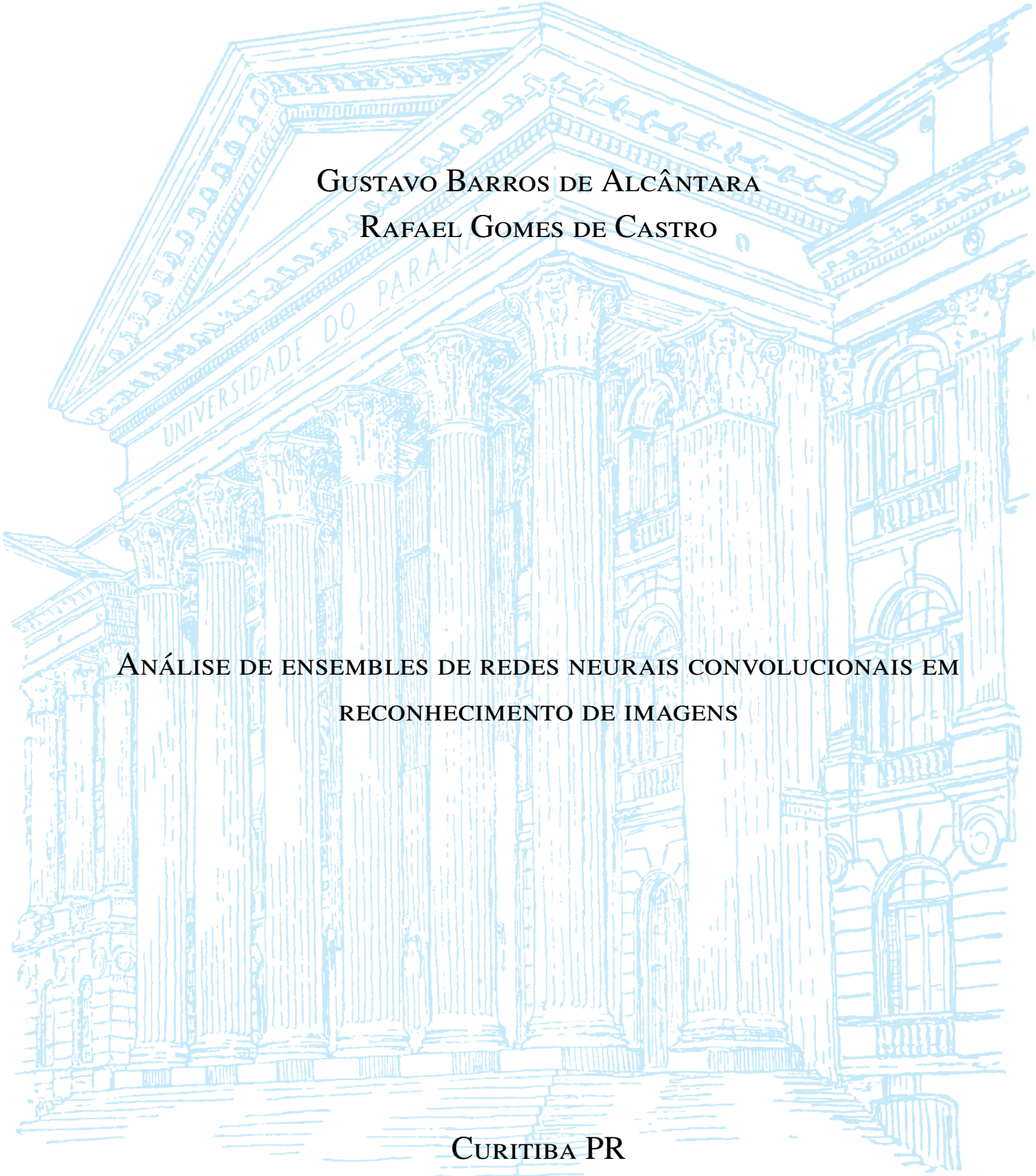
Universidade Federal do Paraná

Gustavo Barros de Alcântara
Rafael Gomes de Castro



Análise de ensembles de redes neurais convolucionais em reconhecimento de imagens

Curitiba PR
2017

Gustavo Barros de Alcântara
Rafael Gomes de Castro

Análise de ensembles de redes neurais convolucionais em reconhecimento de imagens

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Eduardo Jaques Spinosa.

Curitiba PR
2017

Resumo

Redes neurais convolucionais são um tipo de rede neural amplamente utilizada no reconhecimento de imagens e têm como característica a habilidade de extrair mapas de características (features maps) das imagens através de filtros (kernels) que aprendem durante o processo de treinamento da rede. Este trabalho propõe utilizar um conjunto dessas redes, sendo distintas entre si, para tentar obter taxas de acertos maiores em problemas desafiadores. Para isso, utilizou-se a biblioteca de inteligência de máquina TensorFlow, desenvolvida pelo Google. Algumas técnicas diferentes para computar a classificação final do ensemble foram utilizadas, entre elas o método da votação simples, das médias e Std-Médias com Fibonacci, sendo esse último um método experimental criado pela equipe.

Palavras-chave: Redes Neurais Convolucionais, Ensembles, Aprendizado Profundo.

Abstract

Convolutional Neural Networks are a specific type of Neural Network widely used in image recognition that have a powerful hability to extract feature maps from images using filters, or kernels, that adapt themselves through learning on the network trainning process. This work propose the use of an ensemble of these networks, all distinct from each other, in an attempt to achieve better classification accuracy in a challenging problems. The implementation used the machine intelligence library TensorFlow, created by Google. Some different techniques were used to compute the ensemble's final classification, including simple voting, average and Std-Average with Fibonacci, an experimental method developed by the authors.

Keywords: Convolutional Neural Networks, Ensembles, Deep Learning.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A área de Inteligência Artificial nos últimos anos tem ganho notoriedade na comunidade científica e nos meios comerciais, principalmente a subárea de aprendizado profundo (Deep Learning). Deep Learning é a subárea do Aprendizado de Máquina que agrega redes neurais com mais de duas camadas, permitindo que características de alto nível sejam automaticamente sintetizadas a partir de características de baixo nível. Com contribuições relevantes em trabalhos envolvendo classificação, agrupamento, previsão, reconhecimento, etc., métodos de Deep Learning tem revolucionado muitas áreas da ciência, como medicina e visão computacional [5].

Tão logo as imagens de exames médicos puderam ser digitalizadas para um computador, sistemas de análise automáticas começaram a surgir [22]. Nos primórdios da década de 1970, esta análise consistia de aplicações sequenciais de filtros para extração de características de baixo nível, como linhas e pontos. Eram comuns códigos com recorrentes ocorrências de desvios condicionais (if-then-else) em algoritmos de análise médica, pois os algoritmos mais sofisticados de Inteligência Artificial da época eram baseados em regras. Ao final dos anos 1990, métodos supervisionados de análise médica tornaram-se extremamente populares e, ainda hoje, os mais sofisticados softwares destinados ao diagnóstico de patologias ainda utilizam em seu cerne o treinamento a partir de um conjunto de dados, bem como a extração de características de alto nível e um classificador estatístico de imagens.

Com os avanços na fabricação e comercialização de sensores 3D, dispositivos que compõem este grupo, como o LIDAR, UAV e até Microsoft Kinect (que possui um custo mais acessível), a captura de imagens RGB-D (três canais para representação de cores e um para profundidade) tornou-se mais viável e acessível [13]. Com esta abundância de imagens tridimensionais, pesquisas explorando métodos de resolução de problemas comuns na área de visão computacional ganharam um novo dataset para beneficiar-se. Entre tais problemas, encontram-se detecção e classificação de objetos 3D e recuperação de formas 3D. Semelhantemente à área médica, a evolução de algoritmos de Deep Learning contribuiu de forma significativa também para a visão computacional. Até recentemente, o processo de manipulação de imagens 3D envolvia a obtenção de descritores locais ou globais de forma manual. A partir de 2012, com os resultados obtidos por Krizhevsky et al. [2012] no 2012 ILSVRC, a utilização de técnicas de Deep Learning em imagens tridimensionais passou a ser uma constante nos trabalhos relacionados à detecção e reconhecimento de imagens.

Com os resultados dos trabalhos envolvendo Deep Learning percebe-se, portanto, que os computadores podem determinar satisfatoriamente o limite ótimo de decisão para diferenciar uma imagem de outra utilizando características de alto nível, embora ainda necessitem da interação humana para determinar as principais características a serem extraídas das imagens. Arquiteturas que visam automatizar este processo entraram em voga na literatura recentemente. Um dos

exemplos mais famosos e bem sucedidos, e que foi utilizado por Krizhevsky et al. [2012], são as Redes Neurais Convolucionais.

Este trabalho concentra-se no uso dessas estruturas para reconhecimento de imagens, e adiciona o conceito de ensembles para uma melhor resolução desse problema. Através desse conceito demonstramos a possibilidade de melhorar os resultados de tais redes num conjunto de dados frequentemente utilizado na literatura. Outros detalhes de implementação e variação de parâmetros também são discutidos. Com ensembles contendo 17 CNNs, uma análise detalhada é realizada neste trabalho, avaliando taxas de acerto individuais de cada uma das CNNs presentes no ensemble e comparando-as com o resultado geral obtido pelo ensemble. Outra contribuição deste trabalho são os métodos do ensemble, onde são realizados diferentes formas de analisar os resultados individuais de cada CNN e juntá-los em um resultado geral. Entre estes métodos, encontram-se a Votação Simples e Médias. Por fim, um terceiro método é proposto, com o funcionamento baseado na Proporção Áurea.

