Um Estudo Sobre Diversos Tipos de Arquiteturas de Redes Neurais Convolucionais para o Reconhecimento de Imagens

Bruna Almeida Osti
Engenharia de Computação
Universidade Tecnológica Federal do Paraná
Cornélio Procópio - PR
brunaosti@alunos.utfpr.edu.br

Abstract

Deep Learning methods are currently state-of-the-art in many possible problems to be solved via machine learning, in particular classification problems. In this article we will introduce theoretical aspects that support the use of deep models and compare the different types of convolutional architectures, in addition to discussing their limitations.

Palavras-chave: deep learning. image processing. computer vision. transfer learnig.

1. Introdução

O aprendizado profundo é uma área que está em constante crescimento, sendo responsável pela automatização da compreensão de dados: imagens, sons e texto. Através desse artifício os computadores tem a capacidade de ver, aprender e reagir de acordo com a situação da mesma forma que os humanos [6].
O aprendizado profundo, do inglês *"Deep Learning"* é baseado em um conjunto de algoritmos que tentam modelar abstrações através de filtros convolucionais que simulam transformações lineares e não lineares. Várias arquiteturas de aprendizagem profunda, tais como redes neurais profundas, redes neurais profundas convolucionais, redes de crenças profundas e redes neurais recorrentes têm sido aplicadas em áreas como visão computacional, reconhecimento automático de fala, processamento de linguagem natural, reconhecimento de áudio e bioinformática, onde elas têm se mostrado capazes de produzir resultados do estado-da-arte em várias tarefas.
Contudo, no campo da visão computacional, a tarefa de classificação de imagens é tida como uma tarefa complexa, pois demanda o aprendizado de modelos a partir de exemplos contidos em bases de dados que forneçam exemplos categóricos, geralmente este processo de aprendizagem consiste na extração de descritores numéricos das imagens, suas características para serem utilizadas para o treinamento de classificadores. Entretanto, como alternativa para suprir a falta de grandes volumes de dados necessário e melhoria do tempo para o treinamento de redes neurais profundas uma solução que se tem mostrado promissora é a técnica de transferência de conhecimento que consiste em utilizar uma rede previamente treinada sobre uma base de dados muito grande para extrair características das imagens a serem classificadas para que essas características possam ser usadas como entrada de treinamento para diferentes classificadores[5].
No presente trabalho buscamos visitar os conceitos abordados em Aprendizado Profundo sob o ponto de vista da Visão Computacional, abrangendo diversos tipos de arquiteturas de Redes Neurais Convolucionais utilizando a técnica de Transferência de Aprendizagem para o reconhecimento de Imagens.

2. Conceitos e Trabalhos Relacionados

O processo de aprendizado de CNNs ainda não é completamente compreendido, do ponto de vista teórico, portanto alguns aspectos devem ser ressaltados à respeito da estrutura das arquiteturas e sobre alguns conceitos para o bom funcionamento dessas técnicas.

2.1. Redes Neurais Convolucionais (CNNs)

As redes neurais convolucionais são compostas basicamente de camadas convolucionais, onde é processado as entradas considerando os campos receptivos locais, como mostra a figura 1, onde primeiramente é feito a extração de características pelas camadas convolucionais. Além disso, inclui operações conhecidas como pooling, sendo responsáveis por reduzir a dimensionalidade espacial das representações. As redes de maior destaque atualmente são as redes residuais (Resnet) e Inception.

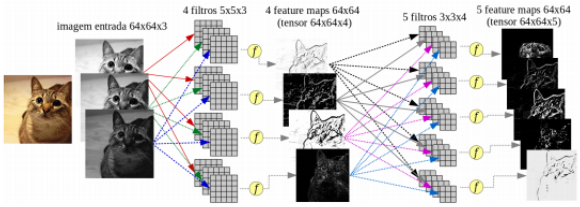


Figura 1: Extração de características pela rede convolucional.[8]

2.1.1 Camada convolucional

Na camada convolucional cada neurônio é um filtro aplicado a uma imagem de entrada e cada filtro é uma matriz de pesos. Cada filtro (neurônio) dessa camada irá processar a imagem e produzir uma transformação dessa imagem por meio de uma combinação linear dos pixels vizinhos.

2.1.2 Feature maps (mapas de características)

Cada representação gerada por um filtro da camada convolucional é conhecida como mapa de características (feature map). Os mapas gerados pelos diversos filtros da camada convolucional são empilhados, formando um tensor cuja profundidade é igual ao número de filtros, que será oferecido como entrada para a próxima camada como mostrado na Figura 2. Note que, como a primeira camada convolucional gera um tensor 64x64x4, os filtros da segunda camada terão que ter profundidade 4, se adicionássemos uma terceira camada convolucional, os filtros teriam que ter profundidade 5.

