# Nanodegree Engenheiro de Machine Learning

## Projeto final

Leandro Humberto Vieira

11 de maio de 2018

## I. Definição

A definição deste projeto consiste em construir um método de multiclassificação de roupas, através das técnicas de Deep Learning.

### Visão geral do projeto

A imagem é a primeira impressão que uma pessoa tem ao ver alguém, é algo que nosso cérebro faz automaticamente para armazenar informações e é realizado a partir do momento em que começamos a interagir com esta pessoa. Uma imagem diferente do que uma pessoa é durante uma conversa causa um "ruído" na comunicação e uma confusão no nosso cérebro.

Hoje a imagem é uma via de comunicação pessoal tão importante quanto a fala, dessa forma surgiu a necessidade de profissionais que ensinam a arte da comunicação não-verbal, através de métodos e análises que permitem ensinar uma pessoa a se conhecer e, assim, expressar melhor seu estilo e personalidade através de sua imagem. O processo de consultoria de imagem ensina a pessoa a identificar seu estilo e sua personalidade, e após este passo, trabalhar com o objetivo de construir uma imagem pessoal mais positiva e coerente com ela mesma.
A consultoria de imagem trabalha em sua grande parte com roupas, portanto há algumas etapas do processo consultivo que visam limpar, organizar e classificar as roupas da cliente.

#### Por que a consultoria de imagem é importante?

A imagem, o estilo e as vestimentas são fatores que regem ou influenciam vários aspectos da vida de uma pessoa, o que torna esta informação muito valiosa e digna de vários estudos como os descritos abaixo:

- Como a roupa afeta sua consciência - Livro descrito no artigo
- Como a roupa impacta no seu sucesso profissional
- A correlação da vestimenta com seu relacionamento afetivo
- Benefícios da organização das roupas

#### Proposta

Este projeto demonstra a tentativa de montar um serviço de classificação de roupas, através da utilização de tecnologia e inteligência artificial para ajudar as pessoas a conhecerem suas roupas, pois saber o que se veste é uma excelente forma de adquirir auto conhecimento. Automatizar a classificação de roupas também é uma forma de melhorar o trabalho de consultoria de imagem, possibilitando a geração de informações gerenciais de roupas para facilitar o processo de decisão destes profissionais durante a consultoria de um cliente.

Para desenvolver a solução de classificação, foram encontrados na internet alguns conjuntos de imagens que são excelentes candidatos para a solução do problema, os mesmos serão discutidos em detalhes posteriormente.

### Descrição do problema

O problema a ser resolvido se define em obter uma visão estratégica do armário de uma mulher. Apenas olhando para o armário, por mais organizado que esteja, não é possível responder perguntas como:

- Quantas blusas pretas eu tenho?
- Quais são as peças essenciais que faltam no armário?
- Qual a proporção casual/trabalho que tenho?

Para responder tais perguntas, é necessário classificar as roupas entre vários tipos de vestimenta, como alguns destes exemplos abaixo:

1. Lingeries
2. Acessórios
3. Bolsas e sapatos
4. Blusas
5. Terceira peça
6. Calças
7. Shorts
8. Saias
9. Vestidos/Macacões/Macaquinhos
10. Roupas de Academia

Apesar das categorias acima abrangerem de forma pragmática o domínio das roupas, é necessário uma classificação mais específica, pois várias roupas possuem subdomínios, que podem alterar totalmente a imagem que uma pessoa quer comunicar, como também casos que a roupa utilizada pode ferir as regras de etiqueta de um ambiente determinado, e estes aspectos são importantes durante a avaliação de um guarda-roupas.

Como exemplo temos a blusinha e o blazer, ambos são de um mesmo grupo de roupas, porém com projeções de imagens diferentes, e a blusa também pode não ser adequada para certos ambientes de trabalho.

Para solucionar o problema, será criado uma interface web onde um usuário consiga classificar uma roupa através de uma imagem, e para atingir esta solução, as etapas de construção da solução são:

## 1. Interface

Construir uma aplicação web em Flask, que terá uma interface para upload de fotos. A aplicação será a mais simples possível, pois não é o foco deste projeto.

