Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Reconhecimento de Ações Humanas em Imagens com Deep Learning

Lucas Henrique Bahr Yau

Monografia Final
MAC0499 - Trabalho de Formatura Supervisionado

Orientador: Profº. Dr. Roberto Hirata Jr.

São Paulo
2022

I suppose we are no good at facing our memories. We would rather, gild the past. Find something worthwhile among the rubble, and build a future with that.

Rufus Shinra, Final Fantasy VII: Advent Children

Agradecimentos

Existem muitas pessoas as quais eu gostaria de agradecer. Agradeço aos meus pais, aos meus irmãos, aos meus amigos do ensino médio, aos meus amigos da faculdade, aos professores do IME, aos funcionários da USP e a todas as pessoas que, de alguma forma, me ajudaram indiretamente a chegar onde eu estou. Em especial, agradeço ao meu orientador, o Profº. Dr. Roberto Hirata Jr., pela ajuda imensurável durante os meus anos no IME.

Resumo

Lucas Henrique Bahr Yau. **Reconhecimento de Ações Humanas em Imagens com Deep Learning**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2022.

Este trabalho tem como objetivo apresentar, na prática, o uso das técnicas e ferramentas de *deep learning* em um problema de reconhecimento real de imagens, em particular imagens de ações humanas. São descritos, com certa brevidade, os conceitos aprendidos e usados para a realização de um experimento de *deep learning*. As informações sobre cada conceito são curtas, mas suficientes para o entendimento do trabalho. Tais conceitos incluem imagens digitais, aprendizado de máquina e redes neurais convolucionais. Em seguida, é descrita a problemática do reconhecimento de ações humanas, e como o problema tem sido tratado. Por fim, é descrito os aspectos do experimento: escolha e manipulação de um dataset, escolha e adaptação da rede neural VGG16, resultados do treinamento e avaliação do modelo construído.

Palavras-chave: deep learning, reconhecimento de ações humanas, VGG16, redes neurais convolucionais.

Sumário

Introdução

Na contemporaneidade, se faz um uso exaustivo de novas tecnologias para solucionar problemas que beneficiam o ser humano. Um problema em particular é a detecção de ações humanas por meio de imagens, que é objeto de estudo de diversos pesquisadores, devido aos benefícios possíveis que a solução deste problema traz. As mais diversas áreas, como saúde, segurança, transporte, qualidade de vida, bem-estar e gestão governamental, podem ser melhoradas com o auxílio de detecção automática de ações a partir de imagens captadas por câmeras.

Podemos fazer as seguintes perguntas: *o que seria uma ação? Como podemos reconhecê-la?* Para nós, seres humanos, não temos problemas em entender o significado de um movimento do corpo humano. Mas, para um modelo computacional, reconhecer uma ação humana é algo complexo, devido aos diversos aspectos que devem ser levados em conta. Na figura 1, temos alguns exemplos de ações humanas. Da esquerda para a direita, podemos dizer que, na primeira imagem, vemos uma pessoa comendo, porque vemos as mãos da pessoa segurando o que se parece com um alimento e levando-o à sua boca. Na segunda imagem, podemos dizer que a pessoa está dançando *ballet*, por causa do passo característico da dança e pelas vestimentas usadas. Na terceira imagem, podemos dizer que a pessoa está digitando em um smartphone, porque ela está segurando-o com as duas mãos, e olhando o aparelho, mas também poderia estar tirando uma foto.

Conseguimos descrever as ações nas imagens porque conseguimos distinguir vários aspectos presentes nelas. Na segunda imagem, por exemplo, entendemos que se trata de uma pessoa com um vestido branco, realizando um passo de dança do *ballet*, possivelmente em um palco, iluminado por luzes azuis. Pessoas familiares com estes aspectos da imagem podem distingui-la sem problemas.

Mas e no caso dos computadores? Atualmente especialistas em *deep learning* utilizam o modelo computacional de redes neurais, que são treinadas para identificar estes mesmos aspectos que nós usamos para reconhecer uma imagem. Porém, o repertório de aprendizado do computador é pequeno comparado com o repertório de um cérebro humano. A todo momento, adquirimos novas experiências a partir de interações com o mundo, melhorando nosso entendimento sobre determinados assuntos. As imagens da figura 1 são relativamente simples de entender, seja humano ou máquina, devido à clareza da imagem e haver apenas uma pessoa.



Figura 1: Exemplos de ações humanas. Da esquerda para a direita, podemos descrever estas imagens como "comendo", "dançando"e "digitando".

Na figura 2, temos uma complexidade maior, com mais pessoas e/ou ruído na imagem. Na primeira imagem, por exemplo, em que há pessoas em pé e sentadas, não é uma distinção simples para um modelo. Na segunda imagem, apesar da ação estar suficientemente clara para entendimento, o brilho no fundo pode prejudicar o reconhecimento para um modelo.



Figura 2: Exemplos de ações humanas. Diferentemente da figura 1, algumas imagens possuem mais de uma pessoa, além da presença de ruído.

Neste trabalho, é feito um experimento para confirmar a plausibilidade do uso de *deep learning* no reconhecimento de ações humanas em imagens, além da aplicação prática de ferramentas e técnicas aprendidas.

Capítulo 1

Conceitos teóricos

Neste capítulo, serão explicados alguns conceitos teóricos sobre imagens digitais, aprendizado de máquina e redes neurais, de forma sucinta, mas suficiente para o entendimento dos termos e conceitos utilizados no experimento.

1.1 Imagens digitais

Imagens digitais são armazenadas e exibidas em computadores como uma sequência de pixels. O pixel é a menor unidade que compõe uma imagem, e descreve a intensidade de uma cor, no caso de imagens coloridas. Pensando no formato RGB (*Red, Green* e *Blue*), cada pixel é composto por 3 valores que representam a intensidade de cada uma das 3 cores. Se todos os valores forem 0, o pixel será preto, e se todos os valores forem 255, o pixel será branco. O brilho em imagens RGB, ou seja, a luminosidade de cada pixel, não é definida no pixel mas no dispositivo que a exibe. Uma ilustração destas propriedades está na figura 3.

1.1.1 Representação matemática

A manipulação de imagens digitais é nada menos que transformações sobre pixels, ou um conjunto de pixels. Matematicamente, podemos definir uma imagem como uma matriz $m \times n$, em que cada elemento da matriz é um vetor de 3 elementos, como é possível ver na equação (1).

Figura 3: Cada pixel de uma imagem digital colorida, com o padrão RGB, possui 3 valores para as intensidades das cores vermelho, verde e azul

$$I_{mn} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & \cdots & p_{1n} \\ p_{21} & p_{22} & p_{23} & \cdots & p_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & p_{m3} & \cdots & p_{mn} \end{pmatrix} \tag{1}$$

- I_{mn} é uma imagem de dimensões m por n
- $p_{ij} = (r, g, b),\ 1 \leq i \leq m,\ 1 \leq j \leq n$
- $0 \leq r, g, b \leq 255$

1.2 Aprendizado de máquina

O aprendizado de máquina é uma técnica computacional para otimizar um problema de acordo com um critério, utilizando informações passadas [8]. Mais especificamente, o uso de dados anteriores para previsão de eventos futuros. Especialistas em aprendizado de máquina estudam formas de aplicar diferentes modelos para resolver problemas de forma mais automática, ao invés de requerer pessoas para trabalhar no problema.

O conceito de aprendizado de máquina, apesar de muito popular na atualidade, é relativamente antigo. Em 1959, Arthur Samuel, pesquisador da IBM na época, utilizou a expressão *machine learning* para mostrar o uso de algoritmos computacionais "inteligentes" para o jogo de damas [9]. O conceito se popularizou no século 21 com o desenvolvimento de computadores mais potentes, capazes de aplicar quantidades massivas de dados para o reconhecimento de padrões.

1.2.1 Classificadores

O princípio fundamental do aprendizado de máquina são seus classificadores. Existe uma variada gama de classificadores para os mais diferentes problemas. Um exemplo de classificador é a regressão logística [10], bastante utilizado para resolver problemas cujos resultados são binários, mas com uma porcentagem associada à semelhança entre 0 e 1.

Por exemplo, podemos utilizar a regressão logística para prever a possibilidade de um infarto, baseado em níveis de colesterol, pressão sanguínea, entre outros. Inserindo essas informações no modelo, podemos calcular a probabilidade de uma pessoa ter (valor 1) ou não ter (valor 0) um infarto [10].