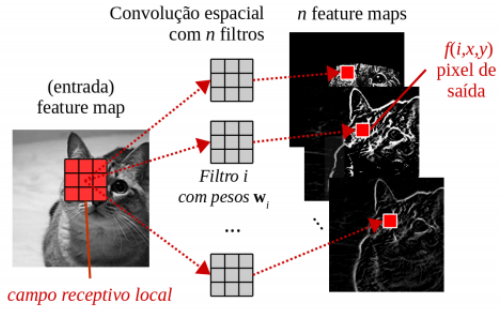


Figura 2: Mapas de Características.[8]

2.1.3 Pooling

A redução da dimensão espacial dos mapas ao longo das camadas da rede é chamada de pooling, tendo como propósito reduzir o custo computacional, mas também obter um tipo de composição de banco de filtros multiresolução que processa imagens em diferentes espaços-escala.

2.1.4 Classificadores

Nesse tipo de camada cada neurônio possui um peso associado a cada elemento do vetor, geralmente são utilizadas para classificação supervisionada, e encontram-se nas redes convolucionais após múltiplas camadas convolucionais, são chamadas de camadas densas(completamente conectadas). As arquiteturas mais recentes utilizam camadas densas ocultas com função de ativação ReLU, e a camada de saída (classificador) com função de ativação softmax.

2.2. Aspectos importantes no treinamento de CNNs

A princípio é preciso entender alguns conceitos que são necessários para o bom funcionamento da rede convolucional.

2.2.1 Data-augmentation

No método utilizado é necessário dispor de um número muitas vezes proibitivo de dados rotulados. Assim, pode-se ser implementado métodos de data augmentation, ou seja, podemos gerar de 5, 10 imagens ou mais por imagem original existente, aplicando-se métodos de processamento de imagens, com processos de como por exemplo: rotacionar a imagem, mudar o zoom, entre outras técnicas, como mostra o exemplo da figura 3.

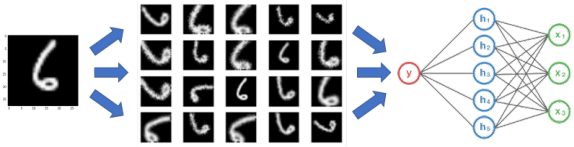


Figura 3: Aplicação de Data Augmentation [1]

2.2.2 Pré-processamento

A base de dados geralmente necessita de alguns ajustes que cumpram as exigências do modelo de entrada, como por exemplo os seguintes itens:

- **Balanceamento da base de dados:** Para um melhor desempenho do algoritmo, é necessário que a base de dados esteja devidamente balanceada, o que pode ser feito através do data-augmentation, já discutido anteriormente.
- **Treinamento e Teste:** O algoritmo é ensinado através de uma quantidade de dados que já contenha o rótulo de classificação, portanto, é necessário dividir a base de dados de modo que seja possível utilizar uma parte

para treinamento e outra para predição(teste), geralmente 80% da base de dados é utilizada para treinamento e 20% para teste.

- **Normalização da Coloração:** Como algumas fotografias podem variar extremamente a iluminação do fundo afetando os valores de intensidade de pixel dentro da imagem e criando uma variação não relacionada aos níveis de classificação, é necessário que através da normalização haja a neutralização das cores.

2.2.3 Transferência de Aprendizado

A técnica de transferência de apredizado, mais conhecida como *transfer learning*, utiliza um modelo pré-treinado por outra tarefa (no caso para o dataset "imagenet") e assim utilizando os pesos do modelo pré-treinado apenas valida os novos dados.
De maneira geral, você deve aplicar transfer learning quando:

- **Os dados das duas tarefas são do mesmo tipo:** a rede original só consegue o que ela foi projetada para fazer, ou seja, ela não vai detectar algo para qual não foi treinada, por exemplo se a rede foi projetada para detectar gatos você não é possível fazer com que ela aprenda a reconhecer quando alguém diz "gato".

- **A tarefa original foi treinada com muito mais dados do que os obtidos:** se a rede original foi treinada para detectar apenas um parametro e é necessário reconhecer vários outros para que a rede não foi treinada, é necessário que haja uma quantidade maior de dados do que anteriormente, portanto a transferência de aprendizado não vai funcionar da melhor forma.

- **O domínio dos dados é parecido:** Se a rede inicialmente foi treinada para detectar carros e pretende-se utilizá-la posteriormente para detecção de pessoas, a transferência de aprendizado não irá funcionar, pois os problemas tem estruturas diferentes.

