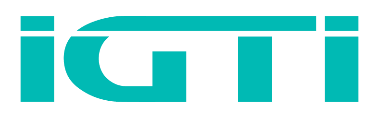
****

Relatório do Projeto Aplicado

Pesquisa Local Imagem

Pelo Conteúdo

|  |  |
| --- | --- |
| Nome | Etienne Vanhaecke |
| Curso | Deep Learning |
| Orientador(a) | [Professor Túlio Vieira](https://igti.instructure.com/courses/2858/users/9218) |
| Data | 03/05/20 (versão inicial)  25/04/21 (revisão depois trancamento PA)  07/08/21 (Sprint 3 completado)  11/08/21 (Sprint 4 completado)  11/09/21 (Sprint 5 completado)  03/10/21 (Sprint 6 completado)  23/10/21 (Sprint 7 completado)  01/11/21 (Relatório Final) |

**Sumário**

[1.1 Professor Túlio Vieira 2](#_Toc87130935)

[1. CANVAS do Projeto Aplicado 4](#_Toc87130936)

[1.1 Desafio 5](#_Toc87130937)

[1.1.1 Análise de Contexto 6](#_Toc87130938)

[1.1.2 Personas 7](#_Toc87130939)

[1.1.3 Benefícios e Justificativas 8](#_Toc87130940)

[1.1.4 Hipóteses 10](#_Toc87130941)

[1.2 Solução 11](#_Toc87130942)

[1.2.1 Objetivos SMART 11](#_Toc87130946)

[1.2.2 Premissas e Restrições 12](#_Toc87130947)

[1.2.3 Backlog de Produto 13](#_Toc87130948)

[2. Área de Experimentação 14](#_Toc87130949)

[2.1 Sprint 1 14](#_Toc87130952)

[2.1.1 Solução 14](#_Toc87130956)

[2.1.2 Lições aprendidas 17](#_Toc87130957)

[2.2 Sprint 2 17](#_Toc87130958)

[2.2.1 Solução 17](#_Toc87130961)

[2.2.2 Lições aprendidas 35](#_Toc87130962)

[2.3 Sprint 3 36](#_Toc87130963)

[2.3.1 Solução 36](#_Toc87130965)

[2.3.2 Lições aprendidas 45](#_Toc87130966)

[2.4 Sprint 4 46](#_Toc87130967)

[2.4.1 Solução 46](#_Toc87130969)

[2.4.2 Lições aprendidas 64](#_Toc87130970)

[2.5 Sprint 5 65](#_Toc87130971)

[2.5.1 Solução 65](#_Toc87130973)

[2.5.2 Lições aprendidas 75](#_Toc87130974)

[2.6 Sprint 6 76](#_Toc87130975)

[2.6.1 Solução 76](#_Toc87130976)

[2.6.2 Lições aprendidas 82](#_Toc87130977)

[2.7 Sprint 7 83](#_Toc87130978)

[2.7.1 Solução 83](#_Toc87130979)

[2.7.2. Lições aprendidas 90](#_Toc87130980)

[3. Considerações Finais 91](#_Toc87130981)

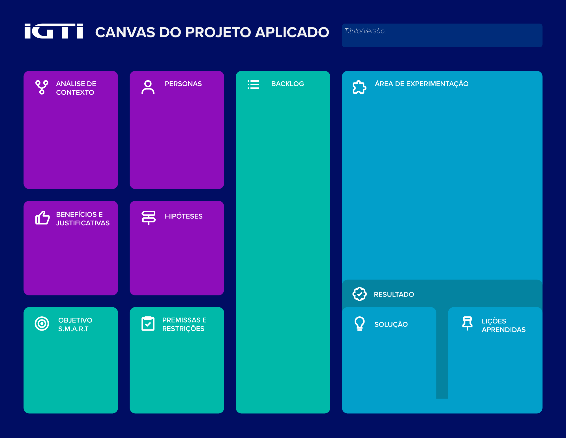
[3.1 Resultados Finais 91](#_Toc87130982)

[3.2 Contribuições 94](#_Toc87130983)

[3.3 Próximos passos 95](#_Toc87130984)

# 1. CANVAS do Projeto Aplicado

**Figura conceitual, que representa todas as etapas do Projeto Aplicado.**



## Desafio

### Análise de Contexto

Desde a aparição da fotografia numérica, quase 25 anos atrás, a volumetria de fotos que se tira e armazena, não para de crescer (ainda mais com o surgimento do smartphone que permite a cada um de tirar foto de qualquer lugar e a qualquer momento).

Não está difícil que uma pessoa tenha nos seus repositórios locais, dezenas até centenas de milhares de fotos, sem as vezes um mínimo de organização delas.

Isso acaba por gerar uma tremenda dificuldade para posteriormente pesquisar as fotos de interesse, ademais de um desgaste de tempo na pesquisa.

Estou ressentindo pessoalmente esta dificuldade e as pessoas do meu círculo pessoal apresentam a mesma dor.

* Matriz Certezas, Suposições e Duvidas CSD:

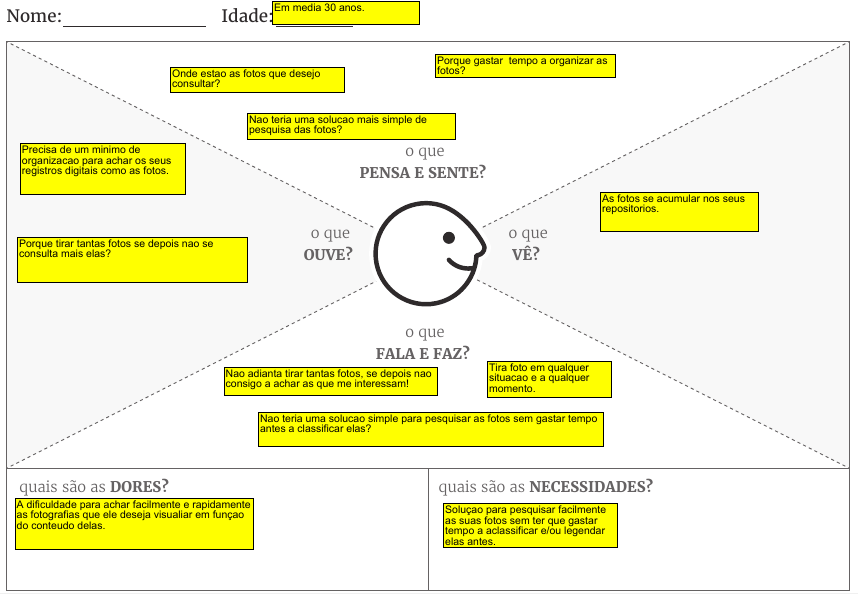
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | Matriz CSD - Listar todas as Certezas, Suposições e Dúvidas | | |
| Certezas | Suposições | Dúvidas |
| Diferentes óticas da analise | Atores | As pessoas desejam pesquisar facilmente e rapidamente as fotos em função do conteúdo delas. |  | As pessoas aceitariam de gastar tempo a montar um jogo de fotos representativo do conteúdo pessoal que elas desejam pesquisar? |
| Cenários | As pessoas acumulam a um ritmo quase exponencial fotografias. | Uma boa parte destas fotos carece de uma organização mínima, complicando muito a pesquisa tradicional delas. |  |
| Regras |  | As pessoas aceitariam de pagar um serviço ou um produto para facilitar a pesquisa das suas fotos. | As pessoas aceitariam de ceder este jogo de fotos legendados em uma plataforma distante para treinamento acelerado do modelo de pesquisa personalizado? |

* Observações POEMS:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PESSOAS | OBJETOS | AMBIENTE | MENSAGEM | SERVIÇOS |
| O grande público em geral, mais orientado para as gerações Y e Z. | Fotos com conteúdo padrão (praia, montanha, cachorro, pôr do sol....). | Repositório local das fotos. | Dificuldade para achar as fotos | Classificação das fotos por conteúdo padrão. |
|  | Fotos com conteúdo pessoal (os meus filhos, os meus pais, o meu marido/esposa, os meus amigos...). | Repositório na Cloud das fotos | Tempo gastado a tentar classificar as fotos | Classificação das fotos por conteúdo pessoal. |
|  |  |  |  | Pesquisa das fotos por conteúdo. |
|  |  |  |  |  |

### Personas

O grande público em geral, mais das gerações Y e Z, que acaba por tirar e armazenar milhares de fotos pessoais e que tem dificuldade para depois consultar as fotos de interesse.



**PESSOA:**

- Nome: Ana

- Idade: 30 anos

- Cargo: Analista de marketing

- Formação: Graduada.

- Ela gosta tirar muitas fotografias durantes seus encontros com a família, amigos, durante seus passeios.

- Ela acumula estas fotografias localmente sem a menor organização.

- Ela está regularmente irritada de passar tanto tempo a pesquisar as fotos que ela deseja visualizar em função de um tema específico (local, pessoa...).

- O nosso produto permitiria de resolver esta dor, lhe permitindo de achar facilmente e rapidamente as fotos, seguindo a passar um mínimo de tempo na organização delas.

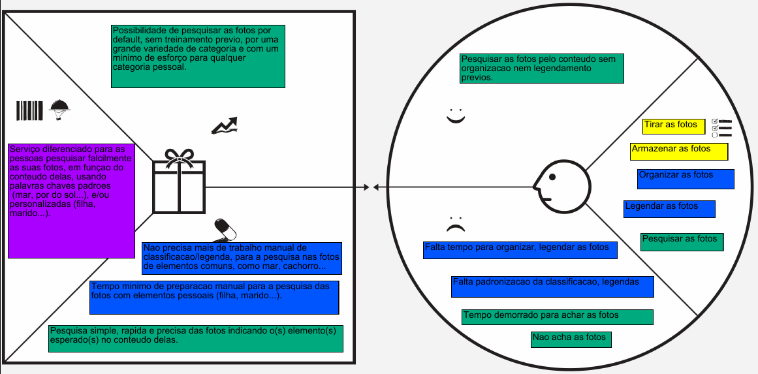
### Benefícios e Justificativas

* Business Design Blueprint:
  + Exploração do problema

|  |  |  |  |
| --- | --- | --- | --- |
| **Ações do**  **Cliente** | **Classifica as fotos** | **Legenda as fotos** | **Pesquisa as fotos** |
| Objetivos | Organizar as fotos | Colocar legendas nas fotos | Encontrar as fotos em função de um conteúdo |
| Atividades | Copia em pasta correspondendo ao conteúdo das fotos | Usar o nome ou tags das fotos para indicar o conteúdo delas. | - Percorrer manualmente a arborescência das fotos.  - Pesquisar palavra(s) chave(s) no nome e/ou nos tags das todos,  - Visualizar as fotos. |
| Questões | Qual está a arborescência mais apropriada para uma futura pesquisa das fotos? | Quais são as palavras mais apropriadas para legendar as fotos? | Como acelerar a pesquisa?  Como achar todas as fotos relativas a um certo conteúdo? |
| Barreiras | Tempo gastado para realizar esta organização das fotos. | Tempo gastado para legendar as fotos.  Falta de padrão nas legendas. | Tempo gastado para realizar esta pesquisa manual. |

* + Exploração da solução

|  |  |  |  |
| --- | --- | --- | --- |
| **Ações do**  **Cliente** | **Classifica as fotos** | **Legenda as fotos** | **Pesquisa as fotos** |
| Funcionalidades | Pesquisa automática baseada sobre palavras chaves, necessitando organização só das fotos usadas para treinamento do modelo de pesquisa. | Pesquisa automática baseada sobre palavras chaves, necessitando a legenda só das fotos usadas para treinamento do modelo de pesquisa. | Pesquisa automática baseada sobre as palavras chaves indicadas pelo usuário, por reconhecimento de padrões ligados a estas palavras chaves, dentro do conteúdo das fotos. |
| Interação | Definição da arborescência de organização das fotos de treinamento (pessoas, lugares...). | Seleção das palavras chaves das fotos de treinamento (pessoas, lugares...). | Indicação pelo usuário das palavras chaves do conteúdo das fotos que ele deseja pesquisar. |
| Mensagem | Não passa mais tempo a classificar as suas fotos. | Não passa mais tempo a legendar as suas fotos. | Nunca foi tanto fácil achar as suas fotos. |
| Onde ocorre | Repositório local do usuário ou na Cloud. | Repositório local do usuário ou na Cloud. | Repositório local do usuário ou na Cloud. |
| Tarefas aparentes | Pesquisa automática das fotos por reconhecimento das palavras chaves dentro do conteúdo das fotos. | Pesquisa automática das fotos por reconhecimento das palavras chaves dentro do conteúdo das fotos. | Pesquisa automática das fotos por reconhecimento das palavras chaves dentro do conteúdo das fotos. |
| Tarefas escondidas | Treinamento dos modelos de pesquisa por deep learning. | Treinamento dos modelos de pesquisa por deep learning. | Treinamento dos modelos de pesquisa por deep learning. |
| Processos de suporte | Site de ajuda on-line | Site de ajuda on-line | Site de ajuda on-line |
| Saída desejável | Fim do trabalho repetitivo de classificação. | Fim do trabalho repetitivo de legenda. | Pesquisa automática, rápida e assertiva das fotos. |

* CANVAS de Proposta de Valor:
* 

### Hipóteses

1. As pessoas acumulam fotografias sem um mínimo de organização para facilitar a pesquisa posterior delas.
2. As pessoas desejam uma solução de pesquisa de fotografias que necessita um mínimo de investimento delas.
3. As pessoas aceitariam pagar para um serviço de organização das fotografias.
4. As pessoas aceitariam ceder as suas fotos em uma plataforma distante para treinamento personalizado das temas que elas indicariam.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | BENEFICIOS | ABRANGÊNCIA | SATISFAÇÃO | INVESTIMENTO | CLIENTE | OPERAÇÕES | TOTAL |
| Aplicação facilitando a classificação e/ou a legenda manual das fotografias | 2 | 3 | 2 | 4 | 4 | 4 | **19** |
| Aplicação usando o deep learning para identificar automaticamente as fotos a traves o seu conteúdo | 5 | 5 | 5 | 3 | 4 | 4 | **26** |
|  |  |  |  |  |  |  |  |



Fonte: <https://engenhariaexercicios.com.br/gestao-de-qualidade/matriz-gut-basico-conceito-aplicacao-das-matrizes-priorizacao/>

## Solução



### Objetivos SMART

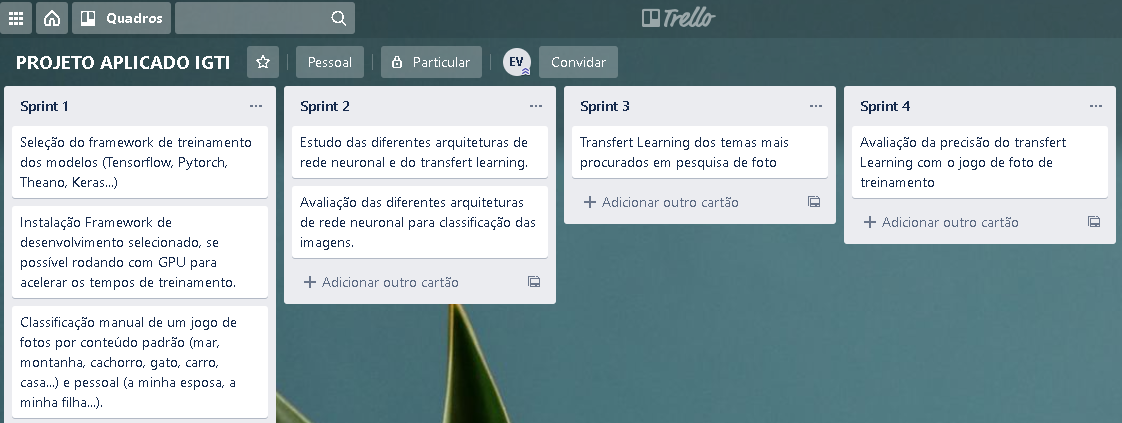
* Entregar, até o fim da pós-graduação, algumas pesquisas automáticas de fotografias com elementos comuns (exemplo presença na foto do mar, de montanha, de cachorro...) pelos quais se recuperara os parâmetros dos modelos de pesquisa do domínio público.
* Entregar, até o fim da pós-graduação, algumas pesquisas automáticas de fotografias com elementos pessoais (exemplo presença na foto do marido, da filha...) pelos quais os parâmetros dos modelos de pesquisa serão definidos por um treinamento prévio guiado pela aplicação.
* Entregar, até o fim da pós-graduação, uma versão beta de uma interface gráfica permitindo ao usuário de pesquisar as fotos indicando o conteúdo esperado delas, realizar os treinamentos para os elementos pessoais de pesquisa e dar feedback da pertinência do resultado das pesquisas.

