UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E ELETRÔNICA
EEL7513 – INTRODUÇÃO AO APRENDIZADO DE MÁQUINA

Aplicação de redes neurais convolucionais para a classificação multirrótulo de peças de roupa

Kauê Cano Souza – 18204680
Ruan Cardoso Comelli – 201901993

Florianópolis
2019

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CNN	Rede Neural Convolucional
LSTM	*Long Short-Term Memory*
MLP	Perceptron Multicamadas
NN	Rede Neural
ReLU	Unidade Linear Retificada
SGD	Método do Gradiente Estocástico

LISTA DE SÍMBOLOS

F_β	*Score* F_β	[−]
F_1	*Score* F_1	[−]
A	Acurácia	[−]
$\mathscr{D}$	Conjunto de dados	[−]
$\mathscr{D}_{\text{test}}$	Conjunto de teste	[−]
$\mathscr{D}_{\text{train}}$	Conjunto de treinamento	[−]
$\mathscr{D}_{\text{val}}$	Conjunto de validação	[−]
$\mathbb{R}$	Conjunto dos números reais	[−]
FN	Falso negativo	[−]
FP	Falso positivo	[−]
J	Função custo	[−]
L	Função perda	[−]
w	Peso das classificações positivas relativo ao das classificações negativas	[−]
P	Precisão	[−]
R	Revocação	[−]
y	Rótulo	[−]
$\hat{y}$	Rótulo predito	[−]
TN	Verdadeiro negativo	[−]
TP	Verdadeiro positivo	[−]
$\boldsymbol{x}$	Vetor de atributos	[−]

SUMÁRIO

1 INTRODUÇÃO

A visão computacional é utilizada em diversas áreas, como máquinas autônomas [11], segurança em tempo real [34], diagnósticos [12] e interação homem-máquina. Filtros faciais disponíveis em mídias sociais são exemplos que mostram como a tecnologia penetrou na sociedade de forma ampla, silenciosamente moldando a relação do usuário com redes neurais.

Como consequência do rápido avanço do *e-commerce* a partir de meados dos anos 2000, a internet possui uma quantidade cada vez maior de imagens dos mais diversos produtos. Naturalmente, a competitividade entre esses *websites* também é crescente, o que cria uma demanda por informações analíticas sobre o portfólio publicamente disposto de concorrentes, os quais eram altamente sigilosos a menos de 10 anos atrás.

Sistemas classificadores de imagens permitem a extração de dados valiosos a partir de fotos de produtos, como cores, estampas, acessórios, características, defeitos, combinações, ambientes e diversos outros padrões aplicáveis para cada nicho.

Para o ramo de moda, o qual atinge recordes de crescimento ano após ano e apresenta novas tendências de produção rápida e em grande escala [2], tais classificadores se tornarão de suma importância para que varejistas se situem perante seus competidores e também acompanhem padrões de consumo de seus clientes.

Parcerias entre a comunidade acadêmica tornaram possível a criação de *datasets* específicos para aplicações de visão computacional em imagens de roupa. Tal disponibilidade de quantidades grandes de dados estruturados permite o desenvolvimento de sistemas categorizadores com alto desempenho.

O intuito deste trabalho é implementar, com as técnicas de *deep learning*, um sistema classificador de imagens de *looks* de roupas e analisar os resultados obtidos.

1.1 OBJETIVOS

Mantendo em foco tanto a proposta de um classificador *multi-label* a nível próximo ao estado-da-arte e ao mercado quanto o estudo estrutural do problema de classificação de imagens com números elevados de possibilidade de atributos, o presente trabalho definiu como objetivos principais:

1. Construir uma rede neural convolucional (CNN – do inglês, *convolutional neural network*) (VGG16) utilizando uma parte do conjunto DeepFashion, empregando mecanismos de atenção envolvendo *landmarks* (pontos limítrofes da peça de roupa) para a categorização de classe (tipos de roupa como camiseta, calça e blusa) e atributos (como azul, estampado e verão).

2. Expandir o mesmo trabalho, aplicado exclusivamente para a categorização de múltiplos atributos (excluindo a classificação e a detecção de *landmarks*), analisando diver-

sos parâmetros de treino, bem como diferentes redes-base pré-treinadas no lugar da VGG16, como a ResNet50V2 e a MobileNetV2.

1.2 TIPO DE TAREFA

Neste projeto, a máquina deve aprender a classificar peças de roupa em um contexto multirrótulo. Esse problema é uma generalização da classificação *single-label*, em que, dada uma entrada, o modelo deve identificar a qual classe ela pertence. Um exemplo de conjunto de dados projetado para a classificação *single-label* é o FashionMNIST. Em contraste, em uma classificação *multi-label* de uma imagem, o modelo deve extrair características adicionais do produto como cor, cortes, estilo e peças adicionais contidas na foto. Para facilitar a visualização da problemática com modelos de categorização simples, a Figura 1 ilustra imagens de roupas em *e-commerces* de moda.

Figura 1 – Exemplos de imagens de roupa em *e-commerces* de moda.

Fonte: dos autores.

Como pode ser visto, marcas de grife raramente expõem produtos isolados nos seus websites. Geralmente as fotos incluem modelos vestidos com diversas peças simultaneamente. Ainda mais, a forma como as peças de roupas são demonstradas varia drasticamente de marca para marca e até entre categorias de uma própria marca. Tem-se desde *fullbody shots* de um *look* neutro, dando bastante destaque para o produto ofertado, até *looks* completos integrados por outras peças de roupas não ofertadas, ou mesmo fotos cortadas de apenas uma área do produto. Portanto fica claro que o *dataset* FashionMNIST, representado

na Figura 2, é inadequado para aplicações no *e-commerce* em virtude da simplicidade dos seus dados.

Figura 2 – Exemplos de imagens do conjunto FashionMNIST.

Fonte: dos autores.

À vista do que foi exposto, este projeto almeja avaliar CNNs na categorização de imagens produtos encontrados em grandes marcas de moda feminina.

O aprendizado empregado será supervisionado, utilizando bancos de dados públicos resultantes de esforços científico-comunitários. Adicionalmente, em decorrência da multiplicidade de formas com que modelos de aprendizado de máquina podem ser desenvolvidos, diferentes modelos serão comparados.

1.3 FORMATO DOS DADOS

O conjunto de dados consistirá de m exemplos, cada um sendo um par $(\boldsymbol{x}, y)$, em que $\boldsymbol{x}$ é uma imagem colorida cujo tamanho depende do conjunto de dados escolhido. Para o conjunto DeepFashion, apresentado na Seção 3.1, o maior lado de cada imagem é igual a 225 px. O rótulo y é um vetor de K componentes, sendo K o número de classes a que cada imagem pode pertencer. A k-ésima componente de y é $y_k = 1$, caso a imagem pertença à classe k, ou $y_k = 0$, caso contrário. Por ser um problema multirrótulo, o vetor y pode possuir um número variável de componentes nulas.

Do conjunto de dados selecionado são extraídos conjuntos de treinamento, de validação e de teste. O primeiro conjunto é aquele sobre o qual os parâmetros do modelo são aprendidos. Por sua vez, o conjunto de validação é usado para se otimizarem hiperparâmetros e estimar o erro de generalização. O conjunto de teste, por fim, é utilizado para se avaliar o desempenho do modelo e compará-lo com outros classificadores.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 APRENDIZADO DE MÁQUINA

De acordo com Mitchell [22], o aprendizado de uma máquina consiste na melhora do seu desempenho ao executar uma tarefa com o aumento de sua experiência. Esse aprendizado se divide em duas categorias: o supervisionado e o não supervisionado.

Algoritmos de aprendizado supervisionado experienciam um conjunto de dados na forma $\left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{m}$ em que $\boldsymbol{x}^{(i)}$ é um vetor de n componentes de denominadas atributos, e $y^{(i)}$ é chamado de rótulo ou alvo. Os principais exemplos de aprendizado supervisionado são a regressão e a classificação. Em um problema de classificação *single-label*, cada exemplo pertence a uma classe dentro de um conjunto discreto. Nessa situação, para K classes, todos os rótulos satisfazem $y^{(i)} \in \{1,\ldots,K\}$. Por outro lado, na classificação *multi-label*, cada rótulo é um vetor $\boldsymbol{y}^{(i)} = \left(y_1^{(i)},\ldots,y_{\mathrm{k}^{(i)}}^{(i)}\right)$, com $y_j^{(i)} \in \{1,\ldots,K\}\ \forall\, j \in \left\{1,\ldots,\mathrm{k}^{(i)}\right\}$ [23, 13].

Utiliza-se o conjunto de treinamento $\mathscr{D}_{\text{train}}$ para se aprenderem os parâmetros do modelo, ou seja, para se definir uma função ótima dentro de uma classe parametrizada. Já o conjunto de validação $\mathscr{D}_{\text{val}}$ é empregado para estimar o erro de generalização e escolher os hiperparâmetros do modelo – parâmetros não treináveis que controlam o comportamento do algoritmo de aprendizado. O desempenho final do modelo é medido sobre o conjunto de teste $\mathscr{D}_{\text{test}}$ [13].

2.2 REDES NEURAIS

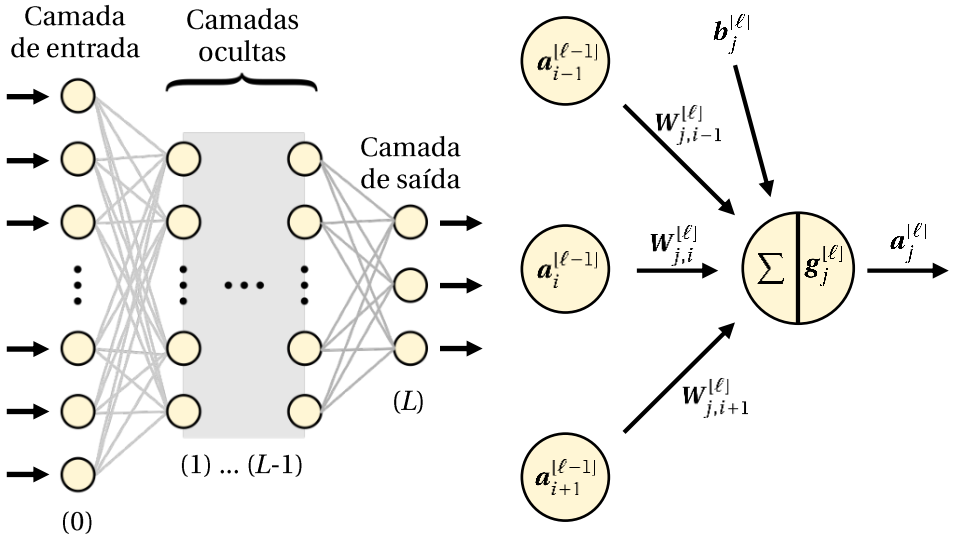
Por simplicidade, serão considerados aqui os perceptrons multicamadas. Um perceptron multicamadas (MLP – do inglês, *multilayer perceptron*) de L camadas pode ser representado como na Figura 3a. A camada $\ell = 0$ é chamada de camada de entrada, correspondente à entrada $\boldsymbol{x} \in \mathbb{R}^n$, ao passo que a camada $\ell = L$ é denominada camada de saída, pois resulta na predição $\hat{\boldsymbol{y}} \in \mathbb{R}^p$. Para $0 < \ell < L$, as camadas são denominadas ocultas [13]. Cada camada ℓ se compõe de n_ℓ unidades, sendo $n_0 = n$ e $n_L = p$. Cada unidade j da camada ℓ se conecta a todas as $n_{\ell-1}$ unidades da camada anterior e recebe como entrada as ativações $\boldsymbol{a}^{[\ell-1]} = \left(\boldsymbol{a}_1^{[\ell-1]},\ldots,\boldsymbol{a}_{n_{\ell-1}}^{[\ell-1]}\right)$ produzidas por elas. A unidade j aplica então uma combinação linear às ativações da camada anterior, computando $\boldsymbol{z}_j^{[\ell]} = \boldsymbol{W}_j^{[\ell]}\boldsymbol{a}^{[\ell-1]} + \boldsymbol{b}_j^{[\ell]}$, sendo $\boldsymbol{b}_j^{[\ell]}$ um termo de *bias*. Por fim, a ativação produzida pela unidade j se obtém por meio da aplicação de uma função não linear $g^{[\ell]}$ à combinação $\boldsymbol{z}_j^{[\ell]}$: $\boldsymbol{a}_j^{[\ell]} = g^{[\ell]}\left(\boldsymbol{z}_j^{[\ell]}\right)$. Essa unidade se encontra representada na Figura 3b.