2.2.4 Modelos pré-treinados: fine-tuning e extração de características

O *Fine-Tuning* é um processo para utilizar um modelo de rede que ja foi treinado para alguma tarefa e para executar uma segunda tarefa semelhante, pois se são tarefas semelhantes o uso de uma rede já treinada permite reutilizar a extração de recursos que acontece nas primeiras camadas da rede sem ser necessário o retreino do modelo.
Geralmente, é substituído a camada de saída, que originalmente é treinada para reconhecer (no caso de modelos "imagenet"ilustrado na figura 4) 1.000 classes, com uma camada que reconhece o número de classes que a segunda tarefa exige. A nova camada anexada ao modelo é retreinada para pegar os recursos e mapeá-los para as classes de saída desejadas.

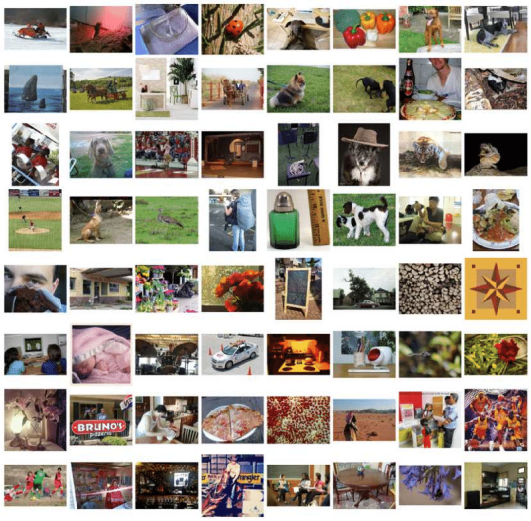


Figura 4: Exemplos no dataset ImageNet [3]

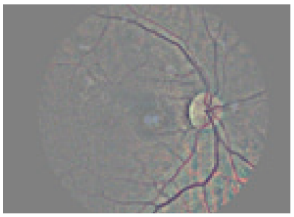
2.3. Trabalhos relacionados

Os sistemas de monitoramento permitem o reconhecimento de padrões preestabelecidos, além de apontar falhas que não seriam identificáveis ao olhar de um funcionário. Do mesmo modo, pode ser aplicado em diferentes contextos, como por exemplo ao controle dos mares como utilizado por [10].
Por outro lado, podem auxiliar a dar diagnósticos mais precisos quando utilizados para doenças específicos, como por exemplo para a retinopatia diabética por [9] que fez o pré-processamento e expansão do *dataset* conforme ilustrado pela figura 5 para diagnósticos mais rápidos e precisos.

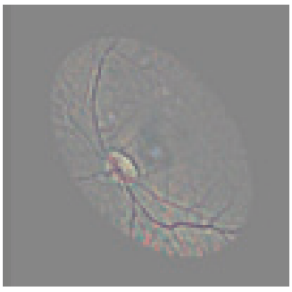
Em relação com a técnica de Transferência de Apredizagem o trabalho [2] apresenta o mesmo princípio de metodologia,

3. Metodologia Proposta

É fato que o método *deep learning* necessita de uma quantidade excessiva de dados, o que torna o problema complexo, considerando o tempo e a dificuldade para rotular, entretanto há táticas para que possamos diminuir o tempo utilizado, por exemplo: utilizando os pesos pré-processados de outras base de dados, como por exemplo o conjunto de dados do *ImageNet*, que possui milhões de imagens pertencentes a diferentes categorias, através da técnica



(a) *Dataset* de retinopatia pré-processado



(b) *Dataset* de retinopatia expandido

Figura 5: *Dataset* de retinopatia, pré-processamento e expansão dos dados [9]

de *Transfer Learning*.
Contudo, se a base de dados utilizada é muito específica para algum problema como no caso da base de dados utilizada *Retinopatia Diabética*, é necessário a manipulação da rede, implicando que as primeiras camadas que classificam os parâmetros gerais permaneçam congeladas e que as últimas camadas que classificam os parâmetros específicos sejam retreinadas a partir da base de dados disposta, através da técnica *Fine-Tuning*.
Desta maneira, utilizamos a técnica de transfer learning para extraímos os hiper-parâmetros da rede convolucional para utilizarmos outros tipos de classificadores, como por exemplo: SVM, KNN, Árvore de Decisão. E assim, comparar a saída desses classificadores com a resposta esperada.
As arquiteturas convolucionais utilizadas para comparação são: *Xception, VGG16, VGG19, ResNet, ResNetV2, ResNeXt, InceptionV3, InceptionResNetV2, MobileNet, MobileNetV2, DenseNet, NASNet.*
Além de tais arquiteturas CNNs pré-treinadas, os classificadores supervisionados para testar as mesmas são: *KNN, SVM, RandomForest, J4.8.*
Cada arquitetura será testada através de todos os classificadores supervisionados e que posteriormente serão avaliados através das métricas: *acurácia, precisão, revocação, matriz de confusão, tempo de treinamento e teste.*

4. Experimentos

Compararemos os resultados obtidos a partir de duas bases de dados distintas, utilizando-se de diferentes arquiteturas convolucionais utilizando-se dos pesos pré-treinados da ”imagenet”, como descrito anteriormente na metodologia, para os diferentes classificadores: KNN, SVM, RandomForest, J4_8.