## 2. Transporte de dados

A foto que foi escolhida para a análise não será analisada no dispositivo cliente, logo é necessário realizar o transporte dos dados para o servidor de análise. Este transporte será realizado através de uma requisição http.

### 3. Recepção dos dados

Para receber os dados através da requisição, será construído um simples servidor RESTful, o servidor irá fazer o papel de receber os arquivos e realizar as operações básicas de controle de sessão e chamada do algoritmo de classificação.

### 4. Pré processamento dos dados

Para realizar a classificação de um look, serão necessárias transformações na imagem de entrada, que consistem de recortes e reescalações

### 5. Análise dos dados

Aplicar para cada requisição a CNN de classificação, e retornar o resultado para a aplicação cliente.

### 6. Resultado final

Após a aplicação cliente receber o resultado na análise, será mostrado para o usuário a resposta da classificação realizada.

## Métricas

A métrica de validação a ser utilizada pelo modelo será a acurácia.

O processo de validação das métricas será dividido nas etapas abaixo:

- Criação de uma rede neural convolucional completa
- Adaptação de uma rede neural convolucional já construída, aplicando a técnica de Transfer Learning
- Treinamento e validação dos modelos
- Comparação da acurácia entre os modelos

# II. Análise

## Exploração dos dados

Para a construção da solução foram avaliados 2 conjuntos de dados, e durante o desenvolvimento da solução, um terceiro conjunto de dados foi encontrado, os três estão detalhados abaixo:

### Fashion MNIST


O Fashion MNIST é um conjunto de dados criado pela Zalando Research, que contém **60 mil imagens** de roupas no conjunto de treino e outras 10 mil no conjunto de teste. Este conjunto de dados tem como objetivo principal substituir o MNIST, pois o mesmo é um conjunto que não condiz com a realidade dos problemas de Deep Learning

## Apparel classification with Style


O Apparel Classification Set contém **80 mil imagens coloridas** retiradas da web através de crawlers e já classificadas em 15 grupos diferentes de categorias de roupa. Os dados deste conjunto são mais semelhantes com a realidade da aplicação final, pois mostram as roupas vestidas em pessoas em lugares naturais, porém contém uma série de ruídos nos dados, como fotos de caixas e fotos com zoom demais.

## Deep Fashion

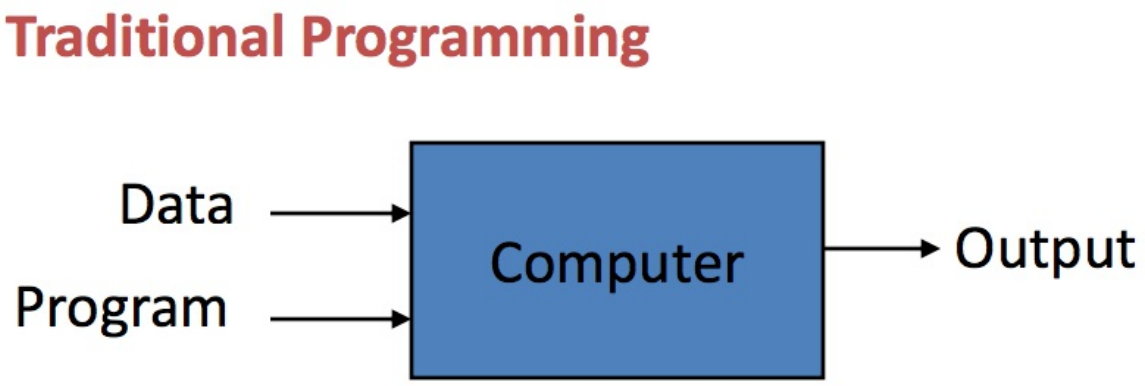
O Deep Fashion foi encontrado durante o desenvolvimento da solução, e devido a sua qualidade, foi escolhido como conjunto de dados definitivo para o treinamento das redes neurais que seriam construídas. O Deep Fashion contém toda a informação necessária para a construção de redes neurais que envolvam roupas. O conjunto de dados possui **200 mil imagens para treinamento, e 40 mil imagens para validação e teste** totalizando um total de **280 mil imagens**, divididos em 50 categorias diferentes de roupas.