Este trabalho foi dividido em 6 capítulos. O Capítulo 1 trata da visão geral das redes neurais convolucionais e qual o seu papel no contexto atual do Aprendizado de Máquina e sua importância nesse cenário atual, assim como uma retrospectiva histórica. No capítulo 2 foram descritas as principais características das CNNs e as camadas que as compõem e quais suas funções no reconhecimento de imagem. Foi descrito também detalhes da implementação, técnicas utilizadas e arquiteturas definidas, além de outros assuntos relativos à fundamentação teórica. No capítulo 3, todas as etapas do desenvolvimento foram descritas desde a concepção da ideia do trabalho e o ambiente onde os testes foram feitos. Ainda nesse capítulo é comentado sobre alguns trabalhos relacionados ao tema e as diversas áreas em que pode ser aplicado e as várias formas de resolver o mesmo problema de maneiras ligeiramente diferentes. No capítulo 4 os resultados são apresentados, as taxas de acerto individuais de cada CNN e a taxa de acerto do ensemble formado por elas. As taxas de acerto de cada técnica utilizada também foram apresentadas. O capítulo 5 foi reservado para a conclusão do trabalho e consolidação dos resultados e os impactos que podem ter em aplicações diversas. O capítulo 6 traz a proposta de trabalhos futuros que serão realizados pela equipe e sugestões de caminhos de pesquisa que talvez levem a resultados interessantes.

Capítulo 2

Fundamentação Teórica

2.1 Redes Neurais Convolucionais

Redes Neurais Convolucionais (Convolutional Neural Networks, CNN) ganharam enorme destaque entre os pesquisadores da área de Aprendizado de Máquina nos últimos anos, mais especificamente na subárea de Aprendizado Profunda (Deep Learning). Em virtude do poder de processamento alcançado pelos computadores modernos, bem como a disponibilidade de bases de dados com grande quantidade de elementos, é possível construir arquiteturas de redes neurais com alto desempenho no reconhecimento e classificação de imagens. Embora tenha demonstrado resultados surpreendentes apenas recentemente, o conceito de CNN possui pelo menos 20 anos de existência. Inspiradas no modelo biológico de percepção da visão dos seres vivos, LeCun et al. publicou em 1990 o que ficou estabelecido como o framework moderno das redes neurais convolucionais [9,21].

Desde que entraram em voga, redes neurais convolucionais possibilitaram contribuições nas mais diversas áreas, sendo aplicadas em trabalhos que envolvem desde reconhecimento de digitais até detecção de câncer. Utilizando imagens hiperespectrais, a CNN criada em [4] foi desenvolvida com o intuito de auxiliar na detecção de tumores cancerígenos na região encefálica e no pescoço. Não se limitando apenas ao reconhecimento de imagens, é possível encontrar aplicações de CNNs em diversas outras tarefas. Como observado em [3], os autores do artigo propuseram uma solução para reconhecimento de texto e fala utilizando CNN. Após uma investigação inicial de diversas arquiteturas de redes neurais convolucionais, os autores criaram uma nova camada de pool utilizando a função de softmax ponderada. Com isto, o tamanho da camada de pool pode ser aprendido automaticamente.

Uma Rede Neural Convolucional pode ser definida como uma rede neural com uma estrutura especial. Similarmente a uma Rede Neural Artifical (RNA) convencional, uma Rede Neural Convolucional consiste de uma rede de unidades de processamentos, conhecidas como neurônios, conectadas por pesos ajustáveis. O papel de cada neurônio na rede é processar os valores de entrada, realizados através de uma função de soma, e gerar um valor de saída, realizado através de uma função ativação. Para ilustrar, a figura 2.1 demonstra a estrutura de um neurônio. Conforme representado, para cada um dos **m** valores de entrada, há um peso para ponderá-los. A função de soma, então, realiza a ponderação e a soma dos valores de entrada, juntamente com um valor de limiar (bias) responsável por atribuir maior flexibilidade à rede. A função de ativação, por fim, gera o valor de saída.

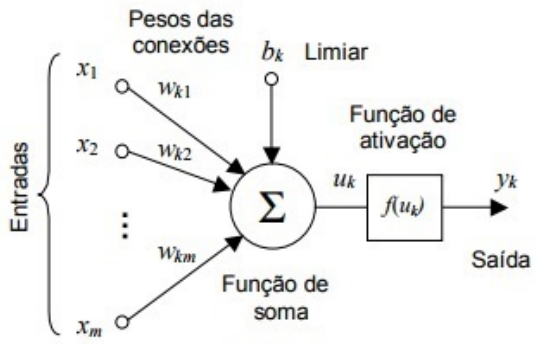


Figura 2.1: Representação de um neurônio, estrutura básica de uma rede neural.

Segmentada em camadas, a primeira camada consiste, geralmente, mas não obrigatoriamente, de uma imagem com largura, altura e profundidade (RGB). A segunda camada desta estrutura, que dá o nome à rede neural, é a camada convolucional. Esta parte da estrutura representa o conjunto de mapas de características e da imagem original, após a aplicação de filtros. Comumente, emprega-se uma função de ativação nas matrizes resultantes da convolução, sendo a mais utilizada a função ReLU (Rectified Linear Unit), permitindo que a rede compute operações não lineares, cuja operação matemática resume-se simplesmente à função max(0,x). A função ReLU é responsável também por acelerar o processo de classificação. Em seguida, também de modo semelhante à rede neural tradicional, a última camada das CNNs é a totalmente conectada, que processa os dados e define a classificação final da rede. Todas estas camadas, colocadas em sequência, definem a arquitetura de uma Rede Neural Convolucional. É importante notar que todas as CNNs presentes na literatura variam parâmetros, como número de camadas, valores de bias e pesos, números de neurônios, stride, padding, entre outros inúmeros parâmetros explicitados neste artigo, mas todas estas CNNs respeitam a organização e estrutura definida originalmente pelo [9]. Na figura 2.2, temos um exemplo da visão geral de uma CNN. As principais camadas, bem como as principais funções de cada uma delas são descritas a seguir.

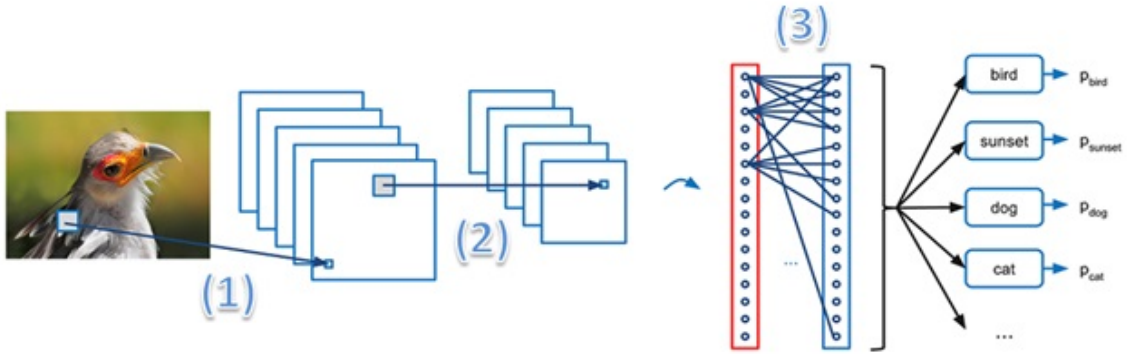


Figura 2.2: Ilustração representando a arquitetura de uma CNN. Em (1), temos a Camada Convolutiva, responsável pela extração de características. A camada de Pooling é representada em (2), realizando a minimização da matriz de convolução. No final da arquitetura temos a Camada Totalmente Conectada (3), finalizando o processo de classificação. Imagem adaptada de https://adeshpande3.github.io.

2.2 Camada Convolucional

A primeira camada contendo operações matemáticas da rede é a camada convolucional, ou camada convolutiva. É camada da arquitetura que demanda a maior parte de processamento de todo o procedimento de classificação. Este passo da classificação consiste de um conjunto de kernels (filtros) que são adaptados ao longo da execução, processo este denominado de aprendizado da rede neural convolucional. Estes kernels são matrizes com dimensões pequenas (5 pixels de largura, 5 pixels de altura e 3 pixels de profundidade, por exemplo) que armazenam valores reais, ou pesos, conforme a nomenclatura empregada para redes neurais. A partir do processo de convolução entre os dados de entrada e os filtros da camada, é obtido um mapa de características, que sinaliza as regiões específicas da entrada onde foram encontradas características específicas em relação ao filtro. Como citado acima, os pesos presentes nos filtros, ao longo do treinamento, são alterados para outros valores reais, fazendo com que a rede aprenda a identificar regiões significantes nos dados de entrada e a extrair características do conjunto de dados.

Para exemplificar o processo descrito acima, consideremos a figura 2.3, demonstrando a representação matricial de um filtro de detecção de curvas a ser aplicado sobre uma imagem. Neste exemplo, cada pixel do filtro possui uma valoração em sua representação matricial. Para pixels pertencentes à característica desejada, no caso uma curva, atribui-se o valor positivo. Para os demais pixels, o valor atribuído a eles será -1. A imagem a ser classificada é selecionada em partes, como representa a figura 2.4. Semelhantemente ao filtro, para cada parte da imagem, será utilizada a sua representação em matriz. O processo da camada convolucional, então possui todos os elementos necessários para sua execução. Conforme demonstrado na figura 2.5, a aplicação do filtro consiste de uma multiplicação entre a representação matricial do filtro e a representação matricial da parte da imagem selecionada. Conforme o resultado dessa multiplicação, conseguimos identificar a presença ou ausência da característica desejada, ou seja, detectar a presença ou não de uma curva na imagem. Caso a operação de multiplicação resulte em um valor grande, a camada convolutiva assume que há a presença de tal característica na imagem em questão. Por fim, a figura 2.6 demonstra um processo convolutivo em uma parte da imagem original onde não há a presença da característica de interesse.