Dessa forma um MLP define a cadeia de transformações

$$\boldsymbol{a}^{[0]} = \boldsymbol{x}, \quad \boldsymbol{z}^{[\ell]} = \boldsymbol{W}^{[\ell]}\boldsymbol{a}^{[\ell-1]} + \boldsymbol{b}^{[\ell]}, \quad \boldsymbol{a}^{[\ell]} = g^{[\ell]}\left(\boldsymbol{z}^{[\ell]}\right), \quad \hat{\boldsymbol{y}} = \boldsymbol{a}^{[L]}, \tag{1}$$

com $\boldsymbol{W}^{[\ell]} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ e $\boldsymbol{b}^{[\ell]} \in \mathbb{R}^{n_\ell}$. A função $g^{[\ell]}$, por sua vez, é conhecida como função de ativação da camada ℓ, e é aplicada a $\boldsymbol{z}^{[\ell]}$ elemento a elemento, resultando na ativação $\boldsymbol{a}^{[\ell]}$.

Figura 3 – Ilustrações (a) de um MLP e (b) de uma unidade.



(a) Fonte: adaptado de Hobold e da Silva [15].

(b) Fonte: dos autores.

Diversas funções de ativação podem ser utilizadas. Algumas das mais difundidas são a linear, a unidade linear retificada (ReLU – do inglês, *rectified linear unit*), a sigmoidal e a tangente hiperbólica, expostas na Tabela 1. Com isso, as camadas ocultas aprendem relações não lineares entre as suas entradas, o que é conhecido como extração de atributos [23].

Tabela 1 – Funções de ativação.

Nome	Definição
Linear	x
Unidade linear retificada	$\text{ReLU}(x) = \max\{0, x\}$
Sigmoidal	$\sigma(x) = \left(1 + \exp(-x)\right)^{-1}$
Tangente hiperbólica	$\tanh(x) = 2\sigma(2x) - 1$

Fonte: dos autores.

Para o treinamento da rede, é escolhida uma função custo J a ser minimizada. O treinamento consiste em buscar os parâmetros $\boldsymbol{W}^{[\ell]}$ e $\boldsymbol{b}^{[\ell]}$ que minimizam essa função custo. Com esse fim, algoritmos baseados em gradientes, como o método do gradiente estocástico (SGD – do inglês, *stochastic gradient descent*) e o Adam são aplicados para a otimização. Nessa situação, o método de *backpropagation* surge como um algoritmo eficiente para o cálculo do gradiente da função custo por meio da utilização da regra da cadeia aplicada à Equação (1). Mais detalhes podem ser encontrados em Bishop [3], Murphy [23] e Goodfellow; Bengio e Courville [13]. De acordo com Goodfellow; Bengio e Courville [13], o poder dos redes neurais (NNs – do inglês, *neural networks*) advém do uso das funções de ativação não lineares, e do algoritmo de *backpropagation*.

2.3 redes neurais convolucionais

Redes neurais convolucionais são um tipo de redes neurais especializado no processamento de imagens ou dados com estrutura similar, e têm encontrado inúmeras aplicações práticas. Essas redes se baseiam em camadas de convolução, que aplicam transformações matriciais especializadas, ou filtros, a pixels localizados próximos uns aos outros.

De acordo com Goodfellow; Bengio e Courville [13], a operação de convolução em uma camada de uma redes neurais convolucionais é definida como a correlação cruzada entre a ativação da camada anterior, $\boldsymbol{A}^{[\ell-1]}$ e um *kernel* $\boldsymbol{K}^{[\ell]}$. Nota-se aqui que $\boldsymbol{A}^{[\ell-1]}$ e $\boldsymbol{K}^{[\ell]}$ são escritos em forma tensorial. Com isso, obtém-se a saída $\boldsymbol{Z}^{[\ell]}$:

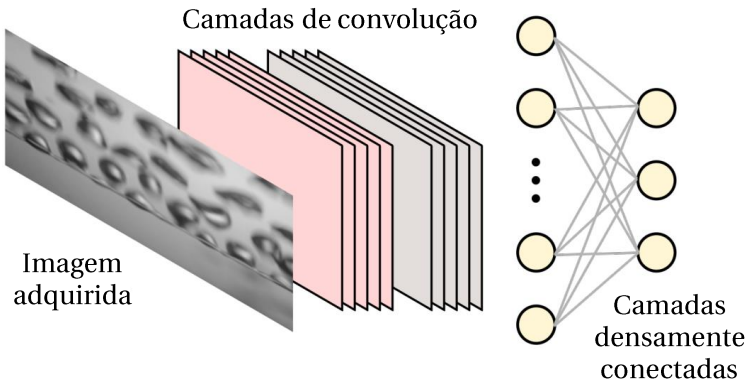
$$\boldsymbol{Z}^{[\ell]}_{(i,j)} = \sum_p \sum_q \boldsymbol{A}^{[\ell-1]}_{(i+p,j+q)} \boldsymbol{K}^{[\ell]}_{(p,q)}, \tag{2}$$

sendo que o subescrito (i,j) denota o elemento da i-ésima linha e j-ésima coluna.

Após a operação de convolução, é aplicada uma função de ativação elemento-a-elemento à saída $\boldsymbol{Z}^{[\ell]}$ para se obter a ativação $\boldsymbol{A}^{[\ell]}$. A ReLU é a função de ativação mais comumente aplicada. Após a operação de convolução, é comum ainda aplicar uma operação de *pooling*, em que regiões retangulares de $\boldsymbol{A}^{[\ell]}$ são mapeadas em um único número real, reduzindo assim a dimensão da informação. Exemplos de camadas de *pooling* são a *max pooling* e a *average pooling* [13].

Dessa forma, CNNs são capazes de encontrar relações entre pixels vizinhos em imagens, e assim identificarem padrões mais complexos e de forma mais eficiente que redes neurais convencionais. Na Figura 4 se encontra representada uma rede neural convolucional, incluindo uma imagem de entrada, uma sequência de camadas de convolução, seguidas de camadas densamente conectadas, ou seja, uma NN convencional.

Figura 4 – Ilustração de uma CNN.



Fonte: adaptado de Hobold e da Silva [15].

Como pode ser verificado, as camadas mais iniciais de CNNs geralmente aprendem atributos de baixo nível, como bordas, sombras e conjuntos de cores. À medida que a informação avança pela rede, os filtros extraem conteúdo progressivamente mais abstrato dos

dados. Assim, observa-se que as primeiras camadas de uma CNN aprendem atributos mais simples e genéricos – e que provavelmente ocorrem em uma ampla gama de problemas –, ao passo que as camadas posteriores aprendem atributos gradativamente mais complexos e específicos. Sendo assim, diversos trabalhos como Yosinski *et al.* [38] e Oquab *et al.* [25] demonstraram os benefícios da transferência de aprendizado, uma técnica em que uma CNN profunda é pré-treinada em um conjunto de dados extenso e genérico, antes de ser efetivamente treinada no conjunto de interesse. Dessa forma, as camadas iniciais da rede utilizam o conjunto de pré-treinamento para aprenderem os atributos mais genéricos, ao passo que as camadas mais posteriores se especializam no problema em questão. O que se observa é que esse procedimento acelera significativamente o treinamento do modelo, além de potencialmente reduzir o erro de generalização [38, 25, 13]. A técnica de transferência de aprendizado já é amplamente usada para o pré-treinamento de CNNs, e existem conjuntos de dados livremente acessíveis [7] e disponíveis para uso [6].

2.4 MÉTRICAS DE DESEMPENHO

Em problemas de classificação, existem diversas métricas bem estabelecidas para avaliar o desempenho de modelos. Dentre elas, destacam-se a precisão P, a revocação R e a acurácia A. Denotando por TP, FP as taxas de verdadeiros e falsos positivos, e TN, FN as taxas de verdadeiros e falsos negativos, a precisão, a revocação e a acurácia associados a uma classe são definidas como

$$P = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \quad R = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}, \quad \text{e} \quad A = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}. \tag{3}$$

Além disso, a acurácia pode ser estendida para o conjunto de dados inteiro como a razão entre a quantidade de classificações corretas e o número total de amostras. Um modelo perfeito obteria $P = R = A = 1$, mas isso não costuma ocorrer na prática. Ainda assim, essas métricas não são suficientemente robustas. Por exemplo, um modelo que classifique qualquer imagem como "camiseta" obterá uma precisão máxima $P = 1$ com relação à classe "calça" pois não haverá nenhum falso positivo, ou seja, $\mathrm{FP} = 0$. Nesse caso, porém, a revocação será baixa. A fim de considerar ambas as métricas simultaneamente, utiliza-se o *score* F_β, definido como

$$\mathrm{F}_\beta = \left(1+\beta^2\right)\left(P^{-1}+\beta^2 R^{-1}\right)^{-1}. \tag{4}$$

Quanto maior o parâmetro β, maior é a importância de R em relação a P. O caso $\beta = 1$ corresponde ao *score* F_1, a média harmônica entre a precisão e a revocação.

Uma explicação mais detalhada sobre essas métricas e outras utilizadas na mensuração do desempenho no treinamento, na validação e no teste de modelos pode ser encontrada em Murphy [23], Pedregosa *et al.* [27] e Goodfellow; Bengio e Courville [13].

2.5 CLASSIFICAÇÃO DESBALANCEADA

A função custo mais comumente utilizada em classificação binária é a entropia cruzada, definida por

$$J = -\sum_{i=1}^{m} \left[y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right) \right]. \tag{5}$$

De forma geral, a função custo pode ser vista como o acúmulo de uma perda L associada a cada par $\left(y^{(i)}, \hat{y}^{(i)}\right)$:

$$J = \sum_{i=1}^{m} L\left(y^{(i)}, \hat{y}^{(i)}\right). \tag{6}$$

Em um problema de classificação desbalanceada, ao menos uma das classes possui mais exemplos no conjunto de dados que as demais. Como consequência, o modelo tende a predizer a classe mais frequente. Por exemplo, para um conjunto de dados $\mathscr{D}$ composto por 99,5 % de pessoas saudáveis e 0,5 % de pacientes com câncer, um modelo constante que estime unicamente a classe majoritária, i.e. $\hat{y}$ = "saudável", obterá uma acurácia de 99,5 %. Isso é indesejável em diversas situações práticas, como na detecção de câncer exemplificada.

