UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA QUÍMICA

TRABALHO DE CONCLUSÃO DE CURSO EM ENGENHARIA QUÍMICA

Identificação e Classificação de Lixo Doméstico Através de Rede Neural Convolucional

Autor: Lucas Pavanelo Siqueira

Orientador: Pedro Rafael Bolognese Fernandes

Porto Alegre, novembro de 2021

Autor: Lucas Pavanelo Siqueira

Identificação e Classificação de Lixo Doméstico Através de Rede Neural Convolucional

Trabalho de Conclusão de Curso apresentado à COMGRAD/ENQ da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de Bacharel em Engenharia Química

Orientador: Pedro Rafael Bolognese Fernandes

Banca Examinadora:

Prof. Dr. Edson Cordeiro do Valle, UFRGS

Dra. Aline de Campos Cássia Pena, UFRGS

Porto Alegre

2021

AGRADECIMENTOS

Agradeço aos meus pais pelo amor e zelo me dão desde sempre, aos amigos e familiares pelos momentos que compartilhamos e ao prof. Pedro Fernandes pela paciência e orientação nesse trabalho.

RESUMO

A geração de resíduos sólidos urbanos no Brasil tem crescido de maneira substancial nos últimos anos, e cerca de 40% de todo esse resíduo é descartado incorretamente, gerando riscos à saúde e ao meio ambiente. A tecnologia de *Machine Learning* vem sendo largamente empregada, particularmente na área de reconhecimento de imagem, para a identificação e classificação desses resíduos. Este trabalho propõe-se a analisar a viabilidade de diversos modelos de redes neurais aplicadas na classificação de lixo reciclável doméstico, com o objetivo de desenvolver um protótipo de rede neural que possa ser continuamente aprimorado em trabalhos futuros para posterior uso como ferramenta de auxílio pessoal na destinação correta do lixo doméstico. Utilizando do conjunto de dados *TrashNet* com adição de algumas classes de imagens adequadamente reclassificadas do *Waste_Pictures*, testou-se a performance inicial de quatro modelos de redes neurais (ResNet34, VGG16, DenseNet121 e MobileNet_V2), para posterior ajuste fino do modelo com melhor performance. O modelo VGG16 atingiu 85,28% de acurácia, mas para classificar não-recicláveis (restos) atingiu somente 47,67%. O modelo ResNet34 obteve 84,99% de acurácia geral, e 66,67% para não-recicláveis, sendo o modelo escolhido para ser otimizado. O novo teste com o modelo ResNet34 foi treinado com imagens menores nos estágios iniciais para acelerar o aprendizado, e com imagens maiores nos estágios finais para aumentar sua acurácia, resultando em uma acurácia final de 91,97%. Para melhores resultados, sugere-se aumentar a qualidade e tamanho dos conjuntos de dados, assim como testar extensivamente modelos e parâmetros mais otimizados.

Palavras-chave: *Visão Computacional*, *Rede Neural, Classificação de Lixo, Trashnet*

ABSTRACT

The generation of urban solid waste in Brazil has grown substantially in recent years, and about 40% of all this waste is disposed of incorrectly, creating risks to the public health and the environment. Machine Learning technology has been widely employed, particularly in the area of image recognition, for the identification and classification of these waste. This work proposes to analyze the viability of several neural network models applied to the classification of household recyclable waste, with the objective of developing a neural network prototype that can be continuously improved over in future works for further use as a tool for personal assistance in the correct disposal of household waste. Using the TrashNet dataset with the addition of some appropriately reclassified image classes from the Waste_Pictures dataset, the initial performance of four neural network models (ResNet34, VGG16, DenseNet121, and MobileNet_V2) was tested, for subsequent fine-tuning of the best performing model. The VGG16 model achieved 85.28% accuracy, but for classifying non-recyclables (trash) it achieved only 47.67%. The ResNet34 model achieved 84.99% accuracy overall, and 66.67% for non-recyclables, and was the model chosen to be optimized. The new test with the ResNet34 model was trained with smaller images in the initial stages to speed up learning, and with larger images in the final stages to increase its accuracy, resulting in a final accuracy of 91.97%. For better results, it is suggested to increase the quality and size of the datasets, as well as to extensively test more optimal models and parameters.

Keywords: *Computer Vision*, *Neural Network*, *Waste Classification, TrashNet*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

RSU – Resíduo Sólido Urbano

CNN – *Convolutional Neural Network* (Rede Neural Convolucional)

ANN – *Artificial Neural Network* (Rede Neural Artificial)

DNN – *Deep Neural Network* (Rede Neural Profunda)

VP – Verdadeiro Positivo

FP – Falso Positivo

VN – Verdadeiro Negativo

FN – Falso Negativo

SUMÁRIO

1 Introdução

A correta gestão, coleta e descarte do lixo urbano faz-se cada vez mais necessária para a redução do impacto ambiental e do risco à segurança e saúde, além da redução de gastos destinados ao descarte incorreto, que no Brasil chega a somar 1 bilhão de dólares por ano. Em 2019, a geração de RSU (resíduo sólido urbano) no país atingiu o volume de 79 milhões de toneladas geradas anualmente, um aumento de 19% comparado à 2010, e mesmo com uma coleta de 92% desse total, mais de 40% de todo resíduo coletado é descartado incorretamente.

Compondo em média 35% de todo o resíduo gerado no país, materiais recicláveis possuem a vantagem de serem reprocessáveis, reduzindo o descarte de lixo e consumo de recursos através dos processos de reciclagem. No entanto, mesmo com todo os incentivos e campanhas governamentais realizados nos últimos anos, 27% dos municípios brasileiros não possuem nenhuma iniciativa de coleta seletiva, e a média nacional de reciclagem é inferior à 4%. (“Panorama 2020 – Abrelpe,” [2020])

Com um aumento previsto de 50% para a geração de RSU até 2050, comparado a 2019, é imprescindível o desenvolvimento de técnicas mais modernas e acessíveis em todas as etapas da gestão do lixo urbano. Uma frente bem popular a esse desafio vem sendo a utilização de *Deep Learning*, e mais especificamente, CNNs (*Convoluted Neural Network* – Rede Neural Convolucional), para a identificação e classificação do lixo por reconhecimento de imagem, com uma vasta gama de possibilidades e aplicações, a ser explorada mais a fundo na seção 2.5, Trabalhos com Aplicações Semelhantes.

As CNNs são um algoritmo computacional com capacidade de aprender a reconhecer e classificar imagens, quando treinado por uma entrada de dados. Uma rede deste tipo, desde que eficaz, poderia ser utilizada como inteligência artificial de auxílio na hora de separar lixo doméstico. A integração de tal algoritmo a um aplicativo de celular, por exemplo, integrado a uma funcionalidade de geolocalização e acesso a bases de dados de localização de pontos de descarte adequado, seria um ótimo instrumento para reduzir os impactos negativos gerados pelo descarte incorreto do lixo doméstico no dia-a-dia.

Assim, este estudo propõe a análise de viabilidade de uso de uma CNN para a classificação de lixo reciclável doméstico, para auxiliar na separação e coleta seletiva. O objetivo inicial é obter uma rede neural com alta precisão na classificação de lixo reciclável, mas através de uma metodologia flexível que permita adaptação posterior da rede para aplicações mais abrangentes como a identificação de lixo não-reciclável (restos) e lixo especial (pilhas, remédios, entre outros).

2 Revisão Bibliográfica

Nesse capítulo será realizado uma apresentação sobre conceitos genéricos *de Machine* e *Deep Learning*, o que define e como funciona uma ANN (*Artificial Neural Network* – Rede Neural Artificial), o que diferencia uma CNN de uma ANN genérica e como ela opera para identificar imagens digitais.

2.1 Introdução à *Machine Learning* e *Deep Learning*

2.1.1 Machine Learning

Em termos gerais, o Aprendizado de Máquina (*Machine Learning)* é uma área das ciências da computação que se utiliza de dados e algoritmos para automatizar a criação de modelos analíticos. Através de algoritmos que se adaptam iterativamente a cada vez que recebem dados, entre outros, é possível encontrar padrões ocultos nos dados, que antes eram desconhecidos.

Os métodos de aprendizado de máquina podem ser classificados como supervisionado, não-supervisionado e semi-supervisionado:

- Supervisionado: o algoritmo é treinado com um conjunto de dados já classificados, e os seus parâmetros vão sendo ajustados iterativamente de modo a melhorar a capacidade de classificação do modelo em vista dos dados informados;
- Não-supervisionado: o algoritmo é treinado com um conjunto de dados não classificados, e realiza o agrupamento desses dados em grupos menores, conforme vai descobrindo padrões escondidos entre os dados;
- Semi-supervisionado: é um meio termo entre os dois anteriores, que inicia o treinamento com dados já classificados, para guiar o treinamento posterior com dados não classificados. É utilizado quando quer realizar um treino supervisionado, mas não possui quantidade suficiente de dados classificados;

2.1.2 Deep Learning

É uma categoria mais específica de *Machine Learning*, inspirada nas conexões neurais do cérebro humano. É formado por um conjunto multicamada de ANNs, onde os neurônios artificiais de uma camada estão conectados e interagindo com os neurônios da camada seguinte.

2.2 Conceitos gerais de *Machine Learning* Supervisionado

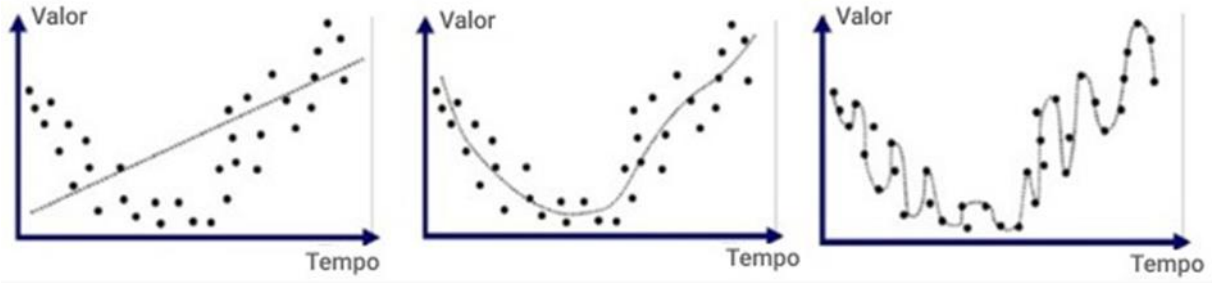
Antes de realizar o treinamento, é necessário dividir o conjunto de dados em três grupos. O grupo de treinamento, contendo a maior parte dos dados, é o utilizado para treinar os parâmetros do modelo, o grupo de validação é utilizado para avaliar o ajuste de hiperparâmetros (parâmetros definidos pelo usuário antes do treinamento) e por fim o

grupo de teste é empregado para calcular a performance final do modelo, ou seja, sua capacidade de generalização.

