSKI
VINICIUS DIONIZIO PATROCINIO

PROFESSOR: ESCOBAR

# Etapas do projeto

01 Estratégia para a Implementação

Apresentação preliminar: 06/06/23 via portal

02 Preprocessamento e treinamento

Apresentação preliminar: 13/06/23 via portal

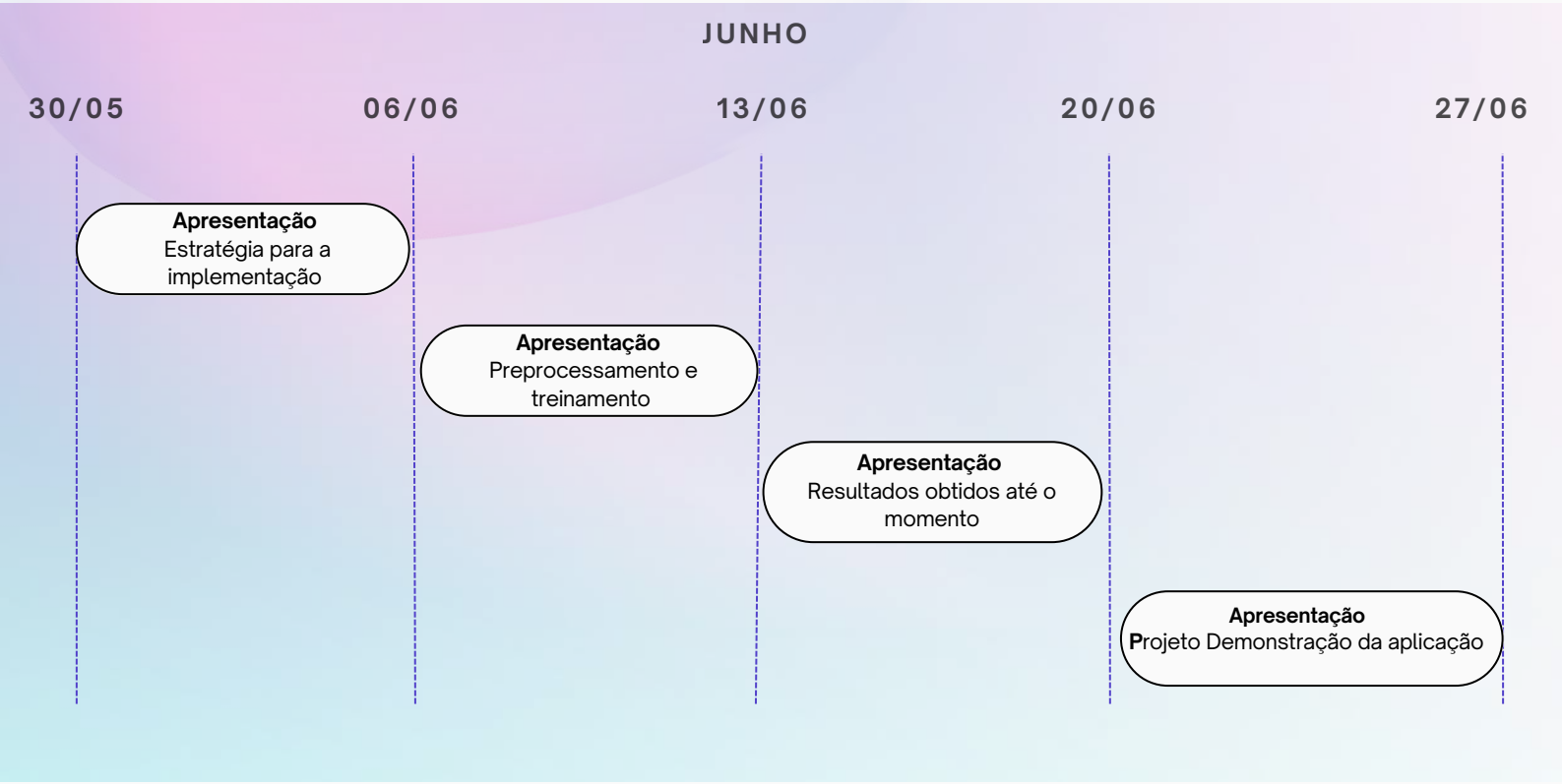
03 Resultados obtidos até o momento

Apresentação preliminar: 20/06/23 via portal

04 Demonstração da aplicação

Apresentação preliminar: 27/06/23 via portal

# Timeline


JUNHO
30/05
06/06
13/06
20/06
27/06
Apresentação
Estratégia para a implementação
Apresentação
Preprocessamento e treinamento
Apresentação
Resultados obtidos até o momento
Apresentação
Projeto Demonstração da aplicação

# Parte 01

***O objetivo dessa etapa é definirmos qual abordagem vamos utilizar no nosso projeto***

Após assistirmos os vídeos e lermos a literatura definida pelo professor, optamos por seguir a abordagem sugerida pelo canal Murtaza's Workshop - Robotics and AI no seu vídeo *Detect 468 Face Landmarks in Real-time | OpenCV Python | Computer Vision*, por ser uma solução que apresenta o maior número de pontos mapeados na sua face mask (468 face points), por mais que seu custo computacional seja maior. Nós seguimos o seu tutorial e atualizamos o mapeamento do atributo *mpFaceMesh.FACE_CONNECTIONS* da biblioteca *mediapipe* para *FACEMESH_CONTOURS*, uma vez que o primeiro estava depreciado.