Essa dificuldade pode ser contornada ao se introduzirem pesos distintos para as perdas das classes positiva e negativa. Seja a função característica

$$\mathbb{1}_{[expr]} = \begin{cases} 1, & \text{se a expressão } expr \text{ for verdadeira,} \\ 0, & \text{se } expr \text{ for falsa,} \end{cases} \tag{7}$$

e definindo 1 como a classe positiva e 0 como a negativa, a função custo ponderada é dada por

$$J = \sum_{i=1}^{m} \left[\mathbb{1}_{[y^{(i)}=1]}\, w_{\mathrm{P}} + \mathbb{1}_{[y^{(i)}=0]}\, w_{\mathrm{N}} \right] L\left(y^{(i)}, \hat{y}^{(i)}\right), \tag{8}$$

sendo w_{P} o peso associado à classe positiva, e w_{N}, à classe negativa. Ainda, visto que a multiplicação por uma constante positiva não altera os valores dos mínimos de uma função, pode-se dividir a Equação (8) por w_{N} para se obter

$$J = \sum_{i=1}^{m} \left[\mathbb{1}_{[y^{(i)}=1]}\, w + \mathbb{1}_{[y^{(i)}=0]} \right] L\left(y^{(i)}, \hat{y}^{(i)}\right), \tag{9}$$

sendo $w = w_{\mathrm{P}} / w_{\mathrm{N}}$ o peso das classificações positivas relativo ao peso das negativas. Esse é o peso considerado nas implementações.

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados trabalhos relacionados ao que os autores desenvolveram neste projeto. Em particular, é exposta uma revisão de literatura, incluindo conjuntos de dados disponíveis e publicações acadêmicas.

3.1 REVISÃO DE LITERATURA

Para a classificação de imagens de roupa, métodos como Random Forest [4] e SVMs [10] foram inicialmente propostos, atingindo acurácias pouco impressionantes, porém expressivas. A verdadeira popularização de soluções para problemas multirrótulos só ocorreu após a introdução de métodos mais profundos e robustos como as CNNs [39] e suas ramificações baseadas em mecanismos de atenção provenientes de RNNs [37].

Propostas inicias partiam de *datasets single-label* [35] para treinar redes classificadoras *multi-label* com diversas camadas de ativação, já demonstrando maior eficácia quando comparado a modelos não convolucionais. Apesar desta solução ser capaz de detectar diversos objetos em uma imagem, a relação entre eles é ignorada, algo não desejado para o caso de roupas, já que atributos devem estar associados a algum objeto.

Esta exigência acabou trazendo maior atenção acadêmica para a classificação *multi-label* de roupas, transformando ela em um *case* para o emprego de CNNs baseadas em um *dataset* de roupas plenamente *multi-label* [18], através da associação de três conjuntos de dados *single-label*.

O principal *tradeoff* destas novas e mais precisas técnicas foi o aumento do uso computacional para o treinamento e teste das redes, o que foi posteriormente abordado por trabalhos *multi-label* e *multi-task* [1].

Com métricas e propostas de solução bem definidas, a classificação multirrótulo de roupas foi progressivamente recebendo mais contribuições. *Datasets* de roupas nativamente *multi-label* surgiram [29], contando com um número de imagens cada vez maior [21], menor esforço humano envolvido [16] e melhor qualidade de anotações [31].

Com base na literatura recente sobre *Attention Mechanisms* integrados a redes convolucionais classificadoras de roupa [32, 20, 8], pode-se afirmar que o uso de CNNs como VGG16 e ResNet50, em conjunto com as técnicas de extração de parâmetros como *landmarks* e *feature maps*, está na fronteira do estado da arte para solucionar o problema abordado neste projeto.

Por meio de buscas acadêmicas, as seguintes referências foram compiladas por sua relevância para a aplicação proposta:

- **Bossard *et al.* [4] (2012)** “Apparel classification with style”. Foi um dos pioneiros na introdução de atributos múltiplos. O núcleo do método consiste em um classificador multiclasse baseado em uma Random Forest que usa *learners* fortemente discrimi-

nativos como nós de decisão. Para tornar a *pipeline* o mais automático possível, sua solução integra dados de treinamento rastreados automaticamente da internet no processo de aprendizado. Para avaliação, definiram-se 15 classes de roupas e introduziu-se um conjunto de dados de referência para a tarefa de classificação de roupas que consiste em mais de 80 000 imagens disponibilizadas ao público. Para cada imagem, o algoritmo seleciona uma janela de interesse que contenha a parte da roupa que está sendo classificada.

- **Wei *et al.* [35] (2014)** "CNN: Single-label to multi-label". Neste artigo, os autores propõem uma solução para o problema de classificação *multi-label* utilizando a solução para a *single-label*. Para isso, é aplicada a técnica BING para a detecção de objetos nas imagens. Cada objeto corresponde a uma hipótese que se faz quanto às classes presentes na imagem, sendo então classificado em um problema multiclasse por uma rede convolucional.A solução do problema *multi-label* é a união de todas as classificações *single-label*.

- **Lao e Jagadeesh [19] (2016)** "Convolutional neural networks for fashion classification and object detection". Um dos primeiros trabalhos dedicados à classificação mais completa de peças de roupa. Escrito por alunos de uma disciplina similar em Stanford, o documento enfatiza a visão macro dos problemas enfrentados e, em diversos quesitos como conjunto de dados, arquitetura de rede e infraestrutura empregada, esse artigo já está ultrapassado.

- **Inoue *et al.* [16] (2017)**: "Multi-label Fashion Image Classification with Minimal Human Supervision". Para a geração do conjunto de dados Fashion550k, os autores desse trabalho utilizaram a arquitetura ResNet-50, disponível na biblioteca Keras e pré-treinada no conjunto ImageNet, finalizada com uma camada de 66 unidades sigmoidais, cada uma capaz de prever a probabilidade de uma das 66 classes do problema multirrótulo. Técnicas auxiliares foram aplicadas para se filtrarem ruídos nos rótulos das imagens no conjunto de treinamento, mas isso está fora do escopo deste trabalho.

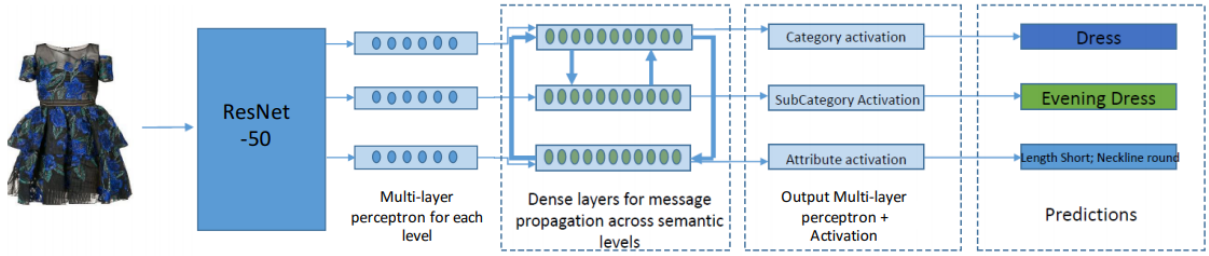
- **Wang *et al.* [32] (2018)**: "Attentive Fashion Grammar Network for Fashion Landmark Detection and Clothing Category Classification". Nesse artigo, é projetada uma arquitetura de rede para a classificação multiclasse, com foco em características concretamente visuais, e pouco baseadas em estilo em si. Os princípios das LSTMs é expandido para empregar também blocos de *attention* e *transformers*. Apesar de ser uma abordagem interessante para a solução do problema, podendo trazer resultados promissores, adotar os passos descritos neste artigo elevaria muito a necessidade computacional da rede, ficando fora do escopo deste projeto.

- **Ferreira *et al.* [8] (2018)**: "A unified model with structured output for fashion images classification". Publicação do ISR (Instituto Superior Técnico, Universidade de Lisboa, Portugal) em parceria com a marca de luxo portuguesa Farfetch.

  O conjunto de dados utilizado corresponde a um banco de dados pertencente à empresa Farfetch contendo imagens de *e-commerce* classificadas com cinco níveis de categorização por produto. Não foram utilizados os conjuntos Fashion550k e DeepFashion por conta da limitação ao uso comercial. Por isso, os modelos foram pré-treinados no ImageNet. Os autores desenvolveram a CNN representada na Figura 5. Essa arquitetura inicia-se com uma ResNet-50 disponível na biblioteca Keras e pré-treinada no conjunto ImageNet. Em seguida, o fluxo de informação se divide em três, cada um representando um nível de categorização. Em cada linha, tem-se um perceptron multi-camadas. Para os níveis de categoria e sub-categoria, a rede culmina em uma ativação softmax, já que esses níveis correspondem à classificação multiclasse. Por outro lado, o nível de atributos considera um conjunto de ativações sigmoidais, o que corresponde a uma classificação *multi-label*.

  Já que pode haver uma dependência entre os diferentes níveis de categorização (por exemplo, uma peça só pode pertencer à sub-categoria "vestido para o dia-a-dia" caso corresponder à categoria "vestido"), os autores incluíram um mecanismo de transporte de mensagem nas camadas densas.

Figura 5 – Representação da arquitetura desenvolvida.



Fonte: Ferreira *et al.* [8].

3.2 CONJUNTOS DE DADOS

Após uma revisão dos conjuntos de dados disponíveis gratuitamente, foram levantadas algumas possibilidades, apresentadas nas subseções seguintes.

DeepFashion

Liu *et al.* [21]: "DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations"

O DeepFashion é um conjunto de dados extraídos principalmente dos sites Forever21[1] e Mogujie[2], bem como resultados de buscas no Google Imagens. Ao total, tem-se 800 000 imagens classificadas em 50 categorias humanamente adquiridas e 1000 atributos descritivos. Esse conjunto se encontra exemplificado na Figura 6.

Figura 6 – Exemplos de imagens contidas no conjunto DeepFashion.

Fonte: Liu *et al.* [21].

Cada imagem possui, em média, de 4 a 8 atributos. Esse *dataset* também introduz *benchmarks*. Sobre esse *dataset*, os desenvolvedores construíram a FashionNet, um grupo de 3 CNNs aplicadas para a recomendação de *looks*. Em termos de dados, as imagens possuem tamanhos diferentes. O lado maior de cada imagem foi fixado em 300 p, mantendo a razão de aspecto da fonte. O subconjunto próprio para a aplicação descrita neste documento é o DeepFashion: Attribute Prediction, o qual contém 289 222 imagens de roupa com um total de 1000 atributos.

Fashion550k

Inoue *et al.* [16]: "Multi-label Fashion Image Classification with Minimal Human Supervision"

O conjunto de dados Fashion550k é composto por 550 661 exemplos, cada um contendo uma imagem e um conjunto de rótulos associados, com exemplos expostos na Figura 7. Esses dados foram extraídos de postagens encontradas na internet provenientes de diversas

[1] https://www.forever21.com/us/shop
[2] http://mogu-inc.com/

partes do mundo, sendo assim irrespectivas da sua localização. Existem 66 classes de produtos possíveis, sendo que uma imagem pode pertencer a mais do que uma classe, e os rótulos do conjunto de dados foram extraídos automaticamente das publicações, contendo um ruído significativo. Por isso, os autores rotularam manualmente um subconjunto dos dados, e treinaram um modelo nesse subconjunto para aprender a remover o ruído. O conjunto de dados final foi então obtido por meio da aplicação desse modelo ao restante do conjunto.