4.1. Descrição das Bases de Imagens

Utilizou-se para o trabalho duas bases de imagens distintas um para controle de algas marinhas e um para detecção de retinopatia diabética.

4.1.1 Plankton

A base de dados [4] é separado por bases de dados coletadas a cada ano, sendo utilizadas para controle de algas marinhas, como exemplificado na figura 6, originalmente contém 101 Classes, entretanto, utilizamos apenas uma pequena amostra de 4 classes: Chaetoceros, Cylindrotheca, dino30 e Rhizosolenia; contendo 97.339 imagens no total, separadas entre treinamento e teste.

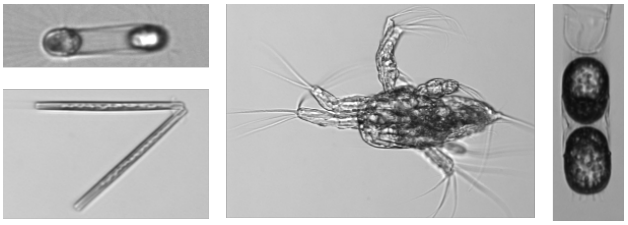


Figura 6: Dataset Plankton

4.1.2 Retinopatia Diabética

A Retinopatia Diabética (DR) é uma complicação do diabetes que pode levar à cegueira, se não for descoberta em tempo hábil, a base de dados [7] como mostrada na figura 7 conta com dois módulos: *DR1 e DR2*, além disso, é separada em 7 classes de imagens: Cotton-wool Spots, Deep Hemorrhages, Drusen, Hard Exudates, Normal Images, Red Lesions, Superficial Hemorrhages. Utilizou-se o dataset DR1, aplicou-se normalização das imagens e também data-augmentation, totalizando 64.672 imagens, divididas entre treinamento e teste.

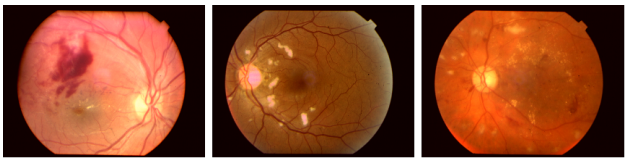


Figura 7: Dataset Retinopatia Diabética

4.2. Cenários

Em relação aos testes, todas as técnicas foram testadas nas seguintes especificações:

Configuração do Computador Utilizado		
Processador	i7 7700hq	3,8 GHz
GPU	GTX 1050ti	2,1 TFLOPS
Memória Ram	16 Gb	2400 MHz

É importante destacar que não houve alteração das arquiteturas e nem dos classificadores para os testes.

4.2.1 Cenário 1

Utilizamos a base de dados de Planktons reduzida, apenas com 4 classes, num total de 97.339 imagens divididos entre treinamento e teste. Os resultados obtidos foram satisfatórios.

4.2.2 Cenário 2

Utilizamos a base de dados de Retinopatia diabética, com 8 classes, primeiramente com a base de dados normal.
Como o resultado obtido não foi satisfatório partimos a aplicação da normalização dessas imagens e o método de data-augmentation, o que nos leva a uma base de dados de 64.672 imagens separadas entre treinamento e teste.
Os resultados obtidos melhoraram, mas ainda continuaram insatisfatórios.

4.3. Resultados e Discussões

Discutiremos os resultados a seguir, de acordo com a base de dados e suas respectivas arquiteturas e de acordo com os cenários previamente estabelecidos.

4.3.1 Resultados - Cenário 1

A arquitetura Xception obteve bom desempenho de acordo a tabela 1, tanto a respeito da acurácia quanto a precisão e revocação.

Classificadores	Acuracia	Precisão	Revocação
KNN	87.96%	87.87%	87.96%
SVM	89.20%	89.37%	89.20%
RandomForest	81.08%	81.93%	81.08%
J4.8	91.06%	91.05%	91.06%

Tabela 1: Arquitetura Xception

A arquitetura VGG16 obteve bom desempenho de acordo com a tabela 2, com alta acurácia, mas também alta precisão e alta revocação.

Classificadores	Acurácia	Precisão	Revocação
KNN	96.08%	96.06%	96.08%
SVM	81.80%	81.92%	81.80%
RandomForest	85.92%	86.41%	85.92%
J4.8	94.42%	94.43%	94.42%

Tabela 2: Arquitetura VGG16

A arquitetura VGG19 obteve bom desempenho de acordo com a tabela 3, com alta acurácia, mas também alta precisão e alta revocação.