2.2.1 Sub-ajuste e Sobre-ajuste

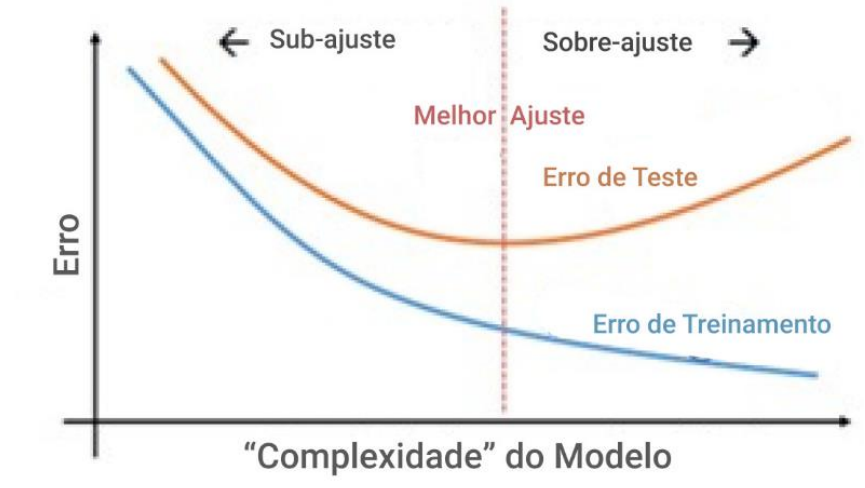
O termo sub-ajuste indica uma situação na qual o modelo não aprendeu suficientemente com os dados de treinamento, ou seja, não foi capaz de extrair toda a informação contida neles, e apresenta erro elevado tanto para os dados de treino quanto de teste. O sobre-ajuste, por outro lado, ocorre quando o modelo se adapta demais aos dados de treino e não consegue identificar dados diferentes, resultando em um erro muito baixo para os dados de treino, mas muito elevados para os de teste. Em geral, esta situação ocorre quando características aleatórias do conjunto de teste são incorporadas à estrutura do modelo.

Figura 1: da esquerda para direita, sub-ajuste, bom ajuste e sobre-ajuste de um modelo.



Comparando-se entre a performance do modelo com os dados de treino e teste, pode-se avaliar o ajuste, conforme o gráfico visto na Figura 2**:** .

Figura 2: Erro X Complexidade do modelo para os dados de treino e teste.



2.2.2 Modelo de Classificação

Para treinar um modelo que classifique os dados entre diferentes classes, o algoritmo é treinado com uma série de dados categóricos, não-contínuos, na qual ele tenta atribuir uma decisão binária se o dado pertence ou não a cada uma das classes.

Considerando um modelo simples de classificação binária, no qual o objetivo do modelo é classificar uma certa observação como verdadeira ou falsa, o resultado de todas as respostas pode ser categorizado na chamada Matriz de Confusão, conforme Figura 3 abaixo:

Figura 3: Matriz de Confusão de um modelo de classificação binário.

		Detectada	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (VN)

A categoria Verdadeiro Positivo (VP) indica quando o modelo classifica corretamente o dado da classe positiva, Verdadeiro Negativo (VN) quando o modelo classifica corretamente o da classe negativa, Falso Positivo (FP) quando o modelo classifica incorretamente o dado da classe negativa como sendo da positiva (chamado também de Erro Tipo I) e Falso Negativo (FN) como quando o modelo classifica incorretamente o dado da classe positiva como sendo da negativa (Erro Tipo II). Considerando essa Matriz de Confusão, podem ser utilizadas as seguintes métricas para avaliar a performance do modelo, dependendo do tipo de conjunto de dados que está sendo analisado:

Acurácia: dentre todas as classificações, quantas o modelo acertou;

Recall: dentre todas as situações da classe positiva como valor esperado (Verdadeiros Positivos e Falsos Negativos), quantas o modelo acertou;

Precisão: dentre todas as classificações de classe positiva que o modelo fez (Verdadeiros Positivos e Falsos Positivos), quantas o modelo acertou;

F1-Score: é a média harmônica de Precisão e Recall;

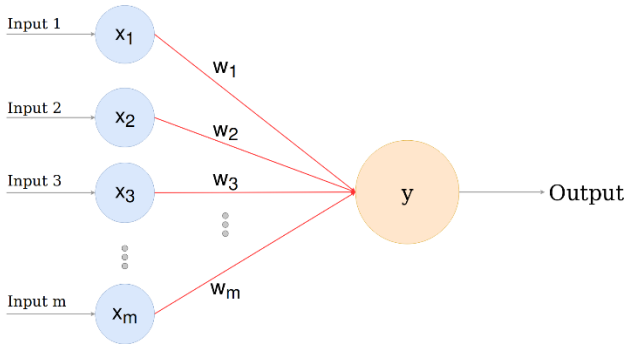
2.3 Introdução a Redes Neurais Artificias (ANNs)

2.3.1 Perceptron e Redes Neurais

O Perceptron é uma rede neural de camada única, desenvolvido na década de 1950 pelo cientista Frank Rosenblatt, inspirado nos mecanismos de aprendizado do neurônio humano.(Minsky e Papert, 2017)

O Perceptron é um modelo matemático que recebe uma ou várias entradas, atribui pesos e um viés para cada entrada, e produz uma única saída. O treinamento de um Perceptron corresponde a um ajuste de seus parâmetros, feito alimentando a rede com dados de entradas e as saídas esperadas.

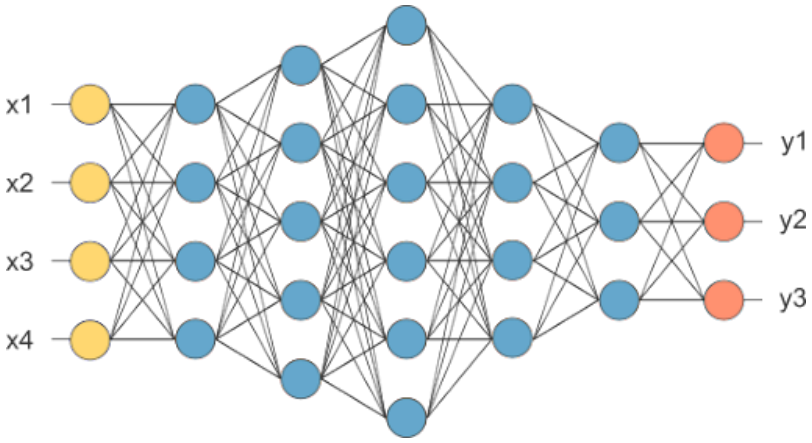
Figura 4: Modelo de um Perceptron recebendo entradas (Inputs), atribuindo pesos (w) e retornando um valor de saída (output).



Fonte: https://towardsdatascience.com/the-perceptron-3af34c84838c

Uma ANN, então, pode ser constituída de um perceptron, ou um conjunto de perceptrons em uma mesma camada, ou até múltiplas camadas. Todo modelo de ANN tem, necessariamente, uma camada de entrada e uma camada de saída, no entanto, ANNs com duas ou mais camadas ocultas já podem ser consideradas DNNs (*Deep Neural Networks* – Redes Neurais Profundas), e seu treinamento é classificado como *deep learning*.

Figura 5: DNN com 5 camadas ocultas.



Fonte: http://www2.decom.ufop.br/imobilis/fundamentos-de-redes-neurais/

2.3.2 Funções de Ativação e Propagação Direta

Considerando um neurônio qualquer, que recebe múltiplas entradas:

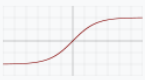
$$z = w^T x + b$$

Onde z é o valor de saída, w^T é o vetor de pesos, x é o vetor de entrada e b é o viés.

As funções de ativação, lineares ou não, são funções que conferem a característica de aprendizado ao neurônio e, em geral, limitam a saída do mesmo em valores entre 0 e 1, ou -1 e 1. Recebem esse nome pois tem a função de "ativar" o funcionamento do neurônio, caso o valor deste seja relevante para a rede.

Existem inúmeros tipos de função de ativação, e sua aplicação depende do tipo de rede que se deseja criar. No geral, camadas ocultas possuem todas a mesma função, e a função da camada de entrada e saída depende do tipo de entrada e saída do modelo.

Figura 6: alguns exemplos de funções de ativação.

Name	Plot	Function, $f(x)$	Derivative of f, $f'(x)$	Range
Identity		x	1	$(-\infty, \infty)$
Binary step		$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$\begin{cases} 0 & \text{if } x \neq 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$\{0, 1\}$
Logistic, sigmoid, or soft step		$\sigma(x) = \frac{1}{1+e^{-x}}$	$f(x)(1-f(x))$	$(0, 1)$
Hyperbolic tangent (tanh)		$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$1 - f(x)^2$	$(-1, 1)$
Rectified linear unit (ReLU)[11]		$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ $= \max\{0, x\} = x\mathbf{1}_{x>0}$	$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$[0, \infty)$

Fonte: https://en.wikipedia.org/wiki/Activation_function

Com a aplicação da Função de Ativação, a saída de um neurônio qualquer é definida por:

$$a = \sigma(w^T x + b)$$

Onde σ é a função de ativação e a é o valor da saída normalizada.

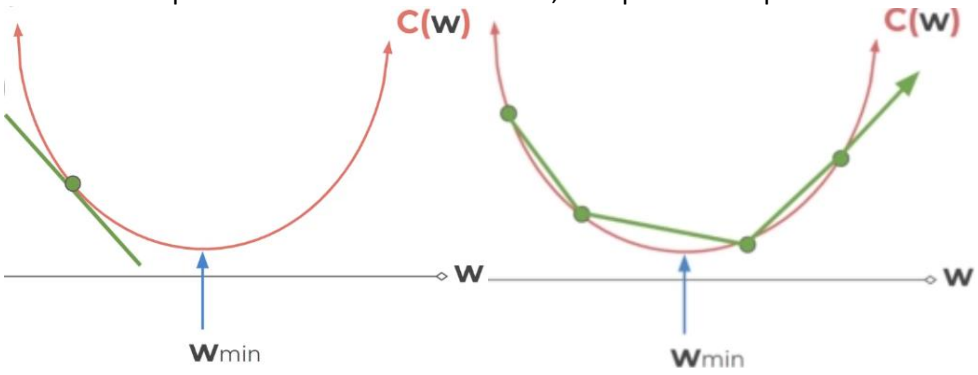
Para uma ANN multicamada, cada neurônio da camada de entrada recebe os dados e passa o seu valor de saída para os neurônios que estão conectados da camada seguinte, Esse processo de propagação direta (*feedforward)* ocorre camada a camada até atingir a saída da rede.