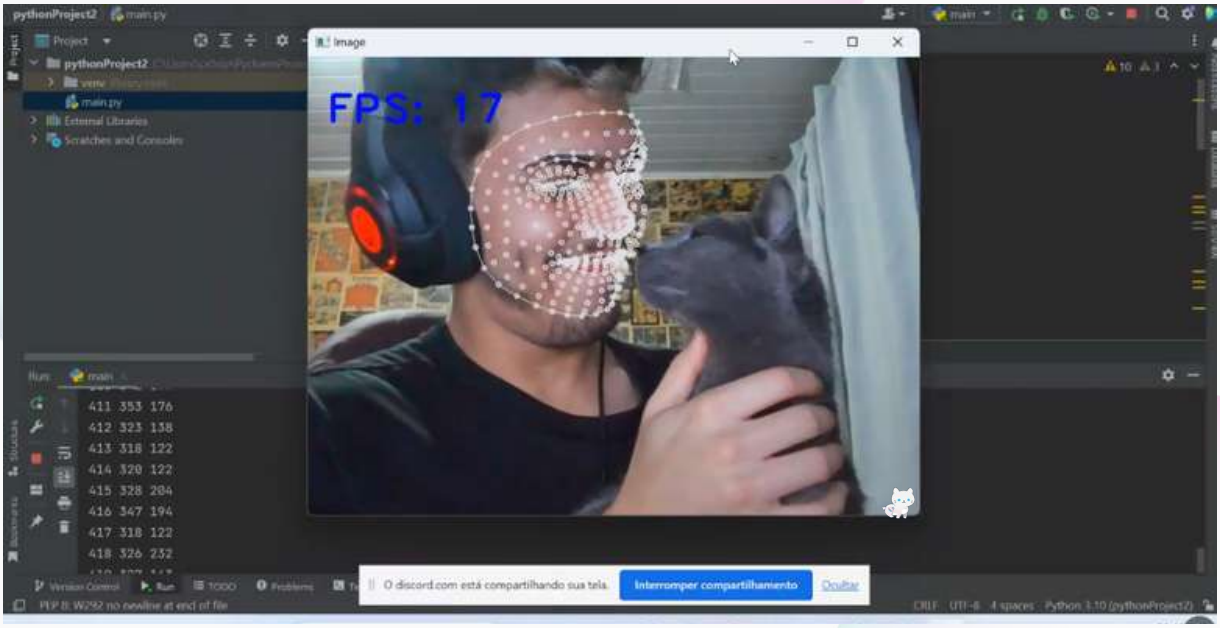
01 **Link para o vídeo de referência:**

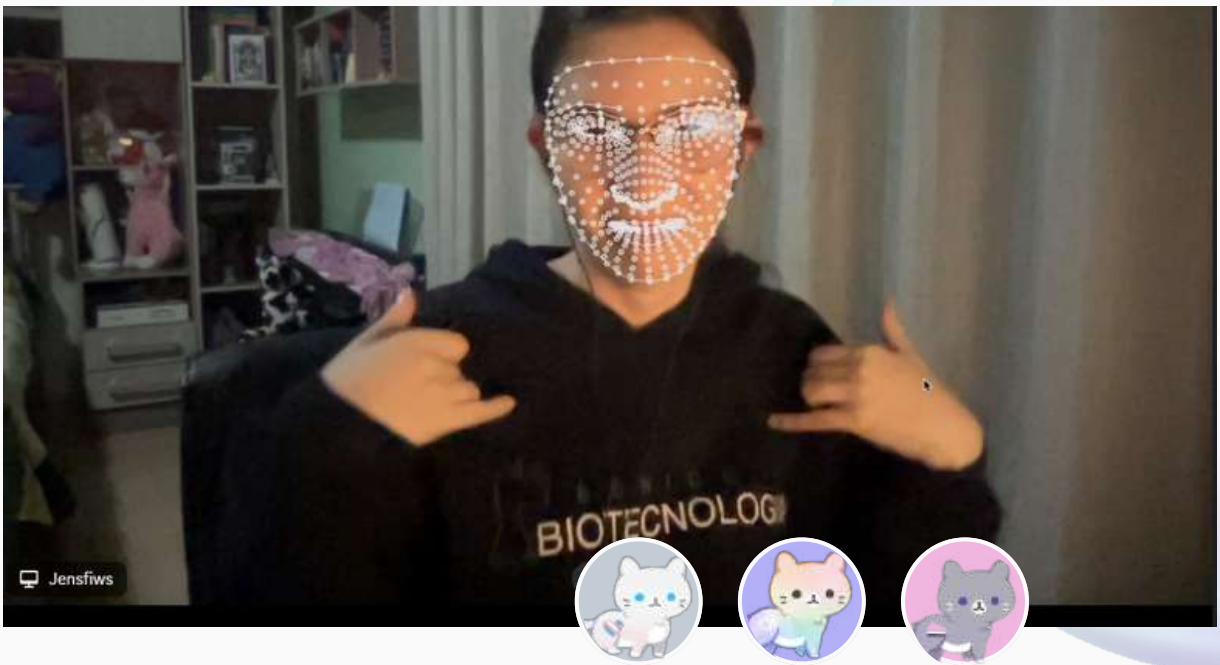
Vídeo original

03 **Link para o notebook com o código:**

Notebook do colab

# Testes realizados pelo programa FaceMesh

# Parte 2

***Para essa próxima etapa, vamos extrair os três canais RGB de cada landmark para usarmos como input da nossa rede e realizar o pré processamento***

Primeiramente, vamos utilizar as bibliotecas *opencv* e *mediapipe* para ligarmos a webcam e criarmos a malha com os 468 landmarks. A cada frame do vídeo, os três canais RGB de cada ponto serão serializados em um *csv*.

Após a serialização, esses dados são lidos e inseridos em um numpy array. EM seguida, nós fazemos a separação entre atributos e os rótulos de cada instância.

Os atributos numéricos são normalizados, e ambos os atributos e os rótulos são balanceados usando Synthetic Minority Oversampling Technique (SMOTE).

Com os dados normalizados e separados, realizamos o split entre base de treino e base de teste, utilizando 80:20 como proporção. Em seguida, utilizamos a técnica de Label Encoder, onde para cada dado categórico ao invés de passarmos uma string como rótulo, transformaremos eles em arrays binários, onde cada posição representa uma pessoa. Por fim, damos reshape nas estruturas de treino e teste para transformá-los em 3D arrays, que são o input de entrada para CNNs utilizando o tensorflow.

# Nosso csv

*Cada vídeo utilizado no treinamento de reconhecimento facial possui o nome do arquivo com o nome do indíviduio a ser reconhecido, e esta é a primeira informação adicionada ao csv. Para cada frame do vídeo é utilizada a detecção facial com 468 pontos estratégicos do rosto, cada ponto possui 3 informações (Pontos R, G e B) as quais são carregadas em nosso arquivo..*

| | |
|---|---|
| PERSON | Nome da pessoa a qual a foto corresponde, para ser usada como label |
| POINT{i}R | Corresponde ao canal vermelho do ponto em questão |
| POINT{i}G | Corresponde ao canal verde do ponto em questão |
| POINT{i}B | Corresponde ao canal azul do ponto em questão |