1.3 Redes neurais convolucionais

Com o tempo, foram desenvolvidos métodos mais sofisticados e possivelmente mais eficientes para criação de modelos computacionais para otimizações e predições de problemas. A ideia de redes neurais artificiais (*artificial neural networks*, ou ANN), em que são usados equipamentos e ferramentas computacionais para simular o comportamento do cérebro humano, existe há muitos anos, mas se popularizou da mesma forma que o conceito de aprendizado de máquina. Mas diferentemente dos classificadores de aprendizado de máquina, em que devemos alimentar o modelo com variáveis específicas do problema, as redes neurais conseguem extrair diversas características das informações de entrada de forma automática, em especial as **redes neurais convolucionais** (*convolutional neural networks*, ou CNN), quando aplicadas a imagens [11]. Nesta seção será explicado, com mais detalhes, o que seria uma rede neural e, em seguida, o funcionamento das convoluções.

1.3.1 Estrutura das redes neurais

O elemento estrutural fundamental de uma rede neural são os **neurônios**. Os neurônios são funções que recebem os valores calculados de uma camada anterior, realizam transformações sobre os dados, e envia os resultados para a camada posterior. A união de vários neurônios é denominada uma **camada**. Cada neurônio de uma camada específica está ligado a um ou mais neurônios de uma outra camada. Na figura 4, temos uma estrutura básica de uma rede neural, denominada de **totalmente conectada**.

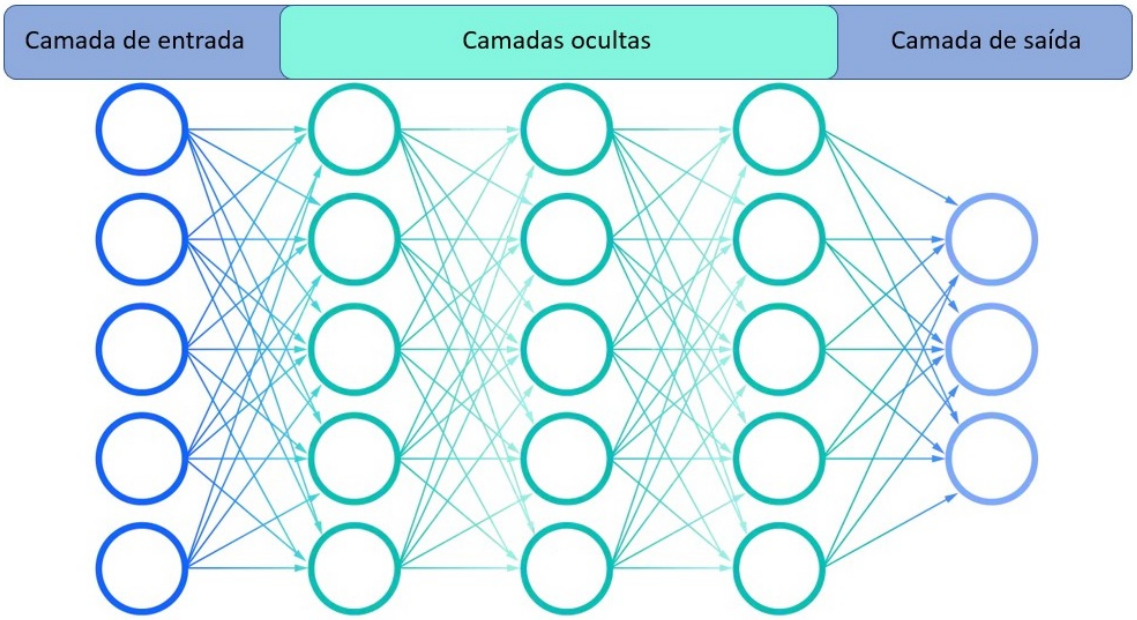

Figura 4: Ilustração de uma rede neural que utiliza apenas camadas totalmente conectadas para modificar os pesos dos neurônios. A estrutura básica de todas as redes neurais assemelha-se a esta. Adaptado de: https://www.ibm.com/cloud/learn/neural-networks

A **camada de entrada** recebe os dados dos quais queremos que a rede aprenda as características [22]. As **camadas ocultas** realizam diversas transformações nos dados inseridos [22]. As transformações dependem do desenvolvedor que cria a rede e da natureza do problema, de acordo com a necessidade de cada transformação. A **camada de saída** recebe os valores das predições para as categorias dos dados inseridos [22]. O número de neurônios e o valor deles depende da natureza do problema.

Levemos em conta o problema de reconhecimento de imagens coloridas. De-

vemos padronizar um tamanho para todas as imagens que serão treinadas/testadas. Se o tamanho das imagens for $m \times n$, o número de neurônios de entrada necessário será $m \times n \times 3$, devido aos 3 canais de cores da imagem, e cada neurônio irá acomodar o valor de cada pixel.

A partir da imagem de entrada, utilizamos regras para a análise e transformação dos valores nos neurônios da camada atual para a camada seguinte. No caso de camadas totalmente conectadas, é realizada uma multiplicação de matrizes. A seguir, veremos uma outra forma de transformação: as convoluções.

1.3.2 Convolução em imagens

Em se tratando de imagens, uma **convolução** é a passagem de um *kernel* (matriz de transformação) sobre os pixels da imagem. Em uma área específica da imagem em que o *kernel* é usado, é computado o produto entre os valores do *kernel* e os valores do pixel sobrepostos na passagem, e os resultados são somados para obter o valor de saída daquela área. Este procedimento pode ser repetido diversas vezes, com diferentes *kernels*, a fim de se obter os resultados procurados [12]. A figura 3 mostra 4 passos de uma convolução.

Um dos principais usos de *kernels* sobre imagens é o reconhecimento de **bordas** entre objetos que aparecem na imagem. As bordas entre objetos são os pixels cujos valores variam muito entre seus pixels vizinhos. Dependendo dos valores selecionados para os *kernels*, podemos delimitar o contorno de uma pessoa, por exemplo.

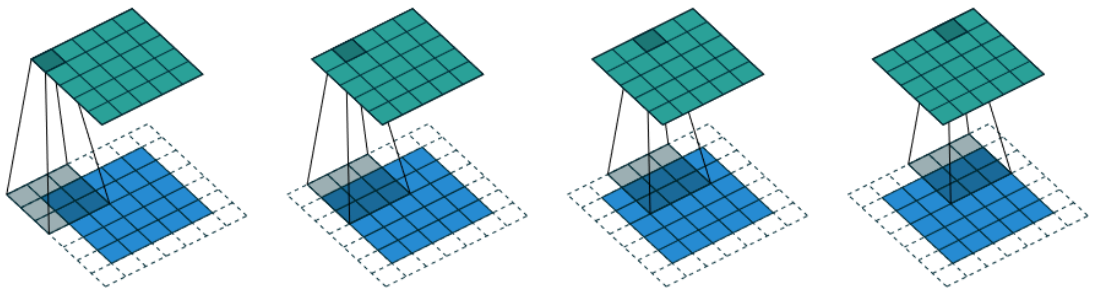

Figura 5: Uma convolução feita sobre uma imagem de dimensões 5 x 5 pixels, com uma borda extra de tamanho 1, utilizando um *kernel* 3 x 3, com salto de tamanho 1. Nesse caso, o resultado é uma imagem de dimensões iguais à original. Imagem obtida de [12].

1.4 Técnicas associadas às redes neurais

Nesta seção, serão explicados brevemente as definições básicas de alguns termos para o entendimento do experimento realizado.

1.4.1 *Epoch*

Uma *epoch* (época) é a passagem do dataset de treino inteiro dentro da rede neural uma única vez. No caso de datasets muito grandes, é comum dividir uma época em iterações. Por exemplo, se dividirmos um dataset em 3 partes (*batches*), serão executadas 3 iterações dentro da época. O número de épocas usado no treinamento pode variar bastante, conforme a necessidade.

1.4.2 ReLu

Uma *rectified linear unit* (unidade linear retificada, ou simplesmente *ReLu*) é uma função de ativação, posicionada ao final de uma camada da rede, que recebe os valores desta camada e retém todos os valores negativos. A equação (2) descreve esta operação.

$$f(x) = max(0, x) \tag{2}$$

- x é o valor calculado no neurônio, antes de passar pela ativação.