2.3.3 Taxa de Aprendizado e Funções de Custo e Otimização

A Função de Custo (ou Perda) C de uma ANN monitora a performance do treinamento, atribuindo um valor para a diferença dos valores previstos e dos valores reais. Essa função normalmente é definida como a diferença média absoluta ou quadrática. Em termos gerais, $C(W, B, S, E)$, onde W é a matriz de pesos da rede, B é a matriz de vieses, S é o conjunto de dados alimentados e E é conjunto de valores reais dos dados.

Para minimizar a perda, deseja-se encontrar o valor de W para que o custo atinja seu valor mínimo. Exemplificando-se para um ANN com um único peso, *w*, pode-se calcular a inclinação de *C(w)*, e aplicar um passo para estimar o próximo valor de *w* na direção da inclinação, até convergir para a menor perda possível. Esse passo para estimar os valores de *w* é chamado de taxa de aprendizado, e enquanto uma taxa muito pequena causa, em geral, uma convergência lenta, uma taxa muito grande corre o risco de ultrapassar o valor ótimo, conforme exemplificado na Figura 7.

Figura 7: taxa de aprendizado estimada muito alta, ultrapassando o ponto ótimo.



Para esse caso simplificado, o ideal seria reduzir o tamanho do passo conforme o gradiente se aproxima de 0. Mas como normalmente ANNs possum múltiplos pesos, existem diversas funções de gradientes mais complexas para esse fim, denominadas funções de otimização.

2.3.4 Retropropagação

Durante o treinamento, após realizar uma previsão da saída, deseja-se saber como a função de custo depende dos pesos e vieses, para que possa atualizá-los com objetivo de minimizar o erro de predição. Então considerando-se uma ANN com L camadas, com um neurônio por camada, tem-se que:

$$z^L = w^L\, a^{L-1} + b^L$$

$$a^L = \sigma(z^L)$$

$$C(a^L) = C(diferença\ absoluta\ ou\ quadrática\ de\ a^L\ com\ y)$$

Onde y é o valor real do dado alimentado para o treinamento da rede.

Deseja-se analisar o efeito dos pesos e vieses em *C* através das seguintes derivadas parciais:

$$\frac{\partial C}{\partial w^L} = \frac{\partial z^L}{\partial w^L}\frac{\partial a^L}{\partial z^L}\frac{\partial C}{\partial a^L}$$

$$\frac{\partial C}{\partial b^L} = \frac{\partial z^L}{\partial b^L}\frac{\partial a^L}{\partial z^L}\frac{\partial C}{\partial a^L}$$

Definindo-se: $\delta^l = \frac{\partial C}{\partial z^l}$ como o termo de erro do neurônio da camada l, então, obtendo-se o erro do neurônio da camada de saída (L):

$$\delta^L = \frac{\partial y}{\partial a^L}\sigma'(z^L)$$

Todas as quantidades nesta equação podem ser obtidas a partir de informações da camada de saída. Então expandindo-se a equação na forma matricial:

$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

Onde $\odot$ é o produto de Hadamard, indicando a multiplicação elemento a elemento dos termos. Assim, é possível ajustar-se os pesos e vieses resolvendo essa equação da camada de saída até a camada de entrada, de forma recursiva, por isso o nome de retropropagação (*backpropagation*).

Definindo-se a função de custo como a diferença média quadrática: $\nabla_a C = (a^L - y)$

A equação de retropropagação para os neurônios da camada oculta l fica:

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^L)$$

2.4 Introdução a Rede Neural Convolucional (CNN)

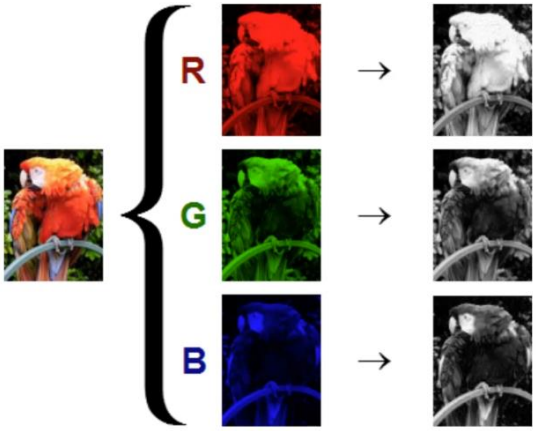
2.4.1 Conceitos de Imagens Digitais

Uma imagem digital se trata de uma matriz composta por pixels, que podem atingir diferentes tons de cores e brilhos. Para o tratamento através de CNN, o conceito de imagem será abordado de forma numérica, considerando a imagem como um tensor de 3 dimensões, sendo elas: altura, largura e canal.

Imagens mais simples, como em preto e branco e escalas de cinza, possuem só um canal de cor, e, portanto, podem ser consideradas como matrizes, cujos valores variam de 0 à 255 (ou de 0 à 1 para imagens binárias), que definem a intensidade do pixel. Tratando-se de imagens coloridas, estas possuem normalmente 3 canais de cores, cujos valores também variam entre 0 e 255, e a sobreposição desses canais gera a imagem final.

Na Figura 8 tem-se um exemplo de uma imagem colorida e sua decomposição nos canais RGB (vermelho, verde e azul), na qual cada canal corresponde a intensidade da sua respectiva cor principal na imagem final. A imagem em cinza é a forma como o canal individual é interpretado digitalmente.

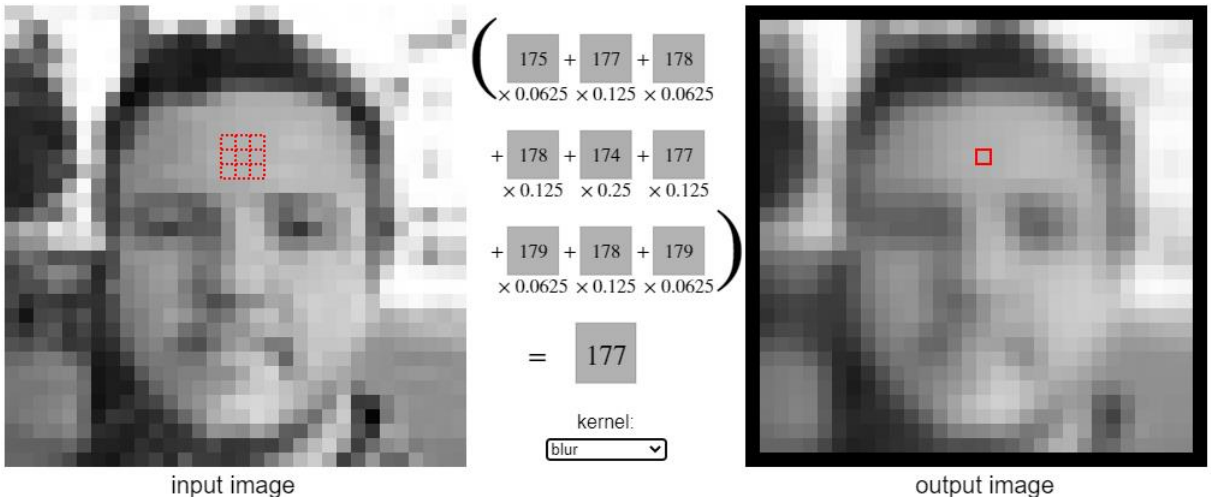
Figura 8: imagem colorida, decomposta nos três canais RGB.



Fonte: https://commons.wikimedia.org/

Um filtro de imagem é uma transformação da matriz original através de operações com outra matriz, de dimensão menor, denominada de máscara ou *kernel*. A máscara percorre toda a imagem, da esquerda pra direita, e cima para baixo, realizando uma multiplicação de matrizes entre a máscara e a porção da imagem que está percorrendo.

Figura 9: valores e operações de uma máscara 3x3 aplicando o efeito de *blur*.



Fonte: https://setosa.io/ev/image-kernels/

2.4.2 CNNs e Camadas de Convolução

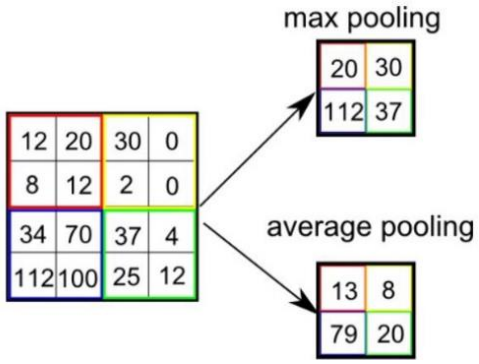
O motivo pelo qual não se recomenda o uso ANNs genéricas para a classificação de imagens é a necessidade de serializar a imagem, que é composta por duas ou três dimensões, em um vetor de dimensão única para alimentar os dados na rede. Isto faz com que se perca a relação espacial entre os pixels da imagem. Além disso, em função da grande quantidade de dados envolvida, o treinamento seria muito custoso do ponto de vista computacional.

O diferencial da CNN é sua camada convolucional na qual, ao invés dos pesos de cada conexão agirem sobre os valores da imagem em si, eles atuam como pesos de uma máscara aleatória aplicada na imagem, e o treinamento da rede torna-se a aplicação de várias máscaras filtrando a imagem em busca de características relevantes. Além disso, sua estrutura de conexões localizadas faz com que os neurônios sejam conectados somente a um subconjunto local da próxima camada, fazendo com que as máscaras se tornem um filtro completo.