| | PERSON | POINT1R | POINT1G | POINT1B | POINT2R | POINT2G | POINT2B | POINT3R | POINT3G | POINT3B | POINT4R | POINT4G | POINT4B | POINT5R | POINT5G | POINT5B | POINT6R | POINT6G | POINT6B | POINT7R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | Jenni | 38 | 15 | 9 | 48 | 24 | 16 | 46 | 23 | 13 | 13 | 7 | 4 | 39 | 18 | 9 | 36 | 17 | 10 | 27 |
| 3 | Jenni | 45 | 23 | 10 | 41 | 23 | 10 | 51 | 31 | 15 | 13 | 10 | 4 | 48 | 31 | 13 | 33 | 19 | 8 | 33 |
| 4 | Jenni | 70 | 39 | 24 | 62 | 38 | 19 | 74 | 46 | 26 | 29 | 19 | 9 | 74 | 48 | 27 | 53 | 32 | 14 | 56 |
| 5 | Jenni | 108 | 62 | 39 | 101 | 64 | 31 | 109 | 67 | 33 | 54 | 36 | 19 | 106 | 66 | 34 | 91 | 55 | 26 | 94 |
| 6 | Jenni | 138 | 82 | 53 | 126 | 78 | 42 | 139 | 89 | 48 | 77 | 54 | 32 | 135 | 86 | 50 | 115 | 71 | 38 | 118 |
| 7 | Jenni | 159 | 93 | 63 | 151 | 99 | 57 | 157 | 101 | 56 | 93 | 66 | 39 | 156 | 101 | 60 | 136 | 88 | 49 | 137 |
| 8 | Jenni | 175 | 103 | 71 | 157 | 98 | 54 | 173 | 114 | 63 | 107 | 73 | 46 | 172 | 112 | 66 | 147 | 95 | 55 | 148 |
| 9 | Jenni | 183 | 107 | 73 | 175 | 113 | 63 | 183 | 120 | 66 | 115 | 76 | 51 | 184 | 122 | 73 | 158 | 102 | 62 | 164 |
| 10 | Jenni | 191 | 114 | 76 | 187 | 123 | 72 | 195 | 126 | 76 | 124 | 83 | 56 | 194 | 130 | 81 | 166 | 108 | 63 | 169 |
| 11 | Jenni | 195 | 117 | 82 | 178 | 113 | 64 | 201 | 132 | 80 | 134 | 88 | 63 | 199 | 134 | 87 | 174 | 111 | 68 | 180 |
| 12 | Jenni | 201 | 123 | 86 | 185 | 119 | 68 | 207 | 136 | 85 | 134 | 91 | 65 | 204 | 137 | 89 | 178 | 115 | 73 | 182 |
| 13 | Jenni | 205 | 130 | 89 | 192 | 125 | 77 | 209 | 138 | 87 | 132 | 91 | 64 | 206 | 140 | 89 | 186 | 120 | 79 | 185 |
| 14 | Jenni | 206 | 132 | 88 | 196 | 131 | 85 | 213 | 142 | 91 | 140 | 98 | 67 | 210 | 146 | 97 | 187 | 121 | 80 | 190 |
| 15 | Jenni | 212 | 137 | 96 | 197 | 133 | 86 | 209 | 140 | 88 | 131 | 94 | 61 | 207 | 145 | 96 | 180 | 121 | 79 | 186 |
| 16 | Jenni | 213 | 133 | 94 | 193 | 128 | 84 | 212 | 145 | 97 | 127 | 91 | 65 | 213 | 151 | 104 | 185 | 123 | 85 | 186 |
| 17 | Jenni | 213 | 137 | 94 | 193 | 128 | 82 | 215 | 146 | 96 | 130 | 91 | 70 | 214 | 149 | 102 | 184 | 125 | 88 | 190 |
| 18 | Jenni | 217 | 140 | 98 | 192 | 129 | 87 | 217 | 148 | 100 | 127 | 90 | 73 | 214 | 152 | 107 | 191 | 129 | 91 | 191 |
| 19 | Jenni | 220 | 141 | 97 | 192 | 129 | 87 | 217 | 148 | 100 | 140 | 98 | 78 | 215 | 150 | 106 | 191 | 128 | 88 | 191 |
| 20 | Jenni | 217 | 138 | 97 | 202 | 137 | 95 | 219 | 150 | 105 | 143 | 99 | 75 | 218 | 150 | 109 | 195 | 131 | 93 | 191 |

# Parte 3

***Com o pré-processamento dos dados prontos, podemos começar a pensar na arquitetura da CNN***

Como estamos trabalhando com landmarks, ou seja, pontos específicos da malha facial, e não imagens propriamente ditas, no lugar de usarmos camadas convolucionais e pooling, estamos utilizando camadas totalmente conectadas, uma vez que elas trabalham melhor com dados unidimensionais que não tem relação espacial intrínseca. Dessa forma, as camadas densas internas utilizam como função de ativação a relu, e a de output utiliza a softmax.
Já como otimizador estamos utilizando por enquanto o *rmsprop.*
Para a parte do treinamento, estamos *batches* de tamanho 16, e um número de épocas de 30.

> *Porém, esses pontos ainda estão na fase de teste, e não são os parâmetros definitivos da rede!*