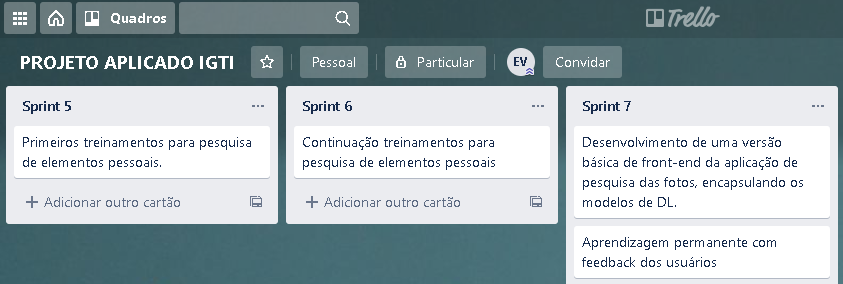
### Premissas e Restrições

* PREMISSAS:
  + Tempo dedicado ao desenvolvimento de 2 horas por dia, contando o fim de semana.
  + Uso de framework de deep learning com interface com Python, a selecionar (Keras/Tensor Flow/Pytorch...).
  + Uso da versão GPU do framework selecionado para acelerar os tempos de treinamento.
  + Capacidades de processamento local, via GPU, suficientes para o treinamento e a pesquisa em um tempo aceitável pelo usuário.
* RESTRICOES:
  + O prazo para entrega do protótipo não pode passar de Outubro 2021.
  + Protótipo baseado sobre uma versão autônoma local da aplicação (sem dependência com recursos da Web, como capacidades de armazenamento, de processamento de uma Cloud).
* MATRIZ DE RISCOS

|  |  |  |  |
| --- | --- | --- | --- |
| **Risco Identificado** | **Impacto potencial** | **Ações preventivas** | **Ações corretivas** |
| **Falha da instalação ou disponibilidade do framework deep learning para GPU local** | Tempo inviável de treinamento dos modelos. | Preferência de um framework com facilidade de instalação local com uso do GPU da carta gráfica. | - Migração dos treinamentos em uma Cloud oferecendo serviço de treinamento com GPU (exemplo de Google Colab).  - Uso de jogo de treinamento menor. |
| **Sobre carga em atividades pessoais/professionais, impedindo de passar 2 horas por dia para o dev** | - Atraso da entrega.  - Entrega parcial. | - Planejamento das atividades professionais.  - Avisar a família, amigos da disponibilidade menor devido à realização do projeto unificado. | - Em caso de sobrecarga professional, usar um período maior durante o fim de semana para avançar no projeto. |
| **Capacidade de processamento local insuficiente** | Tempo inviável de treinamento dos modelos. | Verificar a configuração local antes de instalar o produto, oferecer o serviço. | Oferecer o serviço na Cloud. |
| **Resistencia do usuário a instalação local do framework de IA usado pela aplicação.** | Perda de cliente |  | Oferecer o serviço na Cloud. |

### Backlog de Produto





# 2. Área de Experimentação

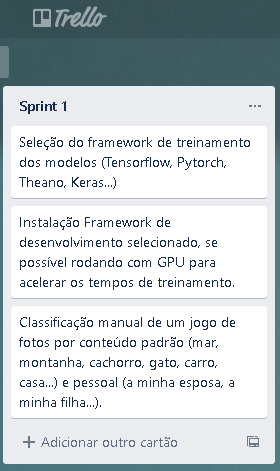


## Sprint 1



### Solução

* Evidência do planejamento:



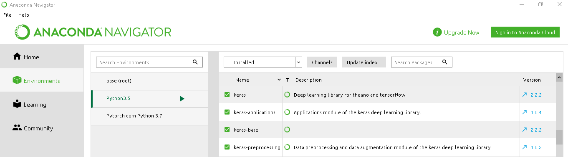
* Evidência da execução de cada requisito:
  + Seleção do framework de desenvolvimento

Como os cursos do MBA DL estão usando Tensorflow em backend e Keras em frontend, foi decidido de desenvolver o projeto com estes mesmos frameworks.

* + Instalação Framework de desenvolvimento

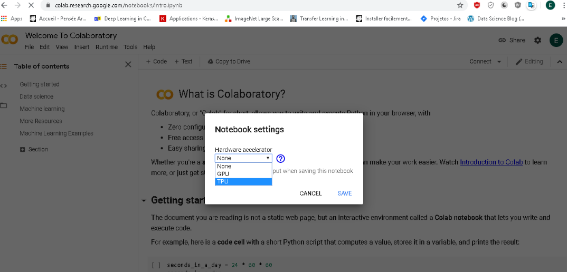
Vai ter uso de duas plataformas de desenvolvimento:

* + - Local no notebook, instalando com Anaconda os módulos Tensorflow e Keras para Python 3.5 e para CPU:

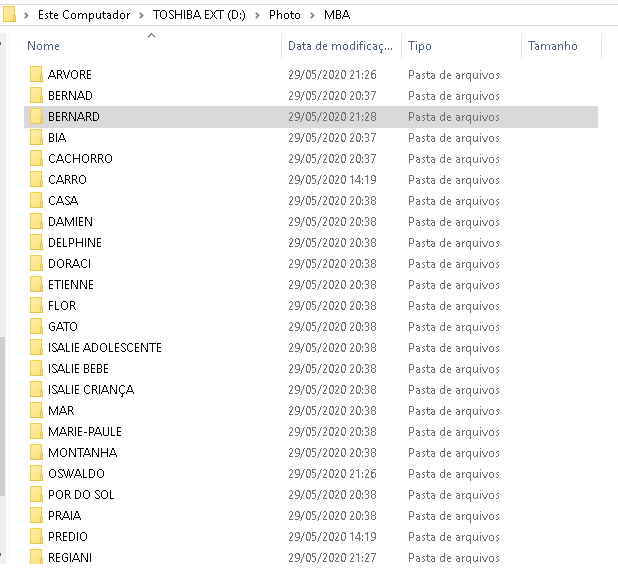




* + - Na cloud, com o Colab da Google, para poder aproveitar do processamento via GPU e TPU:



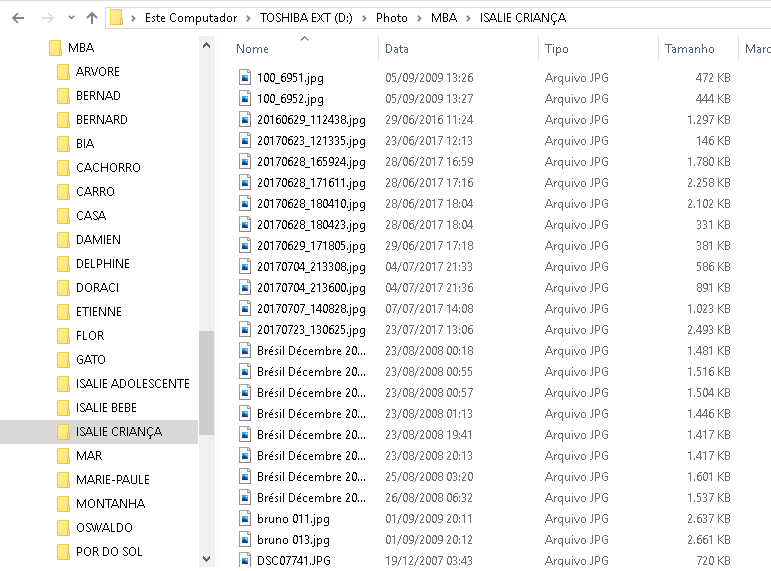
* + Classificação manual de um jogo de fotos pessoais por categoria:



* Evidência da solução:

As fotos do jogo de treinamento foram classificadas por categoria:

Exemplo do conteúdo de uma pasta:



### Lições aprendidas

* As GPU Intel não estão adequadas para acelerar o treinamento de modelos Deep Learning.
* Visto a volumetria relativamente pequena de fotos de cada categoria, não parece oportuno de separar este jogo de fotos entre jogo de treinamento e jogo de teste.

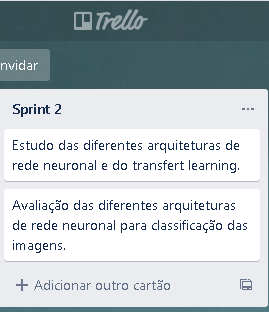
A melhor solução, para avaliar a generalização de cada modelo depois do treinamento, parece estar a validação cruzada.

## Sprint 2



### Solução

* Evidência do planejamento:

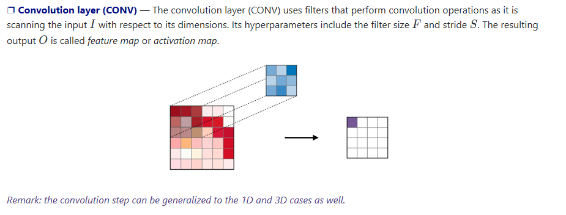


* Evidência da execução de cada requisito:
  + Estudo das diferentes arquiteturas de rede neural para a classificação de imagem

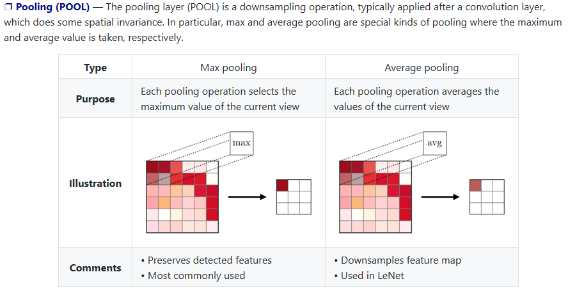
A classe de rede neural a mais apropriada para o processamento e analise de imagem digital, incluindo as técnicas de classificação de imagem e de detecção de objetos dentro das imagens, que serão usadas no projeto, está a CNN (Convolutional Neural Network).

Se trata de uma rede neural artificial profunda de tipo *feed-forward,* formada de dois principais elementos simples, característicos desta classe de rede:

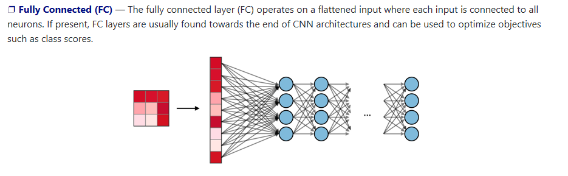
* Camada de Convolução.



* Camada de Pooling



Elas costumam acabar no final por camadas densamente conectadas, onde se realiza a classificação, a parte das features extraídas automaticamente pelas camadas especializadas anteriores de convolucao e de pooling.



Observação: As ilustrações foram copiadas do site https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks

A grande força da CNN, em comparação aos outros algoritmos tradicionais de classificação de imagem, esta que ela aprende automaticamente os filtros, que em algoritmos tradicionais deveriam estar implementados manualmente.

A pesar da relativa simplicidade destes dois elementos característicos da CNN (Convolucao e pooling layers), existe uma infinidade possibilidade de arranjo delas para resolver um problema de visão computacional.

Felizmente, existem padrões de configuração e de arquitetura CNN, que podem estar usados para desenvolver CNN muito profundas. Mais uma rede está profunda, mais ela pode identificar e classificar imagens em função de detalhes finos da imagem, mas esta profundidade faz correr o risco do “*vanishing gradient problem”,* ligado ao sinal de erro zerar, durante a fase de retro propagação do treinamento, antes de chegar nas primeiras camadas da rede, impossibilitando o treinamento destas camadas iniciais.

Abaixo um resume de quatro padrões muito bem-sucedidos, que foram precursores de padrões ainda de atualidade hoje:

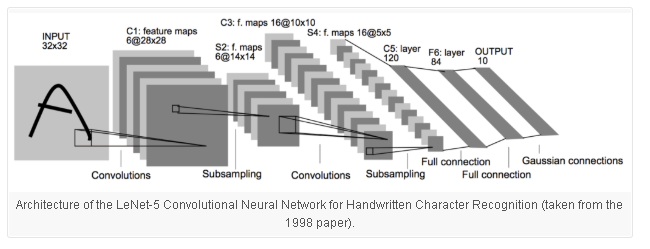
* LeNet5:

Provavelmente a mais antiga CNN a haver-se destacado na classificação de imagem.

Ela foi desenvolvida por Yan Lecun em 1998, para o reconhecimento de caracteres escritos a mão.

“[Gradient-Based Learning Applied to Document Recognition](https://ieeexplore.ieee.org/abstract/document/726791)” ([get the PDF](http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf)).

Ela está formada de 7 camadas, de input imagens em nível de cinza de tamanho 32x32.



O esquema de bloco de camada convolucional e de camada de pooling, agrupadas e se repetindo, corresponde a um esquema padrão das CNN que segue a estar aplicado nas CNN mais recentes.

O número de filtro aplicado está relativamente baixo, em comparação as aplicações mais recentes de CNN (6 para o primeiro bloco e 16 para o segundo), mas ela obedece a um padrão, ainda de atualidade nas CNN de hoje, de aumento do número destes filtros mais se avança na rede

A camada flatten (enfileiramento de todas as features extraídas com os filtros dos blocos de convolucao/pooling), seguida das camadas densamente conectadas, está também um padrão que segue a estar aplicado nas CNN atuais.

Elas correspondem ao classificador da rede, enquanto a primeira parte das camadas convolucionais e de pooling corresponde ao extrator das features.

* AlexNet

Modelo decorrente do trabalho de Alex Krizhevsky, et al., em 2012, titulado “[ImageNet Classification with Deep Convolutional Neural Networks](http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks).”

Ele está na origem da volta de interesse do Deep Learning e o início na dominação dele em grande parte dos problemas de visão computacional, ganhando este ano o concurso ImageNet de classificação de imagens de objetos em 1000 categorias.

Ele implementou vários métodos que estavam novidades na época e que acabaram por virar um padrão no uso das CNN:

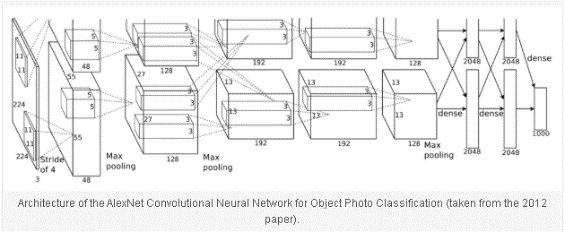
- Uso do ReLU (“*Rectified Linear*”, como função de ativação não linear, em vez de funções como Tanh/Sigmoid, que estavam comumente usadas nesta época.

- Uso do Softmax como função de ativação da camada de saída, para os problemas de classificação multi-classe mono label.

- Pooling médio foi substituído por max-pooling.

- Uso do *“Drop out”* entre as camadas densamente conectadas para ter uma melhor generalização do modelo.

O stride de 2 está o mesmo para as camadas convolucionais e de pooling.



O input está formado de imagens de dimensões 224x224 com 3 canais de cor (RGB).

Tem 5 camadas convolucionais na parte de “*feature extractor*” e 3 camadas densamente conectadas na parte de classificação.

Segue, em grande parte, o padrão do número crescente de filtros avançando na rede (96, 256, 384, 384 e 256) e a diminuição do tamanho dos filtros (11x11, 5x5 e 3x3). Os tamanhos de filtro 5x5 e 3x3 estão agora a norma.

Outra novidade, que ficou também um padrão, está a sequência de duas camadas de convolucao de mesmo tamanho.

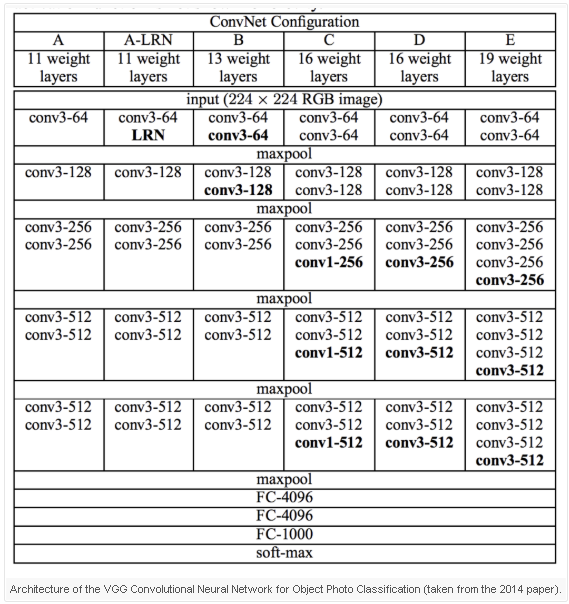
O modelo foi treinado com o método de “Data Augmentation”. Consiste a aumentar artificialmente o tamanho do dataset, gerando novas imagens aplicando vários filtros (translação, zoom, rotação...) sobre imagens originais, afim de permitir ao modelo de aprender as mesmas features sobre várias orientações.

* VGG

Este modelo, do nome do laboratório onde ele foi concebido (Visual Geometry Group a Oxford), foi descrito, em 2014, no artigo “[Very Deep Convolutional Networks for Large-Scale Image Recognition](https://arxiv.org/abs/1409.1556)”, por Karen Simonyan and Andrew Zisserman.

Ele ficou o standard das arquiteturas de CNN profundas e está obtendo resultados melhores que os anteriores.

Uma das principais diferenças, que ficou um standard, está o uso de uma grande volumetria de filtros, aumentando avançando na rede (64, 128, 256 e 512 ao final da parte de extração), de pequeno tamanho (3x3 ou 1x1) com um stride de 1 (o modelo LeNet5 usa filtros de grandes tamanhos e o modelo AlexNet filtros menores, mas de amplo stride).



Tem várias variantes deste modelo em função do número de camadas de convolucao.

As variantes mais usadas, devido a performance delas, são as chamadas VGG-16 e VGG-19, com respectivamente 16 e 19 camadas.

Tem agrupamento de várias camadas de convolucao (2, 3 até 4), de mesmo tamanho (comprimento x largura x filtros), antes da aplicação de uma camada de max pooling (de mesmo tamanho 2x2 e stride em toda a rede), formando um bloco de convolucao. O racional esta que várias camadas de convolucao de filtros pequenos aproximam o efeito de uma camada de filtros maiores (assim 3 camadas de filtros 3x3 aproximam o resultado, em termo de identificação de feature, de 1 camada de filtro 7x7).

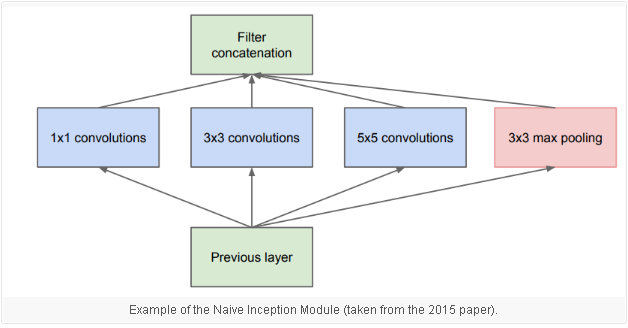
Este trabalho foi também o primeiro a liberar os pesos, permitindo o reuso de modelos já treinados para resolver uma grande variedade de problemas de visão computacional, a traves o método de transfer learning.

* Inception e GoogLeNet

Tem como origem o artigo, escrito em 2015 por Christian Szegedy, et al. Intitulado “[Going Deeper with Convolutions](https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html).”