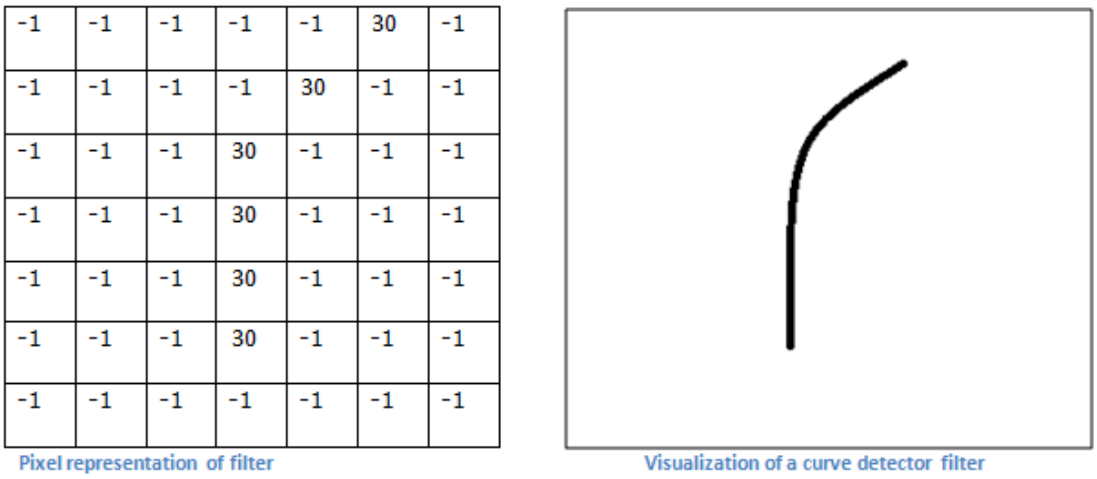


Figura 2.3: Representação matricial de um filtro. Imagem adaptada de https://adeshpande3.github.io.

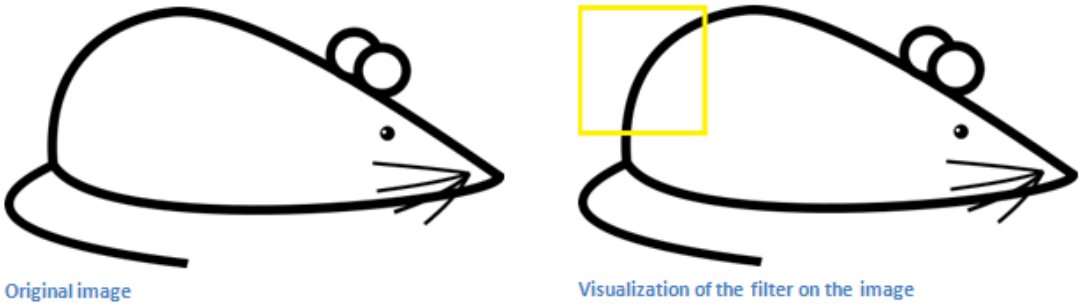


Figura 2.4: Seleção da área da imagem a ser aplicado o filtro. Imagem adaptada de https://adeshpande3.github.io.

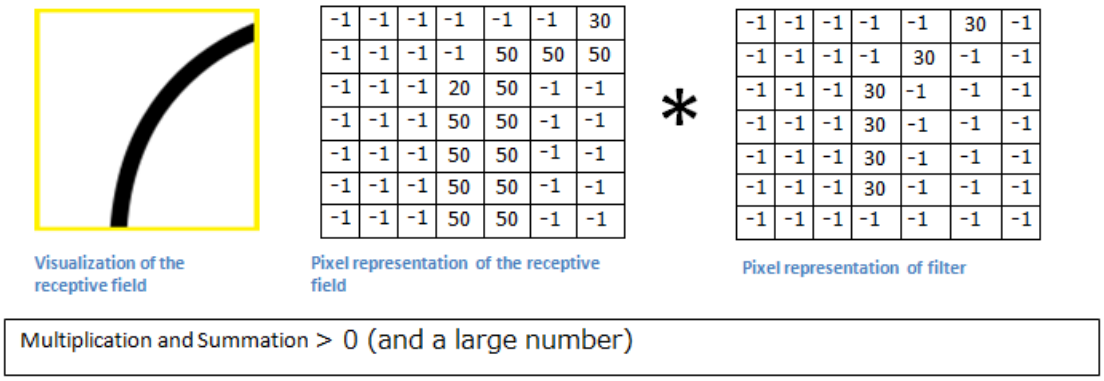


Figura 2.5: Representação do processo de convolução onde há presença da característica desejada. Imagem adaptada de https://adeshpande3.github.io.

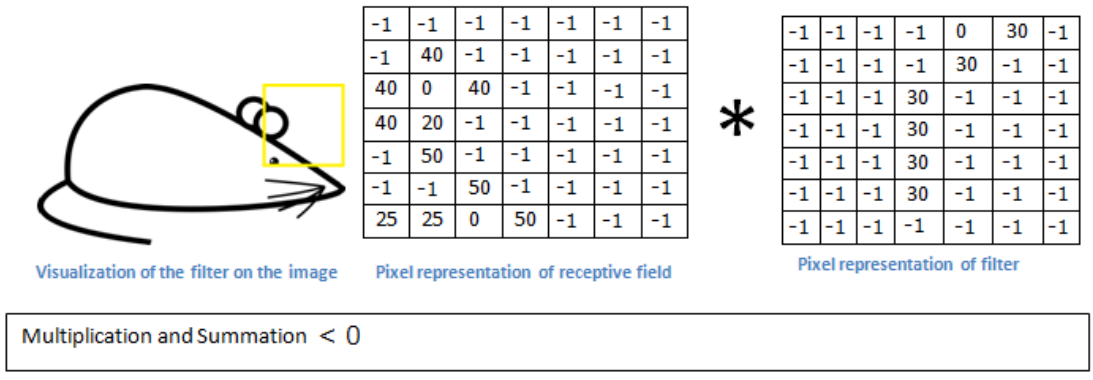


Figura 2.6: Representação do processo de convolução onde não há presença da característica desejada. Imagem adaptada de https://adeshpande3.github.io.

2.3 Camada Pooling

Por envolver milhares de repetições de operações matemáticas em matrizes, qualquer redução no tamanho destas matrizes ocasionará melhora no desempenho computacional e,

consequentemente, deixará o processo de classificação ao qual as CNNs se propõem mais eficiente. Com este intuito, a camada pooling tem fundamental importância. Mesmo não sendo uma camada essencial para o processo de classificação, é empregada em todas as redes neurais utilizadas neste trabalho, bem como em todas as redes neurais presentes nos trabalhos citados na revisão bibliográfica. A técnica de pooling é empregada com o objetivo de reduzir a dimensão das matrizes resultantes da convolução, tendo como consequência a redução da quantidade de parâmetros a serem aprendidos pela rede (composto, em grande maioria, pelos pesos dos kernels.) Além do ganho computacional, esta técnica contribui também para a mitigação do efeito de overfitting. Esse efeito ocorre quando a rede neural se torna extremamente especializada no conjunto de dados que usa durante o treinamento, e como consequência a rede perde a capacidade de classificar imagens que sejam diferentes das contidas no conjunto de imagens de treinamento, ou seja perde a capacidade de generalizar.

A técnica de pooling consiste em deslizar uma janela com dimensões predefinidas, por exemplo 2x2, pela matriz obtida pelo processo de convolução. Como as implementações de redes neurais em geral tendem a maximizar os resultados, é feita a seleção do maior valor dentro dos valores selecionados pela janela. A figura 2.7 demonstra este processo. Assim como as dimensões da janela, o valor de stride da camada de pooling também deve ser estabelecido previamente. Esta constante determina o deslizamento da janela dentro da matriz. É importante observar que, dependendo da dimensão da janela e da dimensão da matriz, pode ocorrer um extrapolamento da janela. Para evitar cálculos errôneos ocasionados por este extrapolamento, o padding da camada de pooling determina os valores a serem inseridos nas bordas da matriz. Na figura 2.7, as bordas da matriz receberam valor 0.0.

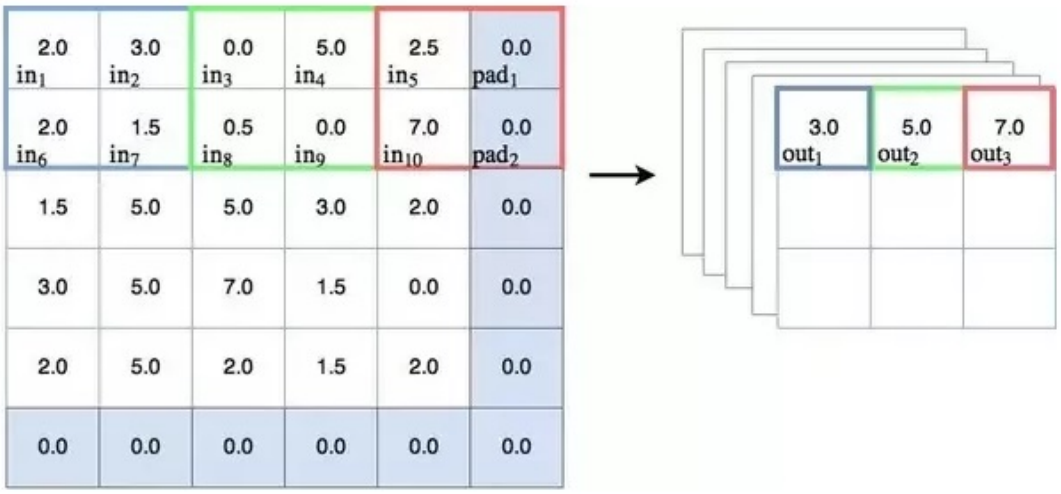


Figura 2.7: Representação da camada de pooling. Os quadrados em azul, vermelho e verde representam a janela em diferentes momentos durante o percorrimento da matriz resultante da convolução. O padding também está representado pelas posições da matriz com coloração mais escura. Imagem obtida de http://adventuresinmachinelearning.com.