1.4.3 Softmax

A função de ativação *softmax* é bastante utilizada em problemas de classificação envolvendo múltiplas classes. Esta função é usada no final da última camada da rede, a fim de calcular as probabilidades que o dado de entrada possa pertencer a cada classe definida pelo problema, ou seja, um valor entre 0 e 1. A soma de todas as probabilidades do vetor-resultado da função *softmax* é 1. A função matemática que calcula essas probabilidades está na equação (3).

$$Y_i = \frac{e^{X_i}}{\sum_{j=1}^{n} e^{X_j}} \tag{3}$$

- Y_i é o i-ésimo resultado da operação *softmax* sobre o elemento correspondente da camada de saída
- X_i é o i-ésimo elemento da camada de saída da rede

- n é o tamanho da camada de saída da rede
- $\sum_{j=1}^{n} e^{X_j}$ é a soma de todos os elementos da camada de saída da rede, precedida pela exponenciação

1.4.4 *Pooling*

Dentre as camadas ocultas das redes neurais, podemos utilizar as camadas de *pooling* (agrupamento) para diminuir a dimensão da camada anterior, reduzindo a complexidade, porém sem reduzir significativamente as características. Existem vários tipos de *pooling*, mas o princípio de todos é o mesmo, que é agregação de informações locais aplicando uma não-linearidade em partes dos dados [12].

Os métodos de *pooling* mais simples são o *pooling*-máximo (*Max Pooling*) e o *pooling*-médio (*Average Pooling*). No primeiro método, a imagem é dividida em seções de tamanho fixo, e sobre elas é passada uma função que devolve o maior número de uma seção. No segundo método, a imagem também é dividida da mesma forma, mas a função que opera sobre as seções devolve o valor médio dos valores dos pixels. Estes métodos estão ilustrados na figura 6.

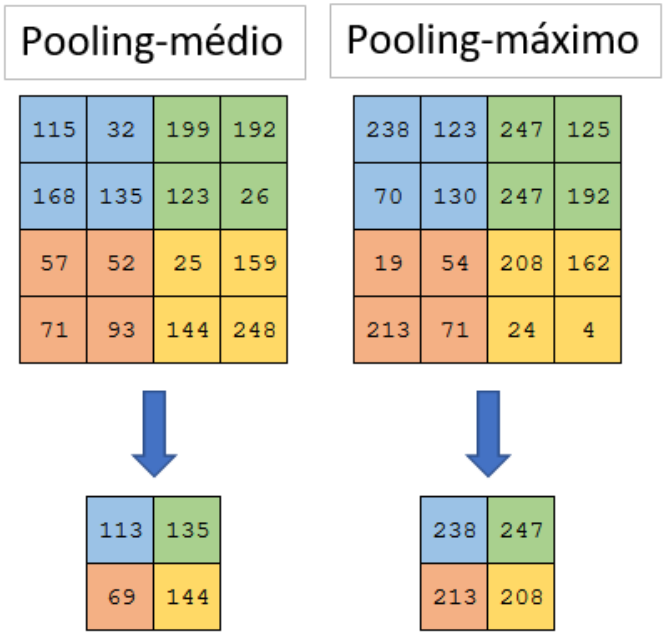

Figura 6: Dois exemplos de uso do *pooling*. Em ambos casos, são usadas imagens de tamanho 4 x 4 pixels e uma região de *pooling* de tamanho 2 x 2. Por exemplo, no caso do *pooling*-máximo, é escolhido o maior número da seção azul, que seria 238.

1.4.5 ADAM

O ADAM (derivado de *Adaptive Moment Estimation*) é um algoritmo de otimização de parâmetros de funções, com o objetivo de maximizar ou minimizar valores. É muito utilizado em modelos de *deep learning* para otimizar de forma automática as taxas de aprendizado (*learning rate*). O conceito deste otimizador é extenso e pode ser encontrado em [13].

1.4.6 *Categorical Cross-entropy*

A *Categorical Cross-entropy* (Entropia Cruzada Categórica) é uma função de perda, cujo objetivo é medir a distância entre o rótulo correto e o rótulo predito, medindo assim a performance do modelo. Quanto menor o valor da função de perda, mais próximas estão as predições feitas dos rótulos verdadeiros. A função é definida na equação (4).

$$CCE = -\sum_{i=1} (L_i \cdot log(SM_i)) \tag{4}$$

- CCE é a *Categorical Cross-Entropy*
- L_i é o valor de uma classe dentre a lista de rótulos (0 ou 1)
- SM_i é o resultado da aplicação da função *softmax* sobre o valor final computado pela rede

1.4.7 *Categorical accuracy*

A *Categorical accuracy* (acurácia categórica) é uma métrica para o cálculo de acertos de um modelo. Dado um vetor de saída da rede neural, cujos valores são resultado da aplicação do *softmax*, a classe (ou categoria) a ser selecionada será aquela com a maior probabilidade. Se a classe selecionada coincidir com a(s) classe(s) verdadeiras, então a predição é considerada certa.

Capítulo 2

Reconhecimento de ações humanas

O reconhecimento de ações humanas por computadores não é uma tarefa trivial. Um dos principais problemas no reconhecimento destas ações é a vasta gama de possibilidades em que elas podem ocorrer, tanto a ação em si quanto o local em que ela acontece. Um exemplo de um problema relativamente menos complexo é a identificação de um ser humano dentre outros seres vivos, como um tigre ou uma coruja. Este exemplo de problema trata da presença ou não de um determinado padrão que assemelha-se a um ser humano. No caso de ações, é necessário levar em conta, além do padrão, o contexto no qual ela ocorre.

Neste trabalho, serão tratadas ações humanas em imagens coloridas.

2.1 Problemática

Existem diversas formas de atacar o problema de reconhecimento de ações humanas em imagens. Vários trabalhos têm diferentes aproximações sobre o problema. Uma forma seria considerar apenas interações entre pessoas e objetos, separando características que compõem a imagem e resolvendo o problema por partes [5]. Uma outra forma seria verificar a pose de uma pessoa numa imagem, e compará-la com uma pose-protótipo, já conhecida [6]. Uma extensão desta aproximação é o uso de 3 dimensões para reconhecer a ação, ao invés de restringir para apenas as 2 dimensões da imagem [7].

Capítulo 3

Experimento

Para a construção do experimento, foi pensado nos seguintes passos:

- Escolha do dataset que será estudado e usado para o treinamento
- Escolha de uma rede neural para estudo e implementação
- Escolha das ferramentas para a implementação
- Adaptação da rede e do dataset para o problema
- Implementação do código para o treinamento e avaliação da métrica
- Estudo e redação dos resultados

Primeiramente devemos escolher o dataset que melhor se encaixa ao problema a ser estudado. Depois, podemos procurar uma rede neural que melhor se adapte ao dataset. Com os elementos-base definidos, podemos escolher as ferramentas a serem usadas para instruir o computador para o treinamento.

Com a "teoria" escolhida, devemos modificar tanto o dataset como a rede para se encaixar aos padrões do problema (neste trabalho, imagens coloridas com 15 classes). Finalmente, treinamos o modelo e avaliamos os resultados.

3.1 Dataset de imagens

O dataset de ações humanas escolhido para o experimento foi elaborado para o *Data Sprint 76*, um desafio promovido pelo AI Planet [1]. Este dataset possui 12600 imagens de treino e 5400 imagens de teste, com dimensões variáveis e coloridas. O padrão de cores utilizado depende da forma como a imagem é carregada. Neste experimento, as imagens são carregadas com formato de cores RGB. O diretório que compõe o dataset possui 2 outros diretórios, com imagens de treino e de teste, e 2 arquivos CSV, com os nomes e rótulos das imagens de treino, e apenas os nomes das imagens de teste. Para cada imagem, são consideradas 15 possíveis classes de ações humanas. As figuras 7.1, 7.2 e 7.3 mostram um exemplo para cada ação.

calling

clapping

cycling

drinking

dancing

eating

Figura 7.1: Exemplos de imagens para algumas classes.