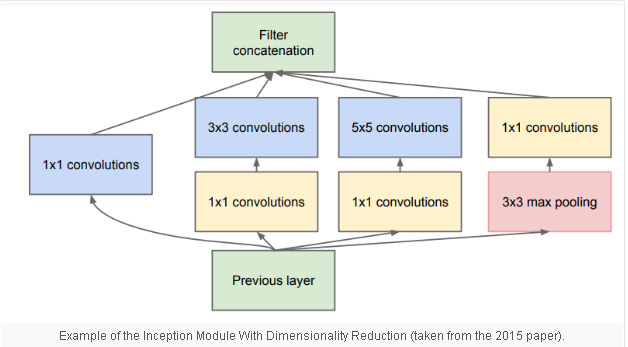
Baseada sobre uma nova arquitetura chamada incepção e um modelo específico chamado GoogLeNet, que atingiram excelentes resultados no concurso 2014 de ImageNet.

A maior inovação está o bloco de incepção, formado de várias camadas convolucionais em paralelo com filtros de tamanhos diferentes (por exemplo 1x1, 3x3 e 5x5) e de uma camada de max-pooling de 3x3:

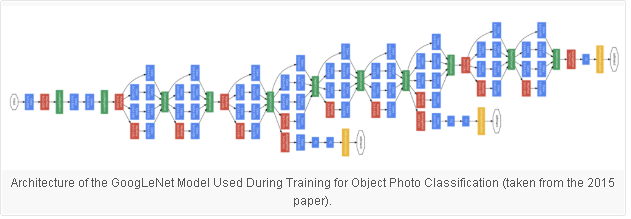


Um problema deste modulo básico de incepcao está a volumetria de filtros que aumenta muito rapidamente, sobre todo quando tem agrupamento de vários módulos de incepcao, o que acaba por estar um desafio em termo de capacidade computacional.

Para contornar isso, tem uso de camadas convolucionais com filtro 1x1, para reduzir o número de filtros, principalmente antes das camadas com filtro 3x3 e 5x5, e depois da camada de max pooling:



Uma segunda decisão de arquitetura impactante está a adição de várias saídas secundarias, treinadas para fazer predições, em diferentes pontos do modelo, usadas só durante o treinamento (retiradas depois), afim de aumentar o sinal de erro e assim limitar o impacto do *“vanishing gradientes problem”.*



O modelo está muito profundo com 22 camadas e a parte de extração das features finaliza com uma camada de average pooling de grande tamanho.

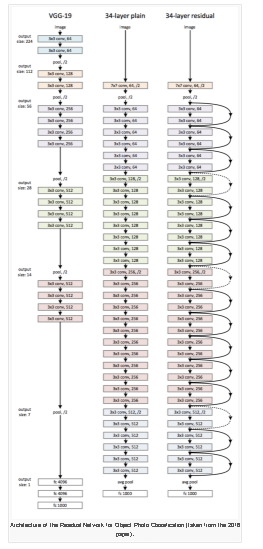
* Residual Netword (ResNet)

Este modelo foi proposto pelo Kaiming He, et al., em 2016, com o artigo intitulado “[Deep Residual Learning for Image Recognition](https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html).”

Este modelo fiz sucesso no concurso 2015 de ImageNet.

Ele está formado de mais de 152 camadas.

A inovação deste modelo está o uso de blocos residuais que usam conexões de atalho, que permitem de pular camadas.



O modelo de base foi um um VGG-19 deep neural convolutional network, com filtros pequenos 3x3, agrupando as camadas de convolucao em blocos sem camada de pooling entre elas.

A parte de detecção de feature finaliza por um average pooling, que precede as camadas densamente conectadas cujo a camada de saída usa Softmax como função de ativação.

* + Estudo das diferentes arquiteturas de transfer learning

Limitações do projeto estão um data set de exemplos relativamente pequeno e uma capacidade de processamento limitada.

Elas iriam impedir de realizar um treinamento de um modelo CNN eficiente, se não existia o recurso muito rico de transfer learning.

O transfer learning permite de recuperar modelos já prontos, com a arquitetura das camadas e os pesos já treinados de cada unidade destas camadas.

Tem uma grande flexibilidade no uso destes modelos pre-treinados:

- Se pode usar eles diretamente, sem ajuste nenhum, para resolver um problema de visão computacional.

- Se pode integrar eles a um modelo específico, por exemplo retirando a parte de classificação (camadas densamente conectadas) e usando ele como extractor de features que irão alimentar um modelo de classificação específico.

- Se pode realizar um tunning fino de uma parte do modelo pre-treinado, descongelando o treinamento de algumas camadas/unidades do modelo.

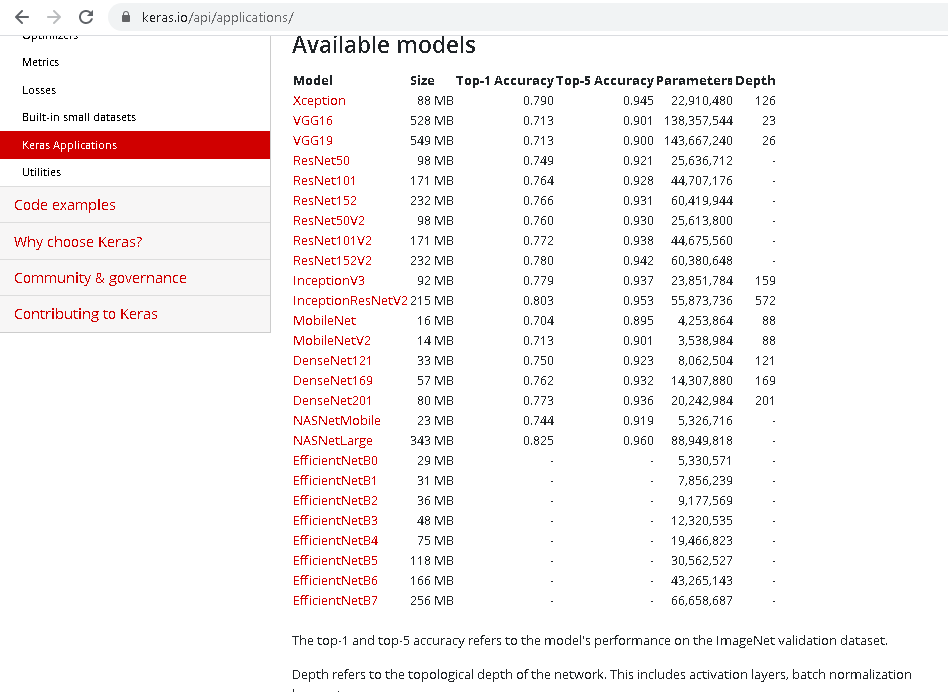
A libraria de rede neuronal Keras, escrita por François Chollet, funcionando como um front end para simplificar a codificação de rede neural usando backend como TensorFlow da Google, permite o transfer learning de vários modelos, incluindo os três últimos estudados encima (VGG, GoogLeNet/Inception e ResNet).

Eles estão acessíveis via [Applications API](https://keras.io/applications/), que deixa opcional a recuperação ou não dos pesos do modelo baixado, das camadas densamente conectadas da classificação.

Esta API contempla também métodos, adequados a cada tipo de modelo baixado, para preparar a entrada no modelo dos dados das imagens (ajuste das dimensões, do valor dos pixels...).

Desconsiderando a parte de classificação do modelo, está também possível de mudar as dimensões da camada de entrada (uso de tamanho de foto diferente de o usado no treinamento do modelo baixado).

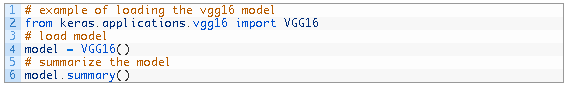
O número de classes da camada de saída pode também estar configurado para estar diferente do número de classe usado durante o treinamento do modelo baixado.

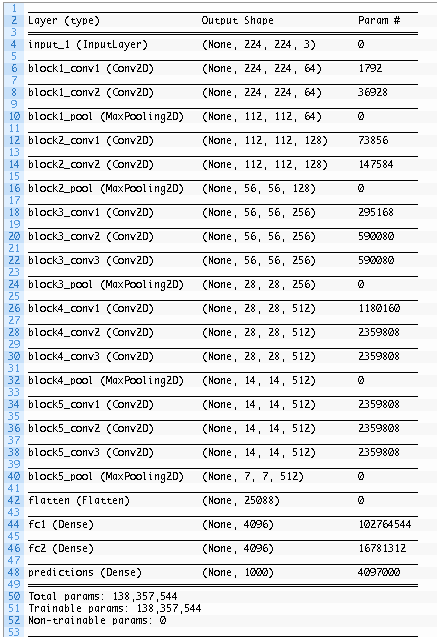


Fonte: <https://keras.io/api/applications/>

* Exemplo de baixa, com Keiras, do modelo VGG16 pre-treinado:

Código para baixar o modelo com os pesos do treinamento e para apresentar a estrutura do modelo:





* Avaliação das diferentes arquiteturas CNN para a classificação de imagem.

Se testou diferentes modelos, baixados já treinados com Keiras, sobre uma classificação binaria de pastas pessoais de Cachorro/Gato:

Foram testados os modelos:

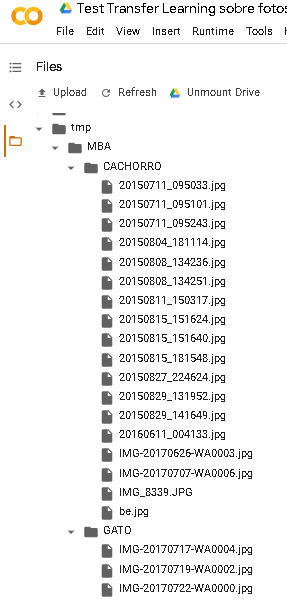
- VGG16

- INCEPTION\_V3

- RESNET50

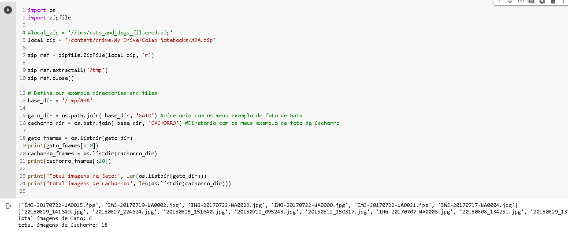
- MOBILE\_NETV2

* + As pastas de imagens pessoais de cachorro e de gato foram subidas no Drive pessoal, que foi montado no Collab Google:

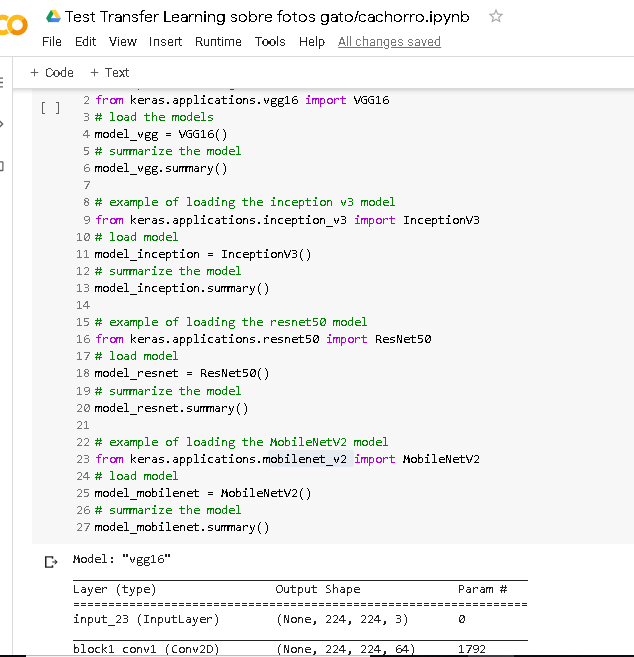


* + Se codificou uma classificação de cada uma destas fotografias pessoais, com os 4 modelos pre-treinados baixados, indicando para cada modelo as 3 classes preditas de maior probabilidade:

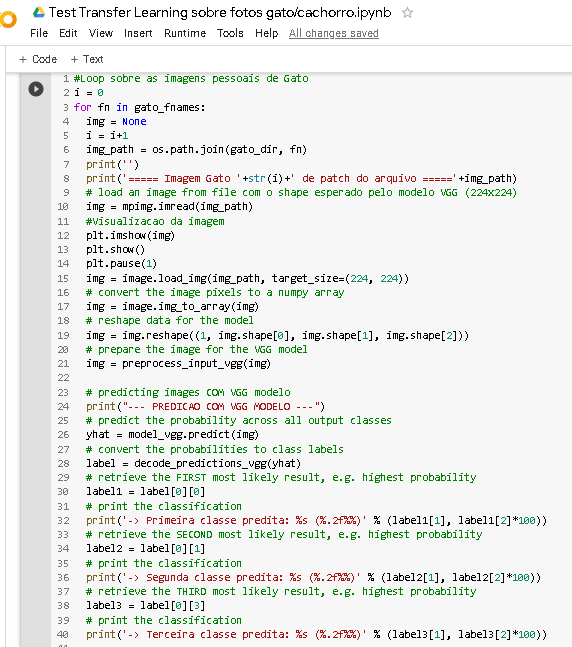
- Organização das fotografias:



- Baixa dos modelos pre-treinados:



- Loop sobre cada fotografia das pastas de gato e de cachorro, para predizer a classificação de cada uma com os 4 modelos, apresentando as 3 classes de maior probabilidade de cada modelo:

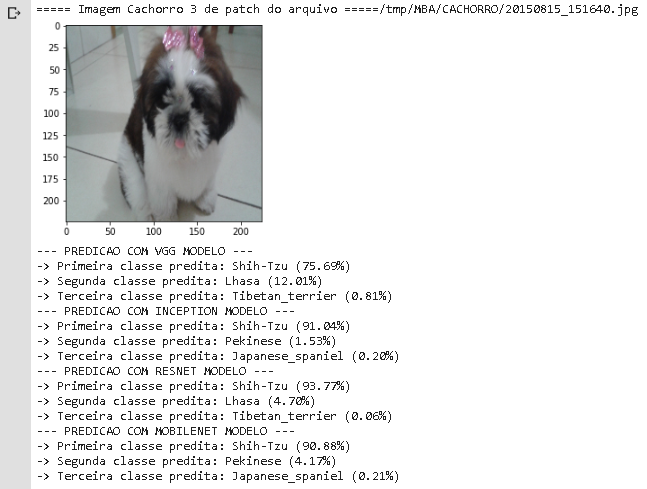


* Evidência da solução:

- Exemplo de classificações de imagem de gato com os 4 modelos pre-treinados baixados:



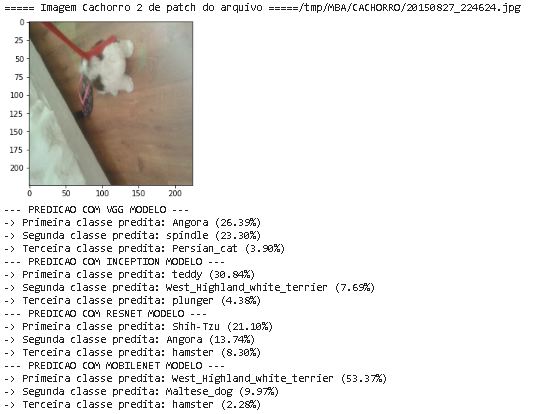
- Exemplo de classificações de imagem de cachorro com os 4 modelos pre-treinados baixados:



### Lições aprendidas

* + O transfert Learning será indispensável para chegar a uma solução de classificação de fotos de boa precisão.
  + Grande facilidade de uso dos modelos pre-treinados.
  + Eles apresentam uma boa precisão de predição da classificação sobre a amostragem de fotos pessoais.
  + Para fotografias mais difíceis (cachorro/gato meio escondido ou representando uma pequena parte da fotografia), alguns modelos estão se saindo melhor, mas não estão sempre os mesmos.
    - * Talvez que o uso de vários modelos pre-treinados, calculando uma média da probabilidade de cada modelo, permitiria de chegar a uma melhor classificação (mas precisa verificar se tem como relacionar o label dos diferentes modelos, correspondendo a uma mesma classe).

Exemplo:

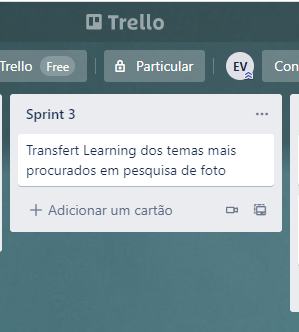


## Sprint 3



### Solução

* Evidência do planejamento:



* Evidência da execução de cada requisito:
* Baixa com a ajuda de Google Imagem de um conjunto de 750 fotos apresentando um ou vários temas (=classes) dentro de uma lista de quinze classes comuns de pesquisa de fotos pessoais, selecionadas arbitrariamente:
* Arvore
* Ave
* Cachorro
* Caminhao
* Carro
* Casa
* Cavalo
* Gato
* Mar
* Montanha
* Ponte
* Praia
* Prédio
* Rio
* Sol
* Complexidade e demora do trabalho manual de labelisacao para garantir a indicação de cada classe pesquisada presente nestas fotos.

Se renomeou as imagens indicando as diferentes classes pesquisadas presentes, separando elas pelo caractere menos “-”.