Para que a camada pooling produza bons resultados no treinamento, é necessário que alguns parâmetros presentes nesta camada estejam devidamente configurados. Efeitos indesejados como baixa taxa de acerto podem ocorrer devido a uma configuração errônea de stride e padding.

Por fim, é importante observar que a camada de pooling não reduz a quantidade de canais da imagem, mas sim a quantidade de elementos em cada canal. Outra observação coerente

é que os pixels vizinhos de uma mesma imagem tendem a ser correlacionados. Esta propriedade garante o bom funcionamento da técnica de pooling.

2.4 Camada Totalmente Conectada

Tipicamente sendo a última camada da CNN, possui o funcionamento equivalente a uma RNA convencional. Assim sendo, as mesmas técnicas empregadas para melhorar o desempenho de uma RNA podem ser aplicadas com sucesso nas CNNs. Como o próprio nome sugere, esta camada está totalmente conectada com a camada imediatamente anterior da rede, tendo como entrada, então, todos os mapas de características resultantes da camada convolutiva e transformados pela camada de pooling, se esta última for empregada na implementação da rede. Por fim, os neurônios desta camada estão totalmente conectados com os neurônios de saída da rede neural.

2.5 Descrição do Problema

O problema de classificação de imagens está na base de muitas aplicações ambientais e socioeconômicas [23]. Muitas técnicas têm sido buscadas pela comunidade acadêmica para melhorar cada vez mais a taxa de acerto de classificadores já existentes e até mesmo novas abordagens têm surgido.

Entre muitos classificadores existentes, Support Vector Machine (SVM) é amplamente empregado por pesquisadores por ser uma clássica técnica de aprendizado de máquina que pode auxiliar na classificação de problemas envolvendo grande quantidade de dados [28]. Os algoritmos de Support Vector machines buscam a separação ótima entre classes através da minimização dos erros, criando, então um limite de decisão [16]. O funcionamento de SVMs baseia-se na análise de dados e reconhecimento de padrões. Para isto, um conjunto de dados de entrada é utilizado, e para cada um dos dados, uma predição é retornada. À priori, é necessária uma base de dados para treinamento, já que SVMs pertencem à classe de algoritmos de aprendizado supervisionado (assim como as RNAs e CNNs). Com base no treinamento, os dados a serem classificados são inferidos para cada uma das possíveis classes. Assim como o SVM, existem muitos outros classificadores que se propõem a resolver o problema de classificação. Neste trabalho optamos por usar CNNs como o classificador principal.

Para o reconhecimento de imagens, utilizamos o dataset CIFAR-10, criado por Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. O dataset contém 60.000 imagens pequenas, coloridas, com dimensões de 32x32x3 pixels (RGB), sendo que 50.000 dessas imagens foram usadas como conjunto de treinamento e as 10.000 restantes como conjunto de teste. O dataset contém as 10 classes a seguir rotuladas de 0 a 9: [avião, automóvel, pássaro, gato, cervo, cão, sapo, cavalo, navio, caminhão]. A figura 2.8 ilustra um exemplo da classe pássaro pertencente ao dataset CIFAR-10.

Diversas publicações envolvendo algoritmos de classificação foram desenvolvidas utilizando o CIFAR-10. Entre as que possuíram melhores resultados, encontra-se [8]. Com uma implementação diferente da camada de pooling comumente utilizada, os autores conseguiram uma taxa de acerto média de 95.78%, chegando a 96.33% quando executadas 12 iterações no treinamento, para a classificação de imagens utilizando, entre outros datasets, o CIFAR-10. A técnica utilizada chama-se “fractional max-pooling”, onde a proporção entre as dimensões da matriz de entrada da camada de pooling e a matriz resultante do processo de pooling não é necessariamente um número inteiro. Para o experimento, os autores trabalharam com

uma proporção com valores variando entre 1 e 2. Durante o treinamento, foi utilizado “data augmentation” para alcançar taxas de acerto maiores e evitar overfitting. O conceito de data augmentation é explicado nesse trabalho na seção 3.6.

Já em [25], os autores proprõem um método para inicialização dos pesos de cada neurônio da rede neural convolutiva. Chamado de Layer-sequential unit-variance (LSUV) initialization, o método consiste de dois passos, onde no primeiro (a pré-inicialização dos pesos), atribui-se para cada matriz resultante de uma convolução da rede um ruído gaussiano com variância unitária. A taxa de acerto do trabalho foi avaliada com o CIFAR-10, alcançando 94.16%.

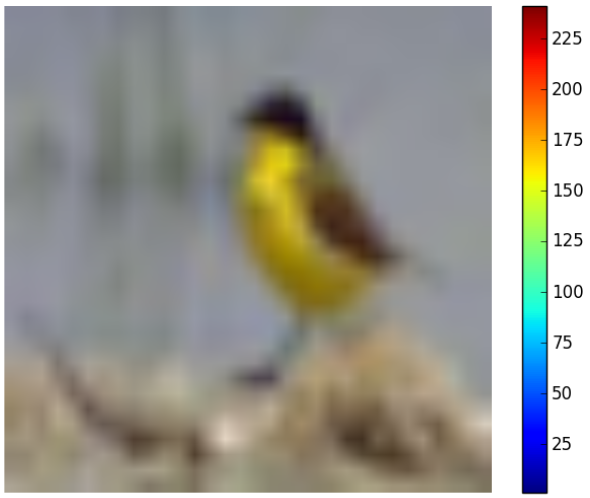


Figura 2.8: Exemplo de imagem da CIFAR-10, classe pássaro.

2.6 TensorFlow

Em conjunto com os avanços em larga escala alcançado pela comunidade acadêmica na área de Aprendizado de Máquina, veio também a criação de novas ferramentas para facilitar a aplicação de algoritmos da área. Entre as ferramentas cujo objetivo é tornar mais prático e acessível o desenvolvimento das primitivas de Aprendizado de Máquina, encontram-se frameworks com bom desempenho até mesmo para sistemas embarcados. Tiny-DNN, por exemplo, é um framework de algoritmos de Deep Learning desenvolvido em C++ com o intuito de obter bom desempenho em sistemas com limitações de recursos, sistemas embarcados e dispositivos de IoT (Internet of Things). Para aplicações de Machine Learning com suporte à GPUs, uma boa recomendação de biblioteca é o CuDNN [4]. Como redes neurais envolvem o cálculo de operações matemáticas com alto custo computacional, a paralelização eficiente para GPU de sua implementação é uma alternativa muito utilizada em ambientes de pesquisa, visando maior agilidade na execução do treinamento. Assim sendo, CuDNN possui otimizações de rotinas comumente utilizadas em Deep Learning, exclusivamente para computadores com placas gráficas da Nvidia. Com implementações de algoritmos considerandos o estado da arte da área de Deep Learning, bem como modelos de referência para utilização destes, Caffe (Convolutional Architecture for Fast Feature Embedding) mostra-se um framework extremamente versátil e aplicável para projetos de pesquisa, softwares empresariais de larga escala e projetos envolvendo visão, fala e multimídia [14]. Grandes corporações utilizam o Caffe em diversas aplicações, como é o caso do Facebook [32] e da Adobe [17].

Para a realização dos experimentos e construção das redes neurais convolucionais, utilizamos a biblioteca de código aberto TensorFlow, criada por pesquisadores e engenheiros do projeto Google Brain [1]. O TensorFlow é considerada uma ferramenta para computação numérica usando grafos de fluxo, ou seja, o TensorFlow utiliza uma representação em forma de grafo para descrever todos os caminhos de execução possíveis, onde cada nó representa uma operação matemática e as arestas representam vetores de dados multidimensionais chamados de tensores [1]. Para exemplificar, a figura 2.9, obtida de [1], demonstra o grafo de fluxo gerado pela computação da função C = ReLU(W * x + b).

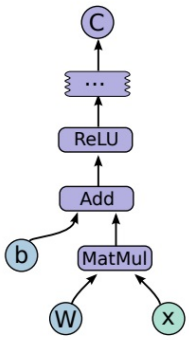


Figura 2.9: Grafo de fluxo da computação da função C = ReLU(W * x + b)

Mesmo antes do desenvolvimento do TensorFlow, a empresa já possuía um histórico promissor de pesquisa em redes neurais [1]. DistBelief foi a primeira geração de sistemas distribuídos e escaláveis de treinamento e inferência do Google. Trabalhos envolvendo treinamento não supervisionado [20], reconhecimento de fala [31,10] e modelos de detecção de objetos [7] foram implementados com sucesso pela equipe de desenvolvimento, entre inúmeros outros. Baseado na experiência com a rede neural DistBelief, o TensorFlow foi desenvolvido. Esta segunda geração de frameworks do Google para implementação e desenvolvimento de sistemas de aprendizado de máquina em larga escala conta com representação em forma de grafo da computação a ser realizada, onde cada nó representa uma operação a ser feita. Cada nó possui zero ou mais entradas, bem como zero ou mais saídas. Ao fluxo de dados entre os nós, dá-se o nome tensor. Cada tensor é transformado na saída de um nó para a entrada de outro nó, ou até mesmo na entrada do próprio nó. Sendo um framework robusto e flexível, o TensorFlow não se limita apenas à utilização em aprendizado de máquina. Pode ser empregado no desenvolvimento de algoritmos nas mais diversas áreas da computação, como robótica, visão computacional, processamento de linguagem natural, reconhecimento de fala, entre outras [7].