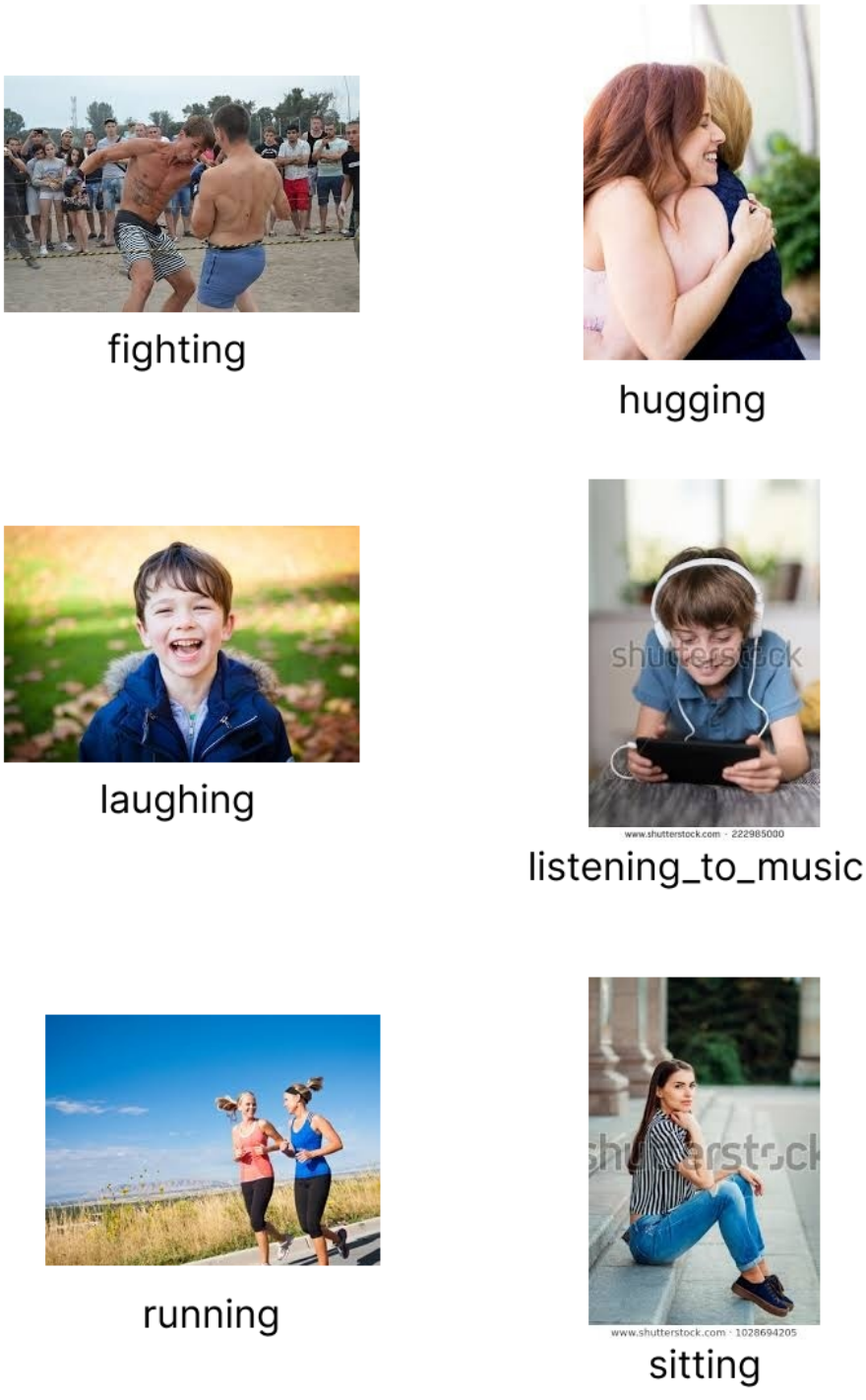

Figura 7.2: Exemplos de imagens para algumas classes.

Figura 7.3: Exemplos de imagens para algumas classes.

3.1.1 Características

Cada imagem do dataset contém uma ou mais ações humanas, com uma ou mais pessoas envolvidas, que podem ser o foco principal da imagem, ou estarem em plano de fundo. No primeiro caso, em que a(s) pessoa(s) envolvida(s) na ação apareceria(m) no centro da imagem, tanto a análise humana quanto a análise do modelo sobre a imagem são mais fáceis, pois a informação relevante ocupa uma boa parte da imagem, tornando o entendimento da ação mais claro.

Muitas ações humanas ocorrem em conjunto umas com as outras, como *eating* e *sitting*, ou *hugging* e *laughing*. No entanto, as imagens de treino estão rotuladas com apenas 1 classe, vinda originalmente do desafio em que elas fazem parte.

O dataset possui imagens como diversos níveis de ruído. No caso em que há relativamente bastante ruído, os elementos que não fazem parte da ação em si atrapalham ou até impedem o reconhecimento correto da ação apresentada. Exemplos destes elementos são marcas d'água, elementos textuais e iluminação variada que

compõem parte da imagem. As marcas d'água são tratadas como parte da imagem, sem um rótulo preferencial indicando a existência ou não da marca. Alguns destes ruídos podem ser observados na figura 8.

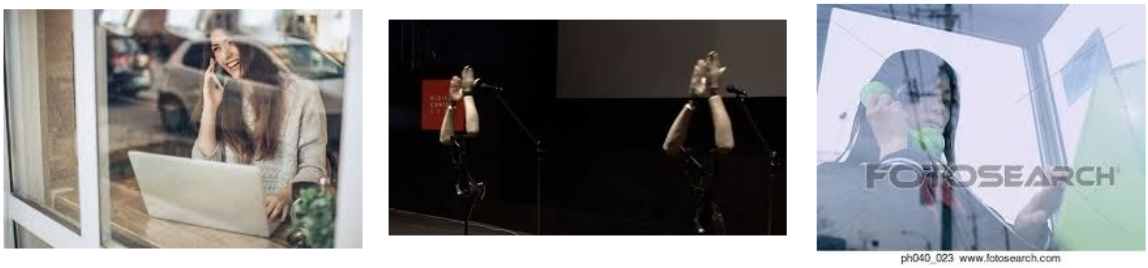

Figura 8: Exemplos de imagens com grandes quantidades de ruído. Na imagem do centro, o ruído considerado é a "camuflagem" da pessoa com o plano de fundo.

As imagens do dataset possuem resoluções consideradas baixas, se comparadas com fotos atuais usuais. Cada imagem possui uma resolução menor do que 500 x 500 pixels. Esta resolução não afetará o treinamento de forma negativa, como veremos nas seções seguintes.

3.1.2 Adaptações

O dataset apresentado passou por modificações para se adaptar ao propósito deste experimento. Originalmente, por fazer parte de um desafio de classificação, o dataset não providenciava os rótulos para as imagens de teste. Assim, foi necessária a rotulação das 5400 imagens de teste manualmente. A rotulação puramente manual, ou seja, inserir em um arquivo CSV o nome de cada imagem e as ações que ocorrem nela, é demorada e extremamente inviável. Assim, para a realização deste trabalho, foi utilizada a ferramenta SLIL (*Street Level Imagery Labeler*) [2], desenvolvida por Artur André A. M. Oliveira. Esta ferramenta realiza a rotulação de 3 elementos encontrados em imagens de ruas. Ela foi, então, adaptada para este experimento, inserindo as 15 classes de ações.

A rotulação das imagens de teste deve seguir um critério definido para uniformizar os dados. A tarefa de rotulação, dependendo da rigorosidade do critério, pode se tornar bastante complicada. Neste trabalho, a rotulação segue os seguintes critérios:

- **Senso comum:** as ações são rotuladas diretamente com o conteúdo que ela apresenta, sem nenhuma forma de distorção ou censura, baseadas no senso comum. Por exemplo, uma pessoa com um *smartphone* em mãos, com os

dois polegares sobre a tela, é considerado *texting*. Uma outra interpretação, fora do senso comum, é considerar esta ação como, por exemplo, *eating*, o que não é uma associação correta.

- **Inferência por senso comum:** o contexto do qual a ação apresentada na imagem é levado em conta no momento de realizar a rotulação, sem considerar metáforas visuais ou possíveis outras interpretações incomuns. Por exemplo, suponha uma imagem de uma pessoa comendo sobre uma mesa, mas não é possível ver se a pessoa está ou não sentada. Em plano de fundo, vemos outras pessoas comendo e sentadas em uma mesa parecida. Assim, podemos inferir que a pessoa em foco principal também está sentada e comendo, sendo a imagem rotulada como *sitting* e *eating*.
- **Ação mais provável:** caso a ação apresentada na imagem não esteja totalmente clara, a imagem é rotulada com a ação mais provável que ela representa, seguindo o critério do senso comum. Por exemplo, uma imagem com apenas uma pessoa deitada sobre uma cama, com a cabeça sobre um travesseiro, mas com os olhos abertos, é considerada como *sleeping*, pois é a ação que melhor descreve a imagem, sem deixar de rotulá-la.

A figura 9 mostra alguns exemplos de imagens de treino e de teste, após o rotulamento das imagens de teste.