Exemplos das fotos renomeadas com indicação das classes pesquisadas:

predio-ponte-rio-2.jpg



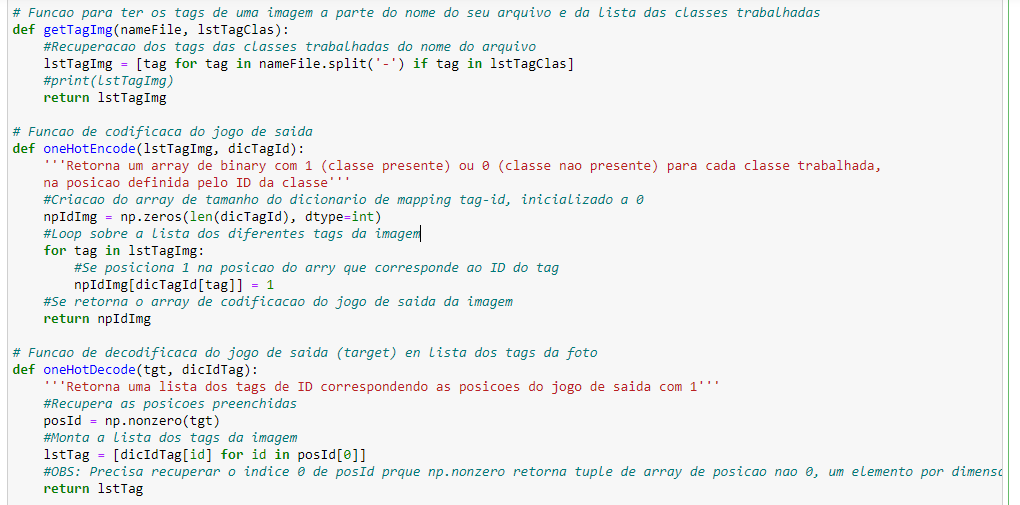
Carro-ponte-montanha-1

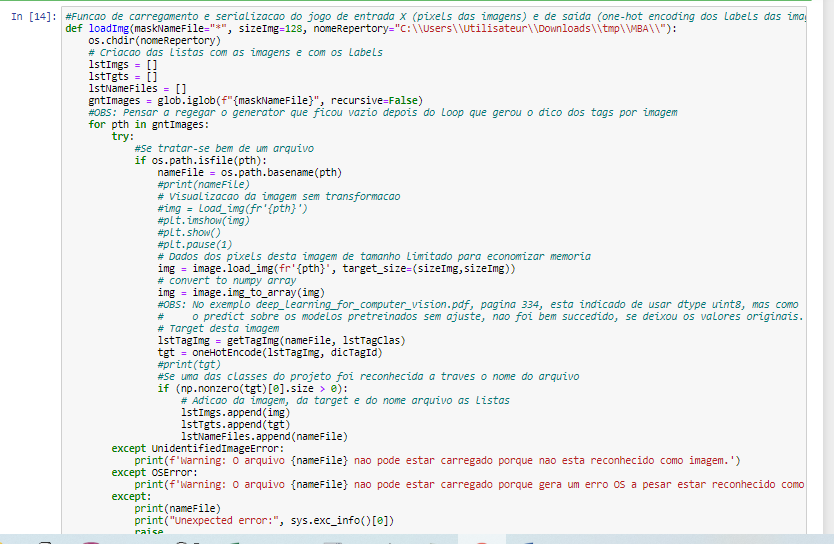


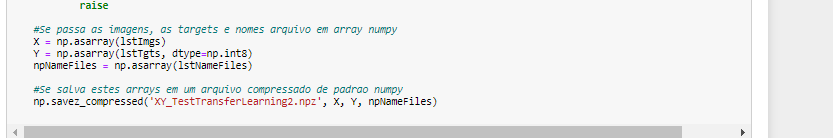
Camnhao-sol-arvore-2.jpg



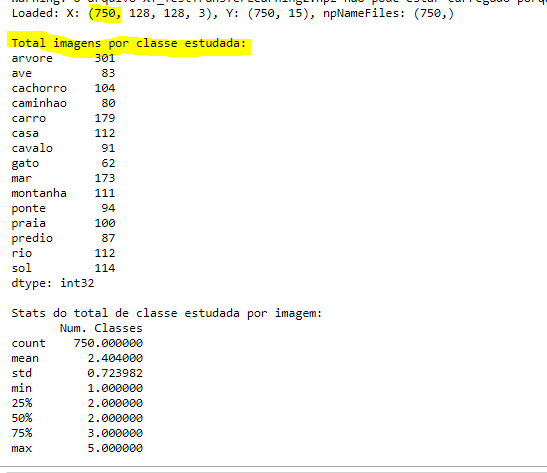
* + - * Este trabalho manual de labelizacao, a traves o nome do imagem, ficou com alguns erros (indicação do label errado) ou com ausências (por exemplo a foto encima apresenta também a classe carro pesquisada que acabou por não estar indicada no nome do arquivo).
      * Isso representa um ruído do jogo cujo o impacto no treinamento do modelo vai estar diretamente relacionado ao tamanho do jogo.
* Desenvolvimento de várias funções para centralizar os tratamentos repetitivos de carregamento, preparação, codificação e decodificação das imagens:







* + - * A função loadImg recebe em entrada a máscara do nome das fotografias a pesquisar, o tamanho para redimensionar as fotos e a pasta a consultar,
      * Para cada imagem achada, tem carregamento em memoria redimensionando a imagem, transformação em array, recuperação e codificação das classes indicadas e conservação em uma lista se uma das classes faz parte da lista das 15 pesquisadas.
      * Ao final se monta arrays numpy com o jogo de entrada e de saída que se serializa para remontar o jogo a vontade mais rapidamente, via deserializacao.
  + Estatísticas do jogo de imagem montado:



* + - * 750 imagens, com em média entre 2 e 3 classes pesquisadas.
      * A classe com maior volumetria está a “arvore”, a com menos está “gato”.
* Aplicação de diferentes modelos pre-treinados, sem ajuste, para predir os labels das fotografias deste jogo:



* + - * Criação de um objeto para cada um dos 4 modelos pre-treinados baixados: VGG16, Inception, RestNet50 e MobileNetV2.
      * Se mutualiza em uma função PredicaoModel o código comum para realizar a predicao de todo o jogo das 750 imagens. Ela recebe em entrada o modelo, o seu nome, a função de preparação da imagem ao formato de entrada esperado pelo modelo, a função de decodificação da predição e o jogo de entrada.
      * Ela aplica o preparação da imagem, faz a predição, decodifica a predição e retorna a lista dos 5 tags com o maior percentual de probabilidade.
* Evidência da solução:









* + Estas predições tem o problema de apontar uma única classe principal, a pesar que as fotos apresentam vários temas a identificar (a probabilidade de presença de um tema não deveria afetar a probabilidade dos outros).
  + Isso está ligado a função de ativação final, softmax, do classificador destes modelos pre-treinados, visto que o objetivo inicial deles estava de resolver um problema de classificação multi-classes mono label (as fotos a classificar tenham objeto(s) de uma única classe a reconhecer dentro das 1000 classes do concurso ImageNet).

### Lições aprendidas

* A parte de classificação, dos modelos pre-treinados (formadas normalmente de camadas densamente conectadas), permite só de selecionar a classe principal de uma imagem e não estão apropriada para uma seleção de imagem considerando várias classes de objeto a pesquisar (por exemplo, as fotos com um fundo formado de uma praia e de uma montanha).
* Para este tipo de pesquisa, que corresponde a um problema “multi-label classification”, se pode aproveitar dos modelos pre-treinados para selecionar as features das imagens, adicionando novas camadas densamente conectadas, que precisam passar por um treinamento para classificar as classes de pesquisa desejadas.
* Devido a esta necessidade de treinamento desta parte de classificação dos modelos, volta a estar crucial um banco de imagem corretamente labelisado.
* Formando este banco de imagem a parte de pesquisas com Google Imagem, um ponto de atenção esta os diretos autorais destas fotos para treinar os modelos.
* Além disso, as classes destes modelos pre-treinados sobre o jogo de dados ImageNet, estão demais precisas considerando as classes da nossa pesquisa (por exemplo dezenas de classes para a classe cachorro).

## Sprint 4



### Solução

* Evidência do planejamento:



* Evidência da execução de cada requisito:

Se usou o modelo pre-treinado VGG16 e se comparou a sua performance (capacidade de generalização sobre novas fotos) com diferentes técnicas de otimização e contra um modelo base de arquitetura parecida, montado “from scratch”.

* A métrica de performance foi a F1, recuperando o código disponibilizado pela Kaggle para o concurso “Planet: Understanding the Amazon from Space” de 2017, afim de funcionar com Keyras.

F1 = 2 × (precision × recall) / (precision + recall)

Com precision = true positives / (true positives + false positives) e recall = true positives / (true positives + false negatives).

Esta métrica está adequada para os jogos formados de várias classes desbalanceados, o que está o casso do jogo do nosso projeto com a classe arvore presente quase 5 vezes mais que a classe gato.

Para verificar a confiabilidade da implementação desta métrica, se comparou o resultado dela contra o da função F1 da biblioteca Scikit Learn, comparando a target real dos jogos de treinamento e de teste (repartição 70/30) contra uma predição forçada a 1 para todas as classes de todas as amostragens:

All Ones (sklearn): train=0.2741, test=0.2714

All Ones (keras): train=0.2741, test=0.2714

* Para melhor entendimento do resultado de cada treinamento, se monta um gráfico de evolução, por época, da função de perda do modelo (“cross-entropy loss”) e da métrica F1, comparando o jogo de treinamento contra o jogo de teste.
* Para ter uma ideia do ganho de performance, iniciando o treinamento a parte de um modelo VGG16 pre-treinado, se treinou primeiro um modelo base de arquitetura parecida:

Model: "sequential\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_6 (Conv2D) (None, 128, 128, 32) 896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_7 (Conv2D) (None, 128, 128, 32) 9248

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_3 (MaxPooling2 (None, 64, 64, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_4 (Dropout) (None, 64, 64, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_8 (Conv2D) (None, 64, 64, 64) 18496

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_9 (Conv2D) (None, 64, 64, 64) 36928

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_4 (MaxPooling2 (None, 32, 32, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_5 (Dropout) (None, 32, 32, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_10 (Conv2D) (None, 32, 32, 128) 73856

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_11 (Conv2D) (None, 32, 32, 128) 147584

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_5 (MaxPooling2 (None, 16, 16, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_6 (Dropout) (None, 16, 16, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_1 (Flatten) (None, 32768) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 128) 4194432

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_7 (Dropout) (None, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_3 (Dense) (None, 15) 1935

=================================================================

Total params: 4,483,375

Trainable params: 4,483,375

Non-trainable params: 0

Se aplicou dois métodos de otimização, para limitar o risco de overfitting particularmente presente usando um jogo de dados tanto reduzido (ficou 525 imagens para o treinamento e 225 para o teste):

* + Drop out:

Esta técnica consiste a zerar aleatoriamente, a cada época, os pesos de um percentual definido de neurônios, afim de obrigar o processo de treinamento a achar outros caminhos para otimizar a função de perda e assim limitar o risco de overfitting.

* + - Configurado com 20% na parte convolucional e com 50% na parte densamente conectada (sem aplicação da técnica de Data Augmentation).

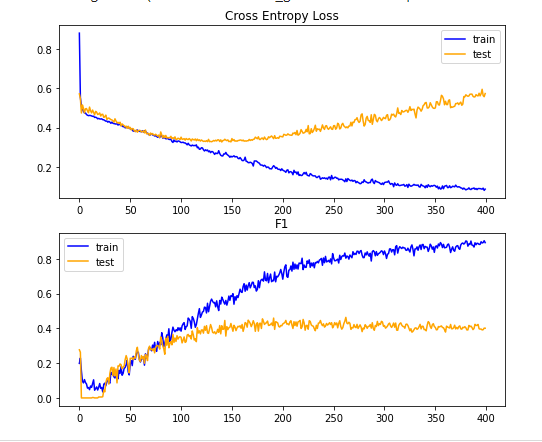
Resultado:

Epoch 400/400

5/5 [==============================] - 1s 166ms/step - loss: 0.0905 - fBetaKeyras: 0.8898 - val\_loss: 0.5741 - val\_fBetaKeyras: 0.4006

2/2 [==============================] - 0s 39ms/step - loss: 0.5741 - fBetaKeyras: 0.4006

> loss=0.574, fbeta=0.401



* + - Configurado com 10% na parte convolucional e com 20% na parte densamente conectada (com aplicação da técnica de Data Augmentation).

OBS: O drop out foi reduzido porque parecia impedir de ter um ganho de performance com a técnica de Data Augmentation,

* + Data Augmentation:

Esta técnica consiste a aplicar aleatoriamente uma serie de transformações das imagens, em função de parâmetros predefinidos, afim de submeter o modelo a visões diferentes de mesmos objetos, que poderiam acontecer na vida real, e assim melhorar a sua capacidade de generalização.

Esta técnica está particularmente adequada para melhorar a performance com um jogo de dados reduzido.

Se aplicou:

dictArgDataAugm = {'rotation\_range':40, 'width\_shift\_range':0.2, 'height\_shift\_range':0.2, 'shear\_range':0.2, \

'zoom\_range':0.2, 'horizontal\_flip':True, 'fill\_mode':'nearest'}

dataGenTrain = ImageDataGenerator(rescale=1.0/255.0, \*\*dictArgDataAugm)

- Rotação até 40%.

- Deslocamentos horizontal e vertical até 20%.

- Zoom até 20%.

- Translação horizontal.

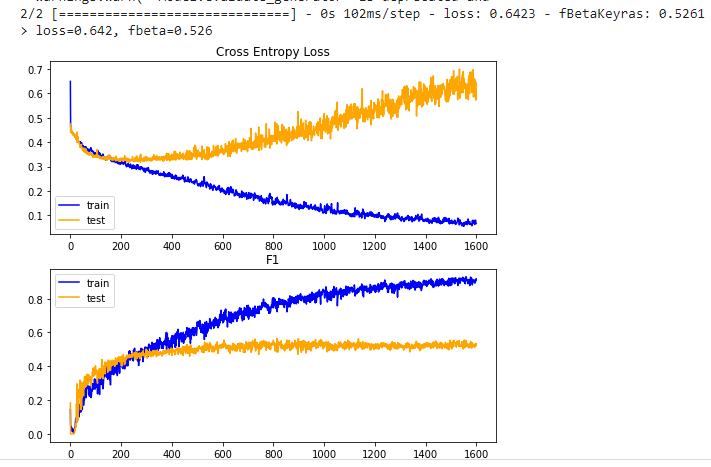
- Distorção até 20%.

OBS:

- Se usou a classe ImageDataGenerator de Keyras para aplicar estas transformações e para normalizar o valor dos pixels (dividindo por 255 os 3 canais de cor). Este objeto funciona como um iterator, para subir em memoria o próximo lote de imagens (de quantidade configurada pelo parâmetro batch size), a processar pelo modelo, ajustando ao final do lote os pesos com a etapa de backpropagation.

- Sobre o jogo de teste, se aplicou só a normalização.

Resultado, aplicando a mais o drop out reduzido:



* + - * A performance, medida com a métrica F1, melhorou passando de 0.40 a 0.53.
      * Se aumentou o número de época, para deixar o modelo estar confrontado a mais variedades de transformações das imagens, mas em torno de 200 épocas foram suficientes para o modelo atingir o seu melhor resultado (a parte de lá, ele começou a entrar em overfitting, se traduzindo pela augmentação do “cross entropy loss” do jogo de teste e da diminuição do “cross entropy loss” do jogo de treinamento.
* O modelo VGG16 pre-treinado foi baixado, dropando a parte de classificação (conjunto de camadas densamente conectadas específicas as classes do jogo Imagenet), preservando os pesos da parte de “extraction feature” (conjunto de camadas de convolucao e de pooling) e marcando as camadas para não estar atualizadas durante o treinamento (afim de não perder os pesos baixados):

*# Se carrega o modelo da API Keyras*

*model = VGG16(include\_top=False, input\_shape=inputShape, weights='imagenet')*

*# Se indica que as camadas (ficou só as da parte convolucional de seleção das features) não podem estar treinadas (isso para congelar os pesos recuperados do modelo VGG16)*

*for camada in model.layers:*

*camada.trainable = False*

* Se adiciona a parte de classificação, formada de:

- Uma camada “flatten” para receber em entrada todas as features extraídas pela parte convolucional, baixada já pre-treinado.

- Uma camada densamente conectada de 128 neurônios, com a função de ativação “relu”.

- Uma camada de drop out (incialmente de 20% e depois reduzido a 10% aplicando a mais a técnica de data augementation).

- A camada de saída com 15 neurônios **e a função de ativação sigmoid**. Cada neurônio vai retornar a probabilidade de uma das classes pesquisadas estar presente na foto.  
Obs: Não se usa a função softmax, usada nativamente nestes modelos pre-treinados com o jogo Imagenet, porque ela retornaria a probabilidade de cada classe estar a classe principal (no jogo Imagenet as fotos apresentam uma única classe da lista pesquisada, diferente do nosso jogo pelo qual as fotos podem apresentar várias classes a identificar, independente uma da outra).

flatten\_3 (Flatten) (None, 8192) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_6 (Dense) (None, 128) 1048704

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_9 (Dropout) (None, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_7 (Dense) (None, 15) 1935

=================================================================

Total params: 15,765,327

Trainable params: 1,050,639

Non-trainable params: 14,714,688

* Como o modelo pre-treinado está esperando um preprocessamento dos pixels, se monta um data generator aplicando a centralização com os valores aplicados durante o concurso Imagenet (a aplicar tanto sobre o jogo de treinamento que sobre o jogo de test):

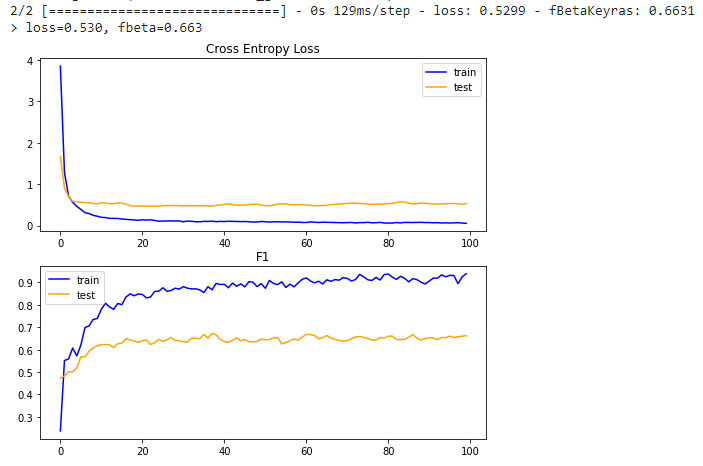
*“# Objeto Generator de Keyras com preprocessamento usando os valores centralizacao do jogo de dados de IMAGENET*

*dataGen = ImageDataGenerator(featurewise\_center=True)*

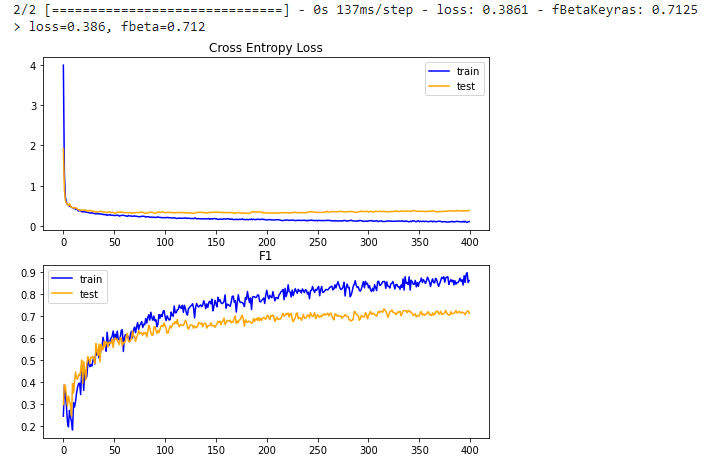
*# Definicao da media do jogo de dados Imagenet para centralizacao das fotos como esperado pelo modelo VGG16 treinado sobre este jogo*

*dataGen.mean = [123.68, 116.779, 103.939]”*

* Foram realizados diferentes treinamentos, aplicando diferentes técnicas de otimização da performance:
  + Só drop out (de 20% depois da camada densamente conectada):



* + - * Sem precisar de outra técnica de otimização, a performance fica já bem melhor que a do modelo de base com aplicação do data augmentation (F1 do jogo de test de 0,66 contra 0,52).
* Data augmentation com drop out reduzido, usando as mesmas transformações que as aplicadas para o modelo base (ver [aqui](#DataAugmentation)).