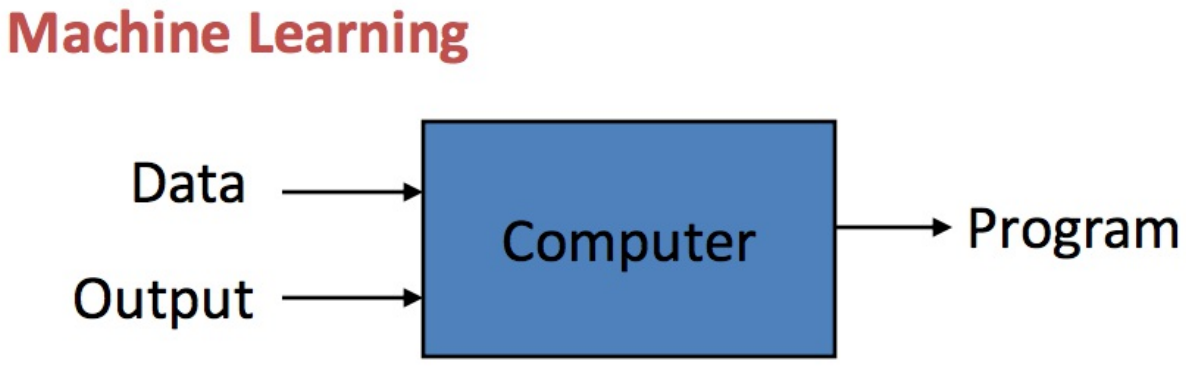
Outro ponto a ser destacado em relação ao ACS Dataset, é que apesar de ser um conjunto de dados muito maior, o Deep Fashion ainda se apresentou como um conjunto de dados com menos ruído, com raras ocorrências de fotos que fugiam do domínio ou com representações exdrúxulas.

## Algoritmos e técnicas

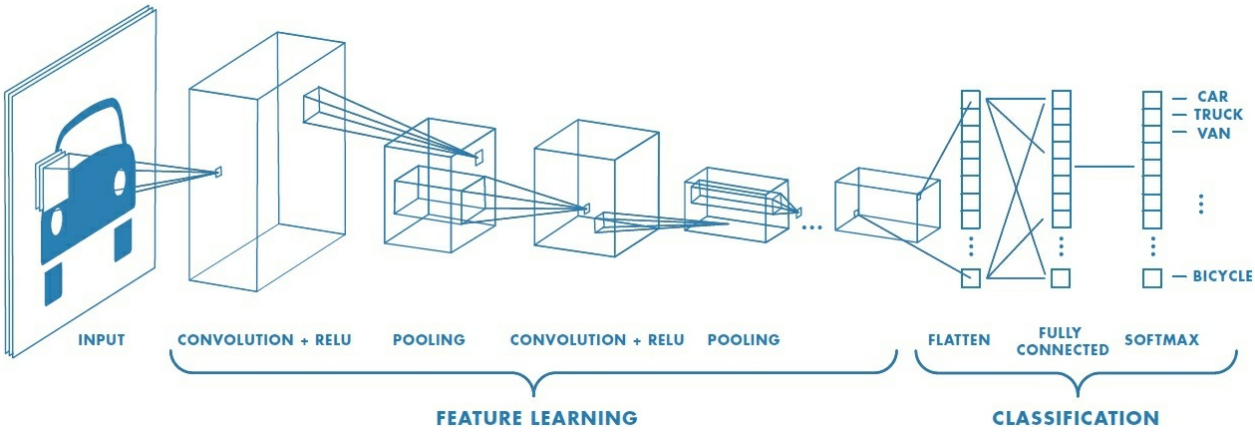
Para realizar a classificação das roupas, serão implementadas **redes neurais**. Redes neurais são conhecidas por atingir o *"estado da arte"* onde o domínio do problema se referem a imagens, assim ,como primeira opção, foram construídas e treinadas duas **redes neurais convolucionais** para a classificação das imagens.

**Deep Learning** é uma técnica específica de aprendizado de máquina, ou seja, o programa deve "aprender" a solução por si, pois ela não será programada explicitamente. Para isso, o programa utiliza dados de entrada para realizar seu aprendizado, utilizando eles para *"treinar"*.