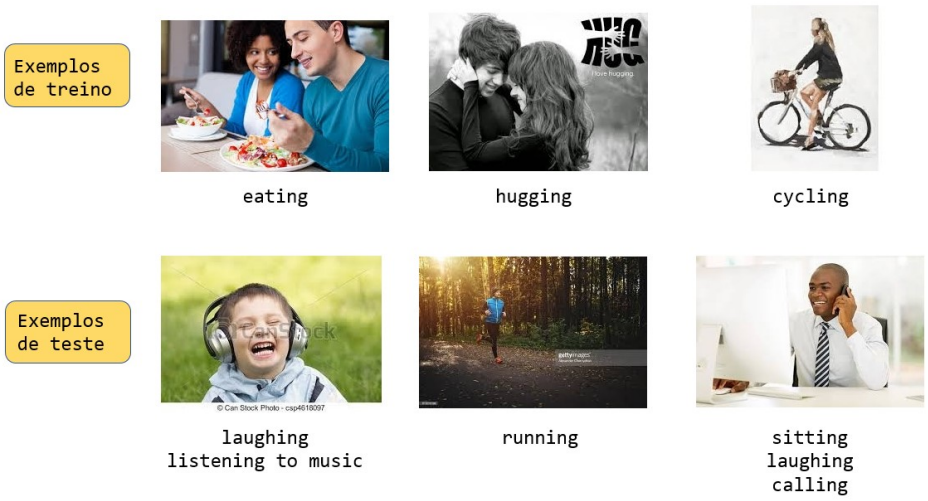

Figura 9: Algumas das imagens que podem ser encontradas no dataset adaptado do experimento. É possível observar as marcas d'água, a variação das dimensões e múltiplos rótulos para as imagens de teste.

Com estes critérios em mente, foram rotuladas todas as imagens de teste. Para cada imagem, é observada se uma ou mais das classes está presente, sendo executada por uma ou mais pessoas. Em seguida, as ações são registradas em suas respectivas classes. Finalmente, para cada classe é associado um número e a presença da ação é registrada como 1 (ação presente) ou 0 (ação não presente). A figura 10 ilustra como estas transformações são feitas. Esta última transformação descrita é denominada *one-hot encoding* [14].

3.2 VGGNet

Em 2015, dois pesquisadores da Universidade de Oxford, Karen Simonyan e Andrew Zisserman propuseram um novo modelo de rede neural, em um *paper* intitulado *Very Deep Convolutional Networks For Large-Scale Image Recognition* [3]. O modelo recebeu o nome do grupo de pesquisa do qual eles atuam, o Visual Geometry Group (VGG). Também podemos chamar o modelo de VGGNet (VGG Network).

O modelo foi submetido ao ImageNet Challenge 2014, utilizando um dataset com 1000 classes de objetos e mais de 1 milhão de imagens, atingindo níveis de acurácia maiores do que os modelos *state-of-the-art* [3]. Além disso, é também aplicável a outros datasets para reconhecimento de imagens [3].

3.2.1 Arquitetura

O modelo VGG possui diversas versões com um número de camadas variável. Neste trabalho, foi utilizada a VGG16 [3]. Este modelo é composto por 13 camadas convolucionais e 3 camadas totalmente conectadas, ambos tipos com uma camada *ReLu* logo após, além de 5 camadas de *pooling*-máximo (vide figura 11). A última camada do modelo ainda passa por uma camada *softmax*. O modelo recebe o nome pela soma da quantidade de camadas convolucionais e camadas totalmente conectadas. A arquitetura com mais detalhes pode ser encontrada em [3].

A rede foi adaptada neste trabalho para realizar o treinamento com 15 classes ao invés de 1000. Para isso, foram removidas as 3 camadas totalmente conectadas e suas funções de ativação, e substituídas por uma camada totalmente conectada com *ReLu* e uma última camada totalmente conectada com *Softmax*. Além disso, foi utilizado tanto o *pooling*-médio quanto o *pooling*-máximo para o experimento. A arquitetura da rede adaptada é ilustrada na figura 11.

calling	não
clapping	não
cycling	não
dancing	não
drinking	não
eating	não
fighting	não
hugging	não
laughing	sim
listening_to_music	sim
running	não
sitting	não
sleeping	não
texting	sim
using_laptop	não

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	0	0	0	0	1	1	0	0	0	1	0

Figura 10: Ilustração do processo de rotulação das imagens. A imagem é analisada, em seguida é registrada as ações que aparecem para cada classe. Finalmente, as classes e a presença da ação são codificadas [14] para um vetor com 15 índices, com cada índice contendo o valor 1 (sim) ou 0 (não).

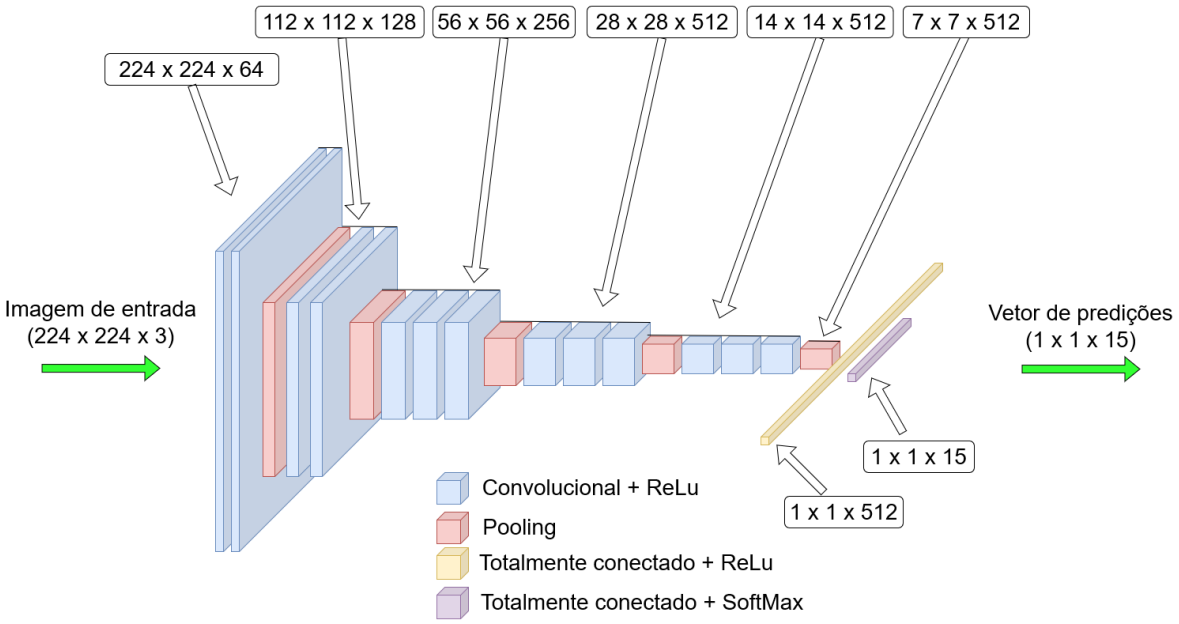

Figura 11: Arquitetura modificada do VGG16 para o experimento. O topo da rede foi removido e substituído por 2 camadas totalmente conectadas, uma com *ReLu* e outra com *softmax*. A figura está fora de escala.

Capítulo 4

Implementação

Neste capítulo, são feitas algumas considerações sobre os softwares usados para o treinamento e classificação das imagens do dataset. Como ponto de partida, foi utilizada uma implementação feita por Meet Nagadia, para o mesmo dataset [23].

4.1 Ferramentas

O experimento foi construído utilizando a linguagem de programação Python, com auxílio das bibliotecas de *deep learning* **Tensorflow** [15] e **Keras** [16]. Para o treinamento do modelo, foi utilizada a plataforma **Colab** [17], com a GPU Tesla T4. Não foi possível escolher uma GPU específica, mas é possível verificar a GPU usada pelo comando `!nvidia-smi`. Também foi utilizada a plataforma **Kaggle** [18] para o treinamento, com a opção "GPU P100". Foram escolhidas estas plataformas pela simplicidade de uso e pelo treinamento acelerado. Para manipulação das imagens e dados, foi utilizada a biblioteca **Pandas** [19]. A geração de gráficos e apresentação de imagens foi feita com a biblioteca **Plotly** [20].

4.2 Processamento dos dados

Antes de realizar o treinamento, os dados foram processados de modo que sejam compatíveis com a entrada do modelo. A princípio, a resolução das imagens era ajustada para 160×160 pixels. Porém, uma segunda tentativa de treinamento demonstrou que em um ajuste de resolução para 224×224 pixels, as dimensões usadas no modelo original do VGG16, melhorava a avaliação do modelo, que veremos adiante. A resolução das imagens foi ajustada tanto para estas novas dimensões quanto para as dimensões antigas, para todas as imagens do dataset, a fim de verificar as diferenças entre as duas. Como certas imagens possuem dimensões que ultrapassam tais pixels, a proporção da resolução (*aspect ratio*) não é mantida.

Os rótulos de treinamento foram transformados em uma lista de zeros, em Python, com o valor 1 na ação que a imagem apresenta. Os rótulos de teste foram apenas ajustados para uma lista válida em Python, do mesmo formato dos rótulos de treinamento.