* + - * A performance passou de 0,66 a 0,71.

OBS: O resultado deste treinamento foi serializado, para servir de base de início dos treinamentos com a técnica de fine tuning.

*‘# Se serializa o modelo*

*model.save('modelVGG16DataAugm.h5')*

*# Se deserializa o modelo*

*model = load\_model('modelVGG16DataAugm.h5', custom\_objects={'fBetaKeyras':fBetaKeyras})’*

* Fine tuning das últimas camadas da parte de “extraction feature”:

Se libera o ajuste dos pesos destas camadas afim de tentar deixar as features, extraídas por estas camadas, mais específicas para as 15 classes da nossa pesquisa:

OBS: Se inicia do modelo já previamente treinado com a nova parte de classificação, para que o sinal de erro não seja demais importante, senão teria risco, durante as primeiras épocas, de ajustar demais os pesos das camadas descongeladas e, por tanto, de perder em grande parte a aprendizagem inicial destas camadas.

*# Fine tuning das 3 ultimas camadas da parte feature selection com um learning rate muito pequeno*

*model.trainable = True*

*set\_trainable = False*

*for layer in model.layers:*

*if layer.name == 'block5\_conv1':*

*set\_trainable = True*

*if set\_trainable:*

*layer.trainable = True*

*else:*

*layer.trainable = False*

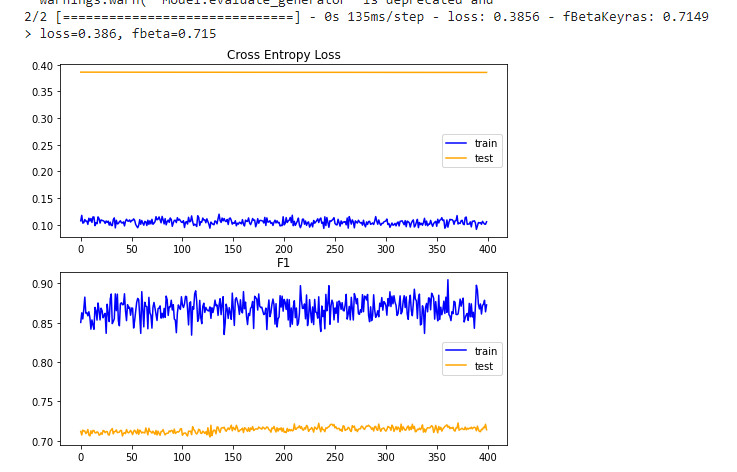
*# Compilacao do modelo*

*opt = SGD(learning\_rate=1e-5)*

*model.compile(optimizer=opt, loss='binary\_crossentropy', metrics=[fBetaKeyras])*

*Obs:* *'block5\_conv1' está o nome da camada antes da penúltima camada da parte de extracao de feature do modelo VGG16.*

Inicialmente se aplicou uma taxa de aprendizagem muito baixa (parâmetro learning\_rate do otimizador a 1e-5):



* + - * Não tive evolução da performance.

Se fiz variar depois este parâmetro learning rate para valores superiores, com 2 otimizadores:

- SGD (Stochastic gradient descente) usado para o treinamento do modelo base.

- RMS Prop (recomendado em vários estudos para o fine tunning).

Se tentou também realizar uma diminuição progressiva deste parâmetro learning rate, usando a técnica de callback que permite de consultar e eventualmente alterar parâmetros do modelo depois de cada época:

*# Definicao dos callbacks*

*lstCallbacks = [ReduceLROnPlateau(monitor='val\_loss', factor=0.2, patience=40),*

*ModelCheckpoint(filepath='modelVGG16DataAugmCallBack.h5', monitor='val\_fBetaKeyras', save\_best\_only=True)*

*]*

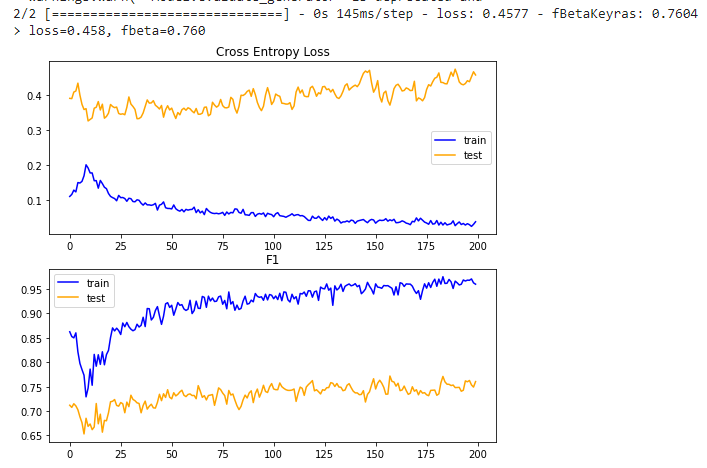
*# Treinamento (fit) do modelo*

*histTrain = model.fit(iterTrain, steps\_per\_epoch=len(iterTrain),*

*validation\_data=iterTest, validation\_steps=len(iterTest), epochs=100\*numMultStepEpoc, verbose=verbose, \*

*callbacks=lstCallbacks)*

* + - * Se ativou o callback Keyras *ReduceLROnPlateau* para dividir por 5 a taxa de aprendizagem, depois de 40 épocas sem melhora do “cross entropy loss” medido sobre o jogo de validação.
      * Se aproveitou para adicionar o callback *ModelCheckpoint* para serializar o melhor modelo, no decorrer das épocas, se baseando sobre a métrica F1 do jogo de validação.



* + - * Da para notar uma perda inicial de performance, durante as primeiras etapas, ligadas as modificações iniciais não muito precisas dos pesos das camadas convolucionais descongeladas.
      * O melhor resultado (F1 do jogo de teste passado a 0,76) foi obtido usando o mesmo otimizador (SGD) e a mesma taxa de aprendizagem (0,01) que os usados para treinar a parte de classificação sem fine tuning (que serve de modelo base para iniciar o fine tuning).
      * Este modelo foi serializado para realizar o treinamento final usando a totalidade do jogo de imagens (750).
* Treinamento final com o jogo de dados completo, usando o modelo serializado que apresentou a melhor métrica F1 com o jogo de test:

Se adicionou um callback para finalizar o treinamento mais cedo, quando não tem mais melhora do indicador F1, passado 30 épocas (pensando a configurar o parâmetro mode a ”max”, já que por default o modo de consulta da melhora está “min”).

*# Definicao dos callbacks*

*lstCallbacks = [ModelCheckpoint(filepath='modelVGG16DataAugmFineTuning2Full.h5', monitor='fBetaKeyras', save\_best\_only=True), \*

*EarlyStopping(monitor='fBetaKeyras', patience=30, mode='max', verbose=1)]*

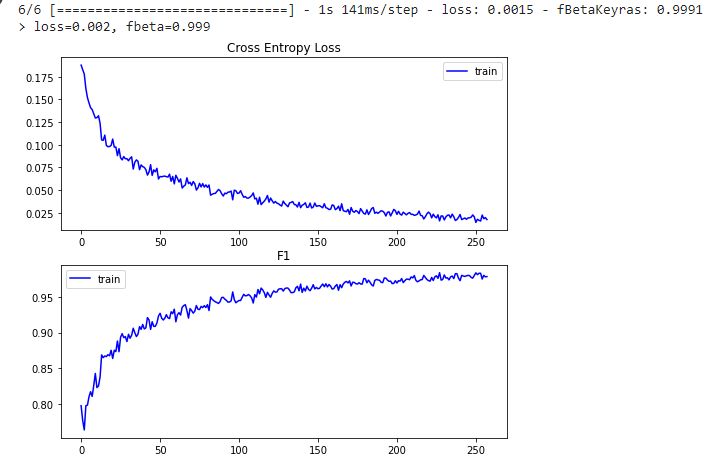
*# Treinamento (fit) do modelo SEM jogo de test*

*histTrain = model.fit(iterTrain, steps\_per\_epoch=len(iterTrain), \*

*epochs=100\*numMultStepEpoc, verbose=verbose, \*

*callbacks = lstCallbacks*

*)*

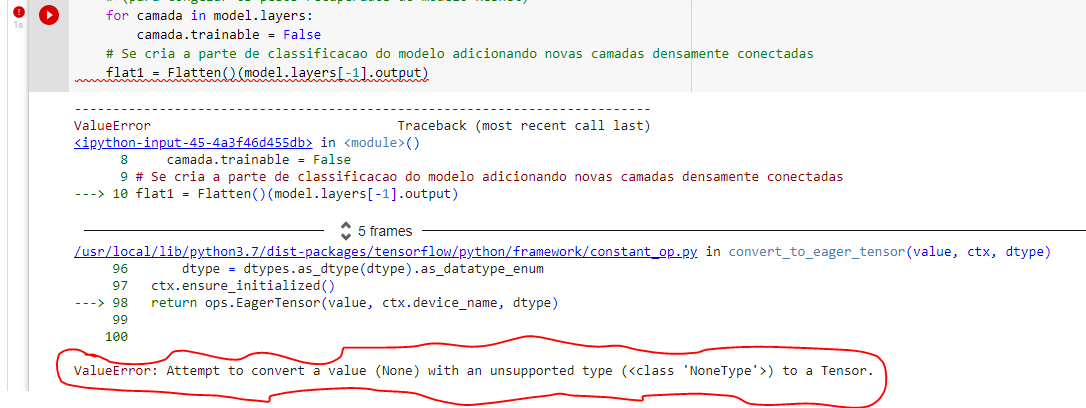


* + - * Se serializa este modelo final para aplicação depois sobre um novo jogo de fotos recuperadas de Internet.

# Se serializa o modelo

model.save('modelVGG16DataAugmFineTuning2Full2.h5')

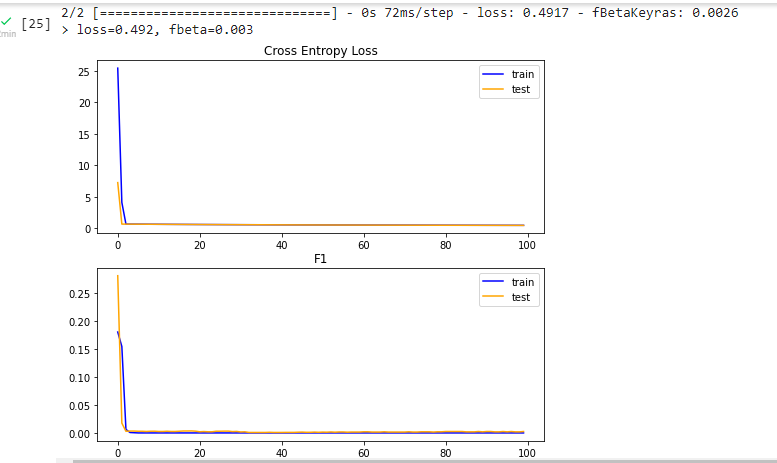
* + - * O score de 0,99 da métrica não pode estar considerado, já que mesmo com uso da técnica de data augmentation, tive overfitting devido ao jogo demais reduzido.
* Se tentou usar outros tipos de modelos pre-treinados, para realizar o transfer learning, mas não tive o mesmo sucesso que com o VGG16:
  + Com Resnet50 (arquitetura de blocos residuais):
    - * Tem um erro no momento de adicionar uma camada flatten a camada de saída do modelo, carregado sem a parte de classificação:



* + - * Não foi encontrado no Internet, como contornar este erro.

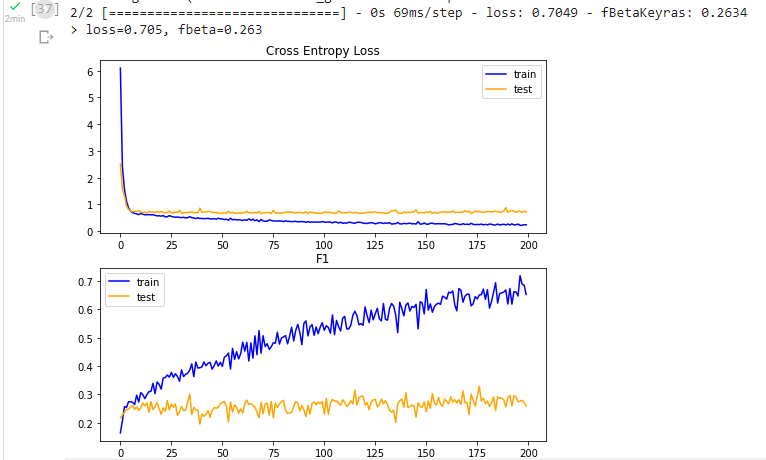
« ValueError: Attempt to convert a value (None) with an unsupported type (<class 'NoneType'>) to a Tensor. »

* + Com InceptionV3 (arquitetura incepção com um modelo específico chamado GoogLeNet)
    - * A parte de classificação adicionada foi idêntica a usada com VGG16: (camadas flatten / densamente conectada de 128 neurônios / drop out / camada saída de 15 neurônios).
      * Compilando o modelo, usando a mesma configuração do otimizador que a usada para o transfer learning com VGG16 (SGD com learning rate de 0,01), não tem aprendizagem nenhum no jogo de teste nem no jogo de treinamento (métrica F1 chegando em poucas épocas próxima de 0):

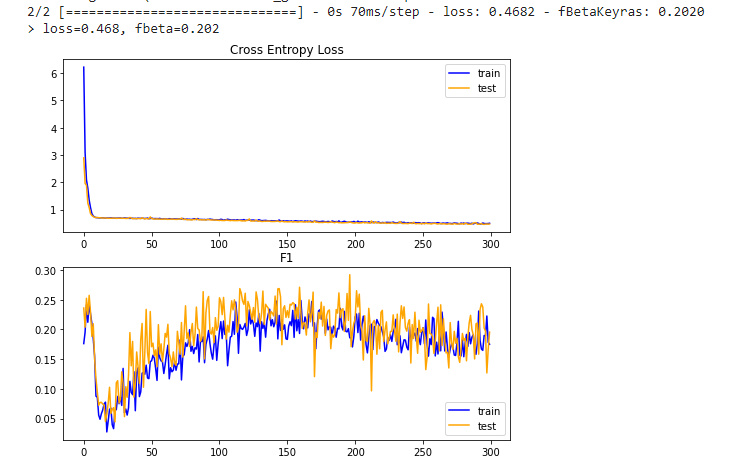


* + - * Usando o otimizar RMSProp com uma taxa de aprendizagem pequena de 0,0001, tem progresso da aprendizagem do jogo de treinamento, mas o jogo de teste fica com uma métrica F1 inferior a metade de a observada com VGG16:

Sem a técnica de Data Augmentation e um drop out de 0,2 depois da camada nova adicionada densamente conectada:



Com a técnica de Data Augmentation (mesmas transformações das imagens que as aplicadas para o VGG16) e um drop out de 0,1 depois da camada nova adicionada, densamente conectada:



* Evidência da solução:
* Se deserializa o modelo, que resultou do treinamento final com todo o jogo de dados, aplicando as técnicas e as configurações que apresentaram a melhor performance (drop out, data augmentation e fine tunning do modelo pre-treinado VGG16):

*#Modelo com jogo de treinamento completo, que treinou com as mesmas tecnicas que o modelo anterior usando callback para*

*# conservar os pesos da melhor epoca e para ter uma fim antecipada depois de 30 epocas sem melhora da metrica*

*model = load\_model('modelVGG16DataAugmFineTuning2Full.h5', custom\_objects={'fBetaKeyras':fBetaKeyras})*

* Se centraliza as fotos, considerando os parâmetros esperados pelo modelo VGG16:

*# Se centraliza manualmente os dados dos 3 canais de cor dos pixels*

*XPrep = X - [123.68, 116.779, 103.939]*

* Predição sobre o jogo usado para o treinamento final, conferindo a métrica F1 e realizando um loop sobre uma parte do jogo para consultar as fotos e comparar os labels da target (recuperadas do nome da foto) contra os da predição:



* Predição sobre um jogo totalmente novo de 60 imagens, conferindo a métrica F1 e realizando um loop sobre uma parte do jogo para consultar as fotos e comparar os labels da target (recuperadas do nome da foto) contra os da predição:

Estatísticas deste novo jogo:

Loaded: X: (60, 128, 128, 3), Y: (60, 15), npNameFiles: (60,)

Total imagens por classe estudada:

arvore 34

ave 8

cachorro 8

caminhao 10

carro 11

casa 8

cavalo 7

gato 7

mar 9

montanha 6

ponte 8

praia 7

predio 8

rio 9

sol 9

dtype: int32

Stats do total de classe estudada por imagem:

Num. Classes

count 60.000000

mean 2.483333

std 0.812856

min 1.000000

25% 2.000000

50% 2.000000

75% 3.000000

max 4.000000

Resultado:

Score F1: test=0.6382



* + - * Como esperado, a performance caiu bastante (F1 a 0,63).