Figura 7 – Exemplos de imagens contidas no conjunto Fashion550k.

Fonte: Inoue *et al.* [16].

3.3 SOLUÇÕES DISPONÍVEIS

Foram identificadas três empresas que oferecem serviços de *API*s para uso de modelos classificadores prontos para peças de roupa: Markable, Imagga e Ximilar.

A intenção inicial da equipe era utilizar as amostras grátis destes serviços para comparação em nível de mercado do modelo entregue por este projeto e estimativa monetária modelo final desenvolvido. Por outro lado, como os testes dos modelos gerados acabaram sendo validados em maior escala e de maneira automática pelo *Google Colab*, o *benchmark* fiel com estes serviços se provou uma tarefa pouco fundada em métricas e pouco prática.

Quanto à estimativa de custo, foi possível compilar a seguinte precificação para o uso dos modelos classificadores disponíveis no mercado.

Tabela 2 – Preço mensal dos serviços concorrentes

Provedor	Requisições mensais	Preço [R$]
Ximilar	100 000	280
Ximilar	1 000 000	2320
Imagga	12 000	60
Imagga	70 000	335
Imagga	300 000	1500

Fonte: dos autores.

4 METODOLOGIA

Neste capítulo é definida a metodologia empregada a fim de se alcançarem os objetivos propostos. Por conta destes, o trabalho contou com duas frentes de desenvolvimento paralelas, tendo em vista uma análise mais ampla.

As redes-base utilizadas para o desenvolvimento deste trabalho contam com diversas transformações intensivas em recursos computacionais. Adicionalmente, os conjuntos de dados atualmente utilizados em modelos performáticos são muito extensos. Em decorrência disso, foram adotadas certas etapas de pré-processamento, bem como simplificações estratégicas para adequar os objetivos às limitações impostas.

4.1 IMPLEMENTAÇÃO

A implementação foi completamente realizada na linguagem Python 3.

O projeto computacional desenvolvido para alcançar o primeiro objetivo definido por este trabalho (rede complexa, se aproximando do estado da arte) será subsequentemente referenciado como P-Net, enquanto o código que trata do segundo objetivo (rede mais simples, utilizando maior número de arquiteturas) será denominado K-Net. Por simplificação, uma rede será denotada por K-Net(*base net*, E, w, N) quando possuir uma arquitetura igual a *base net*, for treinada por E épocas com um peso w e sobre um conjunto de N imagens. Por exemplo, a rede K-Net(VGG16, $E = 20$, $w = 100$, $N = 10\,000$) possui a arquitetura VGG16 e foi treinada por 20 épocas com 10 000 imagens, aplicando um peso de 100 sobre as classificações positivas.

Para otimizações e processamento de dados, bibliotecas como Scikit-learn (0.21.3), Pandas (0.25.3) e Numpy (1.17.4) foram empregadas em ambos os códigos, todas elas em suas versões mais recentes (02/12/2019) enquanto as redes foram construídas com as duas bibliotecas líderes do mercado de redes neurais para *deep learning*.

As redes K-Net foram completamente desenvolvidas utilizando a interface integrada de desenvolvimento Google Colab utilizando funções e redes-base do Tensorflow 1.15.0 e Keras 2.2.5.

Já a rede P-Net foi desenvolvida pela interface do Visual Studio Code 1.40.1 com a biblioteca Pytorch 1.3.1 [26] para Cuda 10.1, contando com uma estrutura clássica de projetos em Python - modularização de funções e variáveis em diversos arquivos com execução por terminal de comandos Unix.

4.2 RECURSOS

As duas redes foram treinadas completamente utilizando a infraestrutura em nuvem da GCP (Google Cloud Platform).

A K-Net teve acesso às GPUs disponibilizadas gratuitamente pelo Google Colab, as

quais claramente aceleraram o processamento apesar de não possuírem especificações e custos claros.

Já a rede P-Net usufruiu de todo o crédito de avaliação cedido a novos usuários da GCP (U$300). Para sua implementação, foram criadas três máquinas virtuais com as especificações da Tabela 3.

Tabela 3 – Máquinas virtuais construídas para a P-Net

Nome	GPU	Núcleos CPU	RAM	Disco	Sistema Operacional
nvidia1	Tesla V100	8	26 GB	32 GB SSD	Ubuntu 18.04
nvidia2	Tesla P100	16	40 GB	32 GB SSD	Ubuntu 18.04
cpu1	-	8	20 GB	32 GB SSD	Debian 10.2

Fonte: dos autores.

Ambas as GPUs utilizadas são especializadas em aplicações relacionadas a inteligência artificial, sendo as máquinas virtuais *nvidia1* e *nvidia2* otimizadas para a execução de códigos em PyTorch pela distribuição AISE PyTorch NVidia GPU Production da Jetware [17]. Elas são consideradas componentes de alta performance no nicho de redes neurais, apresentando preços condizentes com esse fato - em média U$3200 para a NVIDIA Tesla P100 e U$10 000 para a NVIDIA Tesla V100. Uma comparação mais detalhada entre as GPUs é apresentada na Figura 8.

Figura 8 – Comparação entre as GPUs V100 e P100.

Processor	SMs	CUDA Cores	Tensor Cores	Frequency	TFLOPs (double)	TFLOPs (single)	TFLOPs (half/Tensor)	Cache	Max. Memory	Memory B/W
Nvidia P100 PCIe (Pascal)	56	3,584	N/A	1,126 MHz	4.7	9.3	18.7	4 MB L2	16 GB	720 GB/s
Nvidia V100 PCIe (Volta)	80	5,120	640	1.53 GHz	7	14	112	6 MB L2	16 GB	900 GB/s

Fonte: Xelerit [36].

4.3 CONJUNTO DE DADOS

Este projeto utilizou uma partição do subconjunto de imagens do DeepFashion denominado DeepFashion: Category and Attribute Prediction. Para as anotações, foram empregadas no treinamento os subconjuntos DeepFashion: Category and Attribute Prediction e DeepFashion: Fashion Landmark Detection Benchmark.

Os criadores do DeepFashion particionaram subconjuntos com base em cada tipo de aplicação. Portanto a diferença entre os dois segmentos utilizados está apenas nos dados das anotações havendo completa intersecção entre as imagens de cada grupo.

A extensão de dados agregados a cada imagem trazida pelas anotações de *landmarks* permite o desenvolvimento de redes mais completas para a classificação de diversos atributos. Esta vantagem vem com um *tradeoff* de exigir mais recursos computacionais, bem como pior qualidade de resultados em menores conjuntos de treinamento.

Para o treinamento e validação, foram utilizados de 5000 a 200 000 exemplos aleatoriamente extraídos do subconjunto de imagens. Dessa seleção, 60 % das imagens são destinadas ao treinamento, 20 %, à validação e 20 %, ao teste.

As dimensões das imagens no conjunto de dados não são padronizadas. Por isso cada arquivo passa por um *pipeline* de *data augmentation* composto por uma etapa de normalização da escala de cores RGB de $[0, 225]$ para $[0, 1]$ a fim de estabilizar o algoritmo de otimização. Posteriormente, as imagens são redimensionadas para 224×224 p por meio de deformação. Tais dimensões são compatíveis com aquelas aceitas pelas redes convolucionais empregadas neste projeto.

Os pesquisadores por trás do DeepFashion disponibilizaram todas as anotações no formato `.txt`, o que resultou em arquivos com até 750 MB. Assim, os arquivos de anotações também tiveram que ser processados em partes para adequação em cada tipo de implementação.

Para as K-Nets, os arquivos de anotações foram processados tendo em mente uma otimização na carga da memória, cuja limitação impede que sejam carregados em um único momento.

Já para a P-Net, um arquivo denominado `info.csv` foi gerado pela compilação de informações de categoria, atributos e *landmarks*, bem como caminho no sistema e legendas de cada campo, para todas as imagens de produtos. Tal lista ocupa cerca de 500 MB em disco e, apesar da baixa taxa de compressão, a simplificação trazida nas leituras e acessos se mostrou benéfica.

4.4 ARQUITETURA DE REDES

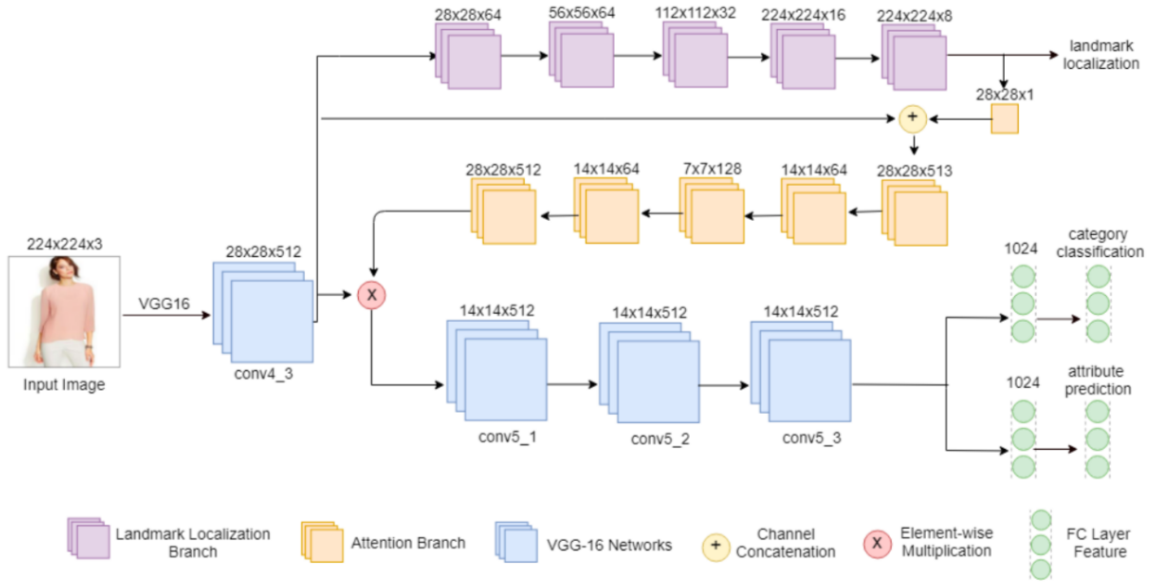
A arquitetura por trás da P-Net foi baseada na rede apresentada no trabalho de Liu e Lu [20] e ilustrada na Figura 9

A rede desenvolvida em PyTorch é uma extensão da rede-base VGG16 [30] para a classificação multirrótulos (como cor, textura e coleção de uma peça) e multiclasses – predizendo tanto a categoria de um produto, por exemplo “blusa”, quanto atributos.

Uma nova rede baseada nos pontos marcantes mais influentes na caracterização de uma peça de roupa - os *landmarks* - foi definida. No trabalho original de Liu e Lu [20], todas as operações vetoriais de dados era feito por meio da biblioteca Numpy, a qual ainda não possui suporte para *Cuda Tensors* e, consequentemente, para as GPUs utilizadas.

Perante este contratempo, houve diversas tentativas de implementação do trabalho desenvolvido em Nishino e Loomis [24], o Cupy, uma biblioteca de sintaxe similar ao Numpy, porém com suporte para operações em CUDA. Por conta de problemas de hierarquia admi-

Figura 9 – Arquitetura utilizada para a P-Net



Fonte: Liu e Lu [20].