4.3 Treinamento e Hiperparâmetros

A definição do modelo implementado, em Python, é descrita no bloco de código seguinte:

```
1  m = Sequential()
2
3  vgg = tf.keras.applications.VGG16(
4              include_top=False,
5              input_shape=input_shape,
6              pooling=pooling,
7              classes=15,
8              weights="imagenet")
9
10 for layer in vgg.layers:
11     layer.trainable = False
12
13 m.add(vgg)
14 m.add(Flatten())
15 m.add(Dense(512, activation="relu"))
16 m.add(Dense(15, activation="softmax"))
```

A rede foi construída como uma sequência de camadas, utilizando o método `Sequential()` [16], que permite a adição de camadas na rede neural de forma customizada. Inicialmente, é adicionada a rede VGG16, implementada pelo Keras, pelo método `tf.keras.applications.VGG16`. Na linha 4, o primeiro argumento do método, `include_top=False`, remove as 3 camadas totalmente conectadas no topo. Na linha 8, o último argumento, `weights="imagenet"`, inicializa a rede com os pesos treinados pelo modelo original do VGG16. Os outros argumentos são autodescritivos.

O uso de uma rede pré-treinada é comum para acelerar o treinamento de problemas relacionados, que é uma técnica denominada *transfer learning* [21]. A inicialização da rede com os pesos originais reduz o número de épocas necessário para atingir uma acurácia de treinamento decente. Os pesos do VGG16 são mantidos intactos durante o treinamento impedindo que sejam modificados, pelas linhas 10 e 11.

Ainda, o topo da rede consiste de um *flatten* (achatamento) e duas camadas totalmente conectadas (linhas 14, 15 e 16). A primeira camada totalmente conectada possui 512 neurônios, seguindo o padrão da camada de *pooling* anterior, e a última camada possui 15 neurônios, um para cada classe.

O modelo é, então, compilado usando o método `compile()`, descrito abaixo:

```
m.compile(optimizer="adam",
          loss="categorical_crossentropy",
          metrics=["categorical_accuracy"])
```

O otimizador ADAM [13] foi escolhido por possuir boa performance de treinamento, além de ser o otimizador recomendado na maioria dos casos. A função de perda *categorical cross-entropy* é usada devido a natureza do problema, pois as imagens de treino possuem apenas um rótulo dentre as possíveis classes (normalmente denominado como um problema de **multiclasses**). Por fim, é usada a métrica *categorical accuracy*, definida na seção 1.4.7.

Com o modelo construído, foi realizado o treinamento, com variado número de épocas, utilizando o método `fit()` [16]. Os resultados da acurácia de treino, bem como a função de perda, para os casos indicados, são apresentados na figura 12.

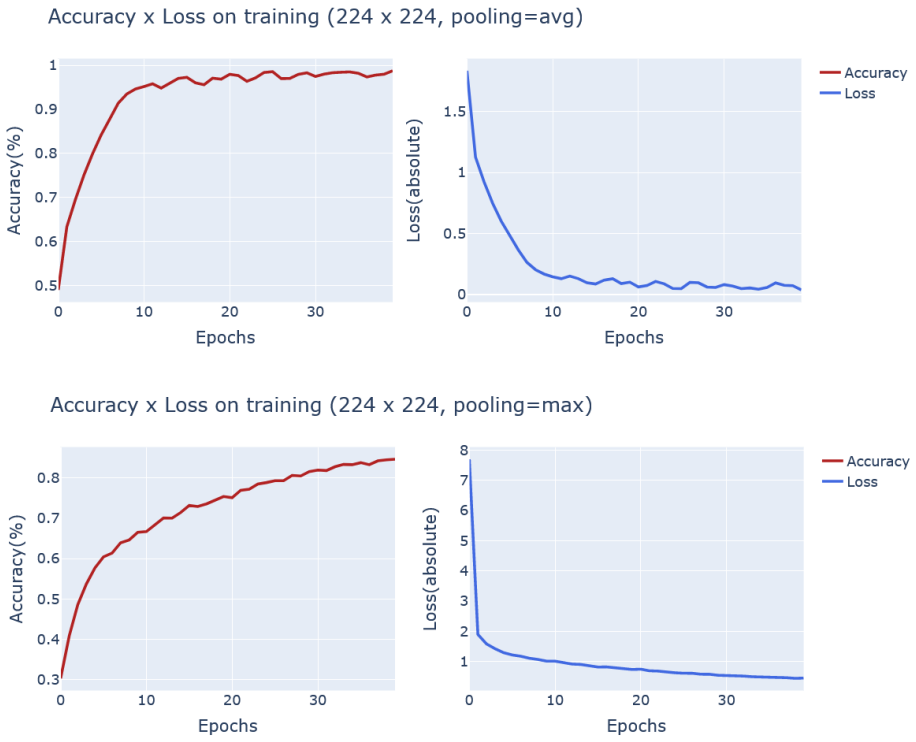

Figura 12: Resultados da acurácia e da função de perda no treinamento, em uma execução do modelo com 40 épocas, com diferentes métodos de *pooling*. Com o *pooling*-médio, o treino alcançou uma acurácia alta mais rapidamente do que o *pooling*-máximo.

4.4 Avaliação

Com os resultados do treinamento, foi feita a avaliação do modelo sobre o conjunto de imagens de teste. As 5400 imagens foram submetidas ao modelo treinado utilizando o método `evaluate()` [16]. O treinamento foi feito diversas vezes, variando diversos parâmetros como o número de épocas, o método de *pooling* e as dimensões da imagem. Na tabela 1 temos os resultados considerando as variações de cada componente.

Podemos observar que o *pooling*-máximo, além de realizar um treinamento mais lento, resulta em uma acurácia menor, porém, possui perdas menores. O *pooling*-médio consegue acurácias melhores, mas com perdas maiores. É possível observar também que, usando o *pooling*-médio e dimensões 224×224, a acurácia com 10 ou 40 épocas é a mesma, mas com 10 épocas a perda é bem menor. Em geral, considerando a acurácia de treinamento *versus* a acurácia de teste, podemos dizer que esse seria um possível caso de *overfitting* [22].

Pooling / Resolução	Épocas 10	20	30	40
Pooling-máximo	0.432	0.418	0.452	0.453
160x160	4.990	7.071	11.701	12.704
Pooling-máximo	0.472	0.467	0.475	0.471
224x224	4.347	5.493	7.637	8.734
Pooling-médio	0.467	0.489	0.495	0.499
160x160	7.544	11.020	13.522	17.334
Pooling-médio	0.531	0.516	0.518	0.531
224x224	6.402	10.351	13.469	14.762

Acurácia
Perda

Tabela 1: Resultados da avaliação do modelo com diferentes hiperparâmetros. Em vermelho, temos a acurácia de avaliação dos testes. Em azul, temos os valores da função de perda.

A avaliação leva em conta a *categorical accuracy*, ou seja, as predições são consideradas corretas se a maior probabilidade associada às classes pertencer a uma das classes verdadeiras. Um conjunto de imagens com seus rótulos originais e rótulos preditos pelo modelo pode ser visto nas figuras 13.1, 13.2, 13.3 e 13.4.