### Lições aprendidas

* Existem técnicas poderosas para treinar um modelo de classificação multi-label, mesmo com um jogo de dados bem reduzido, permitindo de chegar a uma performance razoável em torno de 65% (VGG16 com drop out e data augmentation).
* Pode ter uma incompatibilidade na aplicação de diferentes técnicas durante um mesmo treinamento, necessitando de calibrar com precisão os parâmetros de cada técnica (caso do drop out demais elevado que impediu o ganho de performance com a técnica de data augmentation).
* Para chegar a uma melhor performance, precisaria de um jogo de dados mais consequente que o considerado (talvez 10 vezes superior).
* Isso representa uma limitação importante, já que toma um tempo importante de montar este tipo de jogo, fazendo cuidado a labelizaçao correta.
* Este tempo aumenta exponencialmente com o número de classes a pesquisar.
* O uso de modelos pre-treinados, de arquitetura mais sofisticada, não garante de ter melhores resultados com o transfer learning (nenhum dos treinamentos realizados com o modelo InceptionV3 tive uma performance que passou da metade das performances obtidas com o VGG16).

## Sprint 5



### Solução

* Evidência do planejamento:



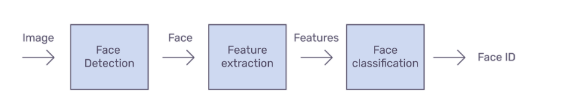
* Evidência da execução de cada requisito:

O objetivo está de pesquisar as fotografias, identificando rostos de pessoas do círculo pessoal.

Para isso, se desenvolveu uma solução em três etapas:  
1) Identificação das regiões das fotografias correspondendo a rostos de pessoas.

2) Extração das features das regiões de rosto.

3) Classificação dos rostos (associação ao label de uma pessoa reconhecida) em função destas features.



Foi usado um conjunto de fotografias pessoais, com membros da minha família, considerando 9 pessoas específicas a reconhecer pela solução.

Abaixo, um exemplo de foto, com 8 das pessoas a reconhecer (a nona pessoa sendo eu, o tirador da fotografia):



* **Etapa 1: Identificação das regiões das fotografias correspondendo a rostos de pessoas**

Uso do modelo já treinado MTCNN (Multi-Task Convolutional Neural Networks).

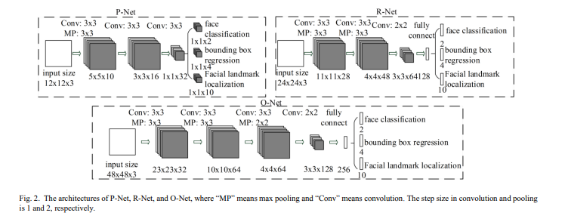
Se trata de uma rede neural que detecta os rostos e os marcos faciais nas imagens.

Ele foi publicado em 2016 por Zhang et al.

Está um dos modelos mais popular e preciso para identificação de rostos.

Ele está formado de três redes neurais (P-Net, R-Net, e O-Net) conectados em cascada.

MTCNN Structure // [Source](https://arxiv.org/ftp/arxiv/papers/1604/1604.02878.pdf)



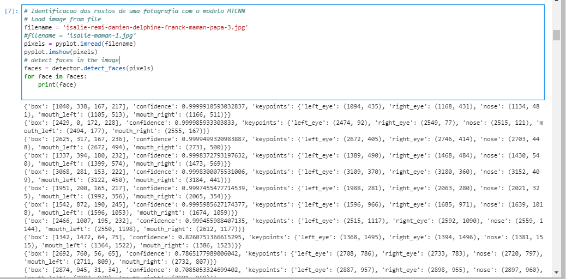
Mais informações sobre este modelo com [este link](https://towardsdatascience.com/how-does-a-face-detection-program-work-using-neural-networks-17896df8e6ff).

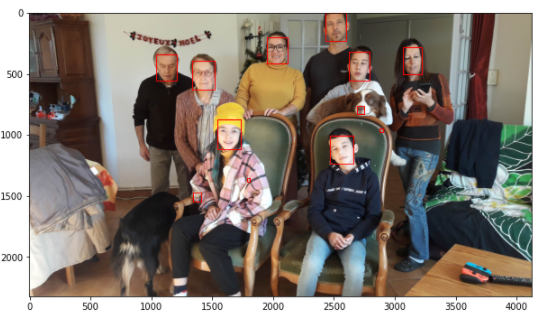
As áreas de rosto identificadas pelo modelo MTCNN foram conservados em novos arquivos de fotografia, para posterior treinamento do modelo de classificação.

Para verificação do resultado, se delimita nas fotos estas áreas, por um retângulo vermelho.

Estas áreas estão definidas a parte das coordenadas X/Y do ponto alto/direto, do cumprimento e da largura:

Exemplo de retorno com a fotografia exemplo:







Para limitar o risco de falso positivo (nesta fotografia quatro áreas estão identificadas erroneamente como rostos), se considerou um limite de probabilidade alta de 0,9.

Observação: para uma identificação correta dos rostos, precisou identificar a orientação nativa das fotografias, para eventualmente realizar uma translação delas antes de passar elas ao modelo.

* **Etapa 2: Extração das features das regiões dos rostos**

Uso do modelo já treinado Facenet.

Ele está uma rede neural particularmente adaptada para extrair as features dos rostos identificados em uma fotografia.

Foi publicado em 2015 pelos pesquisadores da Google Schroff et al.

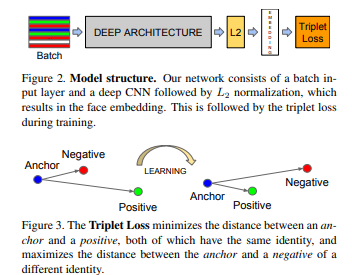
Em resume, ele retorna um vector de 128 números que representam as features mais importantes de um rosto (chamado em inglês “embedding”).

Fotografias de um mesmo rosto tem um retorno com valores bem próximos deste vector “embedding”.

Tem uso da técnica, chamada de “triplet loss”, com a qual se compara uma entrada de referência (rosto de uma pessoa) a uma entrada positiva (rosto de mesma pessoa) e a uma entrada negativa (rosto de pessoa diferente).

O treinamento consiste a minimizar a distância entre as features de rostos de mesma pessoa e a maximizar a distância entre as features de rostos de pessoas distintas.

Facenet structure [Source](https://arxiv.org/pdf/1503.03832.pdf) :



Estes vectores “embedding” serão usados como jogo de entrada do modelo de classificação dos rostos.

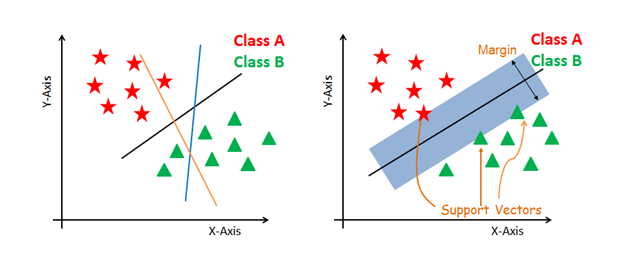
Observação: As imagens de rosto precisam passar por um preprocessamento de standardizacao, esperado pelo modelo Facenet (retirada da média e divisão pelo desvio standard), antes de estar passadas ao modelo.

* **Etapa 3: Classificação dos rostos em função do vector “embedding” das features**

Se treinou um modelo Machine Learning tradicional, de tipo SVM (Support Vector Machine), particularmente adaptado, já que o treinamento consiste a calcular, para cada pessoa a identificar, o limite maior da distância dos vectores “embedding” desta pessoa.

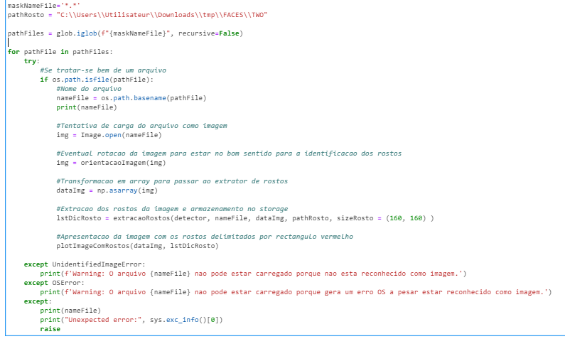
O princípio básico de funcionamento do SVM está a separação por hiperplanos dos pontos de cada classe, de tal maneira que os pontos de mesma classe ficam próximos e que os de classes diferentes ficam mais longe.

SVM principio [Source](https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python) :



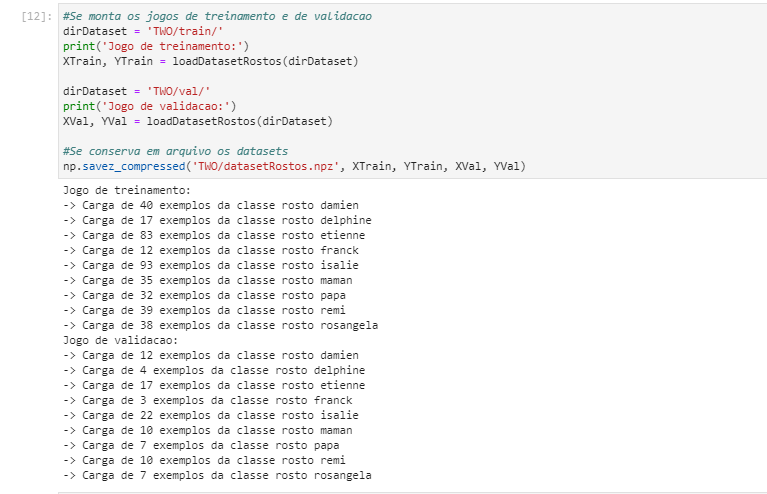
Para isso:

* + Se gerou, com o modelo MTCNN, os arquivos de rosto de centenas de fotografias pessoais, com presença de uma ou várias das nove pessoas que se pretende identificar.

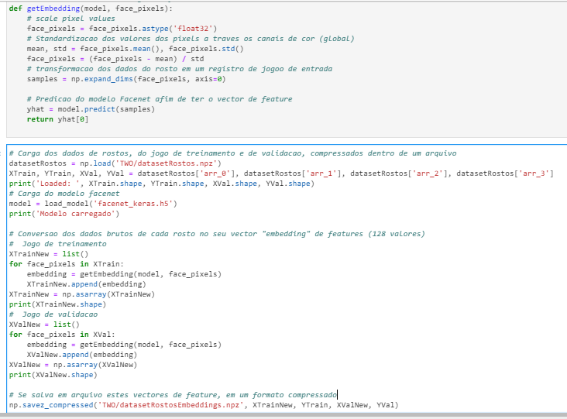


* + Elas foram classificadas manualmente em uma arborescência, separada entre treinamento e validação, com uma pasta por pessoa de nome o label da classe correspondendo a pessoa:

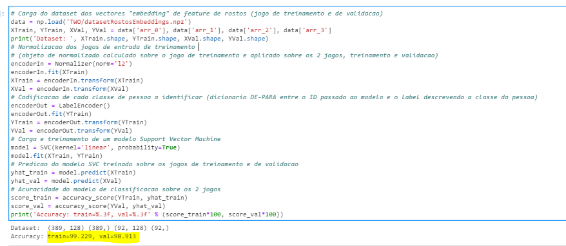




* + Para cada fotografia de rosto se gerou, e conservou em arquivo, um vector “embedding” de feature com o modelo Facenet.

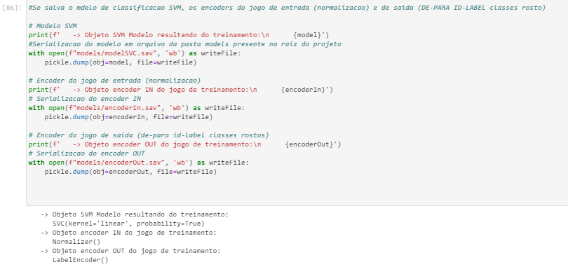


* + Estes vectores “embedding” foram usados para o treinamento de um modelo ML Support Vector Machine:



Observações:

* + - Os vectores “embedding” passaram por uma normalização linear afim de facilitar o treinamento, baseado sobre o cálculo de distancias entre os vectores.
    - Mesmo com relativamente poucas fotos de rosto por pessoa, se chega a uma acuracidade muito boa de quase 99% sobre o jogo de validação.
  + Os objetos necessários para novas predições (modelo SVC treinado, codificadores de normalização linear do jogo de entrada e de de-para ID-Label do jogo de saída) foram conservados em arquivos para posterior reuso:

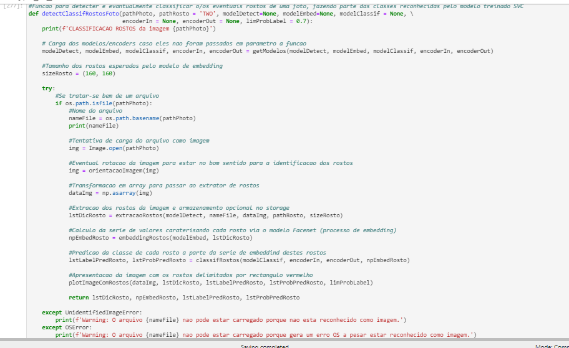


* Evidência da solução:
* Para confirmar a acuracidade do modelo de classificação, se seleciona e se mostra de maneira aleatória rostos do jogo de validação, indicando a classe real da pessoa e a classe predita assim que a probabilidade:



Observações:

* Em presença de um acessório não muito comum nas amostragens de rosto do jogo de treinamento de uma classe (para a classe me representando, a presença da máscara ou de óculos), a probabilidade cai (de mais de 90% para menos de 70%), mas não impede de selecionar a classe correta.
* Mesmo para as classes com poucas amostragens no jogo de treinamento (como a da minha irmã Delphine com 17 amostragens), a classificação está quase sempre correta, mas com uma probabilidade menor que a das outras classes treinadas com o dobro de amostragens de treinamento.
* Se montou um pipeline, para apresentar diretamente, em uma fotografia passada em parâmetro, os diferentes rostos presentes, labelisando os rostos reconhecidos indicando a probabilidade da predição:



Observação: Se considerou um limite de probabilidade de 0,7 para labelisar os rostos com a classe predita.

Exemplo com uma fotografia, apresentando só rostos de pessoas consideradas durante o treinamento do modelo de classificacao SVM:



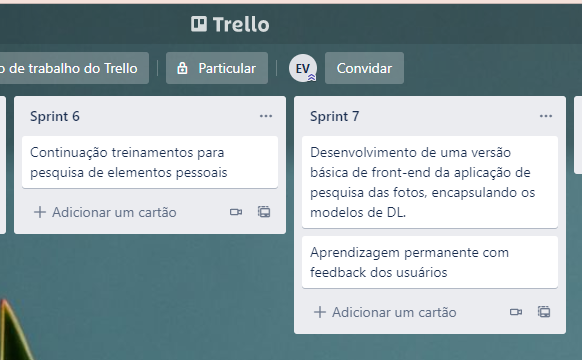
### Lições aprendidas

* A identificação e a classificação de rostos em uma fotografia são finalmente tarefas mais simples que a tarefa de classificação de elementos comuns, estudada durante os sprints 3 e 4.
* Além disso, se chega a uma acuracidade elevada, com muito poucas amostragens de cada classe.
* Esta relativa simplicidade e grande acuracidade vem da disponibilidade de modelos Deep Learning pre-treinados, usados sem necessidade de tuning, para as atividades:
  + De identificação das áreas de uma fotografia correspondendo a rostos de pessoa.
    - * Uso do modelo MTCNN (Multi-Task Cascaded Convolutional Neural Network), retornando para cada rosto as coordenadas da área do rosto na fotografia e de alguns pontos do rosto como os olhos.
  + De montagem das features das áreas de rosto.
    - * Uso do modelo Facenet, retornando para cada rosto um vector “embedding” de 128 valores caracterizando as features de rosto de cada pessoa, de distância próxima para os rostos de mesma pessoa e de distância maior para os rostos de pessoas diferentes.
* A última atividade de classificação dos rostos, considerando uma lista de classes de pessoas conhecidas, única etapa precisando de um treinamento específico, acaba por estar simple, usando um jogo de entrada do treinamento formado dos vectores “embedings”, das features de rostos, gerados pelo modelo Facenet.

## Sprint 6

### 2.6.1 Solução

* Evidência do planejamento:



Visto que, durante o Sprint 5, foi possível construir um modelo de reconhecimento de pessoas, já com uma boa acuracidade, se decidiu de aproveitar do Sprint 6 para já iniciar o trabalho de desenvolvimento de uma aplicação front-end simple, permitindo a um usuário comum a pesquisa das suas fotos, usando os modelos de pesquisa dos temas comuns (já pre-treinado) e de rostos (de treinamento a integrar na aplicação, já que a lista das pessoas a reconhecer irá variar por usuário).

* Evidência da execução de cada requisito:
  + Se pesquisou no Internet os diferentes módulos disponíveis para desenvolver uma GUI (Graphical User Interface) com a idioma Python.

A página wiki Python seguinte lista todas as alternativas disponíveis:

<https://wiki.python.org/moin/GuiProgramming>

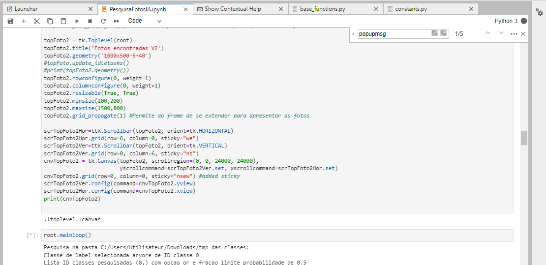
- O mais antigo (presentes desde os anos 90), e a princípio o mais simple para iniciar, está o Tkinter. Está a libraria padrão de Python para GUI e pelo tanto não precisa de uma instalação adicional. Ela encapsula a interpretação da linguagem Tcl para GUI.

- Outros módulos mais recentes, como PyQT e a sua ferramenta de design QTDesigner, wxPython, PyGTK, permitem de obter resultados mais profissional.

Mas a mais da complexidade maior de aprendizagem inicial, tem uma questão de licenciamento das aplicações desenvolvidas com estes outros módulos, que está mais delicada (por exemplo GPL version 3 para PyQT).