Figura 10 – Arquitetura introduzida pela VGG16

Fonte: Simonyan e Zisserman [30].

nistrativa e versionamento de componentes nas máquinas virtuais da Google, não se obteve êxito.

Alternativamente, recursos nativos do PyTorch para o manejamento e direcionamento de tensores no sistema foram utilizados com sucesso.

Com a rede de *landmarks* funcional, sua predição é, em certos casos, ignorada para a adequação ao escopo deste trabalho, porém seu impacto na porção atentiva do restante da rede não pode ser desconsiderado. Essa retroalimentação na etapa de mecanismos de atenção de atributos, bem como o multiplicação elementar entre os mapas subsequentes da VGG são os refinamentos que serão avaliados.

Para isso, a função de custo recebeu diferentes pesos para cada uma das três ramificações da rede, permitindo que o impacto de cada tipo de dado – localização de *landmarks*, classificação de categorias (rótulo único) e classificação de atributos (multirrótulo) – seja ajustado. Em certos modelos, foram definidos pesos mínimos para as redes não relacionadas

aos atributos, para fins comparativos.

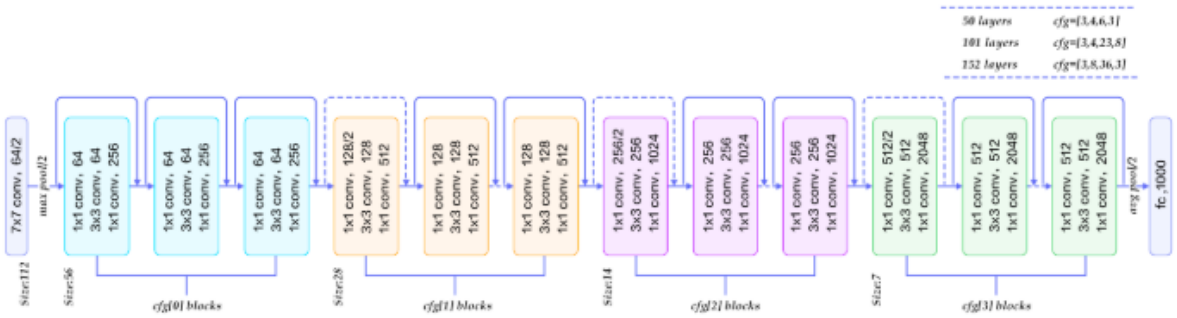
Similarmente, um peso menor para atributos negativos foi definido para adaptar as redes-base ao desbalanceamento do problema. Visto que cada imagem possui, em média, de 4 a 8 atributos dentro de um total de 1000 possibilidades, as classes negativas são muito mais frequentes que as negativas. Assim sendo, a ordem de magnitude do peso w se torna outro parâmetro analisado neste trabalho.

No código são definidos avaliadores distintos para as redes VGG16 principal (categorizadoras de categoria e atributos) e para a rede de *landmarks* (localizadora de pontos-chave).

As demais classes – geradoras de camadas, mapas gaussianos e parâmetros de treinamento – foram mantidas exatamente iguais às especificações de Liu e Lu [20]. Isso ocorre pois diversos outros parâmetros da redes diferem, como número, dimensões e operações das camadas.

As redes K-Net consideraram, além da VGG16, as redes-base ResNet50 [14] e MobileNetV2 [28] com o intuito de comparar a taxa de erro entre elas.

Figura 11 – Arquitetura introduzida pela ResNet50



Fonte: He *et al.* [14].

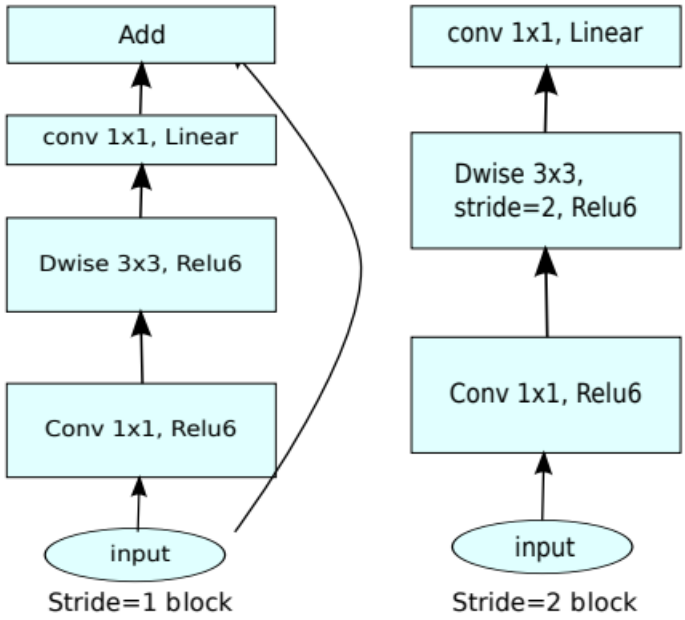
4.5 DEFINIÇÃO DE CASOS

Como função custo a ser minimizada, foi utilizada a entropia cruzada para classificação binária ponderada, com o peso das classes negativas fixado em $w_N = 1$. Sendo assim, o único parâmetro de peso a ser estudado é o peso w.

Como parâmetro para avaliação dos modelos, foi utilizada a taxa de erro serial das predições. Ou seja, a performance será associada à razão entre o número absoluto de atributos individuais corretamente previstos e o total de atributos individuais presentes nas imagens avaliadas.

Por meio da literatura [9] [33] [20], o alcance do número de épocas entre 10 e 100 foi escolhido.

Figura 12 – Arquitetura introduzida pela MobileNetV2



Fonte: Sandler *et al.* [28].

Como mencionado anteriormente, a rede de *landmarks* adiciona mais dados que podem vir a auxiliar predições de atributos, porém isso pode também impactar negativamente a acurácia do modelo

Com essa configuração definida, cinco casos de estudo foram considerados:

1. *Simplificação da rede:* comparação do treinamento da P-Net completa com a P-Net simplificada (i.e. VGG16, desconsiderando a componente de predição de *landmarks*). Como hiperparâmetros de treinamento, utilizam-se 20 épocas de treinamento, peso $w = 100$ e um conjunto de 10 000 imagens.

2. *Comparação entre implementações*: para 10 000 imagens, 20 épocas, e peso de $w = 100$ treinar a rede VGG16 utilizando a biblioteca Keras (K-Net) e a PyTorch (P-Net).

3. *Curva de treinamento*: com a P-Net simplificada, 10 000 imagens e $w_{\mathrm{P}} = 100$, executar um treinamento de 100 épocas.

4. *Curva de aprendizado*: com a VGG16, por 20 épocas e $w_{\mathrm{P}} = 100$, treinar a rede com 1000, 5000, 10 000, 50 000, 100 000 e 200 000 imagens. Simplificadamente, treinar K-Net(VGG16, $E = 20$, $w = 100$, N) para $N = 1000$, 5000, 10 000, 50 000, 100 000 e 200 000.

5. *Influência dos pesos*: treinar a VGG16 (K-Net) com 10 000 imagens e em 20 épocas, utilizando como pesos $w = 1$, 10, 100 e 1000. De forma compacta, treinar K-Net(VGG16, $E = 20$, w, $N = 10\,000$) para $w = 1$, 10, 100, e 1000.

6. *Comparação entre arquiteturas*: utilizando a implementação K-Net, para 10000 imagens, 20 épocas, e peso de $w = 100$, variar a arquitetura da rede, testando a VGG16, a MobileNetV2 e a ResNet50V2, todas disponíveis na biblioteca Keras. As redes são inicializadas com os pesos obtidos do seu treinamento sobre o conjunto ImageNet, e todas as unidades são treinadas.

Os parâmetros por trás dos modelos performáticos finais gerados pela P-Net são mostradas na Tabela 4. Os seis modelos explicitados foram planejados com base no tempo de recursos computacionais em nuvem disponível. Analogamente, na Tabela 5 estão expostos os hiperparâmetros para a construção das K-Nets.

Tabela 4 – Modelos gerados pela P-Net.

Nome	Máquina	*N*	Rede *Landmark*	*w*	Épocas	*Batch Size*	*Workers*
exp1	nvidia2	50000	Ausente	100	100	32	16
proj1	cpu1	10000	Presente	10	20	16	4
proj2	nvidia1	10000	Ausente	1000	20	32	16
proj3	nvidia2	10000	Presente	100	20	32	10
proj4	nvidia1	20000	Presente	100	20	40	14
proj5	nvidia1	50000	Presente	200	20	50	10

Fonte: dos autores.

Nas tabelas, *batch size* é o tamanho do *batch* usado no treinamento e na validação e *workers* é a quantidade de processos simultâneos na CPU.

Todos os modelos foram otimizados pelo algoritmo Adam com taxa de aprendizado inicial igual a 1×10^{-4} e hiperparâmetros $\beta_1 = 0{,}9$ e $\beta_2 = 0{,}999$. Esses valores foram extraídos da literatura e, após uma etapa de experimentação preliminar, se mostraram os mais adequados.

O subprojeto P-Net está disponível na internet em Cano [5].

Tabela 5 – Modelos gerados pela K-Net.

Nome	Arquitetura	N	w	Épocas E	*Batch Size*	*Workers*
K-Net(VGG16, $E = 20$, $w = 100$, $N = 10\,000$)	VGG16	10 000	100	20	32	6
K-Net(VGG16, $E = 20$, $w = 1$, $N = 10\,000$)	VGG16	10 000	1	20	32	6
K-Net(VGG16, $E = 20$, $w = 1$, $N = 50\,000$)	VGG16	50 000	1	20	32	6
K-Net(VGG16, $E = 20$, $w = 1$, $N = 200\,000$)	VGG16	200 000	1	20	32	6
K-Net(VGG16, $E = 20$, $w = 10$, $N = 10\,000$)	VGG16	10 000	10	20	32	6
K-Net(VGG16, $E = 20$, $w = 10$, $N = 100\,000$)	VGG16	100 000	10	20	32	6
K-Net(VGG16, $E = 20$, $w = 100$, $N = 100\,000$)	VGG16	100 000	10	20	32	6
K-Net(VGG16, $E = 20$, $w = 1000$, $N = 10\,000$)	VGG16	10 000	1000	20	32	6
K-Net(VGG16, $E = 20$, $w = 1000$, $N = 5000$)	VGG16	5000	1000	20	32	6
K-Net(VGG16, $E = 20$, $w = 1000$, $N = 200\,000$)	VGG16	200 000	1000	20	32	6
K-Net(MobileNetV2, $E = 20$, $w = 100$, $N = 10\,000$)	MobileNetV2	10 000	100	20	32	6
K-Net(ResNet50V2, $E = 20$, $w = 100$, $N = 10\,000$)	ResNet50V2	10 000	100	20	32	6

Fonte: dos autores.

5 RESULTADOS E DISCUSSÕES

Neste capítulo são apresentados os principais resultados obtidos pela equipe por meio da metodologia apresentada.

Inicialmente, a métrica adotada para avaliação era a função perda na validação, mas a taxa de erro se mostrou um dado mais fiel à problemática, visto que essa é a métrica que se busca otimizar.