A escolha pelo TensorFlow foi pela qualidade do código criado pelo Google e por permitir maior controle sobre alguns tipos de operações e estruturas de código. A biblioteca integra também algumas funções que facilitam o desenvolvimento de aplicações em aprendizado de máquina, como a criação de redes neurais comuns, redes neurais convolucionais entre outras arquiteturas comumente empregadas na área. Outra vantagem é a portabilidade e a possibilidade de utilizar esses algoritmos em dispositivos móveis.

2.7 Arquiteturas utilizadas

Neste trabalho, todas as CNNs utilizadas baseiam-se na arquitetura AlexNet e SqueezeNet descritas mais adiante nesta seção. Ambas as arquiteturas possuem ligação com o dataset

ImageNet, formado por um conjunto de imagens organizado de acordo com a hierarquia do WorldNet. Mesmo sendo um dataset ainda em desenvolvimento, já possui mais de 100.000 classes, cada uma com centenas ou milhares de imagens.

AlexNet é uma Rede Neural Convolucional compreendida por 8 principais camadas, sendo 5 camadas convolucionais e 3 camadas totalmente conectadas [18]. A primeira camada convolucional recebe como entrada uma imagem de dimensões 224x224x3 e são aplicados 96 filtros de dimensões 11x11x3. A segunda camada convolucional, por sua vez, recebe a matriz resultante da primeira camada convolutiva, após processo de pooling. Nesta segunda camada, 256 filtros de dimensões 5x5x48 são aplicados sobre a matriz de entrada. A matriz resultante passa novamente pela processo de pooling. Já a terceira, quarta e quinta camadas convolucionais estão conectadas entre si sem nenhuma camada de pooling intermediária. Na terceira camada, 384 filtros de dimensões 3x3x256 são aplicados. Na quarta camada convolucional da arquitetura, novamente 384 filtros são aplicados, mas com dimensões 3x3x192. A quinta camada convolucional conta com a aplicação de 256 filtros de dimensões 3x3x192. Por fim, as camadas totalmente conectadas contam com 4096 neurônios cada uma. Originalmente concebida para classificação utilizando o dataset ImageNet, a saída da rede AlexNet produz um vetor de 1000 posições. Este vetor contém em cada uma das suas posições a probabilidade de que a imagem pertença à classe representada. A figura 2.10 ilustra o funcionamento da AlexNet. Com uma arquitetura baseada na AlexNet, a CNN participante do campeonato ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) conseguiu uma taxa de erro de apenas 15.3%. Este feito trouxe popularidade para a arquitetura e teve papel fundamental na propagação de Redes Neurais Convolucionais entre os trabalhos acadêmicos.

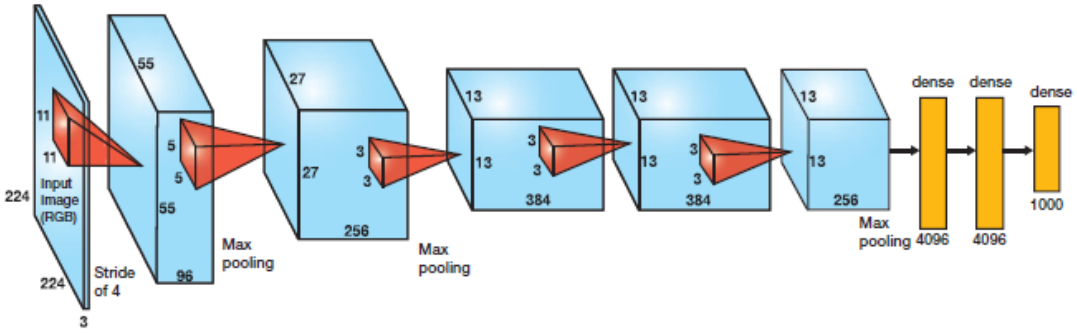


Figura 2.10: Arquitetura básica da AlexNet. Imagem obtida de http://cv-tricks.com.

SqueezeNet também é uma Rede Neural Convolucional, foi desenvolvida com o foco em obter economia de parâmetros de treinamento sem que essa redução causasse um deterioramento da taxa de acerto antes prevista para a arquitetura. Em comparação, a SqueezeNet consegue obter taxas de acerto no dataset ImageNet no mesmo nível que a AlexNet, mas usando 50 vezes menos parâmetros [6]. Essa economia permite que o treinamento seja mais rápido em sistemas distribuídos, precisa de poucas trocas de informação em comparação à outras redes como a AlexNet, permite atualização de parâmetros em sistemas de clientes com maior facilidade além de conseguir carregar essa arquitetura em dispositivos com pouca memória. A SqueezeNet consiste de camadas e módulos "fire", esses módulos por sua vez são formados pela concatenação de outras três camadas: Duas camadas convolucionais em sequência com kernels de tamanho 1x1, seguidas por uma camada convolucional com kernels de tamanho 3x3. A estrutura final, original, proposta pelos autores é ilustrada pela figura 2.11 abaixo.

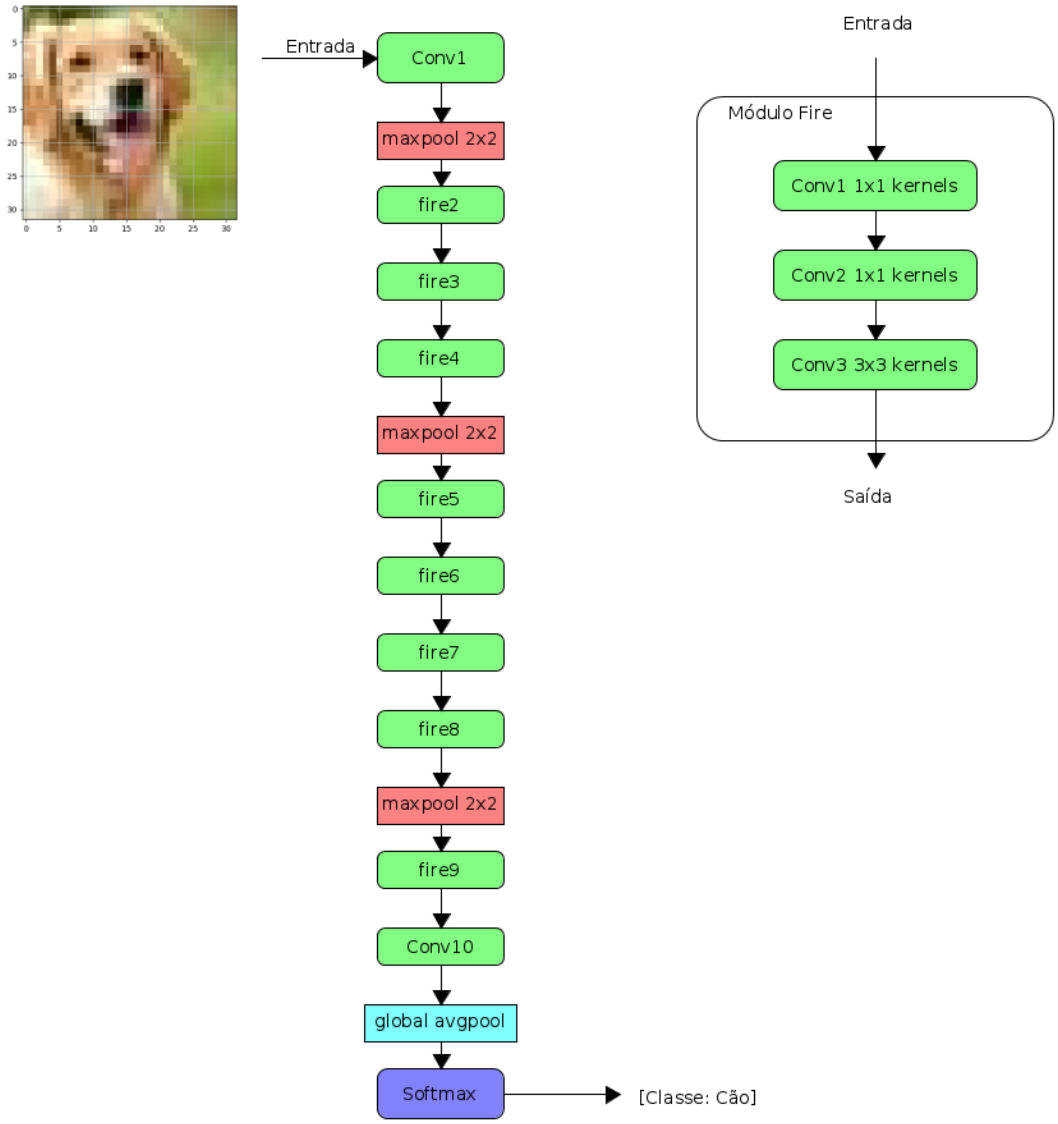


Figura 2.11: Arquitetura básica SqueezeNet.

2.8 Testes de parâmetros

Em algumas das redes usadas nesse trabalho foi feita a inicialização dos pesos de cada camada de maneira alternada camada a camada, usando métodos diferentes de inicialização como o método “Xavier” e “Normal Truncada”. O método Xavier é feito de maneira a manter a escala dos pesos aproximadamente a mesma em todas as camadas, sem que sejam muito discrepantes entre si. Já o método da Normal Truncada gera uma distribuição normal truncada, onde valores maiores que a média em duas vezes o desvio padrão são descartados e gerados novamente. A alteração desses parâmetros foi puramente experimental com o intuito de gerar configurações iniciais dessas redes de maneira única no ensemble.