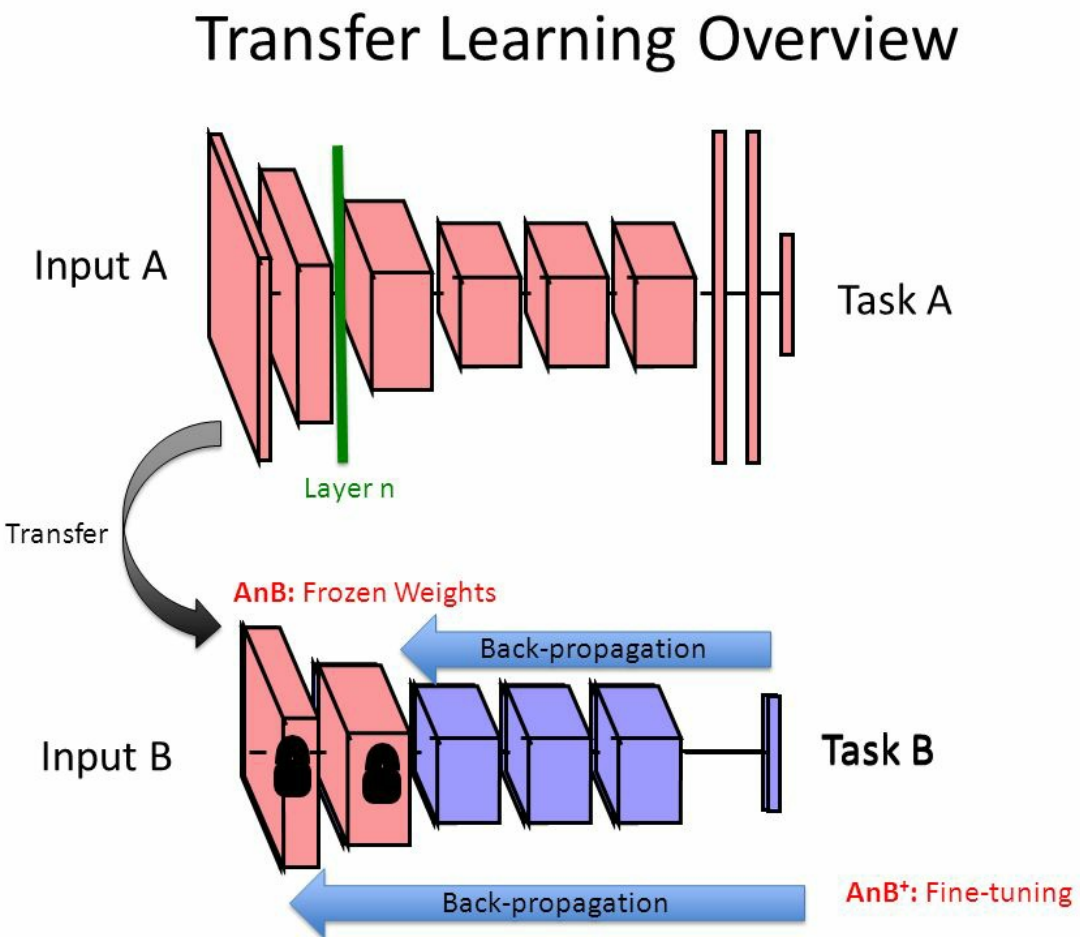



E por fim, dentro das redes neurais temos outros subtipos, sendo os mais conhecidos MLPs e CNNs. Para esta solução, as CNNs são a melhor abordagem para o problema, pois suas camadas convolutivas deixam a rede melhor preparada para lidar com imagens que tem seu objeto de interesse transladado ou rotacionado.




### Mas porque duas CNNs?

Enquanto um uma das redes criarei um modelo sequencial comum, do zero, na outra CNN será aplicado o conceito de Transfer learning, que consiste de instanciar uma rede neural já treinada(geralmente treinada com dados do ImageNet) e adequá-la para o seu problema, realizando alguns ajustes e um treinamento adicional.