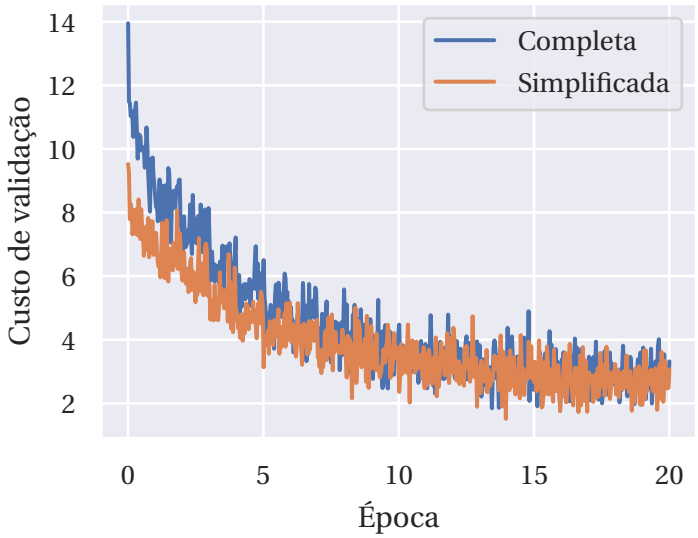
Diversos experimentos foram realizados para determinar os hiperparâmetros adotados. Esse processo foi de vital importância para a eficiência dos modelos gerados, dadas as limitações de tempo e de recursos.

5.1 SIMPLIFICAÇÃO DA REDE

O primeiro caso de estudo é a simplificação da rede implementada em PyTorch. A rede simplificada é treinada desprezando-se o peso da categorização e da localização de *landmarks* sobre a função custo otimizada. Assim, a rede fica impossibilitada de utilizar essas informações adicionais para o seu aprendizado.

Não foi possível implementar em tempo hábil o monitoramento da acurácia sobre o conjunto de validação durante o treinamento. Em vez disso, tem-se o custo sobre o conjunto de validação em função da época de treinamento, representado na Figura 13.

Figura 13 – Comparação entre complexidade de redes P-Net.



Fonte: dos autores.

Como se observa, a rede simplificada, isto é, sem extração de atributos, tem um desempenho ligeiramente superior ao da rede completa, em especial nas primeiras épocas de treinamento. Isso pode ser justificado pois, para poucas épocas e com um conjunto de dados

reduzido, a extração de atributos é prejudicada, e contribui com um valor elevado para a função custo.

5.2 COMPARAÇÃO ENTRE IMPLEMENTAÇÕES

No caso da comparação entre as redes implementadas em Keras e PyTorch, são testadas as redes P-Net e comparadas às redes K-Net compatíves – isto é, treinadas com hiperparâmetros iguais. Valores comparativos se encontram expostos na Tabela 6.

Tabela 6 – Taxa de erro dos modelos gerados pela P-Net.

Modelo	Taxa de Erro
P-Net: exp1	0,00756
P-Net: proj1	0,01912
P-Net: proj2	0,00731
P-Net: proj3	0,00956
P-Net: proj4	0,00901
P-Net: proj5	0,00825
K-Net(VGG16, $E = 20$, $w = 10$, $N = 10000$)	0,00345
K-Net(VGG16, $E = 20$, $w = 100$, $N = 10000$)	0,00336
K-Net(VGG16, $E = 20$, $w = 1000$, $N = 10000$)	0,00339

Fonte: dos autores.

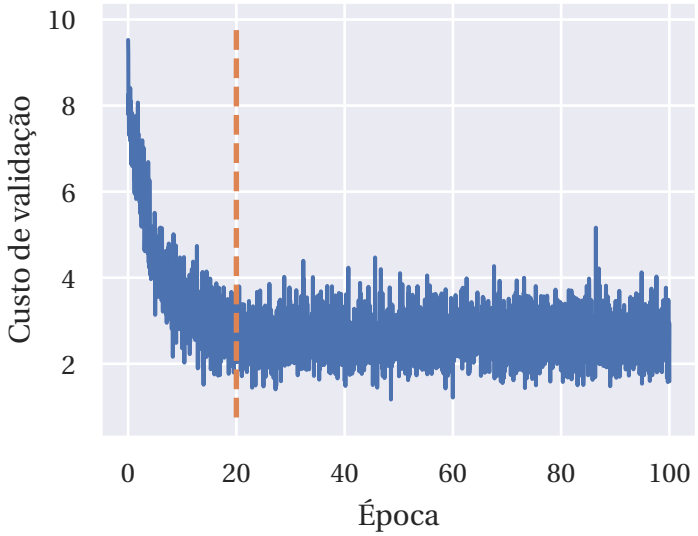
Como se observa, o melhor modelo P-Net possui uma taxa de erro pelo menos duas vezes superior ao pior modelo K-Net. Uma explicação para isso é a possível influência das camadas de detecção de *landmark* presentes nas P-Net. Embora elas possam ser removidas do cálculo da função custo, a sua presença na arquitetura pode dificultar o treinamento das demais camadas, prejudicando o desempenho da rede.

5.3 CURVA DE TREINAMENTO

A fim de se determinar o número de épocas para o treinamento das redes, um estudo sobre a curva de treinamento foi executado. Foi considerada a rede *exp1*, treinada por 100 épocas com peso nas categorias positivas de $w = 100$ sobre 10000 imagens. A curva de treinamento para a perda sobre o conjunto de validação como função da época de treinamento se encontra representada na Figura 14.

Como se observa, a partir de aproximadamente 20 épocas de treinamento, a perda sobre o conjunto de validação se mantém praticamente constante. Como consequência, esse valor foi estabelecido como o limite de treinamento para as demais redes.

Figura 14 – Curva de treinamento da rede *exp1*.

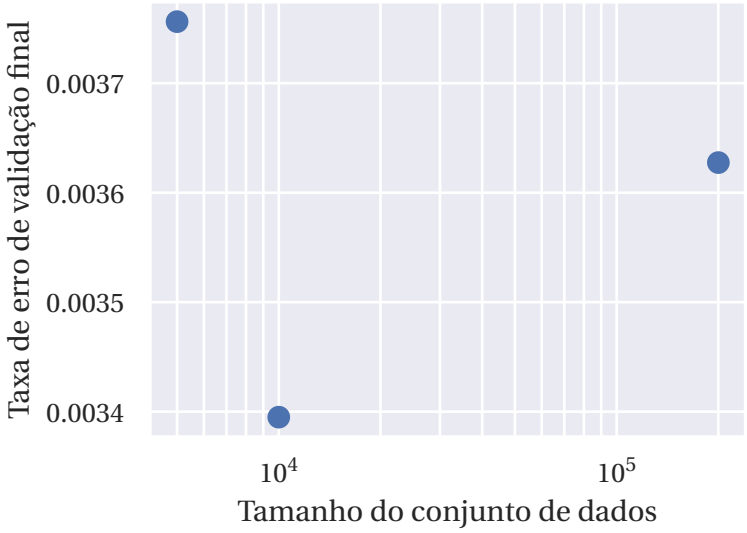


Fonte: dos autores.

5.4 CURVA DE APRENDIZADO

A curva de aprendizado é definida como a métrica de desempenho ao final do treinamento como função do tamanho do conjunto de dados. Em consequência da limitação de tempo, foi possível obter apenas três pontos dessa curva, expostos na Figura 15. Esses pontos foram obtidos para as redes K-Net(VGG16, $E = 20$, w, N).

Figura 15 – Curva de aprendizado obtido no subprojeto K-Net.



Fonte: dos autores.

Como se observa, a taxa de erro sobre o conjunto de validação alcança um mínimo na redondeza de $N = 10\,000$. O primeiro ponto, em $N = 5000$, pode representar uma região

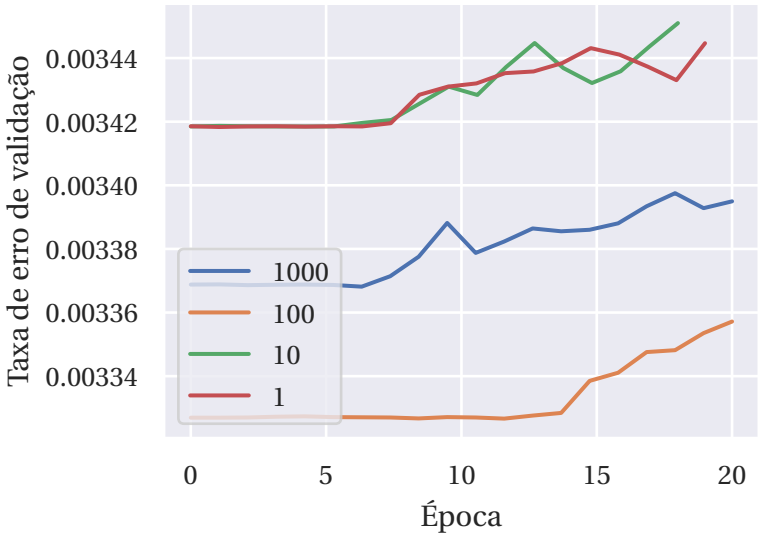
de *underfitting*, na qual, devido à escassez de imagens, o modelo é incapaz de aprender a função desejada. Por outro lado, para $N = 200\,000$, o modelo pode não possuir capacidade de representação suficiente.

O mínimo em torno de 10 000 justifica o uso desse valor como tamanho do conjunto de dados para os demais casos.

5.5 INFLUÊNCIA DOS PESOS

O peso w define o formato da função custo, determinando a importância das categorias positivas em relação às negativas. O resultado obtido se encontra representado na Figura 16.

Figura 16 – Influência do peso relativo na taxa de erro obtida.



Fonte: dos autores.

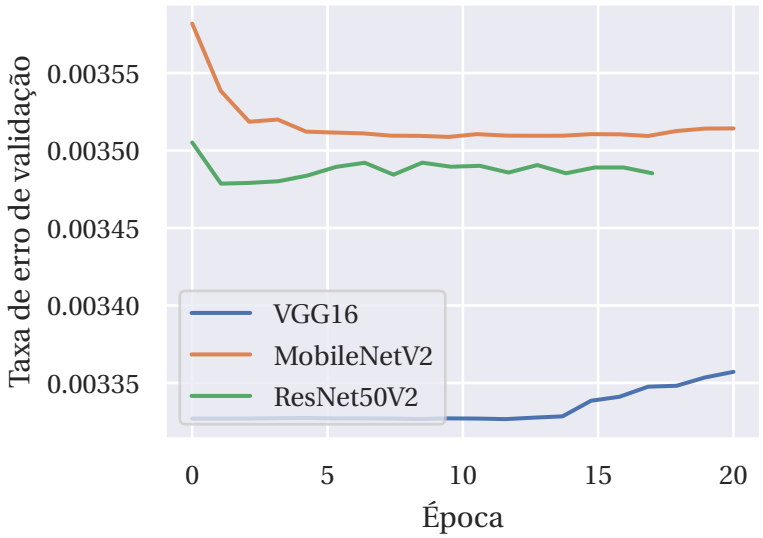
Como observado, a taxa de erro sobre o conjunto de validação começa em um patamar, mas se eleva após $N \cong 10$ épocas. Isso provavelmente significa um sobreajuste, em que a taxa de erro se reduz no conjunto de treinamento, mas se eleva no conjunto de validação. O peso w tem influência não linear sobre a métrica indicada, sendo que $w = 100$ se aproxima da região de mínimo. Pesos inferiores a esse valor favorecem a tendência da rede a predizer apenas a classe negativa, visto que esta é a mais comum. Para pesos elevados, a rede tende a arriscar mais classificações positivas.