Classificadores	Acurácia	Precisão	Revocação
KNN	95.20%	95.20%	95.20%
SVM	74.76%	76.55%	74.76%
RandomForest	83.60%	83.79%	83.60%
J4.8	93.54%	93.55%	93.54%

Tabela 3: Arquitetura VGG19

A arquitetura ResNet obteve bom desempenho de acordo com a tabela 4, com alta acurácia, mas também alta precisão e alta revocação.

Classificadores	Acurácia	Precisão	Revocação
KNN	89.5%	89.61%	89.50%
SVM	51.36%	56.22%	51.36%
RandomForest	80.90%	81.65%	80.90%
J4.8	89.12%	89.10%	89.12%

Tabela 4: Arquitetura Resnet

A arquitetura ResNetV2 obteve bom desempenho de acordo com a tabela 5, com alta acurácia, mas também alta precisão e alta revocação.

Classificadores	Acurácia	Precisão	Revocação
KNN	84.28%	84.07%	84.28%
SVM	86.04%	86.28%	86.04%
RandomForest	71.22%	74.55%	71.22%
J4.8	90.70%	90.66%	90.70%

Tabela 5: Arquitetura ResnetV2

A arquitetura ResNext obteve bom desempenho de acordo com a tabela 6, com alta acurácia, mas também alta precisão e alta revocação.

Classificadores	Acurácia	Precisão	Revocação
KNN	87.56%	87.65%	87.56%
SVM	89.60%	90.05%	89.60%
RandomForest	73.94%	77.98%	73.94%
J4.8	91.62%	91.59%	91.62%

Tabela 6: Arquitetura ResNext

A arquitetura InceptionV3 obteve bom desempenho de acordo com a tabela 7, com alta acurácia, mas também alta precisão e alta revocação.

Classificadores	Acuracia	Precisao	Revocacao
KNN	86.12%	86.16%	86.12%
SVM	89.40%	89.61%	89.40%
RandomForest	80.48%	81.80%	80.48%
J4.8	88.28%	88.23%	88.28%

Tabela 7: Arquitetura InceptionV3

A arquitetura InceptionResNetV2 obteve bom desempenho de acordo com a tabela 8, com alta acurácia, mas também alta precisão e alta revocação.

Classificadores	Acuracia	Precisao	Revocacao
KNN	86.74%	86.77%	86.74%
SVM	84.68%	85.53%	84.68%
RandomForest	78.62%	79.15%	78.62%
J4.8	90.26%	90.26%	90.26%

Tabela 8: Arquitetura InceptionResNetV2

A arquitetura MobileNet obteve bom desempenho de acordo com a tabela 9, com alta acurácia, mas também alta precisão e alta revocação.

Classificadores	Acuracia	Precisao	Revocacao
KNN	89.72%	89.77%	89.72%
SVM	91.90%	91.85%	91.90%
RandomForest	85.20%	85.33%	85.20%
J4.8	94.32%	94.30%	94.32%

Tabela 9: Arquitetura MobileNet

A arquitetura MobileNetV2 obteve bom desempenho de acordo com a tabela 10, com alta acurácia, mas também alta precisão e alta revocação.

Classificadores	Acuracia	Precisao	Revocacao
KNN	90.26%	90.22%	90.26%
SVM	92.56%	92.56%	92.56%
RandomForest	88.20%	88.16%	88.20%
J4.8	93.24%	93.21%	93.24%

Tabela 10: MobileNetV2

A arquitetura DenseNet obteve bom desempenho de acordo com a tabela 11, com alta acurácia, mas também alta precisão e alta revocação.

Classificadores	Acuracia	Precisao	Revocacao
KNN	91.82%	91.82%	91.82%
SVM	92.56%	92.50%	92.54%
RandomForest	88.46%	82.75%	82.46%
J4.8	93.34%	93.02%	93.04%

Tabela 11: Arquitetura DenseNet

A arquitetura NASNet obteve bom desempenho de acordo com a tabela 12, com alta acurácia, mas também alta precisão e alta revocação.

Classificadores	Acuracia	Precisao	Revocacao
KNN	87.20%	87.13%	87.20%
SVM	88.36%	88.39%	88.36%
RandomForest	81.44%	82.38%	81.44%
J4.8	91.64%	91.59%	91.64%

Tabela 12: Arquitetura NASNet

Portanto, analisando-se os resultados, a arquitetura DenseNet obteve resultados superiores das demais arquiteturas, deste modo, podemos observar de acordo com as matrizes de confusão representadas a seguir, que há uma alta concentração de imagens em uma das diagonais da matriz de confusão, representando os acertos de cada classificador.