Durante o treinamento foram utilizados dois tipos de otimizadores, o otimizador “Adam” e o “RMSProp”. No geral o otimizador Adam promove uma convergência mais rápida da rede durante o treinamento. Da mesma forma como a inicialização dos parâmetros, a utilização de dois tipos diferentes de otimizadores foi com a intenção de gerar diferenças entre as redes.

2.9 Ensembles

Nos últimos anos, uma quantidade considerável de artigos demonstrou que o uso do método ensemble agrega um excepcional aprimoramento na taxa de acerto do sistema de classificação como um todo [24]. Um dos exemplos mais notáveis foi o [18], onde Krizhevsky demonstrou no ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 que o seu modelo contendo 5 convnets alcançou uma taxa de erro de apenas 38.1%. Um mérito e tanto, tendo em vista que a taxa de erro mais baixa na época era de 40.7% (utilizando apenas uma rede neural convolutiva para classificação). Semelhantemente, em 2013, Zeiler e Fergus reduziram a taxa de erro de 40.5% para 36.0% utilizando seis convnets para a classificação de imagens. Mesmo com resultados tão promissores até o momento, não se tem uma clara definição da relação entre o ganho de taxa de acerto do sistema classificador e o número de CNNs a serem utilizados [24].

O método consiste em treinar um número arbitrário de classificadores (podendo estes ser de qualquer tipo, desde simples RNA até complexas estruturas convolutivas.) Neste trabalho, foram utilizadas exclusivamente CNNs para compor o sistema de classificação. Todas estas redes neurais, após o treinamento e já na fase de testes, irão classificar cada imagem em uma determinada classe. A seguir, são analisados todos os resultados e, em seguida, determinada a classe à qual a imagem será atribuída. A figura 2.12 demonstra este processo.

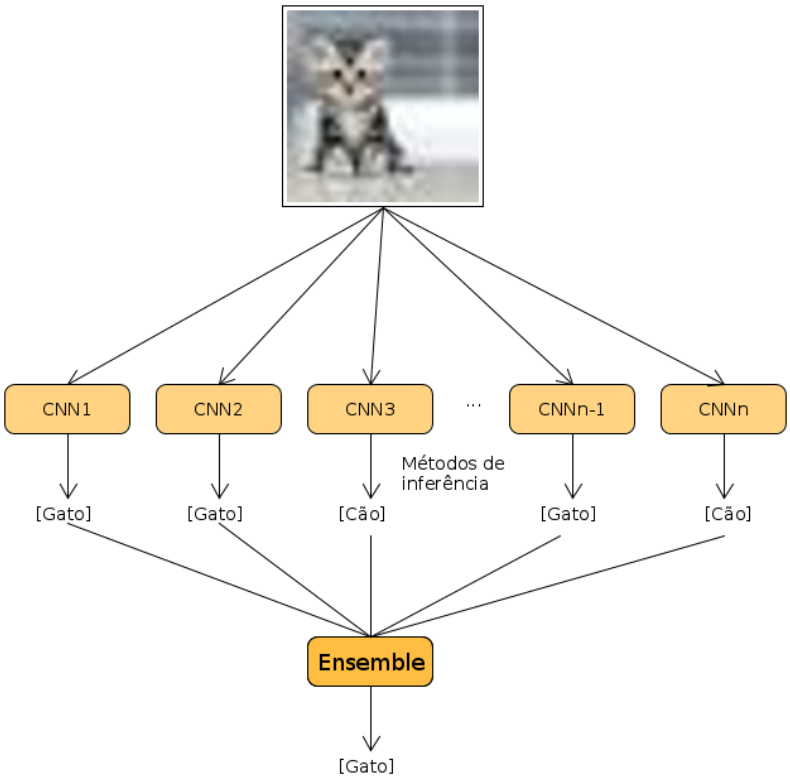


Figura 2.12: Esquema do ensemble.

2.10 Trabalhos relacionados

Em [12] os autores propuseram combinar dois ensembles diferentes de CNNs para classificar imagens nos datasets CIFAR-10 e CIFAR-100. Para cada ensemble utilizaram um dos dois tipos distintos escolhidos de CNN: AlexNet e NIN. Usaram ensembles de 20 CNNs para cada tipo, atingindo uma taxa de acerto na CIFAR-10 de 83.75% no ensemble usando AlexNet e 90.42% usando NIN. Entretanto, o modelo proposto pelos autores, denominado AECNN (Aggregate Ensemble model based on deep CNN) e composto por duas CNNs distintas, atingiu uma taxa de acerto pior, 90.19%, que o ensemble original usando arquitetura NIN.

Outros problemas podem ser solucionados por meio de ensembles de classificadores. Em [19], utilizou-se um ensemble de 3 CNNs distintas (números de camadas variando de 8 a 10) para tratar o problema de classificação de expressões faciais humanas (Facial Expression Recognition problem, *FER*), entretanto as taxas de acerto ficaram em torno de 65.03% na classificação em 7 classes (Raiva, Nojo, Medo, Felicidade, Tristeza, Surpresa, Neutro), o que pode ser considerado baixo apesar de ter sido citado que a taxa de acerto humana nesse mesmo dataset utilizado é de 65% ±5% [19].

Em [26] utilizaram-se ensembles de redes profundas entre outros métodos para classificar ondas cerebrais e prever o estado emocional de uma pessoa usando um eletroencefalograma (EEG).

Já em [2] foi criado um ensemble de CNNs utilizando arquiteturas como AlexNet e GoogLeNet pré-treinadas no dataset da ImageNet (mais de 1 milhão de imagens). Entretanto o ensemble não foi puramente composto por CNNs, utilizou-se também o classificador SVM. Aplicou-se uma técnica chamada “Fine-Tuning” para adaptar as redes convolucionais aos conjuntos de dados e para diminuir a taxa de erro através do uso do algoritmo backpropagation. Posteriormente esse ensemble foi utilizado para classificação de imagens médicas e mostrou-se que em alguns casos o ensemble melhorou a classificação geral.

Capítulo 3

Desenvolvimento

3.1 Proposta do trabalho

A proposta do trabalho surgiu de forma intuitiva pela equipe durante o desenvolvimento dos testes usando redes neurais convolucionais. Apesar de não ser uma ideia inovadora, é interessante notar que a necessidade perante as dificuldades levaram naturalmente à solução apresentada, a criação de ensembles com essas redes classificadoras. Esse trabalho, então, propõe-se a demonstrar as vantagens de tal método e como ele pode melhorar os resultados de classificação mesmo em cenários desafiadores.

3.2 Ambientes de testes

Os testes foram executados usando uma máquina do laboratório de informática da Universidade Federal do Paraná com 16 CPUs, cada qual com clock de 2.401.000 MHz, e 70.8 GB de memória. Não foi utilizado nenhum tipo de GPU durante os treinamentos das CNNs.

3.3 Desenvolvimento

O trabalho teve início com a tentativa de obter-se e treinar uma CNN com arquitetura eficiente de forma a atingir taxas de acerto superiores a 86%. Esses testes iniciais usaram várias CNNs baseadas na arquitetura AlexNet e usaram apenas as 50.000 imagens do conjunto de dados para treinamento. Observou-se que muito tempo estava sendo gasto no treinamento e os resultados não pareciam animadores, girando em torno de 70% de acerto no conjunto de testes. Dessa forma surgiu a ideia de reaproveitar todas as CNNs já treinadas (e o tempo gasto para treiná-las) para tentar obter-se taxas de acerto mais altas.

A metodologia experimental utilizada possui as seguintes etapas: construção de arquiteturas diversas de CNNs, treinamento dessas redes, teste individual de taxa de acerto, teste de taxa de acerto do ensemble para três técnicas diferentes que serão explicadas na sessão 3.5. O processo foi repetido diversas vezes de maneira incremental, adicionando novas CNNs ao ensemble.

Cada arquitetura diferente processa os dados de maneira particular dadas as quantidades de camadas na rede, o tipo e a ordem delas. Somado a isso, os parâmetros são inicializados de maneira aleatória de forma que cada CNN com estrutura diferente aprende informações a respeito do conjunto de dados de maneira única atrelada à arquitetura e a forma com que os parâmetros foram inicializados. Dessa forma algumas CNNs aprendem mais sobre determinadas partes do

conjunto de dados do que outras, acertando mais na classificação em algumas classes e errando mais em outras. Dessa maneira, se existir uma grande quantidade de CNNs diferentes com conhecimento diverso a respeito do conjunto de dados e forem unidas em uma espécie de conselho com um sistema de votação, existe uma probabilidade maior de que se a maioria do conselho votar numa mesma classe a chance de ser realmente essa classe é alta. Por meio de experimentos esse fato foi comprovado, estes consistiram na criação de um ensemble experimental de CNNs usando apenas a votação pela maioria e os resultados mostraram que o método era melhor do que qualquer CNN sozinha.

Após a comprovação da melhoria na taxa de acerto usando o ensemble de CNNs, a maioria das redes treinadas posteriormente foram baseadas na arquitetura SqueezeNet por obterem boa taxa de taxa de acerto e sendo bem mais rápidas e leves para treinar.

3.4 Diversidade de topologias

Foi observado que ao adicionar novas CNNs no ensemble, pouco impacto tiveram na taxa de acerto se possuíam estruturas muito parecidas com as outras já presentes. Mas sempre que uma CNN bem diferente era adicionada ao ensemble, os resultados eram expressivamente maiores. E os maiores ganhos na taxa de acerto foram com redes de topologias diversas, mudando a forma, a ordem e a quantidade de camadas. Isso ocorre pois essa diferença topológica causa uma maior variância das respostas de cada rede em relação às outras, fazendo com que sejam mais abrangentes. Tal abrangência quando computada em conjunto com muitas outras redes, formaliza o conhecimento a respeito dos dados de maneira mais concreta e portanto aumenta a taxa de acerto.