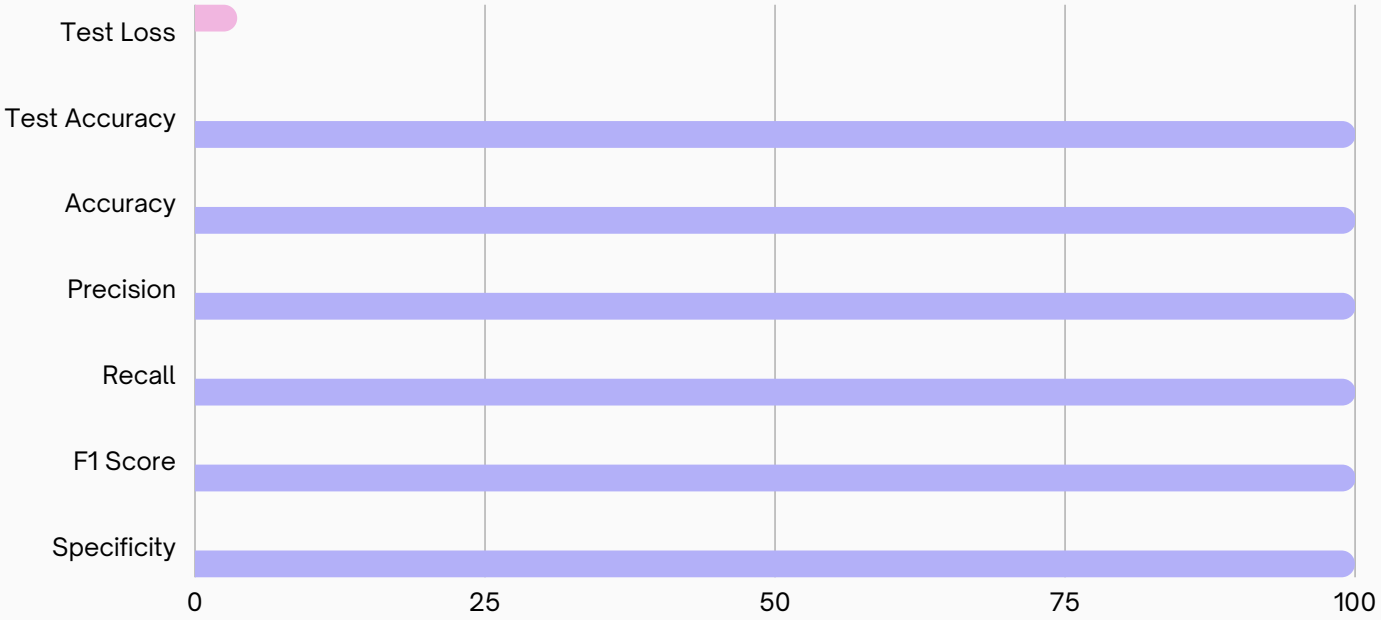
Para a avaliação da performance da rede pós treinamento, estamos utilizando várias métricas:

- Test loss;
- Test accuracy;
- Precision;
- Sensitivity;
- F1 Score;
- Specificity;
- AUC-ROC.

Além disso, estamos plotando os gráficos de acurácia e de perda total do treinamento.