Uma ferramenta muito importante da CNN é a camada de *pooling*, logo após a camada convolucional, que serve como uma condensação dos parâmetros recebidos mais relevantes em uma máscara menor, reduzindo muito a quantidade de parâmetros da rede (

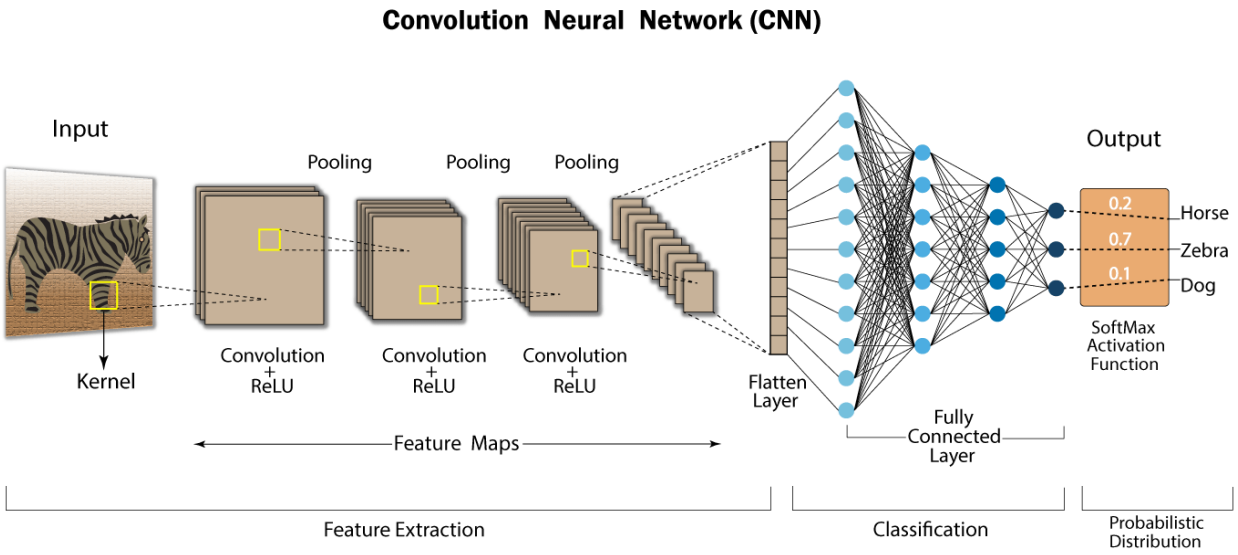
Figura 10). Outra técnica muito comum empregada em CNN é o *dropout*, na qual subconjuntos da rede são randomicamente desativados para evitar o sobreajuste.

Figura 10: exemplo de *pooling* condensando os parâmetros por média ou máximo.



Fonte: *https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/*

Figura 11: esquema simplificado de uma CNN e suas camadas.



Fonte: https://www.analyticsvidhya.com/blog/2021/05/20-questions-to-test-your-skills-on-cnn-convolutional-neural-networks/

2.5 Trabalhos Com Aplicações Semelhantes

2.5.1 Classificação de Lixo para Status de Reciclabilidade

O estudo de YANG e THUNG (2016) selecionou um conjunto de mais de 2.500 imagens, dividido em 6 classes (papelão, papel, plástico, vidro, metal e restos). O conjunto de dados popularizou-se em aplicações de CNN, sendo utilizada em outros trabalhos voltados para a classificação de lixo.

O conjunto de imagens representa uma série de fotos de lixo caseiro, com uma boa variação de objetos para cada tipo de classe, cada foto contendo somente um objeto posando sobre fundo branco.

O objetivo dos autores foi treinar um modelo de classificação de lixo utilizando SVM (*Support Vector Machine* – Máquina de Vetores de Suporte) e CNN, obtendo 63% e 22% de acurácia, respectivamente. Ao final, concluíram que, para classificar corretamente algo como lixo, com tantas classes diferentes, era essencial um conjunto de dados maior no caso da CNN.

2.5.2 Classificação de Bandejas de Comida com Deep Learning

SOUSA et al (2019) desenvolveram uma CNN para identificar o tipo de material e a forma de lixo deixados em bandejas de alimentos de praças de alimentação. A partir de um conjunto de 1002 imagens próprias, classificadas entre 4 classes de material (plástico, papel, vidro e metal) e 10 classes de formato (copo, garrafa, talheres, misto, entre outros), desenvolveram uma CNN hierárquica que classificava inicialmente por um tipo de classe, e posteriormente para os tipos da outra classe. Deste modo, um material classificado como papel não poderia ser classificado como talher, ou um formato classificado de copo não poderia ser classificado como metal.

Figura 12: exemplos de lixo identificado em bandejas de comida.

Fonte: (Sousa et al., 2019)

Treinando o modelo de três formas diferentes: classificando por formato e material simultaneamente, por formato primeiro e por material primeiro, a CNN de formato obteve uma precisão média de 86%, comparada aos 81% da classificação primária por material e 74% da classificação simultânea. Os autores sugeriram o aumento base de dados e o teste de novas arquiteturas em trabalhos futuros.

3 Metodologia

Nesta seção serão apresentadas as informações referentes ao material, *softwares*, conjuntos de dados e técnicas utilizadas para o desenvolvimento do trabalho.

3.1 *Softwares*

O presente estudo foi realizado utilizando a linguagem de programação Python, versão 3.8, na plataforma online Google Colab. O desenvolvimento das CNNs foi feitoatravés da biblioteca fast.ai, construída em cima da PyTorch.

Outras bibliotecas auxiliares foram: NumPy e Pandas para tratamento de dados, Matplotlib e Seaborn para construção de gráficos, Scikit-learn para construção das matrizes de confusão e Glob2 para manejo de pastas e arquivos.

3.2 Conjunto de Dados

O conjunto de dados principal empregado no trabalho foi o *TrashNet* ("GitHub - garythung/trashnet: Dataset of images of trash; Torch-based CNN for garbage image classification," [s.d.]), contendo mais de 2.500 imagens de boa qualidade, com todas as imagens de mesmo tamanho e sem muito ruído externo, separadas em 6 classes de lixo (papel, papelão, plástico, vidro, metal e restos).

Também se utilizou de 1710 imagens do conjunto de dados de ("waste_pictures | Kaggle," [s.d.]), onde as imagens de latas, papelão, garrafas de vidro, caixa de leite, guardanapo, jornal, garrafas de plástico e sacolas plásticas foram organizadas nas 6 classes principais do *TrashNet*. Também se criou uma nova classe denominada "especial", com imagens de lâmpadas e baterias, para designar um pequeno conjunto de materiais que são comumente confundidos como recicláveis, mas deveriam ser descartados em pontos de coletas específicos.

Os dados do segundo conjunto foram coletados utilizando *web-scraping* e, portanto, algumas imagens são diferentes do que seria uma foto real de lixo doméstico. No entanto a expansão do conjunto de dados foi o principal ponto de melhoria citado pelos criadores do *TrashNet*, e a adição desses dados extras é um passo nesse sentido. Deve-se observar que esta adição aumenta a abrangência da CNN ao custo da maior acurácia que teria com um conjunto de esta adição aumenta a dados menor.

As 7 classes são propostas em acordo com o objetivo desse trabalho, pois as 5 classes recicláveis (papelão, papel, plástico, vidro e metal) estão conforme a divisão feita para triagem de material reciclável, e a classe de especial e restos são importantes para não correr o risco de designar incorretamente esses tipos de materiais para reciclagem ("DMLU,").

3.3 Treinamento Inicial

Realizou-se incialmente um teste somente com os dados do *TrashNet*, a fim de se obter uma avaliação prévia do tipo de dados necessários, a definição de parâmetros e o tempo de execução.

3.3.1 Organização e tratamento de dados

O conjunto de dados foi separado aleatoriamente em 50% treino para (1262 imagens), 25% para validação (630) e 25% para teste (635). Para aumentar artificialmente o número de dados, realizou-se uma etapa chamada *Data Augmentation* (Aumento de Dados), que corresponde a pequenas mudanças aplicadas aleatoriamente nas imagens a cada ciclo do treinamento, como rotacionar, espelhar, esticar, comprimir, cortar e dar zoom, com o objetivo de aumentar artificialmente o conjunto de dados (Shorten e Khoshgoftaar, 2019).

Por se tratar de imagens de lixo reciclável, diversas opções poderiam ser aplicadas na etapa de *Data Augmentation,* visto que fotos reais de lixo podem ser capturadas de diversas maneiras e ainda devem ser reconhecidas como lixo, independente se está virado, amassado ou tenha outras imperfeições. Para esta etapa inicial, no entanto, aplicou-se a inversão horizontal e vertical das imagens.

Figura 13: exemplo de uma batelada de 16 imagens utilizadas para treinar a CNN,

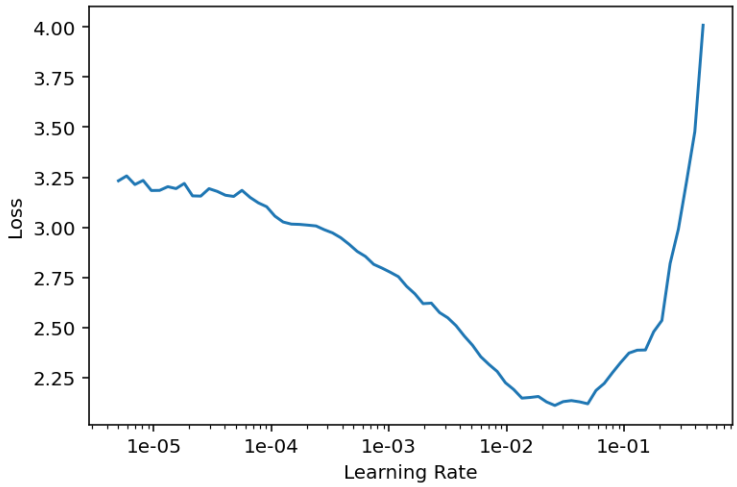
3.3.2 Configuração e Treinamento da Rede

Para a etapa inicial, escolheu-se o modelo ResNet34, uma CNN residual de 34 camadas, pré-treinada no conjunto de dados ImageNet. O termo residual significa que quando um de seus neurônios não possui resíduo (o neurônio apresentou um ajuste ótimo de pesos e viés), ele não será ajustado pela propagação, resultando num atalho entre as conexões. Além disso, uma rede pré-treinada apresenta melhor performance em novos treinamentos por já possuir algum aprendizado de padrões visuais.

Por padrão de redes de classificação, a função de custo utilizada foi a *CrossEntropyLoss* (“CrossEntropyLoss — PyTorch 1.10.0 documentation,” [s.d.]), e a função de otimização escolhida foi a Adam (Kingma e Ba, 2015).