Já o uso de redes com uma mesma arquitetura, mas utilizando alterações de parâmetros diversos não trouxeram consigo grandes impactos no resultado final do ensemble sugerindo que a topologia realmente é importante e tem grande impacto.

3.5 Mecanismo de votação

O mecanismo de voto funciona da seguinte maneira: O primeiro indivíduo, CNN1, é carregado em memória utilizando o checkpoint gravado previamente durante o treinamento da rede. A seguir, todas as imagens do conjunto de teste são carregadas e alimentadas na rede que está votando, sempre na mesma ordem. A rede então retorna sua resposta em forma de um vetor de votos que será armazenado.

Conforme este trabalho foi evoluindo, utilizamos alguns outros métodos diferentes para computar o vetor de votos final do ensemble. Os métodos usados nesse trabalho são: Votação simples, das médias, Std-Médias com Fibonacci, onde o último é experimental. A seguir serão descritos um a um. Para o método da votação simples um vetor de saída com todas as respostas, classificações das classes (0 a 9), é retornado e armazenado após a votação de cada CNN. Todos os outros indivíduos seguem o mesmo padrão até que no final seja obtido um conjunto de votos para todas CNNs da seguinte forma:

Votos = ([3 8 8 ..., 5 1 7], [3 8 8 ..., 5 1 7], [3 1 8 ..., 5 4 7], [3 8 8 ..., 5 3 7], [3 8 8 ..., 5 1 7], [3 8 8 ..., 5 1 7], [5 8 1 ..., 5 1 7], [3 8 8 ..., 5 1 7], [3 8 8 ..., 5 1 7], [5 8 8 ..., 5 1 7], [3 8 0 ..., 5 1 7], [3 8 8 ..., 5 1 7], [3 8 8 ..., 5 1 7], [5 8 8 ..., 5 1 7], [3 8 8 ..., 5 1 7])

Sendo que $|Votos[i]| = 10.000 \; \forall \; i \in \{0..14\}$.

A moda estatística é então aplicada nesses vetores e ao término da operação é obtido um vetor final de votos cujos valores é a moda dos valores de todas as CNNs para cada índice (decisão da maioria). Esse vetor então é comparado ao vetor de rótulos das imagens do conjunto de teste e a taxa de acerto é computada.

Para o método das médias, diferente do método da votação simples as respostas são adquiridas num estágio anterior na rede que consiste em um vetor de vetores. Esse vetor contém 10.000 vetores com 10 valores cada em ponto flutuante que indicam a chance de cada classe de 0 a 9 ser a correta. Antes de computar o valor máximo dentre esses valores e transformar esses números na classe estimada, é feita a média desses valores e só então é computado o valor máximo que retorna o vetor de votos com valores de 0 a 9 indicando as classes. Já no método Std-Médias com Fibonnaci, antes de calcular o valor máximo, é calculado o desvio padrão de todos os valores e os resultados são multiplicados por 0.618 e em seguida esses resultados são somados à média dos dados multiplicados a 1.618 e só então é computado o valor máximo. Os números foram escolhidos de forma arbitraria e o nome advém da sequência de Fibonnacci, que para valores grandes dividindo-se um valor pelo seu posterior na série obtém-se a proporção áurea 0.618, mas não é estritamente necessário utilizar esses valores. Observou-se que o Std-Médias pode conseguir valores maiores de taxa de acerto que o método das médias se os parâmetros (0.618 e 1.618) estiverem ajustados à distribuição dos dados, indicando que o resultado pode ser maximizado. A fórmula final para o método Std-Médias é:

$vetorFinal = DesvioPadrao(ensembleDados) * 0.618 + Media(ensembleDados) * 1.618$, onde ensembleDados são os valores contidos na CNN.

Essa fórmula será maximizada em trabalhos futuros. Esses métodos, no final, assim como a votação simples retornam um vetor único com a classificação das 10.000 imagens do conjunto de teste e é usado para computar a taxa de acerto final de cada método.

3.6 Data Augmentation

Com o sistema de votação funcionando e com a maioria das CNNs atingindo taxas de acerto em torno de 70%, foi utilizada a técnica chamada de aumento de dados (Data Augmentation) que consiste em aplicar nas imagens originais algum tipo de transformação. No caso desse trabalho foi aplicado o espelhamento em torno do eixo vertical da imagem de forma a obter-se ao final duas cópias de cada imagem, sendo uma o espelhamento vertical da outra. Assim o conjunto de treinamento foi duplicado e obteve-se 100.000 imagens ao final do processo.

Com o uso dessa técnica as taxas de acerto subiram para valores em torno de 80% além de reduzir o overfitting da rede nos dados. Essa técnica é muito comum e realmente melhora os resultados como também foi comentado em [2].

Várias CNNs, baseadas na AlexNet e com taxas de acerto menor que 70%, foram sendo abandonadas durante o processo para que as que tivessem os melhores resultados assumissem as posições no ensemble. Esse processo mostrou um aumento consistente e progressivo na taxa de acerto do ensemble. O critério utilizado para a eliminação das redes foi que essas tivessem taxa de acerto inferior à 70%, e posteriormente esse valor foi sendo ajustado progressivamente.

Capítulo 4

Resultados

4.1 Estrutura do ensemble

Desenvolveu-se um algoritmo para administrar o ensemble de CNNs e computar os votos usando algumas técnicas. O ensemble idealmente deve ser composto por um número ímpar de indivíduos (CNNs) para evitar empates. As redes foram inseridas no ensemble seguindo o critério de possuir taxa de acerto superior à 70%. Conforme os treinamentos levaram algumas redes a atingir taxas de acerto superiores a 80%, esse critério foi sendo alterado ao longo do tempo. No estágio final dos experimentos, a quantidade de redes no ensemble chegou a 17 CNNs com arquiteturas diferentes.

O ensemble assemelha-se à estrutura ilustrada na imagem 2.12, em que uma sequência de imagens é fornecida às CNNs do ensemble. Cada CNN produz um resultado de classificação e no final todos esses resultados são processados em conjunto para produzir o resultado final.

O símbolo "X" nas colunas "D. Aumentados" e "P. Alternadas" na Tabela 4.1 indica que as respectivas redes usaram dados aumentados e inicialização de parâmetros alternados durante o treinamento.

A Tabela 4.2 mostra os resultados finais do ensemble usando as três técnicas diferentes, onde as colunas taxa de acerto média (T.A. Média) é a média das taxas de acerto de todas as redes, AVG-Gain é o ganho do ensemble em relação à taxa de acerto média, Max-Gain é o ganho do ensemble em relação à melhor CNN do conjunto, Min-Gain é o ganho do ensemble em relação à pior CNN e finalmente a taxa de acerto final (T.A. Final) mostra o valor obtido pela votação do ensemble para cada técnica.

De acordo com os valores obtidos na tabela 4.2, é possível notar que o ensemble teve um resultado 4.14% superior que a melhor CNN do ensemble na votação simples, 4.64% nas médias e 4.98% nas std-médias. Esse resultado corrobora nossa hipótese inicial de que a união dos conhecimentos das diversas redes a respeito do dataset gera um conhecimento menos fragmentado e mais acurado a respeito do todo. Claramente o ganho do ensemble é muito melhor em relação à média geral e do acerto da pior CNN do conjunto. No geral o ensemble ganha em robustez também.

Tabela 4.1: Estrutura do ensemble e características

CNN ID	**Arquitetura**	**D. Aumentados**	**Camadas**	**Taxa de acerto**
CNN 1	SqueezeNet	X	19	79.23%
CNN 2	SqueezeNet	X	16	81.06%
CNN 3	SqueezeNet	X	32	78.59%
CNN 4	AlexNet	–	19	74.52%
CNN 5	SqueezeNet	–	16	74.84%
CNN 6	SqueezeNet	X	15	81.27%
CNN 7	SqueezeNet	X	22	78.58%
CNN 8	SqueezeNet	X	16	80.04%
CNN 9	SqueezeNet	X	16	80.39%
CNN 10	SqueezeNet	X	16	81.10%
CNN 11	AlexNet	X	12	72.72%
CNN 12	AlexNet	X	12	71.49%
CNN 13	SqueezeNet	X	16	81.18%
CNN 14	SqueezeNet	X	29	**82.33**%
CNN 15	SqueezeNet	X	15	79.84%
CNN 16	SqueezeNet	X	29	81.95%
CNN 17	SqueezeNet	X	29	81.61%

CNN ID	**T. Steps**	**Otimizador**	**P. Alternados**	**Taxa de acerto**
CNN 1	136.700	RMSProp	–	79.23%
CNN 2	58.600	RMSProp	–	81.06%
CNN 3	76.000	Adam	X	78.59%
CNN 4	8.000	RMSProp	–	74.52%
CNN 5	27.500	RMSProp	–	74.84%
CNN 6	75.600	RMSProp	–	81.27%
CNN 7	111.100	Adam	–	78.58%
CNN 8	58.800	RMSProp	–	80.04%
CNN 9	75.200	RMSProp	–	80.39%
CNN 10	55.500	Adam	X	81.10%
CNN 11	69.400	Adam	–	72.72%
CNN 12	11.800	Adam	–	71.49%
CNN 13	51.600	Adam	X	81.18%
CNN 14	75.900	Adam	X	**82.33**%
CNN 15	58.400	RMSProp	–	79.84%
CNN 16	100.500	Adam	–	81.95%
CNN 17	100.800	Adam	–	81.61%

Tabela 4.2: Resultado final e análises complementares

Método do Ensemble	**T.A. Média**	**AVG-Gain**	**Max-Gain**	**Min-Gain**	**T.A. Final**
Votação Simples	78.87%	7.60%	4.14%	14.98%	**86.47**%
Médias	78.87%	8.10%	4.64%	15.48%	**86.97**%
Std-Médias	78.87%	8.44%	4.98%	15.82%	**87.31**%

Capítulo 5

Conclusão

Observamos que o método de ensemble de redes neurais convolucionais realmente melhora a capacidade de classificação da tecnologia atual das CNNs, adicionando a essas um funcionamento mais robusto em relação à variação dos dados. Os ensembles devem ser explorados, pois existem diversas limitações que o modelo atual de classificação por CNN traz consigo, como a limitação da quantidade de dados rotulados, a incapacidade de dividir determinadas classes por causa da dimensionalidade dos dados e a limitação de tempo para treinar uma rede ao extremo de sua capacidade, onde o ganho de acurácia torna-se um processo muito custoso.