# Métricas

***O que geramos de métricas para avaliar nossa CNN!***

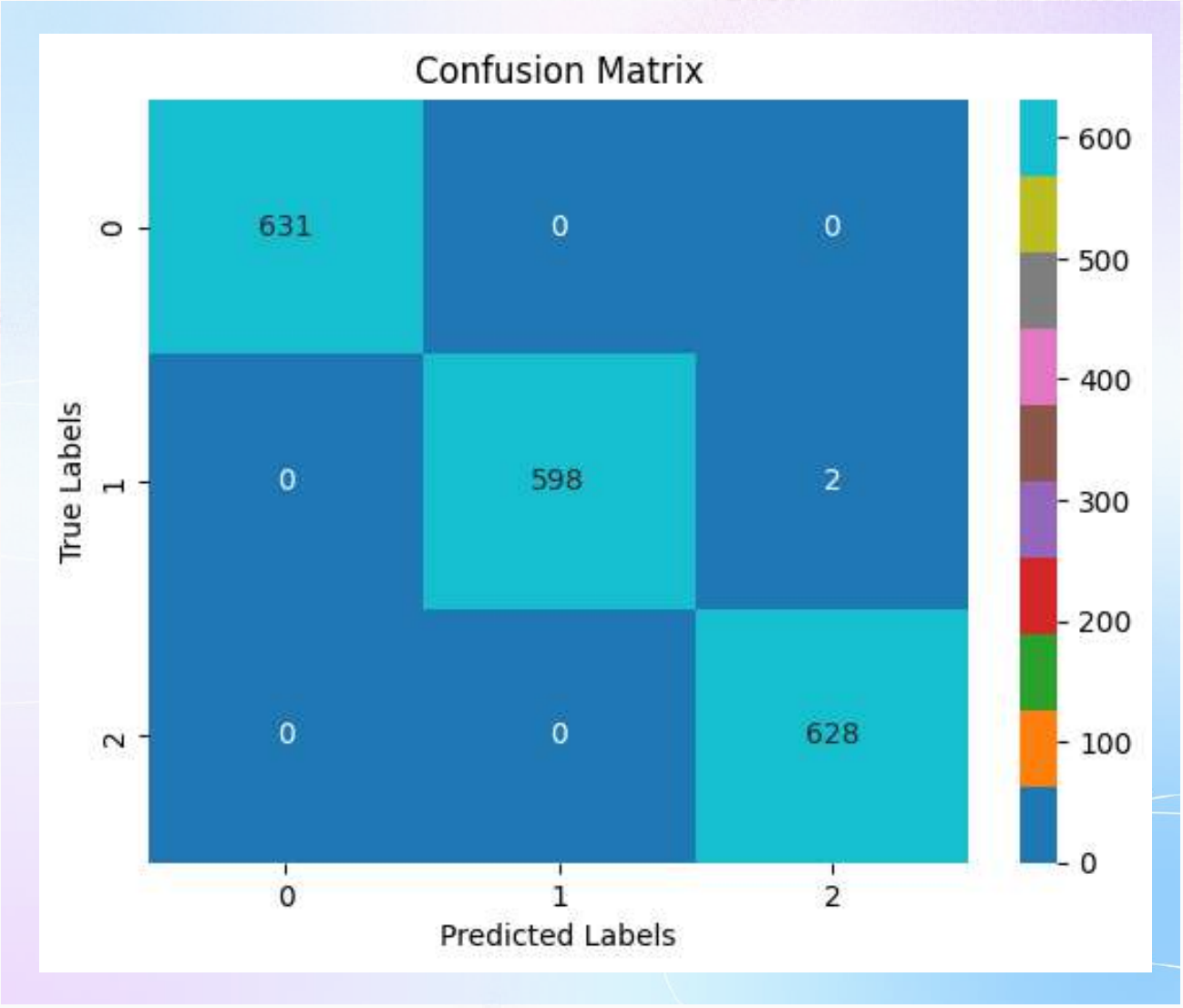



***O que é cada métrica utilizada?***

* **Precision**: proporção dos predictions realmente verdadeiros de todas as predictions, para ajudar a diminuir falsos positivos.
* **Recall (Sensitivity)**: proporção dos predictions realmente verdadeiros de todas os resultados positivos,para ajudar a diminuir falsos negativos.
* **F1 Score**: balanço entre precisão e sensibilidade.
* **Specificity**: mede a proporção de predictions realmente negativos de todo o total de amostras negativas.

# Confusion Matrix

*0 -> Label referente à Gabi*
*1 -> Label referente à Jenni*
*2 -> Label referente ao VIni*



# Parte 4

***Com a hiperparâmetrização da rede, podemos avançar com o uso do tensorflow para realizarmos o treinamento incremental da rede***

Após hiperparametrizarmos a rede, processo que levou por volta de quatro dias para ser concluído, conseguimos os melhores hiperparâmetros para a nossa rede, que foram:

> *batch size: 8*
> *Epochs: 10*
> *Optimizer: rmsprop*

Com essa etapa concluída, podemos pensar no treinamento incremental do nosso modelo.

Para isso, primeiro o modelo é pré treinado com a base de treino, e depois avaliado com a base de teste para podermos extrair suas métricas e termos numericamente seu desempenho.
Depois, utilizamos o input do usuário para saber qual opção deve ser executada: inserir uma nova pessoa no treinamento do modelo, ou fazer a inferência de uma pessoa já existente nele.
Ao optar por inserir uma nova pessoa no modelo, os landmarks dessa nova pessoa são extraídos e serializados em um csv com o mesmo formato utilizado durante o pré treinamento.

# Inserção de novos dados

## *Treinamento incremental vs Inferência*

O nosso csv gerado recebe o mesmo pré processamento utilizado no treinamento, onde os dados são normalizados, balanceados e as labels passam pelo encode. Após isso, o modelo pré treinado é carregado e atualizado com a nova classe a ser inserida. Após isso, o modelo atualizado é salvo novamente para ser utilizado pela inferência.

> *Com a nova classe inserida no modelo pelo treinamento incremental, ela pode ser inferida com o predict!*

A segunda opção para o usuário é fazer a inferência de uma nova instância, ou seja, reconhecer pessoas que já foram apresentadas ao modelo pelo treinamento. A cada frame do vídeo da webcam, os landmarks são capturados, normalizados e formatados para o formato esperado pelo modelo. O modelo faz o predict desse dado e exibe na tela a label correspondente à pessoa sendo exibida na webcam.
*É essencial lembrar que a inferência não consegue identificar corretamente a label da nossa instância se ela não foi previamente inserida no modelo através do treinamento incremental.*