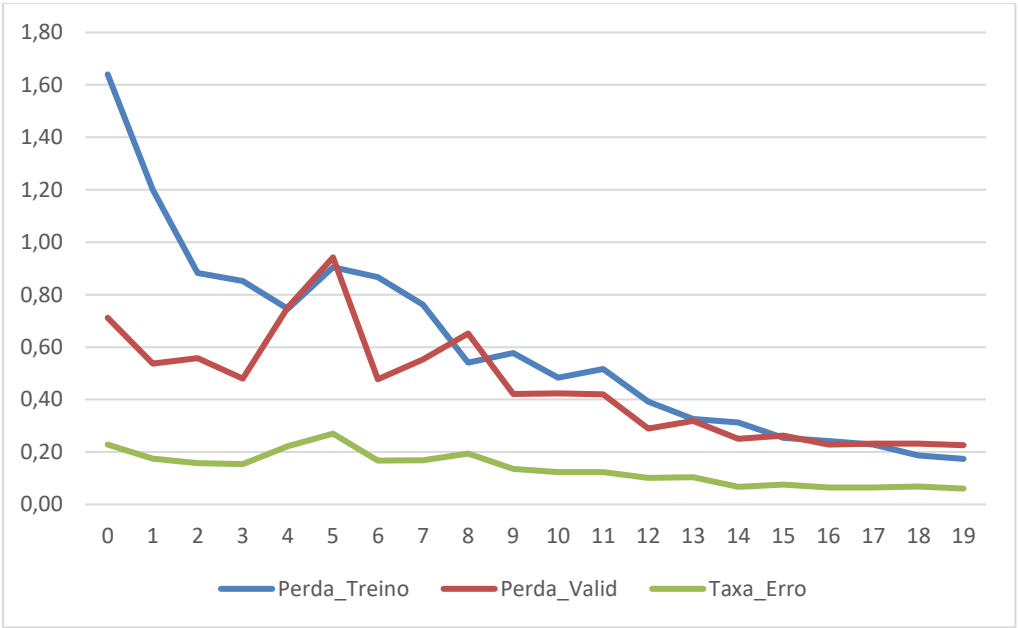
Escolheu-se treinar a rede com uma batelada de 16 imagens por vez, durante 20 ciclos, para economizar o processamento. Antes de realizar o treinamento, analisou-se a curva de taxa de aprendizado em função do erro, definindo-se uma taxa para que o gradiente convirja rapidamente sem perder o ponto ótimo. Conforme o gráfico da Figura 14, escolheu-se a taxa de aprendizado de 0,003.

Figura 14: curva da perda do treinamento por taxa de aprendizado.



A evolução da perda ao longo dos ciclos de treinamento é vista na Figura 15.

Figura 15: Status do treinamento da CNN.



3.3.3 Validação

Uma vez concluído o treinamento da rede, analisou-se os resultados das imagens de validação. Plotando um gráfico com as 9 previsões mais erradas (

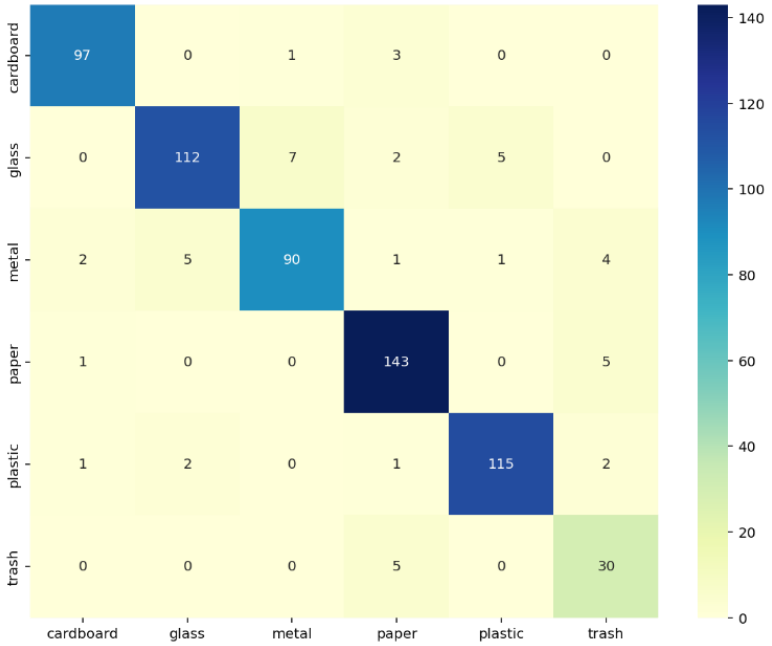
Figura 16), notou-se que muitas imagens são difíceis de serem classificadas apenas pelo aspecto visual, e verificando-se a matriz de confusão correspondente no Apêndice A (Figura A1), nota-se que a classe com mais erros de classificação foi o plástico, sendo muito confundido com vidro e metal.

Figura 16: previsões mais incorretas do teste inicial da rede.

3.3.4 Teste de Classificação

Aplicando a CNN treinada às imagens de teste, obteve-se uma acurácia de 92,44%. Plotando a matriz de confusão (Figura 15) notou-se que o material com mais erros de classificação foi o vidro, com 14 classificações erradas (11%).

Figura 15: matriz de confusão dos dados de teste.



4 Resultados

4.1 Testando outros Modelos para a CNN

Diversos trabalhos na literatura envolvem CNNs treinadas com o *TrashNet*, empregando diferentes parâmetros e com performances distintas. Nessa etapa do trabalho, adicionou-se 1710 imagens do conjunto de dados *Waste_Pictures* em suas respectivas classes do conjunto TrashNet, conforme detalhado no Capítulo 3.2, e testou-se cada um dos modelos pré-treinados disponíveis mais encontrados na literatura, conforme a Tabela 1:

Tabela 1: acurácia de vários modelos de CNN treinadas com *TrashNet* na literatura.

Trabalho	**Modelo**					
	VGG-16	**Densenet_121**	**Recycle_Net**	**Mobile_Net**	**ResNet**	**Inception**
(BIRCANOGLU *et al.*, 2018)		95%	81%	76%	75%	85%
(VO *et al.*, 2019)		91%	68%		72%	
(RUIZ, Victória et al., 2018)	77%				89%	94%
(ARAL *et al.*, 2019)		95%		84%		89%
(COSTA, Bernardo et al., 2018)	93%					

Para um teste preliminar com cada modelo, definiu-se:

- 70% dados de Treino, 20% dados de validação e 10% dados de teste;
- Tamanho das imagens de 120x120, inversão vertical e horizontal;
- Batelada de 16 imagens, 20 ciclos;

O resultado de cada modelo é exibido na Tabela 2, a seguir. As matrizes de confusão para os dados de validação e teste estão disponíveis nos Apêndices B à E.

Tabela 2: acurácia e piores materiais de cada modelo.

Modelo	Acurácia	Pior Acurácia	Pior Material	2ª Pior Acurácia	2º Pior Material
MobileNet_V2	73,02%	53,33%	Restos	57,39%	Vidro
DenseNet121	73,78%	46,67%	Restos	60,16%	Metal
Resnet34	84,99%	66,67%	Restos	76,52%	Vidro
VGG16	85,28%	47,67%	Restos	77,39%	Vidro

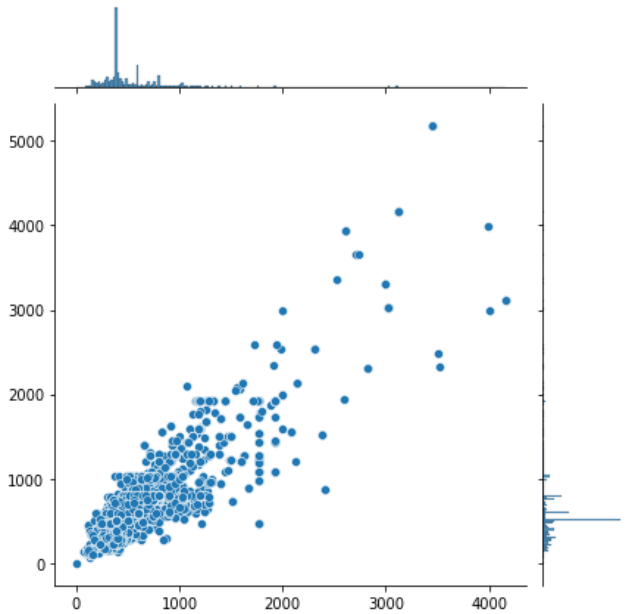
Para esse teste inicial, os modelos de Resnet34 e VGG16 obtiveram a melhor performance. Desta forma, escolheu-se o modelo Resnet34 para ser otimizado, devido a boa acurácia em geral e a melhor precisão para classificar restos. É importante mencionar que cada modelo recebeu uma divisão diferente e aleatório de imagens, e que a performance pode melhorar muito simplesmente ajustando-se os hiperparâmetros cada modelo.

4.2 Otimizando a CNN com ResNet34

Uma técnica experimental que tem se provado muito útil para obter melhores resultados em CNNs é o aumento gradual do tamanho das imagens conforme o treinamento. Imagens de menor resolução no início do treinamento facilitam na velocidade de aprendizado da rede e imagens maiores ao final permitem aprender mais detalhes.(Howard e Gugger, [s.d.])

Definiu-se o tamanho inferior para o início de treino e o tamanho superior para o final pela análise do gráfico da Figura 16:

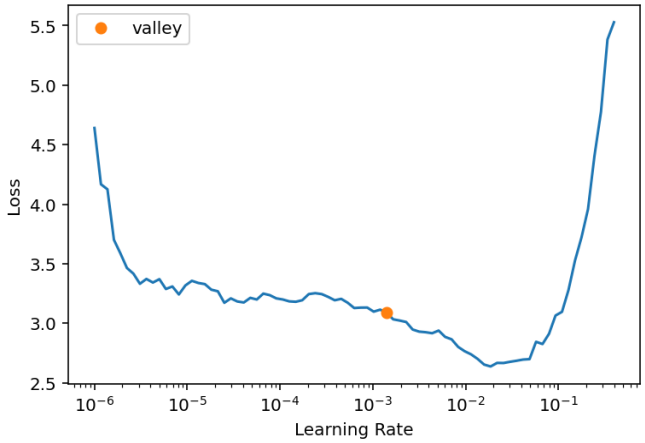
Figura 16: altura e largura de todas as imagens do conjunto de dados.



Com a grande maioria das imagens sendo aproximadamente 500x500, optou-se por realizar 10 ciclos inicias com imagens de 240x240 e 10 ciclos finais com 720x720. Definiu-se

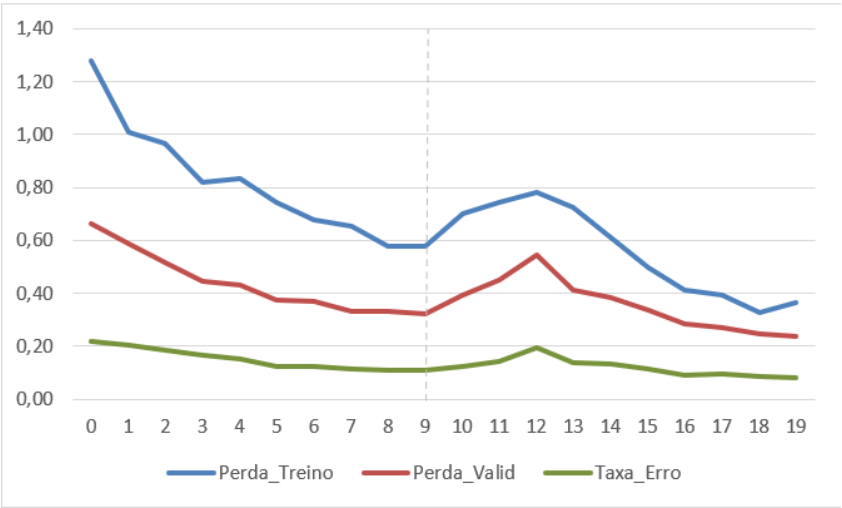
uma batelada de 8 imagens, e *data augmentation* com inversão, rotação, distorção, *blur*, zoom e diferença de brilho. A taxa de aprendizado definida foi de 0,001, conforme Figura 19.