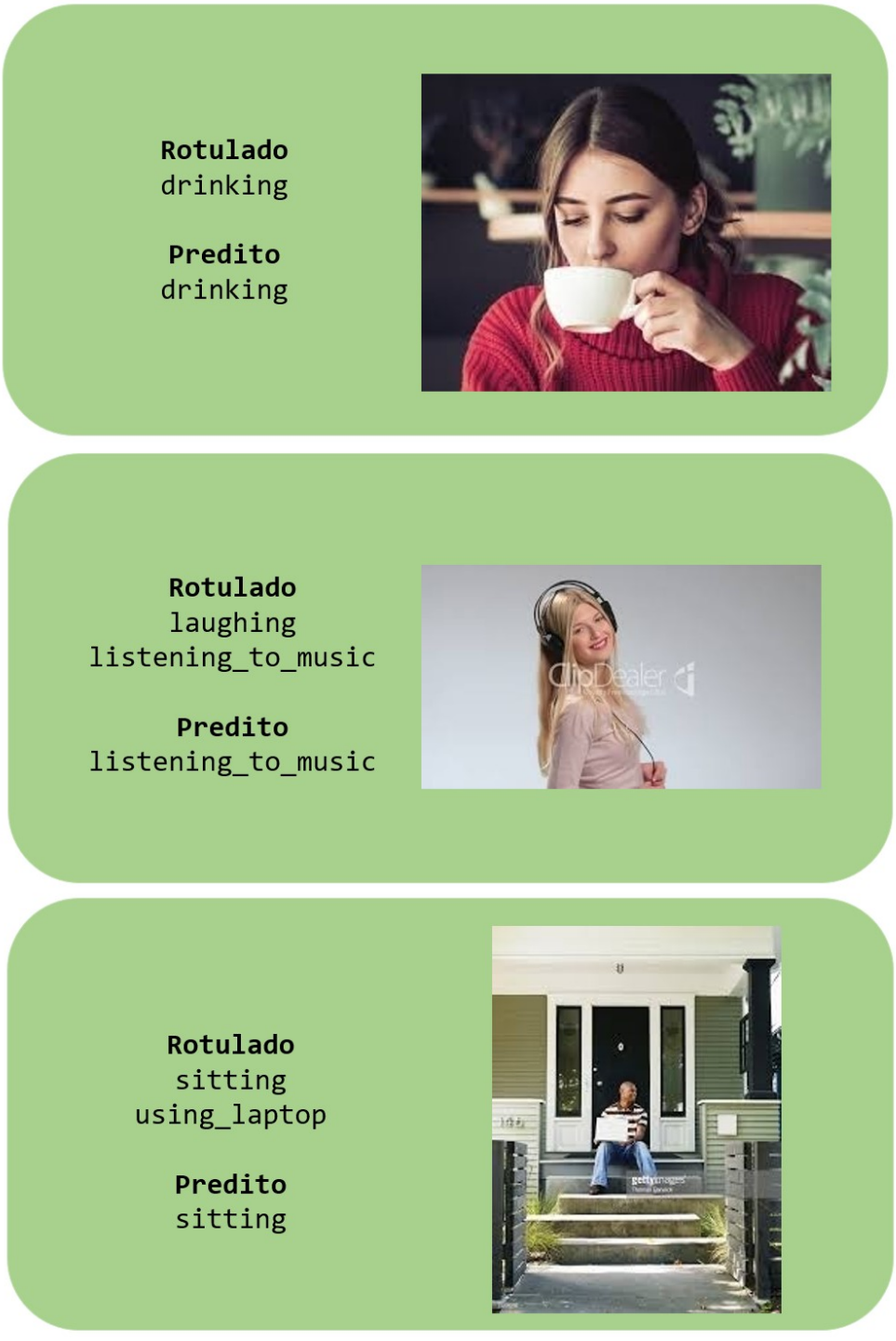

Figura 13.1: Exemplos de imagens de teste preditas **corretamente**, com seus rótulos originais e os rótulos preditos na avaliação do modelo.

Figura 13.2: Exemplos de imagens de teste preditas **corretamente**, com seus rótulos originais e os rótulos preditos na avaliação do modelo.

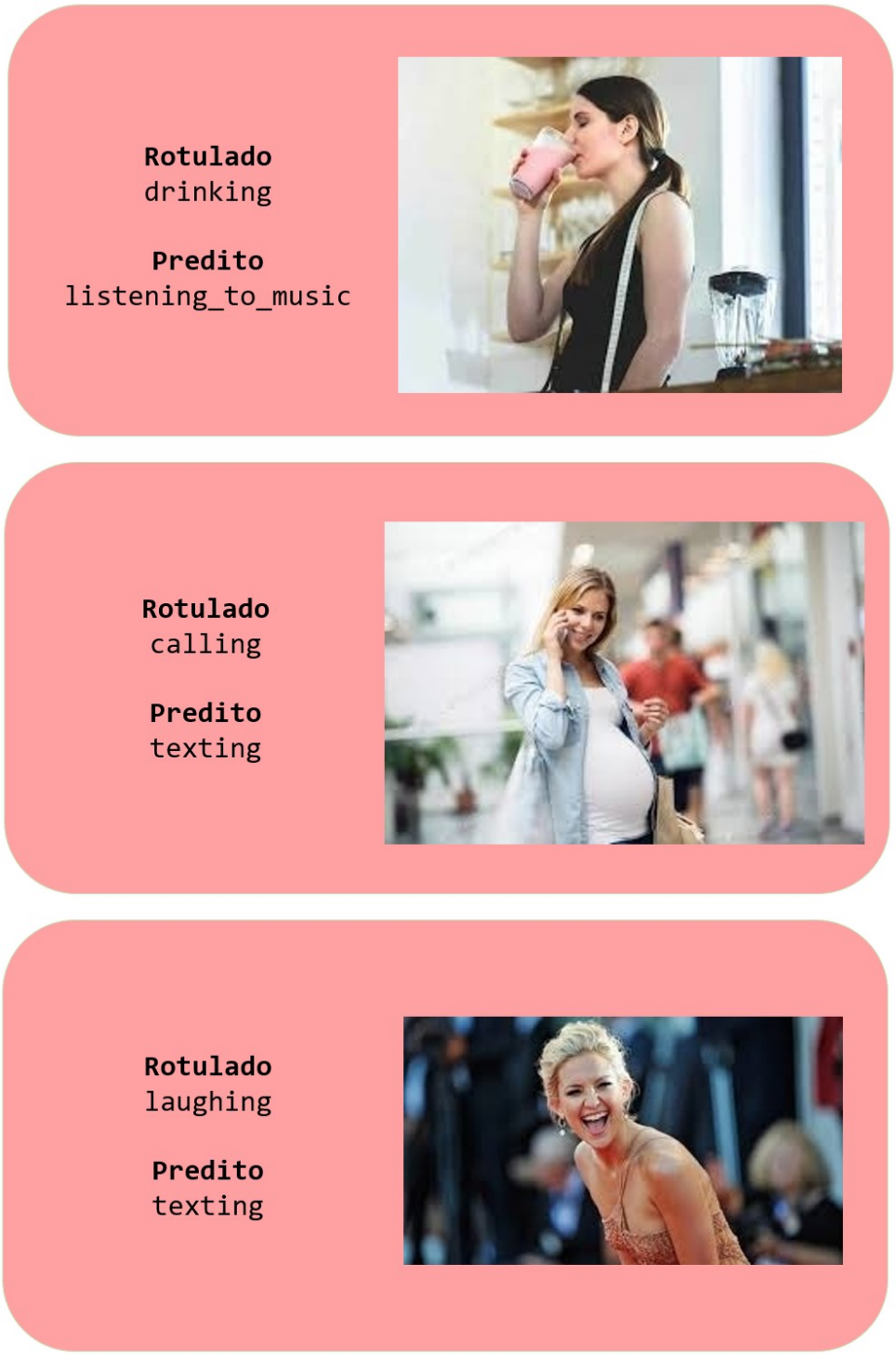

Figura 13.3: Exemplos de imagens de teste preditas **incorretamente**, com seus rótulos originais e os rótulos preditos na avaliação do modelo.