5.6 COMPARAÇÃO ENTRE ARQUITETURAS

A comparação entre arquiteturas é uma forma de se avaliar qual é a rede-base mais adequada para a extração de atributos. Essa adequação se dá em termos de velocidade

de processamento, memória RAM consumida, métrica de desempenho obtida e convergência. A curva de treinamento para a K-Net(VGG16, $E = 20$, $w = 100$, $N = 10000$), a K-Net(MobileNetV2, $E = 20$, $w = 100$, $N = 10000$) e a K-Net(ResNet50V2, $E = 20$, $w = 100$, $N = 10000$) se encontra na Figura 17.

Figura 17 – Curvas de treinamento considerando a VGG16, a MobileNetV2 e a ResNet50V2 como redes-base.



Fonte: dos autores.

Durante a execução, observou-se que a MobileNetV2 teve convergência mais de dez vezes mais veloz que a VGG16 e a ResNet50V2. A VGG16 foi a rede que mais consumiu memória RAM, exigindo um *batch size* de 16 imagens, ao passo que, com a ResNet50V2, foi possível utilizar 32, e, com a MobileNetV2, 64. Em contrapartida, a VGG16 obteve maior capacidade de generalização, alcançando taxas de erros sobre o conjunto de validação inferiores às das demais redes.

5.7 MODELOS PERFORMÁTICOS

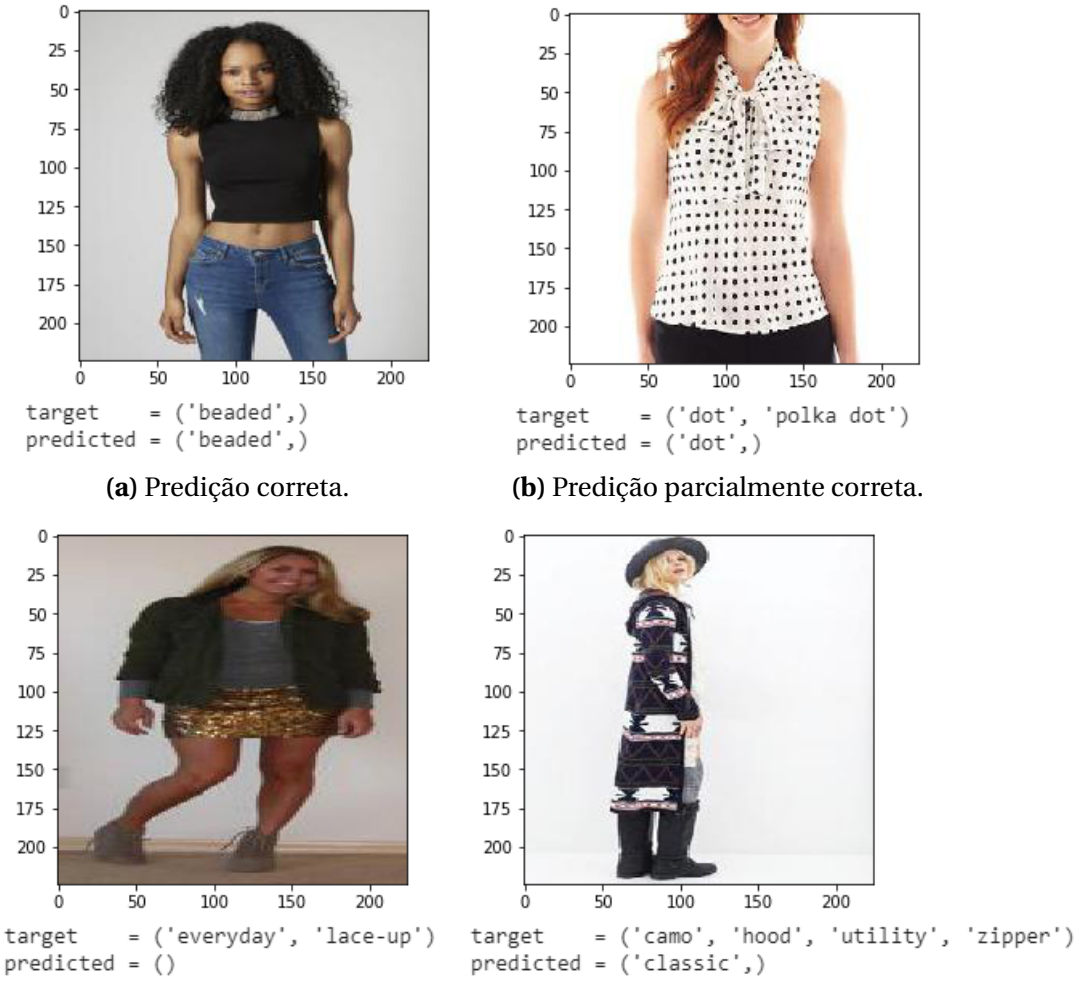
Todos os modelos P-Net foram gerados em três dias. A discrepância entre as máquinas virtuais pode ser claramente observada ao comparar o tempo de execução das redes entre elas. A máquina virtual *cpu1* permaneceu treinando o modelo *proj1* durante todo o período, enquanto que as máquinas equipadas com GPUs NVIDIA foram capazes de treinar diversos modelos.

Pela ampla variação de parâmetros entre as redes, não é possível quantizar, porém a máquina virtual *nvidia1*, a qual contém a NVIDIA Titan V100, claramente se destacou em termos de tempo de execução de recursos e eficiência, apesar do preço elevado.

5.8 PREDIÇÕES DOS MODELOS

Ao se treinar a rede K-Net(VGG16, $E = 20$, $w = 100$, $N = 10\,000$), observa-se que existem quatro tipos de predição: (a) a predição correta, em que, para uma imagem, o modelo prevê todas as categorias corretamente; (b) a predição parcialmente correta, em que o modelo prediz apenas algumas das categorias presentes na imagem; (c) a predição vazia, em que todas as categorias são previstas como pertencentes às classes negativas; (d) e, por fim, a predição errônea, em que o modelo prediz categorias que não existem na imagem. A Figura 18 exemplifica essas predições.

Figura 18 – Exemplos de predições feitas pela rede VGG16 treinada em 20 épocas com 10 000 imagens, peso 100.



(a) Predição correta. **(b)** Predição parcialmente correta.

(c) Predição vazia. **(d)** Predição errônea.

Fonte: dos autores.

6 CONCLUSÃO

O desafio proposto neste trabalho se provou repetidas vezes pouco trivial. A equipe iniciou o projeto com diversas limitações como tempo, recursos computacionais e experiência de nicho, o que exigiu diversos aprendizados, pesquisas e simplificações estratégicas para o concluir.

Apesar disso, os resultados deste projeto foram abundantes, tanto qualitativamente quanto quantitativamente. Diversas hipóteses trazidas pela comunidade acadêmica puderam ser confirmadas, como a importância do peso w e a evolução com o número de épocas. Por outro lado, hiperparâmetros específicos puderam ser adaptados para alcançar a máxima eficiência.

Olhando para trás, muitas das escolhas simples se mostraram mais profundas e impactantes que previstas, como a utilização da rede P-Net completa durante a maior parte do treinamento na GCP. Similarmente, conclui-se que arquiteturas complexas são muito atreladas à quantidade de dados disponíveis, o que acaba concentrando o uso destas a grandes centros de pesquisas e empresas no ramo de computação.

Por outro lado, diversos pontos importantes se mantiveram persistentes perante diferentes complexidades arquitetônicas, o que confirma o valor deste trabalho.

Para futuras desenvolturas na classificação *multi-label* de imagens, os resultados obtidos neste projeto serão de suma importância e, sem dúvida, impulsionarão a geração de novas e mais assertivas conclusões.

Para a equipe, diversas áreas do conhecimento foram exercitadas e desenvolvidas, como arquitetura de computadores, engenharia de dados, álgebra vetorial, algoritmos performáticos, infraestrutura em nuvem, aprendizado de máquinas e, claro, redes neurais.

Por fim, mesmo não atingindo completamente as expectativas apresentadas na proposta inicial do projeto, este trabalho apresentou resultados até então não vistos nas pesquisas bibliográficas realizadas, desenvolveu extensivamente o conhecimento dos alunos envolvidos e, acima de tudo, despertou ainda mais o interesse e o entusiasmo da equipe pela área.

6.1 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

A equipe descobriu, posteriormente aos treinamentos, a razão $w_\mathrm{P}/w_\mathrm{N}$ pode ser definida automaticamente como

$$w = \frac{w_\mathrm{P}}{w_\mathrm{N}} = \frac{\text{Número de classificações negativas}}{\text{Número de classificações positivas}}. \tag{10}$$

A biblioteca Scikit-learn [27] possui implementada a função `sklearn.utils.class_weight.compute_class_weight` que permite o cálculo automático desses pesos. Recomenda-se a utilização dessa função futuramente.

Além disso, a disponibilidade de maior tempo de treinamento das redes poderia ter habilitado a equipe a alcançar melhores acurácias, especialmente com a P-Net completa. Porém isso está atrelado ao dispêndio de recursos em serviços em nuvem.

O trabalho apresentado aqui será prosseguido extensivamente utilizando GPUs físicas durante períodos muito maiores de tempo, além de *clusters* específicos para treinamento de redes neurais da IBM Cloud, parte da premiação do Hackathon Gov SC obtida por um dos membros.

Os aprendizados aqui adquiridos serão vitais para o melhor uso desses recursos.

6.2 *FULL DISCLOSURE*

Este projeto faz parte do TCC do integrante Kauê Cano, orientado pelo Professor Danilo Silva. Os modelos desenvolvidos também servirão de base para entregas da Elint, empresa de um dos integrantes da equipe, para clientes do ramo de fabricação e varejo de roupas femininas.

Apenas os alunos aqui citados estão envolvidos neste projeto.

REFERÊNCIAS

[1] ABDULNABI, Abrar H *et al.* Multi-task CNN model for attribute prediction. **IEEE Transactions on Multimedia**, IEEE, v. 17, n. 11, p. 1949–1959, 2015.

[2] BHARDWAJ, V. e FAIRHURST, A. Fast fashion: response to changes in the fashion industry. *In:* p. 165–173.

[3] BISHOP, Christopher M. **Pattern recognition and machine learning**. [*S.l.*]: springer, 2006.

[4] BOSSARD, Lukas *et al.* Apparel classification with style. *In:* SPRINGER. ASIAN conference on computer vision. [*S.l.*: *s.n.*], 2012. p. 321–335.

[5] CANO, Kauê. **EEL7513**. 2019. Disponível em: `https://github.com/canokaue/EEL7513/`. Acesso em: 6 dez. 2019.

[6] CHOLLET, François *et al.* **Keras**. 2015. Disponível em: `https://keras.io`. Acesso em: 25 out. 2019.

[7] DENG, J. *et al.* ImageNet: A Large-Scale Hierarchical Image Database. *In:* CVPR09. [*S.l.*: *s.n.*], 2009.

[8] FERREIRA, Beatriz Quintino *et al.* A unified model with structured output for fashion images classification. **arXiv preprint arXiv:1806.09445**, 2018.

[9] FERREIRA, Beatriz Quintino *et al.* A unified model with structured output for fashion images classification. **arXiv preprint arXiv:1806.09445**, 2018.