Figura 19: curva da perda do treinamento por taxa de aprendizado.



O resultado do treinamento é demonstrado na Figura 20. Os 10 ciclos iniciais, treinados com imagens de dimensão 240x240, levaram apenas 3 minutos, enquanto que os 10 ciclos finais levaram 18 minutos. Observou-se que nos últimos ciclos com imagens menores, as perdas estavam quase se estabilizando, e, quando a rede começou a receber as imagens maiores, as perdas começaram a aumentar devido o ajuste aos novos dados.

Figura 20: progressão das funções de perdas e taxa de erro do treinamento da CNN otimizada.



O treinamento foi encerrado com uma acurácia de 93,13% para os dados de validação, e, devido as curvas de perdas decrescentes até o final do treinamento, não houve sobreajuste aparente do modelo. Conforme os dados da Tabela 3 , baseada na matriz de confusão da Figura 21**:** , a classificação da classe de restos foi bem mais acurada que os modelos anteriores, mas a precisão continua baixa, indicando que o modelo atual não é confiável. As classificações mais erradas podem ser vistas no Apêndice A, Figura A2**:** .

A diferença entre a acurácia e a precisão para restos, além da diferença entre a acurácia atual perante os modelos anteriores (Tabela 2), deve-se ao número de imagens na classe, muito menor se comparado às demais. Esses resultados reforçam a necessidade de escolher

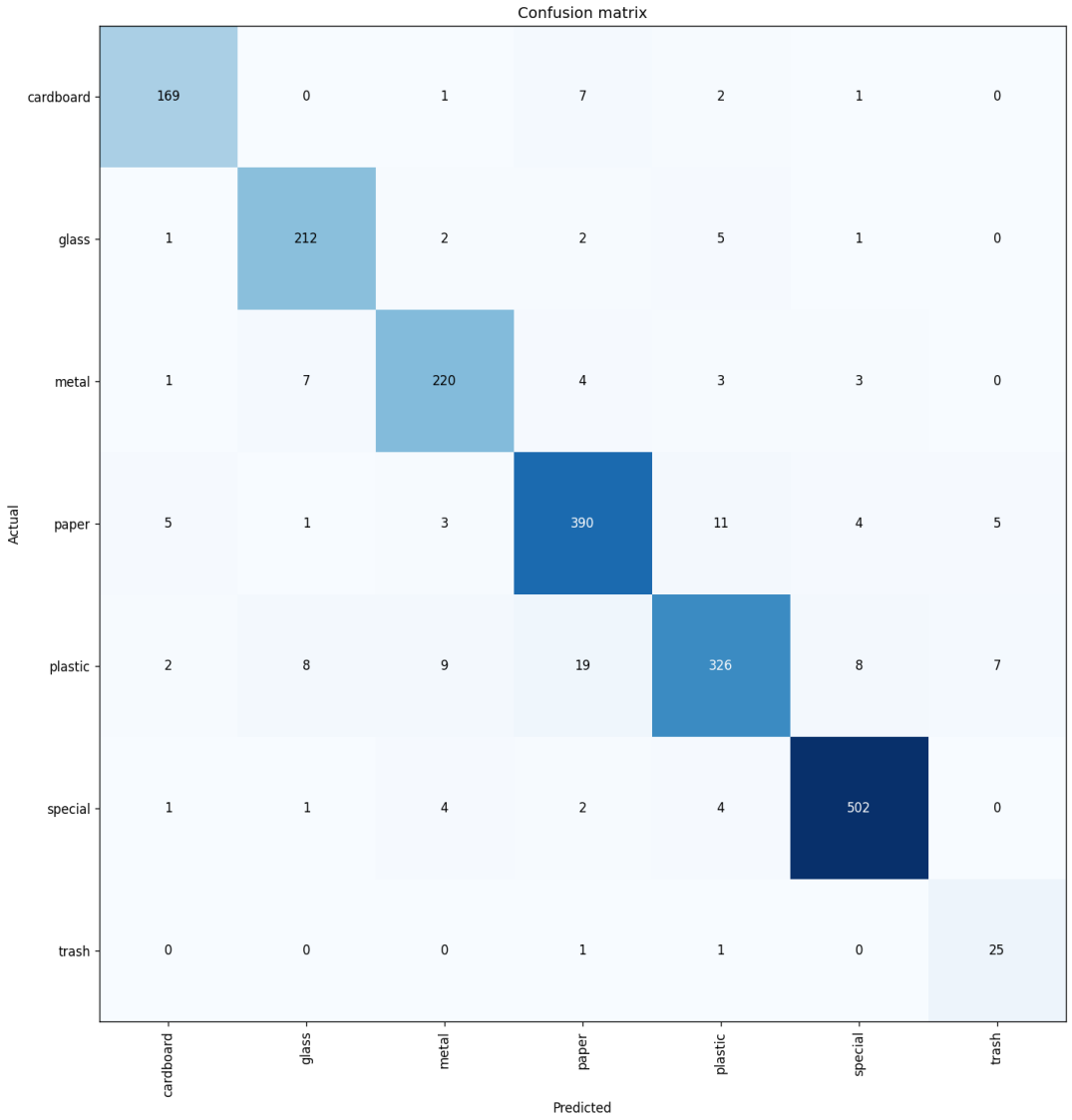
a métrica correta para analisar o modelo, além do impacto que um conjunto de dados desbalanceado causa.

Tabela 3: performance por classe da CNN otimizada.

Classe	Nº Imagens	Acurácia	Precisão
Papelão	180	93,9%	94,4%
Vidro	223	95,0%	91,8%
Metal	238	92,4%	92,1%
Papel	419	93,0%	91,8%
Plástico	379	86,0%	92,6%
Especial	514	97,7%	96,7%
Restos	27	92,6%	67,6%

Aplicando a CNN otimizada nos dados de teste, obteve-se uma acurácia de 91,97%, quase 7 pontos percentuais acima do modelo com melhor acurácia nos testes anteriores. Não foi possível gerar os mesmo gráficos e matriz de confusão nativamente para os dados de teste, devido à diferença de como o treinamento otimizado é implementado na biblioteca fast.ai.

Figura 21: matriz de confusão dos testes de validação da CNN otimizada.



5 Conclusões e Trabalhos Futuros

A acurácia atingida pelo modelo final foi de 91,97%. Com base nesse resultado, conclui-se que a otimização proposta melhorou o resultado da CNN, mas continua abaixo do alcançado por outras arquiteturas que treinaram somente com os dados do *TrashNet*, como o modelo DenseNet_121 de ((Aral et al., 2019; Vo et al., 2019).

Considerando o objetivo desse trabalho como o desenvolvimento inicial de uma CNN eficaz em classificar lixo doméstico, para posterior utilização como ferramenta de auxílio

pessoal na destinação correta de lixo, os resultados obtidos são satisfatórios para justificar mais estudos nesse meio. A performance atual da rede é insuficiente para a mesma ser implementada como a única agente em aplicações reais, mas com auxílio de outras ferramentas ou até interferência humana, ela pode ser viabilizada.

Aumentar o tamanho e a qualidade da base de dados é o ajuste que se faz mais essencial. Realizar um pré-processamento do conjunto de dados *Waste_Pictures*, retirando as imagens incorretas, seria um bom passo inicial para incrementar o conjunto do *TrashNet* sem adicionar ruído desnecessário. Outra possibilidade aquém é utilizar de outros bancos de dados destinados à classificação de lixo, como os disponibilizados em *Waste Datasets Review* ("GitHub - AgaMiko/waste-datasets-review: List of image datasets with any kind of litter, garbage, waste and trash," [s.d.]), após passarem por tratamentos e ajustes para adequarem-se aos padrões do *TrashNet.*

Acrescentar imagens de lixo de tipos mais variados, especialmente para as classes menos representadas no conjunto usado, faz-se necessário para aplicar os modelos às condições reais. O baixo número de imagens da classe de restos inviabiliza o desenvolvimento de redes com o objetivo de separar lixo reciclável de não reciclável. O fato de as imagens de metal serem em sua maioria latas de alumínio, e de que o tipo especial serem somente lâmpadas e baterias, também acrescenta um viés muito grande para os modelos treinados, que não são capazes de classificar objetos mais gerais dessa categoria.

Em relação à CNN, notou-se diferentes desempenhos entre os modelos testados. O baixo número de dados e os diferentes hiperparâmetros que podem ser ajustados para otimizar o treinamento tornam a escolha de um único modelo inconclusiva. Para definir os melhores parâmetros e ajustes na hora de treinar uma rede, muito mais testes se fazem necessário.

Apesar das melhorias necessárias, esse trabalho demonstrou a criação de um protótipo de performance semelhante a outros trabalhos e modelos discutidos na literatura, utilizando de um conjunto de dados mais variado e com mais classes para classificar. Os resultados iniciais obtidos e a flexibilidade com que se criou o modelo permite a oportunidade de melhoria contínua desse estudo em trabalhos futuros.

REFERÊNCIAS

ARAL, R. A.; KESKIN, S. R.; KAYA, M.; et al. Classification of TrashNet Dataset Based on Deep Learning Models. **Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018**, p. 2058–2062, 22 jan. 2019.

CrossEntropyLoss — PyTorch 1.10.0 documentation. Disponível em: <https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html>. Acesso em: 16 nov. 2021.

DMLU. Disponível em: <https://www2.portoalegre.rs.gov.br/dmlu/default.php?p_secao=188>. Acesso em: 10 nov. 2021.

GitHub - AgaMiko/waste-datasets-review: List of image datasets with any kind of litter, garbage, waste and trash. Disponível em: <https://github.com/AgaMiko/waste-datasets-review>. Acesso em: 12 nov. 2021.

GitHub - garythung/trashnet: Dataset of images of trash; Torch-based CNN for garbage image classification. Disponível em: <https://github.com/garythung/trashnet>. Acesso em: 12 nov. 2021.

HOWARD, J.; GUGGER, S. Deep learning for coders with fastai and PyTorch : AI applications without a PhD. p. 594, [s.d.].