- KNN:

1068	76	61	45
28	1181	23	18
39	46	1157	8
45	11	9	1185

- SVM:

1061	86	55	48
46	1160	26	18
15	14	1213	8
42	12	3	1193

- RandomForest:

787	180	155	128
93	950	108	99
15	27	1204	4
22	9	37	1182

- J.4_8:

1106	70	28	46
64	1155	20	11
24	21	1194	11
43	5	5	1197

4.3.2 Tempo - Cenário 1

Podemos observar que o classificador que treinou seu modelo mais rápido é o KNN para todas as arquiteturas, tendo em média 1.229 segundos.

Arquiteturas	KNN	SVM	RDF*	J4.8
Xception	1.245	4.451	1.868	41.679
VGG16	1.211	4.660	1.554	43.377
VGG19	1.221	4.597	1.771	41.925
ResNet	1.210	3.999	1.428	49.935
ResNetV2	1.270	4.180	1.682	40.688
ResNeXt	1.229	3.761	1.760	42.696
InceptionV3	1.276	3.469	1.772	37.830
InceptionResNetV2	1.227	3.792	1.576	46.255
MobileNet	1.225	3.166	1.837	44.076
MobileNetV2	1.183	3.363	1.847	43.434
DenseNet	1.237	3.305	1.932	47.492
NASNet	1.214	3.756	1.957	50.661

Tabela 13: Tempo de Treinamento (Segundos)

* RDF = RandomForest

Entretanto, o KNN foi o classificador que mais demorou para testar seu modelo, levando em média 6.336 segundos.

Arquiteturas	KNN	SVM	RDF*	J4.8
Xception	4.990	0.401	0.209	0.004
VGG16	3.703	0.030	0.107	0.003
VGG19	4.133	0.030	0.108	0.004
ResNet	2.571	0.026	0.108	0.005
ResNetV2	9.742	0.030	0.107	0.004
ResNeXt	2.571	0.026	0.108	0.005
InceptionV3	10.127	0.024	0.108	0.004
InceptionResNetV2	8.334	0.024	0.107	0.003
MobileNet	7.647	0.028	0.108	0.004
MobileNetV2	7.265	0.027	0.108	0.005
DenseNet	8.416	0.028	0.107	0.003
NASNet	6.536	0.0288	0.108	0.005

Tabela 14: Tempo de Teste (Segundos)

*RDF = RandomForest

4.3.3 Resultados - Cenário 2

Em primeiro caso, utilizamos a base de dados desbalanceada e contendo poucas imagens, não gerando resultados tão interessantes, portanto a base de dados foi normalizada e realizou-se data-augmentation e gerando-se os resultados apresentados a seguir.
A arquitetura Xception obteve péssimo desempenho de acordo com a tabela 15, com baixa acurácia, mas também baixa precisão e baixa revocação para todos os classificadores.

Classificadores	Acuracia	Precisao	Revocacao
KNN	44.89%	45.41%	44.78%
SVM	33.14%	33.19%	32.73%
RandomForest	29.27%	29.74%	28.74%
J4.8	49.84%	49.55%	49.74%

Tabela 15: Arquitetura Xception

A arquitetura VGG16 obteve péssimo desempenho de acordo com a tabela 16, com acurácia razoável, precisão razoavél e revocação razoável para os classificadores: KNN e J.4_8, os demais apresentaram baixa acurácia, baixa precisão e baixa revocação.

Classificadores	Acuracia	Precisao	Revocacao
KNN	60.58%	61.26%	60.43%
SVM	29.02%	27.29%	28.82%
RandomForest	29.33%	23.32%	29.26%
J4.8	55.91%	56.07%	55.81%

Tabela 16: Arquitetura VGG16

A arquitetura VGG19 obteve péssimo desempenho de acordo com a tabela 17, com acurácia razoável, precisão razoavél e revocação razoável para os classificadores: KNN e J.4_8, os demais apresentaram baixa acurácia, baixa precisão e baixa revocação.

Classificadores	Acuracia	Precisao	Revocacao
KNN	60.60%	61.04%	60.43%
SVM	27.41%	26.05%	27.21%
RandomForest	26.68%	22.59%	26.78%
J4.8	56.52%	56.54%	56.43%

Tabela 17: Arquitetura VGG19

A arquitetura ResNet obteve péssimo desempenho de acordo com a tabela 18, com acurácia razoável, precisão razoavél e revocação razoável para os classificadores: KNN e J.4_8, os demais apresentaram baixa acurácia, baixa precisão e baixa revocação.

Classificadores	Acuracia	Precisao	Revocacao
KNN	59.05%	59.95%	58.92%
SVM	20.89%	19.78%	20.35%
RandomForest	25.84%	28.42%	25.79%
J4.8	55.32%	55.34%	55.24%

Tabela 18: Arquitetura Resnet

A arquitetura ResNetV2 obteve péssimo desempenho de acordo com a tabela 19, apresentaram baixa acurácia, baixa precisão e baixa revocação para todos os classificadores.