## Benchmark

Como o modelo de benchmark, será utilizada uma CNN extremamente simples, construída do zero, no Keras. A rede consistirá de 13 camadas, representadas abaixo:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 300, 200, 16)      304
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 150, 100, 16)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 150, 100, 32)      3104
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 75, 50, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 75, 50, 64)        12352
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 37, 25, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 37, 25, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 59200)             0
_________________________________________________________________
dense_1 (Dense)              (None, 500)               29600500
_________________________________________________________________
dropout_2 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_2 (Dense)              (None, 500)               250500
_________________________________________________________________
dropout_3 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_3 (Dense)              (None, 50)                25050
=================================================================
Total params: 29,891,810
Trainable params: 29,891,810
Non-trainable params: 0
_________________________________________________________________
```

Todos os detalhes da implementação do modelo de benchmark estão localizados aqui

# III. Metodologia

## Pré-processamento de dados

Durante o pré processamento dos dados, foram realizadas algumas operações para habilitar o treinamento da rede neural.

- As imagens vieram todas juntas, assim foi necessário separar as imagens em diretórios diferentes, baseado nas informações contidas no arquivo **list_eval_partition.txt**.
- As imagens não possuiam a mesma resolução, e ao construir uma rede neural, é necessário definir um tamanho de entrada fixo para as imagens. Para contornar este problema, todas as imagens foram transformadas para a resolução de 300px de altura por 200px de largura. Esta medida foi escolhida pois é a que menos deforma as imagens, já que a maioria das imagens são fotos em formato de retrato, e também é constatado na documentação do dataset que a maior parte das fotos foi mantida com 300px em sua parte maior. *"The long side of images are resized to 300"*.
- Foi realizada a reescalação dos pixels, realizando a mudança da amplitude numérica deles de (0,255) para (0,1). Esta atitude foi tomada principalmente pelas seguintes razões: Economia de memória, eficiência de processo e um tratamento mais "justo" entre as imagens do treinamento.

Para realizar estas operações de Pré-processamento, foi criado uma célula com código customizado para separar as imagens entre os diretórios de treino, validação e teste. Porém para as outras transformações, foi utilizada funções pré definidas do Keras, o que facilitou bastante o trabalho, exemplo abaixo:

```
# Reescalação dos pixels
from keras.preprocessing.image import ImageDataGenerator
```

```
train_datagen = ImageDataGenerator(rescale=1./255)

# Transformação da resolução das imagens para o tamanho (300,200)
tgt_size = (300,200)
train_generator = train_datagen.flow_from_directory(
    '/content/train/',
    target_size= tgt_size,
    batch_size= bat_size,
    class_mode='categorical'
)
```

**Este pré-processamento foi realizado para ambas as redes**.

## Implementação

Para a implementação das redes foram realizadas duas construções, feitas em Keras. Uma rede neural simples de 13 camadas(detalhes aqui) e uma rede de transfer learning, baseado na VGG19(detalhes aqui).

Estas implementações foram escolhidas pelos seguintes motivos:

A CNN simples é um modelo que já havia sido discutido durante as aulas do módulo, e assim apresentava a familiaridade e simplicidade necessária para um modelo de benchmark.

A VGG19 foi selecionada por um semi acaso, apenas ponderei que não queria utilizar um modelo complexo demais(como a resnet50, por exemplo), que iria gerar em tempos mais lentos de treinamento, assim peguei o **meio termo** entre os modelos oferecidos pelo keras.

## VGG19 detalhes

A implementação da VGG19 foi realizada a partir de transfer learning. Para realizar essa técnica, importamos uma rede neural já construída, juntos com seus pesos adquiridos de semanas de treinamento com os dados do banco de imagens ImageNet. Como o ImageNet não é o dominio do nosso problema, a rede neural está preparada para executar classificações de outros objetos, como animais e coisas.

Para convergir uma rede neural pronta para o dominio do sue problema, é necessário remover a última camada da rede neural importada e substituir por uma camada que contém o dominio do seu problema, a nova camada de dominio geralmente vem acompanhada com outras hidden layers, para fortalecer os resultados através de treinamento.

No Keras a implementação de transfer learning foi facilmente realizada, o Keras já permite que se importe a rede neural sem sua ultima camada, através do comando abaixo:

```
model = VGG19(weights = "imagenet", include_top=False, input_shape = (300, 200, 3))
```

Após a importação, foi necessário apenas criar as camadas finais do modelo e "plugar" elas no final da rede neural, esta atividade foi providenciada pelo código abaixo:

```
from keras.models import Sequential, Model
from keras.layers import Flatten, Dense, Dropout
x = model.output
x = Flatten()(x)
x = Dense(1024, activation="relu")(x)
x = Dropout(0.5)(x)
x = Dense(1024, activation="relu")(x)
predictions = Dense(50, activation="softmax")(x)
model_final = Model(input = model.input, output = predictions)
```

## Refinamento

Após a arquitetura de ambas redes neurais, se deu inicio a etapa de aprendizado a partir dos dados, para os modelos implementados, foram testados os seguintes parâmetros:

- CNN básica:
    - função de perda='categorical_crossentropy', otimizador='adam', métrica=['accuracy'], 3 epochs
    - função de perda='categorical_crossentropy', otimizador='rmsprop', métrica=['accuracy'], 3 epochs
- VGG19:
    - função de perda='categorical_crossentropy', otimizador=SGD(lr=0.0001, momentum=0.9), métrica=['accuracy'], 1 epoch
    - função de perda='categorical_crossentropy', otimizador='adam', métrica=['accuracy'], 1 epoch

# IV. Resultados

Os resultados as execuções contrariaram minhas expectativas, imaginava que a VGG19 teria um resultado superior comparado à CNN simples, mas os resultados mostram uma alta discrepância entre suas acurácias.
Durante a construção e treinamento dos modelos, 4 modelos foram treinados utilizando os dados do Deep Fashion e obtivemos os seguintes resultados:

- CNN básica:
    - Adam, 3 epochs: 48,7% de acurácia no conjunto de dados de teste
    - Rmsprop, 3 epochs:51,2% de acurácia no conjunto de dados de teste
- VGG19:
    - SGD(lr=0.0001, momentum=0.9), 1 epoch: 12,75% de acurácia no conjunto de dados de teste
    - Adam', 1 epoch: 12,65% de acurácia no conjunto de dados de teste

## Modelo de avaliação e validação

Após os modelos validados, o modelo CNN simples com o otimizador rmsprop foi selecionado entre os quatro candidatos, devido a sua maior acurácia.

A acurácia de 51,2% ainda é considerada muito baixa, logo o melhor modelo ainda não possui a robustez necessária para um amabiente de produção.

O modelo também foi testado com algumas fotos caseiras e tambem não houve um resultado satisfatório.

## Justificativa

Como justificativa para o resultado, tenho em mente que ambos os modelos não foram treinados o suficiente para generalizar os dados de maneira correta, suspeito que ambos se encontram em um estado de **Underfitting**, necessitando de mais treinamento.

O modelo de transfer learning, comparado ao modelo de benchmark, está muito pior nos resultados apresentados no conjunto de dados de teste.

Para a solução final, foi enviado o modelo CNN simples, mesmo que não tenha adquirido uma acurácia satisfatória.

# V. Conclusão

Como conclusão, temos 3 artefatos entregues, duas redes neurais e um servidor w eb para interface como o usuário.

Ainda permanece confuso para mim qual seria a causa da discrepância entre os resultados das duas redes neurais. Tenho a

teoria que, como a VGG19 é uma rede neural muito mais complexa do que a utilizada no benchmark, ela deve necessitar de mais tempo de treinamento, logo com a execução de mais *epochs*, ela culminaria em um modelo com acurácia semelhante ou maior ao modelo de benchmark.

## Reflexão

O projeto como um todo foi uma experiência tanto traumática quanto edificadora para mim.

Mergulhar em um problema fora do meu domínio, codificar em Python, aprender sobre como implementar IA como serviço, cloud computing e gpu computing foram áreas que fugiam totalmente de meus conhecimentos prévios, o que me agregou muito conhecimento.

Como um resumo geral de minha jornada na entrega deste projeto, posso afirmar que houveram vários altos e baixos, como também vários imprevistos que impactaram de forma negativa no projeto. A jornada do projeto se resume nos parágrafos abaixo, onde em cada parágrafo aprendi uma lição sobre todo este processo:

Iniciei o projeto utilizando o conjunto de dados escolhidos na proposta de conclusão de projeto que já havia sido aprovada, e percebi que os dados se encontravam muito sujos, com várias imagens de objetos que nada tinham a ver com o problema proposto, além de haver poucas amostras (80 mil ao todo). Para contornar este problema, encontrei outro conjunto de dados, com mais imagens e categorias para classificar. *Aprendi que soluções que demandam dados para ser construídas precisam de **dados de qualidade***.