Pelo tanto se optou por usar Tkinter, aproveitando do sub-modulo ttk relativamente recente (2007), que permite a aplicação de ter uma aparência mais moderna (por exemplo, pela montagem de temas que podem se aplicar a um conjunto de objetos da aplicação, em vez de precisar ajustar os objetos um por um).

* Para facilitar a aprendizagem deste modulo Tkinter, se desenvolveu uma primeira versão da aplicação, com um mínimo de opções para lançar a pesquisa dos temas comuns, desde um note-book Jupyter, deixando as funções de pesquisa e de apresentação das fotografias em um modulo Python.





* + - Com esta primeira versão o usuário pode:

- Selecionar a pasta onde se encontram as fotografias a pesquisar (não foi previsto de recursividade para pesquisar nas sub-pastas).

- Selecionar os temas comuns das fotos a pesquisar, indicando se todos os temas devem estar presentes (opção “AND” da GUI) ou qualquer um (opção “OR”).

- Indicar o percentual do limite de probabilidade dos temas encontrados pelo modelo, a parte de qual se considera que o tema está bem presente.

- Indicar os parâmetros de apresentação das fotografias em uma janela separada, com o número de colunas e o tamanho (largura/altura) das fotos.

- Visualizar as fotografias achadas nesta janela, montada respeitando estes parâmetros, usando se necessário barras de rolagem vertical e horizontal. Encima das fotos, tem indicação do nome do arquivo e das classes pesquisadas encontradas com a probabilidade delas (só as de probabilidade superior ao limite configurado pelo usuário estão apresentadas).

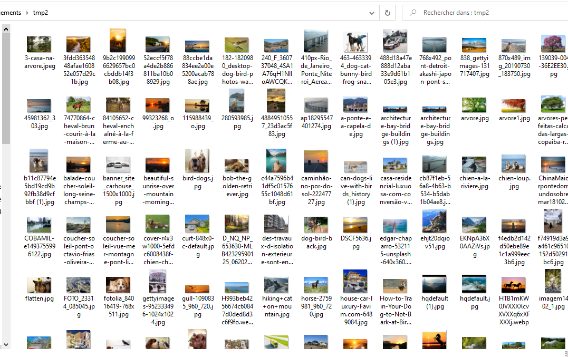
Observação: Tem presença de uma barra de progressão, prevista para o usuário poder acompanhar o andamento da pesquisa das fotos (que pode durar minutos em função do número de fotos da pasta selecionada), mas não foi possível deixar ela sincronizada. Pelo momento ela muda (passando totalmente verde), quando a pesquisa foi concluída...



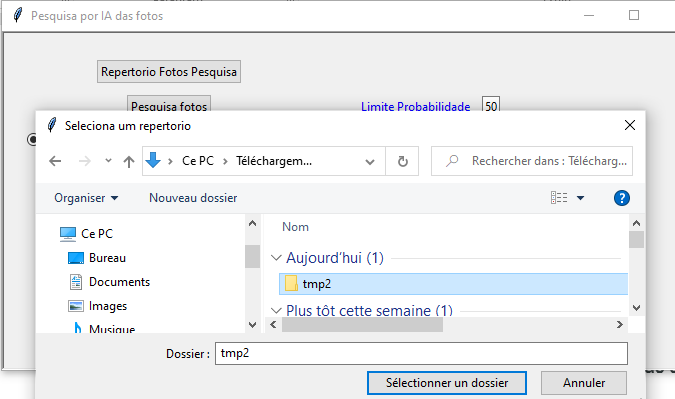
* Evidência da solução:
  + Recuperação de Internet, via a ferramenta de pesquisa de imagem da Google, de algumas centenas de fotos apresentando um ou vários dos 15 temas comuns do modelo treinado durante o sprint 4:

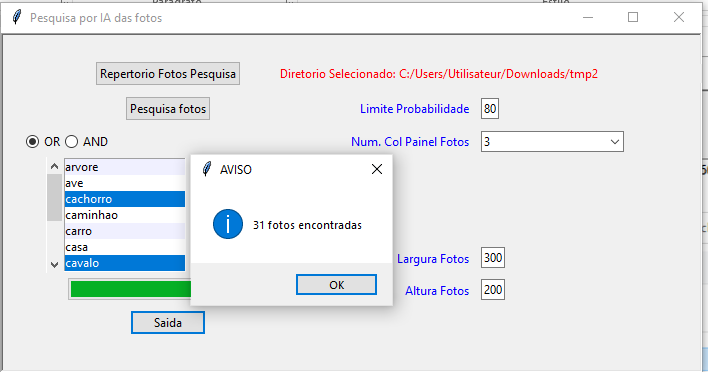
'arvore', 'ave', 'cachorro', 'caminhao', 'carro', 'casa', 'cavalo', 'gato', 'mar', 'montanha', 'ponte', 'praia', 'predio', 'rio', 'sol'

Esta vez não tive o trabalho de classificar estas fotos manualmente, já que elas irão servir a testar a pesquisa desde a aplicação front end e não a treinar ou a avaliar a acuracidade do modelo:



* Exemplos de pesquisas realizadas com esta primeira versão da aplicação, sobre o conjunto encima de fotos:
  + Fotos com presença de cachorro **OU** cavalo, com uma probabilidade superior a 80%, apresentando elas sobre 3 colunas com 300 pixels de largura e 200 de altura:



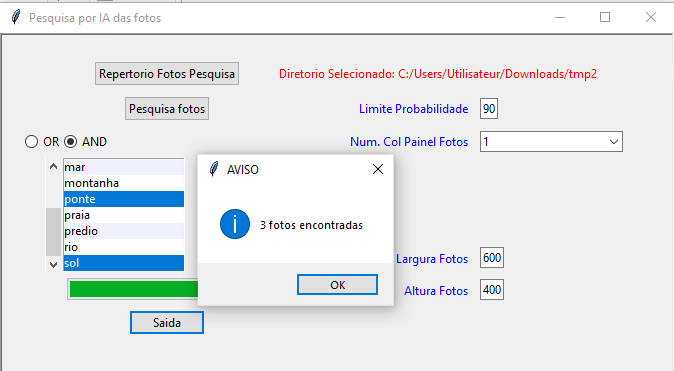


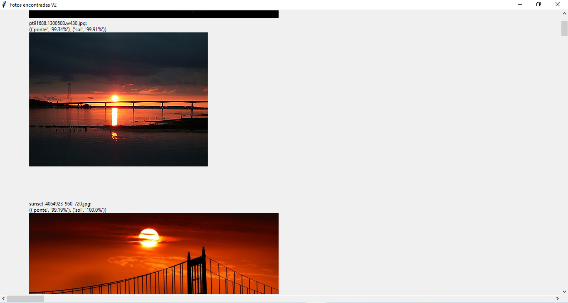


Observação: A pesar de haver configurado para considerar só as predições realizadas com uma probabilidade superior a 80%, tem alguns falsos positivos aparecendo. Se poderia retirar eles aumentando ainda mais a probabilidade, mas no detrimento de aumentar o número de falsos negativos (fotografias com os temas pesquisados mas não selecionados porque a probabilidade de predição destes temas não passou do limite configurado).

A fotografia com uma ponte e o mar, selecionada indevidamente com uma probabilidade de 83% da presença de um cachorro, atira a atenção já que aos nossos olhares humanos, tem nada na foto que poderia gerar esta confusão...

* + Fotos com presença de ponte **E** sol, com uma probabilidade superior a 90%, apresentando elas sobre 1 coluna com 600 pixels de largura e 400 de altura:





### 2.6.2 Lições aprendidas

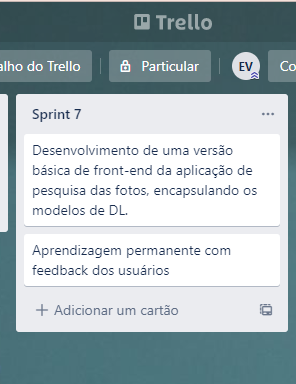
* Dificuldade de sincronizar o avanço de uma barre de status com o processamento backgroud de pesquisa de fotos, que, em função do número de fotos da pasta selecionada, pode passar de vários minutos...
* Se concretiza o que havia sido avaliado durante o treinamento do modelo, que tem seleção errada de fotos devido a imprecisão de predição do modelo. Para limitar isso, o usuário pode aumentar o limite de probabilidade de predição, mas gerando um outro erro de não apresentar as fotos que correspondem aos critérios de pesquisa.

A solução a este problema seria de voltar a treinar o modelo com muito mais amostragens.

## Sprint 7

### 2.7.1 Solução

* Evidência do planejamento:



Continuação do desenvolvimento da aplicação, implementando as funcionalidades de extração dos rostos, treinamento do modelo de reconhecimento de pessoas específicas, a parte dos rostos extraídos, e de pesquisa de fotos com estas pessoas, com a possibilidade de acoplar ela a pesquisa dos temas comuns.

* Evidência da execução de cada requisito:

Diferente da pesquisa dos temas comuns, pela qual se usa um modelo pre-treinado, precisa, para a pesquisa de pessoas específicas, treinar o modelo via a aplicação em três etapas:

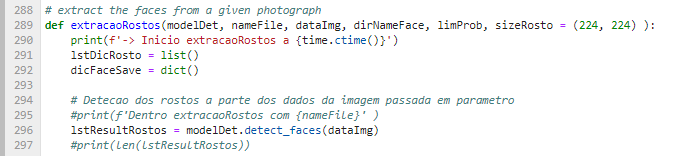
1. Extração dos rostos das pessoas que o usuário deseja pesquisar:  
    Para isso, se adicionou:

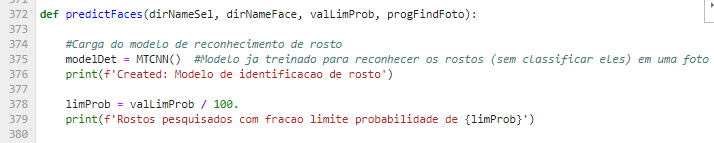
- Um campo para o usuário indicar o caminho até a pasta a usar para conservar os rostos extraídos em novos arquivos.

- Um botão para lançar a extração dos rostos, das fotografias presentes na pasta de pesquisa das fotos indicada pela usuário:



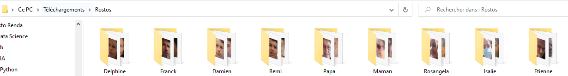
A identificação dos rostos está realizada pelo modelo MTCNN, estudado durante [o Sprint 5](#MTCNN) , chamando o seu método detect\_faces.





Os rostos identificados com uma probabilidade superior ao limite indicado pelo usuário estão conservados em arquivos dentro da pasta dos rostos.

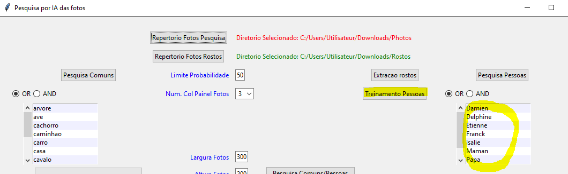
1. Organização manual pelo usuário, dos arquivos de rosto de cada pessoa a pesquisar, dentro de sub-pastas da pasta dos rostos, nomeando elas com o label de identificação da pessoa dos rostos:



1. Treinamento de um modelo de pesquisa das pessoas com uma sub-pasta dentro da pasta dos rostos:

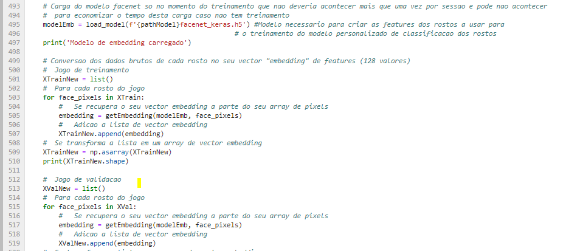
- Os labels das pessoas, com uma sub-pasta da pasta dos rostos, aparecem em uma “list-box”.

- Um botão permite de lançar o treinamento com target todas as pessoas indicadas nesta list-box.



- O treinamento começa pela criação de um vector “embedding” para cada imagem de rosto, usando o modelo facenet estudado durante o [sprint 5](#FACENET):

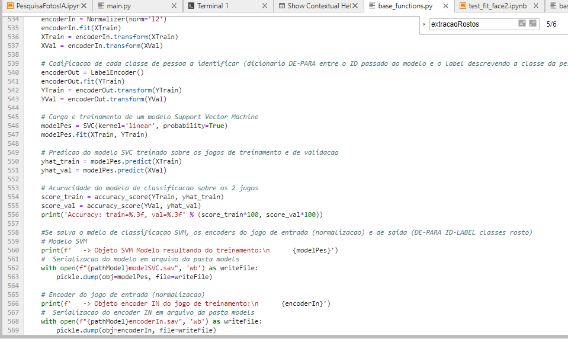
- Para cada classe de rosto, se separa 20% para avaliar a predição do modelo ao final do treinamento.



- Um modelo SVR está treinado, usando os vectores “embedding” de cada target.

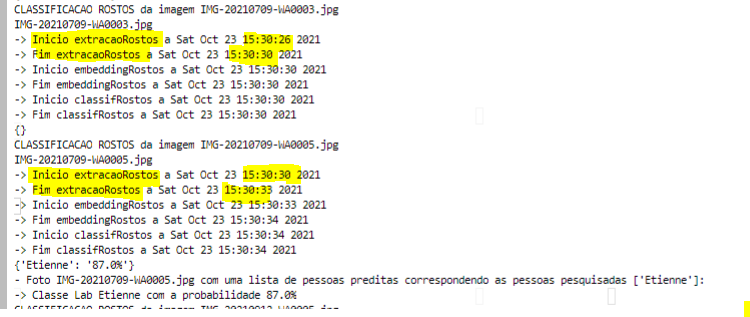
- Se conserva em arquivos, o modelo treinado e os objetos de normalização e de codificação das targets.

- Um pop-up aparece para indicar ao usuário a precisão do modelo treinado (precisão avaliada sobre as 20% de fotografias de rosto que foram retiradas do jogo de treinamento).



Depois do treinamento, o usuário tem a possibilidade de pesquisar as fotografias usando com critérios de seleção uma ou várias pessoas, indicando se todas as pessoas tem que estar achadas ou só uma.

OBS: Esta pesquisa está atualmente relativamente lenta, devido ao tempo passado para extrair os rostos de cada fotografia analisada:

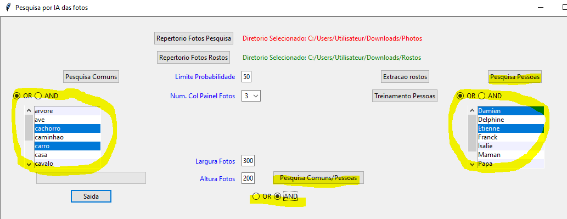


* + - * Para cada fotografia achada no repertorio de pesquisa, se realiza as 3 etapas de pesquisa de pessoa: extração dos rostos, geração do vector “embedding” de cada rosto e classificação destes vectores para comparar contra as pessoas selecionadas.

A primeira etapa de extração dos rostos demora de 3 a 10 segundos em função do número de rostos presentes na fotografia.

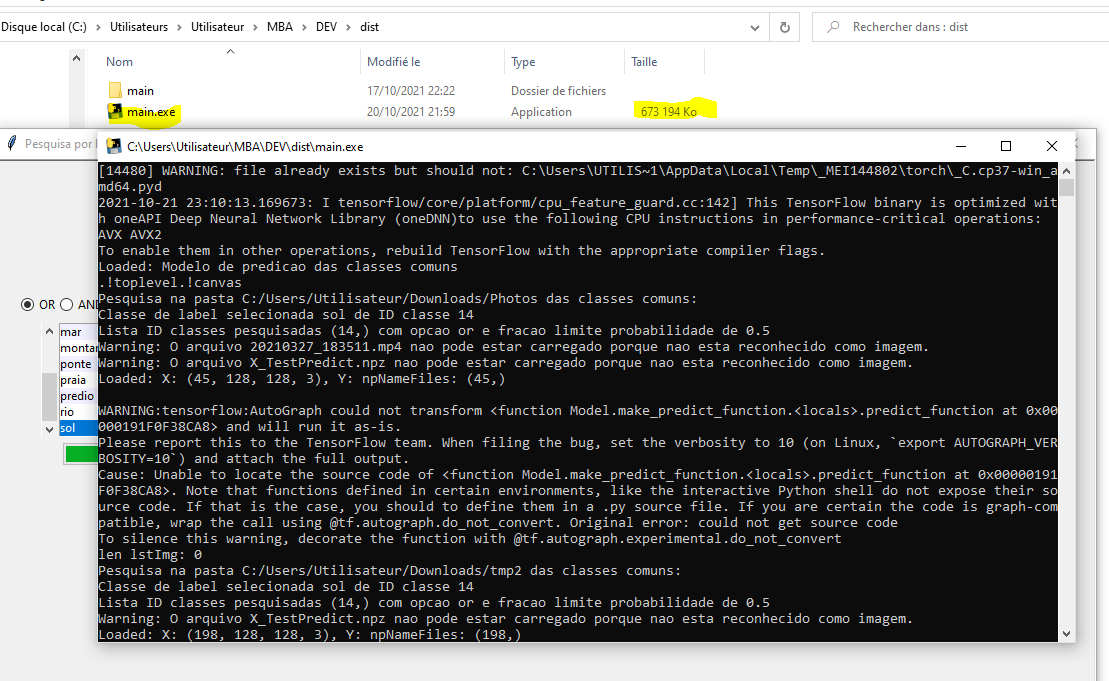
Uma possível otimização seria de rodar o modelo MTCNN uma única vez por pesquisa, juntando todas as fotografias em uma única matriz...

O usuário pode também juntar critérios de seleção de pessoas específicas com critérios de elementos comuns:



O código foi finalmente estruturado para lançar a aplicação via um programa Python.