# Como incrementar?

***Como vamos realizar a etapa de treinamento incremental utilizando o Keras?***

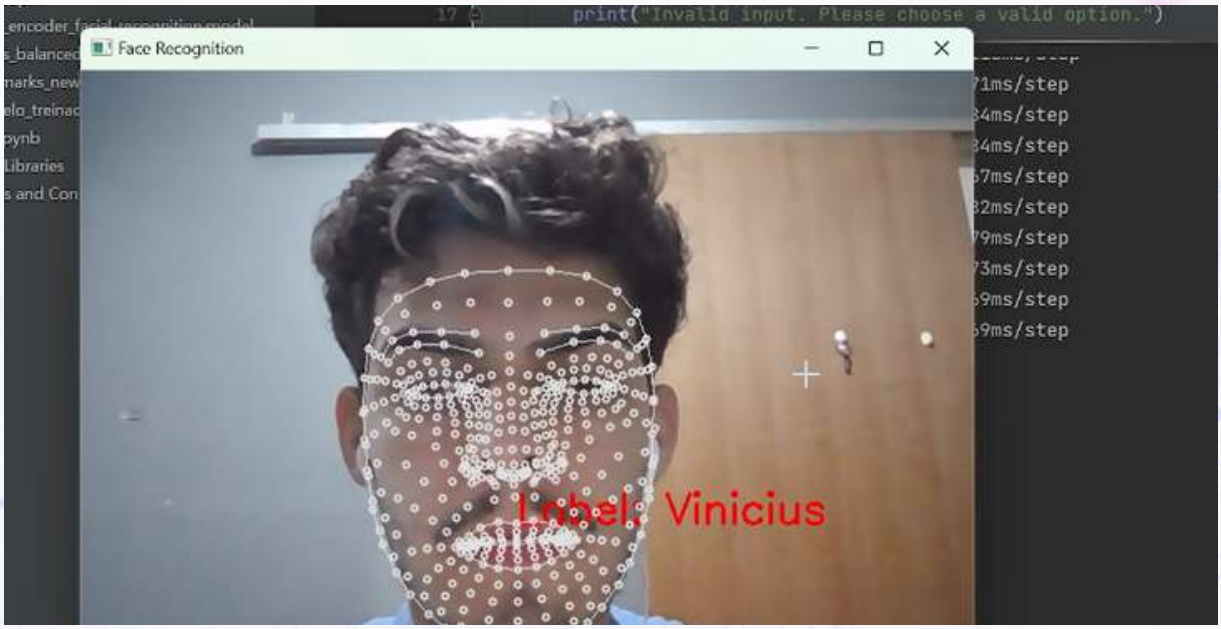
Essa é, com certeza, a etapa mais difícil de ser realizada até então. Para podermos adicionar mais classes mantendo as classes previamente treinadas, há dois pontos de atenção a serem tomados.

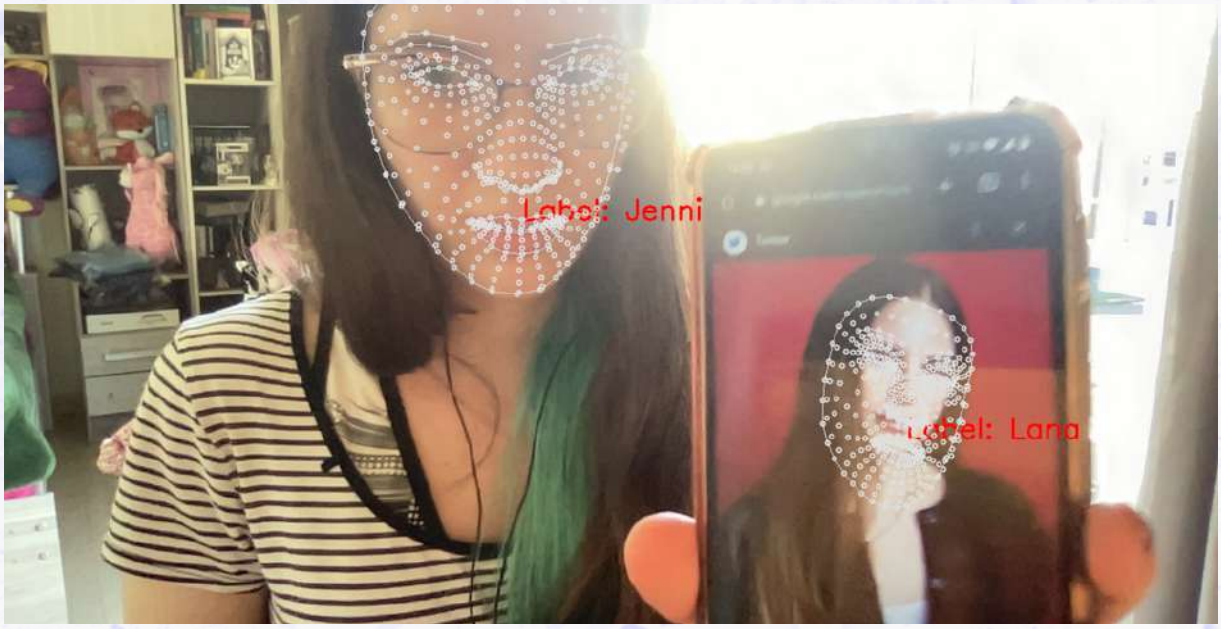
O primeiro é em relação ao Label Encoder e a encodificação das labels. Para cada nova classe, um novo Label Encoder é criado, e ele recebe tanto as labels de treinamentos passados quanto as labels da nova classe, para que seja capaz de reconhecer todos. Além disso, a cada nova classe inserida as labels precisam ser remodeladas, já que as colunas de dados categóricos se alteram a cada inserção. O novo Label Encoder é salvo em memória para ser utilizado no processo de inferência.