KINGMA, D. P.; BA, J. L. Adam: A method for stochastic optimization. **3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings**, 2015.

MINSKY, M.; PAPERT, S. A. **Perceptrons, Reissue of the 1988 Expanded Edition with a new foreword by Léon Bottou: An Introduction to Computational Geometry**. [s.l.] MIT Press, 2017.

Panorama 2020 – Abrelpe. Disponível em: <https://abrelpe.org.br/panorama-2020/>. Acesso em: 10 nov. 2021.

SHORTEN, C.; KHOSHGOFTAAR, T. M. A survey on Image Data Augmentation for Deep Learning. **Journal of Big Data**, v. 6, n. 1, p. 1–48, 1 dez. 2019.

SOUSA, J.; REBELO, A.; CARDOSO, J. S. Automation of Waste Sorting with Deep Learning. **Proceedings - 15th Workshop of Computer Vision, WVC 2019**, p. 43–48, 1 set. 2019.

VO, A. H.; HOANG SON, L.; VO, M. T.; et al. A Novel Framework for Trash Classification Using Deep Transfer Learning. **IEEE Access**, v. 7, p. 178631–178639, 2019.

waste_pictures | Kaggle. Disponível em: <https://www.kaggle.com/wangziang/waste-pictures>. Acesso em: 13 nov. 2021.

APÊNDICE A

Figura A1: matriz de confusão da rede ResNet34 inicial.

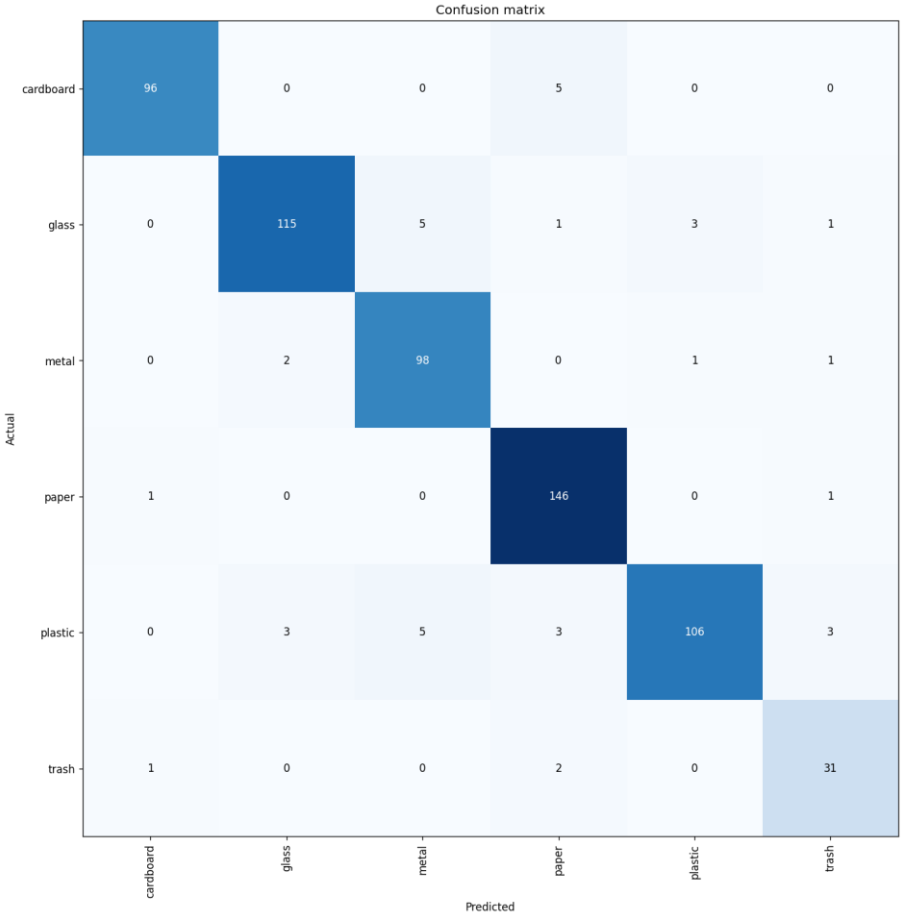


Figura A2: previsões mais erradas da rede otimizada.

APÊNDICE B

Figura B1: matriz de confusão do teste da rede ResNet34, para os dados de validação.

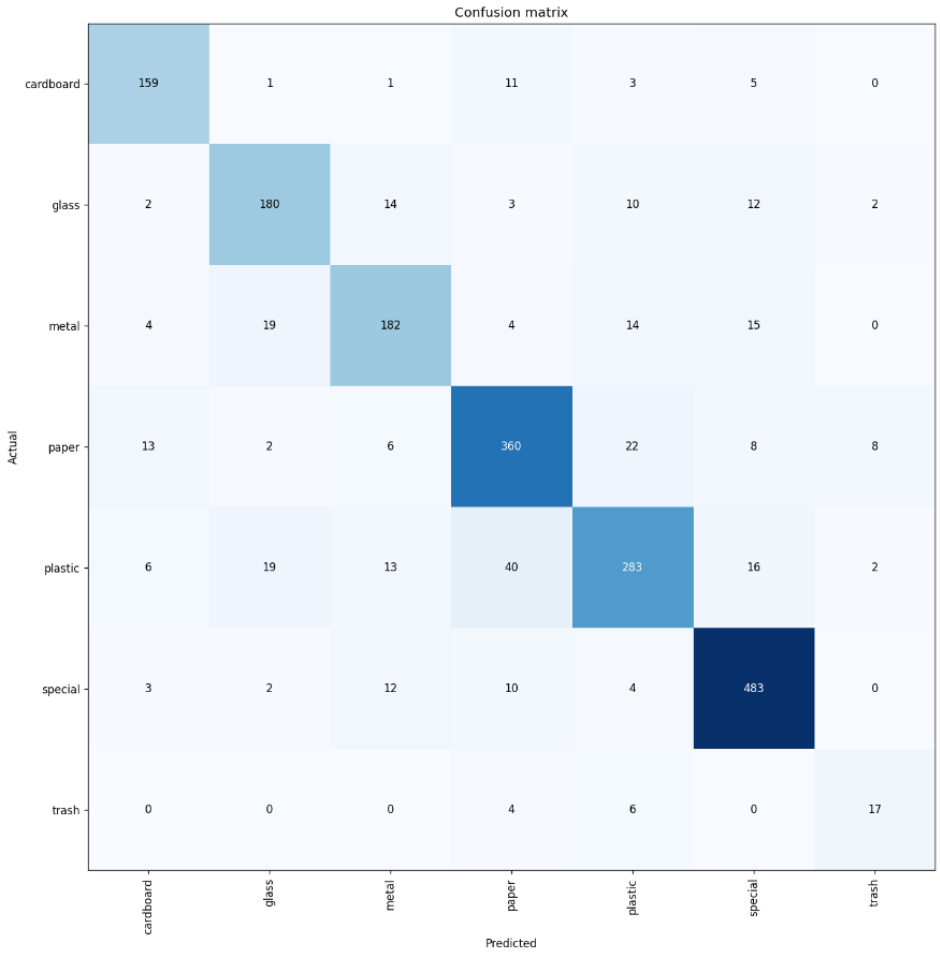
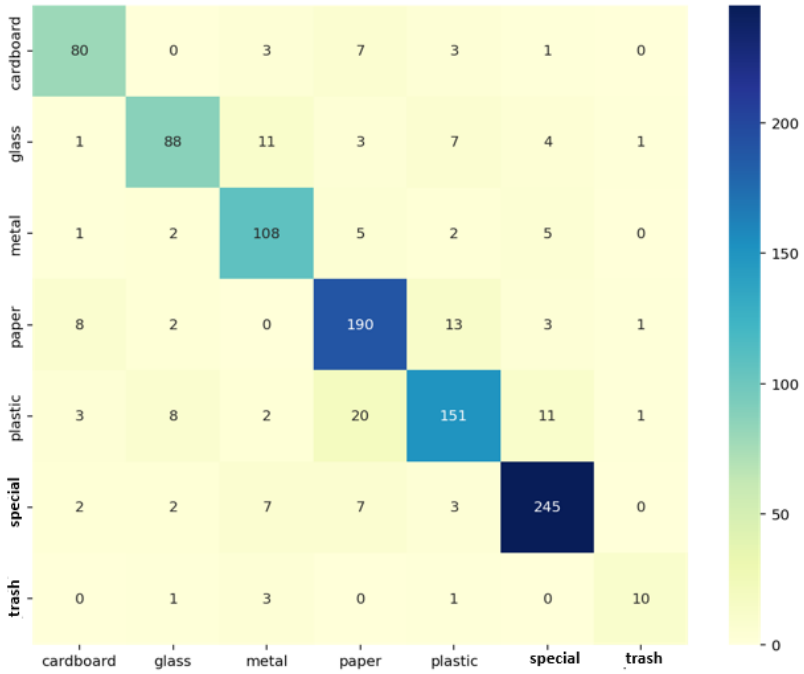


Figura B2: matriz de confusão do teste da rede ResNet34, para os dados de teste.



APÊNDICE C

Figura C1: matriz de confusão do teste da rede VGG16, para os dados de validação.