[10] FOODY, G. e MATHUR, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. **IEEE Transactions on Geoscience and Remote Sensing**, p. 1335–1343, jun. 2004. DOI: `10.1109/TGRS.2004.827257`.

[11] FRIDMAN, Lex et al. Large-Scale Naturalistic Driving Study of Driver Behavior and Interaction with Automation, nov. 2017. DOI: `10.1109/ACCESS.2019.2926040`.

[12] GAO, Junfeng *et al.* Computer Vision in Healthcare Applications. *In:* p. 4.

[13] GOODFELLOW, Ian; BENGIO, Yoshua e COURVILLE, Aaron. **Deep learning**. [*S.l.*]: MIT press, 2016.

[14] HE, Kaiming *et al.* **Deep Residual Learning for Image Recognition**. [*S.l.*: *s.n.*], 2015. arXiv: `1512.03385 [cs.CV]`.

[15] HOBOLD, Gustavo M. e DA SILVA, Alexandre K. Analysis of neural network architecture for pool boiling regime identification. *In:* 10TH International Conference on Boiling & Condensation Heat Transfer. Nagasaki, Japan: [*s.n.*], mar. 2018.

[16] INOUE, N. *et al.* Multi-label Fashion Image Classification with Minimal Human Supervision. *In:* 2017 IEEE International Conference on Computer Vision Workshops (ICCVW). [*S.l.*: *s.n.*], out. 2017. p. 2261–2267. DOI: 10.1109/ICCVW.2017.265.

[17] JETWARE. **AISE PyTorch NVidia GPU Production**. 2019. Disponível em: https://console.cloud.google.com/marketplace/details/jetware/pytorch03-python3-cuda91?q=pytorch&id=27714030-3b38-42d3-9239-5a57adbf5c71&project=tactical-patrol-259017. Acesso em: 2 dez. 2019.

[18] LAO, Brian e JAGADEESH, Karthik. Convolutional neural networks for fashion classification and object detection. **CCCV 2015: Computer Vision**, p. 120–129, 2016.

[19] LAO, Brian e JAGADEESH, Karthik. Convolutional neural networks for fashion classification and object detection. **CCCV 2015: Computer Vision**, p. 120–129, 2016.

[20] LIU, Jingyuan e LU, Hong. Deep Fashion Analysis with Feature Map Upsampling and Landmark-driven Attention. *In:* PROCEEDINGS of the European Conference on Computer Vision (ECCV). [*S.l.*: *s.n.*], 2018.

[21] LIU, Z. *et al.* DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations. *In:* 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [*S.l.*: *s.n.*], jun. 2016. p. 1096–1104. DOI: 10.1109/CVPR.2016.124.

[22] MITCHELL, T. M. **Machine Learning**. New York: McGraw-Hill, 1997. (Springer Series in Statistics). ISBN 0070428077.

[23] MURPHY, Kevin P. **Machine Learning**: A Probabilistic Perspective. Cambridge: The MIT Press, 2012. ISBN 978-0-262-01802-9.

[24] NISHINO, ROYUD e LOOMIS, Shohei Hido Crissman. CuPy: A NumPy-Compatible Library for NVIDIA GPU Calculations. *In:* PROCEEDINGS of Workshop on Machine Learning Systems (LearningSys) in the Thirty-first Annual Conference on Neural Information Processing Systems (NIPS). [*S.l.*: *s.n.*], 2017.

[25] OQUAB, Maxime *et al.* Learning and transferring mid-level image representations using convolutional neural networks. *In:* PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [*S.l.*: *s.n.*], 2014. p. 1717–1724.

[26] PASZKE, A. et al. **Pytorch**. 2016. Disponível em: https://pytorch.org. Acesso em: 2 dez. 2019.

[27] PEDREGOSA, F. *et al.* Scikit-learn: Machine Learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

[28] SANDLER, Mark *et al.* **MobileNetV2: Inverted Residuals and Linear Bottlenecks**. [*S.l.*: *s.n.*], 2018. arXiv: 1801.04381 [cs.CV].

[29] SIMO-SERRA, Edgar e ISHIKAWA, Hiroshi. Fashion style in 128 floats: Joint ranking and classification using weak data for feature extraction. *In:* PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition. [*S.l.*: *s.n.*], 2016. p. 298–307.

[30] SIMONYAN, Karen e ZISSERMAN, Andrew. **Very Deep Convolutional Networks for Large-Scale Image Recognition**. [*S.l.*: *s.n.*], 2014. arXiv: 1409.1556 [cs.CV].

[31] UMAASHANKAR, Venkatesh; PRAKASH, Aditi *et al.* Atlas: A Dataset and Benchmark for E-commerce Clothing Product Categorization. **arXiv preprint arXiv:1908.08984**, 2019.

[32] WANG, W. *et al.* Attentive Fashion Grammar Network for Fashion Landmark Detection and Clothing Category Classification. *In:* 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. [*S.l.*: *s.n.*], jun. 2018. p. 4271–4280. DOI: 10.1109/CVPR.2018.00449.

[33] WANG, Wenguan *et al.* Attentive fashion grammar network for fashion landmark detection and clothing category classification. *In:* PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition. [*S.l.*: *s.n.*], 2018. p. 4271–4280.

[34] WANKHEDE, Kirti; WUKKADADA, Bharati e NADAR, Vidhya. Just Walk-Out Technology and its Challenges: A Case of Amazon Go. *In:* DOI: 10.1109/ICIRCA.2018.8597403.

[35] WEI, Yunchao *et al.* CNN: Single-label to multi-label. **arXiv preprint arXiv:1406.5726**, 2014.

[36] XELERIT. **Comparação de modelos NVIDIA Tesla**. 2019. Disponível em: https://www.xcelerit.com/computing-benchmarks/insights/benchmarks-deep-learning-nvidia-p100-vs-v100-gpu/. Acesso em: 2 dez. 2019.

[37] YIN, Wenpeng e SCHÜTZE, Hinrich. Attentive Convolution: Equipping CNNs with RNN-style Attention Mechanisms. **Transactions of the Association for Computational Linguistics**, MIT Press, v. 6, p. 687–702, 2018.

[38] YOSINSKI, Jason *et al.* How transferable are features in deep neural networks? *In:* ADVANCES in neural information processing systems. [*S.l.*: *s.n.*], 2014. p. 3320–3328.

[39] ZARÁNDY, Ákos *et al.* Overview of CNN research: 25 years history and the current trends. *In:* IEEE. 2015 IEEE International Symposium on Circuits and Systems (ISCAS). [*S.l.*: *s.n.*], 2015. p. 401–404.

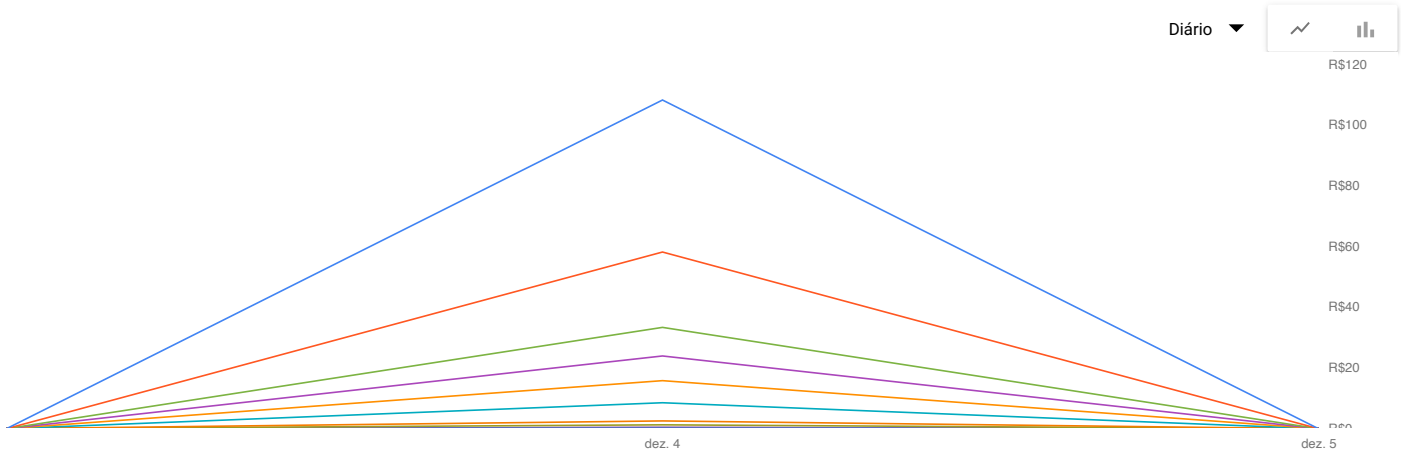
ANEXO A – FATURAMENTO

Conta de faturamento: Minha conta de faturamento (ID da conta: 01FAE7-FBCD1F-B4C3B1)
Intervalo de tempo: 3 – 5 de dezembro de 2019
Granularidade de tempo: Diariamente
Agrupar por SKUs: Todos os SKUs (82)

3 – 5 de dezembro de 2019 (custo total)

R$ 108,12
inclui -R$ 805,82 de crédito

↑ –
R$ 108,12 durante 30 de novembro – 2 de dezembro de 2019



SKU	Produto	Código SKU	Utilização	Custo	Créditos únicos	Descontos	Subtotal
Nvidia Tesla V100 GPU running in Americas	Compute Engine	DC0B-9926-40C5	32,02 hour	R$ 386,03	-R$ 335,77	–	R$ 50,26
Nvidia Tesla P100 GPU running in Americas	Compute Engine	7929-334B-6348	32,04 hour	R$ 227,44	-R$ 202,60	–	R$ 24,84
N1 Predefined Instance Core running in Americas	Compute Engine	2E27-4F75-95CD	768,87 hour	R$ 118,15	-R$ 108,85	–	R$ 9,31
N1 Predefined Instance Core running in Sao Paulo	Compute Engine	E8F8-A4B0-C91F	256,1 hour	R$ 62,47	-R$ 54,35	–	R$ 8,12
N1 Predefined Instance Ram running in Sao Paulo	Compute Engine	E128-228B-EA2A	1.664,68 gibibyte hour	R$ 54,42	-R$ 47,35	–	R$ 7,08
N1 Predefined Instance Ram running in Americas	Compute Engine	6C71-E844-38BC	2.883,28 gibibyte hour	R$ 59,39	-R$ 53,17	–	R$ 6,22
SSD backed PD Capacity	Compute Engine	B188-61DD-52E4	3,85 gibibyte month	R$ 3,18	-R$ 2,07	–	R$ 1,11
SSD backed PD Capacity in Sao Paulo	Compute Engine	28F9-6E36-3030	1,8 gibibyte month	R$ 2,23	-R$ 1,39	–	R$ 0,84
Network Internet Egress from Sao Paulo to EMEA	Compute Engine	D1C6-FB81-E09E	0,3 gibibyte	R$ 0,17	-R$ 0,00	–	R$ 0,17
Network Internet Egress from Americas to EMEA	Compute Engine	DFA5-B5C6-36D6	1,18 gibibyte	R$ 0,44	-R$ 0,27	–	R$ 0,17

Linhas por página: 10 1 – 10 de 67

Subtotal	R$ 108,12
Tributo	–
Total filtrado	R$ 108,12

Projetos incluídos (1)	tactical-patrol-259017
Produtos incluídos (1)	Compute Engine