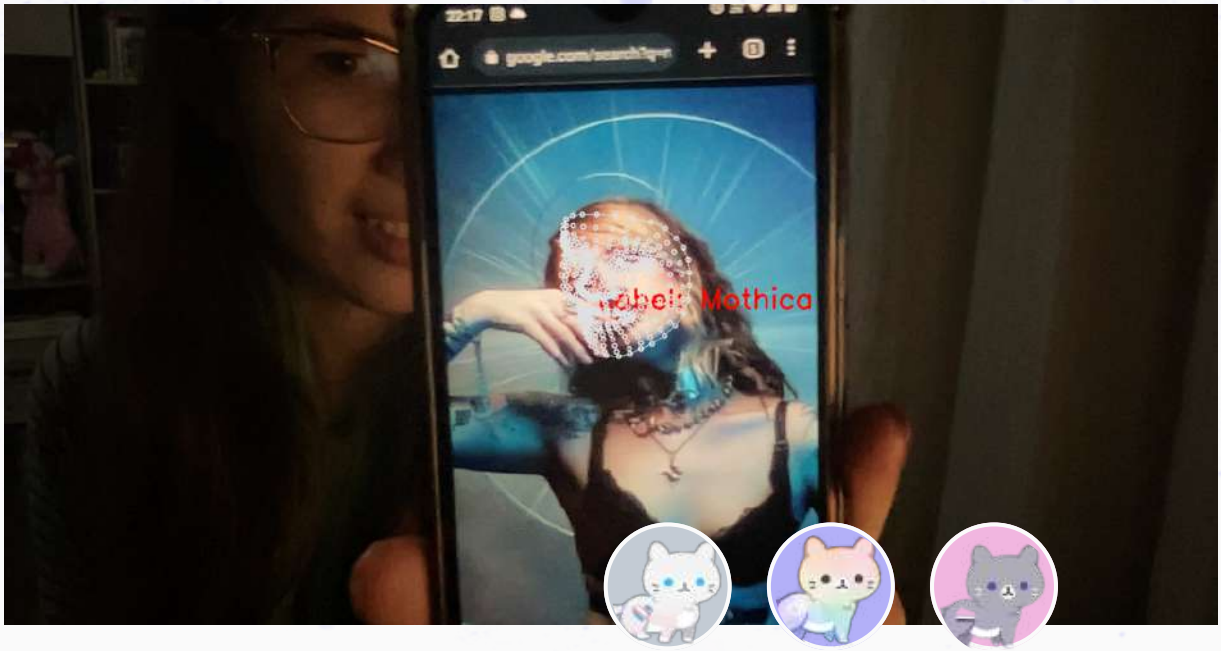
O próximo ponto de atenção é com o modelo. Não é possível fazer o fit de novas classes diretamente, pois isso sobrescreveria treinamentos passados. Para realizar o treinamento incremental, é necessário adicionar camadas no modelo que comportem os novos dados, além de alterar o output da camada final, para que ele comporte o novo número máximo de classes.

Outro cuidado é de, durante a nova inserção, congelar os pesos das camadas de treinamento já existentes, para que essas não se percam conforme o modelo é atualizado. Por fim, o modelo atualizado é salvo em memória para poder ser reutilizado no processo de reconhecimento facial.

# Teste realizados inserindo novas classes



# Próximos passos

## Definições

Até 06/06

Agora que temos a parte com a câmera funcionando, podemos serializar as coordenadas faciais em um csv para podermos usar-las no treinamento da rede.

Com esses dados, podemos começar o processo de balanceamento e normalização dos dados, além de fazermos a tratativa das cores.

Também podemos começar a pensar na arquitetura da CNN e na construção das camadas convolucionais e no filtro.

## Preprocessing e Treinamento

Até 13/06

Agora temos o pré processamento dos dados prontos para serem consumidos pela rede. Podemos gerar o csv definitivo de treinamento com amostras de todos os integrantes do trio.

Também temos o esqueleto da CNN, mas precisamos realizar testes para definirmos os *hiperparâmetros* da nossa rede, assim como realizarmos testes de perturbação para avaliarmos as métricas retornadas.

Também podemos começar a rascunhar o processo de inferência de novas instâncias da nossa rede. Podemos tentar fazer o fluxo tanto por vídeo quanto passando fotos estáticas para a rede.

# Próximos passos

## Resultados obtidos

Até 20/06

Temos o modelo pré treinado com nossa base, podemos começar a desenhar e testar o fluxo do treinamento incremental.

Também precisamos criar o fluxo de inferência de novas instâncias, e testar como mostrar as labels em vídeo simultaneamente com a webcam.

## Apresentação

Até 27/06

Terminar o fluxo de inserir novas classes, consertando o problema com as one-hot labels incrementais.

Testar e coletar evidências do funcionamento tanto da inferência quanto da inserção de novas classes.

Terminar o relatório e gravar o vídeo demonstrativo do projeto.

# That's it folks! :)

*Wow much AI*
*Such cool code*