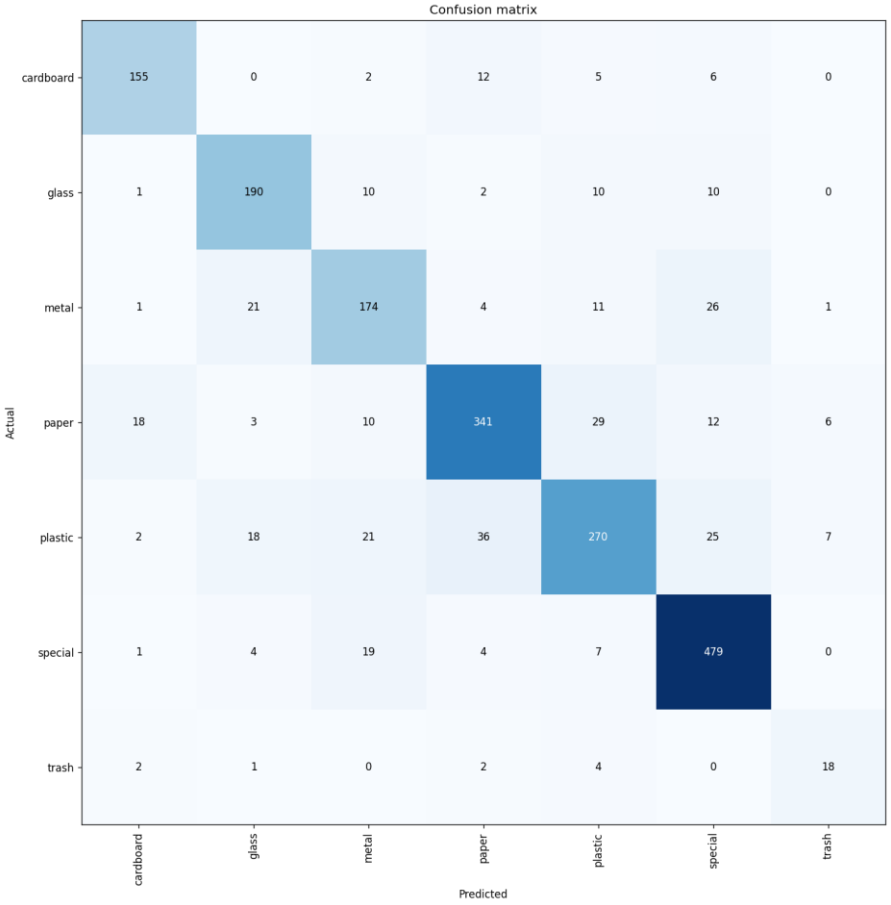
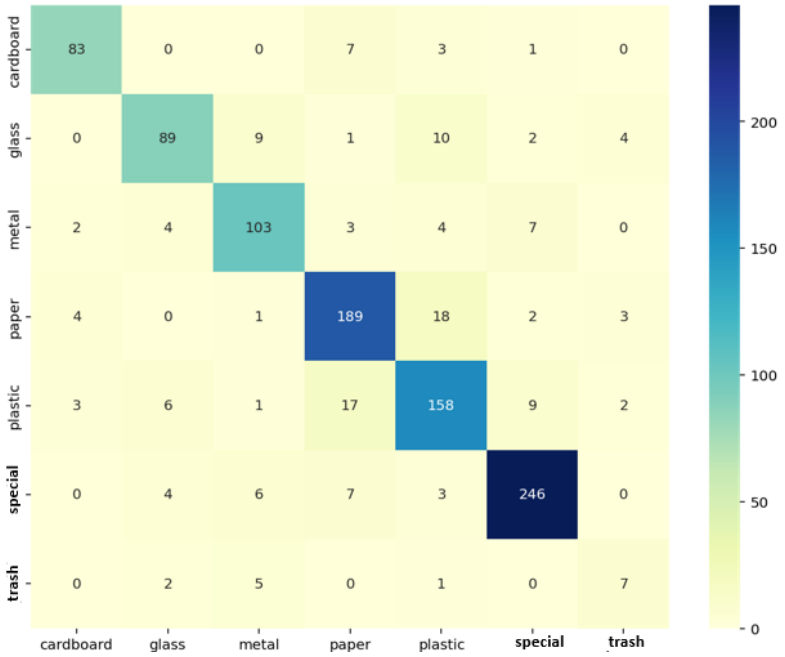


Figura C2: matriz de confusão do teste da rede VGG16, para os dados de teste.



APÊNDICE D

Figura D1: matriz de confusão do teste da rede DenseNet121, para os dados de validação.

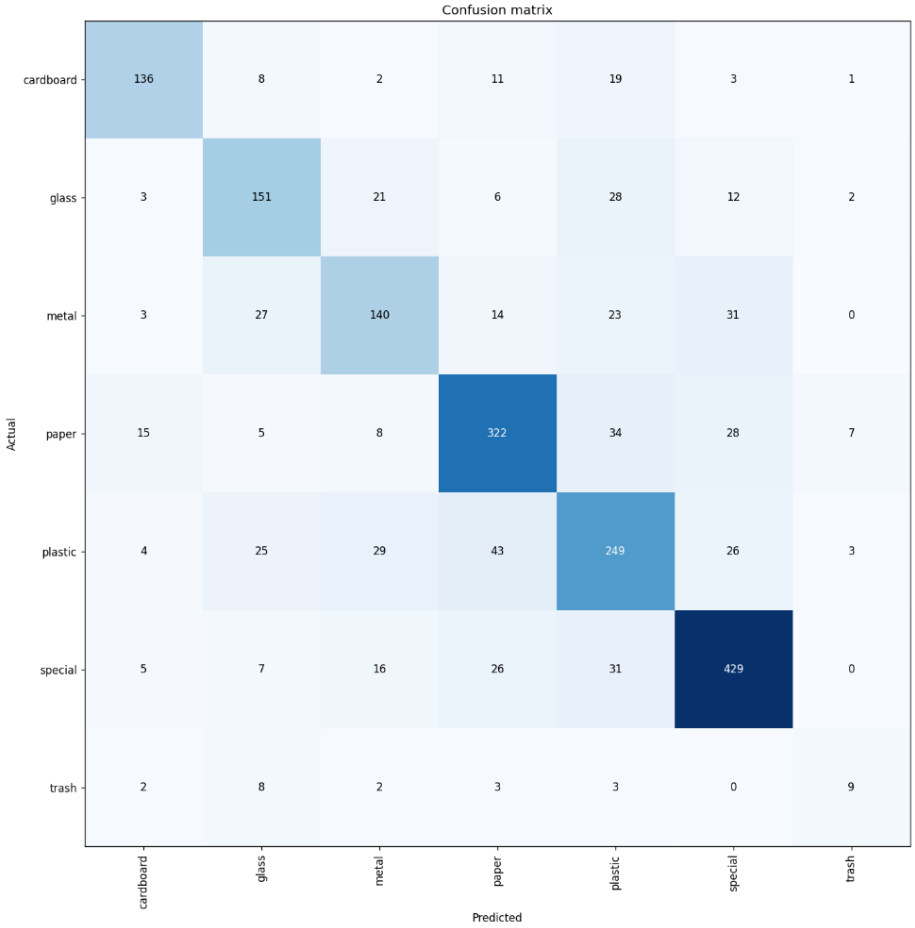
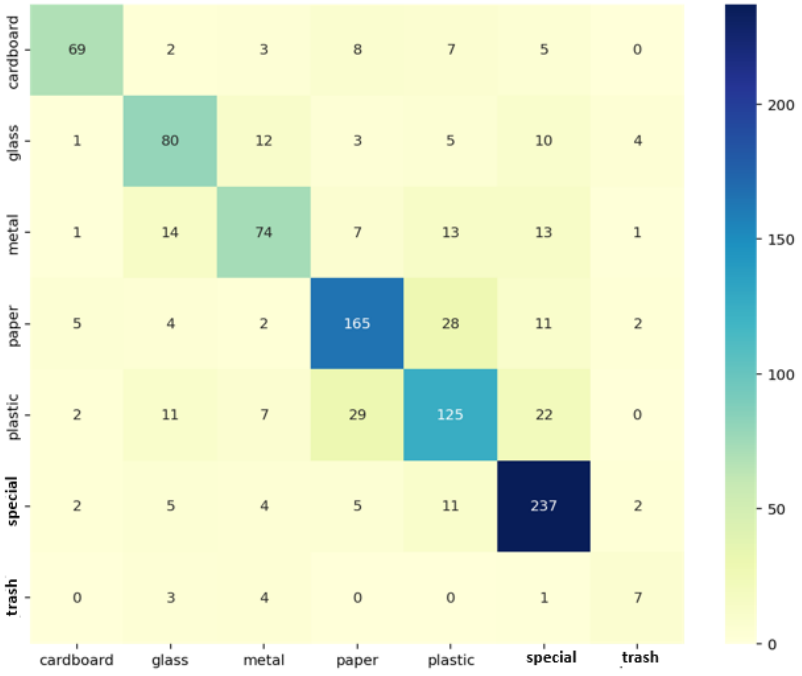


Figura D2: matriz de confusão do teste da rede DenseNet121, para os dados de teste.



APÊNDICE E

Figura E1: matriz de confusão do teste da rede MobileNet_V2, para os dados de validação.

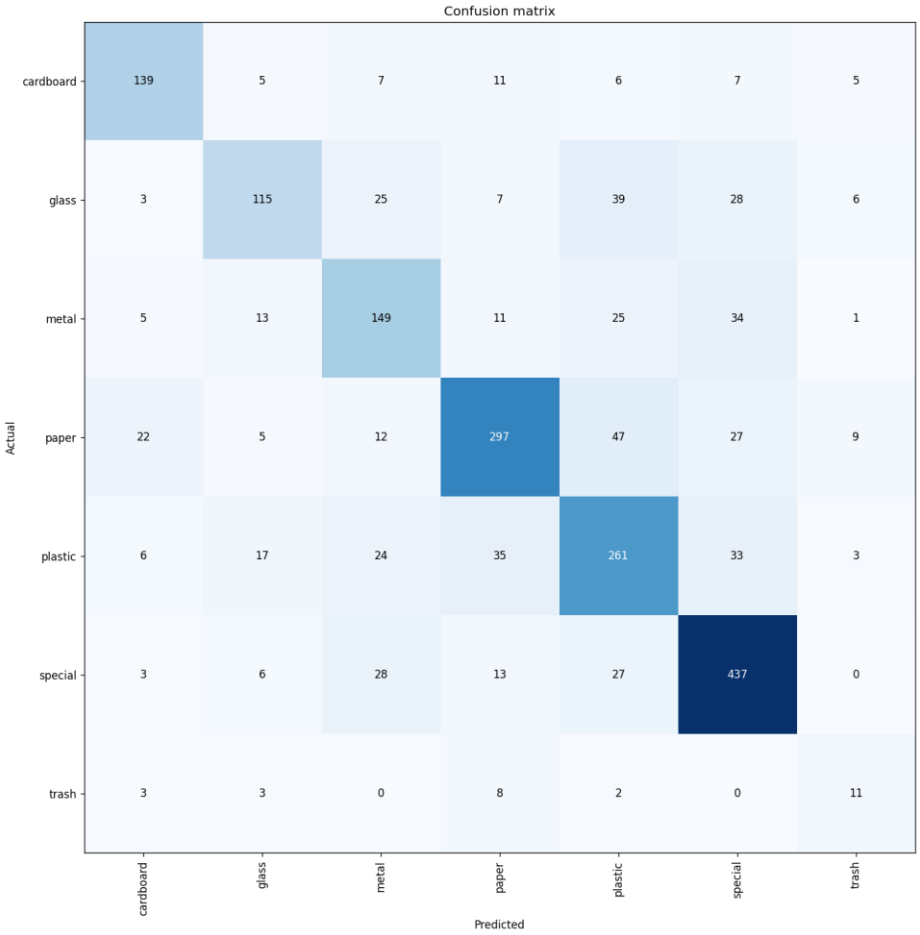


Figura E2: matriz de confusão do teste da rede MobileNet_V2, para os dados de teste.