Classificadores	Acuracia	Precisao	Revocacao
KNN	44.96%	45.06%	44.87%
SVM	33.17%	33.12%	32.87%
RandomForest	27.02%	25.41%	26.60%
J4.8	47.37%	47.06%	47.30%

Tabela 19: Arquitetura ResNetV2

A arquitetura ResNeXt obteve péssimo desempenho de acordo com a tabela 20, apresentaram baixa acurácia, baixa precisão e baixa revocação para todos os classificadores.

Classificadores	Acuracia	Precisao	Revocacao
KNN	45.95%	46.35%	45.83%
SVM	33.99%	33.24%	33.74%
RandomForest	27.69%	28.18%	27.33%
J4.8	49.80%	49.44%	49.73%

Tabela 20: Arquitetura ResNeXt

A arquitetura InceptionV3 obteve péssimo desempenho de acordo com a tabela 21, apresentaram baixa acurácia, baixa precisão e baixa revocação para todos os classificadores.

Classificadores	Acuracia	Precisao	Revocacao
KNN	44.73%	44.94%	44.61%
SVM	34.12%	33.33%	33.80%
RandomForest	27.99%	27.93%	27.52%
J4.8	48.61%	48.22%	48.53%

Tabela 21: Arquitetura InceptionV3

A arquitetura InceptionResNetV2 obteve péssimo desempenho de acordo com a tabela 22, apresentaram baixa acurácia, baixa precisão e baixa revocação para todos os classificadores.

Classificadores	Acuracia	Precisao	Revocacao
KNN	42.67%	42.97%	42.58%
SVM	31.69%	31.71%	31.31%
RandomForest	29.57%	28.65%	29.12%
J4.8	49.22%	48.88%	49.12%

Tabela 22: Arquitetura InceptionResNetV2

A arquitetura MobileNet obteve péssimo desempenho de acordo com a tabela 23, apresentaram baixa acurácia, baixa precisão e baixa revocação para todos os classificadores.

Classificadores	Acuracia	Precisao	Revocacao
KNN	45.52%	46.02%	45.44%
SVM	33.45%	32.78%	33.29%
RandomForest	29.27%	29.25%	28.92%
J4.8	51.10%	50.97%	51.03%

Tabela 23: Arquitetura MobileNet

A arquitetura MobileNetV2 obteve péssimo desempenho de acordo com a tabela 24, apresentaram baixa acurácia, baixa precisão e baixa revocação para todos os classificadores.

Classificadores	Acuracia	Precisao	Revocacao
KNN	49.94%	50.66%	49.83%
SVM	36.29%	35.57%	35.95%
RandomForest	29.90%	28.30%	29.51%
J4.8	50.87%	50.81%	50.78%

Tabela 24: Arquitetura MobileNetV2

A arquitetura DenseNet obteve péssimo desempenho de acordo com a tabela 25, apresentaram baixa acurácia, baixa precisão e baixa revocação para todos os classificadores.

Classificadores	Acuracia	Precisao	Revocacao
KNN	49.72%	50.47%	49.66%
SVM	34.99%	33.42%	34.65%
RandomForest	30.19%	29.90%	29.86%
J4.8	51.17%	51.01%	51.09%

Tabela 25: Arquitetura DenseNet

A arquitetura NASNet obteve péssimo desempenho de acordo com a tabela 26, apresentaram baixa acurácia, baixa precisão e baixa revocação para todos os classificadores.

Classificadores	Acuracia	Precisao	Revocacao
KNN	42.88%	43.19%	42.76%
SVM	31.59%	32.42%	31.31%
RandomForest	25.58%	25.51%	25.02%
J4.8	46.88%	46.63%	46.80%

Tabela 26: Arquitetura NASNet

Portanto, analisando-se os resultados arquitetura VGG19 obteve resultados superiores das demais arquiteturas, deste modo, podemos observar de acordo com as matrizes de confusão representadas a seguir, que há uma dispersão das diagonais da matriz de confusão, o que representa que o modelo está encontrando dificuldades na classificação.

- KNN:

1355	36	29	61	31	56	51
24	655	36	187	7	393	328
17	26	1405	45	65	40	40
68	230	53	834	39	330	109
53	29	125	65	1428	49	31
39	359	43	320	19	587	289
38	322	23	137	4	326	771

- SVM:

844	155	7	12	226	294	81
429	480	49	35	281	71	285
357	272	67	8	430	387	117
310	384	32	24	274	431	208
363	124	17	3	877	352	44
305	316	30	18	243	576	168
502	418	41	25	273	48	314

- RandomForest:

715	383	0	0	16	293	212
210	956	2	0	36	0	426
192	812	0	0	202	252	180
154	747	1	0	120	321	320
232	694	0	0	485	172	197
149	632	0	0	131	462	282
256	871	0	0	15	0	479

- J.4_8:

1241	53	42	82	48	75	78
37	705	37	145	21	241	444
57	48	1238	71	123	54	47
69	257	82	708	85	282	180
83	45	168	105	1280	72	27
49	408	58	275	69	461	336
44	283	41	112	30	183	928

4.3.4 Tempo - Cenário 2

Podemos verificar que o classificador que responsável por treinar o modelo mais rápido é o RandomForest para todas as arquiteturas exceto para a ResNeXt, tendo o tempo de treinamento em média 4.920 segundos.