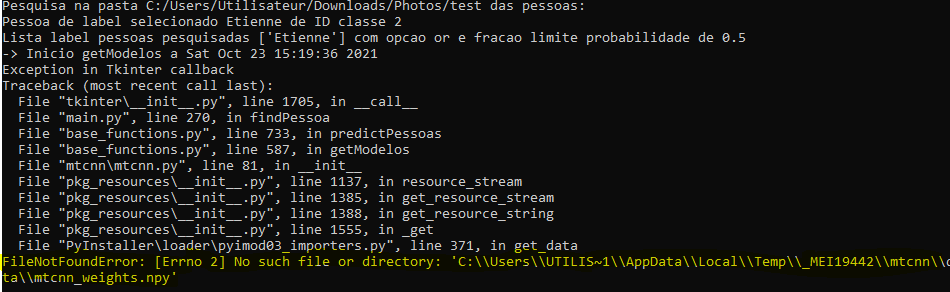
Se gerou um binário .exe, do programa mais de todos os módulos que ele importa, via o modulo pyinstaller, afim de poder rodar a aplicação em uma estação sem o Python instalado.



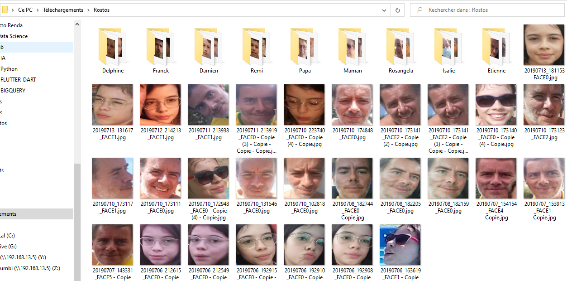
* + - * O binário ficou com um tamanho importante de quase 700 megas, provavelmente por causa dos numerosos módulos Python importados inteiro. Deve estar possível de reduzir o tamanho, importando só as classes/funções usadas pelo aplicativo.

Mas algumas funcionalidades acabam por falhar, rodando a aplicação via este binário...

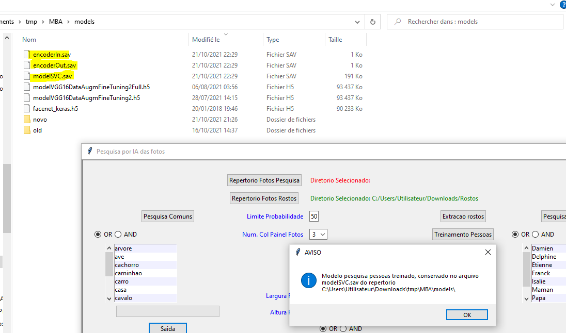
Exemplo da pesquisa de fotos selecionando pessoas, falhando com a mensagem de erro abaixo:



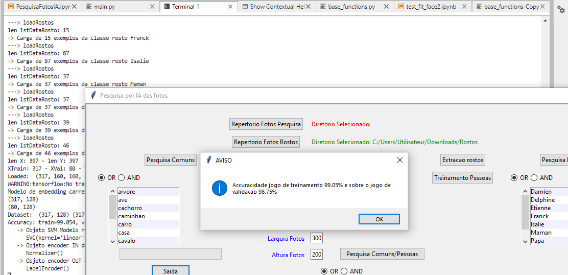
* Evidência da solução:
* Como a execução via binário “stand alone” apresenta falhas, se evidencia a solução rodando o programa via o interpretador Python.
* Resultado da extração, em arquivo, dos rostos identificados nas fotografias, organizando manualmente eles em sub-pastas para o posterior treinamento de pesquisa das pessoas destes rostos:



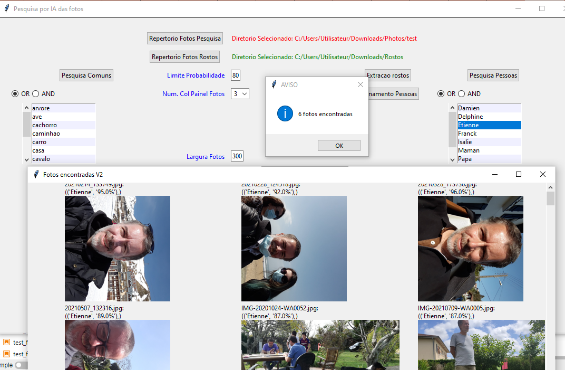
* Resultado do treinamento, usando a organização em sub-pastas dos arquivos de rosto, de nome o label da pessoa destes rostos:



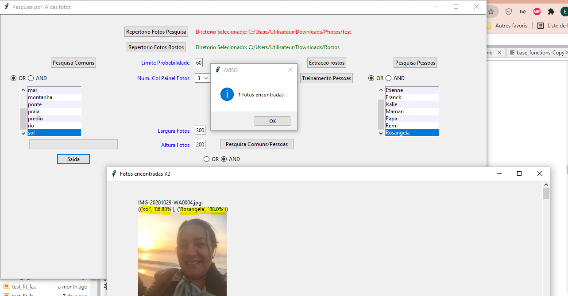
* + - * O modelo treinado, assim que os objetos de normalização e de codificação, estão conservados em arquivos para posterior uso durante a pesquisa de fotografias selecionando pessoas.



* + - * Apresentação ao usuário da precisão do treinamento sobre o jogo de treinamento e de validação (80% e 20% das amostragens).
      * Resultado da pesquisa selecionando o label “Etienne”, me representando:



* + Resultado da pesquisa de pessoa (a minha esposa com o label “Rosangela”) acoplada com a pesquisa de um elemento comum (de label “sol”).



### 2.7.2. Lições aprendidas

* + Foi visto, durante o precedente Sprint, que a pesquisa das fotografias por tema comum está limitada pela precisão baixa do modelo, principalmente ligada a volumetria baixa do jogo de treinamento, mas a pesquisa estava bem rápida e podia se aplicar sem problema em um repertorio com centenas, milhares de fotos.
  + Para a pesquisa com tema pessoal, a situação está inversa. Tem uma precisão alta mas um tempo de processamento por fotografia demais cumprido para a pesquisa poder estar realizada sobre um repertorio com centenas de fotografia.
  + A funcionalidade planejada inicialmente neste sprint, de treinamento permanente dos modelos usando o feedback do usuário, acabou por não estar desenvolvida.

Ela seria complicada a implantar pelas razoes seguintes:

- Precisaria de uma interface gráfica bem convivial para o usuário poder indicar facilmente as predições erradas/certas das fotografias trazidas pela pesquisa.

- Se poderia marcar só os falsos positivos (fotografias apresentadas ao usuário a pesar de estar com os temas selecionados pelo usuário), mas faltaria todos os falsos negativos (fotografias não apresentadas, a pesar que elas correspondem aos temas/critérios de pesquisa).

- O uso deste feedback seria principalmente interessante para melhorar a acuracidade do modelo de predição dos temas comuns, mas o treinamento deste modelo ficaria muito lento rodando em uma estação com CPU.

- Os modelos estão treinados usando um jogo de fotografias contendo ao mínimo um dos temas de pesquisa, portanto, como o treinamento poderia se aproveitar das fotografias com temas preditos erroneamente, que corresponderiam a nenhum dos temas do treinamento?

# 3. Considerações Finais

## 3.1 Resultados Finais

Este projeto permitiu de alcançar um protótipo do objetivo do projeto, de ter uma solução “inteligente” (reconhecimento pelo conteúdo), local (rodando no sistema operacional da estação do usuário), de busca de fotografias, mas o principal ganho foi a aprendizagem pessoal das tecnologias e das oportunidades do campo da visão computacional.

Se trata de um campo de conhecimento, com um desenvolvimento exponencial desde duas décadas, devido à explosão dos dados disponíveis via Internet e utilização das redes neurais convolucionais.

Está revolucionando uma multitude de atividades quotidianas como:

* A segurança: liberação de acesso a locais/aplicações, identificação de pessoas procuradas, reconhecimento de infrações em tempo real, envelhecimento da fisionomia de pessoas desaparecidas, controle do uso de EPI (equipamento de proteção individual, da máscara no contexto atual de pandemia) ...
* O controle qualidade: identificação visual de defeitos na cadeia de produção...
* O controle do estoque: identificação de ruptura em gondola, inventario, data de consumo vencida, presença de etiqueta com o preço correto, lojas autônomas...
* A circulação automatizada: robôs de tarefas domesticas/militares, carros autônomos, ajuste do deslocamento de misseis...
* A fluidez de fluxos: sincronização dos semáforos em função da densidade de carros circulando, abertura de caixas de atendimento em função das filas de pessoas...
* A cultura: restauração de fotos/filmes antigos, digitalização de textos antigos...
* A fotografia, na assistência para tirar as fotos, na pesquisa delas, na movimentação de fotos estáticas...
* A saúde, com a assistência a pessoas com deficiência visual, o processamento de imagens de radiografia, de tomografia, ressonância magnética...
* Etc.

Mas como qualquer tecnologia poderosa, ela permite aplicações perigosas, que precisam da instauração de novas legislações, para tentar controlar os riscos que elas trazem:

* Deep fake com a propagação de imagens e vídeos falsos
* Novas armas,
* Perda de privacidade.
* Etc.

Ao nível dos resultados práticos do projeto, se pode destacar:

* O entendimento de algumas das redes convolucionais de referencia: Lenet5, Alexnet, VGG, Inception, Resnet.
* O entendimento e a implantação do pipeline das diferentes etapas de geração de modelos de visão computacional:
  + Preparação dos dados, com a coleta, limpeza, preprocessamento (normalização, tamanho...), separação em jogo de treinamento/validação.
  + Definição da métrica de acuracidade, para acompanhar o avanço do treinamento e comparar diferentes modelos.
  + Treinamento com aplicação de técnicas de otimização, a mais poderosa sendo o transfer learning, mas foi também estudado e aplicado o “drop out”, “data augmentation”, “fine tuning”, “call-back”.
  + Conservação dos modelos mais promissores.
  + Aplicação destes modelos em novos jogos de fotografias.
* O entendimento das diferenças para treinar um modelo para reconhecer temais comuns (cachorro, gato) ou pessoais (pessoas do nosso círculo pessoal).

*Observações:*

* + Diferente do que eu imaginava inicialmente, ficou mais fácil atingir uma precisão boa para modelos de temas pessoais que comuns, devido ao aproveitamento de modelos já prontos e performantes, para isolar os rostos e gerar as features destes rostos. O treinamento do modelo final de classificação, a parte das features destes rostos, acaba por estar relativamente simple, visto que ele não necessita de uma volumetria importante de fotos para atingir uma precisão alta.
  + Mas, o uso deste modelo, dentro da aplicação, ficou bem lento, devido ao tempo médio de 4 a 6 segundos por foto, para a separação dos rostos.
  + Ao contrário, os resultados atingidos com o melhor modelo treinado para temas comuns, foram abaixo das expectativas. O uso deste modelo na aplicação ficou rápido, mas acaba por gerar uma proporção elevada de falsos positivos (fotos apontadas a pesar que elas não tem os temas procurados) e de falsos negativos (fotos não apontadas, a pesar que elas tem os temas procurados).
  + Isso se deve em parte a minha inexperiência, mas sobre todo ao jogo baixo de fotos coletadas, para treinamento, de cada tema a reconhecer (entre 50 e algumas centenas por tema).
  + Esta coleta foi a tarefa a mais trabalhosa e entediada do projeto, já que, mesmo para chegar a um jogo pequeno, precisou de vários dias de trabalho para escolher manualmente as fotos da Internet e para marcar os temas delas, a traves de um padrão de nomeação dos arquivos.

Foi um bom exemplo para demonstrar que, para algumas aplicações do Deep Learning, o mais importante, atualmente, para atingir um bom resultado, está a volumetria, a variedade, a qualidade (labels corretos) dos dados a fornecer para o treinamento do modelo, que a sofisticação do tipo de modelo usado.

* O entendimento e o desenvolvimento de uma aplicação local, encapsulando os modelos, para a pesquisa “inteligente” de fotos, por um usuário comum.

Foi uma boa surpresa de descobrir que a linguagem Python, escolhida para o treinamento dos modelos, tem na sua distribuição padrão (não precisa de instalação de módulos complementares) um modulo para criação de aplicação destkop (Tkinter), de fácil entendimento e que permite relativamente simplesmente e rapidamente de desenvolver aplicações básicas.

Com este modulo foi possível, em 2 sprints, de desenvolver uma aplicação que atinge os objetivos fixados ao início do projeto:

* + Aplicação local que pode rodar em sistema operacional da estação do usuário.
  + Funcionalidade de pesquisa de fotos em função de um ou vários temas comuns selecionados pelo usuário, usando um modelo já treinado (sem possibilidade de evolução desde a aplicação).
  + Funcionalidade de treinamento on-line do modelo de pesquisa de temas pessoais, em três etapas:
    - Extração automática dos rostos das fotografias fornecidas.
    - Classificação manual, pelo usuário, destes rostos em pastas nomeadas com o label da pessoa correspondente.
    - Geração automática das features destes rostos e uso deles para treinamento final do modelo de classificação.
  + Apresentação das fotos reconhecidas pelos modelos, em uma janela separada da aplicação, com indicação do label e da probabilidade dos temas reconhecidos em cada foto.
  + Configuração pelo usuário dos limites mínima da probabilidade de pesquisa (um para os temas comuns/pessoais e outro para os rostos), de parâmetros de apresentação das fotos (número de colunas, comprimento e largura das fotos.

Mas se trata ainda de um protótipo de aplicação rudimentar, com muitas limitações que precisariam estar trabalhadas, para ter um produto distribuível (descritas no capítulo sobre os próximos passos), como o exemplo da atualização em tempo real da barra de progressão, para avisar o usuário que a pesquisa está bem andando e não está travada (crucial, visto o tempo que pode estar cumprido das atividades de treinamento/pesquisa on-line das fotos por temas pessoais).

## 3.2 Contribuições

* Solução local de pesquisa “inteligente” das fotos.

Pode interessar as pessoas com conexão limitada a Internet e/ou com receio de compartilhar dados pessoais na cloud.

Pesquisando no Internet, não achei soluções standalone (rodando na estação do usuário sem depender de recursos da Internet), para uso doméstico, de pesquisa de fotos via IA.

A página seguinte, que lista as melhores soluções atuais de visão computacional, apresenta principalmente soluções web para uso professional.

<https://www.g2.com/categories/image-recognition>

* Envolvimento mínimo do usuário na preparação dos modelos de pesquisa
  + Para a pesquisa dos temas comuns, a aplicação já vem com um modelo treinado, para uma lista predefinida de temas (que poderia estar atualizado de vez em quando, por uma funcionalidade de download a desenvolver).
  + Ele precisaria se envolver, para a pesquisa via temas pessoais, só na organização em pastas dos arquivos com os rostos extraídos automaticamente. Bastara depois de um click, para o resto do treinamento de um modelo de pesquisa das pessoas com rostos organizados. Só precisa pelo momento de paciência, visto o tempo cumprido de treinamento e de pesquisa, sobre tudo na etapa de extração dos rostos.
* Opções de pesquisa
  + Por tema (comum ou pessoal), se pode selecionar uma ou várias classes e indicar o critério de pesquisa (só fotos com presença de todas as classes selecionadas ou todas as fotos com presença de ao mínimo uma destas classes).

Exemplos de pesquisa possível:

- Fotos com presença do sol e de cavalo.

- Fotos com presença de cachorro ou gato.

* + Possibilidade de combinar as pesquisas por tema comum com a por tema pessoal, indicando o critério de combinação (or/and):

Exemplos de pesquisa possível:

- Fotos com presença da minha esposa ou da minha filha, com mar ou praia.

- Fotos com minha presença ou de carro.

* Opções para calibrar a precisão da pesquisa.
  + O usuário pode indicar o percentual de probabilidade mínimo de presença das classes dos 2 temas que ele selecionou.

- Aumentando este limite, se diminui o risco de apresentar fotos não relevantes (falsos positivos), mas em contrapartida, se aumenta o risco de passar ao lado de fotos que contemplam os critérios de pesquisa (falsos negativos).

- Diminuindo este limite, se aumenta a probabilidade de apresentar todas as fotos que correspondem aos critérios de pesquisa (verdadeiros positivos), mas em contrapartida se aumenta o risco de apresentar fotos não relevantes (falsos positivos).

* + Um outro campo permite de indicar o percentual limite de probabilidade para calibragem da extração dos rostos (primeira etapa da feature de treinamento on-line do modelo de pesquisa dos temas pessoais).
* Apresentação das fotografias
  + O usuário pode configurar a disposição das fotos encontradas, indicando o cumprimento/largura e o número de colunas.
  + Os temas de pesquisa encontrados estão indicados encima da foto, com a probabilidade.

## 3.3 Próximos passos

Tem cinco pontos principais a trabalhar, para chegar a um produto que poderia estar distribuído:

* + - * 1. Voltar a treinar o modelo de pesquisa dos temas comuns, incluindo muito mais temas (passar dos 15 atuais a mais de centenas), e montando para cada tema um jogo de fotografia consequente (milhares por tema em vez das centenas atuais) para chegar a uma precisão aceitável (mais próximo de 80% que dos 60% atuais).

Considerando um deploy desta solução, um novo botão permitiria ao usuário de realizar o download do modelo resultando do último treinamento, aproveitando assim de uma lista estendida de tema comum de pesquisa, e de uma melhor precisão.

* + - * 1. Acelerar a pesquisa dos temas pessoais, que hoje demora em torno de 6/8 segundos por fotografia.

Uma eventual solução seria de submeter as fotografias a analisar, uma única vez por modelo (o de separação dos rostos, o de geração das features destes rostos e o de classificação em função das features), juntando elas em uma matriz, em vez de chamar os modelos a cada fotografia.

Uma outra solução seria de solicitar o feedback dos usuários sobre as previsões corretas, para conservar elas como tags das fotos e assim acelerar as próximas pesquisas usando estes tags em vez das predições.

* + - * 1. Melhorar a ergonomia da aplicação, permitindo a pesquisa recursiva (descendo toda a arborescência da pasta de pesquisa selecionada pelo usuário) e o acesso direto, desde a aplicação, aos arquivos originais das fotos apresentadas.
        2. Atualizar on-line a barra de progressão das pesquisas.
        3. Gerar um binário “stand-alone” (não precisando de instalação de Python), com 100% das funcionalidades operacionais (se consegui a gerar este binário durante o projeto, mas algumas funcionalidades falharam).