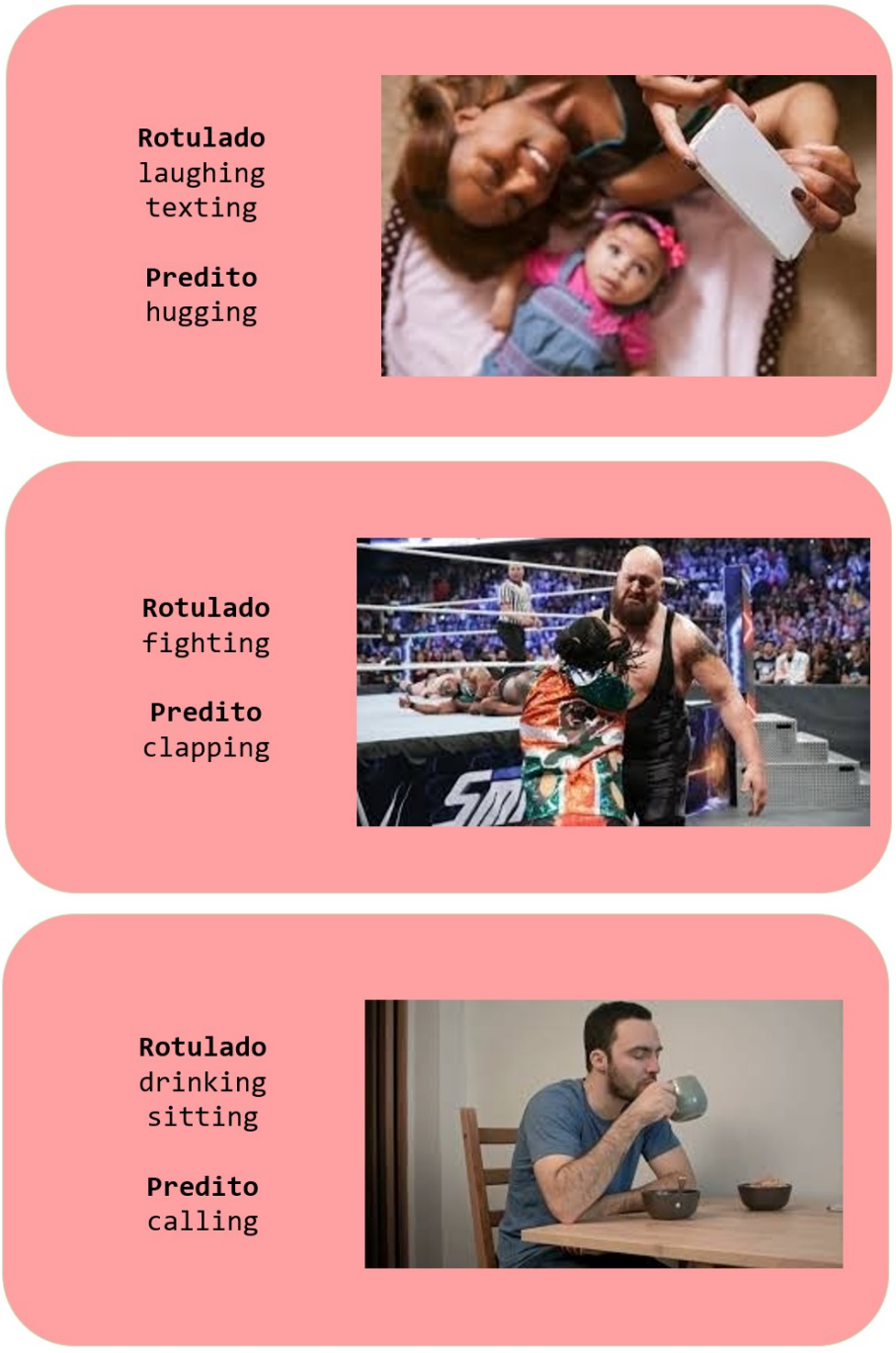

Figura 13.4: Exemplos de imagens de teste preditas **incorretamente**, com seus rótulos originais e os rótulos preditos na avaliação do modelo.

Conclusões

O reconhecimento de ações humanas em imagens com o uso de técnicas de *deep learning* é plausível. Apesar da complexidade do problema, é possível criar um modelo e realizar seu treinamento, obtendo resultados palpáveis e utilizáveis.

Grandes quantidades de ações humanas capturadas em imagens são extremamente úteis para o treinamento de modelos para o reconhecimento destas ações. Na atualidade, o problema ainda é considerado complexo, porém houve grandes progressos nesta área de estudo. Com os avanços em tecnologias de *smart cities*, carros autônomos e ações automáticas baseadas em movimento humano, por exemplo, o estudo do reconhecimento de ações humanas se torna muito promissor.

Um importante fator para a construção de modelos capazes de reconhecer ações humanas é o estudo aprofundado dos conceitos teóricos que envolvem a área. Entender o funcionamento das redes neurais, bem como os conceitos matemáticos que as permeiam, é fundamental para o trabalho de um desenvolvedor da área. Atualmente, é relativamente simples treinar uma rede neural e ajustar seus hiperparâmetros, mas deve-se entender como a ferramenta e os algoritmos internos se comportam ao realizar as operações.

Existe a necessidade de possuir um dataset com características suficientes para o treinamento de um modelo. A diversidade de ações humanas ocorrendo em diversos planos de fundo, por exemplo, é primordial para o treinamento correto do modelo, pois assim este entenderá melhor a diferença de um ser humano e o plano de fundo no qual ele está presente. Em geral, durante a construção de um dataset para classificação, deve haver uma grande diversidade de ocorrências das classes nas imagens, a fim de separar estas ocorrências das quais se deseja classificar do plano de fundo.

Há diversos pontos que podem ser melhorados neste experimento, abrindo novas possibilidades para estudo. Exemplos de melhorias seriam: re-rotulação das imagens de treinamento, descrevendo todas as ações que ocorrem nelas ao invés de apenas uma; utilização de outros datasets com ações humanas, para verificar diferenças tanto entre datasets como no comportamento do modelo em situações diferentes de treinamento; ajustes mais exaustivos dos parâmetros, a fim de entender mais profundamente a influência que certos valores têm sobre o treinamento, e o uso de um dataset extra para validação cruzada (*cross validation*).

Referências Bibliográficas

[1] AI Planet (formerly DPhi) - Ecosystem Engineering the future of AI, Technology Innovation. **"Data Sprint 76 - Human Activity Recognition. AI Planet (formerly DPhi)"**. Disponível em: https://aiplanet.com/challenges/data-sprint-76-human-activity-recognition/233/overview/about. Acesso em: 19 dez. 2022.

[2] Artur André A. M. Oliveira. **"Street Level Imagery Labeler (SLIL)"**, GitHub, github.com/arturandre/SLIL. Acesso em: 20 dez. 2022.

[3] Karen Simonyan and Andrew Zisserman. **"Very Deep Convolutional Networks for Large-Scale Image Recognition"**, International Conference on Learning Representations", 2015.

[4] Zhu, Wenbo and Yeh, Wei-Chang and Chen, Jianwen and Chen, Dafeng and Li, Aiyuan and Lin, Yangyang. **"Evolutionary Convolutional Neural Networks Using ABC"**, ICMLC '19: Proceedings of the 2019 11th International Conference on Machine Learning and Computing, pp. 156-162, Fevereiro 2019.

[5] D. Girish, V. Singh and A. Ralescu. **"Understanding action recognition in still images"**, 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1523-1529, 2020.

[6] Thurau, Christian, and Václav Hlavác. **"Pose primitive based human action recognition in videos or still images"**, 2008 IEEE conference on computer vision and pattern recognition, 2008.

[7] D. C. Luvizon, D. Picard and H. Tabia, **"Multi-Task Deep Learning for Real-Time 3D Human Pose Estimation and Action Recognition"**, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 8, pp. 2752-2764, 1 de Agosto 2021.

[8] Alpaydin, Ethem. **"Introduction to Machine Learning"**, Reino Unido: MIT Press, 2020.

[9] A. L. Samuel. **"Some Studies in Machine Learning Using the Game of Checkers"**, IBM Journal of Research and Development, vol. 3, no. 3, pp. 210-229, Julho 1959.

[10] Magdon-Ismail, M., Abu-Mostafa, Y. S., Lin, H. **"Learning from Data: A Short Course"**, Estados Unidos: AMLBook.com, 2012.

[11] Zijie J. Wang and Robert Turko and Omar Shaikh and Haekyu Park and Nilaksh Das and Fred Hohman and Minsuk Kahng and Duen Horng Chau. **"CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization"**, IEEE Transactions on Visualization and Computer Graphics, vol. 27, pp. 1396-1406, 2020.

[12] Dumoulin, Vincent and Visin, Francesco. **"A guide to convolution arithmetic for deep learning"**, arXiv e-prints, Março 2016.

[13] Kingma, Diederik Ba, Jimmy. **"Adam: A Method for Stochastic Optimization"**. International Conference on Learning Representations, 2014.

[14] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. **"Product-Based Neural Networks for User Response Prediction over Multi-Field Categorical Data"**. ACM Trans. Inf. Syst., vol. 37, nº 1, artigo 5, Janeiro de 2019.

[15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. **"TensorFlow: Large-scale machine learning on heterogeneous systems"**, 2015. Software disponível em https://www.tensorflow.org/.

[16] Chollet, François e outros. **"Keras"**, 2015. Software disponível em https://keras.io.

[17] Google. **"Colaboratory: Frequently Asked Questions"** Acesso em: 29 de dezembro de 2022. Disponível em: https://research.google. com/colaboratory/faq.htm.

[18] Kaggle. **"Your Machine Learning and Data Science Community"**. Acesso em: 29 de dezembro de 2022. Disponível em https://www.kaggle.com/.

[19] Wes McKinney. **"Data Structures for Statistical Computing in Python"**, Proceedings of the 9th Python in Science Conference, pp. 56-61, 2010.

[20] Plotly Technologies Inc. **"Collaborative data science"**, Plotly Technologies Inc, Montréal, QC, 2015. Disponível em: https://plot.ly

[21] Weiss, K., Khoshgoftaar, T.M. Wang, D. **"A survey of transfer learning"**, J Big Data 3, 9, 2016.

[22] Keiron O'Shea and Ryan Nash. **"An Introduction to Convolutional Neural Networks"**, CoRR, 2015.

[23] Meet Nagadia. **"HAR - VGG"**, Kaggle, 2022. Disponível em: https://www.kaggle.com/code/meetnagadia/har-vgg/notebook.