Arquiteturas	KNN	SVM	RDF*	J4.8
Xception	3.586	22.816	3.488	56.890
VGG16	4.135	17.384	3.382	60.255
VGG19	3.653	17.233	3.283	61.786
ResNet	3.609	16.195	3.052	53.779
ResNetV2	4.122	21.805	3.255	51.476
ResNeXt	3.706	21.736	21.736	52.338
InceptionV3	3.695	21.324	3.518	57.478
InceptionResNetV2	3.718	22.381	3.347	55.779
MobileNet	3.820	22.458	3.212	53.799
MobileNetV2	3.692	19.852	3.751	62.588
DenseNet	3.702	20.209	3.362	54.118
NASNet	3.727	22.225	3.661	57.819

Tabela 27: Tempo de Treino (Segundos)

*RDF = RandomForest

Entretando, o classificador que testou seu modelo mais rápido é o J.4_8 para todas as arquiteturas, tendo em média 0.010 segundos.

Arquiteturas	KNN	SVM	RDF*	J4.8
Xception	8.978	0.069	0.109	0.013
VGG16	3.163	0.060	0.113	0.015
VGG19	3.325	0.061	0.111	0.012
ResNet	3.488	0.062	0.111	0.013
ResNetV2	8.234	0.066	0.113	0.011
ResNeXt	10.031	0.064	0.110	0.014
InceptionV3	11.519	0.060	0.111	0.012
InceptionResNetV2	4.390	0.056	0.112	0.013
MobileNet	8.956	0.057	0.211	0.013
MobileNetV2	9.894	0.060	0.213	0.011
DenseNet	10.436	0.060	0.213	0.011
NASNet	10.489	0.059	0.213	0.014

Tabela 28: Tempo de Teste (Segundos)

*RDF = RandomForest

5. Conclusões

Pode-se observar que a técnica de Deep Learning é um processo muito custoso computacionalmente, mas apesar disso, é uma técnica que tem uma capacidade de abstração muito alta, não sendo necessário a utilização de um especialista para tal feito. Portanto, a utilização do método de transferência de aprendizagem reduziu o tempo consideravelmente necessário para a extração e classificação dos parâmetros, apesar de que no caso de uma base de dados mais específica como o da retinopatia diabética, o modelo teve dificuldade em classificar, visto que a "imagenet"não tenha imagens parecidas com a retina.
Uma hipótese seria que pela complexidade do problema e devido a poucas amostras a base de dados de retinopatia acaba sendo mais difícil de classificar.

Referências

[1] Z. H. J. D. C. R. Alex Ratner, Henry Ehrenberg. Learning to compose domain-specific transformations for data augmentation, 2019.

[2] G. Carneiro, J. Nascimento, and A. P. Bradley. Unregistered multiview mammogram analysis with pre-trained deep learning models. In *Lecture Notes in Computer Science*, pages 652–660. Springer International Publishing, 2015.

[3] C. Chen, Y. Ren, and C.-C. J. Kuo. Global-attributes assisted outdoor scene geometric labeling. In *SpringerBriefs in Electrical and Computer Engineering*, pages 93–120. Springer Singapore, 2016.

[4] E. F. B. Heidi M. Sosik, Emily E. Peacock. Annotated plankton images - data set for developing and evaluating classification methods.

[5] D. López-Sánchez, A. G. Arrieta, and J. M. Corchado. Deep neural networks and transfer learning applied to multimedia web mining. In *Distributed Computing and Artificial Intelligence, 14th International Conference*, pages 124–131. Springer International Publishing, June 2017.

[6] NVIDIA. O poder do aprendizado profundo, 2019.

[7] R. Pires, H. F. Jelinek, J. Wainer, E. Valle, and A. Rocha. Advancing Bag-of-Visual-Words Representations for Lesion Classification in Retinal Images. 10 2016.

[8] M. A. Ponti and G. B. P. da Costa. Como funciona o deep learning. *CoRR*, abs/1806.07908, 2018.

[9] H. Pratt, F. Coenen, D. M. Broadbent, S. P. Harding, and Y. Zheng. Convolutional neural networks for diabetic retinopathy. *Procedia Computer Science*, 90:200–205, 2016.

[10] O. Py, H. Hong, and S. Zhongzhi. Plankton classification with deep convolutional neural networks. In *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*. IEEE, May 2016.