Além disso notou-se um aumento consistente na taxa de acerto do ensemble a cada nova CNN adicionada. Isso leva a acreditar que se forem utilizadas redes com alta taxa de acerto (próximas de 96%), pode-se obter taxas de acerto próximas a 99% ou o máximo que for possível dada a dimensionalidade dos dados para então ser utilizado em aplicações que precisam de um maior nível de confiança.

Para aplicações que precisem de tempo de resposta rápido, pode-se criar grafos simultâneos usando TensorFlow e carregar todos os checkpoints, em memória, de todo o ensemble, isso garantiria a agilidade na hora de processar novas informações vinda de uma câmera por exemplo. Esse trabalho mostrou ser possível atingir melhores taxas de acerto usando ensembles de CNNs com arquiteturas diferentes do que usando apenas uma arquitetura específica.

Outro fator importante observado durante os testes foi que a forma com que os dados são tratados tem um grande impacto nos resultados, o simples fato de adicionar dados aumentados ao dataset original acarretou um ganho de quase 10% na taxa de acerto das redes que usaram essa técnica. Isso sugere que técnicas de pré-processamento das imagens e outras abordagens que visem o aumento do volume de dados é um bom ponto de partida para desenvolver maneiras inovadoras de melhorar o desempenho dessas redes.

Capítulo 6

Trabalhos Futuros

Como trabalho futuro, pretende-se continuar a explorar novos tipos de ensembles com CNNs de arquiteturas diferentes e melhores.

Outro passo natural também é continuar treinando todas as redes de forma que elas atinjam o máximo da taxa de acerto que seja possível para cada uma, o que não foi feito por limitações de tempo.

Novas formas de calcular os votos do ensemble também devem ser exploradas. Nesse trabalho foram utilizados os métodos de votação simples, das médias, do Std-Médias com Fibonacci, mas existem muitas outras formas de computar esses valores. Observou-se que o Std-Médias com Fibonacci varia seus valores de acordo com a distribuição dos dados, assumindo valores maiores que o método das médias, valores menores e em algumas vezes valores iguais. Isso indica que é possível derivar uma fórmula que capture o comportamento dos dados e encontre os valores dos parâmetros que maximizem os resultados. Isso será realizado em próximos trabalhos.

Será abordado também uma análise de variância entre as CNNs de forma a compreender quais CNNs são essenciais para uma melhor taxa de acerto do ensemble e quais podem ser abandonadas, também saber a quantidade ótima de CNNs no conjunto. Se quanto mais redes melhor o resultado, ou se existe um valor ótimo para a quantidade de redes dada a configuração das CNNs integrantes do sistema.

Outra ideia interessante é criar um algoritmo para descobrir quais classes são as mais difíceis de classificar para todas as redes (classes com maiores taxas de erro). Provavelmente serão classes que são muito similares entre si e de difícil distinção. Uma solução para este problema, embora precise ser avaliada, é com o aumento da quantidade de dados nessas classes específicas ou com a criação de sub-redes especialistas nessas classes.

Referências Bibliográficas

[1] 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, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, 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 available from tensorflow.org.

[2] IEEE Jinman Kim Member-IEEE David Lyndon-Michael Fulham Ashnil Kumar, Member and IEEE Dagan Feng, Fellow. *An Ensemble of Fine-Tuned Convolutional Neural Networks for Medical Image Classification*. IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, VOL. 21, NO. 1, JANUARY 2017, 2017.

[3] Frédéric Bimbot, Christophe Cerisara, Cécile Fougeron, Guillaume Gravier, Lori Lamel, François Pellegrino, and Pascal Perrier. *Exploring convolutional neural network structures and optimization techniques for speech recognition*. ISCA, 2013.

[4] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *CoRR*, abs/1410.0759, 2014.

[5] P. N. Druzhkov and V. D. Kustikova. A survey of deep learning methods and software tools for image classification and object detection. *Pattern Recognition and Image Analysis*, 26(1):9–15, Jan 2016.

[6] Matthew W. Moskewicz Khalid Ashraf-William J. Dally Kurt Keutzer Forrest N. Iandola, Song Han. *S QUEEZE N ET : A LEX N ET - LEVEL ACCURACY WITH 50 X FEWER PARAMETERS AND <0.5MB MODEL SIZE*. ICLR 2017, 2016.

[7] Andrea Frome, Greg Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In *Neural Information Processing Systems (NIPS)*, 2013.

[8] Benjamin Graham. Fractional max-pooling. *CoRR*, abs/1412.6071, 2014.

[9] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, and Gang Wang. Recent advances in convolutional neural networks. *CoRR*, abs/1512.07108, 2015.

[10] Georg Heigold, Vincent Vanhoucke, Andrew Senior, Patrick Nguyen, Marc'aurelio Ranzato, Matthieu Devin, and Jeff Dean. Multilingual acoustic models using distributed deep neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, CA, 2013.

[11] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*, 2012.

[12] Chien-Hao Kuo Yu-Chi Wu Narisa N.Y. Chu Pao-Chi Chang Hsin-Kai Huang, Chien-Fang Chiu. *Mixture of Deep CNN-based Ensemble Model for Image Retrieval*. IEEE 5th Global Conference on Consumer Electronics, 2016.

[13] Anastasia Ioannidou, Elisavet Chatzilari, Spiros Nikolopoulos, and Ioannis Kompatsiaris. Deep learning advances in computer vision with 3d data: A survey. *ACM Comput. Surv.*, 50(2):20:1–20:38, April 2017.

[14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 675–678, New York, NY, USA, 2014. ACM.

[15] Jason Kuen Lianyang Ma Amir Shahroudy-Bing Shuai Ting Liu Xingxing Wang-Li Wang Gang Wang Jianfei Cai Tsuhan Chen Jiuxiang Gu, Zhenhua Wang. *Recent Advances in Convolutional Neural Networks*. `https://arxiv.org/pdf/1512.07108.pdf`, 2017.

[16] Milton Kampel, José Stech, Evlyn Novo, Leão de Moraes Novo, Leila Fonseca, and Caixa Postal. O algoritmo support vector machines (svm): avaliação da separação ótima de classes em imagens ccd-cbers-2. 12 2017.

[17] Sergey Karayev, Aaron Hertzmann, Holger Winnemoeller, Aseem Agarwala, and Trevor Darrell. Recognizing image style. *CoRR*, abs/1311.3715, 2013.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[19] Zhigeng Pan Kuang Liu, Minming Zhang. *Facial Expression Recognition with CNN Ensemble*. International Conference on Cyberworlds, 2016.

[20] Quoc Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg Corrado, Jeff Dean, and Andrew Ng. Building high-level features using large scale unsupervised learning. In *International Conference in Machine Learning*, 2012.

[21] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 396–404. Morgan-Kaufmann, 1990.

[22] Geert J. S. Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A. W. M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *CoRR*, abs/1702.05747, 2017.

[23] D. Lu and Q. Weng. A survey of image classification methods and techniques for improving classification performance. *International Journal of Remote Sensing*, 28(5):823–870, 2007.

[24] Debapriya Maji, Anirban Santara, Pabitra Mitra, and Debdoot Sheet. Ensemble of deep convolutional neural networks for learning to detect retinal vessels in fundus images. *CoRR*, abs/1603.04833, 2016.

[25] Dmytro Mishkin and Jiri Matas. All you need is a good init. *CoRR*, abs/1511.06422, 2015.

[26] RUOYU DU RAJA MAJID MEHMOOD and HYO JONG LEE. *Optimal Feature Selection and Deep Learning Ensembles Method for Emotion Recognition From Human Brain EEG Sensors*. IEEE Access, 2017.

[27] Samyam Rajbhandari, Yuxiong He, Olatunji Ruwase, Michael Carbin, and Trishul Chilimbi. Optimizing cnns on multicores for scalability, performance and goodput. *SIGOPS Oper. Syst. Rev.*, 51(2):267–280, April 2017.

[28] Shan Suthaharan. *Support Vector Machine*, pages 207–235. Springer US, Boston, MA, 2016.

[29] J.S.Denker D.Henderson R.E.Howard W.Hubbard and L.D.Jackel Y.LeCun, B. Boser. *Handwritten Digit Recognition with a Back-Propagation Network*. `https://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf`, 1989.

[30] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

[31] M.D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q.V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G.E. Hinton. On rectified linear units for speech processing. In *38th International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, 2013.

[32] Ning Zhang, Manohar Paluri, Marc'Aurelio Ranzato, Trevor Darrell, and Lubomir D. Bourdev. PANDA: pose aligned networks for deep attribute modeling. *CoRR*, abs/1311.5591, 2013.