Iniciei o desenvolvimento localmente, executando os códigos em minha máquina, logo percebi que os programas estavam demorando absurdamente para mostrar resultados, como uma solução paliativa, migrei o código para o Google Colab, resultando em alguns ajustes no código. Após alguns dias desenvolvendo no Colab, tive novas dificuldades, e novamente tive que migrar meu código para outra plataforma, o Paperspace. *Aprendi com essas adversidades que o Deep Learning é um método de aprendizado que exige um altíssimo poder computacional, com resultados muito demorados*.

Tive muitas dificuldades em lidar com o Tensorflow, ele apresentou erros diferentes para cada um dos ambientes em que testei meus programas, o que atrasou muito a entrega dos artefatos do projeto. *Aprendi que tenho que migrar para o Pytorch, RÁPIDO*.

A solução desenvolvida ainda está longe do esperado para uma aplicação de produção, mas tenho certeza que foi um passo na direção certa, após as melhorias a ser propostas, tenho certeza que será possível alcançar resultados satisfatórios.

## Melhorias

Como melhorias a este projeto, considero que há a necessidade de estudar técnicas mais avançadas de treinamento de redes neurais, que no momento não podem ser implementadas com o meu conhecimento neste campo de estudo. Estou começando a assistir algumas aulas do curso do fast.ai, e percebi que existem técnicas avançadas para se obter melhores resultados de classificação em redes neurais.

Outro ponto a ser melhorado no projeto consiste na utilização do serviço de uma forma mais completa. Devido ao prazo do projeto chegar ao fim, realizei a entrega do serviço em uma aplicação web simples, o que não agrega o valor necessário para o usuário devido a fatores simples como: entrega de pouca informação(tipo da roupa apenas), falta de persistência da informação recebida e disponibilização de dados analíticos. Como proposta de melhoria para o projeto, seria ideal realizar as seguintes modificações:

- Incrementar as informações geradas pelo serviço, como cor dominante, corte da roupa, tipo de tecido e estilo da roupa(o conjunto de dados de entrada já tem essas informações marcadas nas fotos, seria necessário apenas o treinamento de novos modelos)

- Entregar o serviço em uma plataforma embarcada, para assim fornecer na plataforma os serviços faltantes como persistência e relatórios gerenciais. Era planejado entregar o serviço em um app mobile, mas devido a aproximação do final do prazo e minha falta de experiência em desenvolvimento, tive que abandonar a idéia e entregar na web, mas a aplicação funciona em partes e se encontra em: https://github.com/leandrohmvieira/SmartDrobe

- Um ponto que seria de grande melhora, seria o enquadramento do objeto antes da classificação. Encontrei algumas fontes na internet como o Detectron e o YOLO que conseguem detectar objetos em uma imagem com várias coisas, isso permitiria cortar a foto antes da classificação, assim teríamos menos ruído durante a classificação das roupas.

- E por fim, seria uma melhora a multiclassificação de uma imagem, onde fosse possível classificar várias roupas simultâneamente, dado uma foto. Realizei algumas pesquisas na internet e vi que isso é possível, porém o material dado pelo curso não abrange este tópico.

## Agradecimentos

Gostaria de agradecer a todos da Udacity pelas aulas de alta qualidade, e da gentileza de ter disponibilizado um módulo extra de aulas de Deep Learning no NanoDegree, sem as aulas, não teria o interesse de buscar este tópico para a execução do projeto final.

Também gostaria de agradecer ao pessoal do grupo de estudo **Deep Learning Brasília**, que com as aulas extras deles, tive a bagagem adicional de conhecimento e a motivação para continuar engajado nas atividades do curso, além das várias dicas valiosas sobre como desenvolver a aplicação web e como utilizar várias funções do Keras.

E por fim, agradeço a minha esposa, Karina Tânia, por me apoiar com seu conhecimento sobre o objeto de estudo, e por me cobrir de vez enquando nas tarefas domésticas que se atrasaram um pouco nestes 6 meses de jornada de aprendizado de máquina.